IBM Cognos Data Manager
Version 10.2.0

User Guide

Note

Before using this information and the product it supports, read the information in “Notices” on page 515.

Product Information

This document applies to IBM Cognos Business Intelligence Version 10.2.0 and may also apply to subsequent releases. To check for newer versions of this document, visit the IBM Cognos Information Centers (http://publib.boulder.ibm.com/infocenter/cogic/v1r0m0/index.jsp).

Contents

Chapter 10. Surrogate Keys . . . 59

Chapter 11. Reference Structures . . . 63

Chapter 12. Constructing and Transforming Hierarchies . . . 95

Chapter 13. Fact Builds . . . 105

Chapter 14. DataStreams . . . 115

Chapter 19. Managing Builds Using JobStreams . . . 227

Chapter 20. Publishing Data Movement Tasks . . . 243

Chapter 21. Executing Builds and JobStreams . . . 245

Introduction

This document is intended for use with IBM® Cognos® Data Manager.

If you are using Cognos Data Manager for the first time, we recommend that you first read *IBM Cognos Data Manager Getting Started* and perform the lessons that it contains.

The examples in this document use BackusNaur Form (BNF) to describe the syntax of the Data Manager language. The BNF syntax consists of the following:

- A set of terminal symbols that are the words, commands, or punctuation of the language and command-line interface.
- A set of non-terminal symbols that are essentially place holders. They appear in angled brackets < >, for example, <refdata_file>. A definition of each non-terminal symbol may appear elsewhere in the syntax definition. However, not all non-terminal symbols are defined. For a complete definition of the scripting language, including the description of all non-terminal symbols, see the *IBM Cognos Function and Scripting Reference Guide*.
- A set of rules that you apply when interpreting the BNF definitions:
  - Colon colon equal sign (::=) means 'is defined to be'.
  - Square brackets [ ] indicate that the enclosed symbols are optional.
  - Braces { } indicate that the enclosed symbols may be repeated zero or more times.
  - The pipe symbol | indicates that you should choose only one of the items that it separates.

The following example defines the <options> symbol to be an optional -C, followed by a <var_list> symbol. It then defines the <var_list> symbol to be zero, one, or more instances of -V followed by a <name>=<value> pair:

```
<options> ::=
[-C] <var_list>

<var_list> ::=
{-V<name>=<value>
```

Audience

You should be familiar with Microsoft Windows operating system and SQL. You should also have an understanding of multi-dimensional data analysis or Business Intelligence.

Finding information

To find IBM Cognos product documentation on the web, including all translated documentation, access one of the IBM Cognos Information Centers (http://publib.boulder.ibm.com/infocenter/cogic/v1r0m0/index.jsp). Release Notes are published directly to Information Centers, and include links to the latest technotes and APARs.

You can also read PDF versions of the product release notes and installation guides directly from IBM Cognos product disks.

Forward-looking statements

This documentation describes the current functionality of the product. References to items that are not currently available may be included. No implication of any future availability should be inferred. Any such references are not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The development, release, and timing of features or functionality remain at the sole discretion of IBM.

Accessibility features

IBM Cognos Data Manager does not currently support accessibility features that help users with a physical disability, such as restricted mobility or limited vision, to use this product.

IBM Cognos HTML documentation has accessibility features. Adobe Acrobat PDF documents are supplemental and, as such, include no added accessibility features.

Samples disclaimer

The Great Outdoors Company, GO Sales, any variation of the Great Outdoors name, and Planning Sample depict fictitious business operations with sample data used to develop sample applications for IBM and IBM customers. These fictitious records include sample data for sales transactions, product distribution, finance, and human resources. Any resemblance to actual names, addresses, contact numbers, or transaction values is coincidental. Other sample files may contain fictional data manually or machine generated, factual data compiled from academic or public sources, or data used with permission of the copyright holder, for use as sample data to develop sample applications. Product names referenced may be the trademarks of their respective owners. Unauthorized duplication is prohibited.

Chapter 1. What's New?

This section contains a list of new and changed features for this release.

For information about new features for this release, see the *IBM Cognos Business Intelligence New Features Guide*.

What's New information for past releases, including versions 8.3 and 8.4, is available by accessing documentation within the IBM Cognos Business Intelligence 10.1.1 Information Center (http://publib.boulder.ibm.com/infocenter/cbi/v10r1m1/index.jsp?lang=en).

To review an up-to-date list of environments supported by IBM Cognos products, such as operating systems, patches, browsers, web servers, directory servers, database servers, and application servers, visit the IBM Cognos Information Centers (http://publib.boulder.ibm.com/infocenter/cogic/v1r0m0/index.jsp).

New Features in Version 10.2.0

Listed below are the new features since the last release. Links to directly-related topics are included.

64-bit execution mode for builds and JobStreams

If you have installed the 64-bit hybrid version of IBM Cognos Data Manager, you can now execute builds and Jobstreams in 32-bit and 64-bit mode, if the operating system and database drivers you are using are also compatible with 64-bit.

Note: If you are using IBM Cognos Data Manager Designer, for example to test database queries, you must also have the 32–bit drivers installed.

For more information, see "Execute builds and JobStreams in 64-bit mode" on page 253.

Load balancing capabilities available via the SAP Message Server

When setting up an SAP R/3 connection in IBM Cognos Data Manager, you can now use the load balancing capabilities available via the SAP Message Server when there is a cluster of SAP Application Servers.

This feature is additional to the current capability of selecting a preset server to perform data extraction. You can continue to use this current functionality or, if you want to leverage load balancing in your SAP environment, you can now specify data extraction requests to be submitted via the SAP Message Server.

Netezza performance improvements

IBM Cognos Data Manager 10.2.0 provides significant performance improvements when loading data into a Netezza® database. No changes are required to builds or JobStreams because these improvements are achieved through internal processing.

Using stored procedure result sets in a data source

IBM Cognos Data Manager 10.2.0 introduces better support for using result sets from stored procedures in data source queries.

New Features in Version 10.2.0

Listed below are the new features since the last release. Links to directly-related topics are included.

Configuring a JobStream Execution to Use IBM Cognos Business Intelligence Load Balancing

It is now possible to utilise IBM Cognos Business Intelligence load balancing for executing fact build and dimension build nodes in a JobStream by choosing to execute these nodes as individual data movement tasks.

For more information, see “Configure a JobStream Execution to Use IBM Cognos Business Intelligence Load Balancing” on page 254.

Changed Features in Version 10.1.0

Listed below are changes to features since the last release. Links to directly-related topics are included.

TM1 Turbo Integrator Delivery Module

The following enhancements have been made to the TM1® turbo integrator delivery module:

- It is now possible to deliver data directly to IBM Cognos TM1, instead of using staged files.
- When delivering data, you can truncate an existing TM1 cube by resetting its values to zero, and specifying which view of a TM1 cube to truncate (if required).
- It is possible to set a variable to ignore minor errors while delivering data.

For more information, see “TM1 Turbo Integrator Delivery Module” on page 451, and “DM_TM1_LOAD_SUPPRESS_ERROR_CODES” on page 295.

Chapter 2. IBM Cognos Data Manager

The main purpose of IBM Cognos Data Manager is to create data warehouses and data repositories for reporting, analysis, and performance management.

Data Manager does this by
- extracting operational data from multiple sources
- transforming and merging the data to facilitate enterprise-wide reporting and analysis
- delivering the transformed data to coordinated data marts

Data Manager can be used to transfer data into single database tables and, where more complex transformations are required, to populate data warehouse fact tables and dimension tables.

Data Manager integrates with other IBM Cognos Business Intelligence products by delivering metadata to IBM Cognos Framework Manager. This allows target data warehouse and data repositories to be modeled and used in IBM Cognos Business Intelligence and Performance Management projects.

Building the Data Foundation for the IBM Cognos Business Intelligence Solution

IBM Cognos Data Manager can deliver data to any supported target database into any design of database schema.

However, Data Manager is best suited to delivering data into dimensional data warehouses and coordinated data marts. This approach has several advantages over non-dimensional data warehouses and uncoordinated data marts.

Dimensional data warehouses are typically made up of a series of coordinated data marts. The key to coordinated data marts is that common dimensions are shared across subject areas. This offers a common view of your business, while allowing flexibility and rapid incremental deployment. This approach is well suited to business intelligence because it allows a fast and iterative development approach to help the successful delivery of projects.

Data Manager supports this development approach by providing many of the tools and functionality required for the delivery of subject orientated data warehouses. The Data Manager dimensional framework allows the shared dimensions, known as conformed dimensions, to be constructed and then shared across all related subject areas. This reuse of the dimensional framework helps ensure that each subject area can be designed incrementally, but integrated consistently with any existing subject areas within the data warehouse.

You can choose to build enough dimensions and fact tables to complete one or more subject areas and then deploy the model to IBM Cognos Framework Manager for further modeling. When the model has been published, it can be used for reporting and analysis. Data Manager allows you to build on the model you have created and extend it to additional areas while maintaining maximum reuse of any completed work.

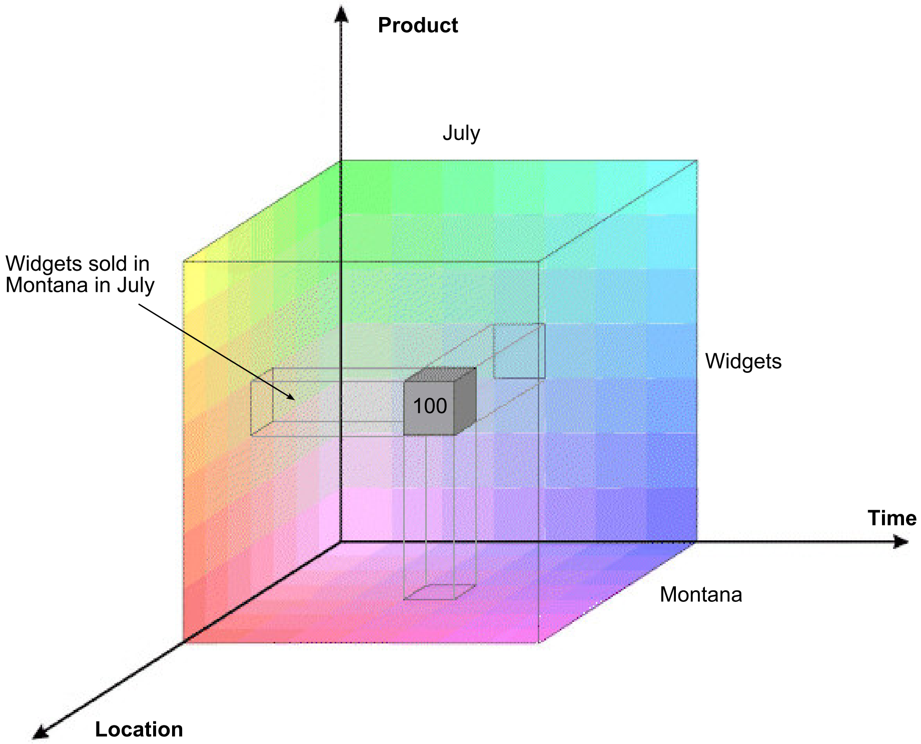
Data Manager provides all of the techniques required to easily build a scalable data foundation for your Business Intelligence project.

Business Dimensions

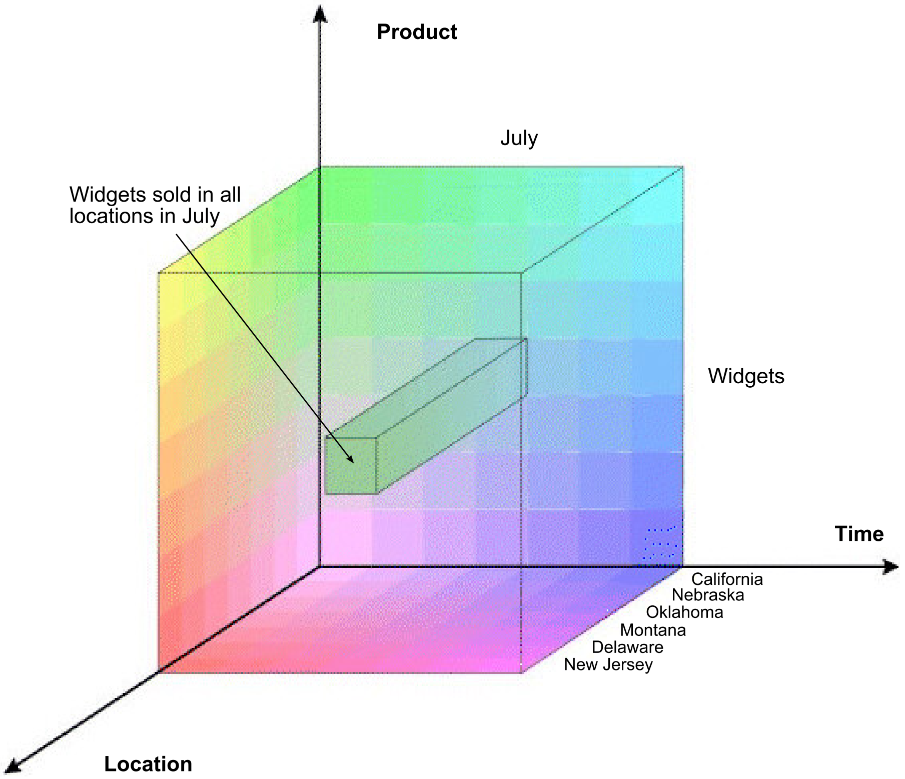
Business dimensions are the core components or categories of a business, anything that you want to analyze in reports.

Business dimensions are fundamental to data organization and provide context for numeric data items, or measures.

For example, a retail chain-store may categorize its sales data by the products that it sells, by its retail outlets, and by fiscal periods. This organization has the business dimensions Product, Location, and Time. The measures of the business, such as how much it sells, lie at the intersection of these dimensions. The following illustration shows these dimensions as the axis of a three-dimensional space. The cube at the center of this space represents 100 units of widgets sold in Montana during July.



You can derive summary information by aggregating data along one or more dimensions. For example, the following illustration shows the aggregation of data along the Location dimension to give the total sales of widgets during July.



To enable aggregation, business dimensions must have a hierarchy structure.

Conformed Data Warehouses

Each set of business dimensions and fact tables can be referred to as a subject oriented data mart because it can be used to address the reporting and analysis requirements for a single subject area of the business.

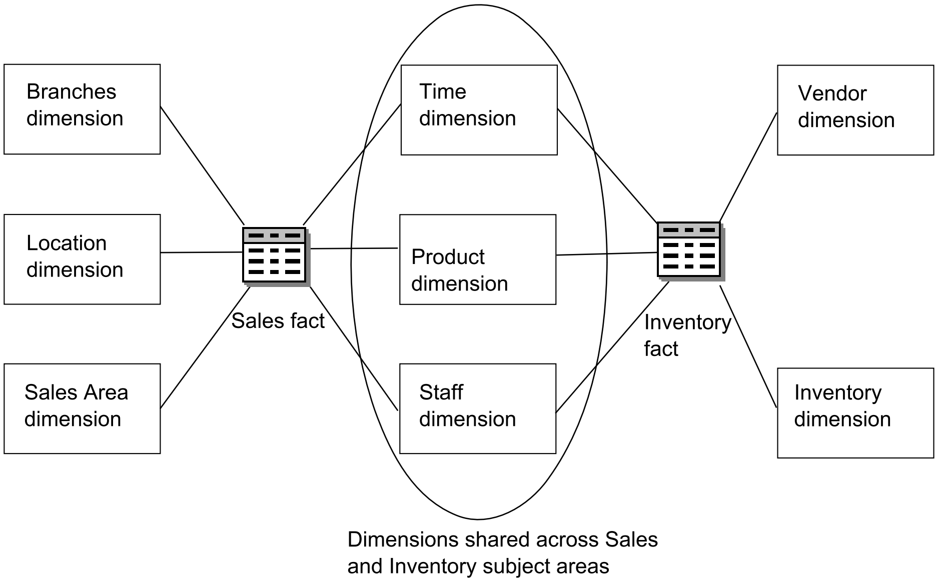
This approach allows for data structures that are very well suited to analysis by business users and are fast to report against and analyze.

By combining multiple subject areas using shared common dimensions it becomes possible to navigate across multiple subject areas while maintaining a consistent view of the business. This combination of multiple dimensional subject areas is known as a dimensional data warehouse.

The key is to coordinate data marts by sharing common dimensions across many subject areas. This strategy offers a common view of your business, which provides flexibility and allows rapid, incremental deployment.

Coordinated data marts rely on a shared dimensional structure to model the key aspects of your business. The key aspects of your business may be, for example, your products and customers, and the time periods over which you account for sales. Because all data marts share this structure, each dimension provides a single view of the business aspect that it represents. Such dimensions are known as conformed dimensions and provide the basis for integration of dependent data marts.

This structure forms the framework on which you build separate, but coordinated, data marts. As each new data mart comes online, it integrates consistently with the existing data marts. Eventually, enough data marts exist to provide an integrated, corporate data warehouse. The IBM Cognos Data Manager dimensional framework is key to ensuring that all subject dimensional areas are built using a common dimensional structure and that maximum reuse is made of shared dimensions. This allows for rapid, but incremental development of the dimensional data warehouse.



Scalable Dimension Architecture

IBM Cognos Data Manager is an extremely flexible and responsive data transformation tool.

It uses an innovative dimensional reference model to coordinate the management of data marts of all shapes and sizes and across many different platforms. In addition, Data Manager can perform massive-scale data merging and aggregation in a fraction of the time possible with hand-made solutions or with SQL-based data warehouse loaders.

IBM Cognos Data Manager Environments

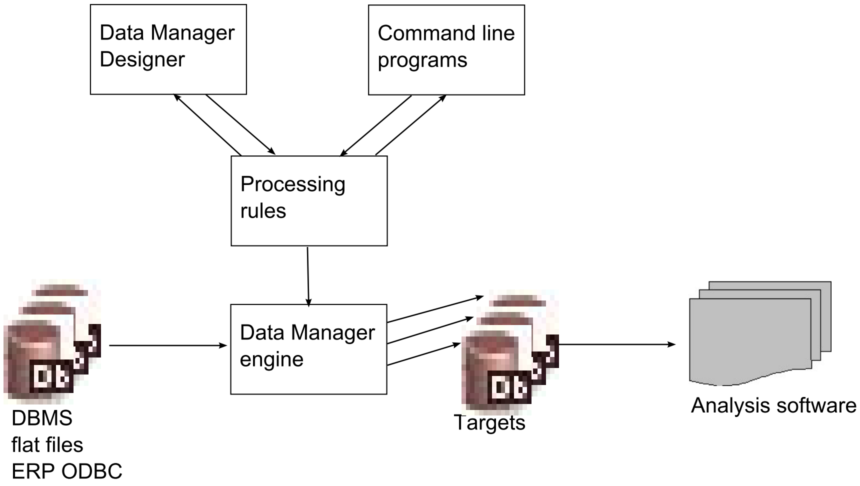
IBM Cognos Data Manager has two environments: Data Manager Designer and the Data Manager engine, or run-time environment.

IBM Cognos Data Manager Designer

In IBM Cognos Data Manager Designer, you use builds to specify a set of processing rules that determine how Data Manager acquires data from the source databases, transforms the data, and delivers it to the target database.

This information is stored in a Data Manager catalog. You then execute the builds using the Data Manager engine that can be either on UNIX or Microsoft Windows operating systems.

The following illustration shows a model of the Data Manager architecture.



To build a data warehouse or data mart using Data Manager, you typically

- create a Data Manager catalog
- set up database connections
- create reference dimensions and templates
- define the reference structures (that is, the hierarchies, auto-level hierarchies, and lookups)
- create the required fact builds and dimension builds
- add and define the fact deliveries and the dimension deliveries
- execute builds
- create a JobStream if required to coordinate the builds with other procedures related to managing the data mart

The IBM Cognos Data Manager Engine

The IBM Cognos Data Manager engine consists of a number of programs that you can invoke either from Data Manager Designer (with Microsoft Windows operating system) or directly on the command line (with Windows and UNIX operating systems).

For most applications, IBM Cognos recommends that you design and prototype using Data Manager Designer on a Windows computer. You can then deploy your builds to either a Windows or a UNIX computer that has the Data Manager engine installed. However, you can also program the Data Manager engine directly in the Data Manager language.

IBM Cognos Data Manager Language

The IBM Cognos Data Manager language is declarative; that is, you specify what must be rather than what Data Manager must do.

A Data Manager project controls the acquisition, transformation, and delivery of data. Data Manager lets you store projects

- in text files
- in a catalog that resides in a database
- in packages

Text-based Projects

In a text-file specification, a project consists of up to three text files, with each text file controlling some aspects of the overall process:

- database alias definition file
- reference structure definition file
- build definition file

You can use each definition file for more than one project. For example, you can define one alias definition file and one reference structure definition file that you use with all your projects.

Catalog-based Projects

In a catalog-based specification, the project information is stored in special tables in a database.

To edit the contents of a catalog, you must either use IBM Cognos Data Manager Designer or export the definition to a temporary text file, which you edit, and then import the modified definition to the catalog.

Like text-based projects, each catalog-based project consists of a set of aliases, reference structures, and a build. Each performs the same function as it does in a text-based project.

Packaged-based Projects

In a package-based specification, the components in the package are saved in a non-editable.pkg file.

You can use a package to move components between catalogs, partially backup and restore a catalog, combine the efforts of several developers, and distribute catalogs or components.

From the command line interface, you can execute fact builds, dimension builds, and JobStreams from within a package rather than from a catalog. This means that the IBM Cognos Data Manager engine can operate in an embedded environment without using a Data Manager catalog.

Application Limits

The following table gives limits on IBM Cognos Data Manager features.

Where a value is given as unlimited, available memory, disk space, or processor specification may impose a practical limit.

Feature	Limit
Maximum levels in a hierarchy.	Unlimited.
Maximum attributes in a reference structure or template.	Unlimited.
Maximum transformation model elements in a fact build.	Unlimited. However, the maximum number of elements that Data Manager can deliver to a target table is limited by the target DBMS.

Feature	Limit
Maximum columns in a query.	Limited by the source or target DBMS.
Maximum deliveries in a fact build.	Unlimited.
Maximum size of temporary files.	Limited by the operating system or file system.

Multi-platform

For information on multi-platform see,

http://www.ibm.com/software/data/support/cognos_crc.html.

Chapter 3. Getting Started with IBM Cognos Data Manager Designer

The IBM Cognos Data Manager Designer window consists of the menu bar, the toolbar, the Tree pane, and the Visualization pane.

The toolbar provides quick access to the main features of Data Manager. For information about the toolbar buttons, see “Toolbar Buttons” on page 497.

The Tree Pane

The Tree pane is on the left of IBM Cognos Data Manager Designer window.

At the top level, it has folders named Builds and JobStreams, Metadata, and Library.

At the bottom of the Tree pane there are a number of tabs:

Icon	Tab name	Description
🟥	Catalog	Shows the full catalog
⚙	Fact Builds	Shows only the fact builds contained in the catalog
⚙	Dimension Builds	Shows only the dimension builds contained in the catalog
⚙	JobStreams	Shows only the JobStreams contained in the catalog
📚	Library	Shows only the dimensions, connections, and functions contained in the catalog
▣	Metadata	Shows only the metadata dimensions and collections contained in the catalog

The Builds and JobStreams Folder

This folder contains the fact builds, dimension builds and JobStreams in the current catalog.

For more information, see Chapter 13, “Fact Builds,” on page 105, Chapter 18, “Dimension Builds,” on page 203, and Chapter 19, “Managing Builds Using JobStreams,” on page 227.

It can also contain user-defined folders that you set up. For more information see “Set Up User-Defined Folders” on page 12.

The Metadata Folder

This folder contains the metadata dimensions and collections of star schemas that represent the conformed model that you are building.

For more information, see Chapter 22, “Exporting Metadata,” on page 267.

The Library Folder

This folder contains all the connections, reference dimensions (including hierarchies, auto-level hierarchies, lookups, and templates), and user-defined functions in the current catalog.

For more information, see Chapter 7, “Connections,” on page 39, Chapter 8, “Reference Dimensions,” on page 49, Chapter 9, “Templates,” on page 51, Chapter 11, “Reference Structures,” on page 63, and Chapter 23, “User-Defined Functions,” on page 277.

Set Up User-Defined Folders

You can group together facts builds, dimension builds, and JobStreams into user-defined folders. The advantage of using folders is the improved usability of catalogs that contain a large number of builds or JobStreams.

It is also possible to set up a sub folder within a folder, up to a maximum of 32.

Note: User-defined folders are shown on the catalog tab only.

Procedure

1. Click the **Builds and JobStreams** folder.
2. From the **Insert** menu, click **Folder**.
3. In the **Name** box, type a unique name for the folder. If you require, type a business name and description.
4. Click **OK**.

Create a User-Defined Folder

This topic describes how to create a user-defined folder.

Procedure

1. Click the **Builds and JobStreams** folder.
2. From the **Insert** menu, click **Folder**.
3. In the **Name** box, type a unique name for the folder. If you require, type a business name and description.
4. Click **OK**.

Create a Sub Folder

This topic describes how to create a sub folder.

Procedure

1. Click the required user-defined folder.
2. From the **Insert** menu, click **Folder**.

   The **Folder Properties** dialog box appears.
3. In the **Name** box, type a unique name for the folder. If you require, type a business name and description.
4. Click **OK**.

   IBM Cognos Data Manager adds a sub folder to the user-defined folder.

Move Builds and JobStreams Into a Folder

This topic describes how to move builds and JobStreams in a user-defined folder.

Procedure

1. Click the required build or JobStream.
2. From the **Actions** menu, click **Move to Folder**.

   The **Move to Folder** dialog box appears.
3. Select the folder to which you want to add the build or JobStream.
   - To create a folder rather than use an existing one, click **New**, and then type a unique name for the folder.
   - To move a build or JobStream from a user-defined folder to the Builds and JobStreams folder, click the catalog node.
4. Click **OK**.
5.

Results

You can move a build, JobStream, or folder by dragging and dropping it into the required folder. You can select multiple items to move into a single folder by holding down the Shift key or the Ctrl key as appropriate as you make your selections.

Delete a User-Defined Folder

You can delete user-defined folders that you no longer require.

Note: When you delete a user-defined folder, any builds or JobStreams that it contains are also deleted. Ensure you move all required builds and JobStreams out of a folder before deleting it.

Procedure

1. Click the required user-defined folder.
2. From the **Edit** menu, click **Delete**.

   If the folder is not empty, a message appears prompting you to confirm whether you want to delete the selected folder.
3. Click **Yes**.

Change the Default Folder Settings

By default, when a catalog is opened, the top level folders are shown open on the Catalog tab. However, you can change this.

Procedure

1. From the **Tools** menu, click **Options**.
2. Click the **General** tab.
3. Clear the **Expand root folders when catalog is opened** check box.
4. Click **OK**.

Select Multiple Components

You can select more than one component in the Tree pane. This is useful if you want to move more than one component, perhaps to a user-defined folder.

Procedure

Hold down the Shift key or the Ctrl key as appropriate, and then click the required components.

The Visualization Pane

The Visualization pane shows information about the selected fact build, dimension build, reference structure, metadata export, or JobStream.

You can right-click an item in the Visualization pane to show a menu that allows you to perform many of the actions that are also available from the Tree pane or the menu bar. For example, you can

- add data sources to a fact build or reference structure
- change the element type in the transformation model
- add levels to a hierarchy
- add delivery modules to a delivery
- export metadata
- execute DataStreams

For more information, see “View Fact Builds in the Visualization Pane” on page 105, “Viewing Dimension Builds in the Visualization Pane” on page 204, “View Reference Structures in the Visualization Pane” on page 64, “View JobStreams in the Visualization Pane” on page 230, or “View Metadata in the Visualization Pane” on page 267.

Change the Default Settings in the Visualization Pane

You can change the default settings in the Visualization Pane.

Procedure

1. From the **Tools** menu, click **Options**.
2. Click the **General** tab.
3. Enter values as required
   - **Show transformation model elements in the visualization by default**.
   - **Maximum number of elements to show in the visualization**

     Note: Limiting the number of elements shown in the **Visualization**pane, may mean that mapping connections on other visualization tabs cannot be shown.

     By default, IBM Cognos Data Manager shows a maximum of 100 elements.
   - **Show build details in the visualization by default**.

     Note: This tab also allows you to change the **JobStream Visualization**pane default settings. For information, see “View JobStreams in the Visualization Pane” on page 230.
4. Click **OK**.

Change the Size of the Information in the Visualization Pane

By default, IBM Cognos Data Manager shows information in the Visualization pane at 100%, but you can change this if you require.

Procedure

From the **View** menu, click **Zoom** and then click the required percentage.
Tip: You can also double-click the Visualization pane to enlarge the objects so that they fill the available space.

Reveal or Hide the Visualization Pane

By default, the Visualization pane shows information about the object selected in the Tree pane. However, you can hide this information if you require.

Procedure

From the **View** menu, click **Show Visualization** .

Copy the Visualization Pane to the Clipboard

You can copy the Visualization Pane to the clipboard.

Procedure

From the **View** menu, click **Copy to Clipboard** .

Set Up Window and Dialog Box Preferences

You can set up your preferences for the following:

- whether you want the Welcome dialog box to appear when you start IBM Cognos Data Manager
- whether you want Data Manager to remember the size and position of windows and dialog boxes when you have changed them

Procedure

1. From the **Tools** menu, click **Options**.
2. Click the **General** tab.
3. Select or clear the following check boxes as required
   - **Show the welcome screen when the application starts**
   - **Remember window sizes and positions**
4. Click **OK**.

Locate Components and Their Dependencies

Sometimes, in a large catalog, it can be difficult to locate a component. IBM Cognos Data Manager can help you to

- locate a component
- locate the dependencies of a component

Components are connections, builds, reference dimensions, reference structures, JobStreams, templates, functions, metadata dimensions, and metadata collections.

Locate a Component

You can search for specific components in a catalog.

Procedure

1. From the **View** menu, click **Navigate** .
2. Click the **Find Components** tab.
3. In the **Search for the text** box, type the name of the component to find.

   Tip: You can include the asterisk (*) and question mark (?) wildcard characters to assist you.

   Note: You cannot search for user-defined folders.
4. Click **Find Now**.

   IBM Cognos Data Manager lists all the components in the current catalog that match your search text, together with their component type and location.
5. To locate a component in the **Tree** pane and to update the **Visualization** pane, click the component name in the list.

Results

You can cancel a search at any time by clicking **Stop**.

You can copy the search results to the Microsoft Windows clipboard by clicking **Copy to Clipboard**.

View and Locate the Dependencies of a Component

You can view the dependencies of a specific component.

Procedure

1. In the **Tree** pane, click the component.
2. From the **View** menu, click **Navigate** .

   The **Navigator** dialog box appears showing all the dependencies for the selected component, together with their component type.
3. Click the **Navigate** tab.

   When you click a component in the navigator, it is located in the **Tree** pane, and the navigator is refreshed to show the dependencies of the currently selected component. The **Visualization** pane is also refreshed to show the selected component.
   - If a component has dependent user-defined functions, the functions are not shown.
   - You cannot view or locate user-defined folders.

   Tip: To move between previously selected components, click the left arrow button or the right arrow button as appropriate.
4. If you want to freeze the contents of the navigator at any point, so that only the **Tree** pane and **Visualization** pane are refreshed when you click a component in the navigator, select the **Freeze** check box.

   Tip: To copy the contents of the navigator to the Microsoft Windows clipboard, click **Copy to Clipboard**.

Create a Similar Component

If you want to create a component that is similar to an existing component, duplicate the component and then make the required changes. This is often quicker and simpler than creating a completely new component.

Procedure

1. Click the component to duplicate.
2. From the **Edit** menu, click **Duplicate**.

   If the catalog has unsaved changes, a message appears prompting you either to save the changes and proceed or to cancel.
3. Click **Save First, Then Duplicate**.

   IBM Cognos Data Manager duplicates the component and assigns a unique name to it by adding a number to the original name. For example, the duplicate of FISCAL would be FISCAL:1.

Case Sensitivity

Names of objects in IBM Cognos Data Manager Designer are not case sensitive. For example, Data Manager considers VendorDrillThru, VENDORDRILLTHRU, and vendordrillthru to be identical.

However, column names used in Cognos SQL queries are case sensitive and must exactly match the database column names. For example, a column named Productid is considered to be different from a column named PRODUCTID or productID.

Native SQL queries are subject to the case rules of the DBMS or external function library.

Cognos SQL

When constructing SQL statements in IBM Cognos Data Manager, you can specify whether you want to use native SQL or Cognos SQL for the database you are accessing. By default, a Data Manager connection accepts any vendor-specific SQL SELECT statement in a data source, including non standard SQL extensions and hints.

Using Cognos SQL provides a greater degree of portability between mixed database environments because a common dialect can be used. Cognos SQL is an extension of SQL99.

When using Cognos SQL to create a data source query, you can use the "call" syntax to execute a stored procedure that generates a result set.

Note: You cannot use Cognos SQL for SQLTXT connections or Published FM Package connections.

Change Fonts

You can change the fonts that are used in IBM Cognos Data Manager. Normally, Data Manager uses a default set of fonts according to the locale settings of your computer. However, if you are working with data that contains characters from different locales, you may want to change the font so that the characters are shown correctly.

Procedure

1. From the **Tools** menu, click **Options**.
2. Click the **Fonts** tab.
3. Click the appropriate browse button ... for the setting that you want to change.
   - **Application Font** applies to the **Tree** pane, the tabs on the **Visualization** pane, and all window and dialog boxes
   - **Expression Font** applies to any expression that you enter, perhaps for a derivation element or user-defined function
   - **Visualization Font** applies to the **Visualization** pane

   The **Select Font** dialog box appears.
4. From the drop down lists, select the appropriate font.

   Tip: If you are changing expression fonts, you can restrict the list of fonts to fixed width fonts only by selecting the **Fixed width only** check box.
5. Click **OK** to close the **Select Font** dialog box.

   Tip: Click **Reset** to return to the default values.
6. Click **OK**.

Quoted Identifiers for Multibyte Character Sets

By default, identifiers are not enclosed in quotation marks in IBM Cognos Data Manager Designer. If, in the Data Manager scripting language, you want to use identifiers that contain characters that may be parsed as reserved words, then they must be enclosed in quotation marks.

You can specify that identifiers are to be automatically enclosed in quotation marks when they are dragged and dropped within the scripting windows.

Procedure

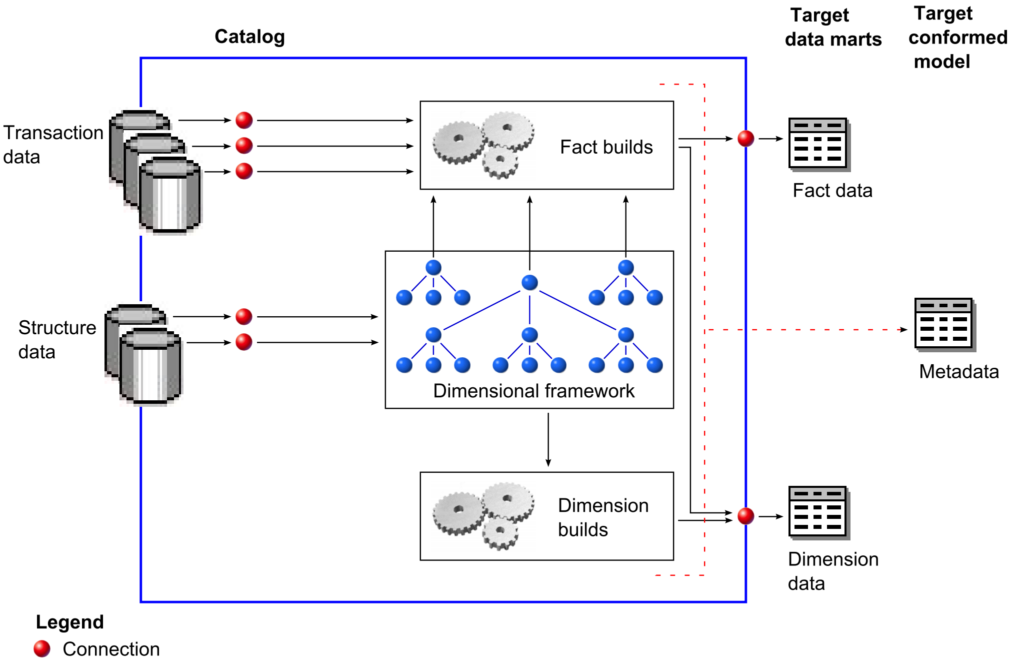
1. From the **Tools** menu, click **Options**.
2. Click the **General** tab.
3. Select the **Quote identifiers** check box.
4. Click **OK**.

Chapter 4. IBM Cognos Data Manager Catalogs

Each IBM Cognos Data Manager catalog contains the components that define how Data Manager extracts, transforms, and delivers data.

This illustration shows the top-level objects within a Data Manager catalog. It also illustrates

- the flow of data between objects, from the data sources, and to the target data marts
- how Data Manager creates conformed models for a data mart that can be exported as metadata



A Data Manager catalog is stored in a database and uses Database Management System (DBMS) drivers to connect to the required databases. For more information, see Appendix B, "Database Drivers," on page 465.

Every Data Manager catalog consists of a number of tables. Each schema within a database server can contain only one Data Manager catalog. Therefore, multiple catalogs must be assigned to distinct schemas. For more information, see "Create an IBM Cognos Data Manager Catalog Schema" on page 22.

Although multiple instances of Data Manager can access a catalog concurrently, we do not recommend that multiple users perform updates concurrently. To detect concurrent updates, Data Manager applies data integrity checks.

For information about a collaborative environment with multiple developers working on the same project concurrently, see Chapter 26, "Multi-Developer Support," on page 315.

Create an IBM Cognos Data Manager Catalog

Only one IBM Cognos Data Manager catalog can be open at any time. If you attempt to create or open a catalog when another catalog is already open, a message appears indicating that the open catalog will be closed.

Procedure

1. Create a data source where the Cognos Data Manager catalog is to reside. The method that you use is specific to the chosen DBMS.

   For information, see the documentation of the relevant software vendor. If you are creating a database using ODBC, see "Open the ODBC Data Source Administrator" on page 46.
2. From the **File** menu, click **New Catalog** .

   Tip: You can also create a catalog from the **Welcome** dialog box.

   The **New Catalog** wizard appears.
3. Type a name for the new catalog and, if you require, a business name and description.
4. Click **Next**.
5. In the left pane, click the connection method to connect to the database that is to hold the catalog.

   Cognos Data Manager updates the dialog box to show fields that are appropriate for the connection method that you chose.
6. Enter the values required by the connection method.
7. Click **Test Connection**.

   If the connection is successful, a message appears indicating this.
8. Click **Finish**.

   If the database that you chose already contains a catalog, an error message appears.

   Cognos Data Manager automatically creates the tables that it requires in the database. However, you may prefer to create these manually. For more information, see "Create an IBM Cognos Data Manager Catalog Schema" on page 22.

Open an IBM Cognos Data Manager Catalog

If you attempt to open a catalog that was created using IBM Cognos DecisionStream Series 7 or earlier, IBM Cognos Data Manager prompts you to upgrade the catalog or to create an empty catalog using the same connection. If you choose to upgrade the catalog, the Upgrade Catalog wizard appears.

For more information see "Perform an Upgrade" on page 25.

Cognos Data Manager allows only one catalog to be open at any time. If you attempt to open a catalog when another catalog is already open, a message appears indicating that the open catalog will be closed.

Procedure

1. From the **File** menu, click **Open Catalog**

.

2. Click the **Existing** tab.

   Tip: Click the **Recent** tab to select one of the last 20 catalogs that you opened. You can clear the recent catalogs list by clicking **Options** from the **Tools** menu, and then clicking **Clear recent catalogs list**.
3. In the left pane, click the connection method to connect to the database that contains the catalog.

   Cognos Data Manager updates the dialog box to show fields that are appropriate for the connection method that you chose.
4. Enter the values required by the connection method.
5. Click **OK**.

   If the target database does not contain catalog tables, a message appears giving you the option to create them.

Change the Properties of a Catalog

You can change the properties of a catalog.

Procedure

1. In the **Tree** pane, click the catalog.
2. From the **Edit** menu, click **Properties**

   .
3. Change the properties as you require.
4. Click **OK**.

Document an IBM Cognos Data Manager Catalog

IBM Cognos Data Manager can document the contents of a catalog in HTML format and open it in your web browser. You can then view and print it.

Procedure

1. From the **File** menu, click **Document Catalog**.
2. In the **Document name** box, type the full path and file name of the file that you want to create.

   You must give the file name an extension so that Data Manager can start a Web browser in which you can view the resulting documentation. If you type a file extension other than .htm or .html, you must register it within your Web browser.
3. In the **Document type** box, click the type of document that you want to create:
   - Full Documentation provides full component details with SQL statements and also includes information about all configured delivery modules.
   - Usage Summary provides a list of all the catalog components.
4. In the **Master document** box, you may want to enter the name of a master document to use as a template for the generated documentation.

   Note: The master document must have HTML tags within it and an <INSERT> tag to mark the point of insertion for the generated document. For example, a simple master document would contain the text "<HTML><INSERT></HTML>".

5. Click **OK**.

   Your Web browser appears showing the generated documentation.

Results

Note: Facts builds are referred to as data builds in the generated document.

Create an IBM Cognos Data Manager Catalog Schema

When you create a catalog, IBM Cognos Data Manager automatically sets up the tables that it requires. However, for some installations, you may prefer to manually set up these tables.

For example, you may want to assign database-specific syntax, such as the containers in which tables are placed within the database.

A Data Manager catalog consists of three sets of tables:

- Run detail tables named dsb_component_run, dsb_run_context, dsb_jobnode_run, dsb_sequence, and dsb_delivery_hist.

  For information on these tables, see Appendix C, "Audit Tables," on page 479.
- Audit message tables named dsb_audit_trail, dsb_audit_msg, and dsb_audit_msg_line.

  For information on these tables, see Appendix C, "Audit Tables," on page 479.
- Catalog tables named dsb_catalog, dsb_component, and dsb_component_line. These tables are used internally by Data Manager.

Data Manager allows you to remove a previous version of a catalog. To do this, from the **Schema** box, click **DecisionStream Series 7 Catalog (V4)**, and then click **Drop**.

To remove run details or audit history from a catalog, from the **Schema** box, select **Data Manager Catalog (V5) Run Details Only)** or **Data Manager Catalog (V5) Audit Messages Only)**, and then click **Drop**.

Procedure

1. From the **Tools** menu, click **Manage Database Schema**.
2. In the **Schema** box, click **Data Manager Catalog (V5)**.
3. In the **Database** box, click the connection to which Data Manager should apply the schema.

   Note: You can only select connections defined in the current catalog.
4. Clear the **Auto apply** check box for Data Manager to actually create the schema.
5. Use the buttons at the bottom of the window to add the SQL statements for the function that you want to perform. You can view and edit the statements on the **SQL** tab.

Button	SQL statement
Create	Creates the catalog tables. You can copy these statements, using the Microsoft Windows clipboard, to other applications to inform your database administrators of the tables to create.

Button	SQL statement
Index	Creates indexes for the selected database tables.
Grant	Grants all permissions to all users for all tables in the schema. However, this may not apply to all databases.
Delete	Deletes the contents of all the catalog tables.
Drop	Drops all catalog tables removing the catalog from the database.

This populates the **SQL** tab with the correct statement and, if you selected the **Auto apply** check box, applies the statement.

6. Click **Save** if you want to save the SQL script, perhaps to use when manually creating the tables.

   The **Save SQL** dialog box appears where you can specify the path and name of the file to create.
7. Click the **Apply** button to apply the SQL statement, as it appears on the **SQL** tab, to the connection and to create the tables.

   The results of applying the SQL statement are shown on the **Log** tab.
8. Click **Close**.

Close a Catalog

IBM Cognos Data Manager allows only one catalog to be open at any time. If you attempt to open or create a catalog when another catalog is already open, a message appears indicating that the open catalog will be closed.

Procedure

From the **File** menu, click **Close Catalog**.

Chapter 5. Upgrading a Catalog

IBM Cognos Data Manager introduces new features. Because of this, catalogs created using IBM Cognos DecisionStream Series 7 must be upgraded before you can use them with Data Manager.

Note: If your catalogs were created using DecisionStream 6.5 or earlier, you must upgrade them to DecisionStream Series 7 first.

When you upgrade a catalog, you have the option to retain the Series 7 catalog tables so that you can continue to use the catalog in DecisionStream Series 7. This is possible because Data Manager uses different catalog tables to those used in DecisionStream Series 7.

Data Manager and DecisionStream Series 7 or earlier can exist concurrently on the same computer.

Upgrading a catalog automatically

- creates catalog tables
- updates the script for each component
- creates a series of new components

In addition to upgrading a catalog, you can also

- restore a catalog backup created in DecisionStream Series 7 into a Data⌂Manager catalog
- import a package created in DecisionStream Series 7 into Data Manager

Notes

- CATIMP files containing DecisionStream Series 7 components are upgraded and converted on import.
- External user-defined functions created in DecisionStream Series 7 will continue to work in Data Manager.
- When you have upgraded a catalog, any subsequent changes to the Series 7 catalog can be brought into Data Manager by importing a component package. For information, see “Import a Component Package” on page 322.
- For information about performing an upgrade from the command line, see “catupgrade” on page 393.

Perform an Upgrade

Before you upgrade a catalog to IBM Cognos Data Manager, ensure that

- all the computers on which the catalog may be used have Data Manager installed on them
- if the Data Manager engine is running on a server with UNIX or another Microsoft Windows operating system, these are upgraded at the same time as the computers running Data Manager Designer
- you list all existing CATEXP or manually created Data Manager script files, and all SQLTXT definition files

- you review all command scripts that may presume that only one version of Data Manager is installed or that reference the old location
- you verify which scripts may need revising that are run through scheduled services such as cron/at
- you have created a backup (either a database backup or a Data Manager backup) of the catalog

  For information, see "Backup and Restore Catalogs" on page 327.

Additional checks are required for catalogs stored as source code control projects because Data Manager upgrades your personal copy of each catalog, not the master catalogs. Ensure that you have

- checked in each catalog to which changes have been made
- synchronized each catalog to ensure that they reflect the master catalogs in the source code control repository

Procedure

1. From the **File** menu, click **Upgrade Catalog**.

   If a catalog is open, a message appears indicating that this action will disconnect the current catalog.
2. In the left pane of the **Upgrade Catalog** wizard, click the connection method to connect to the database that contains the catalog to upgrade.

   Data Manager updates the dialog box to show fields that are appropriate for the connection method that you chose.
3. Enter the values required by the connection method, and then click **Next**.
4. If the selected catalog is stored as a source code control project, a message appears asking whether you want to check in any catalog components before you continue.

   Note: You must use IBM Cognos DecisionStream Series 7 to check in the catalog. You may want to click **Cancel**at this stage, and begin the upgrade again when you have checked in the catalog.

   When you are ready to continue, select the **Continue the upgrade** check box, and then click **Next**.
5. If you want Data Manager to migrate the catalog history by copying all the audit trails currently held in the Series 7 catalog tables into the upgraded catalog, select the **Migrate catalog history** check box.
6. If you want Data Manager to retain the DecisionStream Series 7 catalog tables after the upgrade, ensure that the **Delete prior version** check box is clear.

   If you delete the earlier version of the catalog, Data Manager deletes the DecisionStream Series 7 catalog tables and creates Data Manager catalog tables. You cannot access the catalog in DecisionStream Series 7 after upgrading the catalog.

   If you do not delete the earlier version, Data Manager retains the DecisionStream Series 7 catalog tables, as well as creating Data Manager catalog tables. You can continue to access the catalog in DecisionStream Series 7 after upgrading the catalog.
7. If you want to keep a backup copy of the catalog, select the **Backup prior version** check box, and then type a new name for the existing catalog in the **Backup file name** box.
8. Click **Finish**.

Upgrade Failure

In some situations, an IBM Cognos Data Manager upgrade may fail leaving the Data Manager project in an inconsistent state. This may happen for a number of reasons, including lack of disk space or if the database system that contains the catalog cannot apply the required changes. It is therefore very important that you create a backup of the previous environment before you begin to upgrade.

If an upgrade fails, you can restore the previous catalog database environment by using the restore functionality for your database.

Note: If you retained the IBM Cognos DecisionStream Series 7 catalog tables when you performed the initial upgrade, you can run the upgrade process again without following the steps below. Data Manager uses the DecisionStream Series 7 catalog tables to perform the upgrade, and overwrites any existing Data Manager catalog tables.

Procedure

1. Using your database's utilities, or the Data Manager **Database Schema** window, drop the Data Manager catalog tables.

   For information on using the **Database Schema** window, see “Create an IBM Cognos Data Manager Catalog Schema” on page 22.

   These catalog tables are named dsb_audit_msg, dsb_audit_msg_line, dsb_audit_trail, dsb_catalog, dsb_component, dsb_component_line, dsb_component_run, dsb_delivery_hist, dsb_jobnode_run, dsb_run_context, and dsb_sequence.
2. Use DecisionStream Series 7 to recreate the old database tables.

   These catalog tables are named ds_audit_msg, ds_audit_msg_line, ds_audit_trail, ds_catalog, ds_component, ds_component_line, ds_component_run, ds_delivery_hist, ds_jobnode_run, and ds_sequence.

   You can do this using the **Database Schema** window. For information, see “Create an IBM Cognos Data Manager Catalog Schema” on page 22.
3. From within DecisionStream Series 7, restore the catalog.
4. When the problem is resolved, perform the upgrade again.

Restore From an IBM Cognos DecisionStream Series 7 Backup

If you create a catalog in IBM Cognos Data Manager, or if you have followed the upgrade process, you can restore a catalog backup from IBM Cognos DecisionStream Series 7. In this case, Data Manager updates the script for the catalog objects and creates components as necessary.

Import an IBM Cognos DecisionStream Series 7 Package

If you create a catalog in IBM Cognos Data Manager, or if you have followed the upgrade process, you can import a package created in IBM Cognos DecisionStream Series 7. In this case, Data Manager updates the script for the catalog objects and creates components as necessary.

For information, see “Component Packages” on page 321.

Post Upgrade Considerations

After upgrading a catalog, you must consider the impact that the upgrade may have on existing components and operations.

Command Lines in Scripts

The default location of IBM Cognos Data Manager executables is \Program Files\ibm\cognos\c10

A script that is used to call Data Manager builds and JobStreams should be updated to the Data Manager location.

For more information, see Chapter 31, “Commands,” on page 377, and the IBM Cognos *Function and Scripting Reference Guide*.

IBM Cognos Data Manager SQLTXT Designer Definition Files

When a SQLTXT definition file (.def) is amended and saved in IBM Cognos Data Manager, it cannot be opened in IBM Cognos DecisionStream Series 7. We recommend that you create a backup copy of all SQLTXT definition files before you use them in Data Manager.

Source Code Control Projects

When you upgrade an IBM Cognos DecisionStream Series 7 catalog that is stored as a source code control project, it is no longer attached to source code control in IBM Cognos Data Manager. It exists as a stand-alone catalog. To share the catalog with other developers, you must add it to source code control again.

Note: To ensure that you retain the full project history, you must add the catalog to a new project in your source code control system.

For more information, see “Add a Catalog to Source Code Control” on page 316.

Metadata Deliveries

IBM Cognos Data Manager no longer delivers metadata using metadata deliveries in fact builds. Any metadata deliveries to IBM Cognos Series 7 Transformer, IBM Cognos Series 7 Architect, and Microsoft Analysis Server that you previously created in an IBM Cognos DecisionStream Series 7 catalog are no longer supported. All metadata is now exported directly to IBM Cognos Framework Manager using the metadata export utility in Data Manager.

DB-LIB Connections and Microsoft SQL Server Bulk Loader Connections

Any DB-LIB connections that you previously created in an IBM Cognos DecisionStream Series 7 catalog are migrated to OLE-DB connections when you upgrade to IBM Cognos Data Manager.

In addition, if your DecisionStream Series 7 catalog contains either of the following fact deliveries with a DB-LIB connection, these connections are also migrated:

- Bulk Copy (BCP) deliveries to Microsoft SQL Server
- Microsoft SQL Server BCP deliveries

You should test all migrated connections in Data Manager to ensure that they function as you expect.

IBM Cognos Data Manager Network Services

When you upgrade to IBM Cognos Data Manager, you must take into consideration changes that are made to the Data Manager Network Services server and client components.

IBM Cognos Data Manager Network Services Server Component

In IBM Cognos Data Manager, the Data Manager Network Services server component has been modified so it is more in line with the rest of the IBM Cognos Business Intelligence infrastructure:

- There is now a common IBM Cognos BI service that replaces the SOAP server used in IBM Cognos DecisionStream Series 7.
- The Socket server now has a different name, Data Manager Network Services Socket Server, and is located on a different port to the one used in DecisionStream Series 7.

As a result, when you upgrade to Data Manager, the Data Manager Network Services server component can exist concurrently with the DecisionStream Series 7 server component.

IBM Cognos Data Manager Network Services Client Component

In IBM Cognos Data Manager, the file dsnethosts.txt no longer exists. Instead the information contained in that file is maintained using IBM Cognos Configuration.

As a result, any changes that you previously made to the dsnethosts.txt file in the IBM Cognos DecisionStream Series 7 client component are not migrated when you upgrade to Data Manager. You must manually incorporate these changes into IBM Cognos Configuration as follows:

- Use a text editor to open the dsnethosts.txt file, and review the changes you have made.
- Open IBM Cognos Configuration and update the configuration as required.

For more information on configuring Data Manager Network Services, see the IBM Cognos Data Manager *Installation and Configuration Guide*.

Chapter 6. SQLTerm and SQL Helper

Use Microsoft SQLTerm and SQL Helper to

- build SQL statements to access source data
- obtain information about database objects, DBMS drivers, and data sources
- examine the contents of database tables
- test SQL extracts from source databases
- tune SQL extracts

SQLTerm and SQL Helper also support

- Data Definition Language (DDL), for example, CREATE TABLE
- Data Manipulation Language (DML), for example, INSERT INTO
- queries about the connection, for example, SOURCES

Note: Whether IBM Cognos Data Manager shows SQLTerm or SQL Helper depends on what you are doing when you require assistance with constructing an SQL statement. You use SQLTerm and SQL Helper in exactly the same way.

For information about using SQLTerm and SQL Helper with Published FM Packages, see “Add Published FM Package Data Source” on page 119. For information about using SQLTerm on the command line, see “sqlterm” on page 399.

Create SQL Statements

Using SQLTerm or SQL Helper, you can

- add a SELECT statement for all columns in the selected table or for a single column
- add a table name or column name to a statement
- add a DROP TABLE, DELETE FROM, SELECT FROM, or DESCRIBE statement for the selected table

Add a SELECT Statement to an SQL Query

In addition to using the method described here, you can also type an SQL statement directly into the box in SQLTerm or SQL Helper.

You can use only a single SELECT statement in a query. Multiple SELECT statements result in an error.

Procedure

1. From the **Tools** menu, click **SQLTerm** SQL.
2. In the box at the top of the **SQLTerm** window, click the database with which you want to work.
3. Use the **Cognos SQL** check box to specify whether you want to use the native SQL or Cognos SQL for the database you are accessing when you construct the SQL statement. Select or clear the check box as appropriate.

   Tip: Using Cognos SQL makes it easier to port your project between database platforms.

Note: The default for the **Cognos SQL** check box is determined by whether you selected the **Cognos SQL** check box in the **Connection Properties** dialog box.

For more information on using Cognos SQL, see "Cognos SQL" on page 17.

4. In the **Database Objects** pane, expand the tree, and then click the table or column for which you want to add a SELECT statement.

   Tip: To refresh the list of tables in the **Database Objects** pane, double-click the database object, or right-click and then click either **Refresh Database** or **Refresh Table**.

5. To generate the SQL statement use one of the following methods:
   - Right-click, and then click either **Add Table Select Statement**, or **Add Column Select Statement**.
   - Hold down **Ctrl** and drag the table or column you selected in the **Database Objects** pane to the **Query** pane.

   IBM Cognos Data Manager clears any existing query that may be in the **Query** pane and generates the SQL SELECT statement for the selected table or column.

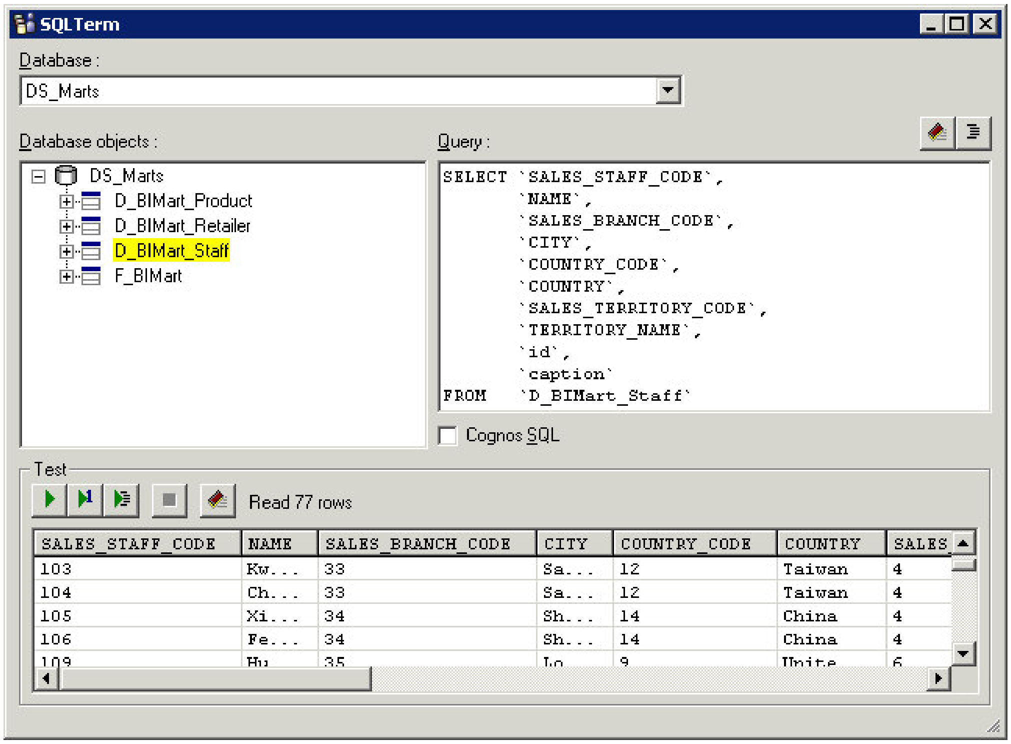
   Tip: To clear the query, click the **Clear the query** button.

6. You are now ready to execute the SQL statement.

   For more information, see "Test a Data Source" on page 123.

Example

This example shows the result of adding a SELECT statement for the table named D_BIMart_Staff and executing the generated statement by clicking the **Return all rows** button.



```
SELECT `SALES_STAFF_CODE`,
       `NAME`,
       `SALES_BRANCH_CODE`,
       `CITY`,
       `COUNTRY_CODE`,
       `COUNTRY`,
       `SALES_TERRITORY_CODE`,
       `TERRITORY_NAME`,
       `id`,
       `caption`
FROM   `D_BIMart_Staff`
```

□ Cognos SQL

Read 77 rows

SALES_STAFF_CODE	NAME	SALES_BRANCH_CODE	CITY	COUNTRY_CODE	COUNTRY	SALES_
103	Kw...	33	Sa...	12	Taiwan	4
104	Ch...	33	Sa...	12	Taiwan	4
105	Xi...	34	Sh...	14	China	4
106	Fe...	34	Sh...	14	China	4
109	Hu	35	Lo	9	Unite	6

Data Manager generates a SELECT statement for all the columns in the PRODUCT_TYPE table. At the bottom of the window, the results pane shows the number of rows and the data returned by the SQL statement.

Add a Table Name or Column Name to an SQL Query

Using SQLTerm, you can easily add table names and column names to an SQL query. You can enter the name alone or you can precede it with a comma.

Procedure

1. In the box at the top of the **SQLTerm** window, click the database with which you want to work.
2. Use the **Cognos SQL** check box to specify whether you want to use the native SQL or Cognos SQL for the database you are accessing when you construct the SQL statement. Select or clear the check box as appropriate.

   Tip: Using Cognos SQL makes it easier to port your project between database platforms.

   Note: The default for the **Cognos SQL** check box is determined by whether you selected the **Cognos SQL** check box in the **Connection Properties** dialog box.

   For more information on using Cognos SQL, see "Cognos SQL" on page 17.
3. In the **Database Object** pane, click the table name or column name and then, either:
   - Right-click, and then click either **Add Table** or **Add Column**, or drag the selected table name or column name to the **Query** pane.
   - Right-click, and then click either **Add Comma and Table** or **Add Comma and Column**, or hold down **Shift** and drag the selected table name or column name to the **Query** pane.

   Tip: To clear the query, click the **Clear the query** button .
4. You are now ready to execute the SQL statement.

   For more information, see "Test a Data Source" on page 123.

Add Other Statements to an SQL Query

You can add statements to drop a table (DROP TABLE), delete rows (DELETE FROM), select rows (SELECT FROM), or describe a table (DESCRIBE).

Procedure

1. In the box at the top of the **SQLTerm** window, click the connection with which you want to work.
2. In the **Database Objects** pane, right-click the table for which you want to add the statement, and then click the action to perform.

   IBM Cognos Data Manager adds the selected statement.
3. You are now ready to execute the SQL statement.

   For more information, see "Test a Data Source" on page 123.

List the available SQLTerm commands

In addition to using SQLTerm to connect to and query SQL databases, you can type commands in the SQL Query pane to obtain additional information. You can type commands to

- obtain information about database objects, DBMS drivers, and data sources

- execute SQL scripts
- obtain help on SQLTerm commands

Procedure

1. In the **Query** pane, type **Help**.
2. Click the **Return all rows** button ▶.

   These are the available commands.

Command	**Description**
DESCRIBE	Lists all the tables within the current database.
DESCRIBE <table_name>	Describes the columns within the specified table. Where possible, these are the column name, its data type and length, whether the column can accept null values, and whether the column forms part of the primary key for the table.
OBJECTS <TVS*><schema>	Lists all the objects within the current table views schema (TVS) and database.
DIFF <tab1><tab2>	Compares two tables and lists the columns that contain differences.
DRIVERS [ALL]	Lists the DBMS drivers for which IBM Cognos Data Manager is configured. Use the ALL argument to obtain a list of all the drivers known to Data Manager.
FETCHSIZE <N>	Sets or shows the fetch row count used to query the database. This setting controls the number of rows that are queried at once by using the bulk fetch facility for your database. You can use this to tune large extractions from a source database. This setting has no effect for databases that do not support bulk fetch operations. The default fetch row size is 50.
ISOLEVEL <N>	Sets or shows the database isolation level defined for the database transaction you are using to query the data source. This setting can be used where you specifically want to control the transaction isolation level for a query transaction. This setting has no effect for database systems that do not support transaction isolation levels. The default value is determined by the source database system.
OUTPUT [ON|OFF]	Sets whether the data that results from SQL SELECT statements is shown. On its own, this command returns the OUTPUT setting (either ON or OFF). OUTPUT ON shows data; OUTPUT OFF does not.
PROGRESS <N>	Shows the select progress for every *n* rows.
PROMPT <text>	Shows a line of text, which is useful for debugging and inserting custom messages in SQLTerm or SQL Helper output files.

Command	Description
SCHEMA	Lists the schema from the current connection.
SCREEN [<width>]	Sets the maximum number of characters on a line before the output is wrapped. The default is to return the current width.
SOURCES [<driver>] [ALL]	Lists all the available data sources. The default is to return a list of all data sources that are available by using the driver that is currently in use. If you specify the optional ALL argument, SOURCES returns all data sources that are available through any DBMS driver. You can also list the data sources that are available through any particular driver.
STATEMENT [SELECT | DML | DDL]	Indicates the type of the next SQL statement.
STATISTICS [ON | OFF]	Determines whether timing statistics are returned after executing each SQL statement. This command is particularly useful in determining SQL statement efficiency.
TIME	Shows the current system date and time.
TYPES	This command shows information about the current connection. For ODBC, it also shows information about the data type conversion.
Help	Shows all the commands included in this table and a brief description of each.

Test an SQL Query

You can choose whether you want to return all the selected rows in the database or just a single row. You can also execute the SQL query and return all the rows.

Return All the Selected Rows in the Database

Procedure

From the toolbar in the **SQLTerm** window, click the **Return all rows** button ▶.

Limit Results to a Single Row

Procedure

From the toolbar in the **SQLTerm** window, click the **Return one row** button ▶1.

Execute the Statement and Return the Rows

Procedure

In the **Test** pane, click the **Execute statement(s)** button ▶.

Interrupt Execution of an SQL Query

Procedure

From the toolbar in the **SQLTerm** window, click the **Interrupt current processing** button ■.

Include Substitution Variables in an SQL Query

You can execute SQL queries that include substitution variables so that you can run an SQL query and view the results as the query is constructed. Run-time variables can be used to control queries, especially filters.

Note: If you include substitution variables in an SQL query for a reference structure, you must declare the variables in the appropriate fact build or dimension build.

For more information about substitution variables, see Chapter 24, "Variables," on page 289.

When you execute an SQL query that includes a variable, a dialog box appears for you to enter the value of each variable used in the query. IBM Cognos Data Manager uses these values when the query is executed.

Procedure

1. In the SQLTerm or SQL Helper **Query** pane, construct the required SQL query that includes a substitution variable.
2. Execute the SQL query.

   For more information, see "Test a Data Source" on page 123.

   The **Values for Expression** window appears listing each substitution variable included in the query.
3. In the **Value** column, type a suitable value for each variable.

   Tip: Click **Set Default Values** to enter the default value that you specified when you defined the substitution variable.

   For more information about default values, variable data type and scope, see Chapter 25, "Testing Expressions and Scripts," on page 309.
4. Click **OK**.

   The window shows the SQL statements that are generated after variable substitution has occurred and after processing any Cognos SQL syntax or functions that are not supported by the underlying database.

View the Executed Query

If required, you can view the executed SQL query.

Procedure

Click the **Show executed query** button ≣.

Tip: Select the **Word wrap** check box to specify that word wrapping should be used.

Chapter 7. Connections

To configure IBM Cognos Data Manager to interact with data sources, you must establish connections.

These connections

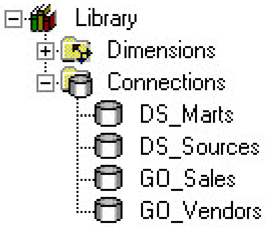
- identify the database
- specify the connection method that Data Manager must use to connect to the data
- provide information that the database management system (DBMS) requires when Data Manager connects to the data; for example, the username and password

Each connection definition is stored in a Data Manager catalog. This means that you define a connection once, and then use it in as many components as you require.

Having created an appropriate connection to interact with the data sources, you treat all data sources in an identical manner.

You can create a connection and change the physical database to which it relates. For more information, see Appendix G, “Cross Platform Deployment,” on page 507.

Connections are shown in the Library tree as follows.



Create a Database Connection

When you create a database connection, you must supply the connection parameters that the database requires. Each database type has its own particular requirements that typically include information to identify the database and a valid username and password.

You must also specify whether you want to use Cognos SQL, which is an extension of SQL 99. Using Cognos SQL, you have a greater degree of portability between mixed database environments because a common dialect can be used. You cannot use Cognos SQL for SQLTXT connections.

By default, a connection accepts any vendor-specific SQL SELECT statement in a data source, including non standard SQL extensions and hints.

For information on creating an IBM Cognos data source connection, see “Create an IBM Cognos Data Source Connection” on page 42.

For information on creating a Published FM Package connection, see "Create a Published FM Package Connection" on page 43.

Procedure

1. In the **Library** folder , click **Connections**.
2. From the **Insert** menu, click **Library**, and then click **Connection**.

   The **Connection Properties** dialog box appears.
3. Click the **General** tab.
4. In the **Alias** box, type an alias for the connection.

   IBM Cognos Data Manager uses this alias whenever it refers to the connection.
5. In the **Description** box, if required, type notes about the connection.
6. Click the **Connection Details** tab.

   Note: You can set up a connection to which you do not have access from your computer. This allows the connection to be used on another computer on which the connection method if available.
7. Click the appropriate connection method for the database to which you want to connect.

   Data Manager updates the dialog box to show fields that are appropriate for the connection method that you chose.
8. Enter the values required by the connection method.
9. In the **Collation sequence** box, type the collation sequence for the selected database.

   For more information, see "Distributed Sorting."
10. Select the **Cognos SQL** check box to use Cognos SQL when you construct an SQL statement for components using this connection. If you clear this check box, you must use native SQL for the database you are accessing.

    The selection that you make here determines the default for the **Cognos SQL** check box in other components that use this connection.
11. To test the connection, click **Test Connection**.

    A message appears stating whether the test was successful. Click **OK** to close the message box.
12. Click **OK**.

Distributed Sorting

Both IBM Cognos Data Manager and source databases can sort data. Problems can arise if the sort order of Data Manager and the source databases differ. This often occurs if Data Manager and the source databases are operating in different locales.

To help avoid problems with compatible sort orders, you can instruct Data Manager that sorting should be performed by the Data Manager computer rather than the underlying data source.

Note: This only affects data sources that use Cognos SQL.

You specify how distributed sorting is to be performed in the Collation sequence box in the Connection Properties dialog box. You can enter

- COMPATIBLE (default)
- NOT-COMPATIBLE

- a database-specific setting

COMPATIBLE

If the database to which you are connecting sorts data in the same order as the computer on which IBM Cognos Data Manager is running, enter COMPATIBLE.

This causes sorting to be distributed between the computer on which Data Manager is running and the database to which you are connecting. This is the default.

NOT-COMPATIBLE

If the database to which you are connecting sorts data in a different order to the computer on which IBM Cognos Data Manager is running, enter NOT-COMPATIBLE.

This causes all sorting to take place on the computer on which Data Manager is running.

Entering NOT-COMPATIBLE decreases the performance of queries that use distributed sorting. This difference in performance is dependent on the relative speed and power of the computer running your database versus the speed and power of the computer running Data Manager. Therefore, you should avoid unnecessary use of the NOT-COMPATIBLE setting.

Database-Specific Setting

A database-specific setting can only be used if you are accessing single-byte data.

The following lists show some example collation sequences.

EBCDIC

This section shows some example s of collation sequences for EBCDIC databases.

Code page	Collation sequence	Description
00037	EBCDIC00037	United States and Canada code page
00500	EBCDIC00500	International #5 code page

Microsoft Access

This section shows some example s of collation sequences for Microsoft Access databases.

Collation sequence	Description
ACCESSENG	English
ACCESSSP	Spanish
ACCESSNOR	Nordic
ACCESSDUT	Dutch

Microsoft SQL Server

This section shows some example s of collation sequences for Microsoft SQL Server databases.

Code page	Collation sequence	Description
	SYBISODOCI	Dictionary order and not case-sensitive
850	SYBCP850DOC1	Not case-sensitive

IBM DB2

This section shows some example s of collation sequences for IBM DB2® databases.

Collation sequence	Description
D2ENCASE	English and case-sensitive

Create an IBM Cognos Data Source Connection

An IBM Cognos Data Source connection allows you to access data sources set up in IBM Cognos Business Intelligence.

Note: IBM Cognos Data Manager can only access data from data sources that accept Cognos SQL.

Before you can create an IBM Cognos Data Source connection, you must

- configure Data Manager to access the Gateway URI and the Dispatcher URI

  For information, see the IBM Cognos Data Manager *Installation and Configuration Guide*.
- be logged on to IBM Cognos Connection, the portal to IBM Cognos BI, which provides a single access point to all corporate data available in IBM Cognos BI.

  Note: If the server to which you are connecting accepts anonymous users, you do not have to log on. For information, see “Manage Credentials” on page 47 and the IBM Cognos Connection *User Guide*.

Procedure

1. In the **Library** folder , click **Connections**.
2. From the **Insert** menu, click **Library**, and then click **Connection**.

   The **Connection Properties** dialog box appears.
3. Click the **General** tab.
4. In the **Alias** box, type an alias for the connection.

   Data Manager uses this alias whenever it refers to the connection.
5. In the **Description** box, if required, type notes about the connection.
6. Click the **Connection Details** tab.

Note: You can set up a connection to which you do not have access from your computer. This allows the connection to be used on another computer on which the connection method if available.

7. Click **Cognos Data Source**.
8. Click the browse button ... .

   The **Select the Data Source** dialog box appears listing all the available IBM Cognos data sources.
9. Click the data source that you want to use, and then click **OK**.

   The name of the selected data source is shown in the **Data Source Name** box.
10. If required, in the **Connection name** box, specify the connection name defined for the data source in IBM Cognos BI.
11. If required, in the **Signon name** box, specify the signon name used to the connect to the data source in IBM Cognos BI.
12. If required, select the **Execute connection/session commands** box to run the database commands associated with data source connections in IBM Cognos Connection.

    For more information on using database commands in IBM Cognos Connection, see the IBM Cognos *Administration and Security Guide*.
13. Select the **Cognos SQL** check box to use Cognos SQL when you construct an SQL statement for components using this connection. If you clear this check box, you must use native SQL for the database you are accessing.

    The selection that you make here determines the default for the **Cognos SQL** check box in other components that use this connection.
14. To test the connection, click **Test Connection**.

    A message appears stating whether the test was successful. Click **OK** to close the message box.
15. Click **OK**.

Create a Published FM Package Connection

A Published FM Package connection allows you to access query subjects from a published IBM Cognos Framework Manager package. Each Published FM Package connection can access one published package.

Note: You cannot use Cognos SQL for Published FM Package connections.

Before you can create a Published FM Package connection, you must

- configure IBM Cognos Data Manager to access the Gateway URI and the Dispatcher URI

  For information, see the IBM Cognos Data Manager *Installation and Configuration Guide*.
- be logged on to IBM Cognos Connection, the portal to IBM Cognos BI, which provides a single access point to all corporate data available in IBM Cognos BI.

  Note: If the server to which you are connecting accepts anonymous users, you do not have to log on.

  For information, see “Manage Credentials” on page 47 and the IBM Cognos Connection *User Guide*.

Note: When using IBM Cognos Connection with multiple data source connections, you can use the DM_DATASOURCE_CONNECTION_SIGNONS variable to specify which data sources to use. For information, see "DM_DATASOURCE_CONNECTION_SIGNONS" on page 294.

Procedure

1. In the **Library** folder, click **Connections**.
2. From the **Insert** menu, click **Library**, and then click **Connection**.

   The **Connection Properties** dialog box appears.
3. Click the **General** tab.
4. In the **Alias** box, type an alias for the connection.

   Data Manager uses this alias whenever it refers to the connection.
5. In the **Description** box, if required, type notes about the connection.
6. Click the **Connection Details** tab.

   Note: You can set up a connection to which you do not have access from your computer. This allows the connection to be used on another computer on which the connection method if available.
7. Click **Published FM Package**.
8. Click the browse button ... .

   The **Select the Package** dialog box appears listing all the available published packages.
9. Click the package that you want to use, and then click **OK**.

   The name of the selected package is shown in the **Package** box.
10. To test the connection, click **Test Connection**.

    A message appears stating whether the test was successful. Click **OK** to close the message box.
11. Click **OK**.

Use an Alias to Connect to a Database

IBM Cognos Data Manager uses connection aliases to describe the connections to databases.

For example, a connection named Sales may use an Oracle connection of username/password@service1 allowing specified connections to be re-used throughout the design process.

Although a connection string may be the preferred definition during the design phase of a Data Manager project, the test or production environment may need to connect to a different physical database. The database alias definition file allows multiple run environments to be defined to determine to which database Data Manager connects when you specify a particular alias. At run-time, the database alias definition file replaces the connection definitions defined as part of the Data Manager catalog. For example, to use a database alias definition file in a Data Manager database, you would type

```
databuild -ePRODUCTION -Ac:\workfiles\Alias.txt "ODBC" "DSN=DS_Tutorial"
CopyTable
```

Create a Database Alias Definition File

Database alias definition files are text files that specify to which databases IBM Cognos Data Manager can connect.

Syntax

A database alias definition file also specifies the parameters that Data Manager must use when connecting to the data. Each database alias definition file must define all the connections that a build requires.

```
<runenv> <name> <driver> <driver_items> [<notes>]
```

Symbol	Description
<runenv>	The run environment context for the database alias. By default, Data Manager uses <DEFAULT> when processing your projects. Additional run environments are created for specific purposes.
<name>	The alias for the connection.
<driver>	The name of the database driver that Data Manager should use to connect to the data. For example, ORACLE or ODBC.
<driver_name>	The information that Data Manager should pass to the database driver to establish the connection. For example, user ID and password.
<notes>	Optional notes to describe the alias. You can only view such notes within the alias definition file.

Example

The following is an example database alias definition file.

```
//******************
//* DEFAULT ALIASES *
//******************
<DEFAULT> 'Tutorial Catalog' ODBC 'DSN=DS Tutorial Catalog' ''
```

Password Encryption

IBM Cognos Data Manager supports both encrypted and plain-text passwords for database aliases.

For encrypted passwords, Data Manager uses its own encryption algorithm. It determines whether a password is encrypted and automatically deciphers encrypted passwords before connecting to the target database. To use encrypted passwords, you must create database connections within Data Manager Designer and then, export those connections to an alias definition file. You can export the connections using the CATEXP command, or **Show Source** from the **Actions** menu.

For more information, see “catexp” on page 387.

Change the Properties of a Catalog

You can change the properties of a catalog.

Procedure

1. In the **Tree** pane, click the catalog.
2. From the **Edit** menu, click **Properties**.
3. Change the properties as you require.
4. Click **OK**.

Test a Connection

You can test a connection to ensure that it is set up correctly.

Procedure

1. Click the connection you want to test.
2. From the **Edit** menu, click **Properties**.

   The **Connection Properties** dialog box appears.
3. Click the **Connection Details** tab.
4. Click **Test Connection**.

   If the test is successful, a message appears indicating this. If the test is unsuccessful, the native database driver error messages appear.
5. Click **OK** to acknowledge the message.

Delete a Connection

You cannot delete a connection that another IBM Cognos Data Manager component is using.

Tip: You can determine the components that use a connection by using the Data Manager navigator. For more information, see “Locate Components and Their Dependencies” on page 15.

Procedure

1. Click the connection you want to delete.
2. From the **Edit** menu, click **Delete**.

   A message appears prompting you to confirm that you want to delete the selected connection.
3. Click **Yes**.

   If another Data Manager component is using the connection, an error message appears showing all the components that use the connection. Click **OK** to acknowledge this message.

Open the ODBC Data Source Administrator

You may want to use the ODBC Data Source Administrator when you create a connection.

You can do this from the Microsoft Windows operating system Control Panel or directly from IBM Cognos Data Manager Designer.

Procedure

From the **Tools** menu, click **ODBC Administrator**.

Manage Credentials

IBM Cognos Connection is the portal to IBM Cognos Business Intelligence and provides a single access point to all corporate data available in IBM Cognos BI.

IBM Cognos BI supports authenticated and anonymous user access. To use IBM Cognos BI as an authenticated user, you must successfully log on. During the log on process, you must provide your credentials, such as user ID and password, as required by your organization. Anonymous users do not log on.

For more information, see the IBM Cognos Connection *User Guide*.

When using IBM Cognos Data Manager to access corporate data available in IBM Cognos BI, there are two types of credentials:

- Designer credentials are used while you are working in Data Manager Designer and are obtained from the IBM Cognos Connection log on window
- catalog credentials are used when you execute a build or JobStream and are read from the catalog during command line exection

This means, for example, that you could log on with Designer credentials as User A to set up a build using Data Manager Designer, and then execute the same build using the catalog credentials of User B.

Designer Credentials

When using IBM Cognos Data Manager Designer, some actions require that you access IBM Cognos Connection to access an IBM Cognos data source or published IBM Cognos Framework Manager package.

You can either log on using Logon As from the Data Manager Actions menu, or you can wait until you perform an action for which you must connect to IBM Cognos Connection, such as creating a Published FM Package connection, when the Log on window automatically appears. If you have previously logged on during the current Designer session, you can access the data without logging on again.

Procedure

1. From the **Actions** menu, click **Logon As**.

   The IBM Cognos Connection **Log on** window appears.
2. Enter the information required to log on.

   For information, see the IBM Cognos Connection *User Guide*.

   Tip: You can also log on using the **Manage Catalog Credentials** dialog box.

   For more information, see "Catalog Credentials."
3. To log off, from the **Actions** menu, click **Logoff.**

   A message appears informing you that you are logged off.

   For information, see the IBM Cognos Connection *User Guide*.

   Tip: You can also log off using the **Manage Catalog Credentials** dialog box.

   For more information, see "Catalog Credentials."

Catalog Credentials

Catalog credentials are used when you execute a build or JobStream from the command line.

Create the Stored Catalog Credentials

If there are no catalog credentials stored, a status message appears to indicate this in the Manage Catalog Credentials dialog box. The user whose credentials you want to store must log on and store them.

Procedure

1. From the **Actions** menu, click **Manage Execution Credentials**.

   If you are not logged on to IBM Cognos Connection, you must do so before you can set the catalog credentials.
2. To log on, click **Logon As**, and enter the information required.

   For information, see the IBM Cognos Connection *User Guide*.
3. Click **Create credentials**, to store the catalog credentials.

Manage the Catalog Credentials

If the catalog credentials do not match the current Designer credentials that you have used to log on, a status message appears to indicate this in the **Manage Catalog Credentials** dialog box. If required, you can store your current Designer credentials as the catalog credentials.

Procedure

1. From the **Actions** menu, click **Manage Execution Credentials**.

   If you are not logged on to IBM Cognos Connection, you must do so before you can set the catalog credentials.
2. To log on, click **Logon As**, and enter the information required.

   For information, see the IBM Cognos Connection *User Guide*.
3. Click **Set to current account**, to set the catalog credentials to be the same as the Designer credentials.

   Tip: To clear the catalog, click **Clear catalog credentials**.

Renew the Stored Catalog Credentials

The catalog credentials currently stored may not be valid if they have expired or the password has changed. If this occurs, the user whose credentials are stored must log on and renew them.

Procedure

1. From the **Actions** menu, click **Manage Execution Credentials**.

   If you are not logged on to IBM Cognos Connection, you must do so before you can set the catalog credentials.
2. To log on, click **Logon As**, and enter the information required.

   For information, see the IBM Cognos Connection *User Guide*.
3. Click **Renew credentials**.

Anonymous Users

Anonymous users do not log on. However, you can choose to override the anonymous account and log on using Logon As from the IBM Cognos Data Manager **Actions** menu. This forces a logon to one of your other namespaces. You can repeat this process for as many namespaces as you have. If you have logged on to all your namespaces, and attempt to log on again, a message informs you of this. Using anonymous access means that there is no authentication required to access IBM Cognos Business Intelligence data sources or published packages.

Chapter 8. Reference Dimensions

A reference dimension groups together all the reference structures (hierarchies, auto-level hierarchies, lookups) and templates that relate to a particular business dimension. Collectively, this is known as the dimensional framework.

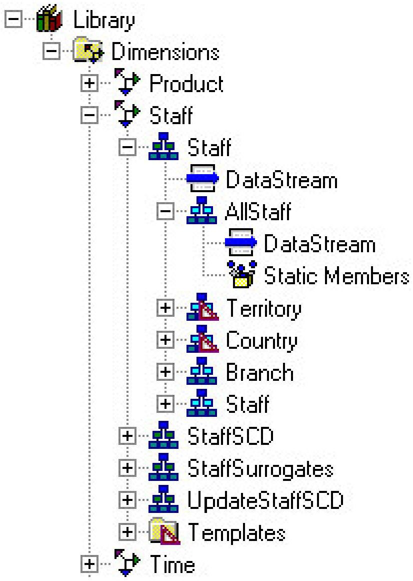
For more information, see “Business Dimensions” on page 4 and Chapter 11, “Reference Structures,” on page 63.

View Reference Dimensions

You can view reference dimensions in the Tree pane from the Catalog tab or the Library tab.

The Dimensions folder contains the dimensional framework. At the top level, the folder contains all the reference dimensions used in the catalog. Each reference dimension corresponds to a single business dimension.

All reference structures and templates that relate to a reference dimension appear within the reference dimension.



Create a Reference Dimension

You must create a reference dimension to hold the required reference structures and templates.

Procedure

1. In the **Library** folder, click **Dimensions**.
2. From the **Insert** menu, click **Library**, and then click **Reference Dimension**.

   The **Dimension Properties** dialog box appears.
3. Type a name for the reference dimension and, if required, a business name and description.

4. If the dimension is time-based, select the **Dimension Represents Time** check box.
5. Click **OK**.

Change the Properties of a Reference Dimension

You can change the properties of an existing reference dimension.

Procedure

1. Click the reference dimension for which you want to change the properties.
2. From the **Edit** menu, click **Properties**.
3. Change the properties as you require.
4. Click **OK**.

Delete a Reference Dimension

You cannot delete a reference dimension if another component from the same catalog is using it.

For information about component dependencies, see “Locate Components and Their Dependencies” on page 15.

Procedure

1. Click the reference dimension that you want to delete.
2. From the **Edit** menu, click **Delete**.
3. In the message box, click **Yes**.

   If you attempt to delete a reference dimension that has dependent components, an error message appears.

Chapter 9. Templates

A database provides basic information about a table, principally the column names and data types. However, some columns have further special significance to IBM Cognos Data Manager and are an integral part of dimension table maintenance. This extra column information is referred to as the column behavior. The extent to which Data Manager uses column behavior depends on the complexity of the dimension management. This complexity can vary between a simple dimension table with a business key value and various attributes, to a dimension table which fully implements attribute change tracking.

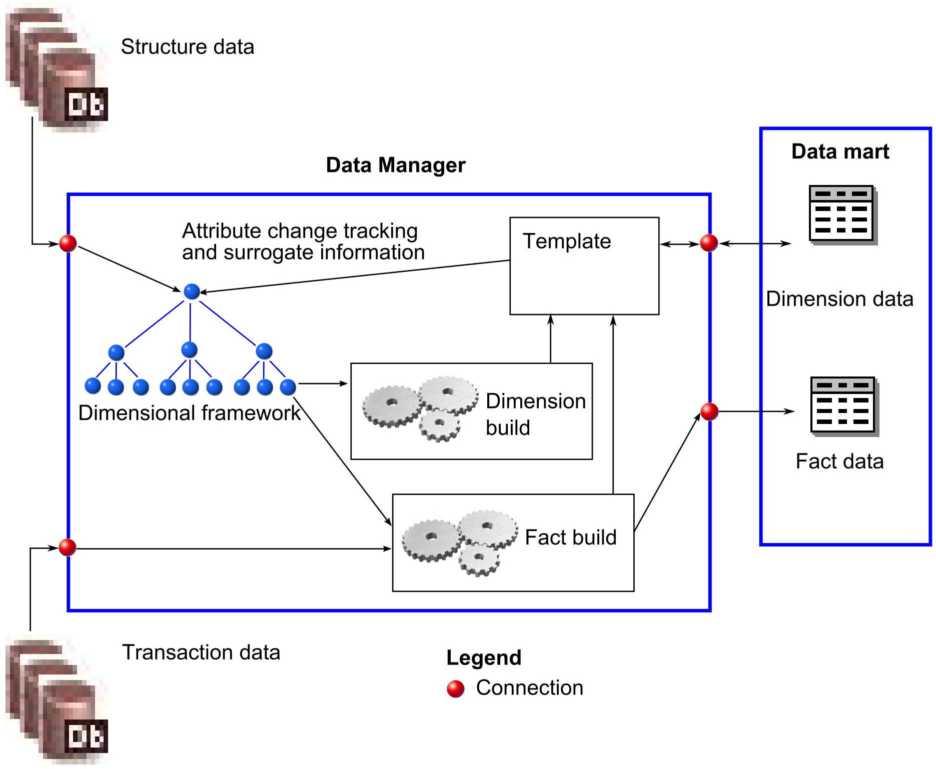
In Data Manager, you use a template to

- define the attributes required in a reference structure
- populate the reference structures with dynamic data
- define the behavior of the attributes when delivering dimension data

By defining the behavior of attributes, Data Manager can manage surrogates and unmatched members, track attribute changes, and minimize updates to the data mart dimension tables.

For more information, see Chapter 11, “Reference Structures,” on page 63, Chapter 10, “Surrogate Keys,” on page 59, and “Track Attribute Changes” on page 210. For information about unmatched members, see “Accept Source Data Containing Unmatched Dimension Values” on page 154.

The following illustration shows how Data Manager uses templates to deliver data to a data mart.



To use a template to deliver structure data and transaction data, you must complete this process:

- Create a template.
- Create a reference structure.
- Create a dimension build and select the template to use to specify the columns to deliver. In addition, specify whether to use attribute change tracking to deliver dimension data.
- Create a fact build to deliver the fact data.
- Specify how unmatched members are to be handled.

You can create a number of different templates for the same reference dimension. This means that by changing the template you can assign a different behavior set to any reference structure, dimension build, or dimension delivery.

Before you can create a template, you must create a reference dimension to hold the template. For information, see “Create a Reference Dimension” on page 49.

Dimension Attributes

A dimension consists of members. For example, the members of a product dimension are the individual products. Members have attributes that identify them and provide further information. For example, some possible attributes for a product dimension are the product code, name, type, color, and size.

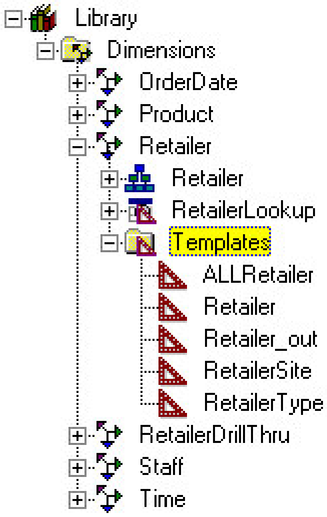
Each member of a dimension must have a business key to identify it in streams of transactional data. If the dimension is defined as a hierarchy, the lower levels of the hierarchy must also have an attribute that identifies the parent of each member.

Another common attribute is the business name, which enables analysis software to make reports more comprehensible by using the business name in place of the business key.

In the environment of a data warehouse, information about each business dimension is stored in one or more dimension tables. Because it is common to store as much information as possible in the dimension tables, a dimension table may have many columns, each of which corresponds to a dimension attribute.

View Templates

All reference structures and templates that relate to a particular business dimension appear within a reference dimension in the Tree pane.



Create a Template

You must select a template for every reference structure, dimension build, and dimension delivery in your catalog.

Note: You do not have to create a separate template in each place that you select one. You can use any template, with suitable attributes defined, within the reference dimension.

When you create a template, you define attributes for it so that attributes are available when you create a reference structure.

Typically, a simple dimension table has attributes such as ID, caption, and surrogate. If you want to track attribute changes, you use attributes such as effective start date and current indicator.

Procedure

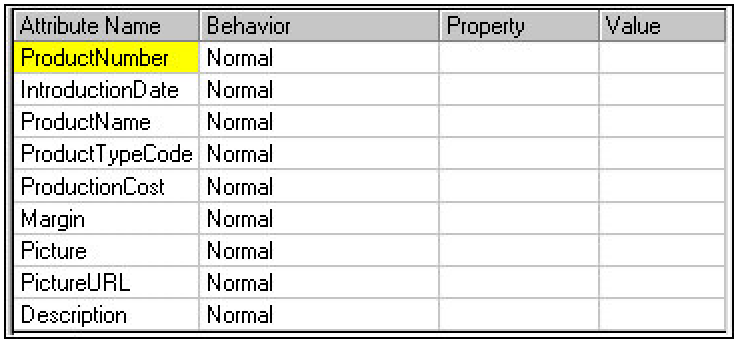
1. In the appropriate **Dimensions** folder, click the **Templates** folder.
2. From the **Insert** menu, click **Library**, and then click **Template**.

   The **Template Properties** window appears.
3. Click the **General** tab.

4. In the **Name** box, type a name for the template and, if required, a business name and description.
5. Click the **Attributes** tab.
6. Click **Import Table** if the attributes you want to define for the template are already defined as columns in a database table.

   The **Select Table** dialog box appears.
7. Select the table from which to import attributes into the template, and then click **OK**.

   The database table columns are inserted into the **Attribute Name** column.



Attribute Name	Behavior	Property	Value
ProductNumber	Normal		
IntroductionDate	Normal		
ProductName	Normal		
ProductTypeCode	Normal		
ProductionCost	Normal		
Margin	Normal		
Picture	Normal		
PictureURL	Normal		
Description	Normal		

   By default, IBM Cognos Data Manager allocates a behavior type of normal.

   Tip: If you do not want to import a table or you want to add further attributes, you can define them manually by clicking **Add**.
8. In the **Behavior** box for an attribute, click the required behavior type.

   For information about behavior types and additional property values, see "Behavior Types in Templates."
9. If you select **Surrogate Key**, **Business Key,** or **Current Indicator** as the behavior type, you must specify additional property values in the **Value** column. Enter appropriate values for the selected behavior type.

   A template can have, at most, only one start date, end date, and current indicator. In addition, a business key can only be used by one surrogate key attribute.
10. Click **OK**.

Behavior Types in Templates

When you create a template, you define attributes for it. For each attribute in the template, you must assign a behavior type.

Surrogate Key Behavior

The surrogate key for the dimension table.

In the Value column, you must specify

- the column to which you have assigned business key behavior
- a start value if you want to ensure unique surrogate keys

  Type the initial number to be assigned by IBM Cognos Data Manager when generating surrogate values.

  By default, Data Manager starts each surrogate key series at 1.

For more information, see Chapter 10, "Surrogate Keys," on page 59.

Business Key Behavior

The identifier for the dimension table.

If a hierarchy has several levels, the template may have several business keys, one for each hierarchy level ID. However, only one of these business keys can be the primary key for the table. The lowest level business key should be defined as the primary key.

In the Value column, specify the primary key by clicking True or False.

Note: If you used a composite key to define the business ID, you must use a DataStream derivation to concatenate it. For more information, see “Derivations” on page 128.

Effective Start Date Behavior

The earliest date to which the dimension data row applies. IBM Cognos Data Manager uses this date to track attribute changes.

The range over which the data row applies is from the effective start date to the effective end date inclusive (or the current date if there is no end date).

For more information, see “Specify Effective Date Options” on page 220.

Effective End Date Behavior

The last date to which the dimension data row applies. IBM Cognos Data Manager uses this date to track attribute changes.

For more information, see “Specify Effective Date Options” on page 220.

Create Date Behavior

The date on which IBM Cognos Data Manager created the dimension data row. Data Manager uses this date to track attribute changes.

Last Update Date Behavior

The date on which IBM Cognos Data Manager last updated the dimension data row. Data Manager uses this date to track attribute changes.

Current Indicator Behavior

Indicates whether the dimension data row contains the current value or past value for the member. IBM Cognos Data Manager uses this value to track attribute changes.

In the Value column, specify the current value and past value. You can specify any non-null value you require. By default, Data Manager enters Y for current value and N for past value.

Normal Behavior

Any other attributes.

Change the Default Column Names for Behavior Types

For some behavior types, IBM Cognos Data Manager allocates a default column name, which you can change if required. The following table shows the default column names.

Behavior	Default column name
Effective start date	eff_date
Effective end date	end_date
Create date	cre_date
Last update date	udt_date
Current indicator	curr_ind
Current value	Y
Past value	N
Surrogate key	skey

Procedure

1. From the **Tools** menu, click **Options**.
2. Click the **Attribute Defaults** tab.
3. Enter values as required.

   Tip: You can revert to the default column names by clicking **Reset Defaults**.
4. Click **OK**.

Change the Properties of a Template

You can change the properties of an existing template.

Procedure

1. Click the template for which you want to change the properties.
2. From the **Edit** menu, click **Properties** .
3. Change the properties as you require.
4. Click **OK**.

Delete a Template

You cannot delete a template if another component from the same catalog is using it.

For information about component dependencies, see “Locate Components and Their Dependencies” on page 15.

Procedure

1. Click the template that you want to delete.
2. From the **Edit** menu, click **Delete**.
3. In the message box, click **Yes**.

Chapter 10. Surrogate Keys

Surrogate keys are numeric keys that can replace business keys in the context of a data mart. Using surrogate keys has many advantages:

- Surrogate keys are numeric.

  Therefore, they tend to be smaller than the keys from operational systems, which are typically text-based.
- Surrogate keys are unique.

  Operational keys may be unique within the database in which they reside. However, they may not be unique across a build that accesses data from multiple databases. To implement attribute change tracking, the keys must be unique.
- Surrogate keys allow you to combine data from tables with incompatible keys.
- Surrogate keys have no intrinsic meaning.

  Therefore, they are not subject to the same changes as operational keys. For example, product codes may expand to accommodate additional information.

The following table shows an extract of the product reference data.

Product_code	Product_name
PRX 002 39 061	Trailchef water bag
PRX 003 40 067	Trailchef canteen
PRY 003 47 086	Trailchef kitchen set

The second table illustrates the use of surrogate keys for product reference data.

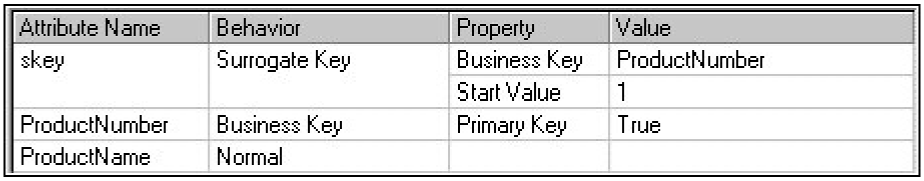
Surrogate_id	Product_code	Product_name
1	PRX 002 39 061	Trailchef water bag
2	PRX 003 40 067	Trailchef canteen
3	PRY 003 47 086	Trailchef kitchen set

Implement Surrogate Keys

To use surrogate keys in IBM Cognos Data Manager, you must first implement them in the appropriate reference structures. When you specify the template to use, ensure that

- the template includes an attribute that has surrogate key behavior
- the surrogate key is linked to a reference structure attribute assigned as a business key

In the following template example, the attribute named skey has surrogate key behavior. It is linked to the ProductNumber attribute, which is defined as the primary business key for the template.



Attribute Name	Behavior	Property	Value
skey	Surrogate Key	Business Key	ProductNumber
		Start Value	1
ProductNumber	Business Key	Primary Key	True
ProductName	Normal		

For information about attributes with surrogate key behavior, see “Behavior Types in Templates” on page 54.

For more information about specifying a template to use for a reference structure, see Chapter 11, “Reference Structures,” on page 63.

Maintain Surrogate Key Values

After you implement surrogate keys in a reference structure, you can use them in

- fact builds that deliver surrogate keys in the output
- dimension deliveries
- dimension builds

By including a dimension element (in a fact build) or a template (in a dimension delivery or dimension build) for which you have implemented surrogate key behavior, IBM Cognos Data Manager assigns any required surrogate key values when you execute the build.

To maintain the surrogate key values, you must specify

- the source from which the surrogate key values are assigned
- the way in which the surrogate key values are delivered

Note: In a dimension build, it is common practice for Data Manager to generate surrogate values automatically. In this case, you do not need to specify a source for surrogate key values. For information, see “Generate Surrogate Key Values Automatically” on page 62.

Specify the Source of Surrogate Keys in a Reference Structure

You can use surrogate keys values from a source that you specify.

For example, you may have a text file that contains surrogate key values.

Procedure

Choose one of the following options:

- From the DataStream mapping window, map a DataStream item to the surrogate key level attribute. For more information, see “Map the DataStream Items to Level Attributes” on page 76.
- From a reference structure properties window, select the **Reference Surrogate** check box.

Note: Use reference surrogates if you want IBM Cognos Data Manager to automatically generate surrogate key values in a fact build or dimension delivery. For more information, see "Reference Surrogate Keys" on page 62.

Deliver Sourced Surrogate Key Values

When you execute a build, IBM Cognos Data Manager delivers the current surrogate value of each dimension member either from the source you specified or from the values that Data Manager generates by using reference surrogates.

You can specify that a fact build, a dimension build, or a dimension delivery is to deliver sourced surrogate key values in the output.

Procedure

Choose one of the following options:

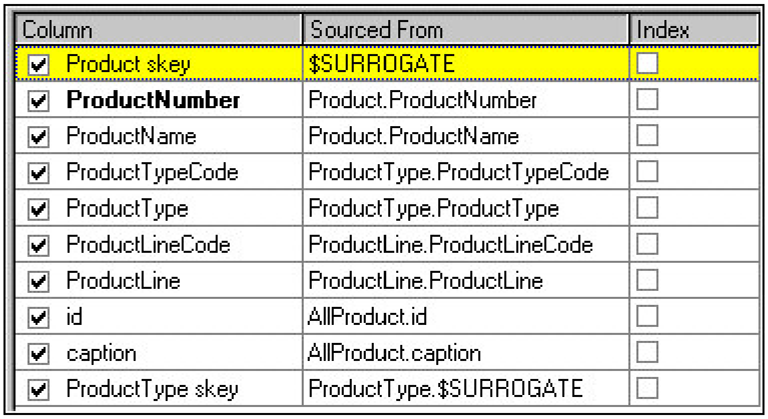
- To deliver sourced surrogate key values using a fact build, when you define the output levels for a dimension element, select the **Use surrogates when available** check box.

  For information, see "Set Output Levels for a Dimension or Derived Dimension" on page 150.
- To deliver sourced surrogate key values using a dimension build or dimension delivery, when you define the columns for a dimension table, you must select one of these column tokens as the source for any attribute with surrogate key behavior, either <level>.$SURROGATE, $SURROGATE, or $PARENT_SURROGATE.

  In the following example dimension table, two attributes have surrogate key behavior:

  Product skey, which is sourced from the $SURROGATE column token

  ProductType skey, which is sourced from the ProductType $SURROGATE column token



Column	Sourced From	Index
☑ Product skey	$SURROGATE	□
☑ **ProductNumber**	Product.ProductNumber	□
☑ ProductName	Product.ProductName	□
☑ ProductTypeCode	ProductType.ProductTypeCode	□
☑ ProductType	ProductType.ProductType	□
☑ ProductLineCode	ProductLine.ProductLineCode	□
☑ ProductLine	ProductLine.ProductLine	□
☑ id	AllProduct.id	□
☑ caption	AllProduct.caption	□
☑ ProductType skey	ProductType.$SURROGATE	□

For more information about sourcing dimension table columns and column tokens, see "Add a Table to a Dimension Build" on page 208.

Reference Surrogate Keys

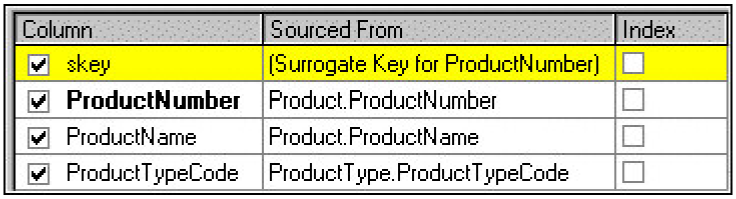
Reference surrogate keys are surrogate keys that are not automatically maintained. They are provided for backward compatibility with IBM Cognos DecisionStream 6.0 and to support snapshot data marts. By default, reference surrogate keys are not maintained and are regenerated each time you execute a build.

You can force IBM Cognos Data Manager to maintain surrogate keys values by using reference surrogates. When you deliver the dimension data, Data Manager assigns new surrogate key values for each dimension member by using the surrogate key start value defined in the associated template.

Generate Surrogate Key Values Automatically

You can generate surrogate key values automatically in a dimension build without sourcing them.

To specify that surrogate key values are to be generated, when you define the columns for a dimension table (in the Dimension Table Properties window), do not select a source for any attribute with surrogate key behavior.



Column	Sourced From	Index
☑ skey	(Surrogate Key for ProductNumber)	□
☑ **ProductNumber**	Product.ProductNumber	□
☑ ProductName	Product.ProductName	□
☑ ProductTypeCode	ProductType.ProductTypeCode	□

For information about setting up dimension table columns, see “Add a Table to a Dimension Build” on page 208.

When you execute a dimension build, the surrogate key values are generated using the appropriate surrogate key start values defined in the associated template, and are delivered to the dimension table.

Determine the Maximum and Minimum Surrogate Values

You can use SQL to determine the current minimum or maximum values of a surrogate key. This may be useful before adding a level to a hierarchy.

Example

The dimension table named MyTable stores surrogate key values in the column named Key. To determine the maximum surrogate key value, use

```
SELECT MAX(Key)FROM MyTable;
```

Chapter 11. Reference Structures

Reference structures model the dimensions of a business and provide context for fact data. You deliver reference structures to target data marts, where they describe the dimensions of the fact data, and provide descriptive and navigational information for analysis tools. IBM Cognos Data Manager builds the reference structures from the structure data that you specify.

Data Manager has these types of reference structure:

- hierarchies
- auto-level hierarchies
- lookups

Reference structures, together with their associated templates, are collectively known as the dimensional framework.

The dimensional framework is crucial for coordinated data marts which have shared dimensions. These tables provide a common definition of business dimensions for the various fact tables in the warehouse.

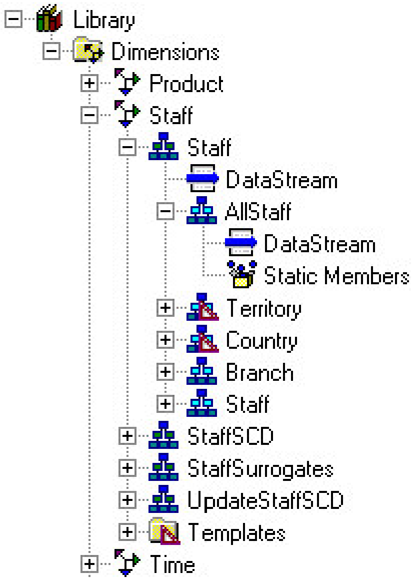
View Reference Structures

You can view reference structures in the Tree pane or the Visualization pane.

View Reference Structures in the Tree Pane

All reference structures and templates that relate to a particular business dimension appear within a reference dimension in the Tree pane.

IBM Cognos Data Manager represents reference dimensions, reference structures, and templates in the Dimensions folder as follows.



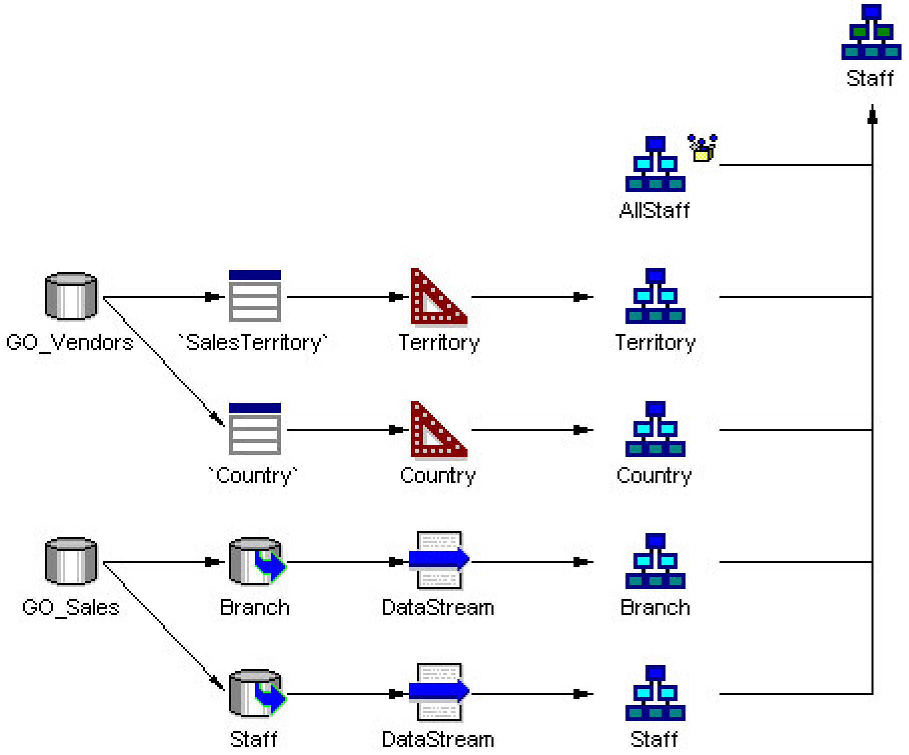
View Reference Structures in the Visualization Pane

When you click a reference structure in the Tree pane, IBM Cognos Data Manager shows a visual representation of the whole structure in the Visualization pane.

The information shown includes

- The database from which reference data is being accessed.
- The method used to access data (a template, DataStream or both).
- Any mapping between the DataStream and the reference structure. The following methods are used to indicate the mapping state:
  - No line means that a template is used for every level.
  - An arrow with a solid line means that a DataStream is used for every level.
  - An arrow with a dotted line means that data is accessed using a template but there is also a data source with the related mapping in place, but the data source mapping is not being used as the source.

This example shows a hierarchy.



For information on each of the symbols used in the Visualization pane, see Appendix F, “Buttons and Icons,” on page 497.

Use the Hierarchy Wizard

The Hierarchy wizard can help you to create hierarchies and auto-level hierarchies. The wizard interprets commonly found reference data representations and creates a hierarchy and levels, a means of populating dynamic members, and a static All member to represent the top of the hierarchy tree.

When you use the Hierarchy wizard, you

- select the hierarchy structure

- define the hierarchy
- select the source data
- optionally define an all level
- define the levels
- assign the remaining attributes

When you have created a hierarchy using the Hierarchy wizard, you can amend it, and add to it. For example, attributes such as ID, parent, and caption are defined using the wizard, but you may want to add other attributes that help to describe the data.

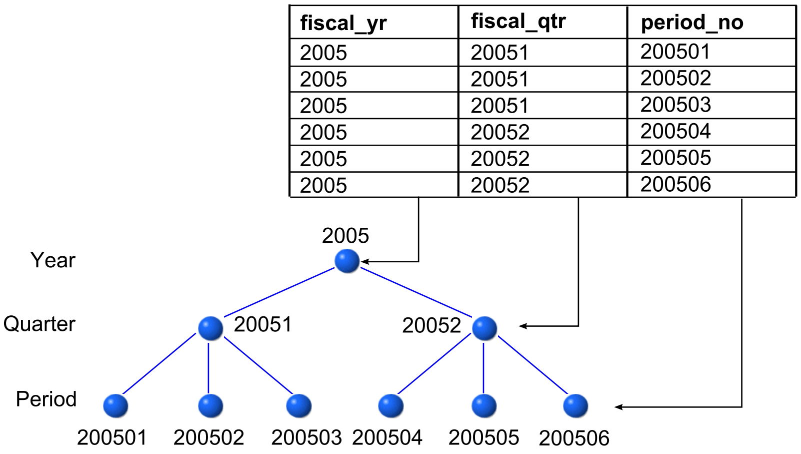
Note: You can use the Hierarchy wizard only if the structure data is accessible to the computer on which IBM Cognos Data Manager is operating. If the structure data cannot be accessed directly, you must create the hierarchy manually.

The format of the wizard depends upon the structure of the source data that you are using to create the hierarchy. You can choose from three options.

Create the Hierarchy from the Columns of One Table (Star Schema)

Use this option when the source data is a single table with columns representing levels in the hierarchy.

The following example shows a fiscal hierarchy based on the relationship between columns in the same data row. The source table includes columns named fiscal_yr, fiscal_qtr, and period_no. Each year contains quarters, and each quarter contains periods.

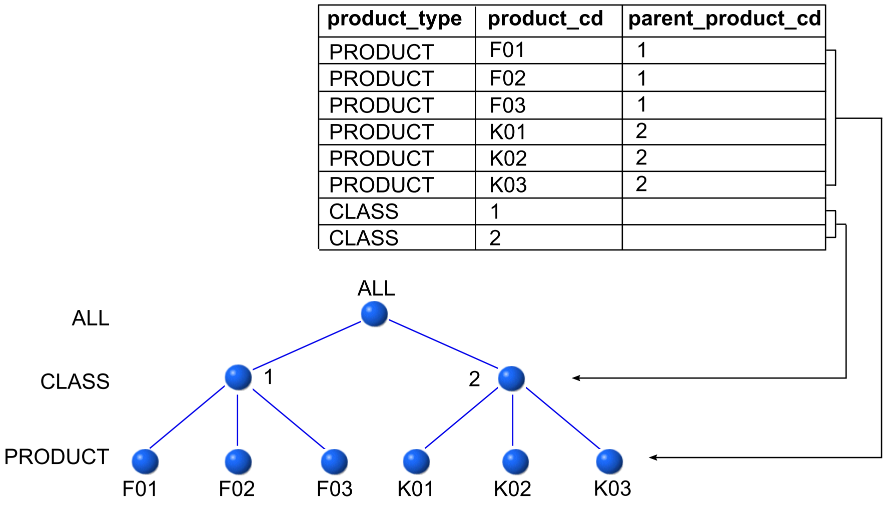


fiscal_yr	fiscal_qtr	period_no
2005	20051	200501
2005	20051	200502
2005	20051	200503
2005	20052	200504
2005	20052	200505
2005	20052	200506

Create the Hierarchy from the Rows of One Table

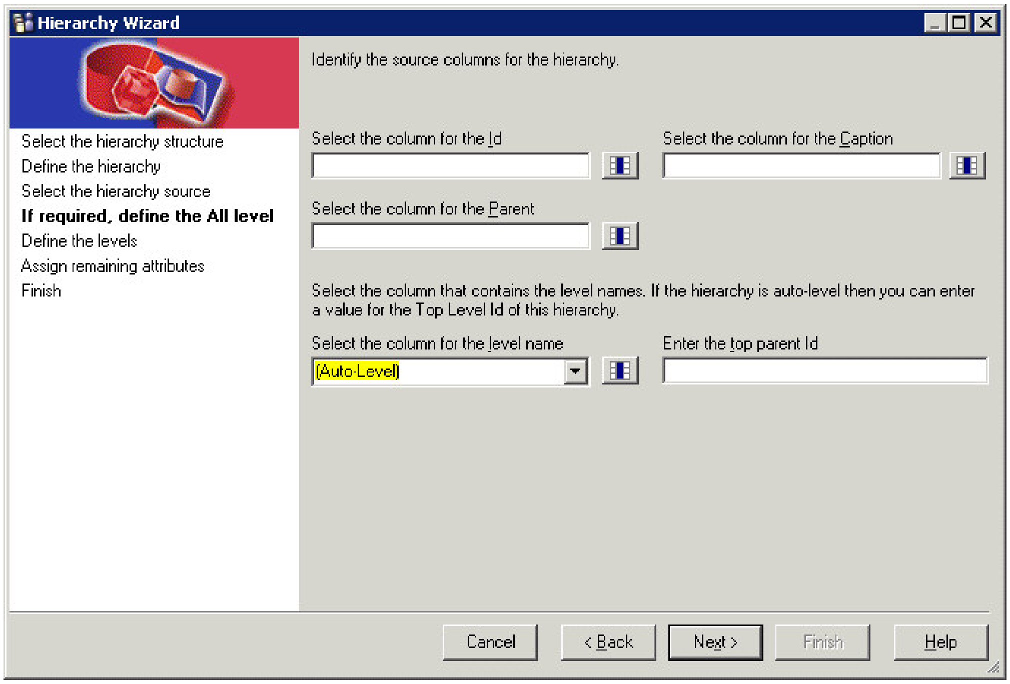
Use this option when the source data is a single table with sets of rows for each level in the hierarchy, with the rows related by a parent ID column.

The following example shows a product hierarchy based upon the relationships between rows of the same table. In relational terms, these are recursive relationships. The source table includes columns named product_type, product_cd, and parent_product_cd. Within the parent_product_cd column, each row refers to the product_cd value of its parent.



product_type	product_cd	parent_product_cd
PRODUCT	F01	1
PRODUCT	F02	1
PRODUCT	F03	1
PRODUCT	K01	2
PRODUCT	K02	2
PRODUCT	K03	2
CLASS	1	
CLASS	2	

Tip: To create an auto-level hierarchy using the Hierarchy wizard, you must choose this option. On the page in the wizard where you can define an All level, in the **Select the column for the level name** box, click **Auto-Level**.



Select the hierarchy structure
Define the hierarchy
Select the hierarchy source
If required, define the All level
Define the levels
Assign remaining attributes
Finish

Identify the source columns for the hierarchy.

Select the column for the Id

Select the column for the Caption

Select the column for the Parent

Select the column that contains the level names. If the hierarchy is auto-level then you can enter a value for the Top Level Id of this hierarchy.

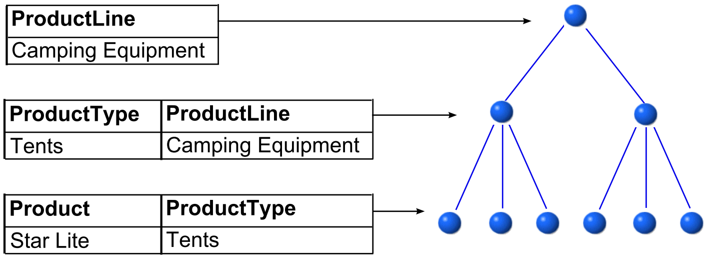
Select the column for the level name: (Auto-Level)

Enter the top parent Id

Create the Hierarchy from Multiple Tables (Snowflake Schema)

Use this option when the source data comes from multiple tables in which each table represents a single level in the hierarchy. Each row has a parent ID that relates to another row in its parent table.

In the following example, the ProductLine, ProductType, and Product levels each come from a different table.



ProductLine
Camping Equipment

ProductType	ProductLine
Tents	Camping Equipment

Product	ProductType
Star Lite	Tents

For examples of creating hierarchies using the Hierarchy wizard, see IBM Cognos Data Manager *Getting Started*. For general information about hierarchies, see "Hierarchies."

Procedure

1. From the **Tools** menu, click **Hierarchy Wizard** .
2. Follow the instructions.

Hierarchies

A hierarchy presents a particular view of a business dimension. It organizes the structure data into levels that represent parent-child relationships. Each hierarchy can have as many levels as you require. Each level contains a set of members.

IBM Cognos Data Manager can create hierarchies from any relational data, but typically from star schemas, snowflake schemas, and parent-child schemas.

You can create a hierarchy manually, or you can use the Hierarchy wizard.

To create a hierarchy manually, you must

- create a reference dimension
- create a hierarchy
- create hierarchy levels and define the attributes
- reposition the hierarchy levels
- acquire the structure data
- if the data is to be acquired using a DataStream, map the DataStream items to the level attributes
- define fostering
- define error handling

For information on using the Hierarchy wizard, see "Use the Hierarchy Wizard" on page 64.

Create a Hierarchy

You must create a hierarchy for a dimension if you want to organize the structure data into levels that represent parent-child relationships.

Procedure

1. In the **Dimensions** folder, click the reference dimension to which you want to add the hierarchy.
2. From the **Insert** menu, click **Library**, and then click **Hierarchy**.

   The **Hierarchy Properties** window appears.

3. Click the **General** tab.
4. Type a name for the new hierarchy and, if you require, a business name and description.
5. If you do not want IBM Cognos Data Manager to automatically provide foster parents for members without explicit parents, select the **Disable fostering** check box.

   For more information, see "Foster Parents" on page 78.
6. If the hierarchy is to support reference surrogate keys, select the **Reference Surrogate** check box.

   For more information, see "Reference Surrogate Keys" on page 62.
7. Click **OK**.

   You can now create the hierarchy levels.

Create Hierarchy Levels and Define the Attributes

Each hierarchy level consists of a DataStream, a set of members containing attributes, and optionally, static members.

Members at a level can relate to their parents at the next highest level and their children at the next lowest level.

Each member of a hierarchy level must have attributes. The available attributes are determined by associating a template with the level. A level must include an ID attribute to identify each member of the level. A level usually has an attribute to provide a caption (business name), and an attribute to identify the parent of each member of the level. A level may also include attributes to explicitly support surrogate keys or attribute change tracking. You can define as many attributes as you require.

A hierarchy level can acquire structure data through the hierarchy DataStream, a level DataStream, or a level template.

If the hierarchy is to acquire structure data through a DataStream rather than a template, you must still associate a template with the hierarchy to determine the level attributes. For more information, see "Create a Template" on page 53.

Procedure

1. Click the hierarchy to which you want to add a level.
2. From the **Insert** menu, click **Library**, and then click **Level**.

   The **Level Properties** window appears.
3. Click the **General** tab.
4. Type a name for the level and, if required, a business name and description.
5. If you want IBM Cognos Data Manager to sort members of the level numerically instead of alphanumerically when a build is executed, select the **Use numeric sort on ID** check box.
6. Click the **Attributes** tab.
7. In the **Template** box, click the template to use for this level.

   Tip: To create a template or edit an existing template, click **New** or **Edit** as appropriate.

   For information about templates, see Chapter 9, "Templates," on page 51.
8. Move the required attributes for the level from the **Available Attributes** pane to the **Chosen Attributes** pane.

9. In the **Chosen Attributes** pane, select the **ID** check box for the attribute that provides the ID for the level.
10. If the level is not at the top of the hierarchy, and you do not want to attach all members to the foster hierarchy, you must also define a parent attribute, by selecting the **Parent** check box for the appropriate attribute.
11. Optionally, define a caption attribute by selecting the **Caption** check box for the appropriate attribute.
12. If you want to use a temporary file-based cache for attributes in order to reduce the memory used, select the **Use file caching for attributes** box.
13. Click **OK**.

    After you define the level attributes, you must specify whether the level is to access source data using a template or a DataStream. If you specify a DataStream, you must map the DataStream items to the level attributes.

    For more information, see “Set Up Data Access” on page 75 and “Map the DataStream Items to Level Attributes” on page 76.

Reposition the Hierarchy Levels

When you add a level to a hierarchy, IBM Cognos Data Manager places it below the other levels for the hierarchy. The order in which the levels are shown in the Tree pane, must correctly reflect the parent-child relationships, in a top-down order.

Procedure

1. In the **Tree** pane, click the level to reposition.
2. From the **Actions** menu, click **Move Up** or **Move Down** as appropriate.

Create Recursive Levels

If the data for a hierarchy contains parent-child relationships, you can create a recursive hierarchy level which allows you to explicitly name the levels contained in the recursive relationships.

Note: Using recursive levels differs from using an auto-level hierarchy because it is level-based.

A recursive level can acquire structure data through a level DataStream or a level template, but not the hierarchy DataStream.

Note: If a hierarchy contains one or more recursive levels, the hierarchy DataStream becomes inactive.

When you create a recursive level, IBM Cognos Data Manager can use different apex detection methods to determine which nodes should be the top level parent nodes. By default, it uses nodes that have missing parents. If required, you can select a different apex detection method:

- member values

  Select this option if all the apex member values exist at the top of the required level, and you want to explicitly state them. Data Manager uses these values as the apex of each branch.
- parent values (default)

  Select this option to choose the apex members based on a set of explicit values of the parent attribute of the recursive level. By default, members with a NULL value for the parent attribute are selected.

- parent existence

  Select this option if all the apex member values exist at the level above the required level, but you do not know what those values are, so cannot explicitly state them.

  Note: You set up a separate hierarchy level at the level directly above the recursive level to use this option.

If you are interested only in a subset of the data, the apex does not have to be the top of the recursive level. By setting the apex to the top of the recursive level, if you have a very large hierarchy, Data Manager must search the entire hierarchy to work out the structure of the data. If you set the apex to a dependent level, you restrict the recursive structure to the nodes in the hierarchy from a specific point.

For example, you may have a recursive level named Continent. If you set the apex to Continent, all data that exists for each continent is included in the hierarchy. However, you may be only interested in data for a particular country. In this case, you would specify a particular country ID as the apex.

Note: You can also specify more than one ID attribute as the apex by typing a comma-separated list of IDs. For example, 'A', 'B', 'C'.

For more information on hierarchy transformation, see Chapter 12, “Constructing and Transforming Hierarchies,” on page 95.

Procedure

1. Click the hierarchy to which you want to add a recursive level.
2. From the **Insert** menu, click **Library**, and then click **Recursive Level**.

   The **Recursive Level Properties** window appears.
3. Click the **General** tab.
4. Type a name for the level that you want to appear at the top of the recursive level and, if required, a business name and description.
5. If you want Data Manager to sort members of the level numerically instead of alphanumerically when a build is executed, select the **Use numeric sort on ID** check box.
6. Click the **Sub-Levels** tab.
7. To create a dependent recursive level, click **Add**, and then type a name for the level.

   Tip: You can add a full business name and description to a dependent level. Select the level and then click **Edit Details**.
8. Click **OK**.
9. Repeat steps 7 and 8 to create further dependent levels.
   - The order of the dependent levels on this tab reflects the parent-child relationships, in a top-down order. To reposition a dependent level, click the level and then click **Move Up** or **Move Down** as appropriate.
   - To edit the name of a dependent level, click the required level, and then click **Edit**.
   - To delete a dependent level, click the required level, and then click **Delete**.
10. In the **Apex detection method** box, specify the detection method you want to use.

    If you select **Parent values** or **Member values**, type the values in the adjacent box.

11. Click the **Attributes** tab and define the level attributes for the recursive level.

    For more information, see "Create Hierarchy Levels and Define the Attributes" on page 68.

12. Click **OK**.

    After you define the level attributes, you must specify whether the recursive level is to access source data using a template or a DataStream. If you specify a DataStream, you must map the DataStream items to the level attributes.

    For more information, see "Set Up Data Access" on page 75 and "Map the DataStream Items to Level Attributes" on page 76.

    Note: You should perform mapping at the recursive level, not at the hierarchy level.

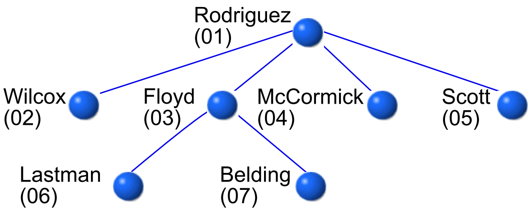
Auto-level Hierarchies

Auto-level hierarchies are hierarchies from parent-child schemas where the structure data table contains no hierarchy level information. IBM Cognos Data Manager ascertains the number of levels required and builds the hierarchy by following the parent-child relationships between the structure data rows.

For example, the following table contains no level information. Data Manager ascertains the levels required using the values in the Parent column.

ID	Name	Parent
01	Rodriquez	00
02	Wilcox	01
03	Floyd	01
04	McCormick	01
05	Scott	01
06	Lastman	03
07	Belding	03

The hierarchical representation of the table looks like this.



Note: You can also set up a parent-child structure as a hierarchy with recursive levels. For more information, see "Create Recursive Levels" on page 69.

Top Parent ID

To help IBM Cognos Data Manager ascertain the structure of an auto-level hierarchy, you specify the top parent ID used in the source data. The top parent ID is the value of the parent attribute of the top-level member, not the ID of the top level member.

For example, the top parent ID for this table is 00.

ID	Name	Parent
01	Rodriquez	00
02	Wilcox	01
03	Floyd	01

To create an auto-level hierarchy manually you must

__ • create a reference dimension

__ • create an empty auto-level hierarchy and define the attributes

__ • acquire structure data

__ • define fostering

__ • define error handling

Tip: To create an auto-level hierarchy using the Hierarchy wizard, you must choose the **Create a hierarchy from the rows of one table** option on the first page of the wizard. On the page where you define the source columns, in the **Select the column for the level name** box, click **Auto-Level**.

Create an Auto-level Hierarchy and Define the Attributes

You must use an auto-level hierarchy when the structure data contains no hierarchy level information.

An auto-level hierarchy consists of a single DataStream, a set of members containing attributes, and, optionally, static members.

Members relate to their parent members at the next highest level and their children members the next lowest level. Each member must have attributes. The available attributes are determined by associating a template with the auto-level hierarchy. An auto-level hierarchy must include at least an ID attribute to identify each member and a parent attribute so that IBM Cognos Data Manager can correctly determine the levels from the row in the data source. You can define as many attributes as you require.

An auto-level hierarchy can acquire structure data through its DataStream or template.

The levels for an auto-level hierarchy are generated when the build is executed. This means that there are no level DataStreams, so data must be accessed using the auto-level hierarchy DataStream, or template.

Note: If the hierarchy is to acquire structure data through a DataStream rather than a template, you must still associate a template with the hierarchy to determine the level attributes. For more information, see “Create a Template” on page 53.

Procedure

1. In the **Dimensions** folder, click the reference dimension to which you want to add the auto-level hierarchy.
2. From the **Insert** menu, click **Library**, and then click **Auto-Level Hierarchy**.

   The **Auto-Level Hierarchy Properties** window appears.
3. Click the **General** tab.
4. Type a name for the new auto-level hierarchy and, if you require, a business name and description.
5. If the structure data table uses a special value in the parent column to indicate the top member, type this value in the **Top parent ID** box. Otherwise, leave the box empty.

   The top parent ID is the value of the parent attribute of the top-level member, not the ID of the top level member. By default the top parent ID is null because the top member has no parents.
6. If you want Data Manager to sort the members of the hierarchy numerically, select the **Use numeric sort on ID** check box.

   By default, Data Manager sorts IDs alphanumerically.
7. If the auto-level hierarchy is to support reference surrogate keys, select the **Reference surrogates** check box.

   For more information, see “Reference Surrogate Keys” on page 62.
8. Click the **Attributes** tab.
9. In the **Template** box, click the template that defines the attributes for the auto-level hierarchy.

   Tip: To create a template, or edit an existing template, click **New** or **Edit** as appropriate. For more information, see Chapter 9, “Templates,” on page 51.
10. Move the required attributes from the **Available Attributes** pane to the **Chosen Attributes** pane.
11. In the **Chosen Attributes** pane, select the **ID** check box for the attribute that provides the ID for the auto-level hierarchy, and the **Parent** check box for the attribute that provides the parent. Optionally, select the **Caption** check box for the appropriate attribute.
12. If you want to use a temporary file-based cache for attributes in order to reduce the memory used, select the **Use file caching for attributes** box.
13. Click **OK**.

    You must now specify whether data is to be accessed using a template or a DataStream. For information see, “Set Up Data Access” on page 75.

    Note: You can balance an auto-level hierarchy, if required. For more information, see “Balance a Hierarchy” on page 96.

Lookups

A lookup is a simple, single-level reference structure with no parent-child relationships. Use a lookup when you have a set of reference members that you do not need to organize hierarchically.

Common uses for lookups are to

- check the data integrity of fact rows against a single level of a conformed dimension
- clean complex unstructured data from source systems before hierarchies can be properly determined
- lookup specific values from a reference structure based on a lookup key
- perform late arriving fact processing
- create tables to assist in data transformation that are not required for dimensional analysis

  For example, you create a currency conversion table to allow different currencies to be translated to a standard currency. The currency table will never by used for dimensional analysis, so a lookup may be more appropriate.

To create a lookup, you must

- create a reference dimension
- create an empty lookup and define the attributes
- acquire structure data
- define error handling

Create a Lookup and Define the Attributes

A lookup consists of a single DataStream, a set of members containing attributes, and, optionally, static members.

Each member must have attributes. The available attributes are determined by associating a template with the lookup. A lookup must include at least an ID attribute to identify each member. You can define as many attributes as you require.

Procedure

1. In the **Dimensions** folder, click the reference dimension to which you want to add the lookup.
2. From the **Insert** menu, click **Library**, and then click **Lookup**.

   The **Lookup Properties** window appears.
3. Click the **General** tab.
4. Type a name for the lookup and, if you require, a business name and description.
5. If you want IBM Cognos Data Manager to sort the IDs of this lookup numerically, select the **Use numeric sort on ID** check box.

   By default, Data Manager sorts IDs alphanumerically.
6. If the lookup is to support reference surrogate keys, select the **Reference surrogates** check box.

   For more information, see “Reference Surrogate Keys” on page 62.
7. Click the **Attributes** tab.

8. In the **Template** box, click the template that defines the attributes for the lookup.

   Tip: To create a template, or edit an existing template, click **New** or **Edit** as appropriate. For more information, see Chapter 9, “Templates,” on page 51.
9. Move the required attributes from the **Available Attributes** pane to the **Chosen Attributes** pane.
10. In the **Chosen Attributes** pane, select the ID check box for the attribute that provides the ID for the lookup. Optionally, select the **Caption** check box for the appropriate attribute.
11. If you want to use a temporary file-based cache for attributes in order to reduce the memory used, select the **Use file caching for attributes** box.
12. Click **OK**.

    You must now specify whether data is to be accessed using a template or a DataStream. For information, see “Set Up Data Access.”

Set Up Data Access

You can access data using a template or a DataStream.

Set up data access using a template

You can access data using a template to improve IBM Cognos Data Manager performance.

Template access is most commonly used

- Against data that does not require custom SQL.
- When a dimension table was built using a dimension build and template and then the same template is used to access the data.
- If you are accessing dimensional data that you delivered to the warehouse using a template, then when you next access this data you should again use the template. This allows IBM Cognos Data Manager to access all the columns it requires and know the intended behavior of each column without this work having to be done again.

A template generates SQL against a single table, and cannot be used against multiple tables. You cannot use a template to access data if literals have been defined on the DataStream.

For a hierarchy, you must specify a template to populate each level in the hierarchy, not the hierarchy as a whole.

For more information, see Chapter 9, “Templates,” on page 51.

Procedure

1. Click the hierarchy level, auto-level hierarchy, or lookup.
2. From the **Edit** menu, click **Properties** .

   The **Properties** window for the selected reference structure appears.
3. Click the **Data Access** tab.
4. Click **Use template for data access**.

   Note: The template that Data Manager will use is the one specified on the **Attributes**tab.

5. In the **Connection** box, click the connection for the database in which the structure data resides.

   Tip: To create a connection, click **New**. For more information, see "Create a Database Connection" on page 39.
6. In the **Table name** box, enter the name of the structure data table.
7. In the **Distributed lock table name** box, enter the name of the lock table to use when distributed handling of unmatched members is enabled for fact builds.

   For more information, see "Distributed Handling of Unmatched Members" on page 155.
8. If it is possible that the data is not distinct, select the **Use DISTINCT in template SELECT statement** check box.

   Selecting this check box adds a DISTINCT clause to the database SELECT statement that returns the reference structure elements. This ensures that duplicate rows are removed before the data is returned to Data Manager.

   When you access a data source using a template, all the levels defined by the hierarchy for which the template is related are read back. In this case, you should not select this check box as all the data will be retrieved anyway. In the same situation, you may want to access only higher levels of a hierarchy. In this case, selecting this check box results in fewer records being returned from the database containing the reference structure.
9. Click **OK**.

Set up data access using a DataStream

DataStream access uses the SQL declared in the data source and gives you direct control over the SQL. You can join tables, use calculated fields, and use ORDER BY or GROUP BY clauses. DataStream access is commonly used against data from operational systems.

You can specify that Data Manager is to access data using a DataStream for a complete hierarchy or for each hierarchy level.

Note: If a hierarchy contains one or more recursive levels, the hierarchy DataStream becomes inactive.

Procedure

1. Click the hierarchy, hierarchy level, auto-level hierarchy, or lookup.
2. From the **Edit** menu, click **Properties** .

   The **Properties** window for the selected reference structure appears.
3. Click the **Data Access** tab.
4. Click **Use DataStream for data access**.
5. Click **OK**.

   You must now map the DataStream items to the level attributes.

Map the DataStream Items to Level Attributes

To enable the flow of data from a data source to the level attributes, you must map the DataStream items to the level attributes.

Procedure

1. Click the hierarchy, hierarchy level, auto-level hierarchy, or lookup for which you want to perform mapping.

2. From the **Edit** menu, click **Mapping**.
3. In the **DataStream Item** column, click the item to which you want to map an attribute.
4. In the **Level Attributes** pane, click the attribute to map.

   Tip: To add an attribute, click **Add** and then click either

   - **Create a new attribute** and type a name for the attribute to create
   - **Subscribe to unused template attribute**, and click the required attribute
5. Click **Map**.

   Tip: You can also perform mapping by dragging the level attribute to the relevant position in the **Maps To** column.
6. Repeat steps 3 to 5 until you complete the required mappings.

   Note: You can map an attribute to more than one DataStream item.

   Tip: To clear the mapping, click the mapped item in the **Maps To** column, and click **Clear**.
7. Click **OK**.

Populate the Reference Structure

IBM Cognos Data Manager can use both static members and dynamic members to populate a reference structure.

Static Members

Static members

- are predefined
- often define an All member at the top of a hierarchy, which can act as a foster parent for any record that does not have a parent associated with it
- are useful for things that never change, such as months
- do not exist in the source data as they are defined and stored entirely in the catalog

Note: You must define the attributes of the reference structure before adding static members.

Dynamic Members

Dynamic members

- are populated using SQL statements to extract data from reference tables in a database
- change as the reference data changes
- should be used wherever possible

IBM Cognos Data Manager acquires dynamic structure data when a build is executed.

Dynamic structure data can reside in master tables, transaction tables, or a combination of both.

Master tables contain semi-static data that a transactional system uses to define valid transaction data. For example, a master table may exist to define a product range or to provide information about salespersons.

It is normal for master tables to provide structure data when these master tables describe all the associated entities, such as, all products or all salespersons.

When adequate master tables do not exist, transaction tables may provide the required structure data. This is particularly true when the transaction data contains the business keys and business names for the associated entities.

When they do not describe the complete range of the associated entities, you can supplement the master tables with transaction data.

Add a Static Member

This section describes how to add a static member.

Procedure

1. In the appropriate reference structure, click **Static Members**.
2. From the **Edit** menu, click **Properties** .

   The **Static Members** window appears. This window has a column that corresponds to the attributes you defined on the **Attributes** tab of the **Level Properties** window. In addition to the column for the required ID attribute. If you are adding a static member to a hierarchy or auto-level hierarchy, there is also a column indicating whether the static member is a foster parent. Note that each level can have one foster parent at most.

   For more information, see “Create Hierarchy Levels and Define the Attributes” on page 68, “Create an Auto-level Hierarchy and Define the Attributes” on page 72, “Create a Lookup and Define the Attributes” on page 74.
3. Click **Add**, and type a value for each attribute.
4. If the static member is to be a foster parent for orphaned members of the level below, select the check box in the **Foster** column and type a name for the foster parent.

   Note: You can either create a static member specifically to be the foster parent, or you can use an existing static member.

   Tip: To change the value for an attribute, click the attribute and then click **Edit**.
5. Click **OK**.

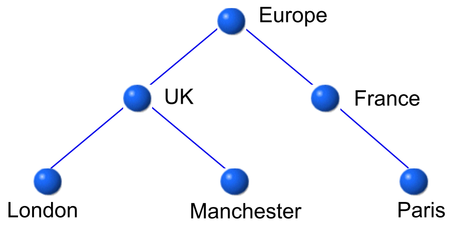
Foster Parents

In a typical hierarchy, each level contains a set of members. The members of each level, except the highest, are related to a parent member from the level above. If a member has no explicit parent, it cannot attach to the hierarchy. A foster parent is an artificially introduced member that acts as a parent for members that either have no defined parent, or whose defined parent cannot be found at the next highest level.

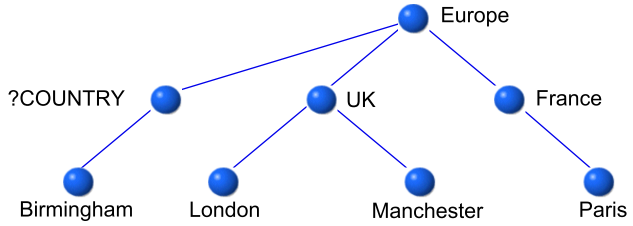
By default, IBM Cognos Data Manager creates automatic foster parents at all but the lowest level. At each level, you can create one static member to act as a foster parent for orphaned members from the level below. Using a static member as the foster parent overrides automatic fostering.

Fostering means that the hierarchy can be represented even when levels are missing for some records. This ensures that aggregations take into account all child members at a level when producing summary data.

For example, consider this hierarchy.



The data includes a member named Birmingham, for which no explicit parent or foster parent exists. Data Manager creates a foster parent using the name of the next-highest level, and prefixes it with a question mark. It adds a foster parent named ?COUNTRY, under which it places the member named Birmingham.



Fostering most commonly occurs when creating a dimension from operational source data. If possible, any fostered members should be examined and assigned to legitimate parents in the operational database.

You can change the default name, and you can disable fostering.

Specify a Name for a Foster Parent

You can specify a name for a foster parent rather than use the default name.

IBM Cognos Data Manager names foster parents by prefixing the level name with a question mark. For example, the foster parent that resides in the LOCATION level is named ?LOCATION. However, if the hierarchy is entirely numeric, Data Manager names the foster parent by prefixing the level number with a minus sign (-).

You specify a name for a foster parent by defining a static member as the foster parent and entering a suitable name. For more information, see “Add a Static Member” on page 78.

You cannot specify a name for an automatic foster parent.

Disable Fostering

By default, IBM Cognos Data Manager automatically provides a foster parent for each orphaned hierarchy member. However, you can disable automatic fostering.

Procedure

1. Click the hierarchy or auto-level hierarchy for which you want to disable fostering.
2. From the **Edit** menu, click **Properties** .
3. Click the **General** tab.

4. Select the **Disable fostering** check box.

   You must now instruct Data Manager to take appropriate action if it encounters orphaned members.
5. Click the **Features** tab.
6. In the **Fostering** box, click **Reject**, **Reject/Warn**, or **Error** as appropriate.

   For information, see “Foster Orphaned Members” on page 81.
7. Click **OK**.

Null ID Values

IBM Cognos Data Manager ignores ID columns that have a null value. This is because null IDs are usually an indication that the data is dirty, or that the wrong column has been specified as the ID.

Set Error Handling

You can specify the action that IBM Cognos Data Manager should take when it encounters particular situations. For each situation, you can choose one of the following options:

Option	Description
Accept	Accepts the item without warning (default)
Warn	Writes a message to the log file
Reject	Rejects the item without warning
Reject/Warn	Rejects the item and writes a message to the log file
Error	Writes an error message to the log file and stops executing the build at the current item

By default, when you select Warn or Reject/Warn, each occurrence of the selected situation is written to the log file. However, you can specify a limit to the number of messages written.

Data Manager always writes a message to indicate that a situation has occurred, even if individual occurrences are not written.

Procedure

1. Click the reference structure for which you want to set error handling.
2. From the **Edit** menu, click **Properties**.

   The **Properties** window for the selected reference structure appears.
3. Click the **Features** tab.
4. In the **Actions** box adjacent to the feature, click the required option.
5. If you set the feature to either **Warn** or **Reject/Warn**, and you want to restrict the number of warning messages issued, select the **Limit** check box for the feature. Then, in the adjacent box, type the maximum number of warnings.

When the number of warnings reaches this limit, Data Manager continues to take the required action. However, it suppresses further warning messages to the log file.

Note: If you type a limit of zero, no warnings are written to the log file, and you have no indication of why a record was rejected.

6. Click **OK**.

Foster Orphaned Members

If you disable fostering, you must configure IBM Cognos Data Manager to take appropriate action if it encounters orphaned hierarchy members.

For more information about fostering, see “Foster Parents” on page 78.

Option	Description
Accept	Accepts the orphaned members
Warn	Accepts the orphaned members, and writes a message to the log file stating that foster members were encountered
Reject	Rejects all orphaned members
Reject/Warn	Rejects all orphaned members, and writes a message to the log file stating that foster members were rejected
Error	Issues an error message and stops executing the build if an orphaned member is encountered

Applicable Reference Structures

Hierarchies and auto-level hierarchies

Multiple Parents

If IDs are not unique, more than one data source row may contribute to a particular hierarchy member. If those rows refer to different parents, the hierarchy member has multiple parents.

If multiple data source rows contain the same ID for a hierarchy level, each data source row may refer to a different parent. IBM Cognos Data Manager merges the non-unique data source rows to create a single member with a single set of member attributes. However, the single member may be delivered as multiple rows to the target dimension.

By default, one row is delivered for each combination of ID and parent ID. You can configure Data Manager to allow or disallow this depending on the option that you set for multiple parents.

If you allow multiple parents, each delivered row has the same business ID and the same surrogate key but different parent IDs.

Support for multiple parents requires each fact row to potentially reference multiple dimension rows, which is a many-to-many relationship.

Option	Description
Accept	Accepts multiple parents
Warn	Accepts multiple parents, and writes a message to the log file stating that multiple parents were encountered
Reject	Accepts the first parent that is encountered and rejects source data rows that refer to other parents
Reject/Warn	Accepts the first parent that is encountered, rejects source data rows that refer to other parents, and writes a message to the log file stating that multiple parents were rejected
Error	Issues an error message and stops executing the build

Applicable Reference Structures

Hierarchies and auto-level hierarchies

Non-Unique IDs

The same ID can never exist within a hierarchy level. However, the same ID can exist at different levels within the same hierarchy.

If multiple data source rows contain the same ID within a hierarchy level, IBM Cognos Data Manager merges the source data to give a single member for that ID. Multiple data source rows with the same ID within a single level may also point to different parents, which may lead to complications. For more information, see Multiple Parents.

Option	Description
Accept	Allows the same ID to exist at different hierarchy levels
Warn	Allows the same ID to exist at different hierarchy levels, and writes a message to the log file
Reject	Accepts the first ID at any hierarchy level and rejects the remainder
Reject/Warn	Accepts the first ID at any hierarchy level, rejects the remainder, and writes a message to the log file
Error	Issues an error message and stops executing the build if the same ID is encountered at any hierarchy level.

Applicable Reference Structures

Hierarchies and auto-level hierarchies

Duplicate Rows at a Level

If two source data rows have the same ID and the same parent, they are considered duplicates. For example, in this table, because both items with the ID value of p010 have the same parent, they are considered duplicate rows.

ID	Caption	Parent
p001	Camping equipment	
p010	Lights	p001
p010	Lanterns	p001

Valid configuration are options are shown below.

Option	Description
Accept	Assigns to each member the last caption that is accepted from the DataStream. In the example table, the caption Lantern is assigned to the member with ID p010.
Warn	Assigns to each member the last caption that is accepted from the DataStream, and writes a message to the log file. In the example table, the caption Lantern is assigned to the member with ID p010.
Reject	Accepts the first instance of each ID and parent pairing and rejects subsequent instances. The first applicable caption is assigned to each member. In the example table, the caption Lights is assigned to the member with ID p010.
Reject/Warn	Accepts the first instance of each ID and parent pairing, rejects subsequent instances, and writes a message to the log file. The first applicable caption is assigned to each member. In the example table, the caption Lights is assigned to the member with ID p010.
Error	Issues an error message and stops executing the build if duplicate data source rows are encountered.

Applicable Reference Structures

All reference structures

Null/Empty Captions

You can specify an attribute with the $CAPTION property for all reference features.

Option	Description
Accept	Accepts the null/empty captions
Warn	Accepts the null/empty captions and writes a message to the log file stating that they were encountered

Option	Description
Reject	Rejects any data that has a null or empty value
Reject/Warn	Rejects any data that has a null or empty value and writes a message to the log file
Error	Issues an error message and stops executing the build if null or empty values are encountered

Applicable Reference Structures

All reference structures

Non-Unique Captions

Non-unique captions may cause an internal error in some reporting tools. You can configure IBM Cognos Data Manager to take appropriate action if this occurs.

Option	Description
Accept	Accepts the non-unique captions
Warn	Accepts the non-unique captions and writes a message to the log file stating that they were encountered
Reject	Accepts the first source data row and rejects subsequent rows
Reject/Warn	Accepts the first source data row, rejects subsequent rows, and writes a message to the log file
Error	Issues an error message and stops processing the build if non-unique captions are encountered

Applicable Reference Structures

All reference structures

Unbalanced Hierarchy

IBM Cognos Data Manager cannot determine whether a hierarchy is unbalanced until it acquires all source data.

For information about unbalanced hierarchies, see “Hierarchy Types” on page 95.

Option	Description
Accept	Accepts that the hierarchy is unbalanced
Warn	Accepts that the hierarchy is unbalanced and writes a message to the log file
Reject	Not available

Option	Description
Reject/Warn	Not available
Error	Issues an error message and stops executing the build if a unbalanced hierarchy is encountered

Applicable Reference Structures

Hierarchies and auto-level hierarchies

Circular References

Circular references occur when a hierarchy member is its own ancestor. For example, if member A is the parent of member B, member B is the parent of member C, and member C is the parent of member A, then each member (A, B, and C) is its own ancestor.

IBM Cognos Data Manager detects circular references by examining each hierarchy member in sorted order and following the child-parent references. For each member, Data Manager stops following the child-parent links when it either reaches the ultimate ancestor or proves that a circular reference exists. If a circular reference exists, the last member that was examined is attached to the foster hierarchy.

For example, in the following table, BUF is the first item in the sort order.

ID	Caption	Parent
BUF	Buffalo	NY
EAST	Eastern Region	BUF
NY	New York	EAST

The following code is the result:

```
?AUTO - {Unknown AUTO}
    ?LEVEL2 - {Unknown LEVEL2}
        ?LEVEL3 - {Unknown LEVEL3}
    EAST - {Eastern Region}
        NY - {New York}
            BUF - {Buffalo}
```

In the following table, EAST is the first item in the sort order.

ID	Caption	Parent
XXX	Buffalo	NY
EAST	Eastern Region	XXX
NY	New York	EAST

The following code is the result:

```
?AUTO - {Unknown AUTO}
    ?LEVEL2 - {Unknown LEVEL2}
        ?LEVEL3 - {Unknown LEVEL3}
    NY - {New York}
        XXX - {Buffalo}
            EAST - {Eastern Region}
```

Valid configuration are options are shown below.

Option	Description
Accept	Accepts the circular references. When circular references are being accepted, you must not disable automatic fostering. This is because Data Manager attaches to the foster hierarchy those hierarchy branches that exhibit circular references.
Warn	Accepts the circular references, and writes a message to the log file stating that circular references were encountered. When circular references are being accepted, you must not disable automatic fostering. This is because Data Manager attaches to the foster hierarchy those hierarchy branches that exhibit circular references.
Reject	Not available because Data Manager cannot check for circular references until it acquires all structure data.
Reject/Warn	Not available because Data Manager cannot check for circular references until it acquires all structure data.
Error	Issues an error message and stops executing the build if circular references are encountered.

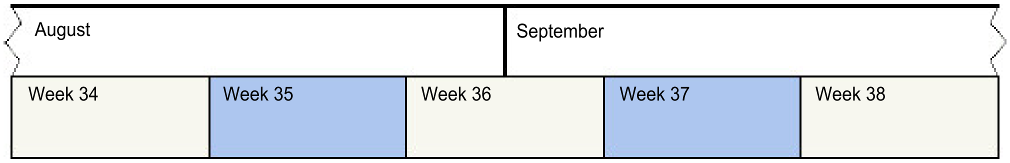
Applicable Reference Structures

Auto-level hierarchies

Create a Date Hierarchy

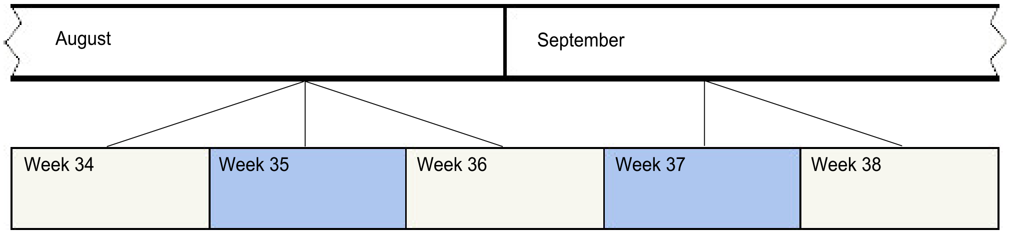
If the structure data does not contain a master table of dates, you can create a static hierarchy of dates. You can include levels for years, quarters, months, weeks, and days.

For most date hierarchy levels, the boundary of each member coincides with a boundary of one of its children. For example, the start and end of each quarter is at the start and end of a month. However, the beginning or end of a month may occur in the middle of a week.

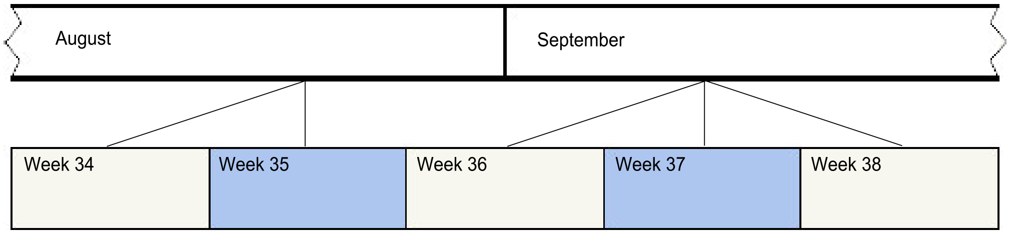


You can choose how to consolidate weeks that go across month boundaries:

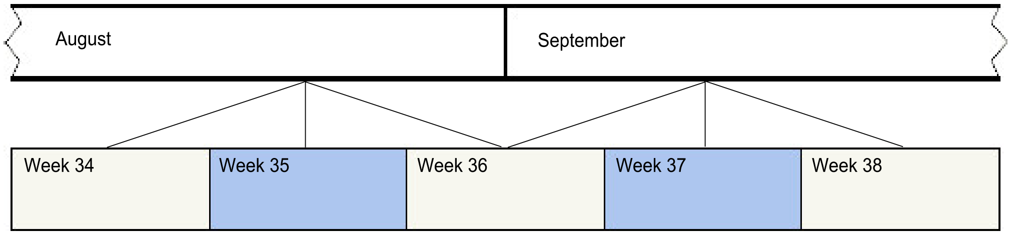
- Weeks roll to parent start



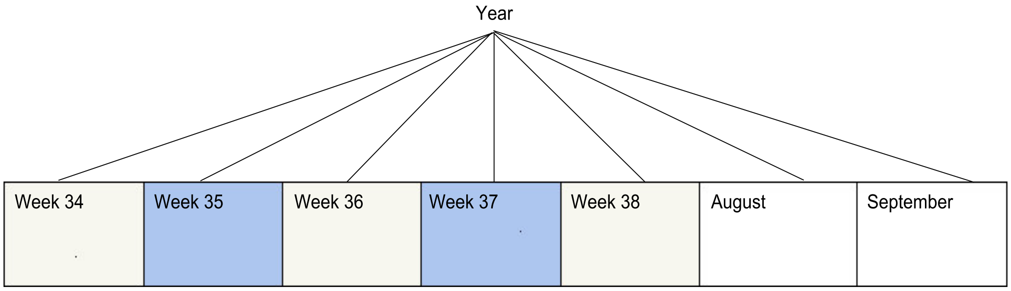
- Weeks roll to parent end



- Weeks roll to parent start and end



- Weeks on same level as parent



Before you create a date hierarchy, ensure that the reference dimension to which you want to add the hierarchy represents time.

Tip: To check, click the reference dimension, and from the **Edit** menu, click **Properties**. If necessary, select the **Dimension represents time** check box, and click **OK**.

Procedure

1. From the **Tools** menu, click **Date Hierarchy Wizard** 16 .
2. Follow the instructions.

Separate Reference Data and Fact Data

It is highly preferable to completely separate reference data and fact data. You can then use a reference structure based on reference data in a number of builds that process fact data. However, this is not always possible.

Sometimes the fact data contains some or all of the reference data for a dimension. For example, suppose you have fact data representing customer transactions, such as goods purchased. The customer dimension in the fact data need only hold the unique customer reference number. A customer hierarchy that holds the same reference numbers, but from reference data, could be associated with the dimension. However, suppose it is the practice for the reference data to contain customer telephone numbers, but for updates to the telephone numbers to appear in the fact data (perhaps because the customer provided the update as part of making their transaction). Now the fact data contains elements of reference data.

In building the hierarchy, IBM Cognos Data Manager could read both the reference data and the fact data. The disadvantage is that Data Manager reads the fact data twice, once in the context of the reference data, and once as fact data. The alternative is to use Data Manager functionality that maps elements of the fact data back to the reference data.

A more extreme example is a fact table that is denormalized. That is, it contains not only the member ID for the fact data, but also the member IDs for the other levels in the same dimension. For example, fact data represents monthly sales, and necessarily has a Month column, but also has Quarter and Year columns. The Month column should be treated as the dimension, and the Quarter and Year columns treated as attributes and probably marked *Never Output*. You should then map the Quarter and Year columns back in to the Quarter and Year levels of the hierarchy associated with the month dimension.

A common example of this is where the fact data relates to transactions for customers. The fact data would contain the customer reference as well as a transaction reference (that is, an order number). The reference data will only have information about previous orders that customers placed with new orders appearing in the fact data.

Note: There is little value in performing updates to the hierarchy in this way, unless the hierarchy itself is delivered as a dimension delivery, because the updates are lost at the end of the build. The dimension delivery should deliver data to the same table from which the hierarchy is built to so that the next time the hierarchy is used, it will contain all the updates made by previous builds.

When this technique is used to add new members to a hierarchy level, the domain size of the hierarchy increases throughout the build. This may mean that a domain size estimated for the hierarchy at the start of the build does not represent a good estimate and may cause the build to fail. In this case, a domain size should be set for the dimension in question.

If you map a transformation model element directly to a hierarchy, the transaction data contributes hierarchy members. Therefore, there can be no unmatched members of that hierarchy.

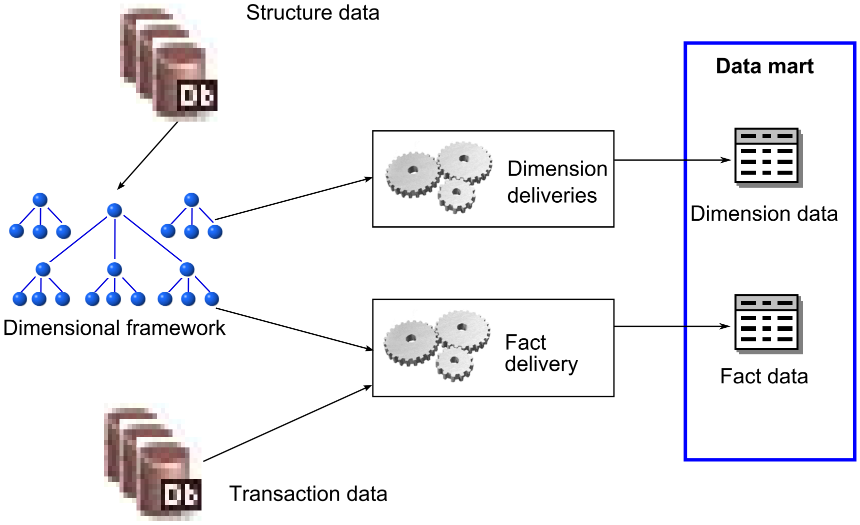
For more information, see “Example Scenarios” on page 89.

Example Scenarios

This section includes several examples of scenarios that you can use to help you understand how to set up your data.

Private Data Mart

A private data mart typically provides a snapshot view of the data. It has one or more private dimension tables and one or more fact tables. The following illustration shows the production of a private data mart.



IBM Cognos Data Manager first builds the hierarchy in memory. As it acquires the transaction data, Data Manager maps this to the hierarchy and transformation model, and performs the transformations that you specify. Finally, Data Manager delivers the fact and dimension data to the data mart.

Because the dimension data is not shared with other data marts, you can acquire structure and transaction data, transform that data, and deliver both the fact and dimension data in one object. Therefore, a fact build that has both fact deliveries and dimension deliveries is a good solution for this scenario. For private data marts that provide a snapshot view, it is usual to configure both fact deliveries and dimension deliveries to fully refresh the target tables.

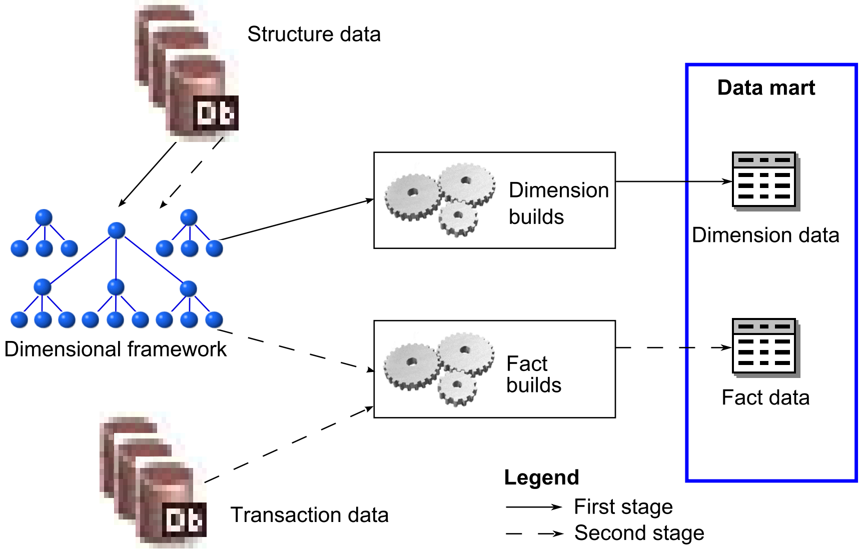
Private data marts require one hierarchy for each business dimension to be modelled.

Conformed Marts Without Surrogate Keys

Conformed marts differ from private marts in that many marts share the delivered dimension data. Therefore, the delivery of dimension data is not the responsibility of any one fact build. The following are strategies for delivery of dimension data:

- All the fact builds have identical dimension deliveries, and the dimension data is delivered to the data mart each time you execute a fact build.
- None of the fact builds have dimension deliveries. Instead, a dimension build delivers the dimension data, and the fact builds deliver only fact data. In this case, it is usual to combine the dimension build and fact builds in a JobStream.

The following illustration shows the latter case.

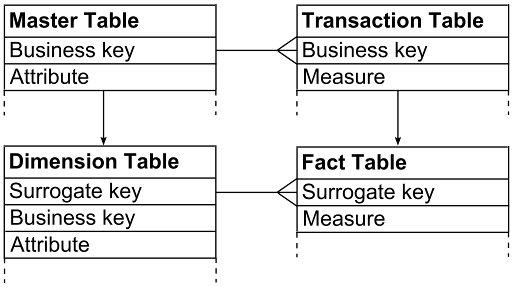


This approach populates the data mart in two stages. In one stage, IBM Cognos Data Manager creates each hierarchy in memory and delivers the corresponding dimension data through the corresponding dimension build. Each business dimension requires one hierarchy and one dimension build. In the second stage, Data Manager delivers the fact tables through the fact builds.

Because both the dimension builds and fact builds refer to the same hierarchies, only one hierarchy is required for each business dimension.

Conformed Marts With Surrogate Keys

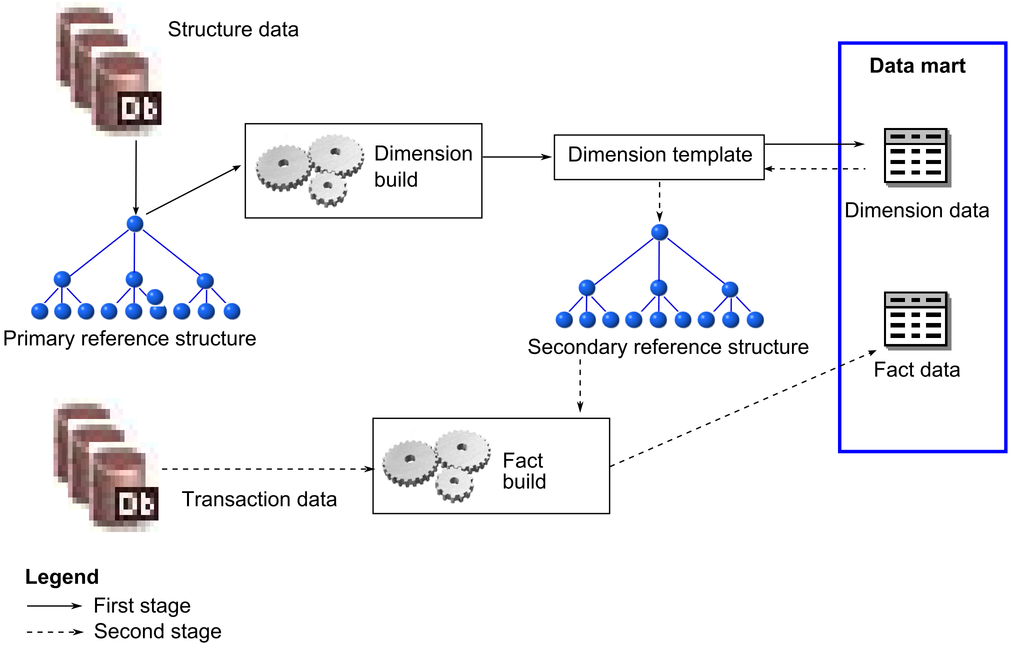
Surrogate keys are numeric values that replace business keys in the data mart. They exist only in the data mart. The following illustration shows the transformation of a master table and related transaction table from an online transaction processing (OLTP) system to dimension and fact tables in a data mart.



Note: The business key becomes an attribute of the dimension table, and the surrogate becomes the primary key. The fact table does not contain the business keys of the transaction system, and the surrogate key replaces the business key.

Because the OLTP system contains no information about the surrogate keys that reside in the data mart, this information is obtained from the dimension tables in the data mart. However, these tables must first be populated.

The requirement to populate the dimension tables before processing the fact data implies the following stages of execution.



This approach has these characteristics:

- Two reference structures are available for each dimension.
- A primary reference structure is constructed from the structure data. It does not contain the attributes to support surrogate keys.
- A dimension build delivers the primary reference structure to the data mart. A dimension table template defines the semantics of the dimension table and delivers the surrogate keys to the data mart.
- A secondary reference structure is constructed from the dimension table in the data mart. It uses the template used to deliver the dimension table. Therefore, IBM Cognos Data Manager can access the surrogate key values, and the fact build can deliver these values to the fact table.

It is common to combine the dimension builds and dependent fact builds in a JobStream. For information about JobStreams, see Chapter 19, “Managing Builds Using JobStreams,” on page 227.

Explore and Test a Reference Structure

You can use the reference explorer to explore and test a reference structure. When you view a reference structure in the reference explorer, IBM Cognos Data Manager acquires the structure data and organizes it according to the specification of the reference structure.

You can also use the Find menu to search the data to locate a member by name.

Procedure

1. From the **Tools** menu, click **Reference Explorer** .

   If you have made changes to the catalog, a message appears where you choose whether you want to save the changes.

   The **Reference Explorer** dialog box appears.

2. In the **Dimension** box, click the reference dimension in which the reference structure resides.
3. In the **Reference item** box, click the reference structure to explore.
4. If you want to limit the depth to which the reference explorer processes the reference structure, in the **Min depth** and **Max depth** boxes, click the minimum and maximum depth (level) respectively.
5. If you do not want the reference explorer to show foster parents that have no child members, select the **Remove unused foster parents** check box.
6. Click **OK**.

   If substitution variables are included in any of the SQL statements for the reference structure you are testing, the **Values for Expression** window appears listing each substitution variable.
7. In the **Value** column, type a suitable value for each variable.

   Tip: Click **Set Default Values** to enter the default value that you specified when you defined the substitution variable.

   For more information on default values, variable data type and scope, see Chapter 25, “Testing Expressions and Scripts,” on page 309.
8. Click **OK**.

   The **Reference Explorer** window appears showing the data structure for the selected reference structure. Note that this process may take some time, particularly if there are errors in the definitions. The status messages appear at the bottom of the window as the reference structure appears.

Results

To interrupt processing at any time, from the **View** menu, click **Interrupt**.

To refresh the information or to view another reference item, from the **View** menu, click **Refresh** . The **Reference Explorer** dialog box appears.

The Reference Explorer Window

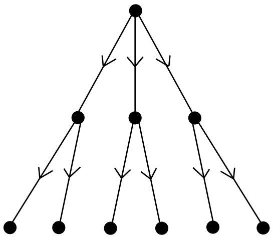
The information in the reference explorer is shown in two panes. The Elements pane shows either the hierarchical relationship view or the level elements view. The Attributes pane shows detail information.

The Elements Pane

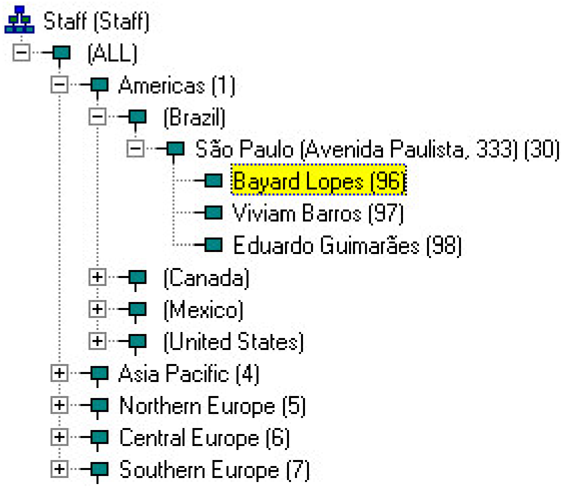
The Elements pane shows either the hierarchical relationship view or the level elements view.

The Hierarchy Relationship View:

The hierarchy relationships view in the Elements pane, shows the structure data as a true hierarchical structure with members linked to their parent.



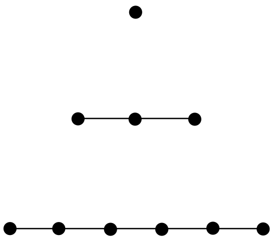
Tip: To select this view, from the **View** menu, click **Hierarchical Relationships** .



```
Staff (Staff)
  (ALL)
    Americas (1)
      (Brazil)
        São Paulo (Avenida Paulista, 333) (30)
          Bayard Lopes (96)
          Viviam Barros (97)
          Eduardo Guimarães (98)
      (Canada)
      (Mexico)
      (United States)
    Asia Pacific (4)
    Northern Europe (5)
    Central Europe (6)
    Southern Europe (7)
```

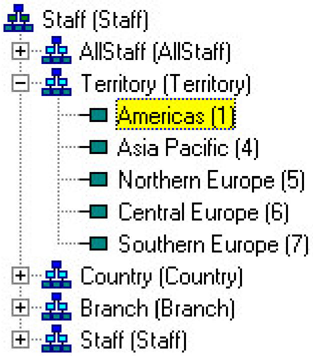
The Elements View:

The elements view shows the hierarchical members linked under the levels to which they belong.



Tip: To select this view, from the **View** menu, click **Elements At Each Level** .

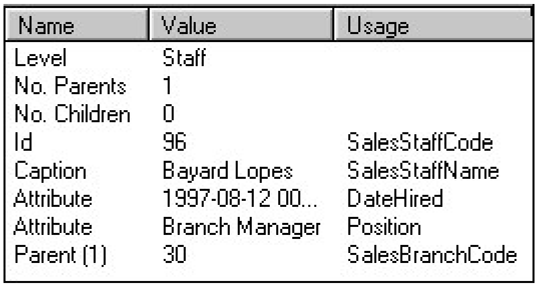
The elements view looks something like this:



- Staff (Staff)
  - AllStaff (AllStaff)
  - Territory (Territory)
    - Americas (1)
    - Asia Pacific (4)
    - Northern Europe (5)
    - Central Europe (6)
    - Southern Europe (7)
  - Country (Country)
  - Branch (Branch)
  - Staff (Staff)

The Attributes Pane

The Attributes pane shows the attributes that are relevant to the item you select in the Elements pane.



Name	Value	Usage
Level	Staff	
No. Parents	1	
No. Children	0	
Id	96	SalesStaffCode
Caption	Bayard Lopes	SalesStaffName
Attribute	1997-08-12 00...	DateHired
Attribute	Branch Manager	Position
Parent (1)	30	SalesBranchCode

Tip: You can hide the **Attributes** pane by clicking the Show/hide attributes pane button . To reveal the pane, click the button again.

Delete a Reference Structure

You cannot delete a reference structure if another component from the same catalog is using it.

For information about component dependencies, see “Locate Components and Their Dependencies” on page 15.

Procedure

1. Click the reference structure that you want to delete.
2. From the **Edit** menu, click **Delete**.
3. In the message box, click **Yes**.

Chapter 12. Constructing and Transforming Hierarchies

You can construct different types of hierarchy using IBM Cognos Data Manager, depending on the structure of the data that you are working with. You can use Data Manager to work with balanced hierarchies, unbalanced hierarchies, and ragged hierarchies.

It is more difficult to work with unbalanced and ragged hierarchies than balanced hierarchies because the number of levels can vary from one branch to another. There are some techniques you can use to construct and deliver data for these hierarchy types so that it is easy to work with the data using IBM Cognos Business Intelligence. You can

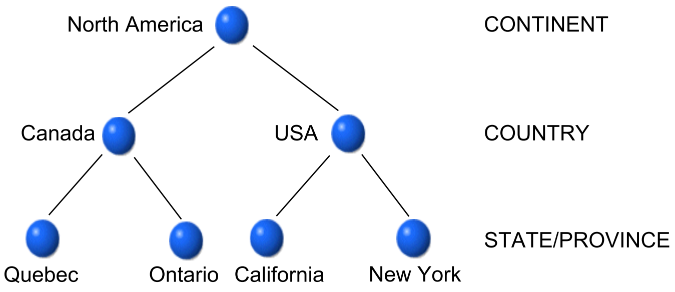
- Balance a hierarchy.
- Link a new level to the bottom of an unbalanced hierarchy.
- Flatten a hierarchy.

Hierarchy Types

Hierarchy types include balanced, unbalanced, and ragged hierarchies.

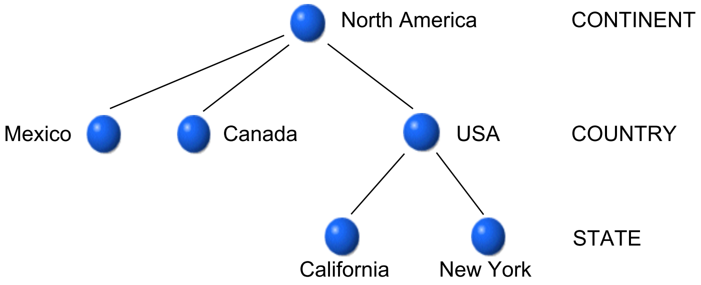
Balanced Hierarchies

In a balanced hierarchy, each branch of the tree contains the same number of levels, and the parent of every member comes from the level immediately above. This is an example of a balanced hierarchy.



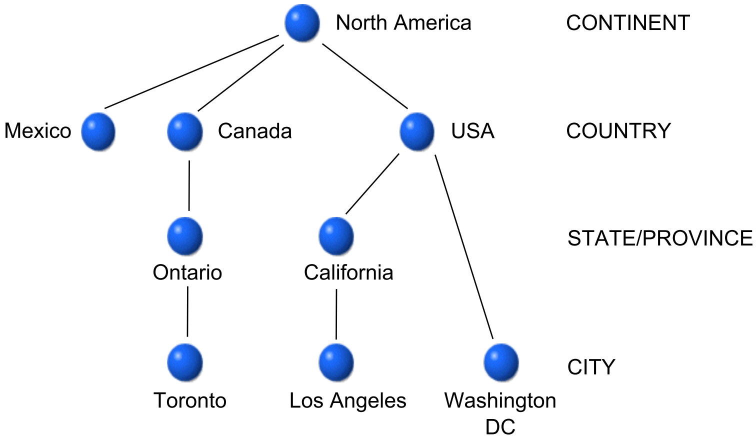
Unbalanced Hierarchies

In an unbalanced hierarchy, each branch of the tree can descend to a different level, so it has leaf nodes at more than one level. The parent of every member comes from the level immediately above. In the following example, state information is available only for the USA. Mexico and Canada are leaf nodes.



Ragged Hierarchies

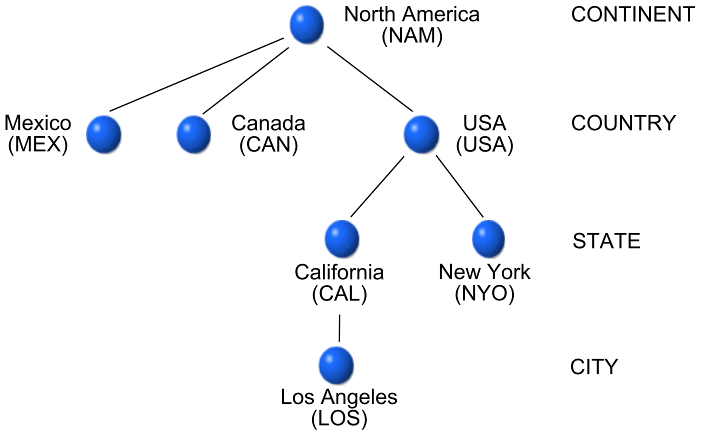
In a ragged hierarchy, the logical parent of at least one member does not come from the level immediately above, but a level higher up. In the following example, city information for Washington DC relates to the country level, not the state level.



Balance a Hierarchy

You can perform balancing on all hierarchy types. IBM Cognos Data Manager applies balancing to a hierarchy only after it has constructed it.

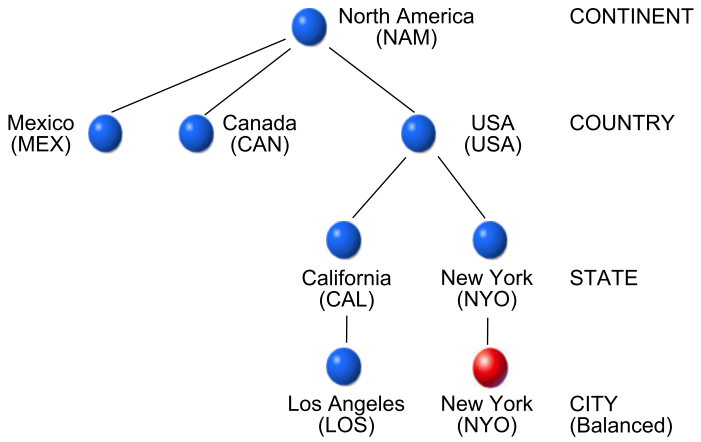
Consider this unbalanced hierarchy.



If you do not balance the hierarchy, only the member LOS is detected at the bottom of the hierarchy. If you want to accurately detect further valid members, you must balance the hierarchy. You can balance a hierarchy at one or more specified levels.

When you balance a hierarchy at a specified level, IBM Cognos Data Manager searches for all leaf nodes at the level directly above and duplicates them at the level you are balancing. Duplicate nodes share the ID and caption of the original node.

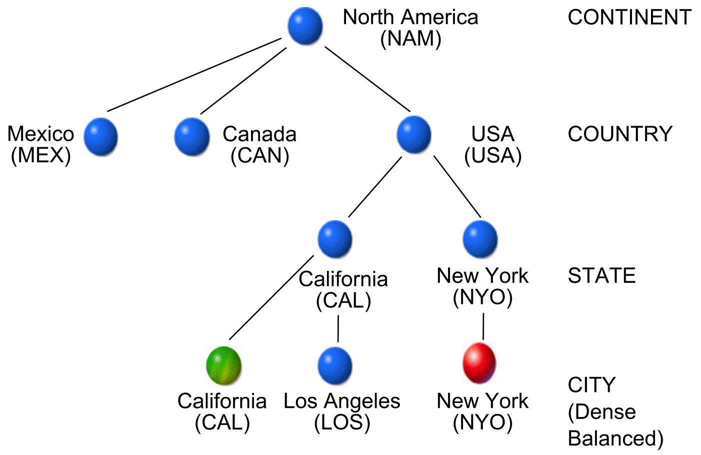
If you balance the previous hierarchy at the City level, it looks like this.



Note: Duplicate nodes, created by Data Manager, are shown in red.

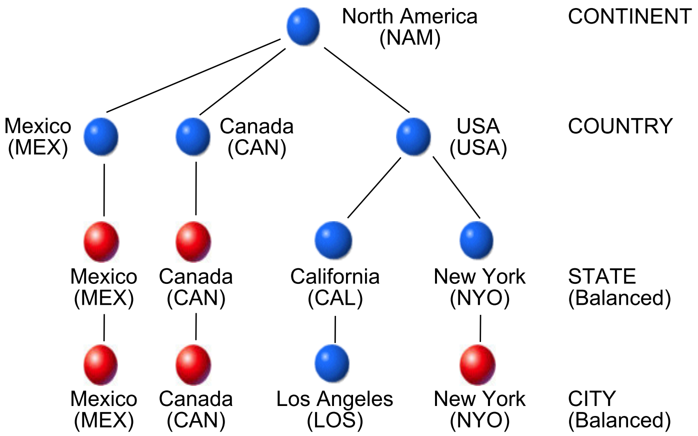
In addition to balancing, you can perform dense balancing. This differs from balancing in that Data Manager duplicates every node from the level directly above, not just leaf nodes.

If you perform dense balancing on the previous hierarchy at the City level, it looks like this.



Note: Duplicate nodes that Data Manager creates as a result of performing dense balancing are shown in green.

In the following example, the State and City levels have been balanced.



Balance an auto-level hierarchy

You can also balance an auto-level hierarchy, but this is not level-based balancing. IBM Cognos Data Manager balances the whole hierarchy, from the highest level to the lowest level it can detect.

Procedure

1. Click the hierarchy level you want to balance.
2. From the **Edit** menu, click **Properties**.

   The appropriate **Properties** window appears.
3. Click the **General** tab.
4. Select the **Balance level** check box.
5. If you also want to perform dense balancing, select the **Dense** option.
6. Click **OK**.

   You can now test the hierarchy by exploring it. For more information, see "Explore and Test a Reference Structure" on page 91.

Balance a hierarchy with recursive levels

You can perform balancing on all hierarchy types. IBM Cognos Data Manager applies balancing to a hierarchy only after it has constructed it.

Procedure

1. Click the hierarchy you want to balance.
2. From the **Edit** menu, click **Properties**.

   The **Recursive Level Properties** window appears.
3. If you want to balance the top level of the recursive hierarchy, click the **General** tab.
4. Select the **Balance level** check box. If you also want to perform dense balancing at the top level, select the **Dense** option.
5. If you want to balance a dependent of the recursive hierarchy, click the **Sub-Levels** tab.
6. Select the dependent level you want to balance and then click the **Edit Details**.

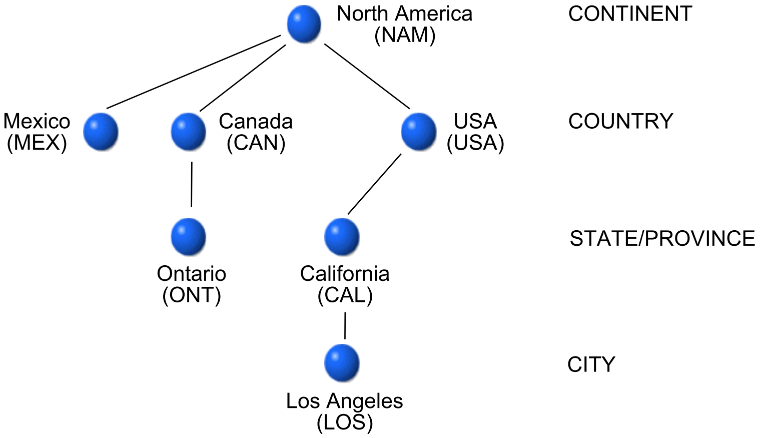
   The **More Details** dialog box appears.

7. Select the **Balance level** check box. If you also want to perform dense balancing at the top level, select the **Dense** option.
8. Click **OK** to close the **More Details** dialog box.
9. Click **OK**.

   You can now test the hierarchy by exploring it. For more information, see "Explore and Test a Reference Structure" on page 91.

Link a New Level to the Bottom of an Unbalanced Hierarchy

Sometimes, data from a separate source needs to be linked to the bottom of a hierarchy. By default, members are only added if the parent exists at the level immediately above. For example, suppose you have an unbalanced hierarchy like this.



If you were to add a Store level to this hierarchy, only stores linked directly to LOS are added by default. Suppose the data for the Store level is as follows.

Store ID	Store Name	Parent ID
101	Shop A	CAN
102	Shop B	ONT
103	Big Store A	LOS
104	Shop D	OTT
105	Big Store C	USA

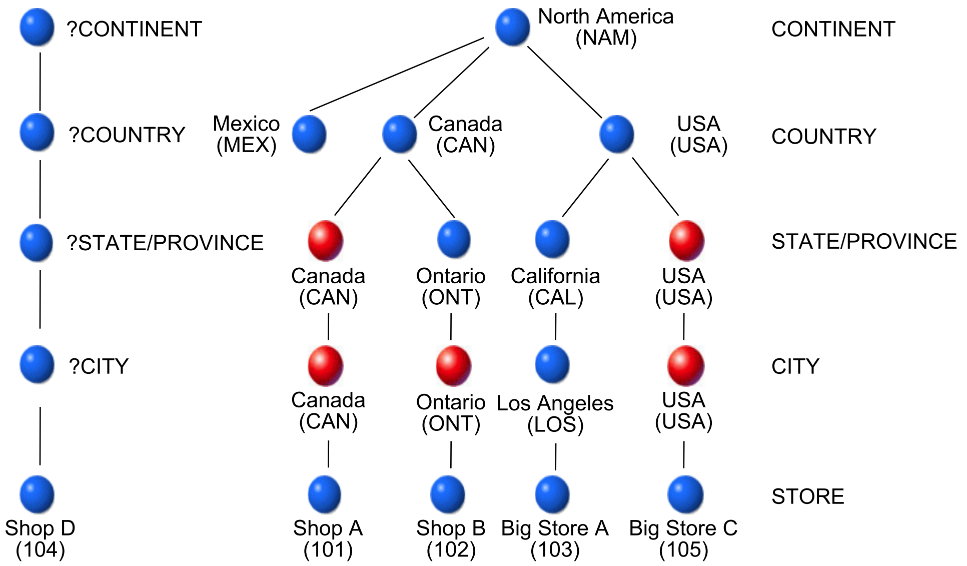
When you link the Store level, only member 103 is correctly added to the bottom of the hierarchy. All the other stores are fostered.

IBM Cognos Data Manager lets you resolve this problem by providing the option to search for parent members further up the hierarchy. This technique allows you transform a ragged hierarchy.

Note: You cannot use balancing to resolve this problem because balancing only occurs after hierarchy construction.

This option is named Link parent at next available level. For each node whose parent is found at a higher level of the hierarchy, Data Manager creates duplicate nodes at every intermediate level to link the data to the hierarchy. If any of the hierarchy data contains links to nodes that do not exist in the hierarchy, Data Manager still fosters it. Duplicate nodes share the ID and caption of the original nodes.

Using the example above, after the store data has been linked to the hierarchy, using the Link parent at next available level option, the hierarchy looks like this.



Duplicate nodes, created by Data Manager, are shown in red.

Data Manager does not create any duplicates of the node MEX because there are no stores linked to it.

Store 104 is still fostered because its parent node, OTT, does not exist in the hierarchy.

You can set up the Link parent at next available level option for all hierarchy types, except auto-level hierarchies.

Procedure

1. Click the hierarchy level you want to link.
2. From the **Edit** menu, click **Properties**.

   The **Level Properties** window appears.
3. Click the **General** tab.
4. Select the **Link parent at next available level** check box.
5. Click **OK**.

   You can now test the hierarchy by exploring it. For more information, see "Explore and Test a Reference Structure" on page 91.

Flatten a Hierarchy

If the source data for a hierarchy contains parent-child relationships, you can flatten the structure so it can be used for reporting. When you flatten a hierarchy, IBM Cognos Data Manager creates a set of records, with each record containing the details of a single branch of a tree.

You flatten a hierarchy by creating a hierarchy with recursive levels, which allows you to explicitly name the levels contained in the recursive relationships. The hierarchy can contain a mixture of normal and recursive levels as required, so the recursive levels can be positioned at any level within the hierarchy.

Note: You can also create an auto-level hierarchy using the source data and flatten it using staging builds, but this requires an intermediate step.

For more information about creating a hierarchy with recursive levels, see "Create Recursive Levels" on page 69.

Example

Suppose you have the following geographical data in a parent-child structure.

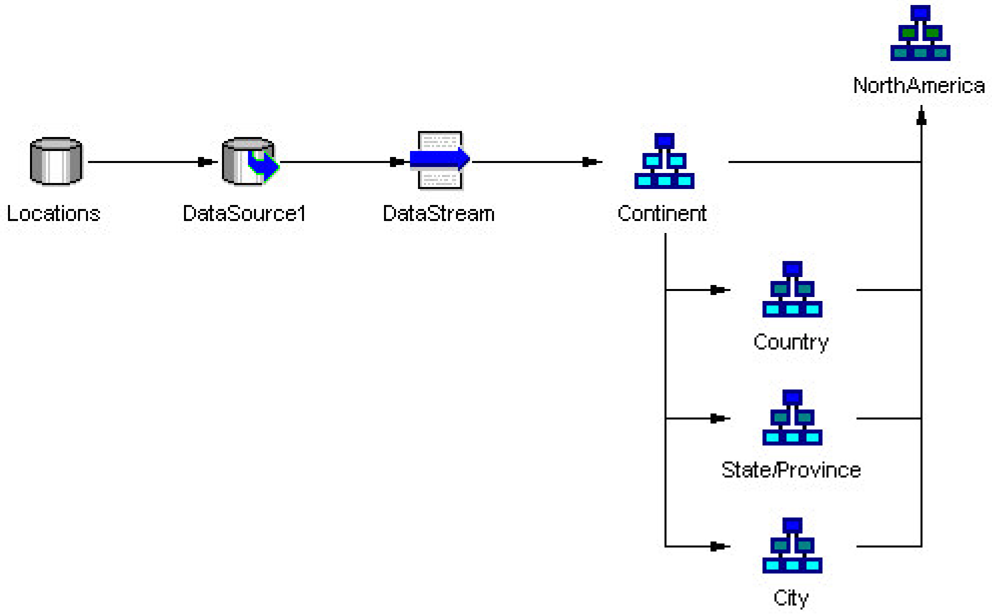
Location ID	**Parent ID**
NAM	<null>
MEX	NAM
CAN	NAM
USA	NAM
ONT	CAN
CAL	USA
LOS	CAL

You want to flatten the data so that it looks like this.

Continent ID	**Country ID**	**State**	**City**
NAM	<null>	<null>	<null>
NAM	USA	CAL	LOS
NAM	CAN	ONT	<null>
NAM	MEX	<null>	<null>

To do this, you must set up a hierarchy with recursive levels for Continent, Country, State/Province, and City.

The following diagram illustrates this structure set up as a hierarchy with recursive levels in IBM Cognos Data Manager.



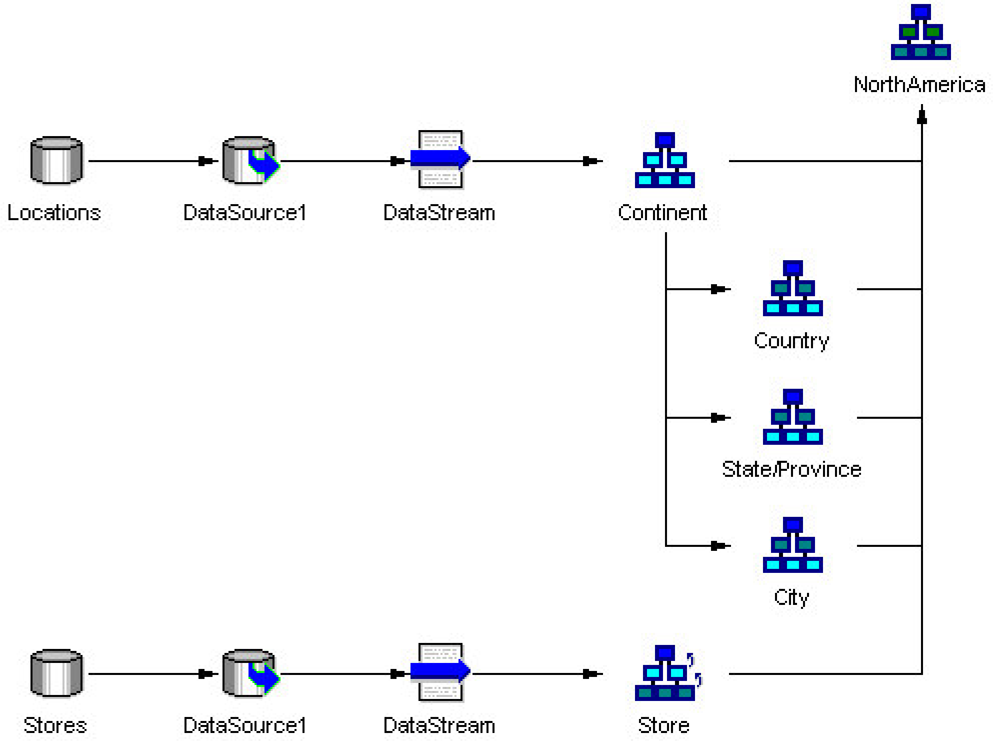
Now you want to link a new level of data, called Store, to this hierarchy. The Store data to be linked is as follows.

Store ID	Store Name	Parent ID
101	Shop A	CAN
102	Shop B	ONT
103	Big Store A	LOS
105	Big Store C	USA

Because the Store level data is separate from the geographical data, you need set up another hierarchy level, Store, at the bottom of the North America hierarchy.

If you do not select the Link parent at next available level option for the Store level, only store 103 would be linked to the hierarchy, because LOS, the parent of store 103, is the only hierarchy member that exists at the lowest level of the hierarchy. The remaining stores would be fostered.

To resolve this issue, ensure the Link parent at next available level option is selected for the Store level, as illustrated here.



If you deliver this hierarchy using a dimension build, the data might look like this.

Continent ID	Country ID	State/Province ID	City ID	Store ID
NAM	MEX	<null>	<null>	<null>
NAM	CAN	CAN	CAN	101
NAM	CAN	ONT	ONT	102
NAM	USA	CAL	LOS	103
NAM	USA	USA	USA	105

Note: If you exclude partially populated rows, Data Manager does not deliver the first record. For more information about excluding partially populated rows, see "Add a Table to a Dimension Build" on page 208.

Now suppose you want to deliver fact data with reference to this hierarchy. The fact data to be delivered is as follows.

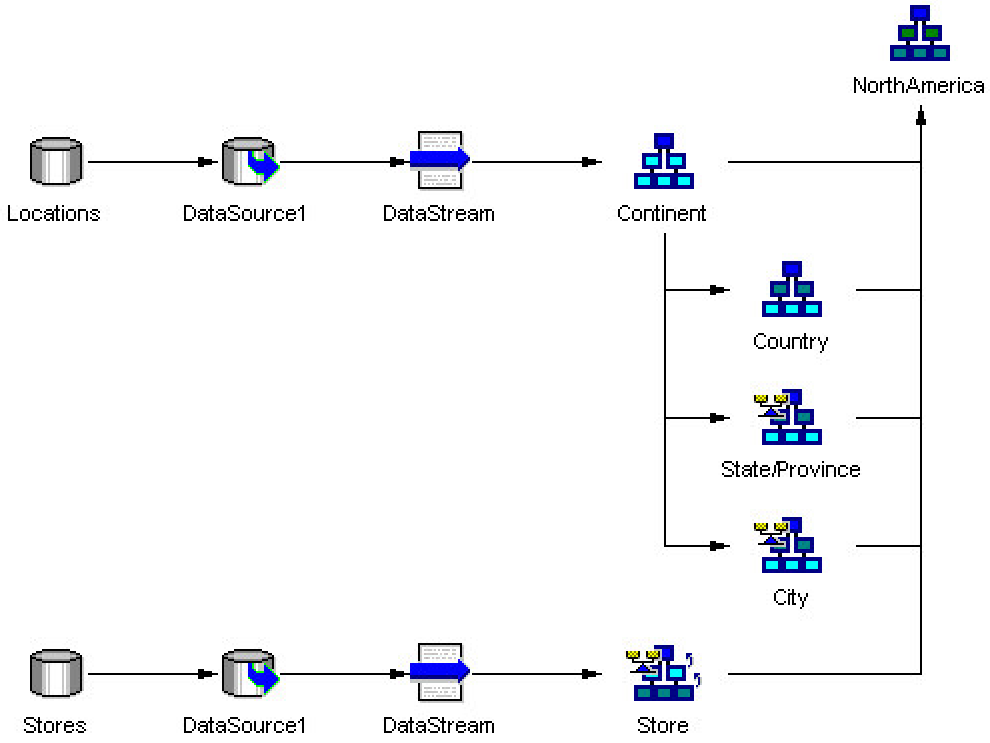
Product ID	Product Name	Location ID	Monthly Sales Total
1	Product A	101	1500
2	Product B	103	7000
3	Product C	MEX	100

If you want Data Manager to accurately deliver all three records, you must balance the hierarchy at the State/Province, City, and Store levels to ensure that MEX is duplicated at the bottom level of the hierarchy, as shown here.

Continent ID	Country ID	State/Province ID	City ID	Store ID
NAM	MEX	MEX	MEX	MEX
NAM	CAN	CAN	CAN	101
NAM	CAN	ONT	ONT	102
NAM	USA	CAL	LOS	103
NAM	USA	USA	USA	105

If you do not balance the hierarchy, only records 1 and 2 are delivered, because stores 101 and 103 exist at the bottom level of the hierarchy. Record 3 is rejected.

The following diagram illustrates this hierarchy in Data Manager, where the Balance level option is selected for the State/Province, City, and Store levels.



If you deliver the fact data with reference to this hierarchy, it looks like this.

Continent ID	Country ID	State ID	City ID	Store ID	Product ID	Monthly Sales Total
NAM	CAN	CAN	CAN	101	1	1500
NAM	USA	CAL	LOS	103	2	7000
NAM	MEX	MEX	MEX	MEX	3	100

Chapter 13. Fact Builds

A fact build acquires transactional data from the data source, transforms the data with reference to the dimensional framework, and delivers the data to target data marts. Optionally, a fact build can also deliver dimension data.

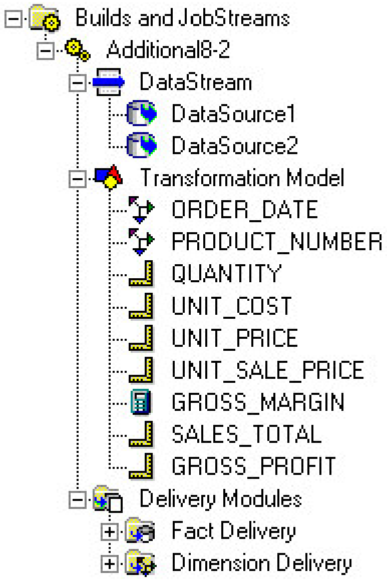
Note: Data can also be transformed independently of the dimensional framework. This might be required when setting up one or more staging builds to cleanse or standardize data sources prior to merging them in a final fact build.

View Fact Builds

You can view fact builds in the Tree pane or the Visualization pane.

View Fact Builds in the Tree Pane

A fact build consists of several types of objects. These objects are shown grouped together in the Tree pane. Each fact build in the Builds and JobStreams folder is represented as follows.



```
⊟ Builds and JobStreams
   ⊟ Additional8-2
      ⊟ DataStream
            DataSource1
            DataSource2
      ⊟ Transformation Model
            ORDER_DATE
            PRODUCT_NUMBER
            QUANTITY
            UNIT_COST
            UNIT_PRICE
            UNIT_SALE_PRICE
            GROSS_MARGIN
            SALES_TOTAL
            GROSS_PROFIT
      ⊟ Delivery Modules
         ⊞ Fact Delivery
         ⊞ Dimension Delivery
```

Tip: You can view just the fact builds contained in a catalog by clicking the Fact Builds tab at the bottom of the Tree pane.

View Fact Builds in the Visualization Pane

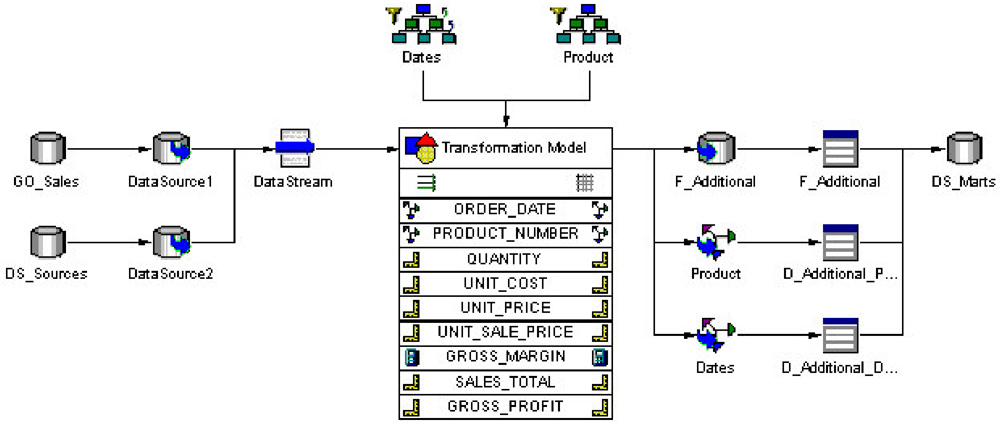
When you click a fact build in the Tree pane, four different views of the selected fact build are shown, each on a different tab in the Visualization pane.

For information about each of the symbols used in the Visualization pane, see Appendix F, "Buttons and Icons," on page 497.

The <Fact Build Name> Tab

The <fact build name> tab visually represents the whole fact build. The name of this tab is the name of the selected fact build.

The tab shows the processing that occurs from data acquisition to delivery of the transformed data. At the top are the reference structures associated with the fact build. Below the reference structures, from left to right, are the processes in sequential order.

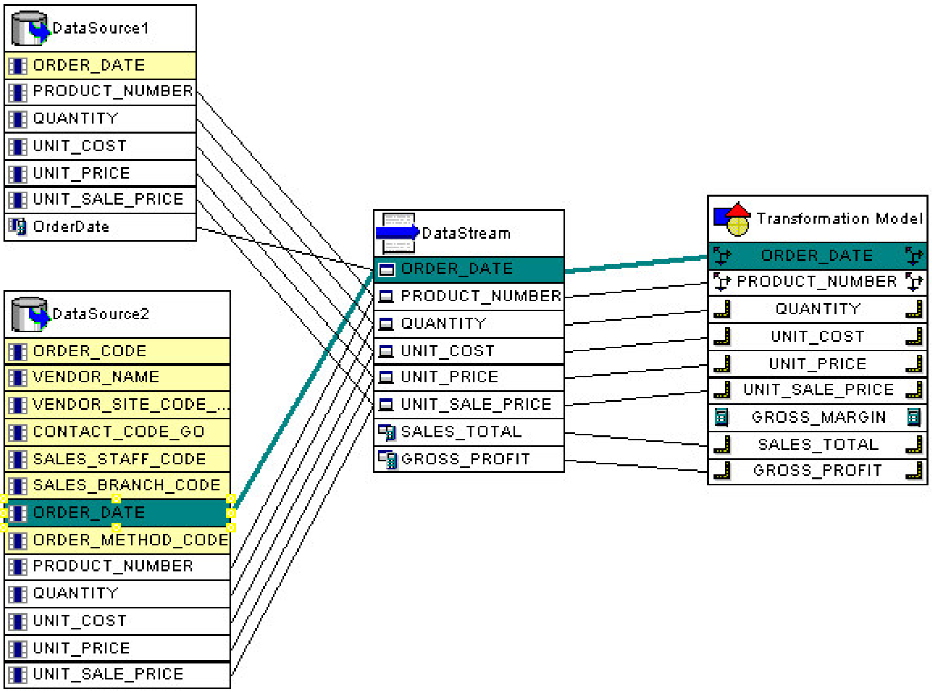


The Mapping Tab

The Mapping tab visually represents the mapping between the data source columns on the left, the DataStream items in the middle, and the transformation model on the right.

When you click a data source column, DataStream item, or a transformation model element, the mapping from the data source through to the transformation model element is highlighted.

Unmapped items are indicated by a change of background color.

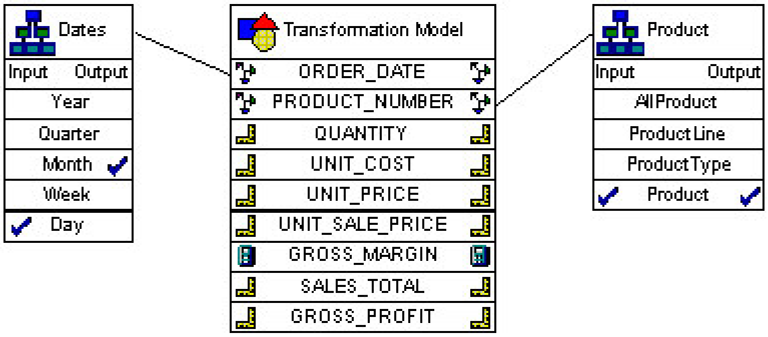


The Transformation Model Tab

The Transformation Model tab shows

- each element in the transformation model
- the reference structure to which each dimension element relates
- the hierarchy levels at which IBM Cognos Data Manager is to access (input) data
- the hierarchy levels at which Data Manager is to deliver (output) data

For more information about the transformation model, see Chapter 15, “Transformation Models,” on page 137.



Dates
Input Output
Year
Quarter
Month ✓
Week
✓ Day

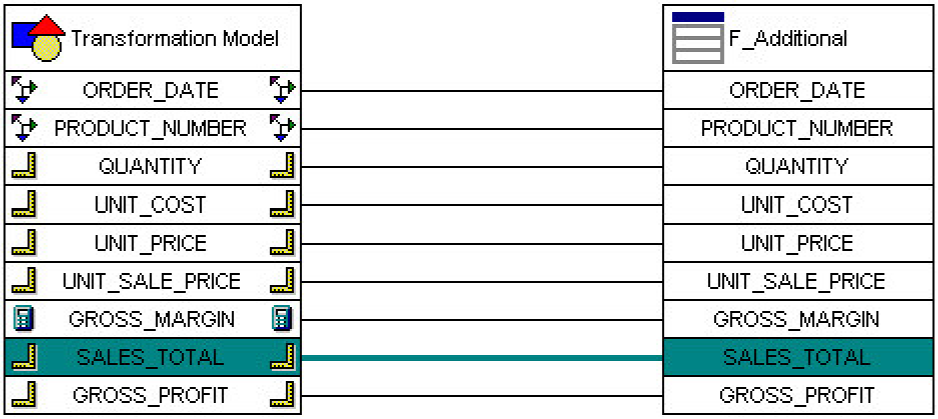
Transformation Model
ORDER_DATE
PRODUCT_NUMBER
QUANTITY
UNIT_COST
UNIT_PRICE
UNIT_SALE_PRICE
GROSS_MARGIN
SALES_TOTAL
GROSS_PROFIT

Product
Input Output
All Product
Product Line
Product Type
✓ Product ✓

The Fact Delivery Tab

The Fact Delivery tab shows all the elements in the transformation model and, for each element, the column name to which each element is to be delivered.

When you click an item, the mapping between the transformation model element and the fact delivery column is highlighted. Unmapped items are indicated by a change of background color.



Transformation Model	F_Additional
ORDER_DATE	ORDER_DATE
PRODUCT_NUMBER	PRODUCT_NUMBER
QUANTITY	QUANTITY
UNIT_COST	UNIT_COST
UNIT_PRICE	UNIT_PRICE
UNIT_SALE_PRICE	UNIT_SALE_PRICE
GROSS_MARGIN	GROSS_MARGIN
SALES_TOTAL	SALES_TOTAL
GROSS_PROFIT	GROSS_PROFIT

For more information about fact deliveries, see Chapter 16, “Delivering Fact Data Using a Fact Build,” on page 179.

The Fact Build Wizard

You can use the Fact Build wizard to help you to create a fact build that delivers the type of data that you specify.

If you want to deliver multiple tables to different databases, or partition data to deliver to different tables, you cannot use the Fact Build wizard, you must add the deliveries manually.

For information, see Chapter 16, “Delivering Fact Data Using a Fact Build,” on page 179.

Each of these standard fact build types provide default settings for various deliveries:

- IBM Cognos BI Mart (Star)
- IBM Cognos BI Mart (Snowflake)
- Relational Datamart
- Data Transfer

Where possible, the Fact Build wizard delivers surrogate keys for all the standard types of fact builds except Data Transfer, and performs no aggregation. It delivers the fact data to a single table. However, if you decide to perform aggregation along any dimension, the Fact Build wizard creates a fact delivery for each combination of hierarchy levels. For example, there are three dimensions (A, B, and C) and you decide to perform aggregation on dimension A, which has three levels (1, 2, and 3), then the Fact Build wizard creates fact deliveries for A.1*B*C, A.2*B*C, and A.3*B*C.

IBM Cognos BI Mart (Star) and IBM Cognos BI Mart (Snowflake)

An IBM Cognos BI Mart (Star) fact build delivers dimension data to tables that represent a star schema. That is, one dimension table is created for each dimension in the fact build.

An IBM Cognos BI Mart (Snowflake) fact build delivers dimension data to tables that represent a snowflake schema. That is, one dimension table is created for each level of each dimension in the fact build.

By default, an IBM Cognos BI Mart delivers fact data only at the lowest level of each hierarchy. If you choose to deliver aggregations, the Fact Build wizard configures the fact build to deliver them to satellite tables.

To use the Fact Build wizard to track attribute changes, you should select IBM Cognos BI Mart (Star). Alternatively, you can select IBM Cognos BI Mart (Snowflake), and then manually add the details of attributes to be tracked.

Relational Datamart

A relational datamart fact build provides the default settings for a fact build that delivers data into a relational database.

Use this type of fact build to deliver dimension data to tables that represent a star schema. That is, create one dimension table for each dimension in the fact build.

Data Transfer

A data transfer fact build provides the default settings for copying data from one DBMS to a single fact table in another DBMS.

By default, the Fact Build wizard creates all transformation model elements as attributes. It does not create dimension data.

Summary of Fact Build Types in the Fact Build Wizard

This table summarizes the default settings for each build type.

Default	Star	Snow	Rel	Data
Surrogate keys	Yes	Yes	Yes	No
Aggregation	No	No	No	No
Elements to suit data type	Yes	Yes	Yes	No
All elements as attributes	No	No	No	Yes
Star schema	Yes	No	Yes	No
Snowflake schema	No	Yes	No	No
No schema	No	No	No	Yes

Use the Fact Build Wizard

When you use the Fact Build wizard, you

- define the purpose of the build, which includes selecting the type of build to create and the connection into which the build is to deliver data
- create the DataStream
- assign the transformation model element types
- define the dimensions by associating each dimension element with a reference structure and specifying whether aggregation is to be performed, and define the measures and attributes if present
- define the fact delivery and its properties
- define the dimension delivery and its properties

When you create a fact build using the Fact Build wizard, you can amend it and add to it as you require.

Procedure

1. From the **Tools** menu, click **Fact Build Wizard** .
2. Follow the instructions.

Automatic Source Table Joins Using the Fact Build Wizard

When you select more than one source table for a fact build, IBM Cognos Data Manager attempts to determine the join conditions for those tables. However, you can edit the SQL statement that the Fact Build wizard generates.

If Data Manager can detect the primary key of a table, it searches in the other tables for the name of each primary key column, and creates join conditions where it finds a match.

Where Data Manager cannot detect a primary key, it searches the tables for any match between the names of columns, and bases join conditions on any matches found.

For example, if a Customer table has a primary key named Customer.CustID and an Order table has a column named CustID, Data Manager detects the primary key. It matches the name of the primary key column with the column named CustID from the Order table, and includes Customer.CustID = Order.CustID in the WHERE clause of the SQL for the data source.

Create a Fact Build Manually

To create a fact build manually, you must

- add a fact build
- define the DataStream and map data source columns
- define the transformation model and map the DataStream
- add the deliveries
- execute the fact build

Add a Fact Build

This section describes how to add a fact build.

Procedure

1. Click the **Builds and JobStreams** folder.
2. From the **Insert** menu, click **Build**, and then click **Fact Build**.
3. Click the **General** tab.
4. In the **Name** box, type a unique name for the fact build. If you require, type a business name and description.
5. Click **OK**.

Results

This is the minimum information necessary to create a fact build. You are now ready to define the DataStream, the transformation model, and the deliveries.

For more information, see Chapter 14, “DataStreams,” on page 115, Chapter 15, “Transformation Models,” on page 137, Chapter 16, “Delivering Fact Data Using a Fact Build,” on page 179, Chapter 17, “Delivering Dimension Data Using a Fact Build,” on page 199.

Define the DataStream

After you create a fact build, you define the DataStream, which includes

- defining data sources so that IBM Cognos Data Manager knows from where to acquire the data
- creating DataStream items for each acquired data source column
- mapping data source columns to the DataStream items

For more information, see Chapter 14, “DataStreams,” on page 115.

Define the Transformation Model

After you specify how the source data is to be acquired, you define the transformation model. The transformation model allows you to manipulate the data in a number of ways, including merging data from different sources, and partitioning and aggregating data.

Defining the transformation model includes

- creating transformation model elements for transformed data
- mapping DataStream items to the transformation model elements to enable the flow of data

For more information, see Chapter 15, “Transformation Models,” on page 137.

Add Deliveries to the Fact Build

You must specify the data repositories to which the data is to be delivered.

For more information, see Chapter 16, “Delivering Fact Data Using a Fact Build,” on page 179 and Chapter 17, “Delivering Dimension Data Using a Fact Build,” on page 199.

Execute the Fact Build

After you define the fact build, you execute it to acquire, transform, and deliver the data.

For more information, see Chapter 21, “Executing Builds and JobStreams,” on page 245.

Rename a Fact Build

When you rename a fact build, it is treated as a new component for logging and auditing purposes. However, the components of the fact build are not renamed. Therefore, renaming the fact build does not change the names of components, target tables, and so on, that the Fact Build wizard derived from the original build name.

When renaming a fact build, remember that the names of objects in IBM Cognos Data Manager Designer are not case sensitive. For example, VendorDrillThru, VENDORDRILLTHRU, and vendordrillthru are considered to be identical.

Procedure

1. Click the fact build you want to rename.
2. From the **Edit** menu, click **Properties** .
3. In the **Name** and **Business name** boxes, type the new name and business name.
4. Click **OK**.

Set the Date Source for Timestamping Data

When you execute a fact build, data is timestamped as follows:

- If the fact build includes dimension deliveries, dimension records that contain create date or update date attributes are timestamped.

- If the fact build is configured to add unmatched members to the dimension reference data, dimension records that contain create date, update date, or effective start date attributes are timestamped.
- If the fact build includes the DS_DATA_TIMESTAMP variable, for example in a derivation, a timestamp is added as required.

For more information about the create date, update date, and effective start date attributes, see “Behavior Types in Templates” on page 54.

For more information about the DS_DATA_TIMESTAMP variable, see “DS_DATA_TIMESTAMP” on page 298.

By default, data is timestamped using the current system date and time on the computer on which the build is executed. If you require, data can be timestamped using a different date source.

Procedure

1. Click the required fact build.
2. From the **Edit** menu, click **Properties** .
3. Click the **General** tab.
4. In the **Data timestamp** box, choose the date source to be used for timestamping data:
   - Click **Client system date/time** to set the date to the current system date and time on the computer on which the build is executed. (default)
   - Click **Variable** to use a variable to set the date and, in the adjacent box, type the variable name.

     Note: The variable must be defined on the **Variables**tab. For information, see Chapter 24, “Variables,” on page 289.
   - Click **Date (yyyy-mm-dd hh:mm:ss)** to set an explicit date and time and, in the adjacent box, type the date and time.
5. Click **OK**.

Handle Duplicate Data

Duplicate source data rows have identical dimension elements. You can specify the action to be taken if duplicate records are found.

Note: When aggregating data, duplicate keys in the input data can produce different results depending on how the duplicates are handled. For more information, see “Aggregate Duplicate Data” on page 166.

Procedure

1. Click the fact build.
2. From the **Edit** menu, click **Properties** .
3. Click the **Input** tab.
4. In **Duplicate key handling**, click one of the following options:
   - **Allow records with duplicate keys** for IBM Cognos Data Manager to accept the duplicated records. You should use this when duplicates are unimportant or when you know that duplicates will not occur. (default)

- **Reject records with duplicate keys** for Data Manager to accept the first data row and reject subsequent rows to the reject file. For information, see Chapter 28, “Handling Data Rejected by a Fact Build,” on page 339.
- **Merge records with duplicate keys** for Data Manager to track the duplicated records and merge all dimension values using the merge functions that you specify. For information, see “Aggregate Duplicate Data” on page 166.
- **Aggregate records with duplicate keys** to track the duplicated records and aggregate all dimension values using the aggregate function you specify. For information, see “Aggregate Duplicate Data” on page 166.

Note: If you are aggregating data using aggregate rules, you cannot allow records with duplicate keys. You must reject, merge or aggregate them.

Tip: Merging or rejecting duplicates works well with breaking to minimize memory usage. For information, see “Dimension Breaks” on page 344.

5. Click **OK**.

   The build details are updated in the **Visualization** pane to graphically show how duplicates are to be handled. For information, see “Fact Build Visualization Icons” on page 500.

Delete a Fact Build

You can delete a fact build that is no longer required.

Procedure

1. Click the required fact build.
2. From the **Edit** menu, click **Delete**.
3. In the message box, click **Yes**.

Chapter 14. DataStreams

Use DataStreams to access source data from a database.

IBM Cognos Data Manager automatically adds an empty DataStream to each fact build and reference structure that you create. If you create a hierarchy, an additional DataStream is added to each level that you create. You can therefore use a DataStream to provide data for the whole hierarchy or for each hierarchy level.

Within a DataStream, you must define the necessary data sources that specify from where, and how, you will access data. A DataStream can contain as many data sources as you require.

After you define the data sources, you must create DataStream items to map the data source columns to the transformation model elements or reference structure level attributes.

You should use a DataStream to

- access data for a fact build
- access data from multiple data sources
- access structure data for multiple levels of a hierarchy
- perform derivations on source data
- add literal values

For reference structures, an alternative to using a DataStream to access data is to use a template. For information on when to use a DataStream or template to access data for a reference structure, see “Set Up Data Access” on page 75.

You can apply a DataStream filter to remove unwanted data rows before they reach the transformation model. For information, see “Filter Source Data” on page 132.

You can execute a DataStream to examine the source data prior to transformation. For information, see “Execute a DataStream” on page 133.

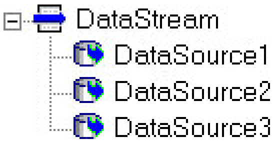
View DataStreams

You can view DataStreams in the Tree pane or the Visualization pane.

Note: You can only view DataStreams for fact builds in the Visualization pane.

View DataStreams in the Tree Pane

In the Tree pane, each DataStream is represented as follows.



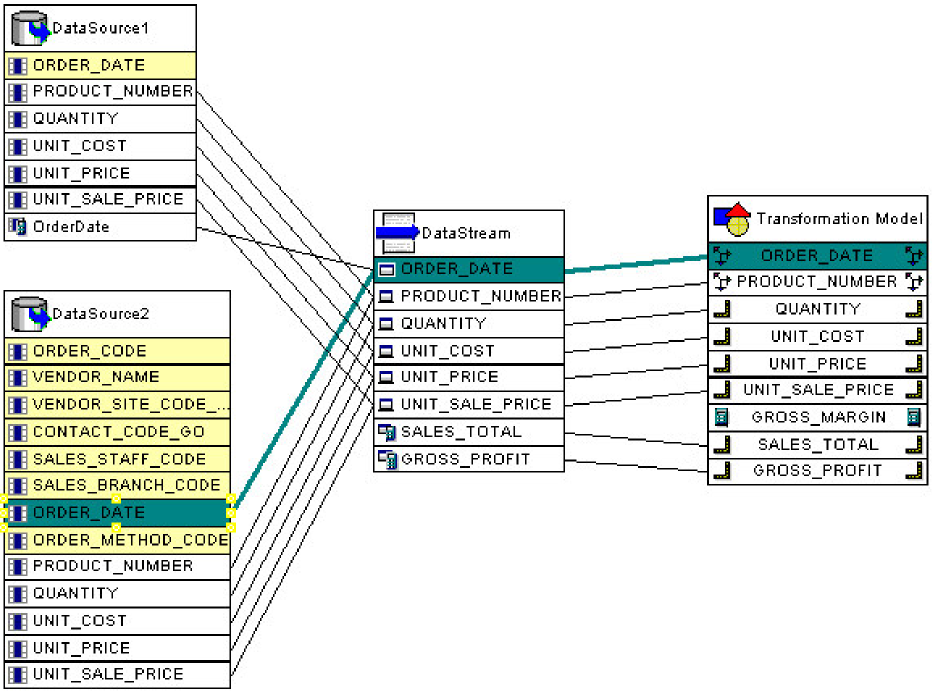
View Fact Build DataStreams in the Visualization Pane

To view the DataStream for a fact build, click the DataStream and then click the Mapping tab in the Visualization pane.

This tab shows the mapping between the data source columns on the left, the DataStream items in the middle, and the transformation model on the right.

When you click on a data source column, a DataStream item, or a transformation model element, the mapping from the data source through to the transformation model element is highlighted.

Unmapped items are indicated by a change of background color.



Set Up a DataStream

This section describes the steps that are necessary to set up a DataStream.

To set up a DataStream, you

- add data sources and add an SQL statement to return columns of data
- add literal values to the data sources if required
- add derivations to the data sources or DataStream if required
- create DataStream items for mapping
- map data source columns returned by the data sources to the DataStream items
- map the DataStream items to either the transformation model elements or the reference
- structure level attributes as appropriate
- add a filter if required

Manage Data Sources

A data source is any location from which IBM Cognos Data Manager can access data.

Each data source either contains an SQL based query, or a link to a query subject contained in a published IBM Cognos Framework Manager package.

A data source can also contain literal values and derivations. For information see "Define Literal Values in a Data Source" on page 123 and "Derivations" on page 128.

After you add a data source, you must map the data source columns to the DataStream items.

You can use multiple data sources to input data from more than one connection. This can be useful when you want to perform data cleansing using different queries. When you use multiple data sources, you must access the data using a DataStream because this is where the data sources are merged.

Add a Data Source for a SQL-based Source Data

Before you add a data source, ensure that you have created a connection to the required database from which you want to access source data.

For information, see Chapter 7, "Connections," on page 39.

For information about creating a Published FM Package data source, see "Add Published FM Package Data Source" on page 119.

Procedure

1. Click the DataStream for which you want to add a data source.
2. Click the **Insert** menu, and
   - if you are adding a data source within a fact build, click **Build**, and then click **Data Source**.
   - if you are adding a data source within a reference structure, click **Library**, and then click **Data Source**.
3. Click the **General** tab.
4. Type a name for the data source and, if you require, a business name and description.

   Tip: To exclude the data source when the build is executed, clear the **Enabled** check box.
5. Click the **Query** tab.

   Note: You must specify the SQL statement before you can save the data source.
6. In the **Database** box, click the database in which the source data resides.

   Tip: Click **New** to create a connection. For information, see "Create a Database Connection" on page 39.
7. In the **Query** pane, enter the SQL statement for the data source. You can type this or you can use one of the following methods:
   - Right-click the required table or column, and then click either **Add table select statement**, or **Add column select statement** as appropriate.

- Hold down **Ctrl** and drag the table or column that you selected in the **Database Objects** pane to the **Query** pane.

For more information on entering SQL statements, see Chapter 6, "SQLTerm and SQL Helper," on page 31.

IBM Cognos Data Manager clears any existing query that may be in the **Query** pane and generates the SQL SELECT statement for the selected table or column.

- You can use Cognos SQL when you construct the SQL statement.
- To view the executed SQL query, click the **Show executed SQL query** button .
- To refresh the list of tables in the **Database Objects** pane, double-click the database object, or right-click and then click either **Refresh Database** or **Refresh Table**.
- To clear the query, click the **Clear the query** button .

8. You are now ready to execute the SQL statement.
9. Click the **Result Columns** tab.
10. Populate the **Result columns** pane by clicking either **Prepare** or **Parse**, and then clicking **Refresh**. For more information, see "Prepare or Parse" on page 122.
11. Click **OK**.

Define Advanced Settings for a Data Source

You can use the advanced settings to specify the type of SQL to use, the number of rows to retrieve in one fetch call, and the isolation level.

Different databases support different isolation levels. Not all databases support transaction isolation levels. A transaction within a database delimits a unit of work. Multiple transactions from many users are usually performed on the database at the same time. Some users may be reading the database, and some users may be entering data into the database. Two transactions can interfere with each other.

Note: You cannot define advanced settings for a Published FM Package data source.

Procedure

1. Click the data source for which you want to define the advanced settings.
2. From the **Edit** menu, click **Properties** .
3. Click the **Query** tab.
4. Click **Advanced**.
5. Select the **Cognos SQL** check box to use Cognos SQL when you construct the SQL statement. If you clear this check box, you must use native SQL for the database you are accessing.

   Note: Selecting the **Cognos SQL**check box in the **Connection Properties**dialog box, determines the default for the **Cognos SQL**check box.
6. If your database supports a bulk fetch operation, in the **Fetch row count** box, type the number of rows that are queried in one fetch call using the bulk fetch facility for your database. Use this to tune large extractions from a source database. The default fetch row size is 50.

Note: There may be a limit on the number of rows that you can retrieve. If that limit is lower than the value in the **Fetch row count**box, that limit is used instead.

7. Use the **Isolation level** box to set the lowest transaction isolation level supported by your database to minimize the potential impact of long running queries on data that might be concurrently updated. Select
   - **Read uncommitted** to make changes made by other transactions immediately available to a transaction.
   - **Read committed** to give a transaction access only to rows that other transactions have committed.
   - **Cursor stability** to stop other transactions updating the row upon which a transaction is positioned.
   - **Repeatable read** to guarantee that rows selected or updated by a transaction are not changed by another transaction until the transaction is complete.
   - **Phantom protection** to stop a transaction from having access to rows inserted or deleted since the start of the transaction.
   - **Serializable** to ensure that a set of transactions executed concurrently produce the same result as if they were performed sequentially.

   See your vendor documentation for more details about transaction and isolation levels.
8. Click **OK**.

Add Published FM Package Data Source

In addition to accessing source data stored in a database, you can also access query subjects contained in a published IBM Cognos Framework Manager package.

Before you can add a Published FM Package data source, you must

- create an IBM Cognos Data Manager connection to the package

  For information, see "Create a Published FM Package Connection" on page 43.
- be logged on to IBM Cognos Connection, the portal to IBM Cognos Business Intelligence, which provides a single access point to all corporate data available in IBM Cognos BI.

  For information, see "Manage Credentials" on page 47 and the IBM Cognos Connection *User Guide*.

  Note: If the server to which you are connecting accepts anonymous users, you do not need to log in.

Procedure

1. Click the DataStream for which you want to add a Published FM Package data source.
2. Click the **Insert** menu, and
   - if you are adding a data source within a fact build, click **Build**, and then click **Data Source**.
   - if you are adding a data source within a reference structure, click **Library**, and then click **Data Source**.
3. Click the **General** tab. Type a name for the data source and, if you require, a business name and description.

   Tip: To exclude the data source when the build is executed, clear the **Enabled** check box.

4. Click the **Query** tab. In the **Database** box, click the required Published FM Package connection.

   Tip: Click **New** to create a connection. For information, see "Create a Published FM Package Connection" on page 43.
5. Populate the **Query items** pane by dragging and dropping a query subject or individual item in the **Query items** pane. There are three columns in this pane:
   - **Name** shows the name of the query subject or item
   - **Expression** shows the expression as it exists in the package
   - **Sort Order** shows the sort order to use

   Tips
   - To clear a single query item, click the **Delete selected query items** button ☒.
   - To clear all the query items in the pane, click the **Delete all query items** button .
6. To apply a sort order for an item, right click in the **Sort Order** column, and select the relevant option:
   - **None**
   - **Ascending**
   - **Descending**
7. You can use filters from the model or you can create a filter in Data Manager.

   To use a filter from the model, in the **Database objects** pane, expand the **Filters** folder, click the required filter, and drag it to the **Filters** pane.

   For information on creating a filter in Data Manager, see "Create a Filter for a Published FM Package Data Source."

   You are now ready to return or execute the query.
8. Click the **Result Columns** tab.

   Populate the **Result columns** pane by clicking either **Prepare** or **Parse**, and then clicking **Refresh**. For more information, see "Prepare or Parse" on page 122.
9. Click **OK**.

Create a Filter for a Published FM Package Data Source

You can use a filter that was previously created in IBM Cognos Framework Manager and included in the published package, create a filter directly in IBM Cognos Data Manager, or use a combination of both.

Use a filter from a published package:

You can use a filter from a published package.

Procedure

1. In the **Database objects** pane, open the **Filters** folder.
2. Click the filter that you require and drag it to the **Filters** pane.
3. Repeat the previous step as required.

   Tips
   - To edit a filter, in the **Filters** pane, double click the filter.
   - To clear a single filter, click the filter and then click the **Delete selected filters** button ☒.

- To clear all the filters in the pane, click the **Delete all filters** button .

Create a filter in Data Manager:

You can use a filter from a published package create a filter in Data Manager.

Procedure

1. Click the **Create a filter** button .

   The **Edit Detail Filter** window appears.
2. To create the filter, drag and drop query items and existing filters from the **Query Items** tab. You can also select user-defined variables, Data Manager Designer and run-time variables from the **Variables** tab.
3. Click **OK**.

   Tip: To edit a filter, in the **Filters** pane, double click the filter.

View the Generated Query Specification

You can view the generated XML for the query, and if required, convert it so that you can manually author it.

When a query is converted you cannot return to using the model.

Note: This functionality is intended for advanced users who are familiar with the query specification syntax and the IBM Cognos Software Development Kit.

Procedure

1. Click the **Show query specification** button XML.

   The **Query Source** window appears.
2. Edit the query specification if required.

   Tip: Select the **Word wrap** check box to use word wrapping.
3. To convert the query items to the query specification, click **Convert**.

   Note: Converting the query items cannot be undone.

View the Generated SQL or MDX

If you convert the query for manual authoring, you can view the generated SQL or, if your data is an OLAP source, the generated MDX query that will be executed when the query specification is processed.

You can also convert the query specification to use the generated SQL or MDX as its data source. When a query is converted you cannot return to using the model.

Procedure

1. Click the **View SQL** button .
2. If you are viewing SQL, from the drop down list, click either **Native SQL** or **Cognos SQL** as appropriate.
3. Edit the SQL or MDX if required.
4. To convert the query items to SQL or MDX, click **Convert**.

   Note: Converting the query items cannot be undone.

Prepare or Parse

After you enter the SQL statement, or for a published FM package data source, populated the Query Items pane, you must specify whether to use prepare or parse to populate the Result Columns tab.

By default, all new data sources use the prepare method.

The returned columns are matched to the results of the query by the column's ordinal position. If you modify the query after you determined the columns, you should regenerate the list of columns by clicking Refresh. When this mapping is complete, the column name controls all other mappings in the DataStream.

Prepare

We recommend that you use the prepare method. It sends the query to the database and returns the column names. For example, the following SQL statement returns ContactCode and Name:

```
SELECT ContactCode,
FirstName & ' ' & LastName As Name
From VContact
```

You can use database-specific syntax. For example, SQL statements that do not begin with SELECT and database-specific concatenation characters.

Tip: To rename columns use the As clause in the SQL statement.

Parse

Parse does not send the query to the database, so you can use it when you cannot connect to the database. The query is parsed and the result set of columns returned as written in the query. For example, the following SQL statement returns FirstName & ' ' & LastName As Name:

```
SELECT ContactCode,
FirstName & ' ' & LastName As Name
From VContact
```

When you use parse, the SQL statement must begin with SELECT. Although using parse is quicker than using prepare, parse may fail if the SQL is too complex. Parse may not evaluate database-specific syntax correctly.

Specify Prepare or Parse

Procedure

1. Click the data source for which you want to specify prepare or parse.
2. From the **Edit** menu, click **Properties** .
3. Click the **Result Columns** tab.
4. Populate the pane by clicking either **Prepare** or **Parse**.
5. Click **Refresh** to update the pane.
6. Click **OK**.

Test a Data Source

You can choose to return all the selected rows in the database in a tabular format, return just a single row in a tabular format, or execute the statement.

Procedure

Do one of the following:

- To return all the rows, in the **Test** pane, click the **Return all rows** button .
- To return a single row, in the **Test** pane, click the **Return one row** button .
- To execute the statement and return the rows, in the **Test** pane, click the **Execute statement** button .
- To interrupt the execution of an SQL query, in the **Test** pane, click the **Interrupt current processing** button .

Change the Properties of a Data Source

If you change the SQL statement, you must check the DataStream mapping because it is not automatically updated.

If you change the SQL dialect, check the SQL statement because it may have become invalid.

Procedure

1. Click the required data source.
2. From the **Edit** menu, click **Properties** .
3. Click the tab that contains the property that you want to change.
4. Change the property.
5. Click **OK**.

Disable and Enable a Data Source

To temporarily prevent a data source from acquiring data, disable and later enable the data source.

Procedure

1. Click the data source.
2. From the **Edit** menu, click **Properties** .
3. Click the **General** tab.
4. Clear or select the **Enabled** check box.
5. Click **OK**.

   Tip: You can also disable or enable a data source by right-clicking the data source in the **Tree** pane, and then clicking **Enabled**.

Define Literal Values in a Data Source

A literal value is a constant that IBM Cognos Data Manager inserts in every row of data that the DataStream returns. Use literal values to identify the source of each data row.

You can add literals to either a data source, or to the SQL statement of a data source.

When a you add a literal value to the SQL statement, the DBMS must return the literal value for each row of data. This is a disadvantage as it may consume additional computer and network resources. Using Data Manager literals is the preferred method to introduce static values.

This is an example of adding a literal to the SQL statement:

```
SELECT 'ALL', Product, Price FROM Product;
```

Procedure

1. Click the required data source.
2. From the **Edit** menu, click **Properties** .
3. Click the **Literals** tab.
4. Click **Add**, and then type the literal value.
5. Repeat step 4 until you have defined all the literal values that you require for the selected data source.

   Tip: To reorder the list, click the literal value that you want to move, and click either **Move Up** or **Move Down**.
6. Click **OK**.

Delete a Data Source

You can delete a data source that is no longer required.

Procedure

1. Click the data source that you want to delete.
2. From the **Edit** menu, click **Delete**.
3. In the message box, click **Yes**.

Manage DataStream Items

Use DataStream items to map the data source columns, returned by data sources, to the transformation model elements or reference structure level attributes.

For information about accessing source data from a database, see “Manage Data Sources” on page 117.

Note: If part of a reference structure accesses data from a template rather than from a DataStream, mapping the data source columns to the reference structure attributes has no effect.

Create a DataStream Item

You must create a DataStream item for each data source column that you want to map to the DataStream.

Tip: IBM Cognos Data Manager can automatically add DataStream items when you map data source columns to the DataStream. For information, see “Map Data Source Columns to the DataStream” on page 125.

Procedure

1. Click the DataStream for which you want to create DataStream items.
2. From the **Edit** menu, click **Properties** .
3. Click the **DataStream Items** tab.
4. Click **Add**.

   The **DataStream Item Properties** window appears.
5. Type a name for the new item and, if you require, a business name and a description.
6. In the **Type** box, click the data type that the DataStream item is to return. For more information, see Appendix E, "Data Types," on page 491.

   If you select **Char**, **Number**, or **Binary** as the return type, the **Precision** box becomes available.
7. In the **Precision** box, type the maximum number of characters or numbers permitted for the DataStream item.

   Note: The precision can be up to 77 for the **Number** data type, and limited by the operating system and computer limits for the **Char** and **Binary** data types.

   If you selected **Number** as the return type, the **Scale** box becomes available.
8. In the **Scale** box, type the number of decimal places permitted for the DataStream item.

   Note: The scale value cannot exceed the precision value.
9. Click **OK**.

   The DataStream item is added to the **DataStream Items** pane.
10. Repeat steps 4 and 5 for each DataStream item you want to add.
11. Click **OK**.

Results

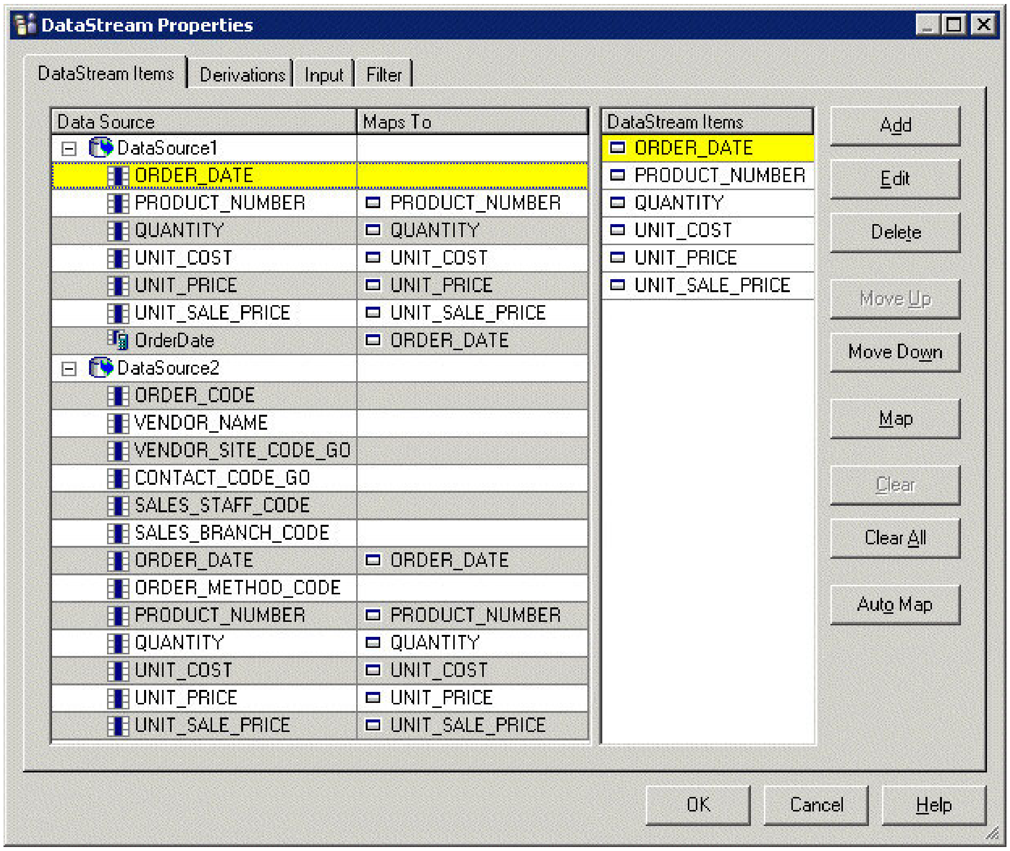
- To reorder the list in the **DataStream Items** pane, click the item that you want to move, and click either **Move Up** or **Move Down**. Reordering DataStream items has no effect on processing.
- To edit the name of a DataStream item, click the item, and then click **Edit**.

Map Data Source Columns to the DataStream

To enable the flow of data from a data source to a fact build or reference structure, you must map the data source columns to the DataStream items.

Once you have mapped the data source columns to the DataStream items, you must map the DataStream items either to the transformation model for a fact build or to the level attributes for a reference structure.

The following is an example of mapping data source columns to DataStream items.



Data Source	Maps To
DataSource1	
ORDER_DATE	
PRODUCT_NUMBER	PRODUCT_NUMBER
QUANTITY	QUANTITY
UNIT_COST	UNIT_COST
UNIT_PRICE	UNIT_PRICE
UNIT_SALE_PRICE	UNIT_SALE_PRICE
OrderDate	ORDER_DATE
DataSource2	
ORDER_CODE	
VENDOR_NAME	
VENDOR_SITE_CODE_GO	
CONTACT_CODE_GO	
SALES_STAFF_CODE	
SALES_BRANCH_CODE	
ORDER_DATE	ORDER_DATE
ORDER_METHOD_CODE	
PRODUCT_NUMBER	PRODUCT_NUMBER
QUANTITY	QUANTITY
UNIT_COST	UNIT_COST
UNIT_PRICE	UNIT_PRICE
UNIT_SALE_PRICE	UNIT_SALE_PRICE

DataStream Items
ORDER_DATE
PRODUCT_NUMBER
QUANTITY
UNIT_COST
UNIT_PRICE
UNIT_SALE_PRICE

Map Data Source Columns to Existing DataStream Items

After you create the required DataStream items, you must map the data source columns to them.

Note: If you map more than one data source column to a single DataStream item for a fact build, the fact data will contain duplicate rows. A separate row is delivered for each data source column you map.

Procedure

1. Click the appropriate DataStream.
2. From the **Edit** menu, click **Properties** .
3. In the **Data Source** column, click the column to map.
4. In the **DataStream Items** column, click the item to which you want to map the column.
5. Click **Map**.

   Tip: You can also perform mapping by dragging the DataStream item to the relevant position in the **Maps To** column.
6. Repeat steps 3 to 5 to complete the required mapping.
7. Click **OK**.

Results

- To map a data source column to more than one DataStream item, repeat steps 3 and 4 to select the required data source column and DataStream item, and then hold down the Ctrl key and click **Map**.

- To clear a mapping, click the mapped item in the **Maps To** column, and then click **Clear**.
- To clear all the mappings, click **Clear All**.

Map Data Source Columns and Create DataStream Items

Rather than creating DataStream items and then mapping data source columns to the DataStream separately, you can perform both tasks together.

Note: If you map more than one data source column to a single DataStream item for a fact build, the fact data will contain duplicate rows. A separate row is delivered for each data source column that you map.

Procedure

1. Click the appropriate DataStream.
2. From the **Edit** menu, click **Properties** .
3. In the **Data Source** column, double-click the data source column that you want to map to the DataStream.

   A DataStream item is created and the selected data source column is mapped to the item.

   - Ensure that nothing is selected in the **DataStream Items** pane because this maps the data source column to the selected DataStream item.
   - If a matching DataStream item already exists, a new item is not created, but the data source column is mapped to the existing one.
4. Repeat step 3 to complete the required mapping.
5. Click **OK**.

Results

- To clear a mapping, select the mapped item in the **Maps To** column, and then click **Clear**.
- To clear all the mappings, click **Clear All**.

Map Data Source Columns Automatically

You can map all unmapped data source columns in a single step.

If DataStream items and data source columns exist with matching names, you can automatically perform mapping. For example, if you have a data source column named OrderDate, and a DataStream item named OrderDate, you can automatically map them.

If no matching DataStream item exists for a data source column, they are created during the automatic mapping process.

Procedure

1. Click the appropriate DataStream.
2. From the **Edit** menu, click **Properties** .
3. Click **Auto Map**.

   All the unmapped data source columns are mapped.
4. Click **OK**.

Delete a DataStream Item

You can delete a DataStream item that is no longer required.

Procedure

1. Click the DataStream for which you want to delete DataStream items.
2. From the **Edit** menu, click **Properties** .
3. Click the **DataStream Items** tab.
4. In the **DataStream Items** pane, click the DataStream item that you want to delete.
5. Click **Delete**.
6. Click **OK**.

Derivations

A derivation is a value that is calculated using an expression that you define, instead of obtained directly from the source data.

You can derive values from individual columns in the source data. For example, you can multiply Price by Units_Sold to create a column named Daily_Sales_Revenue.

You can add derivations to the DataStream or data sources of a fact build or reference structure. You can also add derivations to the transformation model of a fact build. Use the following guidelines when deciding where to perform a derivation.

Where to Perform a Derivation

You can perform derivations in the DataStream or in the data source.

Perform derivations in the DataStream when

- the same calculation is used for each of the data sources, that is, the data for each data source must undergo the same derivation.
- the result of the derivation is required before the data is used in the transformation model, that is, the derivation concatenates keys before the result is used in the transformation model for a reference structure.
- the result of the derivation is to be pivoted.

Perform derivations in the data source when

- you have more than one data source, but the derivation will only be performed on one of them.
- the result of the derivation will be used as an input for other derivations in the DataStream.

Perform derivations in the transformation model when

- the values are only available after the data has been merged.
- the data is only available after pivoting has been performed.
- the data uses input values that are stored from reference structures.

For information about creating derivations in the transformation model, see “Add a Derivation Element to a Fact Build” on page 139.

Expressions in Derivations

When you are entering the derivation expression, you can use the tree structure that is in the left pane of Calculation tab to select items.

The following folders are available:

- Operators to select operators
- Functions to select predefined and user-defined functions
- Control to select controls
- Variables to select variables and static variables

If you are defining a DataStream derivation, you can also select items from the following folders:

- DataStream Items to select DataStream items
- DataStream Derivations to select other derivations defined within the same DataStream

If you are defining a data source derivation, you can also select items from the following folders:

- Columns to select columns returned by the data source
- Literals to select literal values defined in the data source
- Data Source Items to select data source items
- Data Source Derivations to select other derivations defined within the same data source

Add a Derivation

This section describes how to add a derivation to a DataStream.

Procedure

1. Click the DataStream or data source for which you want to define a derivation.
2. From the **Edit** menu, click **Properties** .
3. Click the **Derivations** tab.

   Any derivations that are already defined are listed.

   Tip: To edit a derivation, click the derivation, and then click **Edit**.
4. Click **Add**.

   The **Derivation Properties** window appears.
5. Click the **General** tab. Type a name for the derivation and, if you require, a business name and a description.

   Tip: If you are creating a DataStream derivation that you do not want to be available for mapping in the transformation model, select the **Private** check box.
6. Click the **Calculation** tab. In the right pane, enter an expression for the derivation.

   You can type the expression, and you can use the tree structure in the left pane to select the required items.

   For information about operators, controls, and functions, see the IBM Cognos *Function and Scripting Reference Guide*. For information about user-defined functions, see Chapter 23, “User-Defined Functions,” on page 277. For information about variables, see Chapter 24, “Variables,” on page 289.

7. In the **Return Type** box, click the expected data type for the value that the expression is to return. For more information, see Appendix E, “Data Types,” on page 491.

   Tip: We recommend that you select the same data type as specified in the source data to avoid inconsistencies when the data is converted.

   Note: If you do not specify a return type, a default data type is assigned during execution.

   If you select **Char**, **Number**, or **Binary** as the return type, the **Precision** box becomes available.
8. In the **Precision** box, type the maximum number of digits permitted for the value.

   Note: **Note:** The precision can be up to 77 for the **Number** data type, and limited by the operating system and computer limits for the **Char** and **Binary** data types.

   If you selected **Number** as the return type, the **Scale** box becomes available.
9. In the **Scale** box, type the number of decimal places permitted for the value.

   Note: The scale value cannot exceed the precision value.
10. If required, test the expression. For information, see Chapter 25, “Testing Expressions and Scripts,” on page 309.
11. Click **OK**.

    The derivation is added to the **Derivations** tab in the **Properties** window.

Results

- To show the full text of each expression listed, select **Show All Expression Text**.
- Derivations are resolved in the order they appear. To reorder the list, click the derivation that you want to move, and click either **Move Up** or **Move Down**.
- To delete a derivation, click the derivation, and then click **Delete**.

Reduce the Amount of Input Data

You may want to restrict the amount of input data to speed up processing while testing a build or to produce statistical data based on a sample of the source data.

You can use the following methods to reduce the amount of data that a DataStream makes available to other IBM Cognos Data Manager objects:

- data sampling
- applying a maximum row limit

You can control the order in which the data sources are processed, and the order in which each data source presents the data it accesses. This provides some control over which data rows are processed and passed to the DataStream output.

Sample the Data

To configure a DataStream to sample the data, you specify a sample row frequency for that DataStream. When you do this, the DataStream processes one data row in each batch of the size you specify. For example, if you specify a sample row frequency of 100, one data row is processed in every 100 available rows of source data.

The order in which the DataStream processes the data sources affects which rows it processes. For example, if the DataStream is configured to process the data sources in parallel, the sampled data may be biased toward one of the data sources, possibly to the exclusion of another data source. You can avoid this by configuring the DataStream to process the data sources in series.

For more information, see “Change the order of data sources” on page 132.

Procedure

1. Click the DataStream.
2. From the **Edit** menu, click **Properties** .
3. Click the **Input** tab.
4. In the **Sample row frequency** box, type the required value.
5. Click **OK**.

Apply a Maximum Row Limit

You apply a maximum row limit to restrict the number of data source rows that the DataStream processes.

For example, if you specify a maximum row limit of 2000, the DataStream processes no more than 2,000 source data rows.

In addition to the maximum row limit that you set, the following other factors affect how many rows of source data are read:

- the sample row frequency setting for the DataStream
- data source pivoting

Procedure

1. Click the DataStream.
2. From the **Edit** menu, click **Properties** .
3. Click the **Input** tab.
4. In the **Maximum input rows to process** box, type the required value.
5. Click **OK**.

Apply Sample Frequency and a Maximum Row Limit

If you apply both a sample frequency and a row limit to a DataStream, the sample rate is applied, and then the row limit is applied to the sampled data.

For example, if you specify a sample frequency of 100 and a row limit of 1,000, IBM Cognos Data Manager processes one row in every 100 rows until 1,000 rows are processed. Data Manager reads up to 100,000 rows from the data sources. If the data sources contain only 50,000 rows of data, the DataStream outputs only 500 rows.

If data source pivots apply to a transformation model element, the pivot is applied, and then the row limit is applied to the pivoted data. For example, consider a transformation model element pivot that results in the generation of twelve rows of data from each row of source data. If you specify a row limit of 12,000, that limit is reached after 1,000 source data rows are processed.

Change the order of data sources

By default, data sources are read in series unless you are using breaking, in which case they are read in parallel. When data sources are read in series, each data source is opened sequentially, as they are required.

For information on breaking, see “Dimension Breaks” on page 344.

When data sources are processed in series, the order in which they appear in the Tree pane is used. You can change this order if you require.

Procedure

1. Click the data source.
2. From the **Actions** menu, click either **Move Up** or **Move Down**.

   Tip: You can also reorder a data source by clicking it in the **Tree** pane and then dragging it to the required position.
3. Repeat steps 1 and 2 until the data sources appear in the correct order.
4. Click **OK**.

Change the Order of Data in a Data Source

You can change the order in which a data source presents data.

For any data source, you can include an ORDER BY clause in the SQL statement of the data source. This clause changes the order in which the DBMS, and hence the data source, presents the data.

If the data source accesses data for a fact build, you can specify dimension breaks. This appends an appropriate ORDER BY clause to every data source of the DataStream.

For information about reference dimension breaking, see “Dimension Breaks” on page 344.

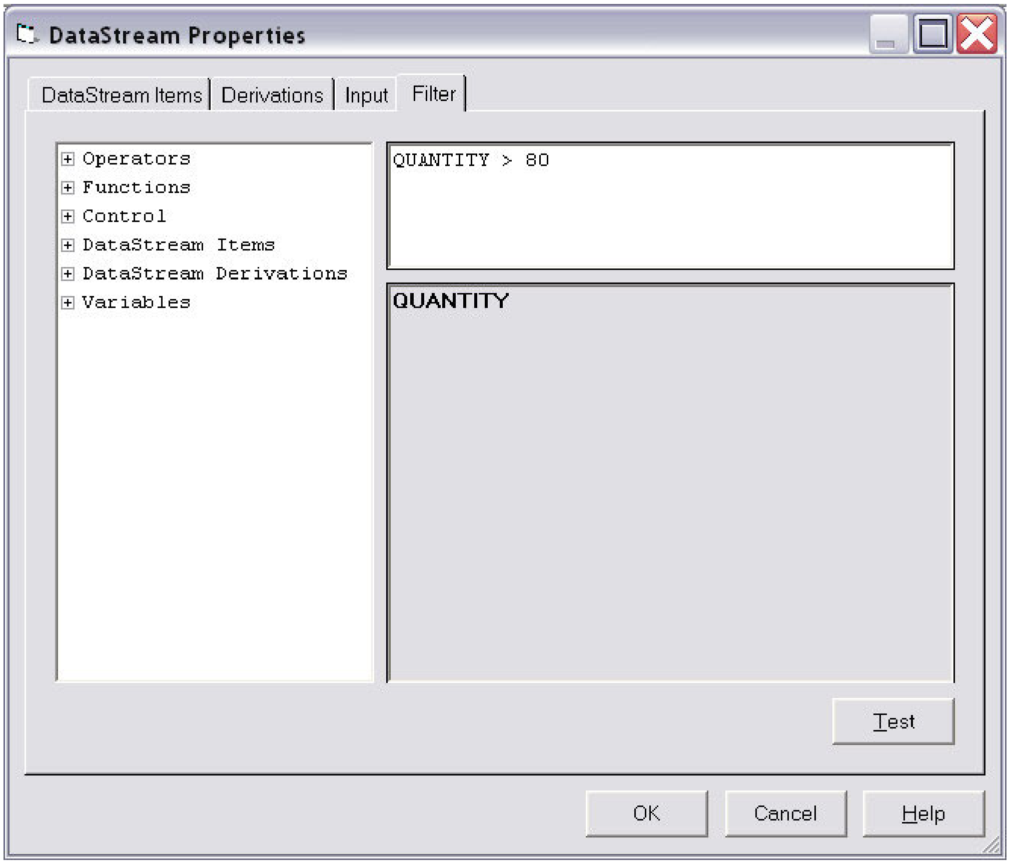
Filter Source Data

A DataStream filter removes unwanted data rows before they reach the transformation model.

This eliminates passing unwanted rows through the transformation model, and then later, filtering them in the deliveries. Using a DataStream filter can improve the transformation capability for dimension data and fact data.

A DataStream filter consists of a logical expression that evaluates to true or false when applied to each source data row. Only the data rows for which the expression evaluates to true are made available to the transformation model. If no DataStream filter is applied, all the data rows are output. When adding an expression, in addition to using operators, functions, controls, and variables, you can also use DataStream items and DataStream derivations for the selected DataStream.

In the following example, the DataStream only allows source data rows where the quantity of units sold is greater than 80.



```
QUANTITY > 80
```

QUANTITY

Procedure

1. Click the DataStream.
2. From the **Edit** menu, click **Properties** .
3. Click the **Filter** tab.
4. In the right pane, enter an expression for the filter.

   You can type the expression, and you can use the tree structure in the left pane to select the required items.

   You can select an item from a folder by dragging it to the right pane, or by double-clicking it.

   For information about operators, controls, and functions, see the IBM Cognos *Function and Scripting Reference Guide*. For information about user-defined functions, see Chapter 23, “User-Defined Functions,” on page 277. For information about variables, see Chapter 24, “Variables,” on page 289.
5. Click **OK**.

   Tip: You can test the filter by executing the DataStream. For more information, see “Execute a DataStream.”

Execute a DataStream

You can execute a DataStream to examine the source data prior to transformation.

By default, IBM Cognos Data Manager executes

- the query for each data source set up within the DataStream.
- all data source and DataStream derivations.

- the first 100 data source rows, regardless of the number of data sources. For example, if you have two data sources, the first 50 rows from each data source are executed. You can, however, specify a different row limit.

You can disable any data source within a DataStream if you do not want to include it in the execution. You can also apply a sample frequency and row limit to the DataStream execution.

Procedure

1. Click the DataStream.
2. From the **Actions** menu, click **Execute** ▶.

   Note: If there are substitution variables included in any of the data source SQL statements, the **Values for Expression**window appears. If required, enter values for the substitution variables, then click **OK**.

   For more information on substitution variables, see “Include Substitution Variables in an SQL Query” on page 36.

   The **Execute DataStream** window appears and the DataStream executes using the default settings. When it has finished, the results appear in the bottom pane.

   Tip: To stop a DataStream from executing before it has finished, click **Interrupt**.
3. If you want to stop a data source from executing, in the **Data Sources** pane, clear the appropriate check box.

   Note: This is different to disabling the data source.

   For information, see “Disable and Enable a Data Source” on page 123.
4. If you want to apply a sample frequency to the DataStream execution, in the **Sample row frequency** box, type the required value.

   Note: The value you set here applies to the DataStream execution only. It has no effect on the sample row frequency set up in the DataStream Properties window.

   For more information, see “Sample the Data” on page 130.
5. If you want to apply a row limit to the DataStream execution, in the **Maximum input rows to process** box, type the required value.

   Note: The value you set here applies to the DataStream execution only. It has no effect on the row limit set up in the DataStream Properties window.

   For more information, see “Apply a Maximum Row Limit” on page 131.
6. If the DataStream has a filter, you can choose to disable it by selecting the **Disable Filter** check box.

   For information, see “Filter Source Data” on page 132.
7. To execute the DataStream again using your new settings, click **Refresh**.
8. When you have finished checking the data, click **Close**.

Preserve trailing Spaces in String Values

By default, trailing spaces are removed from values that are accessed. This conserves memory and improves performance, particularly with large data sets. However, you can preserve trailing spaces if they are significant.

For example, consider an Access table named Example that has a TxtVar column. You could replace SELECT TxtVar FROM Example with SELECT TxtVar & '' As CalcVar FROM Example, and then create a derivation transformation model element that uses the formula RTRIM(CalcVar, '') to remove the concatenated pipe symbol. Then you could deliver the derivation to the target fact tables.

Procedure

1. Replace the column with a calculated value in the SQL for the data source.

   The calculated value should concatenate a non-space character (for example, the piping symbol) to the column value.
2. Create a derivation element in the transformation model that removes the concatenated character from the calculated value.

   For more information, see "Aggregate Data" on page 158.
3. Deliver the derivation element, not the SQL column.

Example

Consider an Access table named Example that has a TxtVar column.

Procedure

1. Replace SELECT TxtVar FROM Example with SELECT TxtVar & '' As CalcVar FROM Example
2. Create a derivation transformation model element that uses the formula RTRIM(CalcVar, '') to remove the concatenated pipe symbol.
3. Deliver the derivation to the target fact tables.

Chapter 15. Transformation Models

The transformation model is central to a fact build. You use it to manipulate the acquired source data in a number of ways, such as merging data from different sources and aggregating data.

The transformation model contains the elements of transformed data. You can create these element types:

- attributes
- derivations
- dimensions
- derived dimensions
- measures

Attributes

An attribute element holds additional information that is not a dimension or a measure but that may be of interest. Attributes differ from measures in that they cannot be aggregated.

Attributes are usually one of the following:

- an attribute of one of the dimensions, such as unit weight or size
- a property of the record, such as the name of the operator who entered the record or the timestamp of record creation

Note: Attributes only have data values at levels retrieved directly from input data SQL queries. In summary information, the value of an attribute is always null.

Derivations

A derivation element is a value that is calculated by using an expression that you define instead of obtained directly from the source data.

Dimensions

A dimension element contains data that is used to provide context for a measure element. For example, a Product_Number dimension element provides context for a Quantity measure element.

Note: Dimension elements provide the link to the dimensional framework.

Derived Dimensions

A derived dimension element allows you to use a calculated expression as the source for a dimension attribute. This expression can be the result of another dimensional element that exists in the transformation model.

Measures

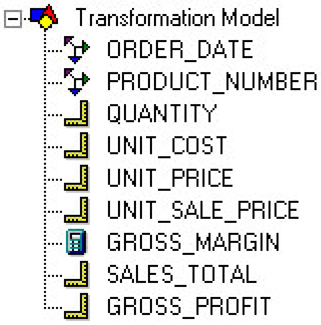
A measure element is a value, frequently numeric, that holds a piece of information for analysis, such as units, revenue, or cost.

View Transformation Models

You can view transformation models in the Tree pane or the Visualization pane.

View Transformation Models in the Tree Pane

The transformation model is represented in the Tree pane as follows.

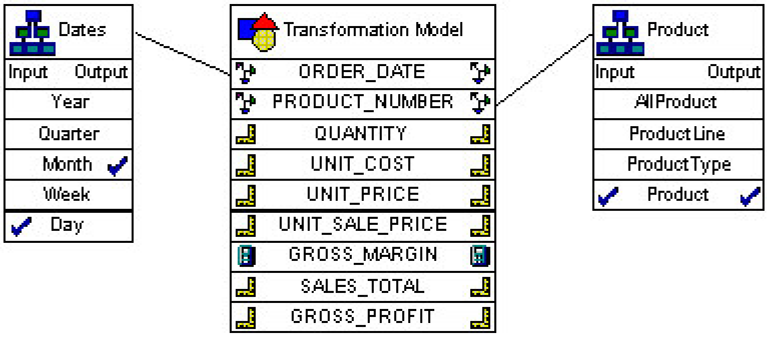


Transformation Model
- ORDER_DATE
- PRODUCT_NUMBER
- QUANTITY
- UNIT_COST
- UNIT_PRICE
- UNIT_SALE_PRICE
- GROSS_MARGIN
- SALES_TOTAL
- GROSS_PROFIT

View Transformation Models in the Visualization Pane

When you click a fact build in the Tree pane, a visual representation appears on the Transformation Model tab in the Visualization pane. The tab shows

- each element in the transformation model
- the reference structure to which each dimension element relates
- the hierarchy levels at which data is accessed (input)
- the hierarchy levels at which data is delivered (output)



Dates

Input	Output
Year	
Quarter	
Month	✓
Week	
✓ Day	

Transformation Model

- ORDER_DATE
- PRODUCT_NUMBER
- QUANTITY
- UNIT_COST
- UNIT_PRICE
- UNIT_SALE_PRICE
- GROSS_MARGIN
- SALES_TOTAL
- GROSS_PROFIT

Product

Input	Output
AllProduct	
Product Line	
Product Type	
✓ Product	✓

Add an Attribute Element to a Fact Build

An attribute element holds additional information that is not a dimension or a measure but that may be of interest. Attributes differ from measures in that they cannot be aggregated.

Tip: You can automatically add an attribute element to the fact build when you map DataStream items to the transformation model. For information, see “Map the DataStream to the Transformation Model” on page 175.

Procedure

1. In the fact build, click **Transformation Model**.
2. From the **Insert** menu, click **Build** and then click **Attribute**.
3. Click the **General** tab.
4. Type a name for the new element. If required, type a business name and description.

   Tip: If you do not want the element made available to the deliveries, select the **Never output** check box. This is useful if an element is required only for use in calculations.
5. Click the **Merge** tab to change the merge method from the default value of first non-null.

   For more information, see "Merge Data for a Measure or an Attribute" on page 173.
6. Click **OK**.
7. If the fact build has deliveries defined, a message appears asking whether you want to add the element to existing delivery modules in the build. Click **Yes** or **No** as appropriate.

   For more information, see "Add an Element to the Deliveries" on page 177.

Add a Derivation Element to a Fact Build

A derivation is a value that is calculated using an expression that you define, instead of being obtained directly from the source data. You can derive values from individual columns in the source data. For example, you can multiply Price by Units_Sold to create a column named Daily_Sales_Revenue.

You can add derivation elements to the transformation model of a fact build. You can also add derivations to the DataStream or data sources of a fact build or reference structure. For information, see "Derivations" on page 128.

Tip: Use derivations in data sources and the DataStream when the result of the derivation is required for pivoting or for looking up a reference element.

Procedure

1. In the fact build, click **Transformation Model**.
2. From the **Insert** menu, click **Build**, and then click **Derivation**.
3. Click the **General** tab.
4. Type a name for the derivation. If required, type a business name and description.
   - If you do not want the element made available to the deliveries, select the **Never output** check box. This is useful if an element is required only for use in subsequent calculations.
   - If you want aggregation performed for a derivation element, select the **Calculate at Source** check box. For more information, see "Calculate a Derivation Before Aggregation" on page 160.
5. Click the **Calculation** tab.

   For information about entering calculations, see "Add a Calculation to a Derivation or Derived Dimension" on page 153.
6. If you selected **Calculate at Source** on the **General** tab, the **Aggregation** tab is available. Click this tab to specify the aggregation details.

   For more information, see "Aggregate Data" on page 158.

7. Click **OK**.
8. If the fact build has deliveries defined, a message appears asking whether you want to add the element to existing delivery modules in the build. Click **Yes** or **No** as appropriate.

   For more information, see "Add an Element to the Deliveries" on page 177.

Add a Dimension Element to a Fact Build

A dimension element contains data that is used to provide context for a measure element. For example, a Product_Number dimension element provides context for a Quantity measure element.

For more information, see "Business Dimensions" on page 4.

Tip: You can automatically add a dimension element to the fact build when you map DataStream items to the transformation model. For information, see "Map the DataStream to the Transformation Model" on page 175.

Procedure

1. In the fact build, click **Transformation Model**.
2. From the **Insert** menu, click **Build**, and then click **Dimension**.
3. In the **Properties** window, click the **General** tab.
4. Type a name for the new element. If required, type a business name and description.

   Tip: If you do not want the element made available to the deliveries, select the **Never output** check box. This is useful if an element is required only for use in subsequent calculations.
5. Click the **Reference** tab. You must use the **Dimension** and **Structure** boxes to provide a link between the fact table and the dimension table.

   For information, see "Associate a Dimension or Derived Dimension Element with a Reference Item" on page 146.
6. Use the remaining tabs to specify other details for the dimension element.

   For information, see the relevant sections in this chapter.
7. Click **OK**.
8. If the fact build has deliveries defined, a message appears asking whether you want to add the element to existing delivery modules in the build. Click **Yes** or **No** as appropriate.

   For more information, see "Add an Element to the Deliveries" on page 177.

Add a Derived Dimension Element to a Fact Build

A derived dimension allows you to use a calculated expression as the source for a dimension attribute.

You can use a derived dimension to perform dimensional lookups against existing reference data. This reduces the need to stage data or use the Lookup function in a data source or DataStream derivation.

You must enter a calculated expression as the source for the derived dimension. The expression can refer to any measure, attribute, or dimension element in the transformation model, and any derived dimension that is before it in the transformation model. However, you cannot use a derivation element in the calculation because these are calculated after merging, so they are not available.

You can create derived dimensions that are nested to more than one level, that is, a lookup on a lookup on a lookup, and so on.

When IBM Cognos Data Manager executes the build, dimension elements are validated first, then the first derived dimension that is listed in the transformation model, followed by the second, and so on. This order is especially important for nested derived dimensions.

Procedure

1. In the fact build, click **Transformation Model**.
2. From the **Insert** menu, click **Build**, and then click **Derived Dimension**.
3. Click the **General** tab.
4. Type a name for the new element. If required, type a business name and description.

   Tip: If you do not want the element made available to the deliveries, select the **Never output** check box. This is useful if an element is only required for use in subsequent calculations.
5. Click the **Calculation** tab. You must enter a calculation to access the reference structure where the information that you require is held.

   For information, see “Add a Calculation to a Derivation or Derived Dimension” on page 153.
6. Click the **Reference** tab. Select the reference dimension and the reference structure that holds the required information.

   For information, see “Associate a Dimension or Derived Dimension Element with a Reference Item” on page 146.
7. Use the remaining tabs to specify other details for the derived dimension element.

   For information, see the relevant sections in this chapter.
8. Click **OK**.
9. If the fact build has deliveries defined, a message appears asking whether you want to add the element to existing delivery modules in the build. Click **Yes** or **No** as appropriate.

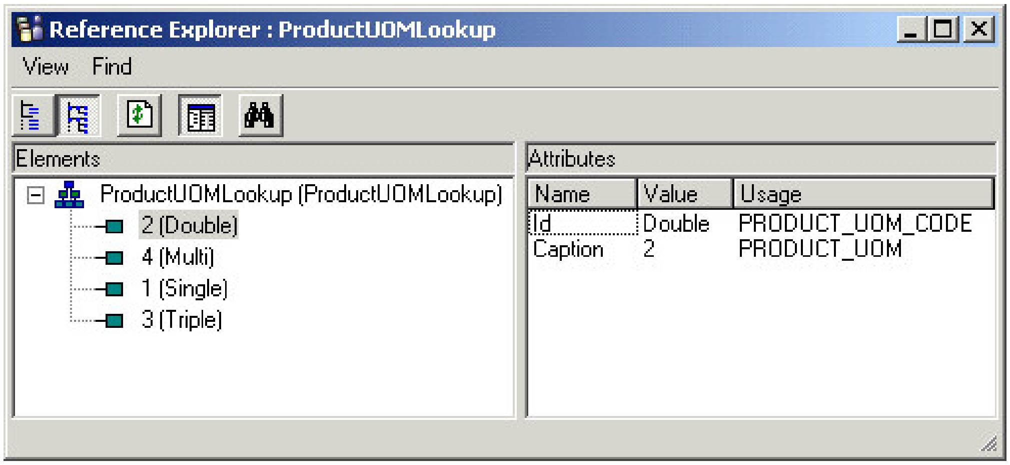
   For more information, see “Add an Element to the Deliveries” on page 177.

Example

This example describes how you can use a derived dimension to perform a lookup on a lookup.

You have sales records that, among other things, show the number of items sold, by which sales person, and on which date. The information is referenced by lookups.

The number of items sold show how many single, double, triple, or multi packs of tent pegs were sold. What you want to know is not how many packs have been sold, but how many individual tent pegs. The Product_UOM table contains a unit of measure and a multiplier. IBM Cognos Data Manager uses these to calculate how many tent pegs have been sold.



Reference Explorer : ProductUOMLookup

View Find

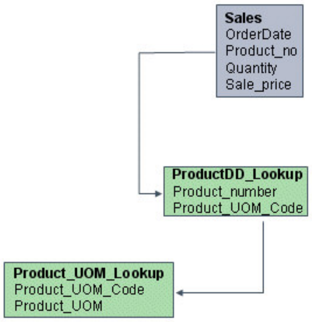
Elements

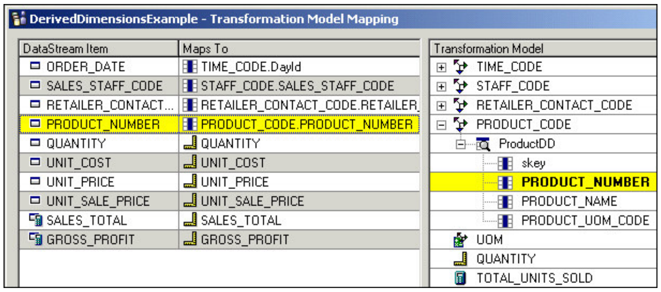
- ProductUOMLookup (ProductUOMLookup)
  - 2 (Double)
  - 4 (Multi)
  - 1 (Single)
  - 3 (Triple)

Attributes

Name	Value	Usage
Id	Double	PRODUCT_UOM_CODE
Caption	2	PRODUCT_UOM

There is no direct join between the sales data and the ProductUOM which is contained in the Product dimension. You can use a derived dimension to link the ProductUOM to the ProductDD table, using the ProductUOMCode which exists in both lookups.





DerivedDimensionsExample - Transformation Model Mapping

DataStream Item	Maps To
ORDER_DATE	TIME_CODE.DayId
SALES_STAFF_CODE	STAFF_CODE.SALES_STAFF_CODE
RETAILER_CONTACT...	RETAILER_CONTACT_CODE.RETAILER_
PRODUCT_NUMBER	PRODUCT_CODE.PRODUCT_NUMBER
QUANTITY	QUANTITY
UNIT_COST	UNIT_COST
UNIT_PRICE	UNIT_PRICE
UNIT_SALE_PRICE	UNIT_SALE_PRICE
SALES_TOTAL	SALES_TOTAL
GROSS_PROFIT	GROSS_PROFIT

Transformation Model

- TIME_CODE
- STAFF_CODE
- RETAILER_CONTACT_CODE
- PRODUCT_CODE
  - ProductDD
    - skey
    - PRODUCT_NUMBER
    - PRODUCT_NAME
    - PRODUCT_UOM_CODE
- UOM
- QUANTITY
- TOTAL_UNITS_SOLD

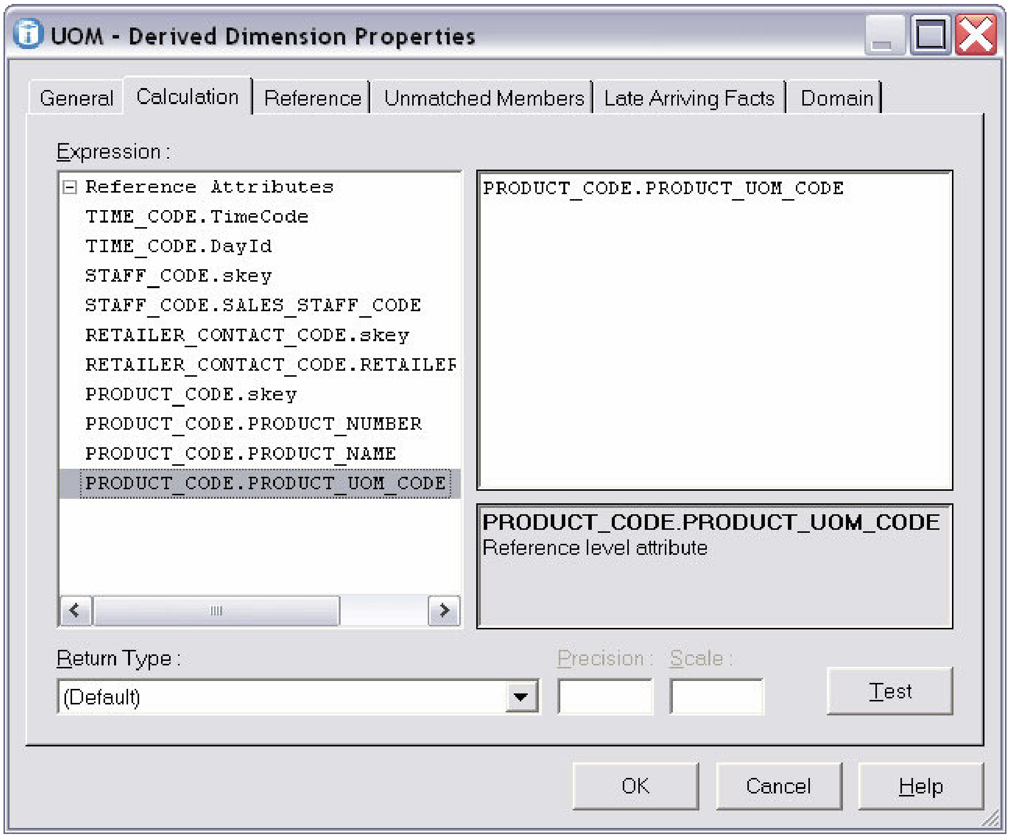
A derived dimension allows us to use the Product_UOM_Code reference attribute from the Product_Lookup as the source for the Product_UOM_Lookup.

Before you create a derived dimension, you must have created the required data sources and DataStream, performed mapping between them, and created the required reference structures.

You can work through the following steps using the ds_advanced.mdb sample catalog, and you can also explore the DerivedDimensionExample fact build.

Procedure

1. Add a derived dimension element to the transformation model.
2. In the **Derived Dimension Properties** window, click the **Calculation** tab.
3. In the **Expression** pane, double click **Reference Attributes** and then PRODUCT_CODE.PRODUCT_UOM_CODE. PRODUCT_UOM_CODE is the attribute on the ProductDDLookup used to link to the Product_UOM_Lookup because it exists in both.



Expression :

```
⊟ Reference Attributes
  TIME_CODE.TimeCode
  TIME_CODE.DayId
  STAFF_CODE.skey
  STAFF_CODE.SALES_STAFF_CODE
  RETAILER_CONTACT_CODE.skey
  RETAILER_CONTACT_CODE.RETAILEF
  PRODUCT_CODE.skey
  PRODUCT_CODE.PRODUCT_NUMBER
  PRODUCT_CODE.PRODUCT_NAME
  PRODUCT_CODE.PRODUCT_UOM_CODE
```

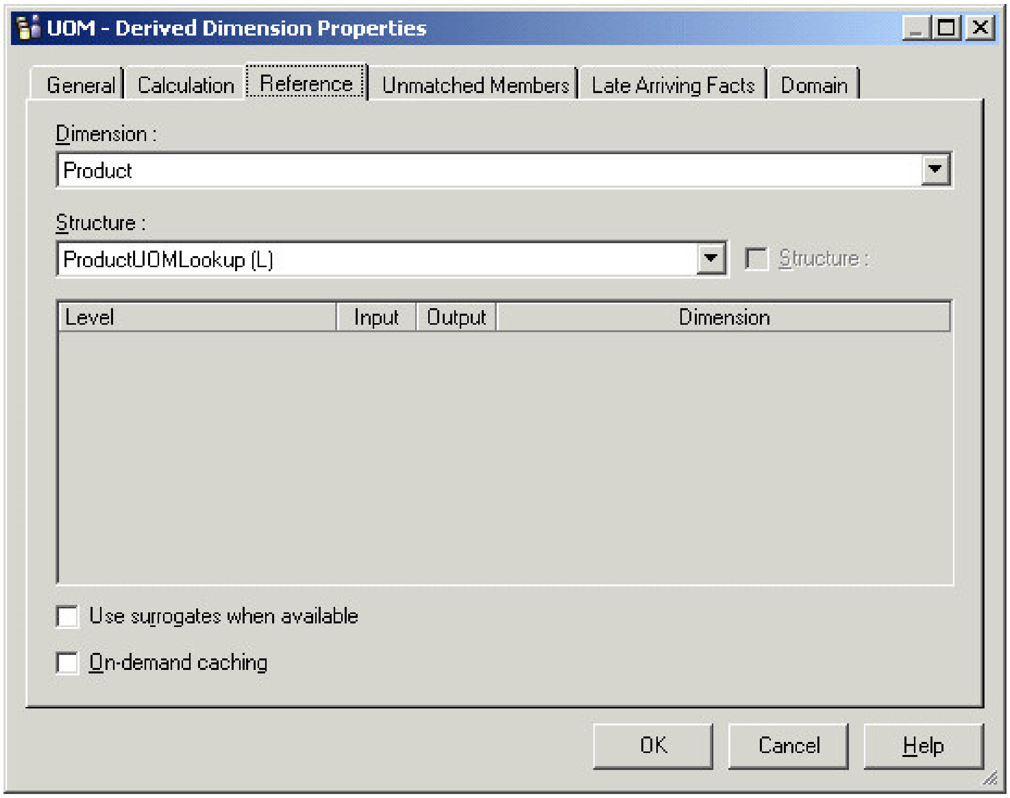
```
PRODUCT_CODE.PRODUCT_UOM_CODE
```

PRODUCT_CODE.PRODUCT_UOM_CODE
Reference level attribute

Return Type : (Default)

Precision : Scale :

4. Click the **Reference** tab.
5. In the **Dimension** box, click Product, and in the **Structure** box, click ProductUOMLookup.



Dimension : Product

Structure : ProductUOMLookup (L)

□ Structure :

Level	Input	Output	Dimension

□ Use surrogates when available

□ On-demand caching

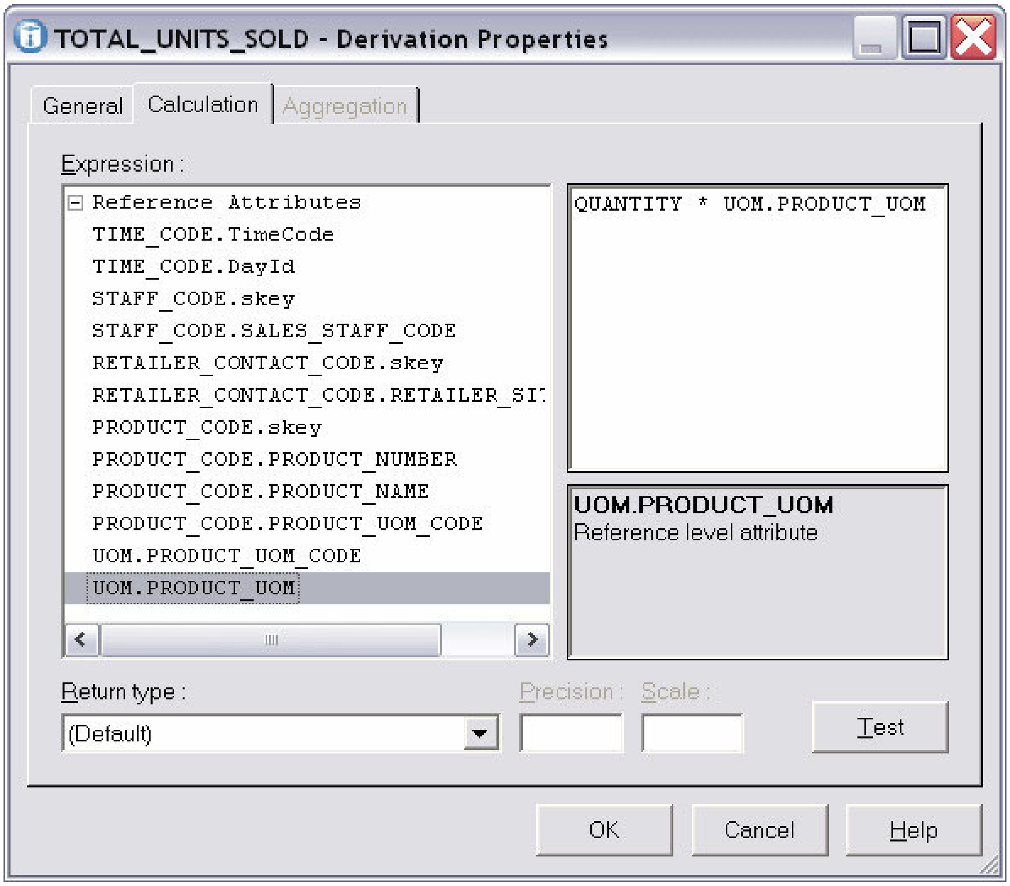
This allows Data Manager to access the ProductUOMLookup that contains the Product_UOM attribute that is the multiplier required to calculate the number of individual tent pegs.

6. Click **OK**.
7. Add a derivation element to the transformation model.
8. In the **Derivation Properties** window, click the **Calculation** tab.
9. In the right pane, enter the following calculation:

   ```
   Quantity * UOM.PRODUCT_UOM
   ```

   This multiplies the quantity, which is the number of packs, with the Product_UOM value which defines the number of tent pegs in the pack.

   Tip: UOM.PRODUCT_UOM is an attribute in the ProductUOMLookup. You can select it by expanding the **Reference Attributes** folder and double clicking the attribute.



```
⊟ Reference Attributes
   TIME_CODE.TimeCode
   TIME_CODE.DayId
   STAFF_CODE.skey
   STAFF_CODE.SALES_STAFF_CODE
   RETAILER_CONTACT_CODE.skey
   RETAILER_CONTACT_CODE.RETAILER_SI
   PRODUCT_CODE.skey
   PRODUCT_CODE.PRODUCT_NUMBER
   PRODUCT_CODE.PRODUCT_NAME
   PRODUCT_CODE.PRODUCT_UOM_CODE
   UOM.PRODUCT_UOM_CODE
   UOM.PRODUCT_UOM
```

```
QUANTITY * UOM.PRODUCT_UOM
```

UOM.PRODUCT_UOM
Reference level attribute

Return type : (Default)

10. Click **OK**.

Add a Measure Element to a Fact Build

A measure element is a value, frequently numeric, that holds a piece of information for analysis, such as units, revenue, or cost.

Tip: You can automatically add a measure element to the fact build when you map DataStream items to the transformation model. For information, see “Map the DataStream to the Transformation Model” on page 175.

Procedure

1. In the fact build, click **Transformation Model**.
2. From the **Insert** menu, click **Build**, and then click **Measure**.
3. Click the **General** tab.
4. Type a name for the new element. If you require, type a business name and description.

   Tip: If you do not want the element made available to the deliveries, select the **Never output** check box. This is useful if an element is required only for use in subsequent calculations.
5. Click the **Aggregation** tab to specify the aggregation details.

   For information, see “Aggregate Data” on page 158.
6. Click the **Merge** tab to change the merge method from the default value of first non-null.

For more information, see “Merge Data for a Measure or an Attribute” on page 173.

7. Click **OK**.
8. If the fact build has deliveries defined, a message appears asking whether you want to add the element to existing delivery modules in the build. Click **Yes** or **No** as appropriate.

   For more information, see “Add an Element to the Deliveries” on page 177.

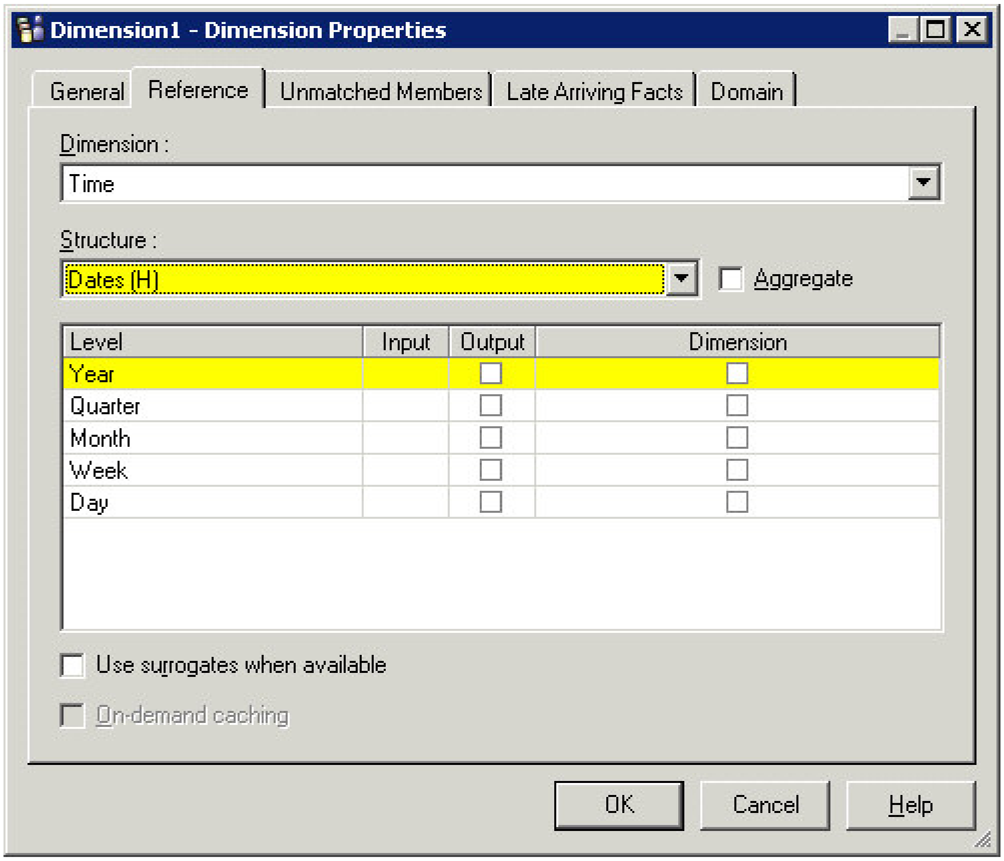
Associate a Dimension or Derived Dimension Element with a Reference Item

You must provide a link between the fact table and the dimension table. You do this by associating each dimension element and derived dimension element with a reference dimension, and with a reference structure from within the selected reference dimension.

You make the associations from the relevant Properties window.

Procedure

1. In the transformation model, click the appropriate dimension element or derived dimension element.
2. From the **Edit** menu, click **Properties** .
3. Click the **Reference** tab.
4. In the **Dimension** box, click the dimension that holds the reference structure that you want to use.
5. In the **Structure** box, click the reference structure that you want to use.



Dimension : Time

Structure : Dates (H) □ Aggregate

Level	Input	Output	Dimension
Year		□	□
Quarter		□	□
Month		□	□
Week		□	□
Day		□	□

□ Use surrogates when available

□ On-demand caching

You can choose from all the hierarchies (shown suffixed with H), auto-level hierarchies (shown suffixed with A), and lookups (shown suffixed with L) in the selected reference dimension.

If you select a hierarchy, the grid becomes available. Use this grid to set the output levels for a dimension and the input and output levels for a derived dimension. For information, see “Set Input and Output Levels for a Dimension or Derived Dimension” on page 148.

If you select a hierarchy or an auto-level hierarchy, the **Aggregate** check box becomes available. For information, see “Aggregate Data” on page 158.

6. Click **OK**.

Associate a Dimension Element With a Reference Item During the Mapping Process

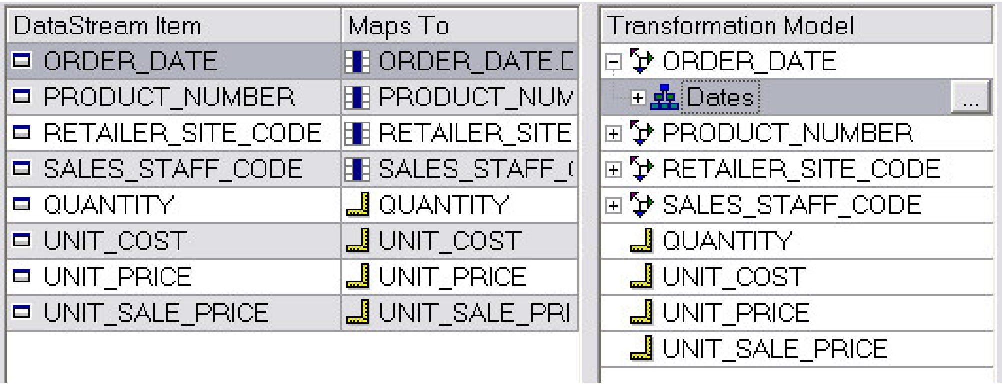
You cannot map to a derived dimension element because the source data comes from a derivation, not the DataStream.

Procedure

1. Click the appropriate transformation model.
2. From the **Edit** menu, click **Mapping**.

   The **Transformation Model Mapping** window appears.
3. In the **Transformation Model** pane, expand the appropriate dimension element.
4. Click **no reference**, and then click the **Browse** button ... that appears.
5. In the **Select Reference** dialog box, expand the dimension that holds the reference structure you want to use, and then click the required reference

structure.



DataStream Item	Maps To
ORDER_DATE	ORDER_DATE.[
PRODUCT_NUMBER	PRODUCT_NUM
RETAILER_SITE_CODE	RETAILER_SITE
SALES_STAFF_CODE	SALES_STAFF_(
QUANTITY	QUANTITY
UNIT_COST	UNIT_COST
UNIT_PRICE	UNIT_PRICE
UNIT_SALE_PRICE	UNIT_SALE_PRI

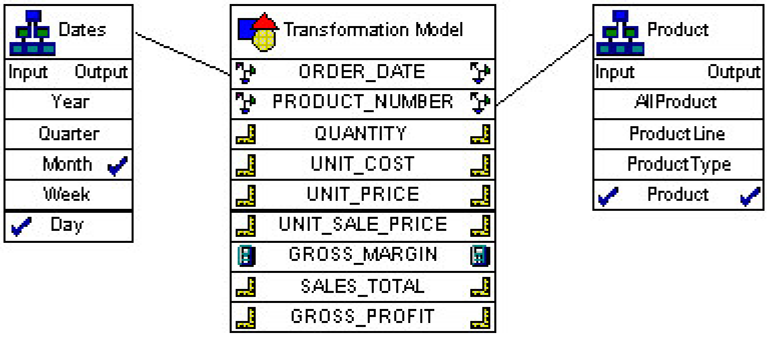
Transformation Model
⊟ ORDER_DATE
⊞ Dates ...
⊞ PRODUCT_NUMBER
⊞ RETAILER_SITE_CODE
⊞ SALES_STAFF_CODE
QUANTITY
UNIT_COST
UNIT_PRICE
UNIT_SALE_PRICE

6. Click **OK**.

Set Input and Output Levels for a Dimension or Derived Dimension

When you associate a dimension element or a derived dimension element with a hierarchy, you can also set the levels at which data is accessed (the input levels) and delivered (the output levels).

You can view the input and output levels for all the dimension and derived dimension elements in the selected build by clicking the Transformation Model tab in the Visualization pane.



Dates		
Input		Output
	Year	
	Quarter	
	Month	✓
	Week	
✓	Day	

Transformation Model
ORDER_DATE
PRODUCT_NUMBER
QUANTITY
UNIT_COST
UNIT_PRICE
UNIT_SALE_PRICE
GROSS_MARGIN
SALES_TOTAL
GROSS_PROFIT

Product		
Input		Output
	AllProduct	
	ProductLine	
	ProductType	
✓	Product	✓

Set Input Levels

Data can enter the fact build process only at the hierarchy input levels.

The method used to set the levels at which source data is to be accessed depends on whether you are defining a dimension element or a derived dimension element.

Set Input Levels for a Dimension

For a dimension, you set the levels at which source data is to be accessed when you map a DataStream item to a transformation model dimension element. Mapping a DataStream item to a dimension element differs from mapping other element types because you cannot map a DataStream item directly to the dimension element. You map the DataStream item to the attribute ID for the required input level.

By default, when you set an input level, you map to the ID attribute of the lowest level of a hierarchy.

Notes

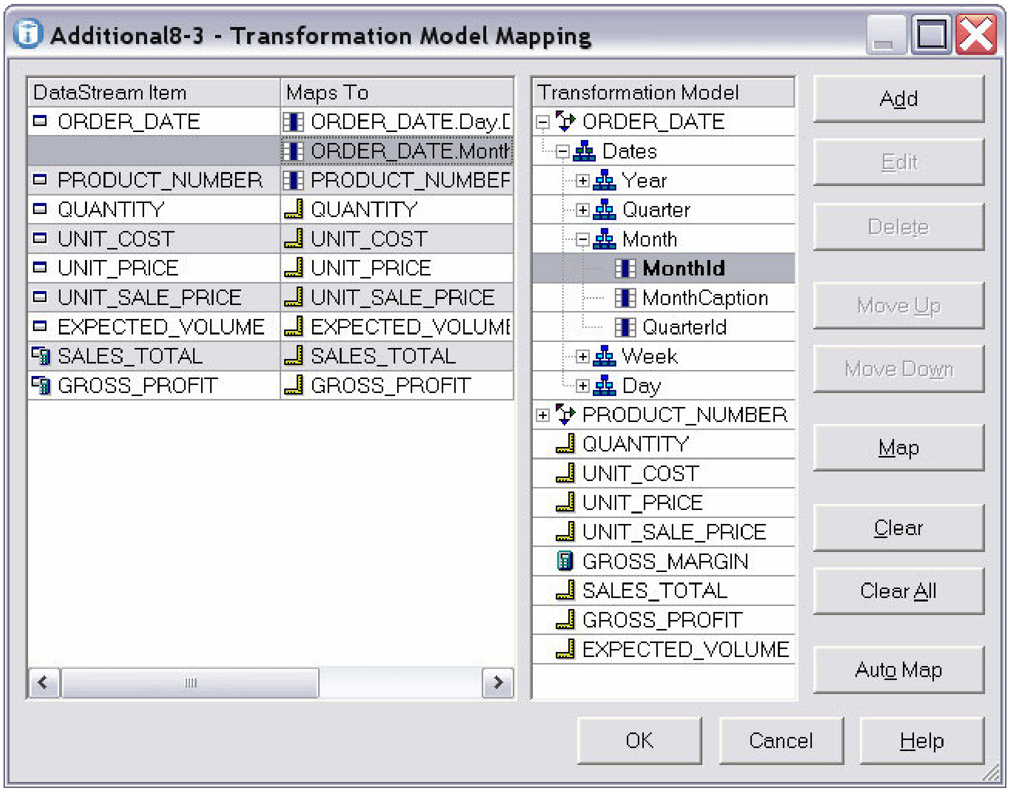
- The ID attribute of the lowest level is always shown in bold.
- You can use only ID attributes to set input levels. It is not possible to map to any other attributes.

For more information, see "Map the DataStream to the Transformation Model" on page 175.

Procedure

1. Click the appropriate transformation model.
2. From the **Edit** menu, click **Mapping**.

   The **Transformation Model Mapping** window appears.
3. In the **Transformation Model** pane, expand the appropriate dimension and reference structure.
4. Click the attribute ID for the required input level.



DataStream Item	Maps To
ORDER_DATE	ORDER_DATE.Day.I
	ORDER_DATE.Montl
PRODUCT_NUMBER	PRODUCT_NUMBEF
QUANTITY	QUANTITY
UNIT_COST	UNIT_COST
UNIT_PRICE	UNIT_PRICE
UNIT_SALE_PRICE	UNIT_SALE_PRICE
EXPECTED_VOLUME	EXPECTED_VOLUMI
SALES_TOTAL	SALES_TOTAL
GROSS_PROFIT	GROSS_PROFIT

```
Transformation Model
⊟ ORDER_DATE
  ⊟ Dates
    ⊞ Year
    ⊞ Quarter
    ⊟ Month
        MonthId
        MonthCaption
        QuarterId
    ⊞ Week
    ⊞ Day
⊞ PRODUCT_NUMBER
  QUANTITY
  UNIT_COST
  UNIT_PRICE
  UNIT_SALE_PRICE
  GROSS_MARGIN
  SALES_TOTAL
  GROSS_PROFIT
  EXPECTED_VOLUME
```

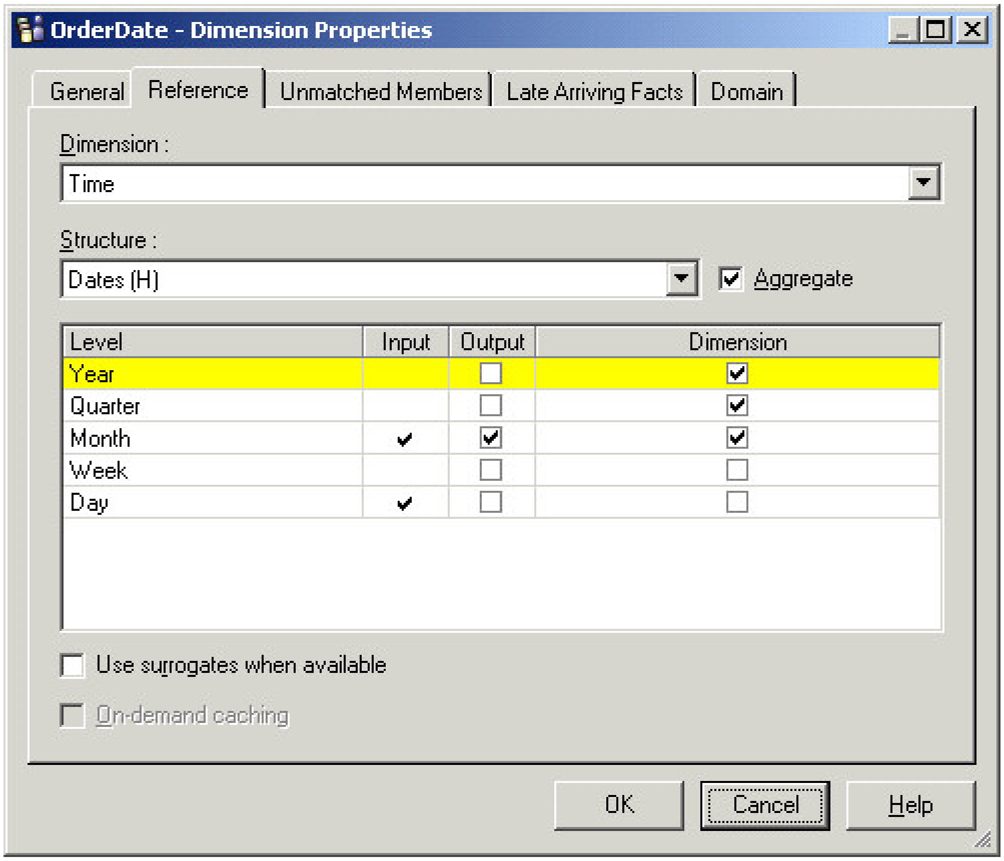
5. In the **DataStream Item** column, click the item for which you want to set the input level.
6. Click **Map**.

   Tip: You can set more than one input level for a DataStream item. Repeat steps 3 to 5 to select the required attribute ID and DataStream item, and then hold down the Ctrl key and click **Map**.
7. Click **OK**.

Set Input Levels for a Derived Dimension

You set the levels at which data is to be input using the grid that appears when you associate a dimension or derived dimension with a hierarchy. In the grid

- **Level** lists the levels in the selected hierarchy.
- **Input** allows you to set the levels at which IBM Cognos Data Manager is to accept input data.
- **Output** allows you to set the levels at which data is to be made available to the deliveries. For information, see "Set Output Levels for a Dimension or Derived Dimension."
- **Dimension** shows the levels that are to be described in dimension deliveries. For information, see "Add a Table to a Dimension Delivery" on page 200.



Dimension : Time

Structure : Dates (H)

☑ Aggregate

Level	Input	Output	Dimension
Year		□	☑
Quarter		□	☑
Month	✓	☑	☑
Week		□	□
Day	✓	□	□

□ Use surrogates when available

□ On-demand caching

Procedure

1. In the transformation model, click the required derived dimension.
2. From the **Edit** menu, click **Properties** .
3. Click the **Reference** tab.
4. In the **Input** column, click the appropriate check boxes to set the levels at which Data Manager is to accept input data.
5. Click **OK**.

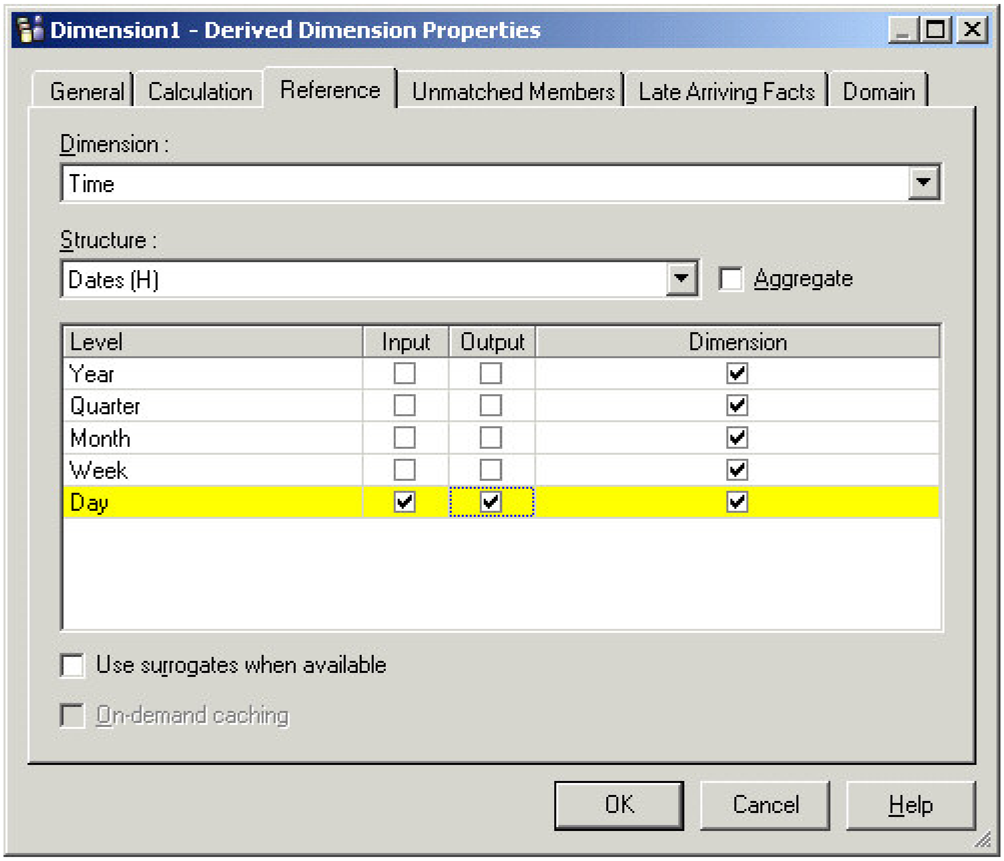
Set Output Levels for a Dimension or Derived Dimension

A fact build delivers data only at the hierarchy output levels that you set.

Aggregation is used to calculate the data to be output. For example, suppose you have a fiscal hierarchy with levels of Year, Quarter, and Period, with data input at the Period level. The values are aggregated at the Period level to obtain values for the Quarter level and the Year level.

You set the levels at which data is to be delivered using the grid that appears when you associate a dimension or derived dimension with a hierarchy. In the grid

- **Level** lists the levels in the selected hierarchy.
- **Input** shows, and for a derived dimension allows you to set, the levels at which IBM Cognos Data Manager is to accept input data. For information, see “Set Input Levels” on page 148.
- **Output** allows you to set the levels at which data is to be made available to the deliveries.
- **Dimension** shows the levels that are to be described in dimension deliveries. For information, see “Add a Table to a Dimension Delivery” on page 200.



Dimension : Time

Structure : Dates (H)

□ Aggregate

Level	Input	Output	Dimension
Year	□	□	☑
Quarter	□	□	☑
Month	□	□	☑
Week	□	□	☑
Day	☑	☑	☑

□ Use surrogates when available

□ On-demand caching

Procedure

1. In the transformation model, click the dimension element or derivation element.
2. From the **Edit** menu, click **Properties** .
3. Click the **Reference** tab.
4. In the **Output** column, click the appropriate check boxes to set the levels at which data is to be output.
5. If Data Manager is to output data for levels for which there will be no input data, select the **Aggregate** check box. Aggregation is then performed to obtain the output data for any associated measures and derivations.

   For more information, see “Aggregate Data” on page 158.
6. Select the **Use surrogates when available** check box to use the surrogate value in the output. For information, see Chapter 10, “Surrogate Keys,” on page 59.

7. Click **OK**.

Exclude Input Data from Fact Build Output

You can specify that the fact build should not deliver any of the input data, but simply use it for aggregation. If you choose this, you can also choose to exclude only the detail level, that is, data at the lowest level of each hierarchy.

Excluding input data and data at the detail level applies only to the fact build, not to a particular dimension or derived dimension.

This table uses a fiscal hierarchy to show the levels at which data is output if you select Exclude Input Data.

Level	**Input**	**Output**
Year	✔	
Quarter	✔	
Period	✔	

This table uses a fiscal hierarchy to show the levels at which data is output if you select Exclude Detail Level Only.

Level	**Input**	**Output**
Year	✔	✔
Quarter	✔	✔
Period	✔	

Procedure

1. Click the required fact build.
2. From the **Edit** menu, click **Properties** .
3. Click the **Output** tab.
4. If you do not want to deliver any of the input data, select the **Exclude Input Data** check box.
5. If source data is obtained from a hierarchy, and you want to exclude data at the lowest level, select the **Exclude Detail Level Only** check box.

Cache Reference Data on Demand

You can cache reference data for a dimension or a derived dimension.

By default, all reference data is cached at the start of a fact build execution. In some cases, this may not be the most efficient method of looking up reference elements.

For example, you may have a large customer dimension that contains records of all customers who have ever made a purchase from a company. If you load sales records on a daily basis, it is unlikely that every customer will have a sales transaction on each day.

Caching all reference data for a large dimension consumes resources. Instead, you can specify that reference data is cached on demand, so that only those reference members that appear in the DataStream of a fact build are retrieved. This means that fewer members are likely to be cached.

Once cached, reference members remain in memory so that they can be referenced again for any subsequent occurrence of a reference key.

Notes

- Caching on demand causes more input and output, as records are retrieved only when they are required. If you know that a large proportion of records need to be cached, it is faster to cache all reference data at the start of the fact build.
- To cache reference data on demand, you must have an appropriate index on the dimension table.
- To cache reference data on demand, the dimension element or derived dimension element must be associated with a lookup that uses a template to access data.

Procedure

1. In the transformation model, click the dimension element or derivation element.
2. From the **Edit** menu, click **Properties** .
3. Click the **Reference** tab.
4. Select the **On-demand caching** check box.
5. Click **OK**.

Add a Calculation to a Derivation or Derived Dimension

The value of a derivation or derived dimension is calculated using an expression that you define.

Procedure

1. Click the **Calculation** tab.
2. In the right pane of the **Expression** box, enter the derivation expression.

   You can use the tree structure in the left pane to select the items from the following folders:

   - **Elements** to select other elements defined within the transformation model
   - **Reference Attributes** to select a dimension and member combination
   - **Operators** to select operators
   - **Functions** to select predefined and user-defined functions
   - **Control** to select controls
   - **Variables** to select variables and substitution variables

   For information about operators, controls, and functions, see the IBM Cognos*Function and Scripting Reference Guide*. For information about user-defined functions, see Chapter 23, “User-Defined Functions,” on page 277. For information about variables and substitution variables, see Chapter 24, “Variables,” on page 289.

You can select an item from a folder by dragging it to the right pane or by double-clicking it.

3. In the **Return Type** box, click the expected data type for the value that the expression is to return. For more information, see Appendix E, “Data Types,” on page 491.

   Note: If you do not specify a return type, a default data type is assigned during execution.

   Tip: We recommend that you select the same data type as specified in the source data to avoid inconsistencies when the data is converted.

   If you select **Char** as the return type, the **Precision** box becomes available. Type the maximum number of characters permitted for the value. The maximum value is limited by the computer resources and the target database limits.

   If you select **Number** in the **Precision** box, the **Precision** box and the **Scale** box become available. In the **Precision** box, type the maximum number of characters or numbers permitted for the value (up to 77). In the **Scale** box, type the number of decimal places permitted for the value.
4. In the **Scale** box, type the number of decimal places permitted for the value.
5. If required, test the expression. For information, see Chapter 25, “Testing Expressions and Scripts,” on page 309.
6. Click **OK**.

   The **Add New Element** dialog box appears, prompting you to add the derivation element to fact and metadata deliveries.

   For information, see “Add an Element to the Deliveries” on page 177.

Dynamic and Reference Domains

A domain is a set of values that exists within a dimension element or derived dimension element.

There are two domain types: dynamic domains and reference domains.

Dynamic Domains

In IBM Cognos Data Manager, a dynamic domain is assembled from the data source rows. A dynamic domain may not necessarily contain all the members in the dimension. For example, suppose the structure data with which the dimension is associated has members A, B, C, D, and E, and your fact data contains values for A, B, and X. The dynamic domain would be A, B, and X. That is, the dynamic domain would not include the dimension members C, D, and E because they have no associated fact data.

The dimension value X does not appear in the structure data. For more information, see “Accept Source Data Containing Unmatched Dimension Values.”

Accept Source Data Containing Unmatched Dimension Values

By default, a source data record containing any dimension value that does not match a member of that dimension's reference data, is rejected when a fact build is executed. These dimension values are known as unmatched members.

There are three options you can set for unmatched members

- **Accept unmatched member identifiers** to include records that would normally be rejected
- **Save unmatched member details via reference structure** to add unmatched members to the dimension reference data so that the data is not unmatched in this and subsequent build executions

  It is not necessary to include a dimension delivery in a fact build to save unmatched members. However, because a template is used to define the behavior of dimension attributes, the reference structure must use a template for data access.

  When unmatched data is added to the dimension reference data, surrogate keys, data defined in the template, and current indicator flags are also updated.
- **Commit each saved member** to commit each unmatched member individually after saving it

  If the fact table delivered has a foreign key constraint on the dimension table, this option is required to successfully save unmatched members.

  Tip: Committing individual members may have a negative effect on performance. To overcome this problem, consider disabling the foreign key constraint while executing the fact build.

Notes

- **Save unmatched member details via reference structure** is available only when the dimension element or derived dimension element is associated with a lookup that uses a template for data access, or with a hierarchy where the lowest input level use a template.
- If you are running more than one fact build simultaneously, and you want to save unmatched members from each build, you must set up distributed handling of unmatched members. For more information, see “Distributed Handling of Unmatched Members.”
- Unmatched members are treated as data at the lowest input level of the reference data.

Procedure

1. Click the required dimension element or derived dimension element.
2. From the **Edit** menu, click **Properties** .
3. Click the **Unmatched Members** tab.
4. To accept source data that does not relate to any dimension reference data, select the **Accept unmatched member identifiers** check box.
5. To add the unmatched members to the dimension reference data, click the **Save unmatched member details via reference structure** check box.

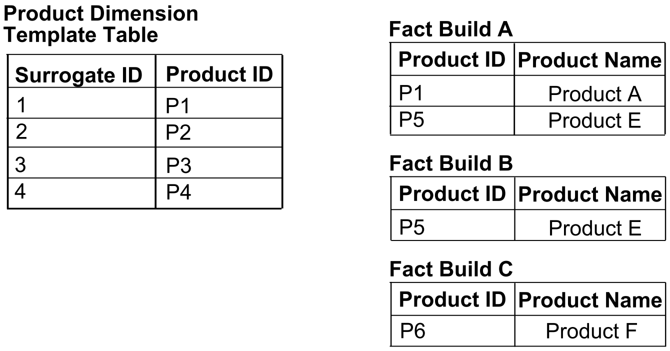
   Note: If the conditions for adding unmatched members are not met, a message appears indicating this and the check box is not available.
6. To commit each unmatched member individually after saving it, select the **Commit each saved member** check box.

   Note: If the conditions for adding unmatched members are not met, a message appears indicating this and the check box is not available.
7. Click **OK**.

Distributed Handling of Unmatched Members

By default, if you attempt to execute more than one fact build simultaneously, and unmatched members are found for the same dimension in the builds, it is likely that the builds will fail as a result of duplicate key errors or data integrity errors.

The following example illustrates this problem. You can see the template table for the Product dimension. The Surrogate ID and Product ID columns accept unique IDs only. Three fact builds, A, B, and C, are executed simultaneously. Builds A and B both contain an unmatched member, P5, and build C contains an unmatched member, P6.



Product Dimension Template Table

Surrogate ID	Product ID
1	P1
2	P2
3	P3
4	P4

Fact Build A

Product ID	Product Name
P1	Product A
P5	Product E

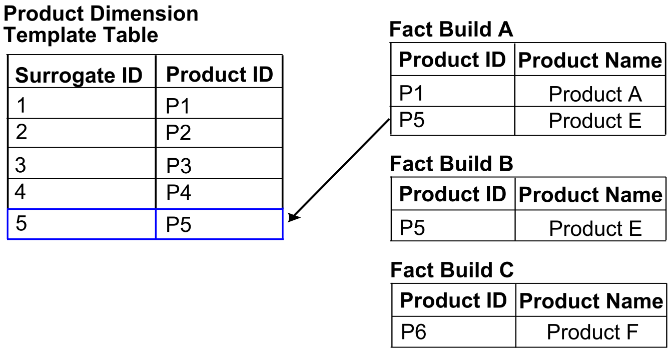
Fact Build B

Product ID	Product Name
P5	Product E

Fact Build C

Product ID	Product Name
P6	Product F

Assuming build A processes P5 before either build B or build C processes its unmatched members, this build will add a new row to the template table, as illustrated.



Product Dimension Template Table

Surrogate ID	Product ID
1	P1
2	P2
3	P3
4	P4
5	P5

Fact Build A

Product ID	Product Name
P1	Product A
P5	Product E

Fact Build B

Product ID	Product Name
P5	Product E

Fact Build C

Product ID	Product Name
P6	Product F

If build B processes next, it will report a duplicate key error on either the Surrogate ID column or the Product ID column when attempting to add the unmatched member P5 to the template table.

Similarly, build C will report a duplicate key error when attempting to add the unmatched member P6 to the template table because it tries to add a new row and insert the value 5 into the Surrogate ID column.

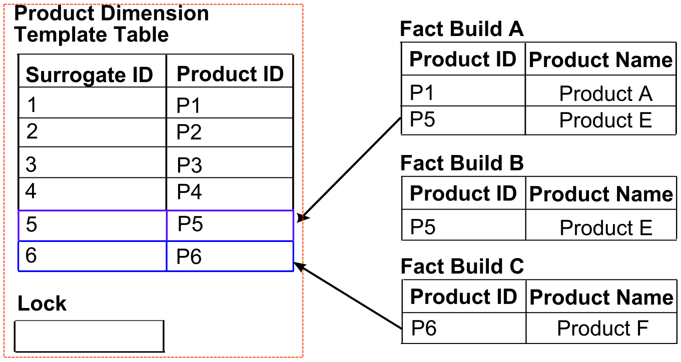
You can overcome these problems by using the distributed handling of unmatched members feature, which involves the use of locks. When executing simultaneous builds, if IBM Cognos Data Manager finds unmatched members in a build, it uses a lock to prevent other builds from accessing the dimension reference data until it has finished adding unmatched members to it. It then releases the lock. When another build containing unmatched members accesses the dimension reference data, it can now match source data to the newly added member. Alternatively, if the data is still unmatched, it can add further unmatched members to the dimension reference data.

In order to add unmatched members using a lock, Data Manager performs the following checks on the template table:

- Determine the existence of an unmatched member.

- Determine the highest surrogate value.

In the example above, using a lock, build A prevents builds B and C from accessing the template table while it adds the unmatched member P5 to it. When it has finished, build B reads the template table and now finds P5, so it no longer reports any errors. In addition, build C can successfully add the unmatched member P6 to the template table.



Product Dimension Template Table

Surrogate ID	Product ID
1	P1
2	P2
3	P3
4	P4
5	P5
6	P6

Fact Build A

Product ID	Product Name
P1	Product A
P5	Product E

Fact Build B

Product ID	Product Name
P5	Product E

Fact Build C

Product ID	Product Name
P6	Product F

If simultaneous builds are executed on multiple computers, a database table must be used as the lock. If simultaneous builds are executed on a single computer, a MUTEX/semaphore can be used instead of a table.

When you create a table for the lock, it must exist on the same connection as the template table. The table must comprise a single numeric column and a single row (unless you are using Netezza, in which case the table must have two columns).

This example shows a single numeric column and a single row:

```
CREATE TABLE product_lock ( lockcol integer );
INSERT INTO product_lock ( lockcol ) VALUES ( 0 );
```

This example shows how two columns could be used for Netezza:

```
CREATE TABLE product_lock ( keycol integer, lockcol integer);
INSERT INTO product_lock ( lockcol ) VALUES ( 0 );
```

Note: You can set up a single lock for all dimensions, or you can have separate locks for each one.

To enable the addition of unmatched members from multiple fact builds, you must

- set the variable named DM_DISTRIBUTED_HANDLING

  For more information, see "DM_DISTRIBUTED_HANDLING" on page 294.
- specify the name of the lock to use when adding unmatched members from distributed fact builds

  For more information, see "Set Up Data Access" on page 75.

Performance Considerations:

Because distributed handling of unmatched members requires additional processing, you should take into consideration how it can affect performance:

- If your source data contains a large number of unmatched members, using exclusive locking can improve performance.

- If your source data contains unmatched members for different dimensions, using separate locks for each dimension, instead of a single lock, can improve performance.

Reference Domains

In IBM Cognos Data Manager, a reference domain is assembled from all the members in the hierarchy or lookup associated with the dimension, regardless of whether the fact data includes them.

You must always use a reference domain for dimensions that have an associated aggregation exception. This is because for each summary data value an aggregation exception returns the value of the first or last child member. Therefore, to identify the first and last child members, Data Manager must be aware of all the dimension members that exist in the domain.

Specify the Domain Details

You can specify a domain size for a dimension element or derived dimension element to size internal transactions when the fact build is executed.

By default, IBM Cognos Data Manager estimates the domain size by referring to the reference structure associated with the dimension or derived dimension. However, if the fact data includes unmatched members, that is, data for which there is no associated reference member, the estimate may be incorrect. Equally, the reference domain may be a large overestimate of the actual domain. In either of these cases, you can enter a better estimate. The domain size should be greater than the maximum number of distinct values in the dimension. If, when the build is executed, the domain size is inadequate, a message appears indicating this, and you should increase the size.

Note: If you want to merge dimension elements that are not associated with a reference dimension, you must set the domain size manually. Set the domain size to be greater than the maximum number of distinct domain members for the dimension.

Procedure

1. Click the required dimension element or derived dimension element.
2. From the **Edit** menu, click **Properties** .
3. Click the **Domain** tab.
4. In the **Domain size** box, type the required domain size.
5. In the **Domain type** box, click the type of domain to use. Remember, for delivery of an aggregation exception for the dimension, you must select **Reference domain**.
6. Click **OK**.

Aggregate Data

In IBM Cognos Data Manager, aggregation is the process of summarizing detail data to levels of a hierarchy or auto-level hierarchy. You can aggregate measure or derivation elements. If you aggregate attribute elements in summarized rows, Data Manager always returns a null value.

You can

- aggregate data to any level of any dimension
- aggregate data simultaneously over a number of levels (multi-level aggregation), a number of dimensions (multi-dimensional aggregation), or both
- exclude detail data from the output to provide compact summary data collections
- include or exclude individual levels

  For example, you can include every conceivable combination of summary data of an in-depth business analysis, or just a high-level summary for management reporting.

Before you begin

Before you can aggregate data, you must

- determine the aggregation levels
- specify that aggregation is to be performed
- set up the regular aggregate
- set up aggregate rules, if required

Memory Considerations

In IBM Cognos Data Manager, aggregation is performed in memory, by default. You can limit the amount of memory that Data Manager uses. For example, when you aggregate large amounts of data, you may need to introduce breaking to control the aggregation and merge process.

For more information about managing memory, see Chapter 29, “Managing Memory,” on page 343.

Determine the Aggregation Levels

IBM Cognos Data Manager can build as many levels of output data as necessary.

For example, a fiscal calendar where months roll to quarters and quarters roll to years means that quarterly and yearly data can be generated from the monthly data. In a multi-dimensional model, levels in all dimensions can be aggregated in this way, including all the combinations of levels across dimensions. Aggregating everything can lead to data explosion; that is, a moderate starting volume of data results in a huge final volume.

There are many factors to consider when deciding whether to aggregate to a level, many of which will be business related. For example, ''Should I aggregate to the market segment level when it gets reorganized every other day?''

Specify that Aggregation is to be Performed

Aggregation cannot be performed until you have specified a reference dimension and reference structure for the dimension element or derived dimension.

Procedure

1. Click the dimension element or the derived dimension element.
2. From the **Edit** menu, click **Properties** .
3. Click the **Reference** tab.
4. Check that you have specified the reference dimension and reference structure to use.

For information, see “Associate a Dimension or Derived Dimension Element with a Reference Item” on page 146.

5. Select the **Aggregate** check box to allow aggregation to be performed.
6. Click **OK**.

Calculate a Derivation Before Aggregation

By default, a derivation element is calculated after aggregation. However, you can choose to perform calculation before aggregation; that is, from the input values.

You can also add derivations before transformation, in the data sources and DataStream of a fact build. For information, see “Derivations” on page 128.

Procedure

1. Click the derivation.
2. From the **Edit** menu, click **Properties** .
3. Click the **General** tab.
4. Select the **Calculate at source** check box.
5. Click **OK**.

Set Up the Regular Aggregate

The regular aggregate is applied to the appropriate measure or derivation element when aggregating over any dimension for which there is no aggregate rule defined. By default, IBM Cognos Data Manager aggregates over these dimensions before aggregating over any dimensions with aggregate rules.

For more information about aggregate rules, see “Set Up an Aggregate Rule” on page 163.

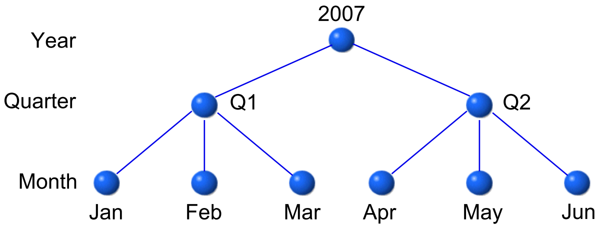
You can aggregate data in a number of different ways. For example, you may want to find the sum of all detail members (SUM) or the average of all these members (AVG). Data Manager assigns SUM as the default regular aggregate.

The following aggregate functions can be used for the regular aggregate.

Function	**Description**
SUM	Calculates a summary data value by adding together the detail values.
MIN	Assigns the minimum detail value to the summary data value.
MAX	Assigns the maximum detail value to the summary data value.
AVG	Calculates a summary data value by taking the average of the detail values.
COUNT	Assigns the count of the number of detail values to the summary data value.
COUNT DISTINCT	Assigns the count of the number of unique detail values to the summary data value.

Function	Description
VAR	Calculates a summary data value by taking the variance of the detail values.
STDDEV	Calculates a summary data value by taking the standard deviation of the detail values.
FIRST	Takes the value from the first detail row. **Note:** This differs from using FIRST for an aggregate rule, which takes the value corresponding to the earliest row that occurs in the reference structure.
FIRST NON-NULL	Takes the value from the first non-null detail row.
LAST	Takes the value from the last detail row. **Note:** This differs from using LAST for an aggregate rule, which takes the value corresponding to the latest row that occurs in the reference structure.
LAST NON-NULL	Takes the value from the last non-null detail row.
ANY	Sets the summary value to one if there are any detail values. Otherwise, the summary value is set to zero.
NONE	Sets the summary value to NULL regardless of any detail values.

The following example illustrates the FIRST and LAST aggregate functions for the regular aggregate. Suppose you have the following Time hierarchy.



Your incoming data looks like this.

Month	Product	Quantity
Jun	Lantern	10
Feb	Lantern	NULL
Mar	Tent	50
Apr	Lantern	30
Jan	Lantern	40

If you aggregate Quantity over Time to the Quarter level using the FIRST aggregate function, these are the results.

Quarter	Product	Quantity
Q1	Lantern	NULL
Q1	Tent	50
Q2	Lantern	10

Data Manager calculates the FIRST regular aggregate according to the order of the incoming data. So in Q1, Feb is the first record found for Lantern, and Mar is the first record found for Tent. Similarly, in Q2, Jun is the first record found for Lantern.

If you aggregate the same data using the FIRST NON-NULL aggregate function, the results are different.

Quarter	Product	Quantity
Q1	Lantern	40
Q1	Tent	50
Q2	Lantern	10

In this case, Data Manager ignores the Feb record for Lantern because it contains a null value, and finds Jan as the first record instead. All other records remain unchanged because there are no other null values.

Similarly, aggregating Quantity over Time to the Quarter level using the LAST aggregate function produces these results.

Quarter	Product	Quantity
Q1	Lantern	40
Q1	Tent	50
Q2	Lantern	30

In Q1, Jan is the last record found for Lantern, and Mar is the last record found for Tent. For Q2, Apr is the last record found for Lantern.

In this case, aggregating this data using the LAST NON-NULL aggregate function produces the same results because there are no null values.

Procedure

1. Click the measure or derivation element for which to perform aggregation.
2. From the **Edit** menu, click **Properties** .

   To perform aggregation for a derivation, you must select the **Calculate at source** check box. For more information, see "Calculate a Derivation Before Aggregation" on page 160.
3. Click the **Aggregation** tab.
4. In the **Regular aggregate** box, click the required aggregate function.
5. Click **OK**.

Set Up an Aggregate Rule

By default, the regular aggregate is applied across every dimension. However, if a measure or derivation element is semi-additive, there may be dimensions across which the function produces meaningless results, depending which aggregate function is used for the regular aggregate.

For example, *inventory* and *head count* are not additive over Time because they represent a snapshot value for a specific point in time. If SUM is used as the aggregate function for the regular aggregate, and a Time dimension is present, an alternative way of aggregating over Time is required.

In IBM Cognos Data Manager, an aggregate rule is the mechanism for specifying alternatives on a dimension-by-dimension basis.

To illustrate further, the following data includes an inventory value that represents the stock held at a particular time.

Month	Product	Inventory	Location
Jan	Lantern	50	London
Jan	Tent	0	Paris
Feb	Lantern	50	London
Feb	Tent	50	Paris
Mar	Lantern	150	London
Mar	Tent	50	London

If you define the regular aggregate to SUM Inventory over Product, Location, and Time, the summary inventory value over time is meaningless. There are three consecutive months in which the stock information for Lantern is 50, 50, and 150 units. The sum for the quarter would be calculated as 250 units, which is much higher than the stock level at any time. This is clearly incorrect.

As an alternative, you can set up an aggregate rule for the Time dimension. For example, you could use the LAST aggregate function to select the last available month in the quarter as the summary value for the quarter.

Aggregate rules are used in addition to the regular aggregate and are applied in the following order:

1. The regular aggregate function is applied to all dimensions for which there is no aggregate rule defined.
2. Aggregate rules are applied to the specified dimensions, in sequential order.

Notes

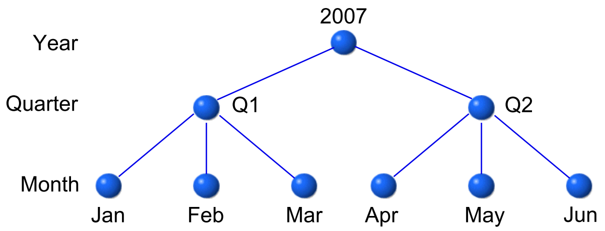
- You can set up a separate aggregate rule for each dimension, but each dimension can only contain a single rule.
- The order of aggregate rules affects calculations because the result of each rule in a sequence is passed to the next rule.
- If an aggregate rule is set up for every dimension, the regular aggregate function is not applied.
- Aggregate rules cannot be used with multi-level input data or dimension breaks.
- If you are using aggregate rules with duplicate input data, you cannot allow records with duplicate keys. You must reject, merge or aggregate them.

For more information, see “Handle Duplicate Data” on page 112.

The following aggregate functions can be used for an aggregate rule.

Method	Description
SUM	Calculates a summary data value by adding together the detail values.
MIN	Assigns the minimum detail value to the summary data value.
MAX	Assigns the maximum detail value to the summary data value.
AVG	Calculates a summary data value by taking the average of the detail values.
FIRST	Takes the value corresponding to the earliest row that occurs in the reference structure. **Note:** This differs from using FIRST for the regular aggregate, which takes the value from the first detail row.
LAST	Takes the value corresponding to the latest row that occurs in the reference structure. **Note:** This differs from using LAST for the regular aggregate, which takes the value from the first detail row.
COUNT	Assigns the count of the number of detail values to the summary data value.
COUNT DISTINCT	Assigns the count of the number of unique detail values to the summary data value.
VAR	Calculates a summary data value by taking the variance of the detail values.
STDDEV	Calculates a summary data value by taking the standard deviation of the detail values.
ANY	Sets the summary value to one if there are any detail values. Otherwise, the summary value is set to zero.
NONE	Sets the summary value to NULL regardless of any detail values.

The following example illustrates the FIRST and LAST aggregate functions for the aggregate rule. Suppose you have the following Time hierarchy.



Your incoming data looks like this.

Month	Product	Quantity
Jun	Lantern	10
Feb	Lantern	40
Mar	Tent	50
Apr	Lantern	30
Jan	Lantern	NULL

If you aggregate Quantity over Time to the Quarter level using the FIRST aggregate function, these are the results.

Quarter	Product	Quantity
Q1	Lantern	NULL
Q1	Tent	50
Q2	Lantern	30

In Q1, Jan is the first row that occurs in the Time hierarchy for which a record exists for Lantern, and Mar is the first row that occurs for which a record exists for Tent. Likewise, in Q2, Apr is the first row for which a record exists for Lantern.

Similarly, aggregating Quantity over Time to the Quarter level using the LAST aggregate function, produces these results:

Quarter	Product	Quantity
Q1	Lantern	40
Q1	Tent	50
Q2	Lantern	10

In Q1, Feb is the last row that occurs in the Time hierarchy for which a record exists for Lantern, and Mar is the last row that occurs for which a record exists for Tent. In Q2, Jun is the last row for which a record exists for Lantern.

Procedure

1. Click the appropriate measure or derivation element.
2. From the **Edit** menu, click **Properties** .
3. Click the **Aggregation** tab.
4. Click **Add**.
5. Select the dimension for which you want to set up an aggregate rule, and then click **OK**.

6. Click the required aggregate function from the **Rule aggregate** drop down list.
7. Repeat steps 4 to 6 to add more aggregate rules.
8. Click **OK**.

   Tips:

   - The order in which aggregate rules are shown reflects the order in which they are applied. To change the position of a specific aggregate rule, select it and click either **Up** or **Down**.
   - To delete an aggregate rule, select it and click **Delete**.

Aggregate Duplicate Data

When aggregating data, duplicate keys in the input data can produce different results depending on how the duplicates are handled:

- **Allow records with duplicate keys** has no cost in terms of memory or performance. It can be used when detail data is not being delivered or you know that duplicates will not occur. If it is used under other circumstances, IBM Cognos Data Manager may deliver multiple detail data rows rather than a single aggregated detail row.
- **Reject records with duplicate keys** can be used under any circumstances because it ensures that only one row with any duplicate key is passed to the aggregation phase.
- **Merge records with duplicate keys** is performed as a separate phase prior to the aggregation phase.

  Tip: To ensure you obtain the results you require, compare the results of merging duplicate input data with aggregating it.
- **Aggregate records with duplicate keys** passes duplicate input data directly to the aggregation phase.

For more information about duplicate data, see “Handle Duplicate Data” on page 112.

Note: If you are aggregating data using aggregate rules, you cannot allow records with duplicate keys. You must reject, merge or aggregate them.

This example illustrates the difference between merging and aggregating input data with duplicate keys. You want to aggregate the following data to the Product Type level (Camping) using the COUNT aggregate function.

Product	Units
Lantern	40
Lantern	70
Lantern	20

If you choose to merge records with duplicate keys, Data Manager merges the three records for Lantern into a single record.

Product	Count
Lantern	3

Data Manager passes the results of the merge phase to aggregation phase. Aggregating the data to the Product Type level produces the following results.

Product	Count
Lantern	3
Camping	1

If you choose to aggregate records with duplicate keys, aggregating the same data produces different results (shown below) because Data Manager performs only the aggregation phase.

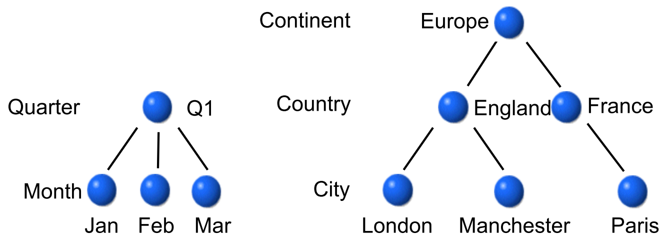
Product	Count
Lantern	3
Camping	3

Procedure

1. Click the required fact build.
2. From the **Edit** menu, click **Properties** .
3. Click the **Input** tab.
4. Select the option to perform on records with duplicate keys.
5. Click **OK**.

Regular Aggregate Examples

Suppose you have Time and Location hierarchies.



The detail data looks like this.

Month	City	Revenue
Jan	London	2,000
Jan	Manchester	1,000
Jan	Paris	1,500
Feb	London	2,500
Feb	Manchester	2,000
Feb	Paris	1,000
Mar	London	3,000

Month	City	Revenue
Mar	Paris	2,000

Example 1

In this example, you aggregate data to Country level using the SUM aggregate function. That is, the revenue for London and Manchester is combined to make England and, because the data only contains information about one city in France (Paris), the revenue for Paris simply becomes France.

Month	Country	Revenue
Jan	England	3,000
Jan	France	1,500
Feb	England	4,500
Feb	France	1,000
Mar	England	3,000
Mar	France	2,000

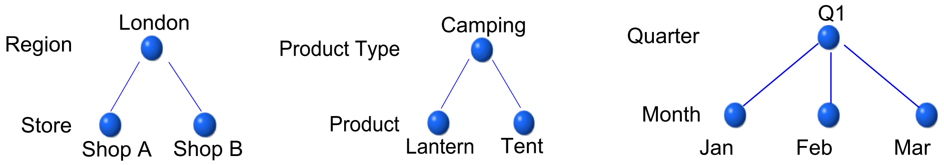
Example 2

In this example, you aggregate data to the Quarter level and the European level using the SUM aggregate function. That is, the revenue for Jan, Feb, and Mar is combined to obtain a revenue value for each level at Q1.

Quarter	Location	Revenue
Q1	London	7,500
Q1	Manchester	3,000
Q1	Paris	4,500
Q1	England	10,500
Q1	France	4,500
Q1	Europe	15,000

Aggregate Rules Example

You have Location, Product, and Time hierarchies.



The detail data looks like this.

Month	Store	Product	Quantity
Jan	Shop A	Lantern	50
Jan	Shop B	Lantern	80
Jan	Shop A	Tent	150
Jan	Shop B	Tent	10
Mar	Shop A	Lantern	40
Mar	Shop B	Lantern	30
Mar	Shop A	Tent	60

You want to aggregate this data as follows:

1. Find the total number of products sold in each region.
2. Calculate the average number of products sold for each product type.
3. Determine the maximum quantity of each product sold in each quarter.

To do this, you set up a regular aggregate using the SUM aggregate function. You also need to set up two aggregate rules: one to aggregate the Product dimension using the AVG aggregate function, and a second to aggregate the Time dimension using the MAX aggregate function.

The regular aggregate (aggregate Quantity over Location to the Region level using the SUM aggregate function) is the first calculation to be performed.

Month	Region	Product	Total Quantity
Jan	London	Lantern	130
Jan	London	Tent	160
Mar	London	Lantern	70
Mar	London	Tent	60

These results are now used in all subsequent calculations.

The first aggregate rule (aggregate Quantity over Product to the Product Type level using the AVG aggregate function) is the next calculation.

Month	Store	Product Type	Average Quantity
Jan	Shop A	Camping	100
Jan	Shop B	Camping	45

Month	Store	Product Type	Average Quantity
Jan	London	Camping	145
Mar	Shop A	Camping	50
Mar	Shop B	Camping	30
Mar	London	Camping	65

These results are passed through to the final aggregate rule (aggregate Quantity over Time to the Quarter level using the MAX aggregate function).

Month	Store	Product Type	Maximum Quantity
Q1	Shop A	Lantern	50
Q1	Shop B	Lantern	80
Q1	Shop A	Tent	150
Q1	Shop B	Tent	10
Q1	London	Lantern	130
Q1	London	Tent	160
Q1	Shop A	Camping	100
Q1	Shop B	Camping	45
Q1	London	Camping	145

Notice that the maximum quantity for some rows is higher than the quantity sold for any product in the input data. For example, in row 6, the maximum quantity of Tents sold is 160. This value represents the maximum of the *total quantity* of Tents sold.

Process Late Arriving Facts

IBM Cognos Data Manager normally delivers fact data by using the business key and retrieves only the current reference record from the reference data. If the fact data contains late arriving facts, the result may be that fact records are delivered with incorrect reference data.

For example, if sales records are delivered at the end of a month, and a reference record was updated during that month, fact records will be delivered using the current reference data.

By using late arriving fact processing, data can be delivered using the surrogate key instead of the business key. This has the advantage that, if the reference data tracks attribute changes, the history of changes to the data can be retrieved.

Note: You must load the dimensional history before you can process late arriving facts. For information, see “Set Up Dimensional History” on page 214.

You can specify how late arriving facts are processed, so that fact records can be delivered using the appropriate surrogate key:

- If the timestamp in a fact record falls within the range of dates for the corresponding reference data, the appropriate data is selected from the reference data and the record is processed. You do not need to specify any further action to be performed on the record.

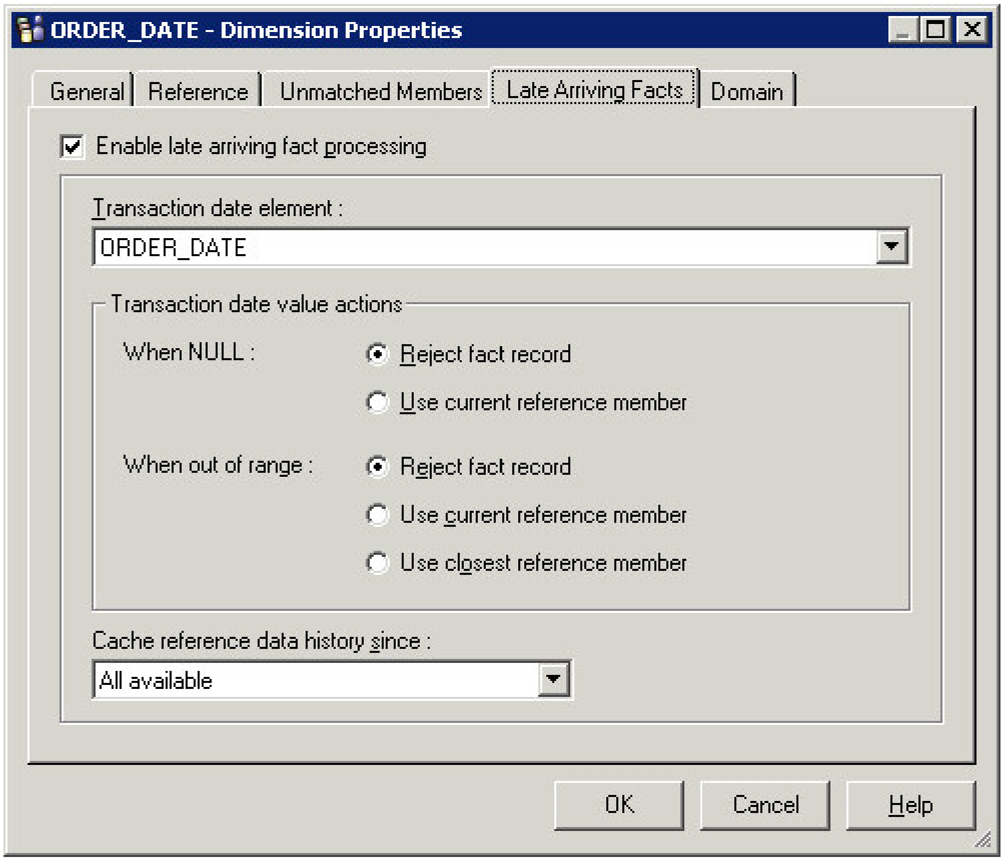
- If the timestamp in a fact record contains a null value, you can choose one of the following actions to perform on the record:
  - Reject fact record (default)
  - Use current reference member
- If the timestamp in a fact record is outside of the effective date range for the reference data, you can select one of the following actions to perform on the record:
  - Reject fact record (default)
  - Use current reference member
  - Use closest reference member

To allow late arriving fact processing on a dimension, the following conditions must be met:

- The fact build must not merge data.
- The fact build must not include any dimension deliveries for the dimension.
- The dimension must use a lookup.
- The lookup must include an attribute with effective start date behavior.
- The lookup must use a template for data access.

Procedure

1. Click the dimension element or derivation element for which you want to specify how to process late arriving facts.
2. From the **Edit** menu, click **Properties** .
3. Click the **Late Arriving Facts** tab.



☑ Enable late arriving fact processing

Transaction date element : ORDER_DATE

Transaction date value actions

When NULL : ● Reject fact record ○ Use current reference member

When out of range : ● Reject fact record ○ Use current reference member ○ Use closest reference member

Cache reference data history since : All available

Note: If the conditions for processing late arriving facts are not met, a message appears indicating this, and the following steps are not available.

4. Select the **Enable late arriving fact processing** check box.
5. In the **Transaction date element** box, click the transformation model element that corresponds to the timestamp value in the fact records.
6. In the **Transaction date value actions** box, specify the action to perform on records containing late arriving facts.
7. The **Cache reference data history since** box allows you to limit the records processed for late arriving facts to a specific date range. This restricts the domain of reference members to include all current attribute change tracking records and those non-current records that are later than the specified date. Click
   - **All available** to specify that all available records should be processed
   - **Current record only** to specify that only current records should be processed
   - **Variable** to use the variable from which to obtain the date and time, and then in the adjacent box, type the variable name

     Note: The variable must be defined on the **Variables** tab. For information, see Chapter 24, “Variables,” on page 289.
   - **Date (yyyy-mm-dd hh:mm:ss)** to specify an explicit date and time from which records should be processed, and then in the adjacent box, type the date and time
8. Click **OK**.

Late Arriving Fact Example

Monthly sales figures for product P1 contain the date of each sale.

Period	Product	Units Sold	Date of Sale
200510	P1	1	2005/10/31
200601	P1	5	2006/01/31
200602	P1	13	2006/02/27
200603	P1	1	<null>

The following table illustrates Product reference data, where changes to the Specification attribute are being tracked.

Product Code	Surrogate Key	Specification	Effective Date	End Date
P1	1	Blue pen	2005/11/01	2005/12/22
P1	2	Black pen	2005/12/23	2006/02/09
P1	3	Red pen	2006/02/10	

You can see that the specification for product P1 has changed since it was first defined and last changed on 2005/02/10. The range of dates for the Product reference data is 2005/11/01 to the current date.

Suppose all sales figures for the period October 2005 to March 2006 are delivered in March 2006. Because the product reference data has changed during this time period, the sales figures contain late arriving facts.

The timestamp for record 2 is 2006/01/31, which falls within the range of dates specified for the Product reference data. The fact record is delivered to the data mart with surrogate key 2.

The timestamp for record 4 is null. If you choose to use the current reference record, the fact record is delivered with surrogate key 3.

The timestamp for record 1 is 2005/10/31, which is outside the effective date range for the Product reference data. If you

- use the current reference record, the fact record with surrogate key 3 is delivered
- use the closest reference record, the fact record with surrogate key 1 is delivered

Merge Data for a Measure or an Attribute

Merging is useful where more than one input record can have the same set of dimension values. It is also useful for merging disparate input data so that similar key sets of figures can be compared, for example, forecast and actual sales figures can be compared.

By default, IBM Cognos Data Manager assigns a merge method of SUM to measures and FIRST NON-NULL to attributes. However, you can change this by choosing from these merge methods.

Method	**Description**
SUM	Adds together the duplicate detail values.
MAX	Takes the detail value with the maximum value.
MIN	Takes the detail value with the minimum value.
COUNT	Counts the detail value members.
COUNT DISTINCT	Assigns the count of the number of unique detail values to the summary data value.
AVG	Averages the detail values.
FIRST	Takes the value from the first detail row.
FIRST NON-NULL	Takes the value from the first non-null detail row.
LAST	Takes the value from the last detail row.
LAST NON-NULL	Takes the value from the last non-null detail row.
ANY	Sets the summary value to one if there are any detail values. Otherwise, the summary value is set to zero.

Procedure

1. Click the measure or attribute element for which you want to perform merging.
2. From the **Edit** menu, click **Properties** .
3. Click the **Merge** tab.
4. In the **Merge behavior** box, click the required merge method.

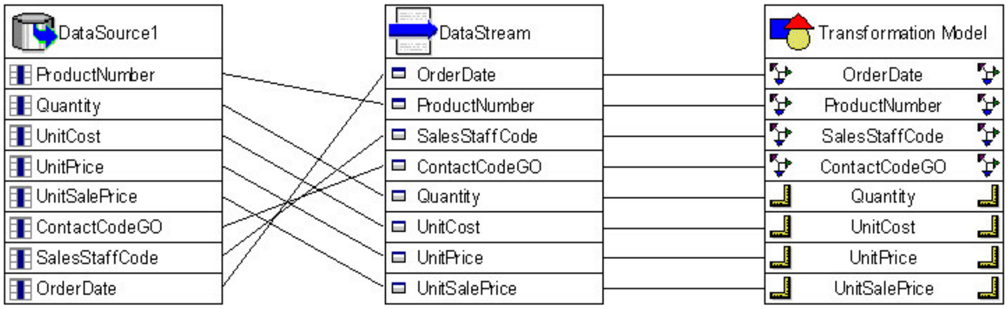
   Note: If the data is coming from multiple data sources, a merge operation is performed first, and then the transformation is specified in the fact build.
5. Click **OK**.
6. Click the fact build that contains the measure or attribute element for which you want to perform merging.
7. From the **Edit** menu, click **Properties** .
8. Click the **Input** tab.
9. Click **Merge records with duplicate keys**.
10. Click **OK**.

The **Visualization** pane is refreshed to show graphically that duplicate records are to be merged.

Map the DataStream to the Transformation Model

To enable the flow of data from a data source to the transformation model, you must map the acquired data source to the DataStream, and then map the DataStream to existing transformation model elements or map the DataStream and create transformation model elements.

The following example illustrates this data flow.



Map the DataStream to Existing Transformation Model Elements

After you add transformation model elements to the fact build, you must map DataStream items to them.

Tip: You can create attribute, measure, and dimension elements while performing mapping. For information, see "Map the DataStream and Create Transformation Model Elements" on page 176.

Procedure

1. Click the appropriate transformation model.
2. From the **Edit** menu, click **Mapping**.

   The **Transformation Model Mapping** window appears.
3. In the **Transformation Model** pane, click the transformation model element to map.

   If you are mapping a dimension element, expand the element and reference structure, and then click the required attribute ID.
4. In the **DataStream Item** column, click the DataStream item to map.
5. Click **Map**.

   Tip: You can also perform mapping by dragging the transformation model element to the relevant position in the **Maps To** column.
6. Repeat steps 3 to 5 to complete the required mapping.
7. Click **OK**.

   TipsTo clear a mapping, click the mapped item in the **Maps To** column, and then click **Clear**.

   To clear all the mappings, click **Clear All**.

   To add a transformation model element, click **Add**, and then click the type of element to create.

To edit the name of a transformation model element, click the element in the **Transformation Model** pane, and then click **Edit**.

To delete a transformation model element, click the element in the **Transformation Model** pane, and then click **Delete**.

Map the DataStream and Create Transformation Model Elements

Instead of creating transformation model elements and then mapping the DataStream items to the transformation model separately, you can perform both tasks together.

You can create attribute, dimension, and measure elements using this method, but not derivations or derived dimensions.

Procedure

1. Click the appropriate transformation model.
2. From the **Edit** menu, click **Mapping**.

   The **Transformation Model Mapping** window appears.
3. In the **DataStream Item** column, click the DataStream item that you want to map to the transformation model.
4. Drag the selected item to the white space in the **Transformation Model** column.

   Note: Be careful not to drag the item to any transformation model elements because this maps the DataStream item to the selected transformation model element.

   A menu appears prompting you to choose the element type to create.
5. Click the element type to create.

   Note: If a matching transformation model element already exists, a new element is not created, and the DataStream item is mapped to the existing one.

   The transformation model element is created and the selected item is mapped to the element.

   If you choose to create a dimension element, the element is not mapped. You must associate it with a reference item first.
6. Repeat steps 3 to 5 to complete the required mapping.
   - To create multiple transformation model elements of the same type, hold down the Ctrl key and click each DataStream item, and then repeat steps 4 and 5.
   - To add a transformation model element, click **Add**, and then select the type of element to create.
   - To edit the name of a transformation model element, click the element in the **Transformation Model** pane, and then click **Edit**.
   - To delete a transformation model element, click the element in the **Transformation Model** pane, and then click **Delete**.
7. Click **OK**.

   The **Add New Elements** window appears.
8. For each listed element, select the **Add to Fact Deliveries** check box to add the element to each fact delivery in the fact build.
9. Click **OK**.

Map the DataStream to Transformation Model Elements Automatically

You can map all unmapped DataStream items in a single step.

If transformation model elements exist with matching names, the DataStream items are automatically mapped to them. For example, if you have a DataStream item named OrderDate, and a transformation model element named OrderDate, they are automatically mapped.

If no matching transformation model elements exist, they are automatically created during the mapping process.

Procedure

1. Click the appropriate transformation model.
2. From the **Edit** menu, click **Mapping**.

   The **Transformation Model Mapping** window appears.
3. Click **Auto Map**.

   If elements need to be created, a message appears prompting you to choose the element type to create.

   Note: All new elements are created with the same type.

   All the unmapped DataStream items are mapped.

   If you choose to create dimension elements, IBM Cognos Data Manager does not map the element. You must associate it with a reference item first.
4. Click **OK**.

   If any new elements have been created, the **Add New Elements** window appears.
5. For each listed element, select the **Add to Fact Deliveries** check box to add the element to each fact delivery in the fact build
6. Click **OK**.

Add an Element to the Deliveries

You can automatically add transformation model elements to the deliveries for the fact build.

Procedure

1. Either add a new element to a fact build, or click an element created previously and, from the **Actions** menu, click **Add to Delivery Modules**.

   The **Deliver <element_name>** dialog box appears.
2. To add the element to each delivery module in the fact build, click **Yes**.

Change an Element Type

You can change the element type of any transformation model element.

Procedure

1. Click the transformation model element to change to a different type.
2. From the **Actions** menu, click **Convert**, followed by the relevant option.

Repositioning Elements

The order in which the transformation model elements are shown in the Tree pane is reflected in the fact columns identified in the fact deliveries.

This affects the column order in the target table if it is created as part of the fact build, but does not affect the ability of IBM Cognos Data Manager to deliver data to the target table.

Another consideration when positioning elements is that when you create an expression for a derived dimension, in addition to using any measure, attribute, or dimension element in the fact build, you can also use any derived dimension that is before it in the fact build.

Reposition elements from the tree pane

You can reposition elements from the tree pane or from the transformation model mapping window. Use the method you prefer to reposition elements.

Procedure

1. In the **Tree** pane, click the transformation model element to reposition.
2. From the **Actions** menu, click either **Move Up** or **Move Down**.
3. Click **OK**.

   Tip: You can also reposition an element by clicking the element in the **Tree** pane and dragging it to the required position.

Reposition elements from the transformation model mapping window

You can reposition elements from the transformation model mapping window or from the tree pane. Use the method you prefer to reposition elements.

Procedure

1. Click the appropriate transformation model.
2. From the **Edit** menu, click **Mapping**.

   The **Transformation Model Mapping** window appears.
3. In the **Transformation Model** pane, click the transformation model element to reposition.
4. Click either **Move Up** or **Move Down**.
5. Click **OK**.

Delete an Element

You can delete a transformation model element that is no longer required.

Procedure

1. Click the required element.
2. From the **Edit** menu, click **Delete**.
3. In the message box, click **Yes**.

Chapter 16. Delivering Fact Data Using a Fact Build

In IBM Cognos Data Manager, you use delivery modules to deliver transformed data to the specified data repositories.

Data Manager organizes the delivery modules into these groups:

- fact delivery modules that deliver data produced by a fact build
- dimension delivery modules that deliver data describing a single dimension

Each fact build can have as many deliveries as you require.

Fact Delivery Modules

Fact deliveries deliver the fact data that a fact build produces.

You can deliver

- data to different tables within the same or different target databases, and to different OLAP targets.
- specific transformation model elements to multiple tables in the same or different databases by subscribing to elements.
- specific data rows by applying an output filter.
- specific dimensions and hierarchy levels by applying one or more level filters.
- columns that are only occasionally used in queries into a separate table by partitioning the data.

A number of fact delivery modules are available. These include delivery modules for relational tables, native bulk loaders for relational databases, OLAP targets, and text files. Each of these modules has different configuration options and is responsible for converting fact data into an appropriate format for the target.

The fact delivery modules fall into two groups, those that deliver the data to tables, and those that do not.

Table Delivery Modules

The following fact delivery modules deliver data to tables.

- DB2 LOAD
- Informix® LOAD
- Microsoft SQL Server BCP Command
- Netezza NZLoad
- ORACLE SQL *Loader
- Red Brick® Loader (TMU)
- Relational Table
- Microsoft SQL Server Bulk Copy via API
- Sybase ASE
- Sybase IQ
- Teradata Fastload
- Teradata Multiload

- Teradata TPump

Non-table Delivery Modules

The following fact delivery modules do not deliver data to tables.

- Essbase Direct Feed
- Essbase File Feed
- Text File
- TM1 Turbo Integrator

Each delivery module has a set of properties. Module properties apply to the delivery as a whole, for example, the table name of a target relational table. Element properties apply to a delivered element. Not all delivery elements have the same properties. The relevance of each property depends on the element type with which it is associated, that is, whether it is a dimension, attribute, measure, or derivation.

Add a Fact Delivery

You can add fact deliveries to a fact build manually or while creating a fact build using the Fact Build wizard.

For information about adding a fact delivery using the Fact Build wizard, see "The Fact Build Wizard" on page 107.

Procedure

1. In the appropriate fact build, expand **Delivery Modules** and click **Fact Delivery**.
2. From the **Insert** menu, click **Build**, and then click **Fact Delivery**.
3. From the **Fact Delivery** menu, click the delivery module that you want to add.

   The **Properties** window for the selected delivery module appears.
4. Click the **General** tab.

   Note: The set of tabs in this window varies according to the fact delivery module that you select.
5. Type a name for the delivery and, if required, a business name and description.
6. If you do not want to deliver the same data rows to any deliveries that are after this one in the top-down order of the delivery tree, select the **Exclusive** check box.
7. If you want to exclude the delivery when the build is executed, clear the **Enabled** check box.

   You can now temporarily disable the delivery without having to delete it, and later recreate it.
8. Click the **Module Properties** tab and enter values as required.

   For information about the properties for each delivery module, see Appendix A, "Fact Delivery Module Properties," on page 413.
9. Click **OK**.

Define the Properties for a Table Delivery

Delivery modules that deliver to tables have different properties than delivery modules that do not deliver to tables.

If you select a delivery module that delivers data to tables, you can

- specify the target database and target tables
- specify how indexing is to be performed
- subscribe to and define the properties for transformation model elements
- change column names
- deliver columns across more than one table
- automatically add any columns that are missing from the target table
- specify how to handle failed data in a table delivery
- filter the delivery data
- partition the data so that columns that are only occasionally used in queries are delivered into separate tables

Procedure

1. Click the fact delivery for which you want to set the table properties.
2. From the **Edit** menu, click **Properties** .
3. Click the **Table Properties** tab.
4. In the **Connection** box, click the connection that corresponds to the database in which the table resides.
5. In the **Table name** box, enter the name of the target table. This value is mandatory.

   If you enter the name of a table that does not exist, IBM Cognos Data Manager creates it.

   Tip: You can also click the **Browse for Table** button and choose the required table from the list in the **Select Table** dialog box.
6. Select the check boxes to the left of the transformation model elements to specify the elements to which the delivery subscribes, that is, which elements are to be delivered.

   For more information, see “Delivering to Multiple Tables by Subscribing to Elements” on page 184.
7. To change the name of a column, click the column name, and type the new name.

   For more information, see “Change the Column Names of Delivered Data” on page 186.
8. Select the check boxes in the **Key** column to indicate which columns form part of the primary key of the table.

   If there are dimension elements included in the delivery table, by default, Data Manager selects them all as the key columns if no other columns are currently selected. To use specific dimension elements only, select the relevant check boxes.

   If there are no dimension elements in the delivery table, and you do not specify your own key columns, Data Manager cannot determine which columns already exist. They are therefore are treated as columns to be inserted.

For information about key columns for the Relational Table delivery module, see "Primary Key Settings for Update/Insert" on page 430 and "Primary Key Settings for Update" on page 438.

9. Select the check boxes in the **Update** column to indicate which columns Data Manager should update in the target table.

   By default, when Data Manager performs an update, it updates all columns in the target table, with the exception of any columns identified as key columns.

   - You cannot select a column that is identified as the key. If you need to update it, you must clear the **Key** column check box first.
   - Performing an update only applies if you use the Relational Table or the Red Brick Loader (TMU) delivery modules.
10. To create an index for all the dimension columns in the table (a composite index), select the **Index** check box.

    You can define the properties for the index by clicking **Index Properties**.

    For more information, see "Define the Index Properties."
11. To create an index for an element (as opposed to creating a composite index for the whole delivery), select the relevant check box in the **Index** column.

    You can define the properties for the index by clicking the **Browse** button ... in the **Index Properties** column.

    For more information, see "Define the Index Properties."
12. If you want Data Manager to automatically add missing columns to the target table, select the **Automatically add columns to table** check box.

    For more information, see "Add Missing Columns to the Target Table" on page 187.
13. To define the properties for an element, click the element, and then click the button in the **Element Properties** column.

    For more information, see "Define the Properties for a Subscribed Element" on page 185.
14. To add control attributes to the table, select the relevant check boxes as required:
    - **Create Date**
    - **Last Update Date**
    - **Record Identity**

    For more information, see "Add Control Attributes to a Table Delivery" on page 184.
15. Click **OK**.

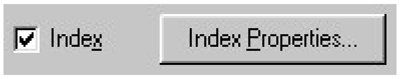
Define the Index Properties

You can define the index properties for all the dimension columns in a table (a composite index), or for a single transformation model element (an element index).

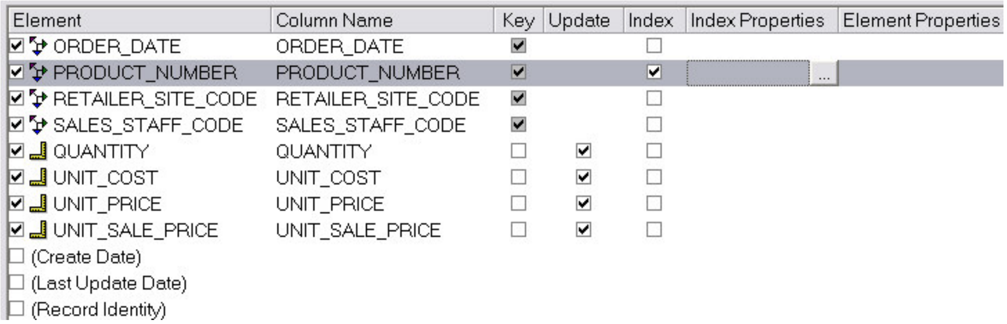
Procedure

1. Click the fact delivery for which you want to set the table properties.
2. From the **Edit** menu, click **Properties** .
3. Click the **Table Properties** tab.
4. Define the properties for a composite index, an element index, or both.

- To create an index for all the dimension columns in the table (a composite index), select the **Index** check box, and then click **Index Properties**.



- To create an index for a single element, as opposed to creating a composite index for the whole delivery, select the relevant check box in the **Index** column, and then click the **Browse** button ... in the **Index Properties** column.



Element	Column Name	Key	Update	Index	Index Properties	Element Properties
☑ ORDER_DATE	ORDER_DATE	☑		□		
☑ PRODUCT_NUMBER	PRODUCT_NUMBER	☑		☑	...	
☑ RETAILER_SITE_CODE	RETAILER_SITE_CODE	☑		□		
☑ SALES_STAFF_CODE	SALES_STAFF_CODE	☑		□		
☑ QUANTITY	QUANTITY	□	☑	□		
☑ UNIT_COST	UNIT_COST	□	☑	□		
☑ UNIT_PRICE	UNIT_PRICE	□	☑	□		
☑ UNIT_SALE_PRICE	UNIT_SALE_PRICE	□	☑	□		
□ (Create Date)						
□ (Last Update Date)						
□ (Record Identity)						

The **Index Properties** dialog box appears.

5. In the **Name** box, type a name for the index.
6. In the **Type** box, click the type of index to create:
   - Click **Unique** for an index that constrains the dimension data so that each value in the indexed column is unique.
   - Click **Bitmap** for an index that is of a compressed format, which some database management systems implement. Using an index of this type can enhance search performance.

   For more information, see the manufacturer's documentation for the type of database to which you are delivering.

   If the **Type** box is empty, the delivery creates the default index type for the target DBMS.
7. In the **Options** box, type any additional options for the index. The contents of this box are appended to the CREATE INDEX statement.

   For information about valid options, see the documentation for the target DBMS.
8. To continue generating the index if errors are encountered, select the **Suppress errors** check box.
9. To drop the indexes on the target table before loading the data, and recreate them when the load is complete, select the **Recreate index** check box.

   When a large amount of data is to be loaded, recreating the indexes after loading greatly improves the performance of the load because the indexes do not require maintenance during loading.

   Note: If a load requires some data to be updated rather than just inserted, do not drop the indexes because they are required to support the update operation. Removing indexes that support updates results in very slow performance. If you require specific index management, consider using an SQL JobStream node.

10. Click **OK**.

Add Control Attributes to a Table Delivery

You can add special control attributes to a fact delivery table.

The following special control attributes are available:

- **Create Date** is the date on which a new row was inserted into the target table
- **Last Update Date** is the date on which an existing row was updated

  For more information, see “Control Attributes” on page 433.
- **Record Identity** is a unique number assigned to each row that indicates that the fact delivery data is partitioned across more than one table

  For more information, see “Partition Columns Horizontally Across a Table” on page 193.

Delivering to Multiple Tables by Subscribing to Elements

For each delivery, you can choose the transformation model elements to which each delivery subscribes by subscribing to the elements to deliver.

For example, you could deliver to two tables as follows:

- Delivery1 for period_no, state_cd, product_cd, units_sold, units_forecast
- Delivery2 for period_no, state_cd, product_cd, revenue, revenue_forecast

This separates the data so that sales data and revenue data are delivered to different targets.

How you do this depends on whether the selected delivery module delivers data to a table. For information, see “Fact Delivery Modules” on page 179.

Deliver to tables

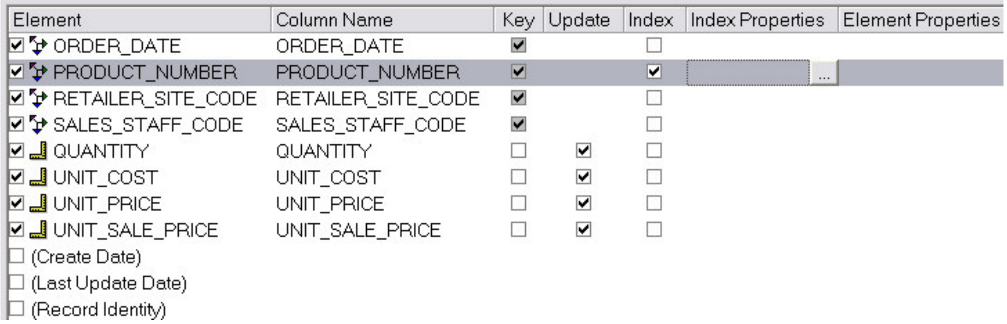
Deliver to table elements when the delivery module you selected delivers data to a table.

For information, see “Fact Delivery Modules” on page 179.

Procedure

1. Click the required fact delivery.
2. From the **Edit** menu, click **Properties** .
3. Click the **Table Properties** tab.

   To the left of each transformation model element name is a check box. By default, these check boxes are all selected so each element is delivered.



Element	Column Name	Key	Update	Index	Index Properties	Element Properties
☑ ORDER_DATE	ORDER_DATE	☑		□		
☑ PRODUCT_NUMBER	PRODUCT_NUMBER	☑		☑	...	
☑ RETAILER_SITE_CODE	RETAILER_SITE_CODE	☑		□		
☑ SALES_STAFF_CODE	SALES_STAFF_CODE	☑		□		
☑ QUANTITY	QUANTITY	□	☑	□		
☑ UNIT_COST	UNIT_COST	□	☑	□		
☑ UNIT_PRICE	UNIT_PRICE	□	☑	□		
☑ UNIT_SALE_PRICE	UNIT_SALE_PRICE	□	☑	□		
□ (Create Date)						
□ (Last Update Date)						
□ (Record Identity)						

4. Clear the check box for any element that you do not want to deliver.
5. Move the elements to the required position.

   The order in which the subscribed elements appear in this window determines their order in the target table.
6. Click **OK**.

Deliver to non-tables

Deliver to non-tables when the delivery module you selected does not deliver data to a table.

For information, see “Fact Delivery Modules” on page 179.

Procedure

1. Click the required fact delivery.
2. From the **Edit** menu, click **Properties** .
3. Click the **Element Properties** tab.

   The **Available Elements** pane shows all the elements in the transformation model. The **Subscribed Elements** pane shows the list of transformation model elements to which the delivery subscribes. This includes all the elements by default. Only values from the subscribed elements list are delivered.
4. For each element that you do not want to deliver, right-click the element in either pane, and click **Unsubscribe**.

   The element disappears from the **Subscribed Elements** pane.
5. Move the elements to the required position.

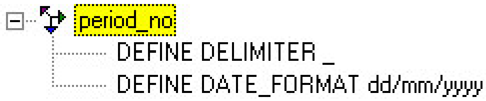
   The order in which the subscribed elements appear in this window determines their order in the target table.
6. Click **OK**.

Define the Properties for a Subscribed Element

You can, for most delivery modules, define properties at the element level and for the delivery module as a whole.

How you do this depends on whether the selected delivery module delivers data to a table.

In this example, values have been entered in the **Field delimiter** and **Date format** boxes.



```
period_no
    DEFINE DELIMITER _
    DEFINE DATE_FORMAT dd/mm/yyyy
```

Deliver to tables

Deliver to tables when the delivery module you selected delivers data to a table.

Procedure

1. Click the fact delivery for which you want to set the transformation model element properties.
2. From the **Edit** menu, click **Properties** .
3. Click the **Table Properties** tab.
4. Click the transformation model element for which you want to set properties, and then click the **Browse** button ... in the **Element Properties** column.

   The **Delivery Element Properties** dialog box appears.
5. Enter values as required.

   For information about the available properties, see Appendix A, “Fact Delivery Module Properties,” on page 413.
6. Click **OK**.

Deliver to non-tables

Deliver to non-tables when the delivery module you selected does not deliver data to a table.

Procedure

1. Click the fact delivery for which you want to set the element properties.
2. From the **Edit** menu, click **Properties** .
3. Click the **Element Properties** tab.
4. In the **Subscribed Elements** pane, right-click the transformation model element for which you want to set properties, and click **Properties**.

   The **Delivery Element Properties** dialog box appears.
5. Enter values as required.

   For information about the available properties, see Appendix A, “Fact Delivery Module Properties,” on page 413.
6. Click **OK**.

   The **Subscribed Elements** pane is updated to show that you added element information.

Change the Column Names of Delivered Data

By default, element names are used for column names when data is delivered to a table. However, you can specify different column names.

Changing a column name is only applicable to table delivery modules.

Procedure

1. Click the required fact delivery.
2. From the **Edit** menu, click **Properties** .

   The **Table Delivery Properties** window appears.
3. Click the **Table Properties** tab.
4. In the **Column Name** column, click the name to change, and then type the new name.
5. Click **OK**.

Add Missing Columns to the Target Table

When IBM Cognos Data Manager delivers data, all the columns in the source table must exist in the target table. Data Manager determines this by checking the tables during execution.

If they are different, and you are

- executing from the Designer, a message appears informing you of the problem and you can drop the tables.

  For information on checking and dropping the target table, see "Check the Target Table Prior to Executing a Build" on page 247.
- executing from the command line, no warning can be given and so the build fails.

You can specify that Data Manager automatically add any missing columns to the target table. The build will then not fail simply because the columns in the table are incompatible. When you select this option, Data Manager does not automatically check the table during execution; you must manually check the table.

Tip: To stop Data Manager automatically checking the tables during execution, from the **Tools** menu, click **Options**, and then click **Skip table compatibility checks on builds**. If you select this option and the tables are incompatible, the build fails.

Procedure

1. Click the appropriate fact build delivery.
2. From the **Edit** menu, click **Properties** .
3. Either
   - In the Table Delivery Properties window, click the **Table Properties** tab, and then click the **Automatically add columns to table** check box.
   - In the Dimension Table Properties window, click the **Table** tab, and then click the **Automatically add columns to table** check box.
4. Click **OK**.

Handle Failed Rows in a Relational Table Delivery

By default, when delivering fact data to a relational table, if IBM Cognos Data Manager encounters an error in a row of fact data, it stops the delivery at that row and reports an exception error. No further rows are delivered and no detailed error reporting occurs for a failed row.

For a specific set of errors, if a row fails, you can configure Data Manager to report the error reason, together with the failed row data, without terminating the table delivery. These errors can include

- unique constraint errors, such as duplicate primary keys
- integrity constraint errors, such as a failing foreign key
- bind errors
- non-unique key errors
- null key errors

You can select the way in which Data Manager handles rows that fail with these errors:

- **Don't write failed rows** (default)

  Data Manager does not provide detailed error reporting for failed rows, and terminates the table delivery.
- **Write failed row to file**

  Data Manager writes row errors to a simple tab delimited text file and continues with the table delivery. The file is recreated every time the build is executed.
- **Write failed row to table**

  Data Manager writes row errors to a table and continues with the table delivery. The error table can reside in any writeable Data Manager connection. Data Manager recreates this table each time the build is executed.

Note: If you are delivering fact data using a DBMS that supports array insert, such as Oracle, Data Manager is unable to report individual row errors. You can overcome this by setting the variable named DS_UDA_BULKWRITE_ROWS to 1. For information, see "DS_UDA_BULKWRITE_ROWS" on page 297.

Write failed records to a file

IBM Cognos Data Manager writes row errors to a simple tab delimited text file and continues with the table delivery. The file is recreated every time the build is executed.

Procedure

1. Click the appropriate fact build delivery.
2. From the **Edit** menu, click **Properties** .
3. Click the **Failure Handling** tab.
4. In the **Failure Handling Method** box, click **Write failed rows to file**.
5. In the **File name** box, enter the full directory path and name for error file to which you want to save failed rows.
6. If required, in the **Failure limit** box, specify the maximum number of errors to report before the table delivery should terminate.

   By default, the limit is set to zero, indicating that no failure limit is set.
7. Click **OK**.

Write failed records to a relationship table

IBM Cognos Data Manager writes row errors to a table and continues with the table delivery. The error table can reside in any writeable Data Manager connection. Data Manager recreates this table each time the build is executed.

Procedure

1. Click the appropriate fact build delivery.
2. From the **Edit** menu, click **Properties** .
3. Click the **Failure Handling** tab.
4. In the **Failure Handling Method** box, click **Write failed rows to table**.
5. In the **Connection** box, click the connection that contains the table where you want the details saved.

   Tip: To create a connection, click **New**. For more information, see “Create a Database Connection” on page 39.
6. In the **Table name** box, enter the name of the table to which you want to save failed rows.
7. If required, in the **Failure limit** box, specify the maximum number of errors to report before the table delivery should terminate.

   By default, the limit is set to zero, indicating that no failure limit is set.
8. Click **OK**.

Set Up Delivery Filters

You can restrict the fact data to be delivered using a delivery filter. A delivery filter can contain

- one or more level filters

  Level filters allow you to restrict a delivery to specific dimensions and hierarchy levels.
- a single output filter

  Output filters allow you to restrict a delivery to specific data rows.

Each delivery filter you set up within a fact build is saved as a generic filter, and you can apply it to any fact delivery within the fact build.

Create a Delivery Filter

Before you can create a level filter or an output filter, you must create a delivery filter. You can then add the level and output filters that you require to the delivery filter.

You can

- create and apply a delivery filter within a specific fact delivery
- create a delivery filter that you can then apply to any fact deliveries within the selected fact build

Procedure

1. In the **Fact Delivery** folder, click **Delivery Filters**.
2. From the **Insert** menu, click **Build**, and then click **Delivery Filter**.
3. Type a name for the delivery filter and, if you require, a business name and description.

   You can now add level and output filters to the delivery filter.

   For more information, see “Use a Level Filter to Deliver Specific Dimensions and Hierarchy Levels” on page 190 and “Use an Output Filter to Deliver Specific Data Rows” on page 192.

4. When you have finished adding level and output filters to the delivery filter, click **OK**.

   The delivery filter is added to the list of delivery filters available in the fact build. To use a delivery filter, you must apply it to one or more fact deliveries. For more information, see “Apply a Delivery Filter to a Fact Delivery” on page 193.

Create a delivery filter within a fact delivery

You can create a delivery filter within a specific fact delivery to apply the filter to the fact delivery.

Procedure

1. Click the required fact delivery.
2. From the **Edit** menu, click **Properties** .
3. Click the **Filters** tab.

   Any defined delivery filters are listed in the **Delivery filter** box.

   Tip: To edit a delivery filter, click the required filter, and then click **Edit**.
4. Click **New**.
5. Type a name for the delivery filter and, if you require, a business name and description.

   You can now add level and output filters to the delivery filter.

   For more information, see “Use a Level Filter to Deliver Specific Dimensions and Hierarchy Levels” and “Use an Output Filter to Deliver Specific Data Rows” on page 192.
6. When you have finished adding level and output filters to the delivery filter, click **OK** to close the **Delivery Filter Properties** window.
7. Click **OK** to close the **Table Delivery Properties** or the **Properties** window.

   The delivery filter is automatically applied to the fact delivery. It is also added to the list of delivery filters available in the fact build.

Use a Level Filter to Deliver Specific Dimensions and Hierarchy Levels

You can restrict a delivery to specific dimensions and hierarchy levels by using a level filter.

You can apply as many level filters to a fact delivery as you require. Create one level filter for each combination of dimension/hierarchy levels to deliver. Express these filters in the following format where element1, element2, and so on, are dimension elements, and level1, level2, and so on are hierarchy levels within the corresponding dimension.

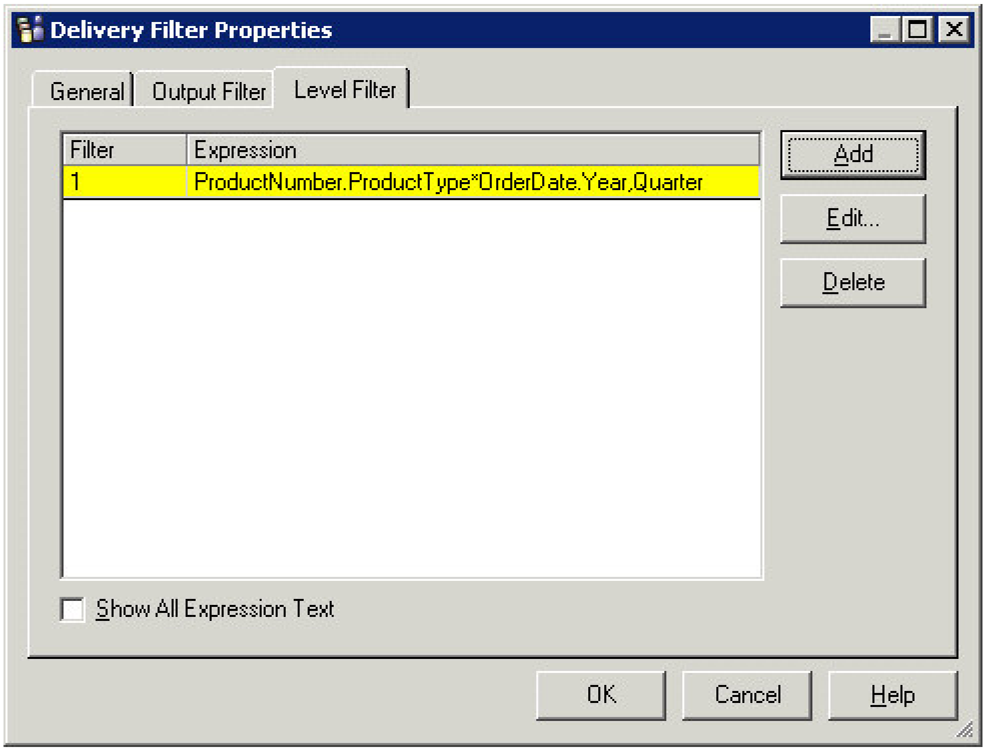
element1.level1 * element2.level2[*element3.level3 ...]

The following is the syntax for a level filter where <element> is the name of a dimension element and <level> is the name of a level from the corresponding dimension. An asterisk (*) specifies the intersection of the levels it joins. You can specify more than one level of the same hierarchy by separating the level names with commas:

<element>.<level>{,<level>}{*<element>.<level>{,<level>}}

For example the following filter expression accepts data that lies at the intersection of the ProductType level of the Product hierarchy and either the Year level or Quarter level of the Dates hierarchy.

```
ProductNumber.ProductType*OrderDate.Year,Quarter
```



Filter	Expression
1	ProductNumber.ProductType*OrderDate.Year,Quarter

□ Show All Expression Text

For example, you could partition data into Period, Quarter, and Year hierarchical levels. To do this, create three deliveries, each having a level filter, and set the filters to period_no.Period, period_no.Quarter, and period_no.Year, respectively.

Procedure

1. Open the **Delivery Filter Properties** window.
2. Click the **Level Filters** tab.

   Any defined level filters are listed.

   Tip: To edit a level filter, click the level filter, and then click **Edit**.
3. Click **Add**.

   The **Level Filter Properties** window appears.
4. Type an expression for the filter, or right-click, click **Dimension Levels**, and then click the levels to insert.
5. Click **OK**.

   The level filter is added to the **Level Filter** tab in the **Delivery Filter Properties** window.

   Tips

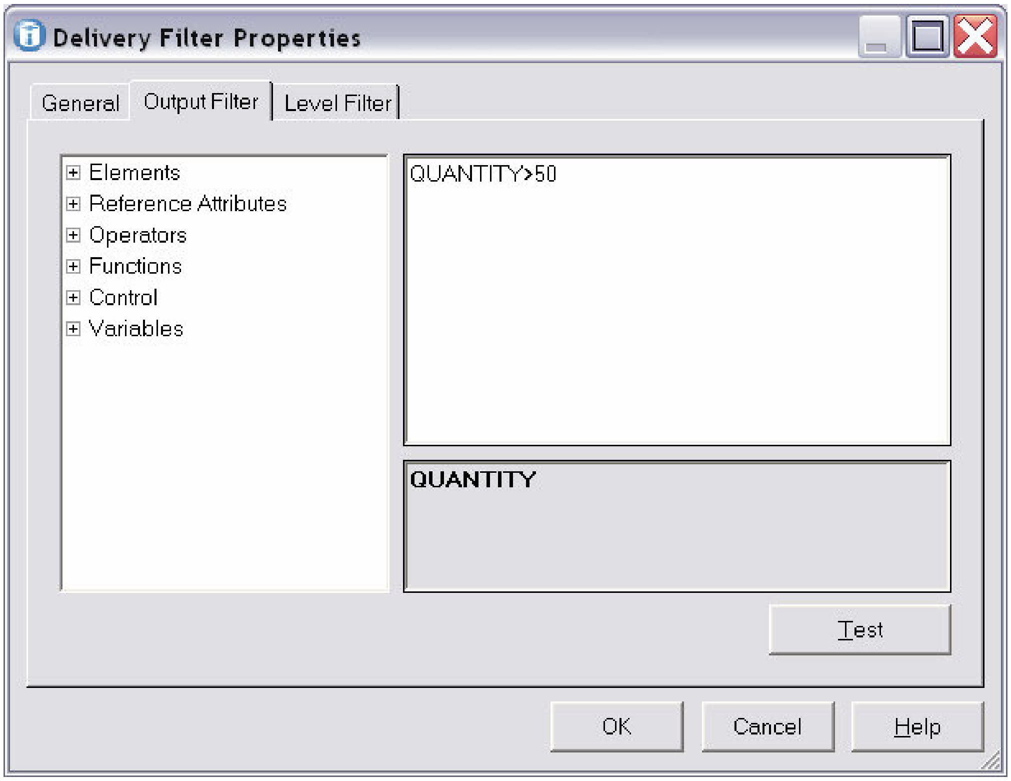
   - To show the full text of each level filter listed, click **Show All Expression Text**.
   - To delete a level filter, click the level filter, and then click **Delete**.

Use an Output Filter to Deliver Specific Data Rows

You can restrict a delivery to specific data rows using an output filter. Each delivery filter may have no more than one output filter.

An output filter is an expression that results in TRUE or FALSE when applied to each output row. The delivery only delivers those data rows for which the expression evaluates to TRUE.

Typically, you would use an output filter to vertically partition data other than by hierarchical levels. For example, you could partition data by sales. Two deliveries, one having the output filter units_sold > 0, and the other having the output filter units_sold = 0, would partition the data according to whether there were sales of that product.



QUANTITY>50

QUANTITY

Procedure

1. Open the **Delivery Filter Properties** window.
2. Click the **Output Filter** tab.
3. In the right pane, enter an expression for the filter.

   You can type the expression, and you can use the tree structure in the left pane to select the items from the following folders:
   - **Elements** to select elements defined within the transformation model
   - **Reference Attributes** to select a dimension and member combination
   - **Operators** to select operators
   - **Functions** to select predefined and user-defined functions
   - **Control** to select controls
   - **Variables** to select variables and substitution variables

   For information about operators, controls, and functions, see the IBM Cognos*Function and Scripting Reference Guide*. For information about user-defined functions, see Chapter 23, “User-Defined Functions,” on page 277. For information about variables and substitution variables, see Chapter 24, “Variables,” on page 289.

   You can select an item from a folder by dragging it to the right pane, or by double-clicking it.

4. If required, test the expression. For information, see Chapter 25, “Testing Expressions and Scripts,” on page 309.
5. Click **OK**.

Apply a Delivery Filter to a Fact Delivery

You can apply a delivery filter to any fact delivery in the fact build.

Note: Any delivery filters that you create in a specific fact delivery are automatically saved as delivery filters and you can use them in other fact deliveries within the fact build.

Procedure

1. Click the required fact delivery.
2. From the **Edit** menu, click **Properties** .
3. Click the **Filters** tab.
4. From the **Delivery filter** box, click the delivery filter that you want to apply.
5. Click **OK**.

Delete a Delivery Filter

If you delete a delivery filter, all level and output filters defined within it are also deleted.

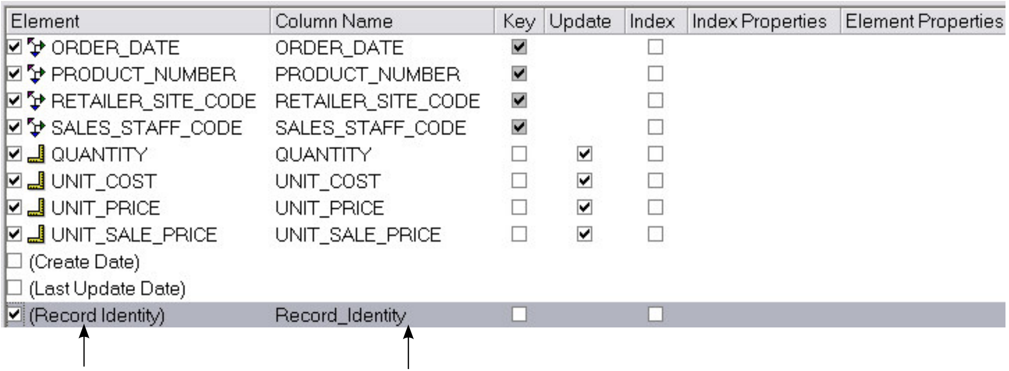
Procedure

1. In the **Fact Delivery** folder, click **Delivery Filters**.
2. From the **Edit** menu, click **Delete**.
3. In the message box, click **Yes**.

Partition Columns Horizontally Across a Table

If your fact data contains columns that are only occasionally used in queries, such as description columns, you can partition the data so that the columns are stored in a separate, but linked, table.

IBM Cognos Data Manager selects the Record Identity control attribute to indicate that the table is partitioned.



Element	Column Name	Key	Update	Index	Index Properties	Element Properties
☑ ORDER_DATE	ORDER_DATE	☑		□		
☑ PRODUCT_NUMBER	PRODUCT_NUMBER	☑		□		
☑ RETAILER_SITE_CODE	RETAILER_SITE_CODE	☑		□		
☑ SALES_STAFF_CODE	SALES_STAFF_CODE	☑		□		
☑ QUANTITY	QUANTITY	□	☑	□		
☑ UNIT_COST	UNIT_COST	□	☑	□		
☑ UNIT_PRICE	UNIT_PRICE	□	☑	□		
☑ UNIT_SALE_PRICE	UNIT_SALE_PRICE	□	☑	□		
□ (Create Date)						
□ (Last Update Date)						
☑ (Record Identity)	Record_Identity	□		□		

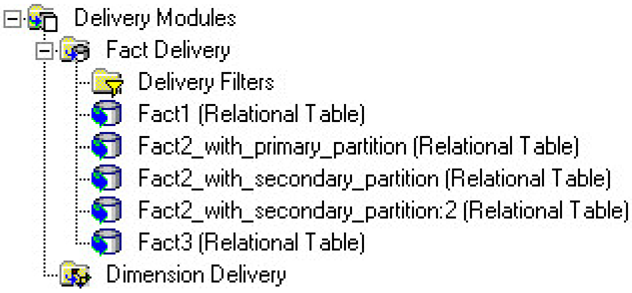
Record identity control attribute

Record identity column

Data Manager distinguishes between the primary and secondary tables using the Record_Identity column. If the Record_Identity column is not selected as the primary key, Data Manager knows that this is the primary partitioned table; if the Record_Identity column is selected as the primary key, Data Manager knows that this is a secondary partitioned table.

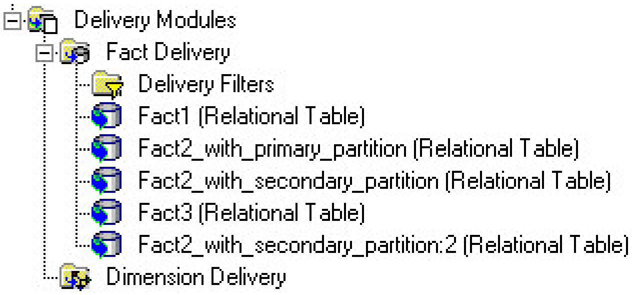
The relationship between the primary and secondary table partitions is determined by their position in the Tree pane. Data Manager assumes that tables with secondary partitioning are associated with the table directly above that has primary partitioning.

For example



- Delivery Modules
  - Fact Delivery
    - Delivery Filters
    - Fact1 (Relational Table)
    - Fact2_with_primary_partition (Relational Table)
    - Fact2_with_secondary_partition (Relational Table)
    - Fact2_with_secondary_partition:2 (Relational Table)
    - Fact3 (Relational Table)
  - Dimension Delivery

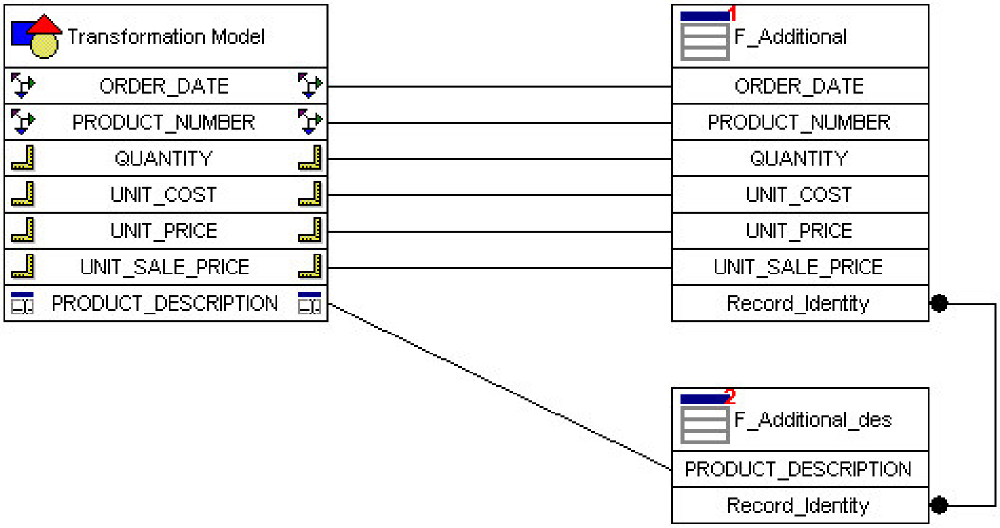
If you insert another delivery between the primary and secondary table deliveries, the table partitioning fails. For example



- Delivery Modules
  - Fact Delivery
    - Delivery Filters
    - Fact1 (Relational Table)
    - Fact2_with_primary_partition (Relational Table)
    - Fact2_with_secondary_partition (Relational Table)
    - Fact3 (Relational Table)
    - Fact2_with_secondary_partition:2 (Relational Table)
  - Dimension Delivery

You can see how the partitioning will work from the Fact Delivery tab in the Visualization pane.

The following example shows a fact build that is to deliver to two tables, the primary partitioned table named F_Additional, and the secondary partitioned table named F_Additional_des. Data Manager allows you to distinguish between the primary and secondary table using a red number one to indicate the primary table, and a red number two to indicate the secondary table or tables. The linked Record_Identity columns show the link between the two tables.



If you apply a delivery filter to a table with primary partitioning, it also applies to any secondary tables. Secondary tables can also have delivery filters applied to them, but these do not affect the primary table.

For information, see “Set Up Delivery Filters” on page 189.

If you want to export metadata for a fact delivery table that has been partitioned, the partitioning is shown in the Metadata Star Properties window. For information, see “Create a Metadata Collection” on page 271.

Partitioning data can result in improved query performance.

Create a table with primary partitioning

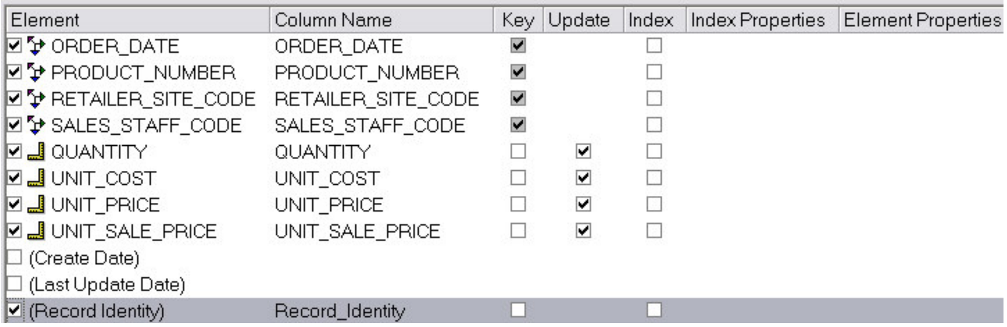
If you apply a delivery filter to a table with primary partitioning, it also applies to any secondary tables.

Secondary tables can also have delivery filters applied to them, but these do not affect the primary table. For information, see “Set Up Delivery Filters” on page 189.

Procedure

1. Click the fact delivery for which you want to partition the data.
2. From the **Edit** menu, click **Properties** .
3. Click the **Table Properties** tab.
4. From the **Partition** box, click **Primary partition**.

   The Record Identity control element is selected to indicate that this table is partitioned.



Element	Column Name	Key	Update	Index	Index Properties	Element Properties
☑ ORDER_DATE	ORDER_DATE	☑		□		
☑ PRODUCT_NUMBER	PRODUCT_NUMBER	☑		□		
☑ RETAILER_SITE_CODE	RETAILER_SITE_CODE	☑		□		
☑ SALES_STAFF_CODE	SALES_STAFF_CODE	☑		□		
☑ QUANTITY	QUANTITY	□	☑	□		
☑ UNIT_COST	UNIT_COST	□	☑	□		
☑ UNIT_PRICE	UNIT_PRICE	□	☑	□		
☑ UNIT_SALE_PRICE	UNIT_SALE_PRICE	□	☑	□		
□ (Create Date)						
□ (Last Update Date)						
☑ (Record Identity)	Record_Identity	□		□		

Note: In a primary table, you cannot select the Record_Identity column as the primary key.

To the left of each transformation model element name is a check box. By default, these check boxes are all selected, therefore each element is included in the table.

5. Clear the check box for any element to exclude from the primary partitioned table.
6. If required, move the selected elements to the required position.

   The order in which the selected elements are shown in the target table is determined by their order in this window.
7. Click **OK**.

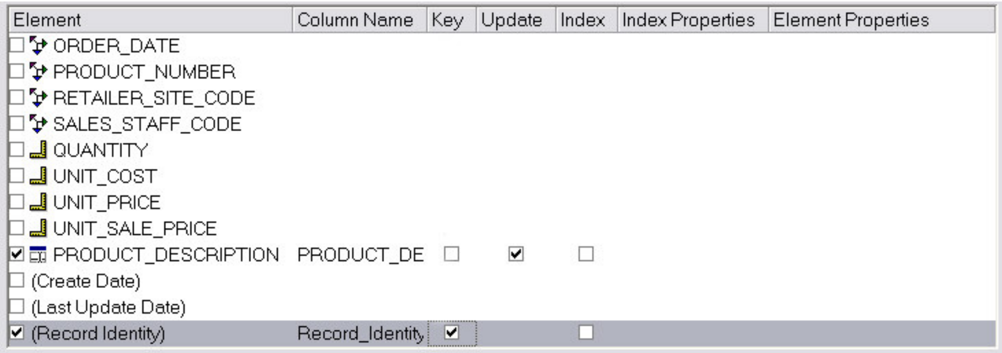
Create a table with secondary partitioning

The relationship between the primary and secondary table partitions is determined by their position in the Tree pane. IBM Cognos Data Manager assumes that tables with secondary partitioning are associated with the table directly above that has primary partitioning.

Procedure

1. Click the fact delivery for which you want to partition the data.
2. From the **Edit** menu, click **Properties** .
3. Click the **Table Properties** tab.
4. From the **Partition** box, click **Secondary partition**.

   The Record Identity control element indicates that this table is partitioned. Data Manager automatically selects the Record_Identity column as the primary key, and any columns that were previously selected as primary keys are cleared. Selecting the Record_Identity column as the primary key is mandatory in a secondary table.



Element	Column Name	Key	Update	Index	Index Properties	Element Properties
☐ ORDER_DATE						
☐ PRODUCT_NUMBER						
☐ RETAILER_SITE_CODE						
☐ SALES_STAFF_CODE						
☐ QUANTITY						
☐ UNIT_COST						
☐ UNIT_PRICE						
☐ UNIT_SALE_PRICE						
☑ PRODUCT_DESCRIPTION	PRODUCT_DE	☐	☑	☐		
☐ (Create Date)						
☐ (Last Update Date)						
☑ (Record Identity)	Record_Identity	☑		☐		

To the left of each transformation model element name is a check box. By default, these check boxes are all selected; each element is therefore included in the table.

5. Clear the check box of any element that you do not want to include in the delivery.

   Note: You can include a column in more than one table.
6. If required, move the selected elements to the required position.

   The order in which the selected elements are shown in the target table is determined by their order in this window.
7. Click **OK**.

Disable and Enable a Fact Delivery

At times, you may want to temporarily disable a delivery. Although you can delete the delivery, and later recreate it, a better option is to disable the delivery. You can then enable it when it is required.

For more information on deleting a fact delivery, see “Delete a Fact Delivery” on page 198.

Note: If you disable a table that has a primary partition, its secondary tables are automatically disabled.

Procedure

1. Click the required delivery.
2. From the **Edit** menu, click **Properties** .
3. Click the **General** tab.
4. Either clear or select the **Enabled** check box.
5. Click **OK**.

   Tip: You can also disable a delivery by clicking the delivery and then, from the **Edit** menu, clicking **Enabled**.

Change the Delivery Module for a Fact Delivery

If required, you can change the delivery module that delivers the fact data.

Each delivery module has a different set of properties. After changing the delivery module type, you must specify the module properties and element properties for the new delivery type. For information, see Appendix A, “Fact Delivery Module Properties,” on page 413.

Procedure

1. Click the required fact delivery.
2. From the **Actions** menu, click **Change Delivery Module**.
3. From the **Change to** list, click the new delivery module.
4. Click **OK**.

Delete a Fact Delivery

You can delete a fact delivery that is no longer required.

Procedure

1. Click the required delivery.
2. From the **Edit** menu, click **Delete**.
3. In the message box, click **Yes**.

   Tip: You may want to temporarily disable a delivery rather than delete it, and then perhaps later have to create it. For information, see “Disable and Enable a Dimension Delivery” on page 201.

Chapter 17. Delivering Dimension Data Using a Fact Build

In IBM Cognos Data Manager, you use delivery modules to deliver transformed data to the specified data repositories.

Data Manager organizes the delivery modules into these groups:

- fact delivery modules that deliver data produced by a fact build
- dimension delivery modules that deliver data describing a single dimension

Each fact build can have as many deliveries as you require.

You can manually add a delivery module to a fact build, or you can use the Fact Build wizard. If you want to deliver partitioned data to different tables within different databases, you must do it manually.

Dimension Delivery Modules

Dimension deliveries deliver data that describes a single dimension. You can deliver the data to a single table (a star schema) or multiple tables (a snowflake schema). You can also deliver more than one dimension delivery per dimension (for example, two star schemas).

Although fact builds can deliver dimension data, they are not as flexible at managing dimensions as dimension builds. For information about when to use a dimension delivery in a fact build, and when to use a dimension build, see “Separate Reference Data and Fact Data” on page 88.

Note: If you want to add a dimension delivery to a fact build, the fact build must have reference dimensions defined for it.

Add a Dimension Delivery

You can add dimension deliveries to a fact build manually or while creating a fact build using the Fact Build wizard.

For information about adding a dimension delivery using the Fact Build wizard, see “The Fact Build Wizard” on page 107.

Procedure

1. In the appropriate fact build, expand **Delivery Modules** and click **Dimension Delivery**.
2. Click the dimension for which you want to add a delivery.
3. From the **Insert** menu, click **Build**, and then click **Dimension Delivery**.
4. From the **Dimension Delivery** menu, click **Relational Table**.

   The **Dimension Delivery Properties** window for the selected dimension appears.
5. Click the **General** tab.
6. In the **Name** box, type a name for the delivery. If required, type a business name and description.

7. If you want to exclude the delivery when the build is executed, clear the **Enabled** check box.

   You can now temporarily disable the delivery without having to delete it and later recreate it.
8. Click the **Output** tab.
9. In the **Deliver into database** box, click the target database.
10. In the **ID of top parent** box, type the value of the parent attribute of the top-level members of the hierarchy.
11. If you want IBM Cognos Data Manager to truncate the table before delivery, select the **Full refresh** check box.
12. If the reference member has more than one parent and you want to deliver only the first parent, select the **Ignore multiple parents** check box.
13. Click **OK**.

    A dimension delivery is added to the selected dimension. You can now add tables to the dimension delivery.

Add a Table to a Dimension Delivery

Each dimension delivery delivers dimension data to one or more tables.

There is no limit to the number of tables that you can add to a dimension delivery.

Procedure

1. Click the dimension delivery to which you want to add a table.
2. From the **Insert** menu, click **Build**, and then click **Table**.
3. Click the **Table** tab.
4. Enter the name of the dimension table:
   - To deliver to a new table, in the **Table name** box, type a name for the table.
   - To deliver to an existing table, click the **Browse for table button** . The **Select Table** dialog box appears. Click the required table, and click **OK**.
5. In the **Commit interval** box, type (in terms of the number of rows updated/inserted) the interval between commits of data in the target table.

   By default, IBM Cognos Data Manager does not issue a COMMIT instruction until it has delivered all the data rows. Data Manager executes the entire delivery as a single transaction. You must decide whether this represents an unacceptably large or long transaction.
6. If you do not want rows for higher level members with no children to be delivered, select the **Exclude Partially Populated Rows** check box.

   If this check box is clear, rows are delivered for higher level members with no children, and the lower level columns are set to null.

   Note: You must clear this check box if you want to deliver an unbalanced hierarchy. For more information on unbalanced hierarchies, see “Hierarchy Types” on page 95.
7. If you want to exclude the table when the build is executed, clear the **Enabled** check box.
8. If you want Data Manager to automatically add any missing columns to the target table, select the **Automatically add columns to table** check box.

   For more information, see “Add Missing Columns to the Target Table” on page 187.

9. Click the **Columns** tab.
10. In the **Use template** box, click the template that defines the behavior of the dimension table.

    Tip: Alternatively, create a template by clicking **New**, or edit a template by clicking **Edit**. For more information about templates, see Chapter 9, “Templates,” on page 51.

    The attributes of the template are shown.

    You can now specify the source of the dimension table columns from the column tokens and reference structure level attributes.
11. In the **Available Sources** pane, click the column token or level attribute and then in the left pane, click the column.

    For information about column tokens, see “Column Tokens” on page 209.
12. Click **Map**.

    Tip: You can also drag the column token or level attribute to the relevant position in the **Sourced From** column.
13. To create an index for a column, select the relevant check box in the **Index** column. You can define the properties for the index by clicking the **Browse** button ... in the **Index Properties** column.

    For more information, see “Define the Index Properties” on page 221.
14. Create any new columns that you require.

    New columns are added to the bottom of the list. You can reposition them as required.
15. Click **OK**.

Disable and Enable a Dimension Delivery

At times, you may want to temporarily disable a delivery.

Although you can delete the delivery, and later recreate it, a better option is to disable the delivery. You can then enable it when it is required.

For more information, see “Delete a Dimension Delivery.”

Procedure

1. Click the required delivery.
2. From the **Edit** menu, click **Properties** .
3. Click the **General** tab.
4. Either clear or select the **Enabled** check box.
5. Click **OK**.

   Tip: You can also disable a delivery by clicking the delivery and then, from the **Edit** menu, clicking **Enabled**.

Delete a Dimension Delivery

You can delete a dimension delivery that is no longer required.

Procedure

1. Click the required delivery.
2. From the **Edit** menu, click **Delete**.

3. In the message box, click **Yes**.

   Tip: You may want to temporarily disable a delivery rather than delete it, and then perhaps later have to create it. For more information, see “Disable and Enable a Dimension Delivery” on page 201.

Chapter 18. Dimension Builds

A dimension build delivers data that describes a single business dimension, such as Product or Customer.

Each dimension build
- acquires dimension data from the reference structure you specify
- delivers the dimension data to the target data mart, using a template to define the behavior of each column

You can choose to deliver the dimension data to a single table (star schema), to one table for each hierarchy level (snowflake schema), or to a table partitioned by custom criteria.

A dimension build can deliver dimension data for many fact builds. Fact builds can also be used to deliver dimension data, but this may mean several fact builds delivering the same data, resulting in a duplication of effort.

Use a dimension build when you
- have a number of target fact tables that share the same dimension tables
- want to deliver all the dimension data, including that for which there is no related fact data
- want to prepare the dimension tables prior to delivering the fact data (for example, where the fact data is temporarily unavailable)
- want to deliver dimension data with attribute changes tracked
- want to build a conformed data mart

For information about tracking attribute changes, see “Track Attribute Changes” on page 210.

You can also use more advanced dimension build techniques to manage data containing changes to dimensional structures, and attributes containing historical changes. These advanced techniques could be used in the following situations:
- Your source dimension data contains specific date columns and you want to use these dates when setting the effective start and end dates for sets of dimension members that share a common business key.

  For more information on using specific date columns to set effective dates, see “Dimensional History” on page 213.

  An example of this would be an Employee dimension that must use employment commencement dates from the source data to control dimensional processing.
- Your source dimension data contains more than one update for a dimension member. Each dimension update must be processed in timestamp order so that when the data is delivered to the dimension table, it accurately reflects all the changes that have occurred to a dimension member.

  For more information about processing dimension data in timestamp order, see “Dimensional History” on page 213.
- Your source fact data contains late arriving fact records and you want to ensure that each record can always be associated with the correct dimension record.

For more information, see “Process Late Arriving Facts” on page 170.

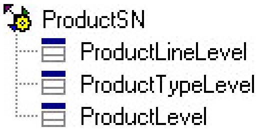
View Dimension Builds

You can view dimension builds in the Tree pane or the Visualization pane.

Viewing Dimension Builds in the Tree Pane

Each dimension build in the Builds and JobStreams folder is represented in the Tree pane

The following is a view of the Tree pane.

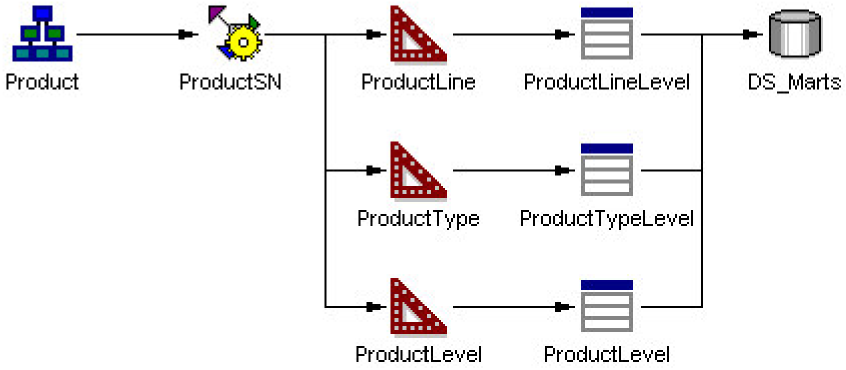


```
ProductSN
├ ProductLineLevel
├ ProductTypeLevel
└ ProductLevel
```

Tip: You can view just the dimension builds contained in a catalog by clicking the Dimension Builds tab at the bottom of the Tree pane.

Viewing Dimension Builds in the Visualization Pane

When you click a dimension build in the Tree pane, a visual representation of the whole dimension build appears in the Visualization pane.



For a description of each of the symbols used in the Visualization pane, see Appendix F, “Buttons and Icons,” on page 497.

Use the Dimension Build Wizard

You can use the Dimension Build wizard to help you to create a dimension build that delivers the schema that you specify.

For information, see “Schema Types” on page 205.

Note: You can also create custom schema types by changing a dimension build to suit your requirements.

The Dimension Build wizard automatically creates the necessary templates that define the behavior of each column of the delivered tables.

When you use the Dimension Build wizard you define the

- schema details
- schema naming conventions
- features of the dimension build
- properties of the dimension build
- attributes for the dimension tables
- change tracking behavior

When you create a dimension build using the Dimension Build wizard, you can amend it, and add to it, as you require.

Procedure

1. From the **Tools** menu, click **Dimension Build Wizard** .
2. Follow the instructions.

Schema Types

When you use the Dimension Build wizard, you must specify the schema to use to deliver dimension data.

You can choose from these schema types: star, optimal star, snowflake, optimal snowflake, and parent-child.

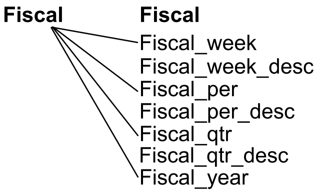
Star Schema

A star schema represents the dimension in a single table.

Each level of the delivered hierarchy is represented by one or more columns of this table. Therefore, this schema is a fully denormalized representation of the dimension.

If the table includes a surrogate key, this is assumed to be the primary key. Otherwise the ID of the lowest level is assumed to be the primary key.

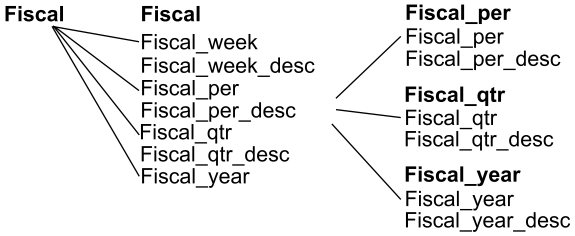
If you want to use the Dimension Build wizard to create a dimension build that supports tracking of attribute changes, you must use this schema type.



Optimal Star Schema

An optimal star schema is a variant of the star schema.

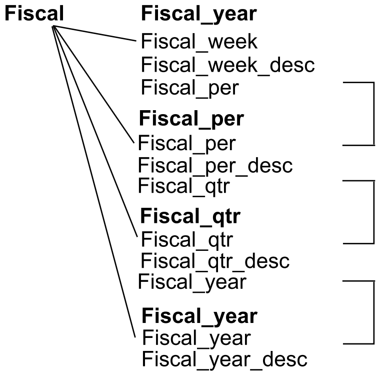
It has a main table that contains the keys of all the levels of the delivered hierarchy. However, it contains the description of only the lowest level. To save storage space, the descriptions of the higher levels reside in supplementary tables.



Snowflake Schema

A snowflake schema is more complex than a star schema.

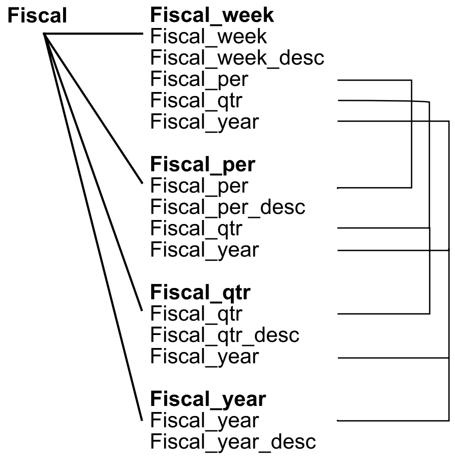
It represents the delivered hierarchy with one table for each hierarchy level. Each table has a foreign key to the level above. The primary key of each table is the member ID of the corresponding hierarchy level. This schema is a normalized representation of the dimension.



Optimal Snowflake Schema

An optimal snowflake schema is similar to a snowflake schema.

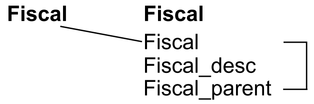
However, the table that corresponds to the lowest level contains foreign keys to all of the levels above. This can improve performance because the analysis software does not need to follow the chain of foreign keys to identify a higher-level member.



Parent-Child Schema

A parent-child schema represents the delivered dimension in a single table.

Each row of this table contains the ID of the member, together with the ID of its parent (where a parent exists). Although not required, a parent-child schema may identify the hierarchy level of each member. This schema is a recursive representation of the dimension.



A variant of this schema uses one table for all dimensions and has an additional column to identify the dimension to which each row relates.

We do not recommend the use of parent-child schemas with IBM Cognos software.

Create a Dimension Build Manually

To create a dimension build manually, you must add a dimension build, and then add the required dimension tables.

Add a Dimension Build

This section describes how to add a dimension build.

Procedure

1. Click the **Builds and JobStreams** folder.
2. From the **Insert** menu, click **Build**, and then click **Dimension Build**.
3. Click the **General** tab.
4. In the **Name** box, type a unique name for the new dimension build. If you require, type a business name and description.
5. Click the **Dimension** tab.
6. In the **Dimension to be delivered** box, click the reference dimension that contains the reference structure to deliver.

7. In the **Hierarchy/Lookup to be delivered** box, click the reference structure that you want the dimension build to deliver.
8. In the **Deliver into database** box, click the target data mart for the dimension build.

   Tip: Click **New** to create a connection. For information, see Create a Database Connection.
9. If you want to deliver the reference structure as a parent-child schema, in the **ID of top parent** box, type the ID of the parent of the top member of the hierarchy.

   Note: This value is normally null because the top member has no parent.
10. To prevent delivery of unused, automatically generated, foster parents, select the **Remove Unused Foster Parents** check box.

    For information, see “Foster Parents” on page 78.
11. If the dimension build should truncate the dimension tables each time it is executed, select the **Full Refresh** check box.
12. If the dimension build should only deliver the first-encountered parent of each member, select the **Ignore Multiple Parents** check box.
13. Click **OK**.

    You have now entered the minimum information necessary to create a dimension build, and are ready to add a table to the dimension build.

Add a Table to a Dimension Build

Each dimension build delivers dimension data to one or more tables.

There is no limit to the number of dimension tables that a dimension build can deliver. However, they must all reside in the same database.

Procedure

1. Click the dimension build to which you want to add a table.
2. From the **Insert** menu, click **Build**, and then click **Table**.

   The **Dimension Table Properties** window appears.
3. Click the **Table** tab.
4. Enter the name of the dimension table:
   - To create a dimension table, in the **Table name** box, type a name for the table.
   - To deliver the dimension data to an existing table, click the **Browse for table button** ▤. The **Select Table** dialog box appears. Click the required table, and click **OK**.
5. In the **Commit interval** box, type (in terms of the number of rows updated/inserted) the interval between commits of data in the target table.

   By default, a COMMIT instruction is not issued until all the data rows have been delivered. The entire delivery is executed as a single transaction. You must decide whether this represents an unacceptably large or long transaction.
6. If you do not want rows for higher level members with no children to be delivered, select the **Exclude Partially Populated Rows** check box.

   If this check box is cleared, then rows are delivered for higher level members with no children, and the lower level columns are set to null.

   Note: You must ensure this check box is cleared if you want to deliver an unbalanced hierarchy. For more information on unbalanced hierarchies, see “Hierarchy Types” on page 95.

7. If you want to exclude the table when the build is executed, clear the **Enabled** check box.
8. If you want IBM Cognos Data Manager to automatically add any missing columns to the target table, select the **Automatically add columns to table** check box.

   For more information, see “Add Missing Columns to the Target Table” on page 187.
9. Click the **Columns** tab.
10. In the **Use template** box, click the template that defines the behavior of the dimension table.

    You can create a template, or use any suitable existing template in the selected reference dimension.

    Tip: To create a template, click **New**, or to edit a template, click **Edit**.

    For more information, see “Create a Template” on page 53.

    You are now ready to select the source of the dimension table columns from the column tokens and reference structure level attributes.
11. In the **Available Sources** pane, click the column token or level attribute and then, in the left pane, click the column.

    For more information, see “Column Tokens.”
12. Click **Map**.

    Tip: You can also drag the level attribute or column token to the relevant position in the **Sourced From** column.
13. If you want to deliver data with the attribute changes tracked, select the **Track changes (Slowly Changing Dimension)** check box. In the **Track** column that appears, specify the columns to track and the tracking method to use, by clicking the appropriate tracking method (either 1, 2, or 0) from the relevant drop down list.

    Note: Only certain columns can be tracked. If a column cannot be tracked, the track value for the column is blank.

    For more information, see “Track Attribute Changes” on page 210.
14. If you want to create an index for a column, select the relevant check box in the **Index** column.

    You can then define the properties for the index by clicking the **Browse** button ... in the **Index Properties** column.

    For more information, see “Define the Index Properties” on page 221.
15. Click **OK**.

Column Tokens

When you add a table to a dimension build or a dimension delivery, you can select the source of the columns in the table from column tokens.

Column tokens represent level attributes in a reference structure that have certain properties, irrespective of the level. For example, the level attribute that provides the ID for each row of data is represented by the $ID column token.

Note: You can specify a column token as the source of more than one dimension table column. This allows you to create as many custom dimension table deliveries as you require.

The following table describes each column token.

Token	Description
$ID	The ID of the member.
$CAPTION	The business name of the member.
$PARENT	The ID of the parent of the member.
<level>.$SURROGATE	The surrogate ID of the ancestor at the specified level. For more information, see Chapter 10, "Surrogate Keys," on page 59.
$SURROGATE	The surrogate ID of the member.
$PARENT_SURROGATE	The surrogate ID of the parent of the member.
$LEVEL_NAME	The name of the level in which the member resides.
$LEVEL_CAPTION	The business name of the level in which the member resides.
$LEVEL_NO	The ordinal number of the level in which the member resides. The topmost level has the ordinal number 1.
$SEQ_IN_LEVEL	The ordinal number of the member in a flat ordering sequence. The sequence numbers each level from left to right with the first number of each level continuing from the level above. For example 1 2 3 4 5 6 7 8 9
$SEQ_IN_ANCESTOR	The ordinal number of the member with respect to its ultimate ancestor. Each member of the top level has the number 1. Under each top-level member, the numbering restarts at 1. For example 1 1 1 2 3 1 2 3 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9

Track Attribute Changes

Tracking attribute changes is a technique for managing historical data.

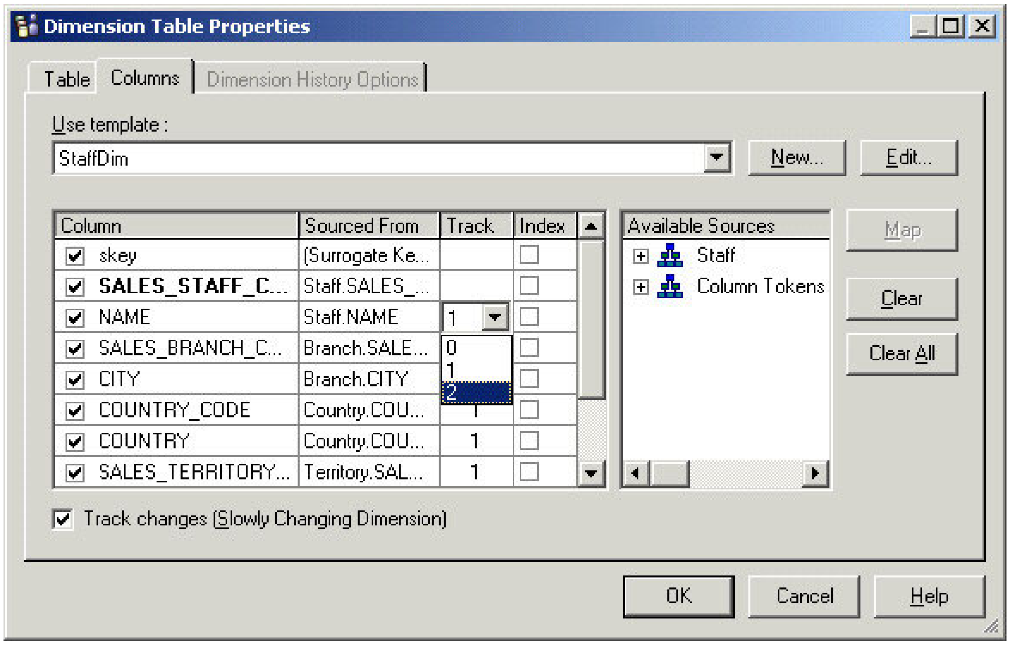
It allows you to maintain dimensions for which non-key attributes can change over time without corresponding changes in the business key. For example, employees may change their department without changing their employee number, or the specification for a product may change without changing the product code.

Each column that you include in a dimension table corresponds to a dimension attribute. These columns can either overwrite attribute changes or track attribute changes. When you add a column it is automatically set to overwrite attribute changes (type 1).

Tracking cannot be performed on the

- primary business key
- maintained surrogate key
- effective start or end dates
- create last update date
- current indicator

You set up attribute change tracking from the Columns tab in the Dimension Table Properties window.



Dimension Table Properties

Use template : StaffDim

Column	Sourced From	Track	Index
☑ skey	(Surrogate Ke...		□
☑ SALES_STAFF_C...	Staff.SALES_...		□
☑ NAME	Staff.NAME	1	□
☑ SALES_BRANCH_C...	Branch.SALE...		□
☑ CITY	Branch.CITY		□
☑ COUNTRY_CODE	Country.COU...	1	□
☑ COUNTRY	Country.COU...	1	□
☑ SALES_TERRITORY...	Territory.SAL...	1	□

☑ Track changes (Slowly Changing Dimension)

Overwrite Attribute Changes (Type 1 Changes)

You overwrite attribute changes when historical values are not required. This is known as a type 1 change and is the default.

For example, a customer's address may change but there is no business requirement to track previous addresses. All customer records containing the address are updated with the new address.

If IBM Cognos Data Manager encounters an attribute that requires a type 1 change, it updates the existing value with the new value, and sets the update date in all applicable records.

For information about how Data Manager sets the update date, see “Set the Date Source for Timestamping Data” on page 111.

Track Attribute Changes (Type 2 Changes)

You track attribute changes when you want to keep a record of historical values. This is known as a type 2 change.

For example, when an employee is moved from branch to branch, all old transactions should remain in the old branch, and only new transactions should relate to the new branch.

If IBM Cognos Data Manager encounters an attribute that requires a type 2 change, it

- creates a new dimension data record with
  - the new attribute values
  - a new surrogate key
  - the effective start date and current indicator
  - current values of unchanged attributes
- updates the previous record by setting the effective end date and current indicator with the appropriate previous record behavior

For information about setting effective dates, see "Specify Effective Date Options" on page 220.

Note: Where type 1 and type 2 changes have occurred for the same record, the type 1 updates that are not marked as type 2 are applied to all records for the business key at the level for which the type 1 attribute is declared, regardless of whether the record is current or not. For an example of this scenario, see "Example 2: Dimension Table With Attribute Change Tracking Behavior" on page 224.

Corrective Changes (Type 0 Changes)

Type 0 changes can only occur where type 2 changes are set to occur and the effective dates are timestamped according to the data.

If there are multiple changes to an attribute on the same date, rather than maintaining a history of each change, IBM Cognos Data Manager overwrites the current record for each change.

For example, suppose a dimension table contains the following Product reference data, where the Prod_Desc attribute has been marked for a type 0 update, and the Prod_Type attribute marked as a type 2 update.

Surr_key	**Prod_Code**	**Prod_Type**	**Prod_Desc**	**Eff_Date**	**End_Date**	**Curr Ind**
1	P1	002	Blue pen	2005/02/10	2006/12/15	N
2	P2	001	Green pen	2005/03/15		Y
3	P1	001	Blue pen	2006/12/15		Y

A single dimension build is then used to load the following source data, to the dimension table shown above.

Prod_Code	Prod_Type	Prod_Desc	Date
P1	001	Orange pen	2006/12/22

A type 0 change is applied to the Prod_Desc attribute for record 3, by overwriting the previous value with the value Orange pen.

Surr_key	Prod_Code	Prod_Type	Prod_Desc	Eff_Date	End_Date	Curr Ind
1	P1	002	Blue pen	2005/02/10	2006/12/15	N
2	P2	001	Green pen	2005/03/15		Y
3	P1	001	Orange pen	2006/12/15		Y

Dimensional History

By default, when processing dimensional dates, IBM Cognos Data Manager uses the system date of the computer on which builds are executed to timestamp data.

However, the source data may already contain dates that should be used instead of the system date. Additionally, the source data may contain more than one occurrence of a business key, and a date column that describes a series of dimensional updates for a common business key. These characteristics are common in employee dimensions or where more than one update occurs between updates being processed. This is known as dimensional history.

When delivering dimension data, Data Manager normally delivers only the most current record for a specific dimension member, ignoring any historical records. If your dimension table includes an attribute for which you are tracking changes, and there has been more than one change to a dimension member since you previously delivered dimension data, the result may be that there are gaps in the dimensional history.

For example, suppose the following Product reference data has already been delivered to a dimension table.

Product Code	Surrogate Key	Specification	Effective Date	End Date
P1	1	Blue pen	2000/11/01	2003/12/22
P1	3	Black pen	2003/12/23	2004/02/09
P1	4	Red pen	2004/02/10	

You can see that changes to the Specification attribute are being tracked. The current dimension record for product P1 is record 4.

If you load the following dimension data, using a single dimension build, to the dimension table shown above, only the most current record (dated 2004/10/10) is delivered.

Product Code	Specification	Date
P1	Orange pen	2004/02/14
P1	Mauve pen	2004/04/27
P1	Green pen	2004/10/10

The records dated 2004/02/14 and 2004/04/27 are ignored. Since the dimension data includes three changes to the specification of product P1, and only one is delivered, the dimension table now contains an incomplete history of changes to product P1.

By delivering the dimensional history, you can include a full and accurate history of changes to a dimension member since the previous delivery.

Notes

You cannot deliver dimensional history that occurs prior to the effective date of the current dimension record in the dimension table. For more information, see “Reject Late Arriving Dimension Details” on page 216.

Delivering dimensional history is necessary if you want to process source data that contains late arriving facts. For more information, see “Process Late Arriving Facts” on page 170.

Before you can deliver dimensional data, you must set up the dimensional history. For information, see “Set Up Dimensional History”

Set Up Dimensional History

To enable dimensional history in IBM Cognos Data Manager and use a date column from the data source as the source for dimensional processing, you must check the column behavior in the source and the delivery templates.

You must ensure that

- the template that is used to define the attributes being read from the data source includes a column with effective start date behavior

  For more information, see “Set the column behavior in the source template” on page 215.
- the template that is used to deliver the dimension build includes the same column, again with effective start date behavior

  For more information, see “Set the column behavior in the delivery template” on page 215.
- the dimension build includes a dimension table column with effective start date behavior that is sourced from a reference structure attribute containing date values

  For more information, see “Add a Table to a Dimension Build” on page 208.
- the dimension table must be set to track changes

  For more information, see “Track Attribute Changes” on page 210.

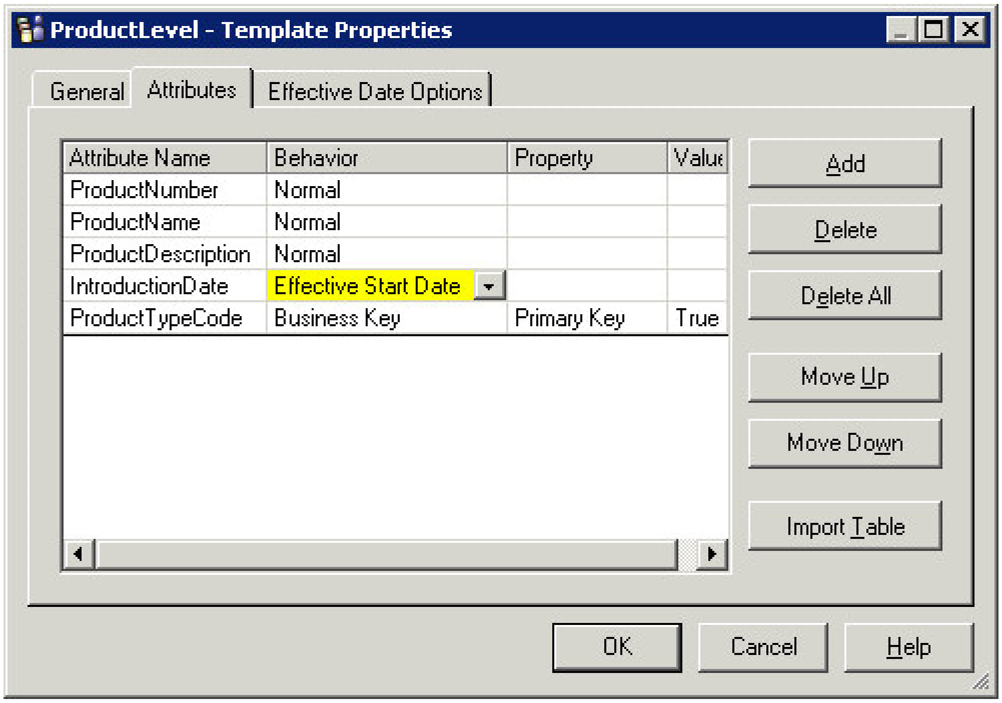
Set the column behavior in the source template

You must ensure that the source template includes a column with effective start date behavior.

Procedure

1. In the **Templates** folder for the appropriate dimension, click the template that is used to define the attributes being read from the data source.
2. From the **Edit** menu, click **Properties** .

   The **Template Properties** window appears.
3. Click the **Attributes** tab.
4. In the **Behavior** column, select **Effective Start Date** for the column that is used as the date attribute to define dimensional history.



ProductLevel - Template Properties

Attribute Name	Behavior	Property	Valu
ProductNumber	Normal		
ProductName	Normal		
ProductDescription	Normal		
IntroductionDate	Effective Start Date		
ProductTypeCode	Business Key	Primary Key	True

5. Click **OK**.

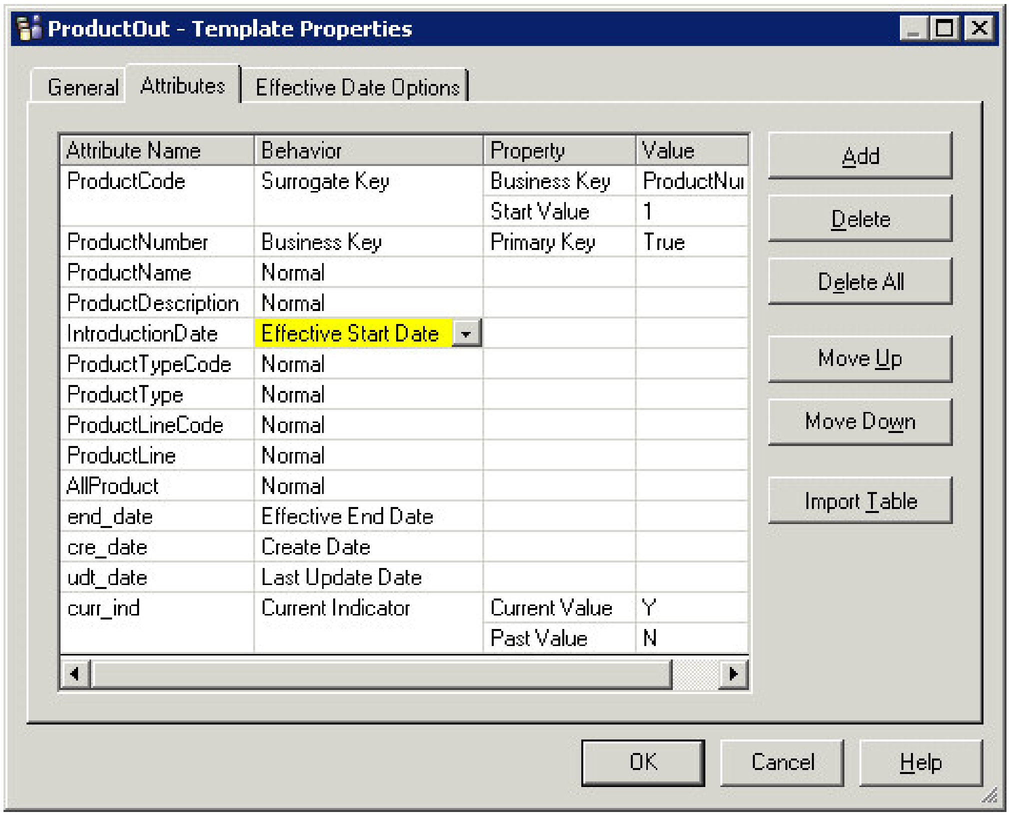
Set the column behavior in the delivery template

You must ensure that the delivery template includes the same column as the source template, with effective start date behavior.

Procedure

1. Click the delivery template that is used to deliver the dimension table in the dimension build.
2. From the **Edit** menu, click **Properties** .

   The **Template Properties** window appears.
3. Click the **Attributes** tab.
4. In the **Behavior** column, select **Effective Start Date** for the column that was also marked as an effective start date in the source.



Attribute Name	Behavior	Property	Value
ProductCode	Surrogate Key	Business Key	ProductNu
		Start Value	1
ProductNumber	Business Key	Primary Key	True
ProductName	Normal		
ProductDescription	Normal		
IntroductionDate	Effective Start Date		
ProductTypeCode	Normal		
ProductType	Normal		
ProductLineCode	Normal		
ProductLine	Normal		
AllProduct	Normal		
end_date	Effective End Date		
cre_date	Create Date		
udt_date	Last Update Date		
curr_ind	Current Indicator	Current Value	Y
		Past Value	N

When the attribute is defined as the effective start date, all dimension date processing for effective start and end dates uses these dates.

Note: It is important to ensure that only one attribute is marked as Effective Start Date.

5. Click **OK**.

Reject Late Arriving Dimension Details

If a dimension table includes an attribute for which you are tracking changes, when IBM Cognos Data Manager encounters a change in that attribute for a specific dimension member, it normally only checks the current dimension record for that member when updating the dimension table.

If the dimension data for a dimension member includes additional historical attribute changes which occurred prior to the effective start date for the current dimension record, they cannot be included in the dimension table and are rejected.

If the dimension data for a dimension member includes updated attributes and the effective start dates in the source data and current record are the same, the attributes are overwritten. This happens even if history is being tracked for one or more attributes. This occurs because the effective start dates are the same, and the record is considered to be a correction.

For example, suppose the following Product reference data has already been delivered to a dimension table.

Product Code	Surrogate Key	Specification	Effective Date	End Date
P1	1	Blue pen	2000/11/01	2003/12/22
P1	3	Black pen	2003/12/23	2004/02/09
P1	4	Red pen	2004/02/10	

You can see that changes to the Specification attribute are being tracked. The current dimension record for product P1 is record 4.

If the specification of product P1 changes on 2004/06/07, record 4 is checked, and the dimension table is updated as required.

Suppose the following dimension data is delivered, using a single dimension build, to the dimension table shown above.

Product Code	Specification	Date
P1	Orange pen	2001/05/14
P1	Mauve pen	2003/04/27
P1	Green pen	2004/10/10

The dimension data includes three changes to the specification of product P1, two of those changes occurring prior to the effective date of the current dimension record in the dimension table (record 4) for product P1. In this case, the dimension changes dated 2001/05/14 and 2003/04/27 are rejected, and only the dimension table is updated by using changes for 2004/10/10.

Save Rejected Late Arriving Dimension Details

When IBM Cognos Data Manager rejects records during build execution, the rejected data can be saved to a file or a table.

If you do not want to save rejected records, click **Don't write reject records**.

Write details to a file:

Data Manager creates a simple tab delimited text file. The first time a build that creates reject records is executed, Data Manager creates a text file. If the build has been executed previously, the existing data in the text file is updated.

By default, the text file is named <build_name>.rej and is stored in the Data Manager Data directory.

Saving reject records to file is the default.

Procedure

1. Click the dimension table for which you want to specify dimension history options.

2. From the **Edit** menu, click **Properties** .
3. Click the **Dimension History Options** tab.

   Note: This tab is only available if the dimensional history has been set up. For information, see "Set Up Dimensional History" on page 214.
4. In the **Late Arriving Dimension Details** box, click **Write details to file**.
5. If you want to change the default name and location of the reject file, in the **File name** box, enter the full directory path and file name.
6. Click **OK**.

Save details to a relational table:

The table can reside in any writeable Data Manager connection. The default is the same DBMS as the delivered fact build.

Procedure

1. Click the dimension table for which you want to specify dimension history options.
2. From the **Edit** menu, click **Properties** .
3. Click the **Dimension History Options** tab.

   Note: This tab is only available if the dimensional history has been set up. For information, see "Set Up Dimensional History" on page 214.
4. In the **Late Arriving Dimension Details** box, click **Write details to table**.
5. In the **Connection** box, click the connection that contains the table where the details are to be saved.

   Tip: To create a connection, click **New**. For information on creating a connection, see "Create a Database Connection" on page 39.
6. In the **Table name** box, enter the name of the table to which you want to save the details.

   Tip: You can also click the **Browse for table** button and then click the required table from the list in the **Select Table** dialog box.
7. Click **OK**.

Save details ready for use in IBM Cognos Data Manager SQLTXT Designer:

You can save rejected late arriving dimension details ready for use in IBM Cognos Data Manager SQLTXT Designer.

Procedure

1. Click the dimension table for which you want to specify dimension history options.
2. From the **Edit** menu, click **Properties** .

Specify the Source of the Effective Start Date in Initial Records

By default, when you deliver dimension data for a dimension member, the effective start date in the initial record is set according to the template. IBM Cognos Data Manager uses source dates for the effective start and end dates when tracking dimensional history updates.

However, it is possible that the effective start date in the initial record for each dimension member may differ, as illustrated below.

Product Code	Surrogate Key	Specification	Effective Date	End Date
P1	1	Blue pen	2000/11/01	
P2	2	White paper	2003/11/01	

It may be important to override the initial start date in the template settings for the first record in a dimensional history load. This could occur, for example, if you want the start date for the initial record to be set to a date that defines the beginning of all records. The initial start date could also be set to a null date.

You can define the source for the first effective start date in a sequence using the Effective Start Date in initial records options on the Dimension History Options tab. The source can be

- a source attribute

  Data Manager sets the initial start date according to the date stored in the source column.
- a reference template

  This allows you to specify how you want the start date to be set.

For more information on setting the effective start date in a template, see “Specify Effective Date Options” on page 220.

Procedure

1. Click the dimension table for which you want to specify dimension history options.
2. From the **Edit** menu, click **Properties** .
3. Click the **Dimension History Options** tab.

   Note: This tab is only available if the dimensional history has been set up. For information, see “Set Up Dimensional History” on page 214.
4. In the **Effective Start Date in initial records** box, you specify the source to use for the effective start date in the initial dimension record. Click
   - **From source attribute** to set the effective start date to the date specified for each dimension member in the dimension data. (default)
   - **According to reference template** to use a template to set the effective date for all dimension members.
5. Click **OK**.

Specify Effective Date Options

You can specify the way in which the effective start date and effective end date are set in the template referenced by a dimension table. This controls the way in which the date sequence of dimensional changes occurs.

For example, sometimes the grain of dimensional updates requires that changes occur throughout a day, and sometimes only whole days are required.

Note: Changing the template affects effective date options for all dimension tables referencing that template. For more information, see Chapter 9, “Templates,” on page 51.

Procedure

1. Click the dimension table for which you want to specify effective date options.
2. From the **Edit** menu, click **Properties** .
3. Click the **Columns** tab, and then click **Edit**.

   The **Template Properties** window appears.
4. Click the **Effective Date Options** tab.
5. To specify the format to use for effective dates, in the **Effective Date granularity** box click
   - **Timestamp (date and time)** if dates should include the date and time. (default)
   - **Date only** if dates should not include the time.
6. To specify how the effective start date in the initial record for a specific dimension member is set, in the **Effective Start Date in initial records** box, click
   - **Use data timestamp value** to specify that the date should correspond to the timestamp set by the dimension build.
   - **Variable** to use a variable to set the date, and then in the adjacent box, type the variable name.

     Note: The variable must be defined on the **Variables**tab. For information, see Chapter 24, “Variables,” on page 289.
   - **Date (yyyy-mm-dd hh:mm:ss)** to set an explicit date and time, and then in the adjacent box, type the date and time.

     Note: When **Date only**is selected in the **Effective Date granularity**box, you cannot include the time.
   - **Null** to specify that no date should be set.
7. You specify how the effective end date should be set when a change in the dimension record is detected in the **Effective End Date in current records** box. For information, see step 6.
8. To specify how the effective end date for the previous row should be set when a new dimension data row is created, in the **Set previous record Effective End Date to** box, click
   - **New Effective Start Date** to set the date to be the same as the effective start date of the new dimension data row.
   - **New Effective Start Date - 1 day or second** to set the date to the effective start date of the new dimension data row minus one day or one second. (default)

Note: When **Date only**is selected in the **Effective Date granularity**box, the **New Effective Start Date** option is set minus 1 day. If **Timestamp (date and time)**is selected, the option is set to minus 1 second.

9. Click **OK**.

Define the Index Properties

When you create an index for a dimension build, you define its properties in the Index Properties dialog box.

Procedure

1. Click the table for which you want to define the index.
2. From the **Edit** menu, click **Properties** .
3. Click the **Columns** tab.
4. In the **Index** column, select the check box for the column to index, and then click the **Browse** button ... that appears.
5. In the **Name** box, type a name for the index.
6. In the **Type** box, click the index type to create:
   - **Unique** for an index that constrains the dimension data so that each value in the indexed column is unique.
   - **Bitmap** for an index that is of a compressed format which some database management systems implement. Using an index of this type can enhance search performance.

   Note: **Note:** If the **Type** box is empty, the default index type for the target DBMS is created.

   For more information, see the manufacturer's documentation for the type of database to which you are delivering.
7. In the **Options** box, type any additional options for the index. The contents of this box are appended to the CREATE INDEX statement.

   For information about valid options, see the documentation for the target DBMS.
8. If you want the index to continue generating when errors are encountered, select the **Suppress errors** check box.
9. If you want the index deleted and then recreated, select the **Recreate index** check box. This method is quicker than updating the existing index.
10. Click **OK**.

Recommended Indexes for Dimension Tables

IBM Cognos Data Manager manages dimension tables by issuing SQL requests to insert and update dimension table records. The speed of processing dimension table updates can be affected by the indexes that are applied to the table in the database. The surrogate key of the lowest level of any hierarchy defined in the dimension is typically the primary key of the dimension table. This column should be set as the primary key for the table in the DBMS. Updates to the dimension table are typically constrained with an SQL WHERE clause, using the business keys of the levels on any hierarchy defined in the dimension. It is therefore recommended that non-unique indexes are implemented on columns that are marked as business keys in the delivery template.

Set the Date Source for Timestamping Data

When you execute a dimension build, records that contain attributes with create date or update date behavior are timestamped.

For more information about these attributes, see "Behavior Types in Templates" on page 54.

By default, the data is timestamped using the current system date and time of the computer on which the build is executed. If you require, you can timestamp the data using a different data source. For example, if your target database server is situated on a different computer to the one on which IBM Cognos Data Manager is running, it may be important that you timestamp data from the database server.

Note: You can use this setting to control the date source for all date attributes, unless the effective start date attribute is sourced from a column in the source template. Where the effective start date attribute is mapped from the source, this setting controls only the create date and update date attributes. For more information, see "Specify Effective Date Options" on page 220.

Procedure

1. Click the required dimension build.
2. From the **Edit** menu, click **Properties** .
3. Click the **General** tab.
4. In the **Data timestamp** box, choose the date source to be used for timestamping data:
   - Click **Client system date/time** to set the date to the current system date and time on the computer on which the build is executed. (default)
   - Click **Variable** to use a variable to set the date and, in the adjacent box, type the variable name.

     Note: The variable must be defined on the **Variables**tab. For information, see Chapter 24, "Variables," on page 289.
   - Click **Date (yyyy-mm-dd hh:mm:ss)** to set an explicit date and time and, in the adjacent box, type the date and time.
5. Click **OK**.

Delete a Dimension Build

You can delete a dimension build that is no longer required.

Procedure

1. Click the required dimension build.
2. From the **Edit** menu, click **Delete**.
3. In the message box, click **Yes**.

   If deleting the dimension build results in a template being unused, the **Delete Templates** dialog box appears.
4. Click **Yes** to delete the templates listed, or **No** to delete the dimension build, but not the template.

Examples of how IBM Cognos Data Manager Maintains Dimension Tables

The examples that follow are based on a simple star schema, and describe how dimension tables are maintained in IBM Cognos Data Manager.

The basic requirement of a star table is that there

- is a column for a business key
- may be columns that are used to maintain the table
- may be further columns representing various attributes of the key. These attributes may or may not represent possible aggregations of the key.

Example 1: Simple Dimension Table With No Tracking Behavior

A simple dimension table must include a business key and may include a surrogate key (although recommended, it is not mandatory) and further attributes of the business key

The Product dimension build in the DS_Tutorial catalog is an example of a simple dimension table.

The following are possible scenarios when adding records to a simple dimension table.

Scenario	Response
The key does not exist.	A record is created in the dimension table, and if required, a new surrogate is generated.
The key does exist and one or more attributes have changed.	The attributes are updated.
The key does exist and no attributes have changed.	The record is ignored.

When a simple dimension table is described using a template, this information is recorded.

Column	Behavior	Meaning
Key	Business key	Identifies the dimension record, but it is not necessarily unique within the dimension table.
Surrogate key	Surrogate key	A surrogate value generated in response to a new dimension record. It is linked to a business key.
Attribute 1, 2, 3, and so on	Normal No tracking behavior defined in dimension table	Attributes that will be overwritten when they change. In other words, type 1 changes are performed on these attributes.

Example 2: Dimension Table With Attribute Change Tracking Behavior

A dimension table that tracks attribute changes builds on the action of the simple dimension table.

The StaffSCD dimension build in the DS_Advanced catalog is an example of a dimension table that tracks attribute changes.

The following are possible scenarios when adding records to a dimension table that tracks attributes changes.

Scenario	Response
The key does not exist.	A record is created in the dimension table, and a new surrogate is generated.
The key does exist and one or more type 1 attributes have changed.	The attributes are updated for any record in the dimension table that has the same business key as the level for which the attribute has changed. The update date column is updated for all affected records.
The key does exist and one or more type 2 attributes have changed.	A record is created in the dimension table, and a new surrogate is generated. The end date and current indicator are updated for the previous record, according to the options defined in the delivery template. The effective date and current indicator are set for the new record, according to the rules defined in the delivery template.
The key does exist and type 1 and type 2 attributes have changed.	A record is created in the dimension table, and a new surrogate is generated. The new record contains the current view of the dimension with all current attributes. All type 1 attributes in non-current records are updated to the current type 1 value, if the records have the same business key that is associated with the updated type 1 attribute. All previous records retain the past values for attributes marked as type 2. Date attributes and current indicator flags are updated according to the rules defined in the delivery template.
The key does exist and no attributes have changed.	The record is ignored.

The objective of such a dimension is to track the changing values of attributes over time. In other words, rather than just overwriting changed attributes, the history within a single table is tracked. There are some additional table columns that make the process of tracking the changing values simpler. These table columns are all concerned with tracking the behavior of the table rows.

Column	Behavior	Meaning
Key	Business key	Identifies the dimension record, but it is not necessarily unique within the dimension table.
Surrogate key	Surrogate key	A surrogate value generated in response to a new dimension record. It is linked to a business key.
Attribute 1, 2, 3, and so on	Normal No tracking behavior defined in dimension table	Attributes that will be overwritten when they change. In other words, type 1 changes are performed on these attributes.
Attribute 4, 5, 6, and so on	Normal Tracking behavior defined in dimension table	Attributes for which a dimension record will be created when they change. In other words, type 2 changes are performed on these attributes.
eff_date	Effective start date	The date the record became effective. If the same business key was already in the table, all previous records become ineffective (see end_date and curr_ind).
end_date	Effective end date	The date the record became ineffective due to the generation of a new dimension record for the same business key. For a current record, this value is set according to the template properties.
curr_ind	Current indicator	A pair of values used to indicate current and past records.
cre_date	Create date	The date the record was created. This date is always the same as udt_date.

Column	Behavior	Meaning
udt_date	Last update date	The last date when the record changed. This could be because an attribute was updated with a type 1 change.

In this case the template has considerable knowledge of the meaning of columns in the dimension table. These column behaviors are used both to maintain the dimension table, and to access the table data. In maintaining a dimension table it is the current dimension values that must be compared against new records and potentially be updated. This means that fast access to current values is important.

These dimension values are checked in the following order when trying to locate a current dimension record.

1. The current indicator.

   This is the simplest and fastest way to locate current dimension records.
2. The effective start date.

   If the current indicator is not stored in dimension records, IBM Cognos Data Manager searches for the record with the latest effective start date. Access to current values is slower because a sort must be performed to find the latest record.

In summary, when delivering dimension data, Data Manager searches the dimension table for the business key value of the row to be delivered. If this search shows that the business key value exists in the dimension table, Data Manager determines whether any attributes have changed value, and updates the existing record, or creates a new record accordingly.

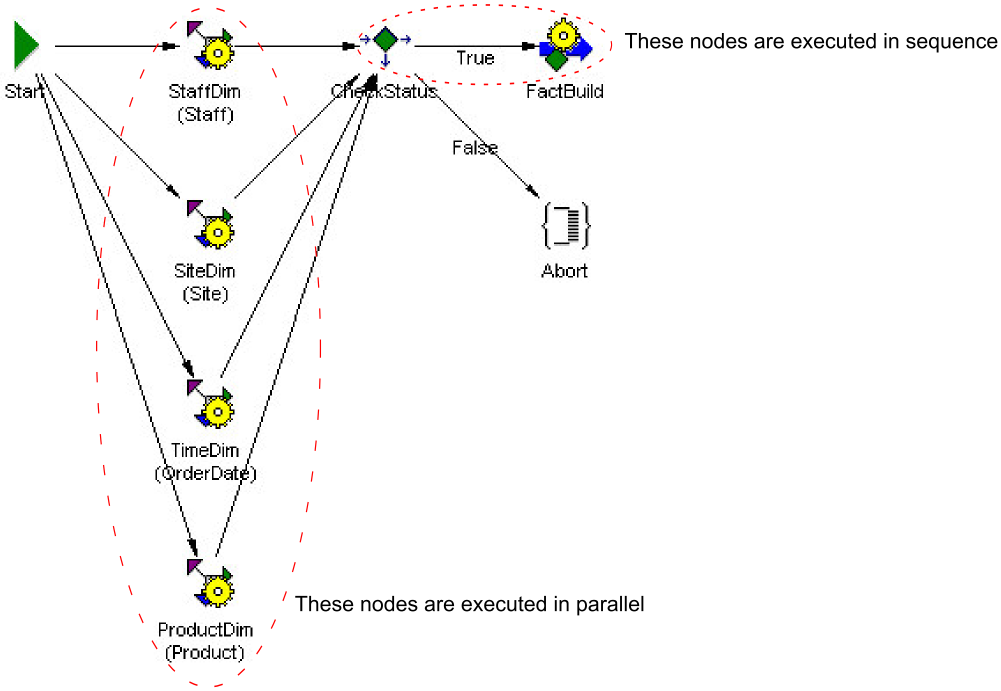
Chapter 19. Managing Builds Using JobStreams

You can automate tasks related to managing the build process by creating a JobStream to contain a series of steps. For example, a JobStream can

- coordinate fact and shared dimension builds
- prefix dimension builds with staging builds
- perform pre and post processing SQL
- set different arrival rates of source data
- distribute tasks over multiple processors
- customize application logging
- provide build status

A JobStream contains a node for each step in a process, which can be a fact build, dimension build, SQL statement, condition, procedure, alert, email, or another JobStream. Each JobStream can use any number of these nodes.

You link the nodes together to create a flow to instruct IBM Cognos Data Manager of the order in which each node is to be processed. Each JobStream can have a single flow or multiple flows. Data Manager can execute a JobStream in sequence or in parallel.



You can execute any JobStream from the command line, or integrate it into an application or scheduling architecture. For more information, see "Execute a JobStream on your Local Computer" on page 248.

Fact Build Node

You can include any fact builds from the current catalog in a JobStream.

For more information, see Chapter 13, “Fact Builds,” on page 105.

Dimension Build Node

You can include any dimension builds from the current catalog in a Jobstream.

For more information, see Chapter 18, “Dimension Builds,” on page 203.

SQL Node

An SQL node can contain a series of SQL statements that are processed when the SQL node is reached in the JobStream flow. An SQL node is useful for pre and post table processing. Data Manager automatically commits successful SQL statements, but rolls back if the statement fails.

Note the following:

- Each SQL node is executed in a database transaction. Some databases have restrictions on the type of statements that can be executed in a database transaction. For specific examples, see the vendor's documentation for your database.
- SELECT statements cannot be used in SQL nodes.

Procedure Node

A procedure node can contain one or more Data Manager functions or statements. A procedure node is useful for checking for input files and generating custom logging and auditing messages.

Condition Node

A condition node provides branching between nodes for conditional execution. Each condition node can have many nodes linking to it, but only two output links, True and False.

JobStream Node

A JobStream node is a JobStream within a JobStream. This allows larger jobs to be broken down into a series of smaller jobs. When a JobStream node is encountered, all the steps within the JobStream node are processed before processing of the next node in the sequence starts.

Alert Node

An alert node writes a user-defined audit record of the type ALERT, into the Data Manager audit tables. You can use these records to record specific events that occur during JobStream and build execution.

You can also use other tools to access the audit tables, for example, custom audit reports using IBM Cognos Business Intelligence.

Email Node

An email node sends event notifications to mail systems. For example, you can set up emails to provide notifications when a JobStream has completed or failed. You can also include attachments with emails.

The method used to send an email varies according to whether or not you are using IBM Cognos BI.

If you are not using IBM Cognos BI, emails are sent using MAPI. However, if IBM Cognos BI is being used, you can specify that emails are sent using the delivery service. You do this by declaring the DM_EMAIL_VIA_SERVICE environment variable.

For more information, see "DM_EMAIL_VIA_SERVICE" on page 295.

Note: Email attachments are not supported on the UNIX operating system.

DataStage Node

A DataStage® node executes an IBM DataStage job using the Data Manager engine. Using a DataStage node in a Data Manager JobStream allows you to separate the processing of data as follows:

- extract and transform source data using DataStage
- deliver transformed data into data marts using Data Manager

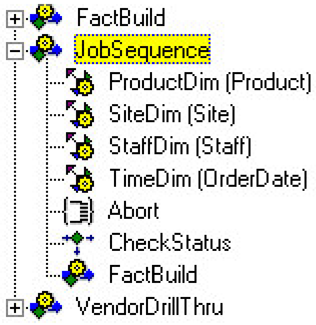
Before you can use DataStage nodes in a JobStream, you must install the Information Server Client.

View JobStreams

You can view JobStreams in the Tree pane or the Visualization pane.

View JobStreams in the Tree Pane

Each JobStream in the Builds and JobStreams folder is represented in the Tree pane as follows.



FactBuild
JobSequence
ProductDim (Product)
SiteDim (Site)
StaffDim (Staff)
TimeDim (OrderDate)
Abort
CheckStatus
FactBuild
VendorDrillThru

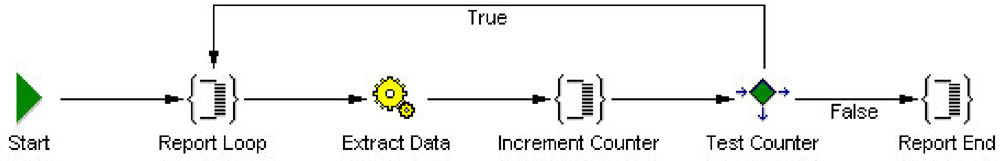
Note: If the associated build or JobStream name differs from the node name, it is shown in brackets.

Tip: You can view just the JobStreams contained in a catalog by clicking the JobStreams tab at the bottom of the Tree pane.

View JobStreams in the Visualization Pane

When you click a JobStream in the Tree pane, a visual representation of the whole JobStream is shown in the Visualization pane.

Viewing a JobStream in this way makes it easier to understand the JobStream flow.



For information about each of the symbols used in the Visualization pane, see Appendix F, “Buttons and Icons,” on page 497.

Note: If the associated build or JobStream name differs from the node name, it is shown in brackets.

You can add, link, and reposition nodes, show the properties for a node, and convert a node to a JobStream from the Visualization pane.

By default, the nodes remain in exactly the position where you placed them. However, you can specify that IBM Cognos Data Manager arranges the nodes in what it considers to be the most logical way. You can also choose to align nodes to an invisible grid. When you save a catalog, the nodes remain in position, whether they were placed manually by you or automatically by Data Manager.

Notes

- If you include a JobStream in a catalog backup or component package, the current layout is stored as an attribute of the JobStream.
- If you are using source code control, you must first check out the JobStream to change the layout.

Use automatic layout to position the nodes

By default, the nodes remain in exactly the position where you placed them. However, you can specify that IBM Cognos Data Manager arranges the nodes in what it considers to be the most logical way.

Procedure

From the **View** menu, click **Layout the diagram automatically** .

Use the grid to position the nodes

You can choose to align nodes to an invisible grid. When you save a catalog, the nodes remain in position, whether they were placed manually by you or automatically by Data Manager.

Procedure

1. From the **View** menu, click **Snap to Grid**.

   Until you turn off this feature, all the nodes that you add are automatically aligned to the grid.

2. To align existing nodes to the grid, select the node or nodes, and from the **View** menu, click **Align selected objects to the Grid** .

   Tip: To select more than one node, hold down the Ctrl key and click each node. Alternatively, click within the Visualization pane close to the first node that you want to select, keep the mouse button depressed, and drag the pointer so that all the required nodes are selected, and then release the mouse button.

Create a JobStream

This section describes how to create a JobStream.

Procedure

1. Click the **Builds and JobStreams** folder .
2. From the **Insert** menu, click **JobStream**, and then click **JobStream**.
3. Click the **General** tab.
4. In the **Name** box, type a name for the new JobStream and, if you require, a business name and description.
5. Click **OK**.

   The JobStream appears in the JobStream tree and in the **Visualization** pane. Note that in the Visualization pane, at this stage the JobStream is represented as a start node.

Define Logging and Audit Information for a JobStream

When you execute a JobStream, progress information is returned to the log file and audit tables. You can specify the type of information that is recorded.

For more information, see “Execution Log Files” on page 255 and “View the Audit Trail Details” on page 261.

Define JobStream Variables

If any nodes in a JobStream use variables, you must define those variables.

Procedure

1. Click the JobStream.
2. From the **Edit** menu, click **Properties** .
3. Click the **Variables** tab.
4. Define the variables as required.

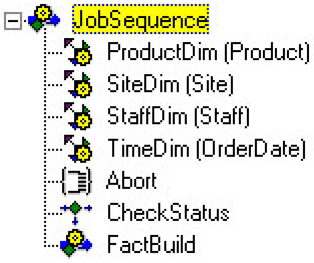
   For information about defining variables, see Chapter 24, “Variables,” on page 289.

   Tip: After creating a JobStream, if you want to change the name of a variable, you must update all references to that variable.
5. Click **OK**.

Add a Fact Build, Dimension Build, or JobStream Node to a JobStream

When you add a fact build, dimension build, or JobStream node, you associate the relevant build or JobStream with the node.

It is possible for the business name that you assign to the node to differ from the actual name of the associated build or JobStream. In this instance, the node is shown in the Tree pane with the associated build or JobStream name in brackets.



JobSequence
ProductDim (Product)
SiteDim (Site)
StaffDim (Staff)
TimeDim (OrderDate)
Abort
CheckStatus
FactBuild

Procedure

1. Click the JobStream to which you want to add a fact build, dimension build, or JobStream node.
2. From the **Insert** menu, click **JobStream**, and then click either
   - **Fact Build Node**
   - **Dimension Build Node**
   - **JobStream Node**

   You can also add a fact build, dimension build, or JobStream by clicking on the arrow to the right of the relevant toolbar button. The **Item List** appears listing all the available components for the selected node type. Double-click the required component.

   If you use the toolbar button or right-click menu to add a node, the pointer changes to a cross-hair allowing you to choose the position where the new node is to be added, and then the **Properties** window appears.
3. Click the **General** tab.
4. If you require, type a business name and a description.

   Note: You do not have to type a name (or ID) for the node, as IBM Cognos Data Manager allocates this automatically, and shows it at the bottom right of the window.
5. In the **Associated build** or **Associated JobStream** box, click the browse button ... and click the build or JobStream to use.

   You can choose from all the fact builds, dimension builds, or JobStreams in the current catalog.
6. Select or clear the **Exclude this node from processing** check box.

   For information, see "Exclude Nodes from Processing in a JobStream" on page 249.
7. Click the **Details** tab.
8. In the **Action on failure** box, click the action to take should a problem be encountered when processing the node.

   For information, see "Specify Error Handling in JobStream Nodes" on page 240.
9. If required, in the **Weight** box, type a suitable value to indicate the total load required to process the node.

For information, see “Specify the Load Required to Process a Fact Build or Dimension Build Node” on page 240.

10. The **Result variable** box shows the name of the variable that is to receive the result of processing the node. By default, this is RESULT however, you can change this if you require. If you specify another variable, you must add the variable to the **Variables** tab of the **JobStream Properties** window.

    For information, see “Define JobStream Variables” on page 231.
11. Click **OK**.

Add a Procedure or Condition Node to a JobStream

You can add procedure and condition nodes to a JobStream.

Procedure

1. Click the JobStream to which you want to add a procedure or condition node.
2. From the **Insert** menu, click **JobStream** and then click either
   - **Procedure Node**
   - **Condition Node**

   Note: If you use the toolbar button or right-click menu to add a node, the pointer changes to a cross-hair allowing you to choose the position where the new node is to be inserted, and then the **Properties**window appears.
3. Click the **General** tab.
4. If you require, in the **Business name** box, type a name for the node and, in the **Description** box, type a description.

   Note: You do not have to type a name (or ID) for the node, as IBM Cognos Data Manager allocates this automatically, and shows it at the bottom right of the window.
5. Select or clear the **Exclude this node from processing** check box.

   For information, see “Exclude Nodes from Processing in a JobStream” on page 249.
6. If you are creating a procedure node and you want the node to run as a separate process rather than the default or running inline, select the **Run as a separate process** check box.

   For information, see “Execute Procedure and SQL Nodes as Separate Processes” on page 250.
7. Click the **Action** tab.
8. In the right pane of the **Action** box, enter an expression to specify the action that Data Manager should take.

   You can type the expression, and you can use the tree structure in the left pane to select the items from the following folders:
   - **Operators** to select operators
   - **Functions** to select pre-defined and user-defined functions
   - **Control** to select controls
   - **Variables** to select variables and substitution variables

   For information about operators, controls, and pre-defined functions, see the IBM Cognos*Function and Scripting Reference Guide*. For information about user-defined functions, see Chapter 23, “User-Defined Functions,” on page 277. For information about variables and substitution variables, see Chapter 24, “Variables,” on page 289.

You can select an item from a folder by dragging it to the right pane or by double-clicking it.

9. If required, test the expression. For information, see Chapter 25, “Testing Expressions and Scripts,” on page 309.
10. Click the **Details** tab.
11. In the **Action on failure** box, click the action to take should a problem be encountered when executing the node.

    For information, see “Specify Error Handling in JobStream Nodes” on page 240.
12. The **Result variable** box shows the name of the variable that is to receive the result of executing the node. By default, this is RESULT however, you can change this if you require. If you do specify another variable, you must add the variable to the **Variables** tab of the **JobStream Properties** window.

    For information, see “Define JobStream Variables” on page 231.
13. Click **OK**.

Add an SQL Node to a JobStream

You can add SQL nodes to a JobStream.

Procedure

1. Click the JobStream to which you want to add an SQL node.
2. From the **Insert** menu, click **JobStream** and then click **SQL Node** SQL .

   Note: If you use the toolbar button or right-click menu to add a node, the pointer changes to a cross-hair allowing you to choose the position where the new node is to be inserted, and then the **Properties**window appears.
3. Click the **General** tab.
4. If you require, in the **Business name** box, type a name for the node and, in the **Description** box, type a description.

   Note: You do not have to type a name (or ID) for the node, as IBM Cognos Data Manager allocates this automatically, and shows it at the bottom right of the window.
5. Select or clear the **Exclude this node from processing** check box.

   For information, see “Exclude Nodes from Processing in a JobStream” on page 249.
6. If you want the node to run as a separate process rather than the default or running inline, select the **Run as a separate process** check box.

   For information, see “Execute Procedure and SQL Nodes as Separate Processes” on page 250.
7. Click the **SQL** tab.
8. In the **Database** box, click the target database to which the SQL statements apply.

   Tip: Click **New** to create a connection. For information, see Create a Database Connection.
9. In the **SQL** box, enter the required SQL statements.

You can click **SQL Helper** to help you construct statements. For information about using SQL Helper, see Chapter 6, “SQLTerm and SQL Helper,” on page 31.

10. Select the **Cognos SQL** check box to specify that you want to use Cognos SQL when you construct the SQL statement. If you clear this check box, you must use native SQL for database you are accessing.

    Note: The default for the **Cognos SQL**check box is determined by whether or not you selected the **Cognos SQL**check box in the **Connection Properties**dialog box.
11. Click the **Details** tab.
12. In the **Action on failure** box, click the action to take should a problem be encountered when processing the node.

    For information, see “Specify Error Handling in JobStream Nodes” on page 240.
13. The **Result variable** box, shows the name of the variable that is to receive the result of processing the node. By default, this is RESULT however, you can change this if you require. If you do specify another variable, you must add the variable to the **Variables** tab of the **JobStream Properties** window.

    For information, see “Define JobStream Variables” on page 231.
14. Click **OK**.

Add an Alert Node to a JobStream

You can add alert nodes to a JobStream.

Procedure

1. Click the JobStream to which you want to add an alert node.
2. From the **Insert** menu, click **JobStream** and then click **Alert Node** .

   Note: If you use the toolbar button or right-click menu to add a node, the pointer changes to a cross-hair allowing you to choose the position where the new node is to be inserted, and then the **Properties** window appears.
3. Click the **General** tab.
4. If you require, in the **Business name** box, type a name for the node and, in the **Description** box, type a description.

   Note: You do not have to type a name (or ID) for the node, as IBM Cognos Data Manager allocates this automatically, and shows it at the bottom right of the window.
5. Select or clear the **Exclude this node from processing** check box.

   For information, see “Exclude Nodes from Processing in a JobStream” on page 249.
6. Click the **Action** tab.
7. In the **Item** box, type the topic of the message.
8. In the **Message** box, type the text of the message.
9. Click the **Details** tab.
10. In the **Action on failure** box, click the action to take should a problem be encountered when processing the node.

    For information, see “Specify Error Handling in JobStream Nodes” on page 240.

11. The **Result variable** box, shows the name of the variable that is to receive the result of processing the node. By default, this is RESULT however, you can change this if you require. If you do specify another variable, you must add the variable to the **Variables** tab of the **JobStream Properties** window.

    For information, see “Define JobStream Variables” on page 231.
12. Click **OK**.
13. Specify that alert messages for the JobStream are to be recorded in the audit trail. For information, see “View the Audit Trail Details” on page 261.

Add an Email Node to a JobStream

You can add email nodes to a JobStream.

Procedure

1. Click the JobStream to which you want to add an email node.
2. From the **Insert** menu, click **JobStream** and then click **Email Node** .

   Note: If you use the toolbar button or right-click menu to add a node, the pointer changes to a cross-hair allowing you to choose the position where the new node is to be inserted, and then the **Properties**window appears.
3. Click the **General** tab.
4. If you require, in the **Business name** box, type a name for the node and, in the **Description** box, type a description.

   Note: You do not have to type a name (or ID) for the node, as IBM Cognos Data Manager allocates this automatically, and shows it at the bottom right of the window.
5. Select or clear the **Exclude this node from processing** check box.

   For information, see “Exclude Nodes from Processing in a JobStream” on page 249.
6. Click the **Action** tab.
7. In the **Profile** box, type the email profile for the computer that you are using.

   For information, see the documentation for the operating system that you are using.
8. In the **Password** box, type the password that you use to access email.
9. In the **To** box, type the email address of the person to whom you want to send the email. If there is more than one recipient, separate each name using a semicolon.
10. In the **Cc** box, type the email address of any recipients to whom you want to send a copy of the email. If there is more than one recipient, separate each address using a semicolon.
11. In the **Bcc** box, type the email address of any recipients to whom you want to send a copy of the email, without the recipient's name being visible to other recipients of the email. If there is more than one recipient, separate each address using a semicolon.
12. In the **Subject** box, type the topic of the message.
13. In the **Attachment** box, type the full file path and name of the file that you want to send with the email message. You can send a maximum of 10 attachments, with each file separated by a comma.

    Note: Email attachments are not supported on the UNIX operating system.

14. In the **Message** box, type the text of the message.
15. Click the **Details** tab.
16. In the **Action on failure** box, click the action to take should a problem be encountered when processing the node.

    For information, see “Specify Error Handling in JobStream Nodes” on page 240.
17. The **Result variable** box, shows the name of the variable that is to receive the result of processing the node. By default, this is RESULT however, you can change this if you require. If you do specify another variable, you must add the variable to the **Variables** tab of the **JobStream Properties** window.

    For information, see “Define JobStream Variables” on page 231.
18. Click **OK**.

Add a DataStage Node to a JobStream

You can add DataStage nodes to a JobStream.

Procedure

1. Click the JobStream to which you want to add a DataStage node.
2. From the **Insert** menu, click **JobStream** and then click **DataStage Node** .

    Note: If you use the toolbar button or right-click menu to add a node, the pointer changes to a cross-hair allowing you to choose the position where the new node is to be inserted, and then the **Properties**window appears.
3. Click the **General** tab.
4. If you require, in the **Business name** box, type a name for the node and, in the **Description** box, type a description.

    Note: You do not have to type a name (or ID) for the node, as IBM Cognos Data Manager allocates this automatically, and shows it at the bottom right of the window.
5. Select or clear the **Exclude this node from processing** check box.

    For information, see “Exclude Nodes from Processing in a JobStream” on page 249.
6. Click the **Action** tab.
7. Enter the required connection details for IBM DataStage in the **Domain**, **Username**, and **Password** boxes.
8. Enter the details of the DataStage job to execute in the **Server**, **Project**, and **Job** boxes.
9. Select the **Reset if required** box to automatically reset a DataStage job that has stopped or aborted.
10. If the DataStage job contains parameters, you must specify the parameter values to use when the DataStage node is executed.

    Click **Parameters**.
11. Click **Add**, and enter the parameter name (as defined in the DataStage job) and the required value.
12. Repeat step 11 add any further parameters as required.
    - To edit the details of a parameter, select it and click **Edit**.
    - To delete a parameter, select it and click **Delete**.
13. Click the **Details** tab.

14. In the **Action on failure** box, click the action to take should a problem be encountered when processing the node.

    For information, see “Specify Error Handling in JobStream Nodes” on page 240.
15. The **Result variable** box, shows the name of the variable that is to receive the result of processing the node. By default, this is RESULT however, you can change this if you require. If you do specify another variable, you must add the variable to the **Variables** tab of the **JobStream Properties** window.

    For information, see “Define JobStream Variables” on page 231.
16. Click **OK**.

Delete a Node from a JobStream

You can delete a node that is no longer required from a JobStream.

Procedure

1. Click the node that you want to delete.
2. From the **Edit** menu, click **Delete**.
3. In the message box, click **Yes**.

Link Nodes in a JobStream

Each node can have one or more nodes linked to it and must, in turn, link to at least one other node. The exception is a condition node, which must link to no more than two nodes; one link must be True, the other False.

Processing begins at the Start node and then progresses through the JobStream following the links that you add. When a JobStream node is encountered, the whole JobStream is processed before progressing to the next node.

You can link nodes using the Visualization pane or the Predecessors and Successors tabs in the node Properties windows.

Link nodes in the Visualization pane

You can link nodes using the Visualization pane or the Predecessors and Successors tabs in the node Properties windows.

Procedure

1. In the Tree pane, click the JobStream for which you want to add links.
2. Click **Insert Link** .

   An arrow appears above the pointer to indicate the direction of the link.

   Note: Link mode is automatically cleared, when you next right-click the **Visualization**pane.
3. Click the node that is to be the starting point for the link, and drag the pointer to the node to which it is to link.

   If the link is from a condition node, a status of True is allocated to the first link that you create and False to the other. However, you can reverse the logic if you want. For information, see “Change the True and False Status for a Condition Node” on page 239.
4. Repeat steps 2 and 3 to create additional links.
   - To delete a link, click the link in the **Visualization** pane, and from the **Edit** menu, click **Delete**, or press the Delete key.

- To return to select mode, click **Select mode** .

Link nodes in the node Properties window

You can link nodes using the Visualization pane or the Predecessors and Successors tabs in the node Properties windows.

Procedure

1. Click the required node.
2. From the **Edit** menu, click **Properties** .
3. Click the **Predecessors** tab.
4. Select the check boxes adjacent to the nodes from which you want to link; that is, the nodes immediately before the selected node.
5. Click the **Successors** tab.

   Note: If you are linking from a condition node, the **Successors** tab is replaced with **True** and **False**tabs. Click the **True**tab to specify the true link, and the **False** tab to specify the false link.
6. Select the check boxes adjacent to the nodes to which you want to link; that is, the nodes immediately after the selected node.
7. Click **OK**.

Change the True and False Status for a Condition Node

Unlike other nodes, a condition node must link to two nodes. By default, when you link a condition node, a status of True is allocated to the first link that you create and False to the other. However, you can reverse the logic.

Procedure

1. Click **Select Mode** .
2. Click either the **True** or **False** link.
3. From the **Actions** menu, click **Reverse Logic**.

   The True output link changes to False, and the False to True.

Split Large JobStreams into Smaller JobStreams

A JobStream node is one or more JobStreams within a JobStream. This allows larger jobs to be broken down into a series of smaller jobs.

You can convert any single node, or group of nodes within a JobStream to a JobStream node, or you can create a JobStream node.

For information about creating a JobStream node, see “Add a Fact Build, Dimension Build, or JobStream Node to a JobStream” on page 231.

Procedure

1. Click the node that you want to convert.

   Tip: To select more than one node, hold down the Ctrl key and, in the **Visualization** pane, click each node. Alternatively, click within the **Visualization** pane close to the first node that you want to select, keep the mouse button depressed, and drag the pointer so that all the required nodes are selected, and then release the mouse button.

2. From the **Actions** menu, click **Convert to JobStream**.

   The **Visualization** pane is updated to show details of the new JobStream. The JobStream is also added to the **Builds and JobStreams** folder in the **Tree** pane.

   Tip: By default, the JobStream is named JOBSTREAM. To give the JobStream a more meaningful name, from the **Edit** menu, click **Properties**, and type a new name.

Specify Error Handling in JobStream Nodes

You can specify how to handle errors that may be encountered while processing a node. This setting can be different for each node.

Procedure

1. Click the node.
2. From the **Edit** menu, click **Properties** .
3. Click the **Details** tab.
4. In the **Action on failure** box, specify the action to take if a problem is encountered with the node by clicking
   - **Terminate** to stop processing the JobStream that contains the problem (default)

     If the JobStream has more than one flow, IBM Cognos Data Manager continues processing the remaining flows.
   - **Abort** to prevent further processing when all currently running nodes have completed
   - **Continue** to skip the selected node and continue processing all the remaining nodes in the JobStream

Specify the Load Required to Process a Fact Build or Dimension Build Node

If a JobStream is set up to execute nodes in parallel, you can restrict the execution of fact build or dimension build nodes to prevent overloading the computer. This is known as load control.

For each computer on which you install IBM Cognos Data Manager, you can set a variable that indicates the processing capacity of the computer. The value of this variable is an arbitrary number assigned by you. Within a JobStream, you can then assign a weight value to each fact build node or dimension build node that indicates the processing resource required to execute each node. The weight value is also an arbitrary value that you assign. When you execute the JobStream, the total load for the computer is calculated as the sum of the weight of all build nodes that are concurrently processing.

Note: If weight of a single node is greater than the capacity specified for a computer, Data Manager assumes the weight to be equal to the capacity.

When processing a JobStream containing a weighted fact or dimension build node, Data Manager calculates the current load prior to processing the node. If the weight of this node, added to the current load, exceeds the capacity of the computer, Data Manager defers processing the node until sufficient capacity becomes available.

For example, suppose a JobStream contains three fact build nodes: node1 (weight 40), node2 (weight 40), and node3 (weight 40). The processing capacity is 100.

When the JobStream is executed, node1 and node2 can run but node 3 is deferred until either node1 or node2 is complete, because the added weight of node3 exceeds the capacity of the computer.

Note: Load control does not affect the order of processing dependent nodes in a JobStream.

Before you begin

To enable load control, you must set the variable named DM_SERVER_CAPACITY, which is used to define the processing capacity of a computer. For more information, see "DM_SERVER_CAPACITY" on page 295.

Procedure

1. Click the appropriate fact build or dimension build node.
2. From the **Edit** menu, click **Properties** .
3. Click the **Details** tab.
4. In the **Weight** box, type a suitable value to indicate the total load required to process the node.

Specifying the execution mode for a node

You can specify certain nodes within a JobStream to execute in 32-bit or 64-bit mode. It is possible to set the execution mode for the following node types:

- Fact build node
- Dimension build node
- JobStream node
- SQL node
- Procedure node

You can select one of the following execution modes for a node:

- **Default** to allow the JobStream to set the execution mode for the node.

  For more information, see "Execute builds and JobStreams in 64-bit mode" on page 253.
- **32-bit** to force the node to execute in 32-bit mode irrespective of the execution mode used by the JobStream.
- **64-bit** to force the node to execute in 64-bit mode irrespective of the execution mode used by the JobStream.

Procedure

1. Click the appropriate node in the JobStream.
2. From the **Edit** menu, click **Properties**.
3. Click the **Details** tab.
4. In the **Execution mode** box, select the required option for the node.

Chapter 20. Publishing Data Movement Tasks

A published data movement task can comprise a JobStream together with all its nodes, or a single fact build or dimension build. Each data movement task is self-contained. When you publish a data movement task from IBM Cognos Data Manager, you can then use IBM Cognos Connection to run or schedule it. IBM Cognos Connection is the portal to IBM Cognos Business Intelligence and provides a single access point to all corporate data available in IBM Cognos BI.

For more information on using IBM Cognos Connection, see the *IBM Cognos Business Intelligence Administration and Security Guide*.

Note: To use the Data Movement Service, the Data Manager engine and the IBM Cognos BI server components must be installed in the same location. For information, see the *IBM Cognos Data Manager Installation and Configuration Guide*.

Procedure

1. From the **Actions** menu, click **Publish Data Movement Task**.
2. Click the build or JobStream to publish.

   The name of the selected component is shown in the **Enter the name for the published task box**.
3. If required, amend the name for the published task.

   Note: If the name is not unique within the Data Movement Tasks folder in IBM Cognos Connection, a message appears asking if you want to replace the existing file. Click **Yes**or **No**as appropriate.
4. Click **OK**.
5. You can now use IBM Cognos Connection to
   - run the entry
   - view the run history
   - change the default properties

   For information, see the *IBM Cognos Administration and Security Guide*.

Chapter 21. Executing Builds and JobStreams

You execute builds and JobStreams to deliver data to the target database.

When you execute a build or JobStream a command window appears showing the progress of the execution process. During execution, progress information is returned to a log file and audit tables.

Builds and JobStreams can be executed

- on your local computer

  For more information, see "Executing a Fact Build or Dimension Build on your Local Computer" or "Execute Procedure and SQL Nodes as Separate Processes" on page 250.
- on a remote computer

  For more information, see "Execute a Build or JobStream on a Remote Computer" on page 250.
- using the Data Movement service

  For more information, see "Execute a Build or JobStream using the Data Movement Service" on page 251.

Note: You can also execute a DataStream. For information, see "Execute a DataStream" on page 133.

Build Execution Modes

When you execute a build on either your local computer or on a remote computer, you can choose from these execution modes:

Normal Execution Mode

IBM Cognos Data Manager executes the whole build and delivers the required fact data and dimension data.

Object Creation Execution Mode

Data Manager creates the required tables but does not deliver any data.

Check Only Execution Mode

Data Manager executes the build without delivering data. Check only mode is useful for testing a build and for estimating the resources required.

Executing a Fact Build or Dimension Build on your Local Computer

After you define a fact build or a dimension build, you execute it to acquire, transform, and deliver the data.

When a build executes, IBM Cognos Data Manager

- validates the build specification
- checks that the columns in the target tables are the same as those being delivered

- finds all the reference structures required by the build
- builds the reference structures by running the structural source queries
- acquires the transactional data
- transforms (calculates, derives, and aggregates) the data in the context of the loaded reference structures
- delivers the data

Tip: If you want to test the source data to be acquired, prior to executing a build, you can execute the DataStream for the build. For more information, see "Execute a DataStream" on page 133.

Use the default settings to execute a build

You can execute a build using the default settings. The default settings are that normal execution mode is used and fact data and dimension data is delivered (if the build contains both).

Procedure

1. Click the build and then click the **Execute** button ▶.

   If you made changes to the build and did not save them, a message appears.
2. Click **OK** to save the changes.

   The build executes using normal execution mode and fact data and dimension data is delivered.
3. When the execution process is complete, press Enter to close the command window and return to Data Manager.

Change or check the default settings and execute a build

You can change the default settings from the relevant build Properties window, or you can temporarily change the default settings from the Execute Build dialog box.

Procedure

1. Click the build to execute.
2. From the **Actions** menu, click **Execute**.

   If you made changes to the build and did not save them, a message appears.
3. Click **OK** to save the changes.
4. In the **Options** section, click **Execute locally**, and then, in the **Execution mode** box, click the required execution mode.
5. If you are executing a fact build, you can choose to deliver only fact data, only dimension data, or both (providing the fact build contains both types). In the **Deliver** section, select the check boxes for the type of data to deliver.
6. To override the current log file or trace frequency information, in the **Trace** section, select the **Override build settings** check box. Select the type of information to include in the log file, and change the message frequency values as required.

   For more information, see "Specify Log File Details" on page 256, and "Specify the Message Frequency in a Build Log File" on page 257.

   Note: Any changes that you make alter the commands shown in the **Command line** box.
7. In the **Additional options** box, type any additional commands that you want to add to the command line.

The command that Data Manager will execute is shown in the **Command line** box.

Tip: By default, the command window remains open when execution is complete, and you must press Enter to close it. To close it automatically, clear the **Pause on completion** check box.

8. To use the catalog credentials instead of Designer credentials when executing the build, select the **Use catalog credentials** check box.

   For more information on credentials, see "Manage Credentials" on page 47.

9. Click **OK**.

   A command window appears and shows the progress of the execution process.

Check the Target Table Prior to Executing a Build

When IBM Cognos Data Manager executes a build, it automatically checks that the columns in the target table are the same as those being delivered. If they are different, a message appears informing you of the problem.

You can also check the target table independently of an execution process.

Note: When checking the target table, Data Manager ignores any tables for which you have specified that Data Manager should automatically add columns to the table. For information, see "Add Missing Columns to the Target Table" on page 187.

Tip: To stop Data Manager automatically checking the tables during execution, from the **Tools** menu, click **Options**, and then click **Skip table compatibility checks on builds**. If you select this option and the tables are incompatible, the build fails.

Procedure

1. Click the appropriate build.
2. From the **Actions** menu, click **Perform Target Table Check**.

   The columns in the target table are checked to ensure that they are the same as those being delivered. If they are the same, or if you have specified that Data Manager should automatically add columns to the table, a message appears stating that there are no known problems. For information, see "Add Missing Columns to the Target Table" on page 187.

   If there are inconsistencies, this type of message appears.

   ```
   Inconsistencies found in the following tables:
   Connection 'DS_Marts'
   Table 'F_Additional'
    Column '"SalesTotal"' missing
   ```

3. Choose whether to correct the problem, delete the target table, or add the missing columns to the target table.
   - If you do not want to delete the table, click **Cancel**, correct the problem, and repeat steps 1 to 2.
   - If you want to add the missing columns, click **Cancel** and in the **Properties** window for the fact build delivery or the dimension build, click **Automatically add columns to table**.
   - If you want to delete the tables from the target database and recreate them upon delivery of the columns, click **Drop Tables**, and in the message box that appears, click **Yes**.

Fine-Tune the Data Structure

When you execute a fact build, the required fact and dimension tables are created if they do not exist in the target data mart. For most purposes, the structure of the created tables is acceptable. However, you can export to a script file data definition language (DDL) statements to create or drop the target tables and indexes. This allows you to fine-tune the DDL statements and create tables and indexes by running script files in SQLTerm or another suitable program.

Procedure

1. Click the required build.
2. From the **Tools** menu, click **Show DDL**.
3. Click the tab for the required type of DDL statement.
4. Click **Save**.
5. In the **Save DDL** dialog box, type a file name for the script file, and click **Save**.
   - To return to the original DDL statements after you have made changes, click **Reset DDL**.
   - To copy one or more DDL statements to the Microsoft Windows clipboard, select the statement, right-click and then click **Copy**.
6. Click **Close**.

Execute a JobStream on your Local Computer

When you execute a JobStream, a command window appears showing the progress of the JobStream execution process. As each node is processed it returns progress information to the log file and audit tables.

If, during execution, IBM Cognos Data Manager encounters a problem with a node, the execution process may fail. You can specify the action that Data Manager is to take if this happens. When any problems have been resolved you can restart the JobStream starting with the node that failed. For information, see “Specify Error Handling in JobStream Nodes” on page 240, and “Restart Processing a JobStream” on page 250.

When you execute a JobStream, you can

- set the level of audit and logging information recorded
- specify how to handle any errors that may be encountered while processing a node
- exclude nodes from processing
- restart processing a JobStream

Use the default settings to execute a JobStream

This section describes how to execute a JobStream using the default settings.

Procedure

1. Click the JobStream and then click the **Execute** button ▶.

   If you made changes to the JobStream and did not save them, a message appears.
2. Click **OK** to save the changes.
3. When the execution process is complete, press Enter to close the command window and return to Data Manager.

Change or check the default settings and execute a JobStream

This section describes how to change or check the default settings when you execute a JobStream.

Procedure

1. Click the JobStream to execute.
2. From the **Actions** menu, click **Execute**.

   If you made changes to the JobStream and did not save them, a message appears.
3. Click **OK** to save the changes.
4. In the **Options** section, click **Execute locally**.

   Tip: If you want to start executing the JobStream from the point at which a previous attempt failed, select the **Restart Last JobStream** check box.
5. To override the current log file information, in the **Trace** section, select the **Override Job/JobStream settings** check box and then select the type of information to include in the log file.

   Tip: If you are using load control, you can check for deferred JobStream nodes by selecting internal log information.

   For more information, see “Specify Log File Details” on page 256.

   Note: The trace options you select alter the commands shown in the **Command line**box.
6. In the **Additional options** box, type any additional commands that you want to add to the command line.

   The command that Data Manager will execute is shown in the **Command line** box.

   Tip: By default, the command window remains open when execution is complete, and you must press Enter to close it. To close it automatically, clear the **Pause on completion** check box.
7. To use the catalog credentials instead of Designer credentials when executing the build, select the **Use catalog credentials** check box.

   For more information on credentials, see “Manage Credentials” on page 47.
8. Click **OK**.

   A command window appears and shows the progress of the execution process.

Exclude Nodes from Processing in a JobStream

You can specify that you want to exclude from processing either a single node or all the nodes that follow a specified point.

This technique can be used during testing to temporarily disable a node. When a node is disabled, it is shown in the Tree pane and the Visualization pane overlaid with a red line.

Procedure

1. Click the node to exclude.
2. From the **Edit** menu, click **Properties** .
3. Click the **General** tab.

4. Choose whether you want to exclude only the selected node, or the selected node and all the nodes that follow it:
   - To exclude only the selected node, select the **Exclude this node from processing** check box.
   - To exclude all the nodes that follow the selected node, select the **Exclude subsequent nodes in this thread** check box.
5. Click **OK**.

Execute Procedure and SQL Nodes as Separate Processes

By default, a condition node, procedure node, SQL node, alert node, or email node is executed inline, that is, run in the JobStream process. The implication of running inline is that the nodes are run in series even if the JobStream design has parallel flows. However, for procedure nodes and SQL nodes there may be instances where the nodes may take a long time to process so parallel execution would be desirable. To facilitate this, you can specify that IBM Cognos Data Manager should create a separate process for a node.

Note: Executing a node as a separate process uses more memory than executing inline.

Procedure

1. Click the appropriate procedure or SQL node.
2. From the **Edit** menu, click **Properties**.
3. Click the **General** tab.
4. Select the **Run as a separate process** check box.
5. Click **OK**.

Restart Processing a JobStream

If a JobStream stops processing and you resolve the problem that caused the failure, you can restart the JobStream beginning with the node that failed.

Note: If the failure occurs in a nested JobStream (a JobStream within a JobStream), processing restarts with the top level JobStream node, not with the actual node that failed.

Tip: You can specify how IBM Cognos Data Manager should handle errors that may be encountered while processing a node. For information, see “Specify Error Handling in JobStream Nodes” on page 240.

Procedure

1. Click the JobStream.
2. From the **Actions** menu, click **Execute**.

   If you made changes to the JobStream and did not save them, a message appears.
3. Click **OK** to save the changes.
4. Select the **Restart last JobStream** check box, and click **OK**.

Execute a Build or JobStream on a Remote Computer

To control and manage the execution of builds and JobStreams on remote servers you use IBM Cognos Data Manager Network Services.

Data Manager Network Services consists of server and client software. The client software is automatically installed when you install Data Manger, and the server software must be installed on the computer on which the build or JobStream is to be executed. However, both the client and server software can exist on the same computer.

For information on setting up Data Manager Network Services, see the IBM Cognos Data Manager *Installation and Configuration Guide*.

Communication is performed using either network sockets or web services.

Procedure

1. Click the required build or JobStream.
2. From the **Actions** menu, click **Execute**.

   If you made changes to the build or JobStream and did not save them, a message appears.
3. Click **OK** to save the changes.
4. Select **Remote execute**, and in the **Server** box, enter the name of the server on which the build or JobStream is to be executed.
5. Select the **Submit only** check box if the client is to submit the command and then exit, rather than waiting for the command output.
6. Enter other details as required.

   For information, see "Executing a Fact Build or Dimension Build on your Local Computer" on page 245 or "Execute a JobStream on your Local Computer" on page 248.
7. Click **OK**.

Execute a Build or JobStream using the Data Movement Service

Using the Data Movement Service to execute a build or JobStream allows you to run and schedule fact builds, dimension builds, and JobStreams on a remote computer using IBM Cognos Connection. IBM Cognos Connection is the portal to IBM Cognos Business Intelligence and provides a single access point to all corporate data available in IBM Cognos BI.

For more information on using IBM Cognos Connection, see the IBM Cognos Connection *User Guide*.

Notes

- To use the Data Movement Service, the IBM Cognos Data Manager engine and the IBM Cognos BI server components must be installed in the same location. For information, see the IBM Cognos Data Manager *Installation and Configuration Guide*.
- For information on publishing data movement tasks, see Chapter 20, "Publishing Data Movement Tasks," on page 243.
- To populate the audit tables when executing a data movement task for the first time, you must define the audit variables. For more information, see "Populate the audit tables" on page 252

Procedure

1. Click the build or JobStream to execute.
2. From the **Actions** menu, click **Execute**.

If you made changes to the build or JobStream and did not save them, a message appears.

3. Click **OK** to save the changes.
4. In the **Options** section, click **Execute via Data Movement Service**.

   Note: You cannot override the current log file information while executing a build or JobStream via the Data Movement service.
5. Click **OK**.

   The **Execute Data Movement Task** window appears. When execution is complete, the top pane shows the name of the fact build, dimension build, or the JobStream and all its nodes, with an icon indicating whether execution was successful or failed, and the start and end timestamps.

   Tip: To stop execution, click **Stop Executing**.
6. You can choose the audit or log details that you want to view.

   Note: For a JobStream you must first select the required fact build node or dimension build node.

   From the drop down box, click one of the following:
   - Audit Trail
   - Log File
   - Audit Messages

   The relevant information is shown in the bottom pane.
7. From the **File** menu, click **Exit** to close the **Execute Data Movement Task** window.

Populate the audit tables

When you execute a Data Movement task, the audit tables are not automatically populated. To populate the audit tables, you must update the [Variable] section of the dm.ini file, or the environment.

Note: To update the environment, you use property variables. These variables are intrinsic to, and affect the operation of Cognos Data Manager. You can set property variables within a build or JobStream or in the environment of the operating system. For information see,

Procedure

1. Open the dm.ini file.

   Note: By default, this file is stored in Program Files\ibm\cognos\c10\datamanager.
2. In the Variable section, add these variables:

   ```
   DM_AUDIT_DB_DRIVER=auditdb_driver
   DM_AUDIT_DB=auditdb_items
   ```

   For more information, see Chapter 31, “Commands,” on page 377
3. Repeat this for each Cognos Data Manager server required.
4. Save and close the dm.ini file.
5. Stop and then restart the Cognos 10 service.

Results

Each time you execute a build or JobStream using the Data Movement service, if you have set the audit tables preference in the **Execute Data Movement Task** window, the audit tables are populated.

Execute builds and JobStreams in 64-bit mode

If you install the 64-bit hybrid version of IBM Cognos Data Manager, and if the operating system and database drivers you are using are compatible with 64-bit mode, you can execute builds and JobStreams in 64-bit mode.

Note: If you are using a published FM connection using 64-bit mode, only dynamic query mode is supported.

You can execute builds and JobStreams in 64-bit mode from the following places:

- On a local computer
- On a remote computer
- Using the Data Movement Service

Execute a build or JobStream in 64-bit mode on a local computer

You can execute a build or JobStream in 64-bit mode on a local computer using Data Manager Designer or from the command line.

Execute in 64-bit mode using Data Manager Designer

When you use Data Manager Designer to execute a build or JobStream, it executes in 32-bit mode by default. However, you can change this to 64-bit.

Note: If you are running on a 32-bit computer, then only 32-bit execution is available.

You can override the default and force Data Manager to execute in 64-bit mode by following this procedure:

1. Select the build or JobStream to execute.
2. Open the **Execute Build** or **Execute JobStream** dialog box.
3. In the **Application Execution Mode** drop down box, click 64-bit.

Note: For this release *default* equates to 32-bit.

Execute in 64-bit mode using the command line

When you use the command line to execute a build or JobStream in 64-bit mode, you must execute it from the `c10 location/bin64` folder.

For information about default 64-bit installation directories, see the *IBM Cognos Data Manager Installation and Configuration Guide*.

Executing a build or JobStream in 64-bit mode using a remote computer

You can execute a build or JobStream in 64-bit mode on a remote computer (using IBM Cognos Data Manager Network Services) from the command line.

To execute remotely in 64-bit mode, you use the DSREMOTE command. For more information, see “dsremote” on page 385.

Publishing a build or JobStream for 64-bit mode using the Data Movement Service

You can execute a build or JobStream in 64-bit mode using the Data Movement Service. By default, the build or JobStream executes in 32-bit mode. You can override this default and force it to execute in 64-bit mode by following this procedure:

1. Create an empty JobStream.
2. Add the required build or JobStream as a new node to the JobStream.
3. Set the execution mode for the node to 64-bit.

   For more information, see “Specifying the execution mode for a node” on page 241.
4. Publish the Data Movement Service task.

Configure a JobStream Execution to Use IBM Cognos Business Intelligence Load Balancing

If a JobStream contains fact build or dimension build nodes, you can execute these nodes as individual data movement tasks. If your IBM Cognos Business Intelligence installation is configured to use more than one dispatcher, it then distributes and executes the Data Movement tasks using standard IBM Cognos BI load balancing for parallel builds.

For more information on data movement tasks, see Chapter 20, “Publishing Data Movement Tasks,” on page 243.

You can configure a JobStream execution to use IBM Cognos BI load balancing

- using the command line

  You must update the rundsjob options to enable this feature. For more information, see “rundsjob” on page 382.
- using the Data Movement Service

  You must enable the parameter DMS.DistributeBuilds in IBM Cognos Administration. For more information see the IBM Cognos *Administration and Security Guide*.

Run-time Considerations

When configuring JobStream executions to use IBM Cognos BI load balancing, you should consider the following:

- Any changes to JobStream variables made by fact build or dimension build nodes executed as separate Data Movement tasks are not reflected in the JobStream.
- Inline JobStream processing always takes place on the host computer. This may cause communications with the Data Movement Service (executing other nodes in the JobStream) to time out. You can overcome this issue by redesigning the JobStream so that the fact and dimension build nodes are moved into a separate JobStream node within the main JobStream.

Set the Default Execution Method

You can specify the default method to use for build and JobStream execution.

Procedure

1. From the **Tools** menu, click **Options**.
2. Click the **General** tab.
3. In the **Default execution method** box, click either **Local**, **Remote**, or **via Data Movement Service** as required.
4. If you selected **Remote**, in the **Default server** box, enter the name of the server on which builds and JobStreams are to be executed.
5. Click **OK**.

Execution Log Files

When you execute a build or JobStream, a command window appears and shows progress information. This information is saved automatically as a log file. By default, log files are stored in the directory named Program Files\ibm\cognos\c10\datamanager\log, however you can change this.

For more information, see "Change the Default Location for Log Files."

The file name of each log file is <component_type>_<component_name>_<number>.log. Where <component_type> is the type of component (either Build, DimBuild, or Job), <component_name> is the name of the component, and <number> is a four digit number that identifies the execution. For example, the log file for a fact build named NewBuild might be Build_NewBuild_0001.log.

Note: If a component name contains characters other than letters and numbers, these are replaced with an underscore in the log file name.

You can specify the level of detail to include in the log files, and view the log files. For information, see "Specify Log File Details" on page 256, and "View the Log Files" on page 258.

If you are using IBM Cognos Data Manager with multilingual or unicode data, you should specify unicode encoding for your log files. By default, the encoding used is platform dependent. For more information, see the IBM Cognos Data Manager *Installation and Configuration Guide*.

Change the Default Location for Log Files

By default, log files are stored in the directory named Program Files\ibm\cognos\c10\datamanager\log. You can change this using

- the -V parameter with the databuild, dimbuild, or rundsjob command

  The following example uses a command-line parameter to direct text data files to directory c:\log\datamanager

  ```
  databuild ODBC DSN=TestCat example -VDS_LOG_DIR=c:\log\datamanager
  ```

  For information, see Chapter 31, "Commands," on page 377.
- the DS_LOG_DIR system variable

  For information, see "DS_LOG_DIR" on page 296.
- IBM Cognos Configuration

For information, see the IBM Cognos Configuration *User Guide.*

Setting the location using the databuild, dimbuild, or rundsjob command takes precedence over a location set using the DS_LOG_DIR system variable or IBM Cognos Configuration, and using the DS_LOG_DIR system variable takes precedence over IBM Cognos Configuration.

Specify Log File Details

You can specify that IBM Cognos Data Manager records these types of information in the log file.

Information type	Description
Progress	Shows the overall progress of the application. In the log file, Data Manager may show a value for Average. For dimension builds, a single Type 1 change may affect many rows. The Average, is the total number of rows affected by all the Type 1 changes, divided by the number of individual Type 1 changes.
Detail	Shows more detailed progress messages.
Internal	Shows internal Data Manager activity messages, including resource usage. For example, memory usage, dimension sizes, and paging information.
SQL	Shows all the SQL statements used at each stage of execution. This information can help resolve database errors.
ExecutedSQL	Shows the executed SQL for SELECT statements.
User	Shows all the application messages written to the log file. You can include additional application messages using the LogMsg function.
Variable	Shows all the variables used in the build or JobStream.

Procedure

1. Click the required build or JobStream.
2. From the **Edit** menu, click **Properties** .
3. Click the **Logging** tab.
4. In the **Trace** box, select the check boxes for the types of information that you want to include in the log file.

When you execute the build or JobStream, you can override the values that you select here. For more information, see “Executing a Fact Build or Dimension Build on your Local Computer” on page 245 or “Execute a JobStream on your Local Computer” on page 248.

5. Click **OK**.

Specify the Message Frequency in a Build Log File

For builds, you can specify the frequency of input, output, and domain messages.

The type of messages used in the log files differ between fact builds and dimension builds.

Message type	**Description**
Input	Used in fact builds only.
Output	Used in fact builds when writing cached rows, and in dimension builds as an indication of general progress.
Domain	Used in fact builds and dimension builds when caching reference data. They are also used in fact builds with dimension elements that use a reference domain.

Procedure

1. Click the required build.
2. From the **Edit** menu, click **Properties** .
3. Click the **Logging** tab.
4. In the **Message frequency (input)** box, type the number of input rows that IBM Cognos Data Manager is to process between Detail and Internal messages.

   Note: This box is not available for dimension builds.
5. In the **Message frequency (output)** box, type the number of output rows to process between Detail messages.
6. In the **Message frequency (domain)** box, type the number of domain members that Data Manager is to process between Detail messages.
7. Click **OK**.

Include User Log Messages for JobStream Nodes

When executing a master JobStream, you may want to include user messages for a JobStream node being executed within it.

Procedure

1. For the master JobStream and the appropriate JobStream node, specify that User detail messages are to be included in the log files.

   For information, see “Specify Log File Details” on page 256.
2. In the **Execute JobStream** dialog box, type -M in the **Additional options** box.

   For information, see “rundsjob” on page 382.

View the Log Files

You can view log files from in the Audit window, or directly from the Log directory.

If the build or JobStream was remotely executed, the log file is retrieved from the remote computer. To retrieve remote log files, you must be using IBM Cognos Data Manager Network Services on the server and it must be configured correctly. For more information on Data Manager Network Services, see "Execute a Build or JobStream on a Remote Computer" on page 250, and for more information on configuration, see the IBM Cognos Data Manager *Installation and Configuration Guide*.

Tip: To view the log files from the Log directory, from the **Tools** menu, click **Browse Log Files**.

Procedure

1. To view log files from the Audit window, click the required build or JobStream.
2. From the **Actions** menu, click **Audit**.
3. Select the execution for which you want to view the log files, and then click **Audit Log**.
   - Click **Find** to search for specific information in the log file.
   - Click **Save** to save the log information to a text file.

   For information on the Audit window, see "View the Audit History."
4. When you have finished, click **Close**.

Save Audit Details

When you execute a build or JobStream, you can specify that the events performed are recorded in a series of audit tables.

The Audit window shows the full audit history for the selected build or JobStream. From the Audit window, you can choose to view audit trail details, audit messages, and JobStream details.

You can also access the audit tables using reporting tools. For details on the structure of the IBM Cognos Data Manager audit tables, see "Create an IBM Cognos Data Manager Catalog Schema" on page 22.

View the Audit History

The audit history provides information for each occurrence of a build or JobStream execution. The execution process can be performed locally or remotely. If a JobStream calls itself during execution, each execution is listed separately.

You can use a filter to restrict the list of execution processes to those performed

- in the last 24 hours
- in the 7 days
- in the last month
- in the last 3 months
- in the last year

Each filter that you set up is automatically applied to all fact builds, dimension builds, and JobStreams.

For each execution process listed in the audit history, the following information is provided.

Item	Description
Status icon	▶ indicates the node is running ✓ indicates the node reached a successful completion ✗ indicates the node failed due to errors ■ indicates the execution process was successfully manually stopped ! indicates the execution process had failed prior to being manually stopped For more information on stopping a JobStream execution, see "Stop Execution" on page 263.
Start TimeStamp	The start date and time of the execution process.
Elapsed	The elapsed time for the execution process. **Note:** The elapsed time is only available when the execution process is complete.
Run Mode	The run mode of the build and the delivery types: • N indicates normal execution mode • C indicates check only execution mode • O indicates object creation execution mode • F indicates a fact delivery • D indicates a dimension delivery
Full Refresh	A Boolean value to indicate whether all the data in the table was refreshed: • Y indicates a full refresh was performed • N indicates a full refresh was not performed
Rows Inserted	The number of rows that were successfully inserted.
Rows Updated	The number of rows that were successfully updated.
Connection	The name of the database connection to which the data was delivered.
Table Name	The name of the table into which the data was delivered.
Host Name	The computer on which the build or JobStream was executed.

Note: For JobStreams, information is provided for only the status icon, Start TimeStamp, Elapsed, and Host Name.

The audit history retrieves information from the dsb_component_run, dsb_run_context, and dsb_delivery history tables. For more information, see

"Component Run Table (dsb_component_run)" on page 479, "Run Context Table (dsb_run_context)" on page 480, and "Delivery History Table (dsb_delivery_hist)" on page 481.

Procedure

1. Click the required build or JobStream.
2. From the **Actions** menu, click **Audit**.
3. In the **Date Range** box, click the filter you want to use.
4. Click the **Execute** button ▶ to execute the filter.

   Tip: To interrupt a filter from executing, click the **Interrupt** button ■.
5. When you have finished, click **Close**.

Specify the Audit Trail Details to be Recorded

For each fact build, dimension build, or JobStream you can specify that audit trail information is recorded during execution. You do this from the Properties window for the selected build or JobStream.

You can specify that these types of information are recorded:

- Timing records overall timing information. For example, the start, end, and elapsed times for the execution process.

  This information is available for fact builds, dimension builds, and JobStreams.
- Acquire records information about the source data acquired. For example, the number of data rows that each data source contributes.

  This information is available for fact builds only.
- Transform records information about the data processed. For example, the number of data rows received from the data sources.

  This information is available for fact builds only.
- Deliver records information about data delivery. For example, the number of data rows delivered.

  This information is available for fact builds and dimension builds.
- Internal records information about the internal measures of performance. For example, the size of the hash table.

  This information is available for fact builds only.
- Alert records a user-defined audit record of type ALERT, into the IBM Cognos Data Manager audit tables.

  This information is available for fact builds, dimension builds, and JobStreams.
- User records user-controlled audit messages.

  This information is available for fact builds, dimension builds, and JobStreams.

Procedure

1. Click the required build or JobStream.
2. From the **Edit** menu, click **Properties** .
3. Click the **Audit** tab.
4. Select the check boxes for the events that you want to record.
5. Click **OK**.

View the Audit Trail Details

You can view the following audit trail details.

Item	Description
Audit Timestamp	The start time that each row of data was written.
Audit Group	The audit group to which information is written.
Audit Item	The audit item of the group to which information is written.
Audit Message	The message from the audit item.

The audit trail retrieves information from the dsb_audit_trail and dsb_component_run tables. For more information, see “Audit Trail Table (dsb_audit_trail)” on page 481, and “Component Run Table (dsb_component_run)” on page 479.

Procedure

1. Click the required build or JobStream.
2. From the **Actions** menu, click **Audit**.
3. Select the execution process for which you want to view the audit trail, and then click **Audit Trail**.
4. When you have finished, click **Close**.

View the Audit Messages

You can drill down from a build or JobStream execution in the audit history to view any audit messages that were recorded during the execution process.

By default, all the audit messages recorded during the execution process are included. You can use a filter to restrict the number of audit messages shown.

Note: When you set up a filter, it is automatically applied it to all fact builds, dimension builds, and JobStreams.

The following details are provided for each audit message.

Item	Description
Message	A unique number that identifies the message that is recorded.
Severity	The severity of the message: • F indicates a fatal message • E indicates an error • W indicates a warning
Message Text	The message text.

The audit message retrieves information from the dsb_audit_msg_line and dsb_audit_msg tables. For more information, see “Audit Message Line Table (dsb_audit_msg_line)” on page 482, and “Audit Message Table (dsb_audit_msg)” on page 482.

Procedure

1. Click the required build or JobStream.
2. From the **Actions** menu, click **Audit**.
3. Select the execution process for which you want to view audit messages, and then click **Audit Messages**.
4. In the **Number of rows to return** box, type the number of messages you want to view.
5. Click the **Execute** button ▶ to execute the filter.
   - To interrupt a filter from executing, click the **Interrupt** button ■.
   - To clear a filter, delete the value in the **Number of rows to return** box, and then click the **Execute** button ▶.
6. When you have finished, click **Close**.

View Node Details for a JobStream Execution

You can drill down from a JobStream execution in the audit history to view the individual nodes executed. For each node listed, the following details are provided.

Item	Description
Status icon	▶ indicates the node is running ✓ indicates the node reached a successful completion ✗ indicates the node failed due to errors ■ indicates the execution process was successfully manually stopped ! indicates the execution process had failed prior to being manually stopped For more information on stopping a JobStream execution, see “Stop Execution” on page 263.
Node Name	The name and type of the node being executed. **Note:** If the associated build or JobStream name differs from the node name, it is shown in brackets.
Start TimeStamp	The start date and time of the node execution process.
Elapsed	The elapsed time for the node execution process. **Note:** The elapsed time is only available when the execution process is complete.
Host Name	The computer on which the JobStream was executed.

Notes

- Nodes are listed in their order of execution.
- If the JobStream includes a JobStream node, the individual nodes in that JobStream are included.
- If the JobStream includes a node that is repeatedly executed using a loop, the node is listed each time it is executed.

Information about executed nodes is retrieved from the dsb_jobnode_run table. For more information, see “Job Node Run Table (dsb_jobnode_run)” on page 482.

Procedure

1. Click the required JobStream.
2. From the **Actions** menu, click **Audit**.
3. Click the JobStream execution for which you want to view executed nodes, and then click **JobStream Details**.

   Tip: Click **Refresh** to update the list of executed nodes whilst a JobStream is still executing.

   If there are fact build or dimension build nodes listed, you can view further audit information for these.
4. Click the required fact build node or dimension build node and click one of the following audit options:
   - **Audit Trail**
   - **Audit Log**
   - **Audit Messages**

   For more information on the audit information available for a build, see “View the Log Files” on page 258, “View the Audit Trail Details” on page 261, and “View the Audit Messages” on page 261.
5. When you have finished, click **Close**.

Stop Execution

If a build or JobStream is being remotely executed, you can manually stop it if the status is running (indicated by a white triangle in a blue circle). When you stop a build or JobStream, the status is updated in the audit history and the audit tables.

For more information on how the status is updated, see “View the Audit History” on page 258 and “The Audit Tables”

Procedure

1. Click the required build or JobStream.
2. From the **Actions** menu, click **Audit**.
3. Click the component that is currently running, and then click **Stop Execution**.

The Audit Tables

The audit tables are used to provide summary information about the audit history, the audit trail, the audit messages, and the JobStream details.

There are the following audit tables:

- component run (dsb_component_run)
- context run (dsb_run_context)
- delivery history (dsb_delivery_hist)
- audit trail (dsb_audit_trail)

- audit message (dsb_audit_msg)
- audit message line (dsb_audit_msg_line)
- job node run (dsb_jobnode_run)

You can view the audit tables by creating a connection to the catalog and opening the connection in SQLTerm.

Create a Log File for Expression and Script Tracing

When you execute a fact build, dimension build, or JobStream, it is possible to create an additional log file that traces any expressions and scripts, user-defined functions, and IBM Cognos Data Manager predefined functions. This is useful when a lower level of debugging is required for complex transformation logic.

If you are executing a fact build or a dimension build, Data Manager creates one log file for the trace information. If you are executing a JobStream, Data Manager creates a separate log file for each executed node.

By default, trace log files are stored in the directory named Program Files\ibm\cognos\c10\datamanager\log, however you can change this. For more information, see “Change the Default Location for Log Files” on page 255.

The file name of each trace log file is <component_type>_ <build_name>_<number>_trace.log, where <component_type> is either Build, DimBuild, or Job, <component_name> is the name of the build and <number> is a four digit number that identifies the execution. For example, the trace log file for a fact build named NewBuild might be Build_NewBuild_0001_trace.log.

Note: If a build name contains characters other than letters and numbers, these are replaced with an underscore in the trace log file name.

Specify Trace Log File Details

These levels of detail can be included in the trace log file: Basic, Datatype, Builtin, and Variable.

Basic shows the input and output values for items contained in the expression, script, or user-defined function.

Datatype shows the data type for all input and output values displayed.

Builtin shows the IBM Cognos Data Manager predefined functions used in the build or JobStream, the components from which they are called, and the values assigned to those functions.

Variable shows the variables called by an expression, script or function, the components in which the variables are declared, and values assigned to those variables.

Procedure

1. If the build or JobStream contains a lot of source data, you should apply a maximum row limit so that the resulting trace log file is smaller.

   For information, see “Apply a Maximum Row Limit” on page 131.
2. Click the required build or JobStream.
3. From the **Actions** menu, click **Execute**.

If you made changes to the build or JobStream and did not save them, a message appears.

4. Click **OK** to save the changes.
5. In the **Additional options** box, type the following string to invoke trace logging, where <OPTION> is the level of detail you require: BASIC, DATATYPE, BUILTIN, or VARIABLE.

   ```
   -VEXP_TRACE_VALUES=<OPTION,[OPTION]>
   ```

   For example, to include the BASIC and DATATYPE levels, type

   -VEXP_TRACE_VALUES=BASIC,DATATYPE

   - You can type * to include all levels of detail.
   - By default, the command window remains open when execution is complete, and you must press Enter to close it. To close it automatically, clear the **Pause on completion** check box.
6. Click **OK**.

   A command window appears and shows the progress of the execution process.

Example Trace Log Files

The following example illustrates a trace log file that IBM Cognos Data Manager generates if the basic level of detail is selected. The fact build named Example build contains one user-defined function named Margin, and two DataStream derivations, named SalesTotal and GrossProfit.

```
***** START TRACING Basic *****
==> Datastream  SalesTotal
<== Datastream  SalesTotal -> 553.56
==> Datastream  GrossProfit
<== Datastream  GrossProfit -> 185.64
==> Example GrossMargin
 ==> Margin( 6.59, 4.38 )
 <== Margin -> 0.33535660091047
<== Example GrossMargin -> 0.33535660091047
==> Datastream  SalesTotal
<== Datastream  SalesTotal -> 830.34
==> Datastream  GrossProfit
<== Datastream  GrossProfit -> 278.46
==> Example GrossMargin
 ==> Margin( 6.59, 4.38 )
 <== Margin -> 0.33535660091047
<== Example GrossMargin -> 0.33535660091047
***** END TRACE *****
```

In lines 1 and 2, you can see that the initial input value of SalesTotal is Null, and the output value is 553.56. Similarly, the initial input value of GrossProfit is Null, and the output value is 185.64.

In lines 5 to 8, you can see that the GrossMargin expression calls the Margin user-defined function. The input value and output values of Margin are shown, and the resulting output of GrossMargin is shown.

In the next example, the fact build named Example is executed again, but with the basic and datatype levels selected.

```
***** START TRACING Basic, Datatype *****
==> Datastream  SalesTotal
<== Datastream  SalesTotal -> FLOAT 553.56
==> Datastream  GrossProfit
<== Datastream  GrossProfit -> FLOAT 185.64
==> Example GrossMargin
 ==> Margin( Price INTEGER, FLOAT or NUMBER 6.59,
Cost INTEGER, FLOAT or NUMBER 4.38 )
 <== Margin -> FLOAT 0.33535660091047
<== Example GrossMargin -> FLOAT 0.33535660091047
==> Datastream  SalesTotal
<== Datastream  SalesTotal -> FLOAT 830.34
==> Datastream  GrossProfit
<== Datastream  GrossProfit -> FLOAT 278.46
==> Example GrossMargin
 ==> Margin( Price INTEGER, FLOAT or NUMBER 6.59,
Cost INTEGER, FLOAT or NUMBER 4.38 )
 <== Margin -> FLOAT 0.33535660091047
<== Example GrossMargin -> FLOAT 0.33535660091047
***** END TRACE *****
```

In addition to the basic details, the data type for each derivation and user-defined function is shown.

In the following extract, a build containing a Data Manager predefined function is executed, and the basic and builtin levels are selected.

```
==> Datasource  DataSource1
 <==> Concat( 'TrailChef Water Bag', ' :', 'Cost :',
4 ) -> 'TrailChef Water Bag :Cost :4'
<== Datasource  DataSource1 -> 'TrailChef Water Bag :Cost :4'
```

You can see that DataSource1 calls the Concat function. The output values of Concat are shown, and the resulting output of DataSource1 is also shown.

The final example shows the trace log file extract for an executed build, where the basic and variable levels are selected.

```
==> Datastream  SalesTotal
<== Datastream  SalesTotal -> 553.56
==> Datastream  GrossProfit
<== Datastream  GrossProfit -> 185.64
==> Datastream  Derivation1
 TestVariable [Additional] = '2005-08-16 00:00:00'
<== Datastream  TestDerivation -> '2005-08-16 00:00:00'
```

You can see that the variable named TestVariable is declared in the fact build named Additional, and assigned a value of 2005-08-16 00:00:00. It is then called in the DataStream derivation named TestDerivation.

Chapter 22. Exporting Metadata

You can create descriptions of the conformed models in a data mart or data warehouse. When you have created the conformed model descriptions, you export the metadata to an XML file. This file can then be used to produce a model of your target data mart or data warehouse in IBM Cognos Framework Manager, without the need to recreate the complete model.

A conformed model contains dimensions that are shared by more than one fact table. Fact tables are grouped together using these shared dimensions to form stars. Similarly, stars are grouped together into subject areas to form collections.

An example of a conformed model is a customer dimension used by a fact table within a business subject area named Sales Analysis Reporting.

When you have created export models, they can be used by Framework Manager for deployment to IBM Cognos Business Intelligence.

To set up metadata you must

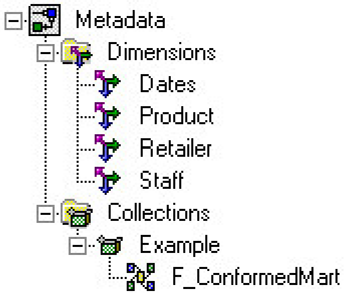
- create the metadata dimensions
- create the metadata collections
- add star models to a metadata collection

View Metadata

You can view metadata in the Tree pane or the Visualization pane.

View Metadata in the Tree Pane

Metadata consists of a number of objects grouped together in the Metadata folder. Metadata is represented as follows.

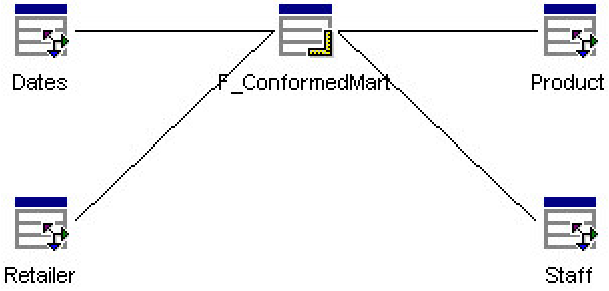


Tip: You can view just the metadata contained in a catalog by clicking the Metadata tab at the bottom of the Tree pane.

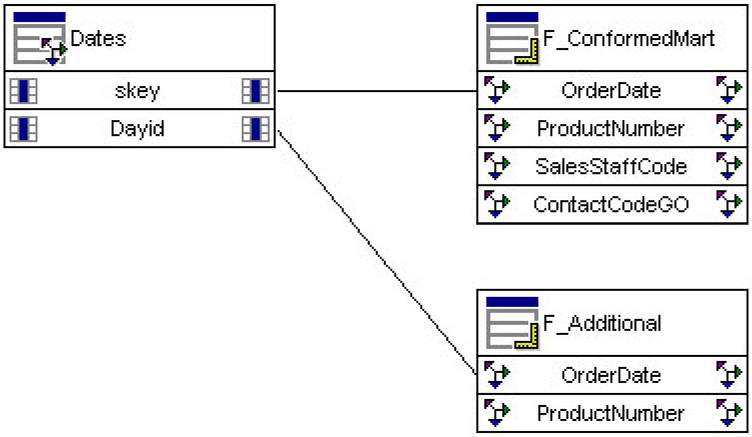
View Metadata in the Visualization Pane

When you click on a metadata object in the Tree pane, a visual representation appears in the Visualization pane.

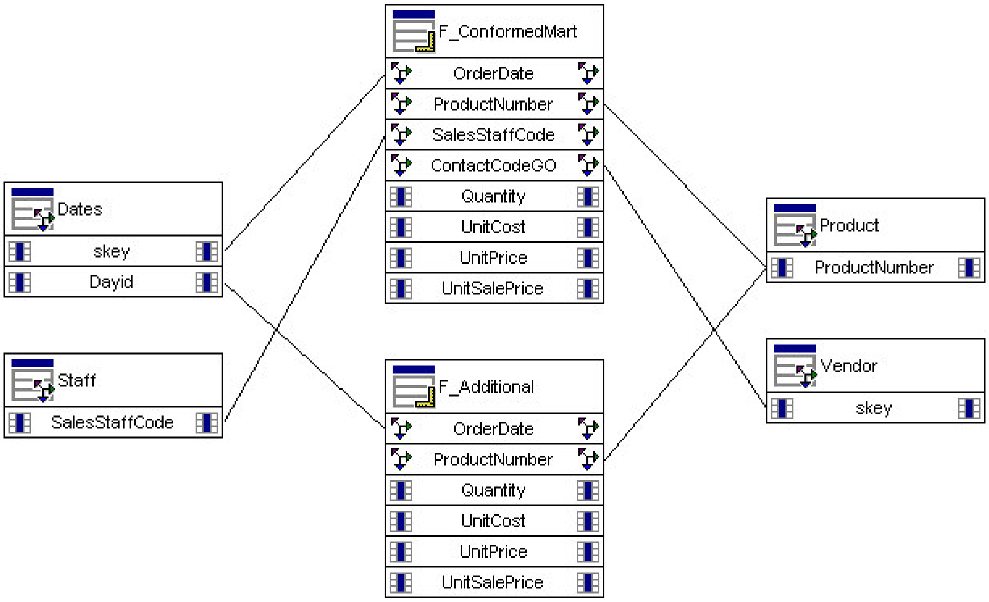
If you click the **Metadata** folder, a visual representation of the whole target conformed model appears.



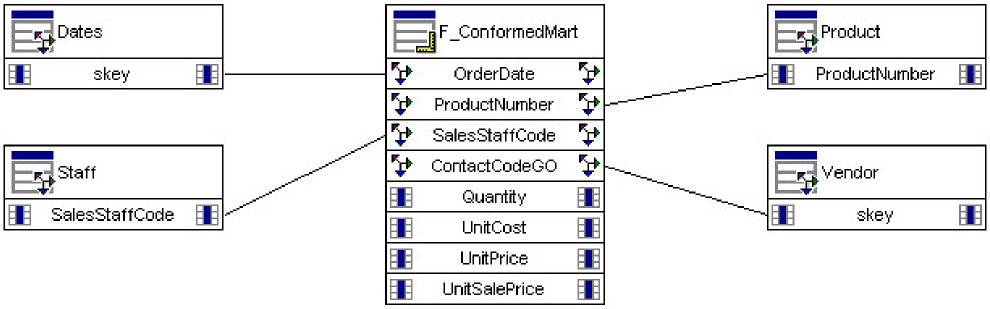
If you click a specific metadata dimension, a visual representation of the dimension, together with the fact tables referenced to it, appears.



If you click a specific metadata collection, a visual representation of the star models contained in the collection appears.



If you click a specific metadata star, a visual representation of the star, together with the metadata dimensions it references, appears.



Create a Metadata Dimension

A metadata dimension contains the description of the conformed dimension that you want to include in a metadata export.

You can create a metadata dimension from the Metadata folder or from a dimension build.

You must create a separate metadata dimension for each conformed dimension that you want to export.

The metadata dimension is a representation of the dimension build delivery table. It does not matter if the dimension is a star (single table delivery for all levels) or some form of snowflake (multiple table delivery), it is still logically represented in the model as a single metadata dimension. The exported metadata will include an object for each of the levels in the snowflake.

Note: It is not possible to directly model parent-child relationships using IBM Cognos Framework Manager. If your catalog contains a conformed dimension that references an auto-level hierarchy, you must create a lookup instead of the auto-level hierarchy, and reference the lookup from the conformed dimension. You can then correctly include the description of the conformed dimension in the metadata export.

Create a metadata dimension from the Metadata folder

You can create a metadata dimension from the Metadata folder or from a dimension build.

Procedure

1. In the **Metadata** folder, click **Dimensions** .
2. From the **Insert** menu, click **Metadata**, and then click **Dimension**.

   The **Metadata Dimension Properties** window appears.
3. Type a name for the metadata dimension and, if required, a business name and description.
4. Click the **Details** tab.
5. In the **Select Dimension Table(s)** box, expand the appropriate folder if required, and then expand the appropriate dimension build to see the dimension tables it contains.
6. If you want to include all the columns in a dimension table by default, select the dimension tables used to deliver the conformed dimension.

   If you want to include only specific columns from a dimension table, expand the appropriate dimension tables, and then select the required columns.
7. Click **OK**.

Create a metadata dimension from a dimension build

You can create a metadata dimension from the Metadata folder or from a dimension build.

When you create a metadata dimension from a dimension build, you are indicating that the dimension tables form part of the conformed mart, and that they are available to be referenced by fact tables. Any dimension tables that you add become available for export. You should identify which dimensions you would like to include in your model before creating fact table stars. This makes it easier to automatically associate fact tables with existing metadata dimension tables.

Procedure

1. Click the dimension build for which you want to add metadata.
2. From the **Actions** menu, click **Add to Metadata**.

   The **Add Metadata Dimension** window appears.

   By default, IBM Cognos Data Manager selects all the dimension tables used by the dimension build. All the columns contained in each dimension table are also selected by default.
3. If you want to exclude specific columns from a dimension table, expand the appropriate dimension table, and then clear the required columns.
4. Click **Next**.
5. Type a name for the metadata dimension and, if required, a business name and description.
6. Click **Finish**.

   Data Manager adds the metadata dimension to the Metadata folder.

Create a Metadata Collection

A metadata collection groups together all the metadata star models that you want to include in a metadata export. A metadata collection can contain an entire data warehouse model, or a subset of a model, for example a single subject area.

Although it is not necessary to create a separate collection for each subject area, it is advisable to do so. Collections facilitate grouping together stars that have a common meaning. For example, a Sales Analysis Reporting subject area might contain one star for sales orders, and another star for sales shipments. By grouping these stars together in a collection, it is easier to view the content of a large metadata export, together with its associated references.

Note: Collections are only used to group together stars within IBM Cognos Data Manager Designer. They are not included in a metadata export file.

Procedure

1. In the **Metadata** folder, click **Collections** .
2. From the **Insert** menu, click **Metadata**, and then click **Collection**.

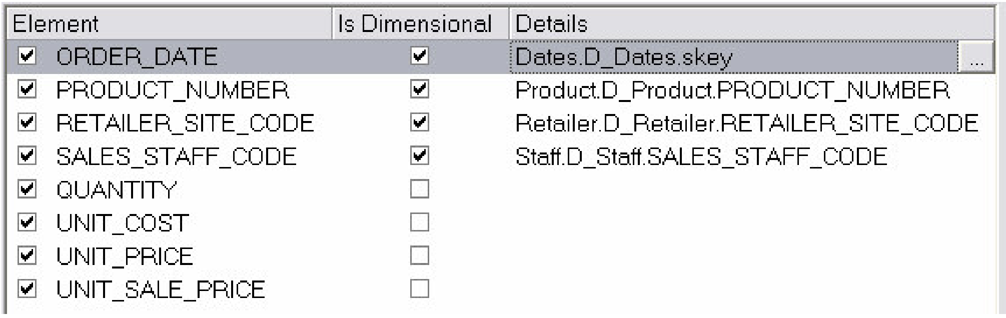
   The **Metadata Collection Properties** window appears.
3. Click **OK**.

   You can now add one or more metadata star models to the collection. For more information, see "Add a Star Model to Metadata Collection."

Add a Star Model to Metadata Collection

A metadata star model contains the description of the fact table that you want to include in the metadata export, with reference to the metadata dimensions you have set up. You must set up a separate star model for each fact table sharing a conformed dimension.

When you are setting up a metadata star model, you must include the details of the star elements.



Element	Is Dimensional	Details
☑ ORDER_DATE	☑	Dates.D_Dates.skey ...
☑ PRODUCT_NUMBER	☑	Product.D_Product.PRODUCT_NUMBER
☑ RETAILER_SITE_CODE	☑	Retailer.D_Retailer.RETAILER_SITE_CODE
☑ SALES_STAFF_CODE	☑	Staff.D_Staff.SALES_STAFF_CODE
☑ QUANTITY	□	
☑ UNIT_COST	□	
☑ UNIT_PRICE	□	
☑ UNIT_SALE_PRICE	□	

You use the following columns to specify the star details:

- **Element**

  IBM Cognos Data Manager automatically includes a star element for each corresponding transformation model element in the selected fact table.
- **Is Dimensional**

  By default, any transformation model elements set up as dimension elements in the fact table, are marked as dimensional columns in a metadata star description. This indicates that, in the model, these items should be considered as foreign key columns to other tables.

- **Details**

  Data Manager automatically links any dimensional column to its associated column in a dimension table if

  – the dimension table is already set up as a metadata dimension in the Metadata folder

  – the element in the fact build references a hierarchy or a lookup that uses a template for data access

Note: By setting up shared dimensions as metadata dimensions in the Metadata folder before creating stars, Data Manager can automatically add the links for you. For more information, see “Create a metadata dimension from a dimension build” on page 270.

If the fact table that you want to include in the metadata export has been partitioned, the columns in the table with the primary partition are automatically selected. However, you can also choose to include those columns that are in tables with secondary partitioning. For more information, see “Partition Columns Horizontally Across a Table” on page 193.

You can create a star model from the Metadata folder or from a fact build.

Set up a star model from the Metadata folder

You can create a star model from the Metadata folder or from a fact build.

Procedure

1. Click the metadata collection to which you want to add a star model.
2. From the **Insert** menu, click **Metadata**, and then click **Star**.

   The **Metadata Star Properties** window appears.
3. Type a name for the metadata star, and if required, a business name and description.
4. Click the **Details** tab.
5. In the **Fact table** box, click the **Browse** button ... .
6. Expand the appropriate fact build, select the required fact table, and then click **OK**.
   - Only fact table deliveries are shown, as you cannot use text deliveries to create a star model.
   - If the table has partitioned deliveries, all the elements for all the partitioned tables are shown.

   The **Star Elements** box lists all the transformation model elements from the fact table that will be included in the star model.

   Tip: To exclude an element from the model, clear the check box to the left of the element name.

   For each dimension element listed, a check mark appears in the **Is Dimensional** column.
7. If required, you can manually link elements marked as dimensional columns.

   Select the relevant dimension element, and click the **Browse** button ... that appears in the **Details** column.

   A list of all metadata dimensions is shown in the **Select the Dimension Column** dialog box.
8. Expand the appropriate dimension table, select the required dimensional column, and then click **OK**.

Note: Only those columns that are included in the metadata dimension table are shown.

9. Click **OK**.

Set up a star model from a fact build

You can create a star model from the Metadata folder or from a fact build.

Procedure

1. Click the fact build for which you want to add metadata.
2. From the **Actions** menu, click **Add to Metadata**.

   The **Add Metadata Star** window appears.
3. Expand the build, and click the required fact table.

   Note: Only fact table deliveries are shown, as you cannot use text deliveries to create a star model.
4. In the **Metadata Collection** box, click the collection to which you want to add the star model.

   Tip: Click **New** to create a collection.
5. Click **Next**.

   All the transformation model elements from the fact table that will be included in the star model are listed. If the table has partitioned deliveries, all the elements for all the partitioned tables are shown.

   Tip: To exclude an element from the model, clear the check box to the left of the element name.

   For each dimension element listed, a check mark appears in the **Is Dimensional** column.

   If required, you can manually link elements marked as dimensional columns.
6. Select the relevant dimension element, and click the **Browse** button ... in the **Details** column.

   A list of all metadata dimensions is shown in the **Select the Dimension Column** dialog box.

   Note: Only those columns that are included in the metadata dimension table are shown.
7. Expand the appropriate dimension table, select the required dimensional column, and then click **OK**.
8. Click **Next**.
9. Type a name for the metadata star and, if required, a business name and description.
10. Click **Finish**.

    Data Manager adds the metadata star to the Metadata folder.

Export Metadata

When you have set up the metadata, you can export the entire conformed model, or a subset of it, to an XML model.

Procedure

1. Click the **Metadata** folder in the **Tree** pane.
2. From the **Actions** menu, click **Export Metadata**.

The **Export Metadata** window appears, listing all the metadata collections in the conformed model. By default, all collections are selected for export, and unreferenced dimensions are included in the export.

3. To exclude a metadata collection from the export, in the **Metadata Collection** box, clear the appropriate check box.
4. Type a name for the metadata export and, if required, add a description.
5. If there are metadata dimensions included in the **Metadata** folder that are not linked to any stars (they are unreferenced) you can choose whether or not to include them in the export file. By default, unreferenced dimensions are included.

   You should include unreferenced dimensions if you want to export definitions for dimensions that may link to tables that already exist in the data mart but are not managed by IBM Cognos Data Manager. In this case, you can use IBM Cognos Framework Manager to link these tables.

   To exclude unreferenced dimensions from the export, clear the **Include referenced dimensions** box.
6. Click **OK**.

   The **Export Metadata** dialog box appears.
7. In the **File name** box, type a name for the metadata export file.
8. In the **Save as type** box, click Metadata Export Files (*.xml).
9. Click **Save**.

   You can now use Framework Manager to import the conformed model.

   The following table defines how metadata model attributes that are published to Framework Manager are derived from Data Manager catalog components. You can use the following tables to determine which attributes must be maintained for them to be automatically included as part of the target XML based model.

Object	**Item**	**Comes from...**
Model	Model name	The name attribute provided when the model is exported. It is not persisted in the source catalog.
Connections	Connection name	The alias name of the connection.

Object	**Item**	**Comes from...**
Physical metadata	Fact table name	The name of the fact table delivery. When a table is part of a partition, the logical table name is taken from the name of the primary partition.
Physical metadata	Dimension table name	The name of the table or tables defined in the dimension table delivery in the dimension build.

Object	Item	Comes from...
Physical metadata	Fact table column name	The column name attribute from the fact table delivery.
Physical metadata	Dimension table column name	The delivery template attribute name from the dimension build that delivers the dimension.
Physical metadata	Dimension column usage	The behavior attribute from the delivery template.
Physical metadata	Fact column usage	The transformation model element type.
Physical metadata	Fact/dimension column joins	The metadata star definition which is derived automatically from the fact build references to the reference dimensions. Can be overridden in the metadata star definition.
Physical metadata	Column data types	Derived automatically by Data Manager based on the target database mapping rules.

Object	Item	Comes from...
Business metadata	Fact table business name	The business name attribute of the metadata star.
Business metadata	Fact table description	The description name attribute of the metadata star.
Business metadata	Dimension table business name	The business name attribute of the metadata dimension.
Business metadata	Dimension table description	The description attribute of the metadata dimension.
Business metadata	Fact business name	The business name attribute of the transformation model element.
Business metadata	Fact description	The description attribute of the transformation model element.

Object	Item	Comes from...
Business metadata	Fact aggregation	For measure transformation model elements only. The aggregation rule of the transformation model element.
Business metadata	Dimension attribute business name	The business name attribute of the delivery template attribute.
Business metadata	Dimension attribute description	The description attribute of the delivery template attribute.
Business metadata	Hierarchy name	The business name attribute of the source hierarchy.
Business metadata	Hierarchy description	The description attribute of the source hierarchy.
Business metadata	Hierarchy level name	The business name attribute of the source hierarchy level.
Business metadata	Hierarchy level description	The description attribute of the source hierarchy level.

Chapter 23. User-Defined Functions

You can use functions to

- calculate derivations and derived dimensions
- define DataStream filters
- define output filters for fact deliveries
- define JobStream procedure nodes and condition nodes
- initialize build or JobStream variables

You can add to the extensive library of predefined IBM Cognos Data Manager functions by creating user-defined functions that you can use as often as you require. You do not have to copy user-defined functions between Data Manager actions, or retype them, you can simply call functions by selecting them from the pop-up menu in the relevant Data Manager window.

Note: You can also call a user-defined function from within itself.

There are two types of user-defined functions

- internal functions that are defined within Data Manager
- external functions that are defined within run-time libraries

User-defined functions for the open catalog are shown in the Functions folder in the Library tree.

Note: You can find samples of both internal and external user-defined functions in the sample catalogs described in IBM Cognos Data Manager *Getting Started*.

Internal User-Defined Functions

Internal user-defined functions are defined within IBM Cognos Data Manager. They are nestable, reusable functions. You can create them by combining existing Data Manager functions, creating functions, or a combination of both these methods.

Create an Internal User-Defined Function

You create an internal user-defined function within IBM Cognos Data Manager.

Procedure

1. In the **Library** folder, click **Functions** .
2. From the **Insert** menu, click **Library**, and then click **Function**.
3. Click the **General** tab.
4. In the **Name** box, type a unique name for the function.

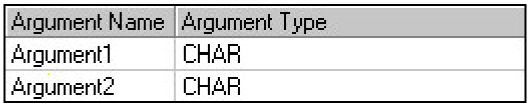
   Tip: To ensure that this name is different from the built-in Data Manager functions, it is recommended that you precede the name of each user-defined function with the letters udf, for example, udf.newfunction
5. If required, type a business name and description.
6. In the **Return type** box, click the data type to be returned by the function.

   For a description of each data type, see Appendix E, “Data Types,” on page 491.

7. Click the **Interface** tab.
8. For each argument that you want to add, click **Add**.

   Note: Each user-defined function can contain a maximum of 16 arguments. If you want to use more than 16 arguments, or define a user-defined function that has a variable number of arguments, then you can use an argument of type array.

   By default, Data Manager names each argument, Argument*n* (where *n* is a unique sequential number), and allocates an argument type of CHAR.



Argument Name	Argument Type
Argument1	CHAR
Argument2	CHAR

9. Change each argument name to something more meaningful, because this is the name that you will use when you implement the function.
10. If a function has more than one argument, the order of the arguments on this tab indicates the order in which you must enter the values when you use the function, you can change this order if you require by clicking an argument and then clicking **Move Up** or **Move Down**.
11. For each argument, click within the **Argument type** column, then click on the arrow that appears to show the available data types. Click the required data type.

    For information about argument types, see Appendix E, “Data Types,” on page 491.

    Tip: To delete an argument, click the argument and then click **Delete**.
12. If required, create the variables required for use within the function.

    For more information, see Chapter 24, “Variables,” on page 289.
13. Click the **Implementation** tab.
14. Click **Internal**.
15. In the right pane, enter the syntax for the function.

    You can type the syntax, and you can use the tree structure in the left pane to select the items.

    You can select an item from a folder by dragging it to the right pane, or by double-clicking it.

    For information on operators, controls, and functions, see the IBM Cognos *Function and Scripting Reference Guide*.
16. If required, test the function.

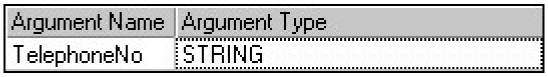
    For information, see Chapter 25, “Testing Expressions and Scripts,” on page 309.
17. Click **OK**.

Example

This example describes how to create an internal user-defined function that determines whether telephone numbers are European toll-free numbers.

Procedure

1. In the **Library** folder, click **Functions** .
2. From the **Insert** menu, click **Library**, and then click **Function**.
3. Click the **General** tab. In the **Name** box, type **FreephoneNumbers**
4. In the **Return type** box, click **Boolean**.
5. Click the **Interface** tab. Click **Add** to add an argument, and name it **TelephoneNo**. Click **String** as the argument type.



Argument Name	Argument Type
TelephoneNo	STRING

6. Click the **Implementation** tab. Select **Internal** and, in the **Calculation** box, type this syntax.

   Left(TelephoneNo,4)='0800'

   This function determines whether the first four characters of the telephone number are 0800, which designates a toll-free number.
7. Click **Test**.

   The **Values for Expression** window appears.
8. In the **Input values** box, click the **Type** column and click **Char**, and then click the **Value** column and type a number that starts with 0800; for example, 0800101010
9. Click **Calculate**.

   If the function works correctly, TRUE appears in the **Result** box.

   For more information about testing functions, see Chapter 25, “Testing Expressions and Scripts,” on page 309.

External User-Defined Functions

External user-defined functions are defined within run-time libraries, each of which must adhere to special rules and conventions. When you have created an external user-defined function, you must register it within IBM Cognos Data Manager before you can use it.

External user-defined functions must adhere to these rules and conventions:

- Libraries of external user-defined functions must use C calling conventions. On the Microsoft Windows operating system, you must use dynamic link libraries (DLLs); and on the UNIX operating system, shared libraries.
- You must store the results of execution in a formal parameter. This is because Data Manager ignores any value that the external user-defined functions return.
- All formal parameters except the second must be of type DS_VALUE_T, a structured data type that supports values of several standard C data types. The first parameter contains the results of executing the function. All other parameters except the second supply input values to the function.

  The second formal parameter must be of type DS_BOOLEAN. It specifies in which mode to execute the function.

  For information, see “Special Data Types for External User-Defined Functions” on page 280.
- Data Manager can execute functions in either validation or normal execution mode, and any functions that you write must support this.

For information, see “Execution Modes for External User-Defined Functions” on page 285.

- An external user-defined function must be registered within Data Manager before you can use it.

  For information, see “Register External User-Defined Functions” on page 286.

To help you create external user-defined functions, Data Manager provides the header file named dsudftypes.h. This file defines the special data types and macros that you need, and is located in the directory named <install directory>\webcontent\samples\DataManager\udfs.

Declare an External User-Defined Function

The simplest way to declare an external user-defined function is

```
void <funcname>(
 DS_VALUE_T *<result>,
 DS_BOOLEAN <mode_flag>{,
 DS_VALUE_T *<input_parameter>} )
```

However, the compiler that you use and the operating system on which you want to use the function dictate how you must write function declarations. For example, using Microsoft C++ version 6, you may use

```
void__declspec (dllexport) __cdecl <funcname>(
 DS_VALUE_T *<result>,
 DS_BOOLEAN <mode_flag>{,
 DS_VALUE_T *<input_parameter>} )
```

Note: For convenience, DS_UDF_CALL has been defined to _declspec(dllexport) _cdecl on Microsoft Windows operating system, so functions can be defined as follows:

```
void DS_UDF_CALL <funcname>
```

The parts of the declaration have these meanings

Symbol	Meaning
<funcname>	The name of the function.
<result>	The name of the parameter that stores the result of executing the function.
<mode_flag>	The name of the parameter that specifies the mode in which to execute the function.
<input_parameter>	The name of a parameter that provides an input value to the function.

Note: All parameters except the second are passed by reference.

Special Data Types for External User-Defined Functions

The header file named dsudftypes.h defines the following:

DS_BOOLEAN

A Boolean data type that can take either the value TRUE or the value FALSE. This data type is used for the <mode_flag>.

DS_VALUE_T

DS_VALUE_T is a structured data type that provides support for several standard C data types. This data type is used for results and parameters.

```
typedef struct
 {
 DS_DATATYPE_T t;
 DS_DATA_T v;
 } DS_VALUE_T
```

Part	Description
t	Identifies the IBM Cognos Data Manager data type.
v	Stores the value.

DS_DATATYPE_T

```
typedef struct
 {
 DS_DATATYPE_E eDataType;
 size_t Precision;
 int Scale;
 } DS_DATATYPE_T;
```

Part	Description
eDataType	An enumeration of the data type.
Precision	For numeric values, this part identifies the maximum number of digits that the value can hold. For string values, this part identifies the maximum number of characters that the string can hold.
Scale	For numeric values, this part identifies the number of digits to the right of the decimal point. For string values, no meaning.

DS_DATATYPE_E

The value of eDataType denotes the data type that a variable holds. The following are the values that variables of type DS_DATATYPE_E can take, together with the IBM Cognos Data Manager data type that each value denotes.

For more information, see DS_DATA_U.

Part	Description
DS_UNKNOWN_VTYPE	The data type is undefined.
DS_CHAR_VTYPE	The value is a string of UTF-8 characters.
DS_INTEGER_VTYPE	The value is a 64-bit integer.
DS_DOUBLE_VTYPE	The value is a double precision, floating point number.
DS_DATE_VTYPE	The value is a string representation of a date.
DS_BOOLEAN_VTYPE	The variable holds a Boolean value.
DS_BINARY_VTYPE	The variable holds binary data.
DS_ARRAY_VTYPE	The value contains a pointer to an array of values of type DS_VALUE_T. For more information, see "Arrays" on page 284.

DS_DATA_T

```
typedef struct
 {
 DS_DATA_U u;
 DS_BOOLEAN bNull;
 DS_BOOLEAN bAlloc;
 } DS_DATA_T;
```

Part	Description
u	A union of C and IBM Cognos Data Manager data types.
bNull	A TRUE value indicates that the variable holds a null value, overriding any value that the union may hold.
bAlloc	Set this to TRUE to indicate that you have dynamically allocated memory to store the value, and that Data Manager should recover this memory during cleanup. For more information, see "Allocating and Freeing Memory" on page 284.

DS_DATA_U

```
typedef union
 {
 char *s;
 DS_INTEGER i;
 double d;
 DS_BOOLEAN b;
 char dt[80];
 DS_BINARY_T bi;
 DS_VALUE_T *p;
 } DS_DATA_U;
```

DS_BINARY_T

```
typedef struct
 {
 size_t nBytes;
 DS_UBYTE *p;
 } DS_BINARY_T;
```

Part	Description
nBytes	The number of bytes of data
p	A pointer to the data

Represent Date Values

You must represent date values as null-terminated strings in this format:

syyyy-mm-dd hh:mi:ss

Part	Meaning
s	The sign of the year value (+ or -) (optional)
yyyy	The year as a four digit number
mm	The month as a two digit number
dd	The day of the month as a two digit number
hh	The hour in two digit, twenty-four hour format
mi	The minutes as a two digit number
ss	The seconds as a two digit number

Note: The time component, that is hh:mi:ss, is optional.

Data Type Mapping and Conversions

IBM Cognos Data Manager provides type conversion functions that can assist when creating and using user-defined functions, for example, ToDouble, ToInteger, and ToChar. In addition, Data Manager provides the TypeInfo function that is useful because NUMBER (10,2) does not necessarily mean that Data Manager will use a precise number to hold the values. Where possible, Data Manager substitutes a type such as Integer or Float. The TypeInfo function tells you the type actually in use.

This table describes the mapping between the parts of this union and the Data Manager native data types.

Data Manager data type	Converted to eDataType	DS_DATA_U
CHAR	DS_CHAR_VTYPE	u.s
INTEGER	DS_INTEGER_VTYPE	u.i
FLOAT	DS_DOUBLE_VTYPE	u.d
NUMBER	DS_INTEGER_VTYPE or DS_DOUBLE_VTYPE	INTEGER if precision is <19 and scale=0, otherwise, other precision/scale combination
DATE	DS_DATE_VTYPE	u.dt
BOOLEAN	DS_BOOLEAN_VTYPE	u.b
BINARY	DS_BINARY_VTYPE	u.bi
ARRAY	DS_ARRAY_VTYPE	u.p
Others	UNKNOWN	bNull=TRUE

Arrays

IBM Cognos Data Manager handles arrays differently to other data types. The argument passed to the function is of type DS_VALUE_T with t.eDataType set to DS_ARRAY_VTYPE. The part v.u.p is a pointer to an array of values of type DS_VALUE_T, the first of which (with index value zero) denotes the number of items in the array. Subsequent elements contain the items that the array stores.

Your function should access the array by explicitly casting it to type DS_VALUE_T. For example

```
DS_VALUE_T *pArgument;  /* This is
the passed argument        */
DS_VALUE_T *pArray;     /* This will
point to the actual array*/
DS_VALUE_T *pElement;   /* This will
point to each element    */
if (pArray = (DS_VALUE_T *)pArgument->v.u.p)
  {
  for (i = 1; i<=pArray->v.u.i; i++)
    {
    pElement = &(pArray[i]);
    /* etc. */
    }
  }
```

Allocating and Freeing Memory

For certain data types it may be necessary for an external user-defined function to allocate memory for the result.

Variable	Union
DS_CHAR_VTYPE	v:u.s
DS_BINARY_VTYPE	v.u.bi
DS_ARRAY_VTYPE	v.u.p and individual array items

To ensure that IBM Cognos Data Manager can successfully free the memory, the function udffree should be present. For example

```
void DS_UDF_CALL udffree (void *p)
{if (p) free (p);}
```

Execution Modes for External User-Defined Functions

IBM Cognos Data Manager can call external user-defined functions in one of two execution modes. These are

- validation execution mode, in which Data Manager calls the function at the start of build execution to check data types and to gauge memory requirements
- normal execution mode, in which Data Manager calls the function to perform the required calculation

Data Manager calls functions in validation execution mode at the start of build execution and in normal execution mode thereafter.

The second actual parameter specifies which execution mode is intended. If this parameter is FALSE, then the function must execute in validation execution mode; otherwise, it must execute in normal execution mode.

Validation Execution Mode (<mode_flag> = FALSE)

IBM Cognos Data Manager uses Validation mode to perform data type checking and to gauge memory requirements. When Data Manager calls a function in validation mode, for each input parameter of type DS_CHAR_VTYPE or DS_BINARY_VTYPE, t.Precision contains the maximum length of the data, or zero if this length is unknown. The function must, within the first parameter, return at least the following results:

- The t.eDataType member must indicate the expected type of the returned data.
- For variable length data only (for DS_CHAR_VTYPE and DS_BINARY_VTYPE), the function should set the t.precision member to the length of the data that the function would return if it actually performed the calculation. Where it is impossible to determine this value, the function should set t.Precision to zero.

Normal Execution Mode (<mode_flag> = TRUE)

IBM Cognos Data Manager calls functions in normal execution mode to perform the calculation that the function provides. It expects the function to exit with the first parameter set as follows:

- The eValueType component indicates the type of data that the union, u, holds.
- If the result is null, the v.bNull member must be TRUE; otherwise, this member must be FALSE.
- If the function dynamically allocated memory, the v.bAlloc member must be TRUE; otherwise, this member must be FALSE. For more information, see "Allocating and Freeing Memory" on page 284.

- If the function returns a variable length value (t.eValueType is DS_CHAR_VTYPE or DS_BINARY_VTYPE), the function must set t.Precision to the string length (excluding any null terminator).

Macros in External User-Defined Functions

The header file named dsudftypes.h defines these macros for use with the DS_VALUE_T data type.

Macro	Description
RETURN_NULL(Result)	Sets Result to null and returns.
RETURN_NULL_WITH_TYPE (Result, Type)	Sets the type of Result to Type, sets the result value to null and returns.
RETURN_NULL_ON_NULL (Param, Result)	Sets the result value to null if the indicated parameter (Param) is null.

Example External User-Defined Functions

To see example external user-defined functions, see the file named sampleUDF.c which is located in the directory named <install directory>\webcontent\samples\DataManager\udfs.

Register External User-Defined Functions

When you create external user-defined functions, you must register them before using them within IBM Cognos Data Manager.

Procedure

1. In the **Library** folder, click **Functions** .
2. From the **Insert** menu, click **Library**, and then click **Function**.
3. Click the **General** tab.
4. In the **Name** box, type a unique name for the function.

   Tip: To ensure that this name is different from the built-in Data Manager functions, it is recommended that you precede the name of each user-defined function with the letters udf, for example, udf.newfunction
5. If you require, type a business name and description.
6. In the **Return type** box, click the data type to be returned by the function. It is this data type that is returned in the pResult parameter.
7. Click the **Interface** tab.
8. For each argument that you want to add, click **Add**.

   Note: Each user-defined function can contain a maximum of 16 arguments.

   By default, Data Manager names each argument, Argument*n* (where *n* is a unique sequential number), and allocates an argument type of char.
9. Change each argument name to something more meaningful, because this is the name that you will use when you implement the function.

   The type and order of the arguments must match the type and order of the actual arguments. The names are unimportant.

Tip: To change the order in which you are prompted for argument values, click **Move Up** or **Move Down**.

10. For each argument, click within the **Argument type** column, then click on the arrow that appears to show the available data types. Click the required data type.

    For information about supported argument types, see Appendix E, “Data Types,” on page 491.
11. Click the **Implementation** tab.
12. Click **External**.
13. In the **Library name** box, type the name of the library file which contains the function you are registering. On the Microsoft Windows operating system this would be a .dll file, on the UNIX operating system, an .so or .sl file.

    You do not have to type the full directory path for the file, the standard rules for locating dynamic libraries for your platform apply.
14. In the **Function name** box, type the actual name of the function. This may be the decorated name.

    For more information, see “Cross-Platform Portability.”
15. Click **OK**.

Cross-Platform Portability

Some compilers decorate function names (for example, with prefixes or suffixes) on export. This can lead to portability issues between compilers and between operating systems.

Depending on the compiler, you may be able to work around these issues by disabling name decoration or by creating aliases between the external and internal names. For example, Microsoft C++ version 6 supports module definition files (*.def) that can include such aliases.

For more information about decorations or aliases, see the documentation for your compiler.

Maintain User-Defined Functions

You can move a user-defined function between catalogs by creating a package and importing it into the target catalog.

For information, see “Component Packages” on page 321.

Use a User-Defined Function

After you create a user-defined function and, in the case of an external user-defined function, register it, you can use it within derivations, derived dimensions, output filters, DataStream filters, and JobStream procedure and condition nodes.

Procedure

1. Open the relevant **Properties** window.
2. For a derivation or derived dimension click the **Calculation** tab, for an output filter click the **Output Filter** tab, for a DataStream filter click the **Filter** tab, and for a JobStream procedure or condition node click the **Action** tab.

3. In the left pane, expand **Functions**, then **User Defined,** and then double-click the user-defined function to use.

   For more information, see "Add a Calculation to a Derivation or Derived Dimension" on page 153, "Use an Output Filter to Deliver Specific Data Rows" on page 192, "Filter Source Data" on page 132, or "Add a Procedure or Condition Node to a JobStream" on page 233.

Delete a User-Defined Function

You can delete a user-defined function that is no longer required.

Procedure

1. In the **Library** folder, click the required function.
2. From the **Edit** menu, click **Delete**.
3. In the message box, click **Yes**.

Establish the Dependencies of a User-Defined Function

If a fact build contains a derivation or derived dimension that references a user-defined function, the fact build is dependent on that function. This dependency extends to the following objects in the catalog:

- packages

  If you add a fact build to a package, referenced functions must also be included.
- source code control

  If you check out a fact build, referenced functions must also be checked out.
- object deletion

  You cannot delete a function that is referenced by a fact build.
- the build visualization

To establish dependencies between the components and a function, you must be able to parse the scripts that reference the function. If the script is successful, dependency analysis is established.

If parsing a script fails, dependency analysis is not established. For example, if a function requires one parameter, but appears in the script with zero or two parameters, the parse fails and dependency analysis is not established.

To ensure that the dependency information is reported correctly, and that dependency analysis is established, test each script that you write.

For more information, see Chapter 25, "Testing Expressions and Scripts," on page 309.

Chapter 24. Variables

There are two types of variable that you can use in IBM Cognos Data Manager:

- user-defined variables which store custom values
- property variables which are intrinsic to Data Manager and affect the operation of Data Manager programs

Declare Variables

You can declare variables within

- a build specification (build variables)
- a data source (data source variables)
- a JobStream (JobStream variables)
- a user-defined function (user-defined function variables)
- the environment of the operating system or in the file dm.ini (environment variables)
- in the configuration file dm.ini
- a command that invokes a Data Manager program (command line parameters)

The variable type determines the way in which you set it. For more information, see Declaration of Variables in Scripts in the IBM Cognos *Function and Scripting Reference Guide*.

You can declare variables that you later use within builds, JobStreams, derivations, derived dimensions, user-defined functions, and the Data Manager scripting language. You can declare these variables

- within the environment of the operating system
- within the definition of a build, JobStream, or user-defined function
- as command-line parameters

Scope of Variables

The scope of a Data Manager variable is the collection of Data Manager objects for which that variable is visible. A variable becomes visible with its declaration and is visible for the object in which it is declared, together with other objects contained within that object. For example, a variable that you declare in the environment of the operating system is visible for all executions of all Data Manager programs, but a variable that you declare as a command line parameter is visible only for the specific execution of the Data Manager program for which you declared it.

A variable remains visible until one of these events occurs:

- The object that contains the declaration of the variable terminates.
- Within the scope of the original variable, you declare another variable that has the same name. The original variable is displaced for the scope of the new variable.

Scoping can become quite complex where you have multiple objects with variables declared against them, because each object has its own scope. This allows recursive calls of Data Manager objects such as user-defined functions.

User-Defined Variables

A user-defined variable is a name and value pair that resides in the memory of the computer. User-defined variables affect the operation of IBM Cognos Data Manager programs, store values for use in builds and JobStreams, and control the flow of JobStreams.

User-Defined Variable Names

The name of a variable must

- start with an alphabetic character
- contain only alphanumeric characters, and underscores

Names of variables are not case sensitive and you can use a mixture of upper-case and lower-case characters in the name of a variable.

Notation of Variables and Substitution Variables

Variables may be referenced within builds and JobStreams using one of two notations.

The notation *$variable* means that the current value of *variable* is applied whenever *$variable* is encountered.

For example, a variable X may be defined and initialized to 0. In a build, a derivation may increment the value of X by 1 as each row is processed, that is, $X := $X + 1. Therefore, X represents the number of rows processed. However, the following expression would result in X always having a value of 1 after the first invocation because 0 would have been substituted for {$X} at the outset, $X := {$X} + 1

Note: The $*variable* syntax cannot be used within SQL SELECT statements.

The notation {$*variable*} means that the value of *variable* at the start of the process is used to perform a one-off replacement wherever {$*variable*} occurs. Even if the value of *variable* changes throughout the process, this is not reflected wherever {$*variable*} was used. This is known as a substitution variable.

Note: The {$*variable*} syntax cannot be used within user-defined functions.

Order of Precedence

The order of precedence for user-defined variables differs from the order of precedence for property variables. If you declare variables with the same name in more than one place

- build and JobStream variables take precedence over command-line parameters and environment variables
- command-line parameters take precedence over environment variables

Because of this, you cannot override user-defined build variables on the command line.

Add a User-Defined Variable

When you add user-defined variables, you can use any of the built-in IBM Cognos Data Manager functions.

For more information about Data Manager functions, see the IBM Cognos*Function and Scripting Reference Guide*.

Procedure

1. Click the Data Manager component to which you want to add a user-defined variable.
2. From the **Edit** menu, click **Properties** .
3. Click the **Variables** tab.
4. Click **Add**.

   The **Variable** window appears.
5. In the **Name** box, type a name for the new variable.
6. In the **Type** box, click the expected data type for the value that the variable is to return. For more information, see Appendix E, “Data Types,” on page 491.

   If you select **Char** or **Number** as the return type, the **Precision** box becomes available.
7. In the **Precision** box, type the maximum number of characters or numbers permitted for the value.

   Note: The precision can be up to 77 for the **Number** data type, and is only limited by the limitations of the operating system for the **Char** data type.

   If you select **Number** as the return type, the **Scale** box becomes available.
8. In the **Scale** box, type the number of decimal places permitted for the value.
9. In the right pane of the **Initial expression** box, enter an initial value for the variable if required.

   For information, see “Initial Values for Variables” on page 292

   The initial value can be an expression or a literal value. If you are entering an expression, you can type the expression, and you can use the tree structure in the left pane to select the items from the following folders:

   - **Operators** to select operators
   - **Functions** to select pre-defined and user-defined functions
   - **Control** to select controls
   - **Variables** to select variables and substitution variables

   Note: You can only reference other variables that are defined within the scope of the Data Manager component.

   Tip: If a variable references another variable in the expression, the referenced variable must be positioned before the variable in which it is used. Use **Move Up** or **Move Down** to reposition variables.

   You can select an item from a folder by dragging it to the right pane or by double-clicking it.

   For information about operators, controls, and functions, see the IBM Cognos *Function and Scripting Reference Guide*. For information on user-defined functions, see Chapter 23, “User-Defined Functions,” on page 277.
10. If required, test the expression.

    For information, see Chapter 25, “Testing Expressions and Scripts,” on page 309.
11. Click **OK**.

    The variable is added to the **Variables** tab in the **Properties** window.

Tip: To show the full variable expression, select the **Show all expression text** check box.

Initial Values for Variables

When you create a user-defined variable, you can enter an initial value for the variable in the right pane of the Initial expression box.

The initial value can be a simple expression or a literal value. For example, a variable named Variable1 can have the following initial values:

- 50 (which is a literal value)
- $var2 + $var3 (which is an expression)

It cannot however, include expressions of this type, as the result must be assigned to Variable1:

- $var4 := 10;

You can type an expression, and you can use the tree structure in the left pane to select the items from the following folders:

- Operators to select operators
- Functions to select pre-defined and user-defined functions
- Control to select controls
- Variables to select variables and substitution variables

You can select an item from a folder by dragging it to the right pane or by double-clicking it.

For information about operators, controls, and functions, see the IBM Cognos*Function and Scripting Reference Guide*. For information about user-defined functions, see Chapter 23, "User-Defined Functions," on page 277.

Note: You can only reference other variables that are defined within the scope of the IBM Cognos Data Manager component.

Reposition User-defined Variables

The order in which the variables are shown on the Variables tab is the order in which IBM Cognos Data Manager evaluates them.

If a variable references another variable in the expression, the referenced variable must be positioned before the variable in which it is used.

Procedure

1. Click the Data Manager object that contains the user-defined variable.
2. From the **Edit** menu, click **Properties** .
3. Click the **Variables** tab.
4. Click the variable to reposition.
5. Click **Move Up** or **Move Down** as appropriate.

Edit or Delete a User-Defined Variable

You can edit an existing user-defined variable or delete it if it is no longer required.

Procedure

1. Click the IBM Cognos Data Manager component that contains the user-defined variable.
2. From the **Edit** menu, click **Properties** .
3. Click the **Variables** tab.
4. Click the required variable.
5. Click **Edit** or **Delete**.
6. If you chose **Edit**, in the **Variable** window make the required changes.

   Note: If you rename a variable, you must update all references to that variable.

Property Variables

Property variables are intrinsic to, and affect the operation of IBM Cognos Data Manager. You can set most property variables within a build or JobStream or in the environment of the operating system.

Portability

Environment variables can aid build portability between computers and operating systems because they store properties specific to a particular computer.

Environment variables can also help optimize performance. For example, you can store performance-related parameters as environment variables. These settings reflect the optimum for executing the majority of builds on a particular computer. To obtain optimum performance on a different computer, modify only the batch file or shell script that prepares the computer for use with Data Manager.

Order of Precedence

If you declare a property variable in more than one place, the scope rules provide an order of precedence to resolve any potential conflict:

- Command-line parameters override build variables and JobStream variables.
- Build and JobStream variables override environment variables.
- Environment variables override variables defined in the Microsoft Windows operating system registry.

The order of precedence feature provides a convenient method for controlling build execution. For example, within the environment of the operating system, you can store the name of a text file that defines your default database aliases. You can use a command-line variable to override this setting, thus executing a particular build with alternative databases.

System Variables

These variables affect other system issues in IBM Cognos Data Manager.

COG_ROOT

This environment variable is specific to the UNIX operating system. It specifies the directory in which IBM Cognos Data Manager is installed.

The following example uses Bourne shell syntax to specify that Data Manager resides in /usr/cognos/c10

```
COG_ROOT="/usr/cognos/c10"; export $COG_ROOT
```

You can declare this variable in the environment of the operating system.

DM_DATASOURCE_CONNECTION_SIGNONS

When using IBM Cognos Connection with multiple data source connections, this variable specifies the data source or data sources to use.

The following example specifies, using a comma delimited list, that the data source details are /data source name/connection/signon name

```
DM_DATASOURCE_CONNECTION_SIGNONS='/gosales/oracle/signon1',gosales/oracle/signon2,
gosales/sybase/signon1
```

You can declare this variable in the environment of the operating system or in the build specification.

DM_DISTRIBUTED_HANDLING

This variable enables distributed handling of unmatched members in fact builds. Defining this variable as an empty string or NULL uses the following default settings:

- Allow more than one fact build to access the lock at a time (non-exclusive locking).
- Use two separate SELECT statements. The first is used to determine the existence of an unmatched member. The second is used to determine the maximum surrogate. This option overrides the default.
- Use a lock only for those dimensions for which it has been set up.
- Do not use the lock name as a MUTEX/semaphore (node locking).

The following table shows optional values that you can use. These values can be combined.

Value	Meaning
X	Use exclusive locking. This option allows only one fact build to access the lock at a time. Within a JobStream, this forces fact build nodes to be executed in sequence. **Note:** For fact builds that contain a large number of unmatched members, this can improve performance considerably.
C	Use a combined SELECT statement. This option overrides the default use of two SELECT statements, which can improve performance. **Tip:** If you are using Oracle, it is recommended that you use this option.
D	Use a default lock. This option uses a generic lock, named "dm_distributed_lock" for all dimensions, overriding any locks previously set up.

Value	Meaning
N	Use node locking. For simultaneous builds executed from a single computer, use the lock name as a MUTEX/semaphore. **Note:** This can improve performance considerably and removes the need to create database tables.

You can declare this variable in the environment of the operating system, in the file dm.ini, in a build or JobStream, or on the command line.

For more information about distributed handling of unmatched members in fact builds, see “Distributed Handling of Unmatched Members” on page 155.

DM_EMAIL_VIA_SERVICE

This variable specifies that when you execute a JobStream containing an email node, the email is sent using the IBM Cognos Business Intelligence delivery service rather than MAPI. The value of this variable should be set to 1.

You can declare this variable in the environment of the operating system or in the Variables section of the file dm.ini.

DM_SERVER_CAPACITY

This variable indicates the processing capacity of the computer. The value of this variable is an arbitrary number assigned by you. When you set this variable to a value greater than zero, this enables load control for JobStreams. When set to zero, load control is disabled.

You can declare this variable in the file dm.ini.

For more information about load control in JobStreams, see “Specify the Load Required to Process a Fact Build or Dimension Build Node” on page 240.

DM_TM1_LOAD_SUPPRESS_ERROR_CODES

(Default value: 136)

This variable specifies the error codes to ignore when delivering data to TM1. You specify error codes using a comma delimited list.

Note: If you decide to ignore minor errors, information about the number of rows successfully delivered may not be accurate.

You can declare this variable in the environment of the operating system, in the file dm.ini.

DS_CHAR_DEFAULT_PRECISION

(Default value: 255)

When IBM Cognos Data Manager creates delivery tables and the data type of a value, or the precision of a character value is not known, Data Manager uses a default precision of 255. This variable overrides the default, up to a limit of 8000. It must be less than or equal to the maximum CHARACTER precision of all target delivery DBMS.

You can declare this variable in the environment of the operating system, in the file named dm.ini, in a build or JobStream, or on the command line.

DS_DATA_DIR

This variable specifies the default directory into which text data files are delivered. If you do not declare this variable, the default data directory is as follows.

Operating system	Default data directory
Microsoft Windows operating system	The data subdirectory of the IBM Cognos Data Manager directory
UNIX operating system	The current directory

The following example uses a command-line parameter to direct text data files to directory c:\temp\ds.

```
databuild ODBC DSN=TestCat example -VDS_DATA_DIR=c:\temp\ds
```

You can declare this variable in the environment of the operating system, or on the command line.

You can also set the default location for data files using the -V option with the databuild, dimbuild, or rundsjob command, or using IBM Cognos Configuration.

Using the databuild, dimbuild, or rundsjob command takes precedence over a location set using the DS_DATA_DIR variable or IBM Cognos Configuration, and using the DS_DATA_DIR variable takes precedence over IBM Cognos Configuration.

For information on using the databuild, dimbuild, or rundsjob commands see, Chapter 31, “Commands,” on page 377. For information on using IBM Cognos Configuration, see the IBM Cognos Configuration *User Guide*.

DS_LOG_DIR

This variable specifies the default directory into which log files are delivered when you execute builds from a catalog. If you do not declare this variable, the default log file directory is as follows.

Operating system	Default data directory
Microsoft Windows operating system	The log subdirectory of the datamanager directory
UNIX operating system	The current directory

The following example uses Windows syntax to create an environment variable that specifies C:\Temp\Log to be the default log directory.

```
SET DS_LOG_DIR="C:\Temp\Log"
```

You can declare this variable in the environment of the operating system, or on the command line.

You can also set the default location for log files using the -V option with the databuild, dimbuild, or rundsjob command, or using IBM Cognos Configuration.

Using the databuild, dimbuild, or rundsjob command takes precedence over a location set using the DS_LOG_DIR variable or IBM Cognos Configuration, and using the DS_LOG_DIR variable takes precedence over IBM Cognos Configuration.

For information on using the databuild, dimbuild or rundsjob commands, see Chapter 31, “Commands,” on page 377. For information on using the IBM Cognos Configuration, see the IBM Cognos Configuration *User Guide*.

DS_PIPE_CREATE_WAIT

Default: 30

This variable specifies the maximum time (in seconds) that IBM Cognos Data Manager should wait for a pipe to be created. If you enter a negative value, Data Manager waits indefinitely.

You can declare this variable in the environment of the operating system.

DS_TMP_DIR, TMPDIR, and TEMP

These variables specify the directory in which IBM Cognos Data Manager should store any temporary files. If DS_TMP_DIR is not found, then TMPDIR is used on the UNIX operating system, and TEMP is used on the Microsoft Windows operating system.

The following example uses Cshell syntax to specify /usr/tmp to be the directory where Data Manager stores temporary files.

```
setenv DS_TMP_DIR /usr/tmp
```

You can declare this variable in the environment of the operating system, or on the command line.

DS_UDA_BULKWRITE_ROWS

(Default value: 50)

Array operations improve performance by reducing the number of round-trip operations between client and server components.

This variable indicates the number of rows to use for array writes if they are supported by the underlying IBM Cognos database driver. If they are not supported, they have no effect. If the driver does support array operations, they can be disabled by setting the variable value to 1.

If supported and enabled, array writes apply to INSERTs performed by relational table fact deliveries.

You can declare this variable in the environment of the operating system, in the file dm.ini, in a build or JobStream, or on the command line.

DS_UDA_FETCH_ROWS

(Default value: 50)

Array operations improve performance by reducing the number of round-trip operations between client and server components.

This variable indicates the number of rows to use for array reads if they are supported by the underlying IBM Cognos database driver. If they are not supported, they have no effect. If the driver does support array operations, they can be disabled by setting the variable value to 1.

If supported and enabled, array reads apply to most queries performed by IBM Cognos Data Manager. For data source queries, you can override the variable default value by using the **Fetch row count** option in Data Manager Designer. For more information, see "Define Advanced Settings for a Data Source" on page 118.

You can declare this variable in the environment of the operating system, in the file dm.ini, in a build or JobStream, or on the command line.

TM1_PATH

This environment variable is specific to the UNIX operating system. You use this variable to specify the location of the parent folder that contains the SSL certificate.

SSL certificates, distributed by the NGTM1API component, are located in the same folder as the libtm1api.so library within the *c10_location*\bin\ssl folder.

You can declare this variable in the environment of the operating system.

Run-time Variables

You should not declare or assign values to any run-time variables.

DS_AUDIT_ID

When you execute a build or a JobStream, this variable is set to a unique number that identifies each type of component.

DS_BUILD_FNAME

When you execute a build, this variable is set to the name of the build.

DS_BUILD_NAME

When you execute a build, this variable is set to the name of the build. If there are non-alphanumeric characters contained in the build name, these are replaced with underscores (_). This allows the variable to be used in file names.

For example, if the build name is ab*cd#, the variable is set to ab_cd_

DS_DATA_TIMESTAMP

When you execute a build, this variable is set to the value specified in the Data timestamp box in the Fact Build Properties window or the Dimension Build Properties window.

For more information about timestamping data, see "Set the Date Source for Timestamping Data" on page 111 or "Set the Date Source for Timestamping Data" on page 222.

DS_JOB_AUDIT_ID

When you execute a JobStream, this variable is set to a unique number that identifies the JobStream nodes.

DS_JOBSTREAM_FNAME

When you execute a JobStream, this variable is set to the name of the JobStream.

DS_JOBSTREAM_NAME

When you execute a JobStream, this variable is set to the name of the JobStream. If there are non-alphanumeric characters contained in the JobStream name, these are replaced with underscores (_). This allows the variable to be used in file names.

For example, if the build name is a*b*cd, the variable is set to a_b_cd

DS_LOG_NAME

When you execute a build or a JobStream, this variable is set to the name of the log file.

DS_RUN_ID

When you execute a build or a JobStream, this variable is set to a unique number that identifies the build or JobStream.

DS_RUN_TIMESTAMP

This is variable is set to the current date and time of the system on which you are running IBM Cognos Data Manager.

Debug-Related Variables

Use debug-related variables to save to the log file trace information that you can use when debugging a build.

DS_REJECT_NULL_VALUE

This variable specifies the value to use to represent NULL when writing to the reject file. Defining this variable as an empty string or NULL, results in an empty string representing NULL.

By default, <NULL> is used for fact builds and (null) for dimension builds.

Note: This variable has no effect when rejecting delivery data to tables.

MEMBER_FREQUENCY

(Default value: 10000)

This variable determines the frequency of DETAIL messages for member domain processing.

Note: To use this variable, you must also include DETAIL within the TRACE_VALUES specification.

This example uses Bourne shell syntax so that IBM Cognos Data Manager writes a DETAIL trace message at every 5,000th domain member it processes.

```
MEMBER_FREQUENCY=5000; export $MEMBER_FREQUENCY
```

You can declare this variable in the environment of the operating system, in a build or JobStream, or on the command line.

OTRACE_FREQUENCY

(Default value: 50000)

This variable specifies the number of output rows to process between DETAIL messages.

Note: To use this variable, you must also include DETAIL within the TRACE_VALUES specification.

This example creates a build variable so that a DETAIL trace message is written every 100,000th output data rows.

```
DECLARE OTRACE_FREQUENCY INTEGER 100000
```

You can declare this variable in the environment of the operating system, in a build or JobStream, or on the command line.

REJECT_FILE

This variables specifies the log file to which data row rejections are logged.

In the following example, all data row rejects are logged to the text file C:\temp\reject.log.

```
DECLARE REJECT_FILE CHAR(20) 'C:\\temp\\reject.log'
```

You can declare this variable in a build or JobStream, or on the command line.

If you are using IBM Cognos Data Manager with multilingual or unicode data, you should specify unicode encoding for your reject files. By default, the encoding used is platform dependent. For information, see "Specify the Encoding to Use for Reject Files" on page 342.

TRACE_FREQUENCY

(Default value: 5000)

This variable specifies the number of input rows to be processed between DETAIL and INTERNAL trace messages.

Note: To use this variable, you must include DETAIL within the TRACE_VALUES specification.

This example uses Korn shell syntax so that IBM Cognos Data Manager writes a trace message every 10,000th input data row.

```
TRACE_FREQUENCY=10000; export $TRACE_FREQUENCY
```

You can declare this variable in the environment of the operating system, in a build or JobStream, or on the command line.

TRACE_VALUES

(Default value: PROGRESS)

This variable determines the events that IBM Cognos Data Manager traces during execution of a build. The following table shows the trace values that you can use.

Value	Meaning
PROGRESS	Shows the overall progress of the application.
DETAIL	Shows more detailed progress messages.
INTERNAL	Shows internal Data Manager activity messages, including resource usage. For example, memory usage, dimension sizes, and paging information.
SQL	Shows all the SQL statements used at each stage of execution.

Value	Meaning
EXECUTED SQL	Shows the executed SQL for SELECT statements.
VARIABLE	Shows all the variables used in the build.
USER	Shows all the application messages written to the log file. You can include additional application messages using the LogMsg function.

Tip: You can specify combinations of values by separating the values with commas.

The following example declares TRACE_VALUES as a build variable and sets it to include general progress and SQL information within the log file.

```
DECLARE TRACE_VALUES CHAR(12) 'PROGRESS,SQL'
```

You can declare this variable in the environment of the operating system, in a build or JobStream, or on the command line.

Audit-related Variables

Audit-related variables determine the auditing information written to the current catalog when a build or JobStream is executed. Auditing is available when you execute builds or JobStreams that reside in a IBM Cognos Data Manager catalog or package.

AUDIT_VALUES

Default: no auditing

This variable specifies the type of audit information that IBM Cognos Data Manager writes to the current catalog. The following table shows the audit values that you can use.

Value	Meaning
TIMING	Records overall timing information. For example, the start, end, and elapsed times for the execution process.
ACQUIRE	Records information about the source data acquired. For example, the number of data rows that each data source contributes. This information is not available for JobStreams.
TRANSFORM	Records information about the data processed. For example, the number of data rows received from the data sources. This information is not available for dimension builds or JobStreams.
DELIVER	Records information about data delivery. For example, the number of data rows delivered. This information is not available for JobStreams.

Value	Meaning
INTERNAL	Information about internal measures of performance. For example, the size of the hash table. This information is not available for JobStreams.
ALERT	Records a user-defined audit record of type ALERT, into the Data Manager audit tables.
USER	Records user-controlled audit messages.

Unless specified in the table, you can declare this variable in the environment of the operating system, in a build or JobStream, or on the command line.

Performance-related Variables

These variables relate to the way that IBM Cognos Data Manager uses computer memory.

DS_BREAK_DIMENSION

This variable is intrinsic to all IBM Cognos Data Manager builds. It is available for each data row that the engine processes. If a dimension break occurs as Data Manager processes the row, then this variable contains the name of the breaking dimension. If a break simultaneously occurs in more than one dimension, then this variable contains the name of the most significant. If no dimension break occurs when processing the data row, this variable contains the null value.

For information about dimension breaking, see "Dimension Breaks" on page 344.

Note: This variable is set at run-time, so you should not declare or assign values to it.

The following expression is a calculation for a derivation transformation model element:

```
IfNull(ToChar($DS_BREAK_DIMENSION),'<NULL>')
```

If the current data row causes a dimension break, this expression returns the name of the breaking dimension. If not, this expression returns the literal value '<NULL>'.

DS_LOB_BUF_SIZE

Default buffer size: 8000 bytes

The size of the buffer (in bytes) used for BLOB and CLOB data transfer. This can be used to fine tune BLOB and CLOB performance.

Note: CLOB data is held in UTF-16 encoding so a buffer size of 8000 is equivalent to 4000 characters.

DS_LOB_MAXMEM_SIZE

Default: 65536 bytes

If a BLOB or CLOB is less than the buffer size then it will be processed in memory otherwise it will be written to a temporary file. This applies to each LOB in a row, so if DS_LOB_BUF_SIZE is 8KB and there are four LOBs in a row then 32KB of memory are allocated.

HASH_TABLE_SIZE

Default: 200,000 slots

The hash table is a structure in computer memory that IBM Cognos Data Manager uses to process and transform rows of data. During build execution, Data Manager calculates the maximum number of hash table slots and uses this value if it is smaller than the default or user-specified value. For any build, you can determine the minimum hash table size by including INTERNAL in the TRACE_VALUES specification. Execute the build in check only mode and then search towards the end of the log for an entry like this where <nnnn> gives the minimum hash table size in slots:

[INTERNAL - *hh:mm:ss*] Hash: minimum table size <nnnn>

The following example declares HASH_TABLE_SIZE as a build variable that sets the minimum hash table size to 64,000 slots.

```
DECLARE HASH_TABLE_SIZE INTEGER 64000
```

You can declare this variable in the environment of the operating system, in a build or JobStream, or on the command line.

MEMORY_LIMIT

Default: no limit

This variable places an upper limit (in megabytes) on the amount of memory that IBM Cognos Data Manager may request from the operating system.

Some operating systems terminate any application that requests more memory than the operating system can provide. Therefore, you may want to impose a memory limit to prevent this. You may also want to impose a memory limit to prevent Data Manager requesting memory that other running applications need.

The following example declares MEMORY_LIMIT as a build variable that prevents Data Manager from requesting more than 64MB of memory from the operating system.

```
DECLARE MEMORY_LIMIT INTEGER 64
```

You can declare this variable in the environment of the operating system, in a build or JobStream, or on the command line.

PAGE_SIZE

Default: 16384 bytes

When IBM Cognos Data Manager reaches the limit of available memory, it creates virtual memory by paging information to disk. This variable determines the maximum page size that Data Manager uses. The maximum page size determines how much paging occurs and how many data rows Data Manager stores on each page. To conserve memory, Data Manager automatically adjusts the maximum page size downward to accommodate exactly a whole number of data rows.

The following example uses Cshell syntax to declare PAGE_SIZE as an environment variable that sets the maximum page size to 32,768 bytes (32kb).

```
setenv PAGE_SIZE 32768
```

You can declare this variable in the environment of the operating system, in a build or JobStream, or on the command line.

PAGE_TABLE_SIZE

Default: calculate at run-time

The page table is a table in computer memory that IBM Cognos Data Manager uses to track the location of all the pages of memory. This variable specifies the initial page table size in number of page entries. By default, Data Manager determines the initial page table size from the size of the hash table.

The following example uses Microsoft Windows operating system syntax to declare PAGE_TABLE_SIZE as an environment variable that sets the initial page table size to 128 page entries.

```
SET PAGE_TABLE_SIZE=128
```

You can declare this variable in the environment of the operating system, in a build or JobStream, or on the command line.

DBMS-related Variables

These variables affect the specification of builds and reference structures.

DS_DBALIAS_FILE

Default: none

This variable affects only file-based projects. It points to a file that contains database alias definitions. Only one database alias definition file may be active at any one time, so this file must contain a definition for each alias that the current project requires.

For catalog-based projects, you must use the -A parameter to replace the connections that the catalog specifies with the aliases of an alias definition file. For information about using the -A parameter with each IBM Cognos Data Manager command, see the IBM Cognos *Function and Scripting Reference Guide*.

The following example uses Microsoft Windows operating system syntax so that the aliases of the file c:\cognos\projects\alias.txt are used by default.

```
SET DS_DBALIAS_FILE="c:\cognos\projects\alias.txt"
```

You can declare this variable in the environment of the operating system.

DS_DB2_LOADER

This variable specifies the command that invokes the DB2 bulk load utility. By default, IBM Cognos Data Manager uses db2cmd /c /w /i db2 under the Microsoft Windows operating system and uses db2 under the UNIX operating system. To specify another load command, either specify it in this variable or define the CMD property of each DBLOAD delivery. The CMD property takes precedence over the DS_DB2_LOADER variable.

The following example uses Bourne shell syntax to cause Data Manager to use db2 when delivering data using the DB2LOAD delivery module.

```
DS_DB2_LOADER='db2'; export DS_DB2_LOADER
```

You can declare this variable in the environment of the operating system, in the file dm.ini, in a build or JobStream, or on the command line.

DS_DBMS_TRIM

This variable causes IBM Cognos Data Manager to right trim white space when reading characters.

Value	Meaning
NONE	Whitespace is not trimmed
SPACE	Only the space character is trimmed (default)
FULL	All whitespace is right trimmed

DS_DBMS_TRUNCATION

This variable determines the action IBM Cognos Data Manager takes when writing character data to a column that is too small to hold that data.

Value	Meaning
ALLOW	Allows Data Manager to truncate characters as required.
WARN	Allows Data Manager to truncate characters as required, but issues a warning to inform you. You can optionally add a maximum size of the value to include in the warning message: • WARN issue a general warning when truncation is detected • WARN=255 issue the same warning and include a separate message with up to 255 characters of the value being truncated
ERROR	Issue an error message and terminate. You can optionally add a maximum size of the value to include in the error message: • ERROR=255 issue a general error when truncation is detected (default)

DS_INFORMIX_LOADER

This variable specifies the command that invokes the Informix bulk load utility. By default, IBM Cognos Data Manager uses dbaccess. To specify another bulk load command, either specify it in this variable or define the CMD property of each INXLOAD delivery. The CMD property takes precedence over the DS_INFORMIX_LOADER variable.

The following example uses Microsoft Windows operating system syntax to cause Data Manager to use dbaccess when performing bulk loads to Informix databases.

```
SET DS_INFORMIX_LOADER="dbaccess"
```

You can declare this variable in the environment of the operating system, in the file dm.ini, in a build or JobStream, or on the command line.

DS_MAX_RECURSION

This variable specifies the maximum recursive depth of user-defined functions which IBM Cognos Data Manager will allow when executing a script. For example, if DS_MAX_RECURSION = 5 then Function1 would succeed but Function2 would fail with error "Maximum recursion of functions has been exceeded". This is to prevent scripts from looping uncontrollably. The default value is 100.

```
Integer Function1(integer count)
If (count < 6) then
    Function1(count+1);
else
    return count;
Integer Function2(integer count)
If (count < 7) then
    Function1(count+1);
else
    return count;
```

DS_MSSQLSERVER_BCP

This variable specifies the command that invokes the Microsoft SQL Server BCP bulk load utility. By default, IBM Cognos Data Manager uses bcp. To specify another load command, either specify it in this variable or define the CMD property of each Microsoft SQL Server BCP delivery. The CMD property takes precedence over the DS_MSSQLSERVER_BCP variable.

The following example causes Data Manager to use fastload when delivering to Microsoft SQL Server BCP databases.

```
SET DS_MSSQLSERVER_BCP="bcp"
```

You can declare this variable in the environment of the operating system, in the file dm.ini, in a build or JobStream, or on the command line.

DS_ORACLE_LOADER

This variable specifies the command that invokes the Oracle bulk load utility. By default, IBM Cognos Data Manager uses sqlldr. For Data Manager to use the correct bulk loader, either specify it in this variable or define the CMD property of each ORALOAD delivery. The CMD property takes precedence over the DS_ORACLE_LOADER variable.

For example, if you have both Oracle 9i and Oracle 10 client installed, and you want to be sure that the Oracle 9i version of sqlldr is used, set this variable to the full path name, such as

```
SET DS_ORACLE_LOADER=c:\Oracle\Product\9.2\Client_1\bin\sqlldr.exe
```

Note: If no value is specified for this variable, and the CMD property of the ORALOAD delivery is not set, the PATH environment variable determines which version to use.

You can declare this variable in the environment of the operating system, in the file dm.ini, in a build or JobStream, or on the command line.

DS_REDBRICK_LOADER

This variable specifies the command that invokes the RedBrick load utility. By default, IBM Cognos Data Manager uses rb_tmu. To specify another load command, either specify it in this variable or define the CMD property of each RBLOAD delivery. The CMD property takes precedence over the DS_REDBRICK_LOADER variable.

The following example uses Microsoft Windows operating system syntax to cause Data Manager to use rb_tmu when delivering data to Redbrick databases.

```
SET DS_REDBRICK_LOADER="rb_tmu"
```

You can declare this variable in the environment of the operating system, in the file dm.ini, in a build or JobStream, or on the command line.

DS_RUNENV

Default: <DEFAULT>

This variable affects only file-based projects. It specifies the run environment to use when acquiring or delivering data. The active database alias definition file must define all the required aliases for this run environment.

To specify a run environment on the command line, use the -e command-line parameter. For more information, see the IBM Cognos *Function and Scripting Reference Guide*.

Note: If you are importing a database alias that is using a run environment other than <DEFAULT>, you must ensure that the DS_RUNENV variable is set to the required value. For more information, see Appendix B, “Database Drivers,” on page 465.

The following example assumes that by default, IBM Cognos Data Manager connects to the Oracle database.

```
<DEFAULT> Sales ORACLE user/password@service
DEVELOPMENT Sales ODBC 'DSN=Sales;UID=user;PWD=password'
```

However, you may not want Data Manager to do this while you are developing a build. If you are using the Korn shell you can specify that Data Manager should use the DEVELOPMENT run environment, and so use the ODBC development data source rather than the Oracle production data. You do this by typing

DS_RUNENV=DEVELOPMENT; export $DS_RUNENV

You can declare this variable in the environment of the operating system.

DS_TERADATA_FASTLOAD

This variable specifies the command that invokes the Teradata Fastload bulk load utility. By default, IBM Cognos Data Manager uses fastload. To specify another load command, either specify it in this variable or define the CMD property of each Teradata Fastload delivery. The CMD property takes precedence over the DS_TERADATA_FASTLOAD variable.

The following example causes Data Manager to use fastload when delivering to Teradata Fastload databases.

```
SET DS_TERADATA_FASTLOAD="fastload"
```

You can declare this variable in the environment of the operating system, in the file dm.ini, in a build or JobStream, or on the command line.

DS_TERADATA_MULTILOAD

This variable specifies the command that invokes the Teradata Multiload bulk load utility. By default, IBM Cognos Data Manager uses mload. To specify another load command, either specify it in this variable or define the CMD property of each Teradata Multiload delivery. The CMD property takes precedence over the DS_TERADATA_MULTILOAD variable.

The following example causes Data Manager to use mload when delivering to Teradata Multiload databases.

```
SET DS_TERADATA_MULTILOAD="mload"
```

You can declare this variable in the environment of the operating system, in the file dm.ini, in a build or JobStream, or on the command line.

DS_TERADATA_TPUMP

This variable specifies the command that invokes the Teradata TPump bulk load utility. By default, IBM Cognos Data Manager uses tpump under the Microsoft Windows operating system and uses tmpump under the UNIX operating system. To specify another load command, either specify it in this variable or define the CMD property of each Teradata TPump delivery. The CMD property takes precedence over the DS_TERADATA_TPUMP variable.

The following example uses Windows syntax to cause Data Manager to use tpump when delivering to Teradata TPump databases.

```
SET DS_TERADATA_TPUMP="tpump"
```

You can declare this variable in the environment of the operating system, in the file dm.ini, in a build or JobStream, or on the command line.

Chapter 25. Testing Expressions and Scripts

It is important to be able to adequately test expressions and scripts to ensure they perform as expected during execution. To facilitate this, IBM Cognos Data Manager provides a testing utility. The objective of this is to simulate execution conditions as closely as possible, and to allow values to be manipulated so that expressions can be tested using a wide range of data values.

You can test expressions and scripts for

- derivation and derived dimension elements
- user-defined functions
- output filters
- JobStream procedure and condition nodes
- variable definitions

Note: You can test all internal user-defined functions in Data Manager Designer. You can also test external user-defined functions created under the Microsoft Windows operating system; those created under the UNIX operating system cannot be tested from Data Manager Designer.

You test expressions and scripts in the Values for Expression window. The Input values and the Output values boxes have the following columns.

Column	**Description**
Status icon	✓ indicates a definition for the item has been located. For information, see “Scope of Expressions and Scripts” on page 310. ✗ indicates a definition for the item has not been located, either because it has not yet been defined, or because it is defined at a higher scope than is currently known to Data Manager. For information, see “Scope of Expressions and Scripts” on page 310. ❗ indicates a definition has been located, but an error exists within it. To view details of the error, double click the exclamation mark. For information, see “Default Values in Expressions and Scripts” on page 313.
Name	Lists all the items for which you must specify an initial value.
Type	If the definition can be located, shows the data type.
Value	You enter a value as required.

The Output values box shows any items to which the expression or script assigns a value.

When you execute a build or JobStream you can use logging to debug expressions and scripts contained within a build or JobStream. For more information, see "Create a Log File for Expression and Script Tracing" on page 264.

Test an Expression or Script

This section describes how to test an expression or a script.

Procedure

1. In the appropriate window, click **Test**.

   Tip: When developing a complex script, you may want to test just a section of the expression or script. You can do this by selecting the relevant section, and then clicking **Test**.
2. If required, in the **Top** box, specify the top scope of the expression or script by clicking the **Browse** button ..., and selecting the appropriate object that contains the definitions.

   For information, see "Scope of Expressions and Scripts."
3. If required, in the **Types** column in the **Input values** box, specify the data type.

   For more information, see "Data Types in Expressions and Scripts" on page 311.
4. In the **Value** column in the **Input values** box, type a value for each item.

   Tip: You can overtype any default values shown.
5. If you want IBM Cognos Data Manager to determine the data types and initial values from the definitions, click **Set Default Values**.

   For more information, see "Default Values in Expressions and Scripts" on page 313.
6. Click **Calculate**.

   The result of the calculation is shown in the **Result** box.

   Note: If you are executing a script, the Result box shows the value from the RETURN statement that caused the script to terminate.

   The **Output values** box shows any items to which the expression or script assigned a value. The final value of each of these items is shown.
7. If the result is incorrect, make the necessary changes, and perform the test again.
8. Click **Close**.

Scope of Expressions and Scripts

By default, IBM Cognos Data Manager only searches for item definitions within the *immediate scope* of the expression or script. For example, if you are testing a derivation in a fact build, the immediate scope is the fact build in which the derivation is defined. If any fact build variables are used in the derivation, Data Manager finds them.

Note: The immediate scope includes any variables defined within the current expression or script as well as variables declared within the operating system environment level.

If a definition is located, a check mark appears next to the item name in the Input values box.

If a definition is not found, a cross appears to indicate that either

- the item has not yet been defined
- the item is defined at a higher scope that is currently unknown to Data Manager

If an item has been defined but its scope is unknown, for testing purposes, you can specify the *top scope* of the expression or script using the Top box.

Suppose you define a variable in a JobStream and reference that variable from an expression defined in a fact build. Because the fact build relies on the inheritance of the variable from the JobStream, the fact build fails if you try to execute it independently of the JobStream. In this instance, the JobStream is the top scope for the expression.

For more information, see “Scope of Variables” on page 289.

When you specify the top scope, Data Manager attempts to resolve the path between the immediate and top scope so that it can locate the required item definitions.

Note: If Data Manager cannot resolve the path, only the immediate scope is used for testing.

Data Types in Expressions and Scripts

If an item definition is found within the immediate or top scope, its data type is determined from the definition. In this case, the data type appears in the Type column for the appropriate item.

For items that require an input value, if a definition is not found or no data type has been set in the definition, the data type is set to UNKNOWN and the data type is deduced from the test value you specify.

Value format	Example test values	Data type deduction
Characters enclosed in single quotation marks	'10.0'	CHAR
Digits without a decimal point	100	INTEGER
Digits with a decimal point	10.0	FLOAT
Text that represents a date in yyyy-mm-dd hh:mi:ss.fffffffff format	2005-06-22 10:30:15 123	DATE
Text that represents a time in hh:mi:ss.fffffffff format	10:30:15.123	TIME
Text that represents a time in hh:mi:ss format	10:30:15	TIME
Text that represents a date in yyyy-mm-dd hh:mi:ss.fffffffff stzh:tzm format	2005:06:22 10:30:15.123 -8.00	DATE WITH TIME ZONE

Value format	Example test values	Data type deduction
Text that represents a time in hh:mi:ss.ffffffff stzh:tzm format	10:30:15.123 -8.00	TIME WITH TIME ZONE
Text that represents an interval in sddddddddd hh:mi:ss	-10 10:30:15.123	INTERVAL DAY TO SECOND
Text that represents an interval in syyyyyyyyy-mm format	1-11	INTERVAL YEAR TO MONTH
TRUE or FALSE	TRUE	BOOLEAN
No value given		UNKNOWN
Text that includes ampersands	'a' & 'b' & 'c' a & b & c	ARRAY
Any other format	abCdef	CHAR

Notes

- When entering a string literal NULL value, you must use single quotation marks, that is, 'NULL'.
- If the precision and scale is greater than the limits allowed for INTEGER or FLOAT, NUMBER is used.
- For quoted character values (for example, '100.12'), the initial character string does not include the quotation marks.
- The expression or script may behave unpredictably if the run-time value is of a different data type to that determined at testing.

If you do not want IBM Cognos Data Manager to deduce the data type, you can explicitly set a data type.

Any input values that you have specified, are changed using the new data type. For example, if you specified an input value of 12.345 before setting the data type, the data type is deduced to be FLOAT. If you then set the data type to INTEGER, the input value changes to 12.

If you enter an input value that is invalid for the specified data type, Data Manager changes the value to 0. For example, if you set the data type to DATE, entering ABCD is an invalid value.

Note: You can only set an explicit data type where the data type is initially UNKNOWN.

Procedure

1. For the appropriate item, click within the **Type** column, then click the **Browse** button ... that appears.

   The **Data Type** dialog box appears.

2. In the **Data type** box, click the required data type. For information, see Appendix E, "Data Types," on page 491.

   If you select **CHAR**, **NUMBER**, or **BINARY** as the data type, the **Precision** box becomes available.
3. In the **Precision** box, type the maximum number of characters or numbers permitted for the value.

   Note: The precision can be up to 77 for the **NUMBER** data type, and for the **CHAR**data type is limited only by the operating system restrictions.

   If you select **NUMBER** as the data type, the **Scale** box becomes available.
4. In the **Scale** box, type the number of decimal places permitted for the value.
5. Click **OK**.

Default Values in Expressions and Scripts

An item definition can include an initial value. Where a definition can be located, the initial value is not automatically obtained because you may want to test your expression or script using your own values.

Note: IBM Cognos Data Manager searches for definitions within the immediate scope, the top scope, if it is specified, and any environment variables.

Procedure

Click **Set Default Values**.
Data Manager searches for definitions of each item listed in the **Input values** and **Output values** boxes. For each definition found, an initial value is entered in the **Value** column, if one has been set in the definition.

Results

If an error exists in a definition, the initial value is not obtained and an exclamation mark appears next to the item name. Although you must resolve the error before execution, for the purposes of testing you can enter a value and continue testing the expression or script.

Note: If an initial value is defined as an expression, obtaining default values may take several minutes because of the processing required to evaluate the expression.

Chapter 26. Multi-Developer Support

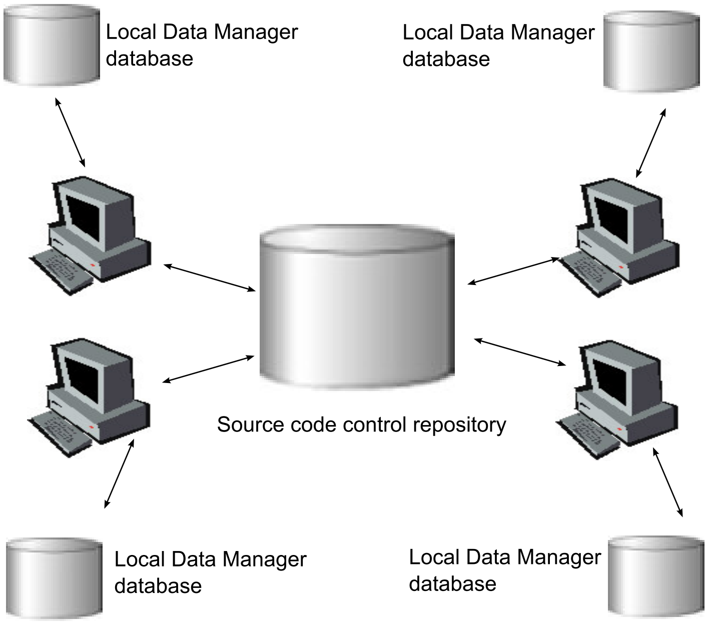
Multi-developer support allows developers to share information in a catalog. There are various ways in which you can achieve multi-developer support in IBM Cognos Data Manager:

- source code control
- component packages
- backup and restore

The method best suited to you depends upon the number of developers involved in your project and the environment in which you are working.

Source Code Control

Storing an IBM Cognos Data Manager catalog in a source code control system allows multiple developers concurrent access. A master Data Manager catalog is added to the source code control system, and each developer then creates a personal catalog that is connected to the source code control repository as illustrated here.


Local Data Manager database
Local Data Manager database
Source code control repository
Local Data Manager database
Local Data Manager database

Note: To use source code control, you must have an external source code control system already set up and you must be connected to it.

Using source code control allows all connected users to view all the contents of the catalog, execute builds and JobStreams, and create components. To maintain existing catalog components, you must first check them out of the source code control system. For more information, see “Check Out Catalog Components” on page 318.

Guidelines for Planning a Source Code Control Project

If you plan to have multiple developers working on a project, it is important to plan the project at the outset.

When planning a source code control project, you should consider the following:

- Each developer must be aware of the project areas on which they are to work.
- Common components that must be shared by all developers, such as connections and user-defined functions, should be implemented first.
- Once implemented, you can lock common components by checking them out so that no developer can change them.
- Dimensional framework components should be implemented next because reference structures are used in many places in a catalog.
- Complete the development of all fact builds and dimension builds before integrating them into JobStreams.

Enable Source Code Control

Before you can use source code control you must enable it within IBM Cognos Data Manager.

Procedure

1. Close any open catalogs.
2. From the **Tools** menu, click **Options**.
3. Click the **Source Code Control** tab.
4. Select **Enable source code control**.
5. Click **OK**.

Add a Catalog to Source Code Control

You add a catalog to source code control to create a master copy of the catalog which other developers can share.

Note: If you have upgraded from IBM Cognos DecisionStream Series 7, and you were previously using source code control, you must add your upgraded catalogs to source code control. For more information, see “Post Upgrade Considerations” on page 28.

Procedure

1. Open the catalog that you want to add to source code control.
2. From the **Tools** menu, click **Source Code Control**, **Add Catalog to Source Code Control**.

   A login dialog box appears.
3. Log in to your source code control system.

   The **Add to Project** dialog box appears.
4. Select the project to which you want to add the catalog.

   Tip: If no suitable project exists, create one by typing a project name in the **Project** box.
5. Click **OK**.

   When the catalog has been successfully added to source code control, a message appears informing you of this, and the following files are created:

   - a catalog project file

- a file for each component in the IBM Cognos Data Manager catalog

Data Manager sets the status of every catalog component to read-only and, in the **Tree** pane, additional icons appear next to each component to indicate their current status under source code control.

For more information, see "Source Code Control Icons" on page 505.

Create a Shared Copy of a Catalog from Source Code Control

When a catalog has been added to source code control, a master copy of the catalog is created. Every developer who wants to share that catalog must create a copy of it on their computer from the source code control project.

Note:

- Multiple developers can all be connected to the same source code control system. However, it is important that they are not using the same connection to a catalog within IBM Cognos Data Manager, otherwise the source code control system may behave incorrectly. Each developer must set up their own personal catalog, and insert a shared copy of the master catalog into it. It is the source code control system that allows multi developer use, not the physical Data Manager catalog.
- When inserting a shared copy of a master catalog into a personal catalog, a developer must log in using their own ID, not a common ID or administrator ID, because user information is stored in the catalog.

Procedure

1. In Data Manager, create an empty catalog or, if you want to overwrite an existing catalog, open the relevant catalog.

   For information, see "Create an IBM Cognos Data Manager Catalog" on page 20.
2. From the **Tools** menu, click **Source Code Control**, **Create Catalog**, **Create Catalog from Source Code Control**.

   A warning appears informing you that the current catalog will be overwritten.
3. Click **Yes** to continue.

   A login dialog box appears.
4. Log in to your source code control system.

   The **Choose Project** dialog box appears.
5. Select the project containing the catalog to copy, and then click **OK**.

   Data Manager overwrites the current catalog with a copy of the catalog from source code control, and sets the status of every catalog component to read-only.

View the Source Code Control Properties of a Component

You can view the source code control properties of a specified component. This is useful for verifying status information such as

- the name of the developer who has checked out a component
- when the component was last updated
- the number of changes that have been made to a component

Procedure

1. Click the component for which you want to check the source code control properties.

2. From the **Tools** menu, click **Source Code Control**, **Source Code Control Properties**.

   A dialog box appears, showing information about the selected component.

Refresh the Status of Components

The source code control icons shown in the Tree pane reflect the current status of components in the catalog. When you open a catalog that is connected to a source code control repository, IBM Cognos Data Manager automatically updates the icons so that you can see what is currently checked in and checked out. To update the icons you must manually refresh the status.

For more information, see “Source Code Control Icons” on page 505.

Note: Data Manager does not synchronize the catalog when it refreshes the status. For information, see “Obtain the Latest Version of a Catalog.”

Procedure

From the **Tools** menu, click **Source Code Control**, **Refresh Component Statuses**. The **Tree** pane is updated to reflect the status of the master catalog.

Obtain the Latest Version of a Catalog

If another developer checks in changes to the master catalog stored in the source code control system, you must synchronize your local catalog to reflect the changes.

Note: When you check in or check out components, if necessary you are automatically prompted to synchronize your catalog.

Before synchronizing the catalog, IBM Cognos Data Manager compares each component in the catalog with the master catalog and shows a summary of the changes required. This can include

- creating components in your local catalog that exist in the master catalog
- updating components in your local catalog
- deleting components from your local catalog that no longer exist

When you synchronize a catalog, Data Manager refreshes its status.

Procedure

1. From the **Tools** menu, click **Source Code Control**, **Get Latest Version of Catalog**.

   The **Results** window appears with a summary of changes required to synchronize your catalog.
2. Click **OK** to continue.

Check Out Catalog Components

To amend components in a catalog that is connected to a source code control repository, you must first check out the components.

You can check out individual components, for example a specific dimension build, or you can check out an entire catalog. If you are checking out individual components, IBM Cognos Data Manager also forces you to check out any component dependents, such as templates used by a dimension build. However, if the selected component includes dependents that are checked out by another developer, you cannot check it out.

Note: When you check out a user-defined folder, only the folder is checked out, not its contents. You must manually check out any required components.

Procedure

1. In the **Tree** pane, click the component to check out.

   Note: You cannot check out components that are already checked out by another developer.
2. From the **Tools** menu, click **Source Code Control**, **Check Component Out**.

   If another developer has made changes to the master catalog, a message appears informing you to synchronize the catalog first.

   For information, see “Obtain the Latest Version of a Catalog” on page 318. Repeat steps 1 and 2 after synchronizing.

   If the selected component has dependents, a list of the dependent components appears. You must also check out these components. Click **OK** to confirm that you want to check them out.

   Data Manager checks out the components from source code control, changes their status to read and write, and updates the source code control icons in the **Tree** pane to indicate their current status.
3. Maintain the components as required.

   When you have finished maintaining the components, you must check the catalog into source code control. For information, see “Check a Catalog into Source Code Control.”

Check a Catalog into Source Code Control

You must check your local catalog into your source code control system when

- you create components
- you have finished maintaining components that you previously checked out

The check-in process applies to an entire catalog.

IBM Cognos Data Manager compares each component in your local catalog with the master catalog and shows a summary of the changes required. This can include

- adding new components
- updating existing components
- deleting components that no longer exist in your catalog

Procedure

1. From the **Tools** menu, click **Source Code Control**, **Check Catalog In**.

   If another developer has made changes to the master catalog, a message appears informing you to synchronize the catalog first. For information, see “Obtain the Latest Version of a Catalog” on page 318.
2. After synchronizing, repeat step 1.

   The **Check In Catalog** dialog box appears.
3. In the **Comment** box, type a description of the catalog changes you are checking in, and then click **OK**.

Data Manager updates the master catalog, resets the status of all catalog components to read-only, and refreshes the icons in the **Tree** pane accordingly.

Undo a Catalog Check Out

If you have checked out components, but not made any changes, or you want to abandon your changes, you can undo the check out.

The undo check-out process applies to an entire catalog.

Procedure

From the **Tools** menu, click **Source Code Control**, **Undo Catalog Check Out**.

Open the Source Code Control System from IBM Cognos Data Manager

You can open the source code control system that you are using directly from IBM Cognos Data Manager.

Procedure

From the **Tools** menu, click **Source Code Control**, **Run Source Code Control Client**.

Detach a Catalog from Source Code Control

You can detach your local catalog from the source code control system. This removes all references to the source code control system and provides full read and write access to the catalog components for single users. To make the catalog available to multiple users again, you must recreate a local catalog from the source code control system.

Procedure

1. From the **Tools** menu, click **Source Code Control**, **Remove Source Code Control Information**.

   If there are components currently checked out, a message appears recommending that you either check in the catalog or undo the component check out before you detach the catalog from source code control.
2. If you do not want to proceed with detaching the catalog, click **No** to cancel. If you do want to proceed with detaching the catalog, click **Yes**.

   IBM Cognos Data Manager removes the source code control icons from the **Tree** pane.

Create a Standalone Copy of a Catalog from Source Code Control

You can create a standalone copy of the master catalog directly from the source code control project files. The catalog you create is not linked to source code control. It is a local catalog that is stored only on your computer. You have full read and write access to the catalog components.

You may want to use this, for example, to roll back to a previous version of the master catalog or to restore the catalog from a specific date.

Note: To roll back to a specific version or date, the catalog must be labelled in the source code control system.

Procedure

1. In the source code control system, use the Get Latest Version command to copy the project files for the master catalog onto your hard drive.
2. In IBM Cognos Data Manager, create an empty catalog or, if you want to overwrite an existing catalog, open the relevant catalog.
3. From the **Tools** menu, click **Source Code Control**, **Create Catalog**, **Create Catalog from Project File**.

   A warning appears informing you that the current catalog will be overwritten.
4. Click **Yes** to continue.

   The **Browse for Folder** dialog box appears.
5. Select the folder containing the catalog project files, then click **OK**.

   Data Manager creates a standalone copy of the catalog, which you can now maintain.

Temporary Files Used By Source Code Control

IBM Cognos Data Manager temporarily stores files on your hard drive whenever you work with source code control. By default, temporary files are stored in the directory named Documents and Settings\<username>\Local Settings\Temp\sccs. You can change this to a directory of your choice.

Temporary files are automatically deleted by Data Manager when the required task is complete.

Procedure

1. From the **Tools** menu, click **Options**.
2. Click the **Source Code Control** tab.
3. In the **Default directory for SCCS files** box, enter the directory where you want the temporary files to be stored.
4. Click **OK**.

Component Packages

You can use component packages to

- move components between catalogs
- combine the efforts of several developers
- distribute catalogs or components
- partially backup and restore a catalog

  Note: If you want to backup and restore a complete catalog, see “Backup and Restore Catalogs” on page 327.

Component packages are useful if you work in an environment where concurrent access to catalogs is not feasible.

When you use component packages to copy components between catalogs, the components are saved in a non-editable .pkg file. If you want to copy a catalog component to an editable text file, you should use the CATEXP command to export the catalog, followed by the CATIMP command to import the component into the target catalog. For more information about using these commands, see Chapter 31, “Commands,” on page 377.

Create a Component Package

A component package can contain as many catalog components as you require. Catalog components are connections, builds, reference dimensions, reference structures, JobStreams, templates, functions, metadata dimensions, and metadata collections.

Note: Connections are not always portable between computers, so you must assess the connection before you attempt to move it.

When you create a component package, IBM Cognos Data Manager lists all the dependent components in the catalog which must also be included in the package. However, when you import the component package into the target catalog, you can choose whether to include these dependent components, or use existing components in the target catalog.

Procedure

1. From the **File** menu, click **Create Package**.
2. In the **Available components** pane, select the check box for the components to add to the package.

   The selected components and all their dependent components are shown in the **Components in package** pane.

   - When you add a user-defined folder to a package, Data Manager automatically adds all the builds and JobStreams contained in that folder. If you do not want to import the whole folder, clear the check boxes for the unwanted builds or JobStreams.
   - If you add a user-defined folder to a package, only the contents of the folder are shown in the **Components in package** pane. Any builds and JobStreams contained in a user-defined folder are shown as a path string showing the folder name followed by the build or JobStream name. For example, if user-defined Folder1 contains JobStreamA, it appears in the **Components in package** pane as Folder1\JobStreamA.

   Tip: To remove a component from the **Components in package** pane, clear its check box. Data Manager also removes any dependents of that component.
3. In the **Description** box, enter a description of the package so that when the package is imported, it can be easily identified.
4. Click **OK**.

   The **Package File** dialog box appears.
5. In the **File name** box, type a name for the package file.
6. In the **Save as type** box, click **Package Files (*.pkg)**.
7. Click **Save**.

Import a Component Package

You can import components from a component package (*.pkg) file to partially restore a catalog or integrate work into a test or production catalog. You can also import components from a catalog backup (*.ctg) file using this method.

Note: You cannot import an IBM Cognos Data Manager package in IBM Cognos DecisionStream Series 7 or earlier.

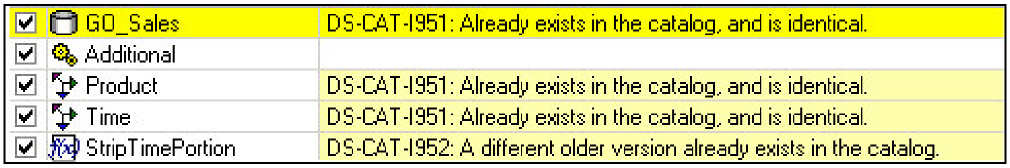
Before you import a component package, it is recommended that you create a backup copy of the current catalog. For information, see “Backup an IBM Cognos Data Manager Catalog” on page 327.

To import a component package you use the Import Package wizard.

Review Information Messages

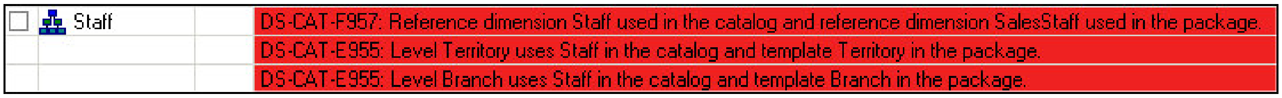
When you select the components to import, each component in the package is compared with the version in the target catalog. For each component that already exists in the target catalog, IBM Cognos Data Manager provides messages to help you decide whether you want to import them. There are two types of messages, information messages and warning messages.

A component may have information messages if, for example, an identical component exists in the target catalog, or if there is a different, newer version of the component in the target catalog. If you choose to import a component with an information message, as you progress through the Import Package wizard, Data Manager gives you the option of renaming the component. Information messages are shown with a yellow background.



☑	GO_Sales	DS-CAT-I951: Already exists in the catalog, and is identical.
☑	Additional	
☑	Product	DS-CAT-I951: Already exists in the catalog, and is identical.
☑	Time	DS-CAT-I951: Already exists in the catalog, and is identical.
☑	StripTimePortion	DS-CAT-I952: A different older version already exists in the catalog.

Warning messages are shown if a component is different from the component with the same name in the target catalog. For example, if a hierarchy uses a different template, or if a template has different attributes. If you choose to import a component with a warning message, as you progress through the Import Package wizard Data Manager forces you to rename it. Warning messages are shown with a red background.



☐	Staff	DS-CAT-F957: Reference dimension Staff used in the catalog and reference dimension SalesStaff used in the package.
		DS-CAT-E955: Level Territory uses Staff in the catalog and template Territory in the package.
		DS-CAT-E955: Level Branch uses Staff in the catalog and template Branch in the package.

Tip: You can copy all the component names, together with any associated messages, to the clipboard. You can then paste them into another application and print them. To do this, click **Copy to Clipboard**.

Dependent Components

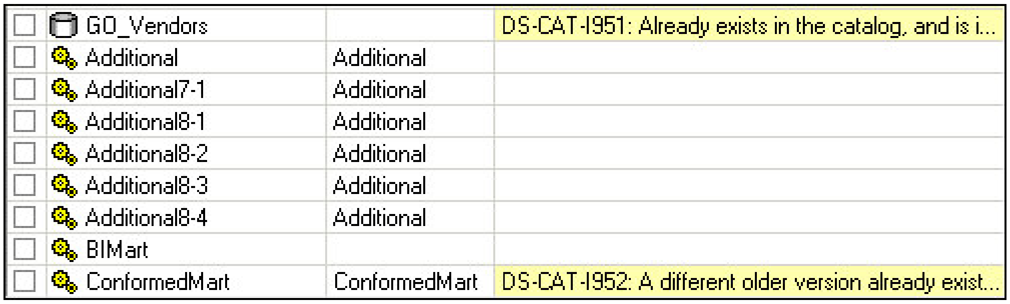
If you select a component to import that depends on unselected components, the Dependent Components window appears. Dependent components are separated into two groups, those that you must import and those that are optional.

IBM Cognos Data Manager provides an information message or a warning message for each of these dependent components.

If you want to remove a component from the import, clear the check box next to the component name. If a component is selected for removal and it has dependent components, the Dependent Components window appears listing all the components that must also be removed.

User-Defined Folders

If a package includes user-defined folders, for each component that belongs in the folder, the folder name appears in the Path column. By default, if you import a component that belongs in a user-defined folder, the folder is also imported.



☐	GO_Vendors		DS-CAT-I951: Already exists in the catalog, and is i...
☐	Additional	Additional	
☐	Additional7-1	Additional	
☐	Additional8-1	Additional	
☐	Additional8-2	Additional	
☐	Additional8-3	Additional	
☐	Additional8-4	Additional	
☐	BIMart		
☐	ConformedMart	ConformedMart	DS-CAT-I952: A different older version already exist...

If you want to import components without their associated user-defined folders, select the **Ignore folders** check box. If a component already exists in the target catalog, IBM Cognos Data Manager imports in it into its current location in the catalog. If it does not exist in the target catalog, Data Manager imports it into the Builds and JobStreams folder.

Rename Components

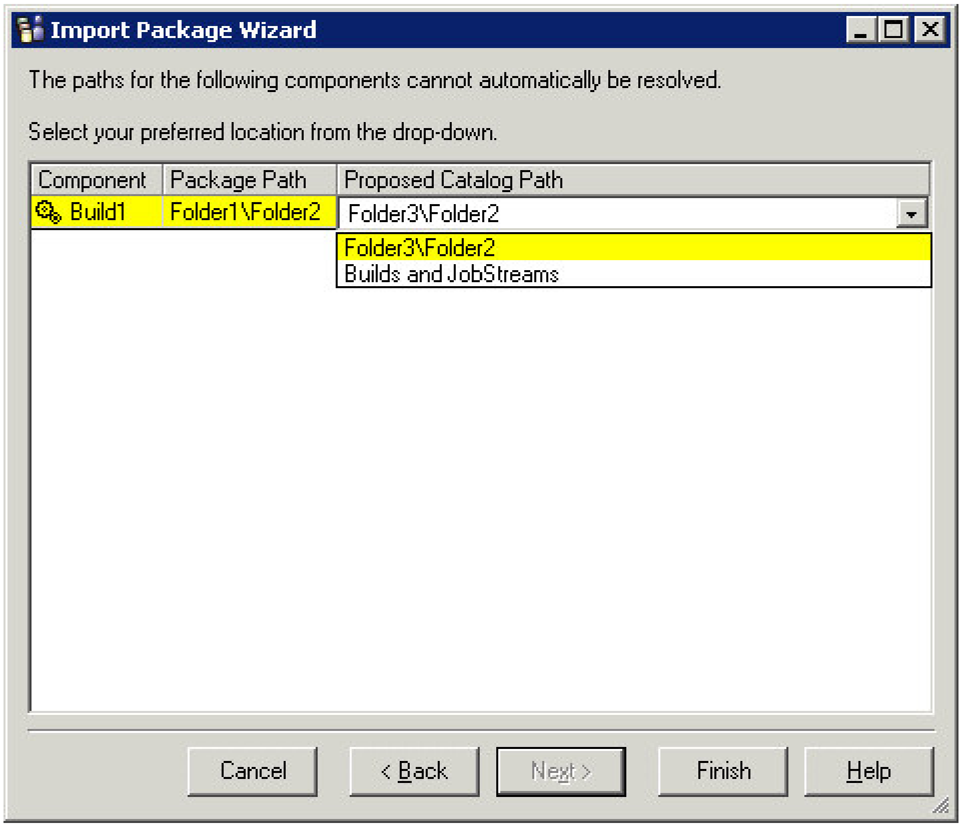
As you progress through the Import Package wizard, any components that you chose to import that had an information message are listed and you are given the option of renaming them. Any components that had a warning message are automatically renamed by IBM Cognos Data Manager, although you can change this name to something more appropriate if you require.

For those components for which renaming is optional, you can type a name for each component yourself, or click **Auto** for Data Manager to provide a default name for them all. If you do not want to rename these components, when the package is imported the component in the package overwrites that in the target catalog.

Note: User-defined folders do not appear in the list because you cannot rename them.

Components With Conflicting Paths

If a component belongs to a user-defined folder, say Folder1\Folder2 and the same component already exists in the target catalog in a different user-defined folder, say Folder3\Folder2, IBM Cognos Data Manager provides a list of all the possible paths you can use to import the component. Select the required path from the Proposed Catalog Path list.



Import Package Wizard

The paths for the following components cannot automatically be resolved.

Select your preferred location from the drop-down.

Component	Package Path	Proposed Catalog Path
Build1	Folder1\Folder2	Folder3\Folder2

Folder3\Folder2

Builds and JobStreams

Note: If you select Builds and JobStreams, Data Manager imports the component into the Builds and JobStreams folder, rather than a user-defined folder.

Source Code Control Information

If source code control is being used and there are components you have selected for import that are not currently checked out, IBM Cognos Data Manager provides a list of these components. If you choose to progress through the Import Package wizard, Data Manager prompts you to check out the components so that you can complete the package import.

Note: You can obtain more source code control information about a listed component by clicking the Details button.

For more information, see "Source Code Control" on page 315.

Use the Import Package Wizard

This section describes how to use the Import Package wizard.

Procedure

1. Open the target catalog.
2. From the **File** menu, click **Import Package**.

   A message appears prompting you to backup the current catalog. Click **Yes** to backup the current catalog.

   The **Package File** dialog box appears.
3. Click the component package to import, and then click **Open**.

The **Import Package Wizard** appears, with a full list of components included in the package.

4. If there are information messages shown for any listed components, ensure you review them before proceeding with the import.

   For more information, see "Review Information Messages" on page 323.
5. For each component you want to import, select the check box next to the component name.
   - To include all listed components, select the check box at the top of the column.
   - To hide unselected components from view, select the **Show selected only** check box.

   If there are dependent components, the **Dependent Components** window appears. For more information, see "Dependent Components" on page 323.
6. If you do not want to import components with their associated user-defined folders, select the **Ignore folders** check box.

   For more information, see "User-Defined Folders" on page 324.
7. Click **Next**.
8. Rename the components as required.

   For more information, see "Rename Components" on page 324.
9. If you are including user-defined folders, and there is a component with a conflicting path, select the required folder to use for import.

   For more information, see "Components With Conflicting Paths" on page 324.
10. Click **Finish**.

    Note: If source code control is being used and there are components that are not checked out, you must click **Next** to show details of those components and complete the package import. For more information, see "Source Code Control Information" on page 325.

Combine the Effort of Several Developers

The recommended method of combining the efforts of several developers is for the developers to have personal catalogs and to use component packages to transfer components between these catalogs and a master catalog. This method is useful when you do not want to set up a source code control environment. It can also be used when developers are working in a very disconnected way, for example, exchanging components via email.

Combining developer catalogs consists of these main steps:

1. Create the master and developer catalogs.
2. In the master catalog, create any components that are common to all catalogs, and then create a component package that contains all these components.
3. Integrate the catalogs by
   - importing into each developer catalog, all components of the master component package
   - reconfiguring the connections if required, to suit each developer catalog
4. Develop the catalog components as required, ensuring that
   - no two developers modify the same component
   - no common components are modified except for reconfiguration of connections to local or development databases

5. In the developer catalog, create a component package that contains all the components to integrate.
6. Import the required components from the component package into the master catalog, taking care not to import common components.

Backup and Restore Catalogs

You can generate a text backup of a complete IBM Cognos Data Manager catalog or catalog tables, including all the catalog components. This is useful for backup or change control as the text file can be stored in a file management system. It is also useful if you want to move from one DBMS environment to another.

Note: Audit tables and delivery history are not included in a backup.

If you want to create a partial backup, you can select the components that you require and then save them as a package. For information, see "Component Packages" on page 321.

Backup an IBM Cognos Data Manager Catalog

You can backup to an existing file or you can create a file.

Procedure

1. From the **File** menu, click **Backup Catalog**.
2. Select the location of the existing file or where the new file is to be stored, and then either type a name for the backup file in the **File name** box, or select an existing file.
3. Click **Save**.
4. If you selected an existing file, a message appears stating that the file already exists. Click **Yes** to replace the file.

   IBM Cognos Data Manager copies the data to the selected file.

Restore an IBM Cognos Data Manager Catalog

You may need to restore a catalog to a previous state using a backup text file. In IBM Cognos Data Manager, backup files are usually (*.ctg) files. You can also restore a component package (*.pkg) file using this method.

Before you restore a catalog, it is recommended that you create a backup copy of the current catalog. This is because Data Manager deletes all data from the current catalog before performing the restoration. If you do not create a backup copy, and the restore file is invalid in any way, your current catalog is lost.

For information, see "Backup an IBM Cognos Data Manager Catalog."

Note: You cannot restore a Data Manager catalog in IBM Cognos DecisionStream Series 7 or earlier.

Procedure

1. Open the required catalog.
2. Create a backup copy of the catalog.
3. From the **File** menu, click **Restore Catalog**.

   A warning message appears informing you that restoring the catalog will overwrite the current catalog.

4. Click **Yes**.
5. Locate the backup file that you want to restore.

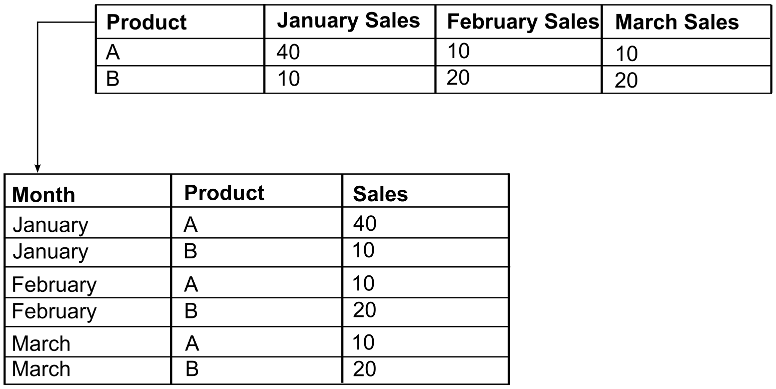
   Important: When you click **Open** in this dialog box, Data Manager deletes all data from the current catalog. If you have not created a backup copy, you may want to click **Cancel** and begin again.
6. Click **Open**.

   Data Manager deletes all the data from the current catalog, and populates the catalog with the selected backup file.

Chapter 27. Using Pivots to Rotate Data

Data pivoting is a technique that treats multiple table columns as though they were a single column with multiple values. The term pivot is used because specified table columns rotate through 90 degrees to form rows.

In the following example, two columns have been created, Month and Sales, as a result of pivoting monthly sales figures.



Product	January Sales	February Sales	March Sales
A	40	10	10
B	10	20	20

Month	Product	Sales
January	A	40
January	B	10
February	A	10
February	B	20
March	A	10
March	B	20

Create Pivots

A typical pivot rotates two or more data source columns to two DataStream items. One DataStream item maps to a transformation model element that records from which data source column the value originates. The second DataStream item maps to a transformation model element that contains the values from the data source columns.

Note: Pivoting data usually increases the number of data rows that a DataStream outputs. If you specify a row limit, this affects the number of rows output, not the number of source data rows processed by the DataStream. For information, see "Apply a Maximum Row Limit" on page 131.

Procedure

1. In the fact build for which you want to pivot data, click **DataStream**.
2. From the **Edit** menu, click **Properties** .
3. Click the **DataStream Items** tab, and then click **Add** to create a DataStream item.
4. In the **Name** box, type a name for the DataStream item.
5. Click the **Pivot Values** tab.
6. Click **Import** if the pivot values you want to define for the DataStream item already exist as columns or values in a database table.

   A menu appears prompting you to choose the location from which to import pivot values.
7. Click

- **Add Columns from Data Source** or **Add Columns from Table** to import database column names. From the window that appears, click the column names to import as pivot values, and then click **OK**.
- **Add Values from Column** to import the values stored within a database column. From the window that appears, click the column containing the values to import as pivot values, and then click **OK**.

If you do not want to import pivot values, or you want to add further values, you can define them manually by clicking **Add** and then typing a name for the pivot value.

To reorder the list, click the pivot value that you want to move, and then click **Move Up** or **Move Down** as required.

8. When you have created a pivot value for each data source column to be pivoted, click **OK**.
9. Click the **DataStream Items** tab and expand the new DataStream item to display its pivot values.
10. Map each data source column to the appropriate pivot value and to any other DataStream items as required.

    For more information, see "Map Data Source Columns to the DataStream" on page 125.
11. Click **OK**.

    Tip: Before mapping to the transformation model, you can check that the data is pivoted correctly by executing the DataStream.

    For more information, see "Execute a DataStream" on page 133.
12. Click the transformation model for the fact build.
13. From the **Edit** menu, click **Mapping**.

    The **Transformation Model** window appears.
14. Create the required transformation model elements, and then map the DataStream items to them.

    For more information, see "Map the DataStream to the Transformation Model" on page 175.
15. Click **OK**.

Examples

The following examples illustrate the different ways in which you can pivot data:

- create a single pivot
- create a double pivot
- create multiple pivot values

These examples are located in the DS_Advanced sample catalog.

Example 1: Create a Single Pivot

You can work through this example using the Pivot1 fact build in the DS_Advanced catalog.

This table records the actual sales and forecast sales data for the products that a company sells.

Product	Actual Sales	Forecast Sales
Aloe Relief	598	701
Bear Edge	1616	1739
Bear Survival Edge	1184	1118
Blue Steel Max Putter	118	114
Blue Steel Putter	352	334

You can pivot the data from the Actual Sales and Forecast Sales columns to two columns, named Scenario and Sales.

Create the DataStream items and specify pivot values

This is the first part of Example 1: Create a single pivot.

Procedure

1. Create a fact build named **Pivot1** and set up a data source that accesses the Pivot1 data.
2. In the Pivot1 fact build, click the DataStream.
3. From the **Edit** menu, click **Properties** .
4. Click the **DataStream Items** tab, and then click **Add**.
5. In the **Name** box, type **Scenario**.
6. Click the **Pivot Values** tab.
7. Type the pivot values **ACTUAL** and **FORECAST**.

   These values correspond to the Actual Sales and Forecast Sales data source columns respectively.
8. Click **OK**.
9. Click the **DataStream Items** tab, and then click **Add**.
10. In the **Name** box, type **Sales**.
11. Click **OK**.

    Leave the **DataStream Properties** window open so that you can map the data source columns.

Map the data source columns

This is the second part of Example 1: Create a single pivot.

Procedure

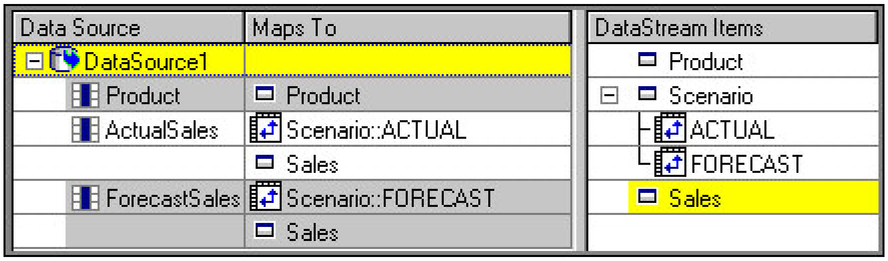
1. In the **DataStream Properties** window, click the **DataStream Items** tab.
2. Drag the Product data source column from the left pane to the white space in the **DataStream Items** pane to create and map and DataStream item for the column.
3. In the **DataStream Items** pane, expand Scenario and then drag the following pivot values to the required data source column in the left pane.

Pivot value	Data Source column
Scenario:ACTUAL	ActualSales
Scenario:FORECAST	ForecastSales

4. Holding down the Ctrl key, drag the following DataStream items from the **DataStream Items** pane to the required data source column in the left pane.

DataStream item	Data Source column
Sales	ActualSales
Sales	ForecastSales

The **DataStream Properties** window should look like this.



Data Source	Maps To
⊟ DataSource1	
Product	Product
ActualSales	Scenario::ACTUAL
	Sales
ForecastSales	Scenario::FORECAST
	Sales

DataStream Items
Product
⊟ Scenario
ACTUAL
FORECAST
Sales

5. Click **OK**.

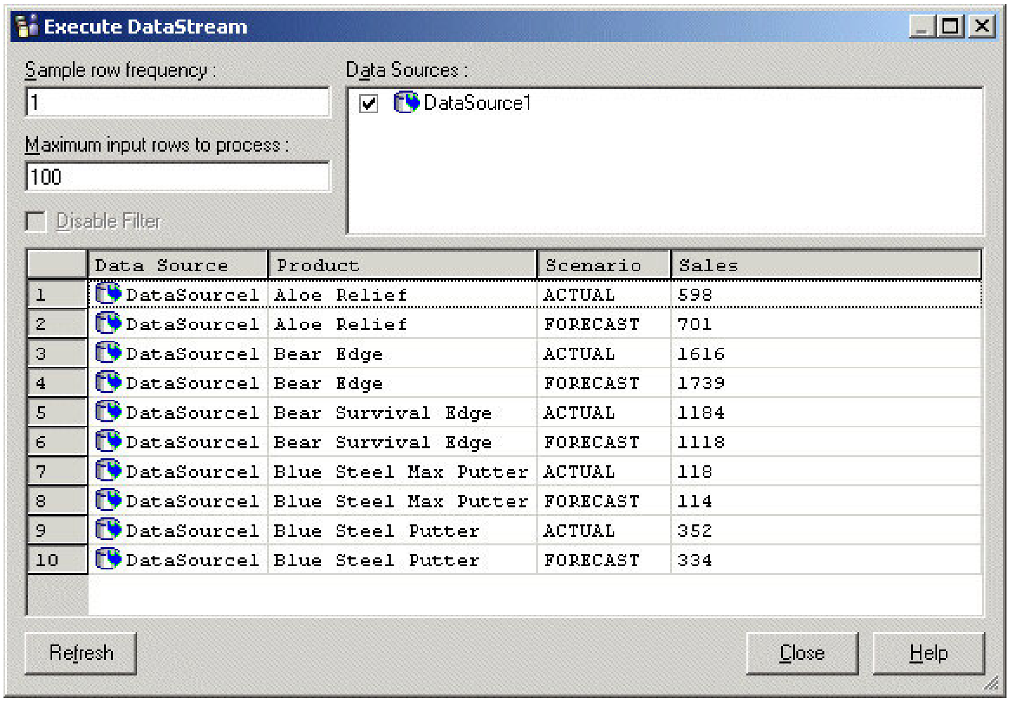
Execute the DataStream to test the pivoted data

This is the third part of Example 1: Create a single pivot.

Procedure

1. In the Pivot1 fact build, click the DataStream.
2. From the **Actions** menu, click **Execute** ▶.

   The **Execute DataStream** window appears with the results of the pivoted data.



Sample row frequency : 1

Maximum input rows to process : 100

□ Disable Filter

Data Sources : ☑ DataSource1

	Data Source	Product	Scenario	Sales
1	DataSource1	Aloe Relief	ACTUAL	598
2	DataSource1	Aloe Relief	FORECAST	701
3	DataSource1	Bear Edge	ACTUAL	1616
4	DataSource1	Bear Edge	FORECAST	1739
5	DataSource1	Bear Survival Edge	ACTUAL	1184
6	DataSource1	Bear Survival Edge	FORECAST	1118
7	DataSource1	Blue Steel Max Putter	ACTUAL	118
8	DataSource1	Blue Steel Max Putter	FORECAST	114
9	DataSource1	Blue Steel Putter	ACTUAL	352
10	DataSource1	Blue Steel Putter	FORECAST	334

The Scenario column records from which data source column each value in the Sales column originates. Each row from the source table becomes two rows in the DataStream.

Map the DataStream items to the transformation model

This is the fourth part of Example 1: Create a single pivot.

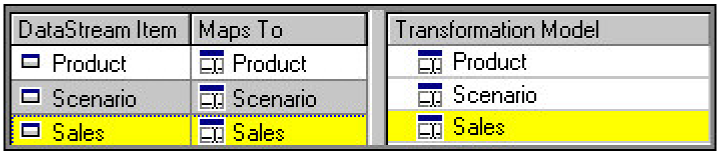
Procedure

1. Click **Transformation Model**.
2. From the **Edit** menu, click **Mapping**.

   The **Transformation Model** window appears.
3. In the left pane, click the Product, Scenario, and Sales DataStream items.
4. Drag the selected items from the left pane to the white space in the **Transformation Model** pane.
5. When prompted, click **Create New as Attribute**.

   The selected DataStream items are mapped to the new transformation model elements.

   The **Transformation Model** window should look like this.



DataStream Item	Maps To
Product	Product
Scenario	Scenario
Sales	Sales

Transformation Model
Product
Scenario
Sales

6. Click **OK**.
7. For each listed element, select the **Add to Fact Deliveries** check box.
8. Click **OK**.

Example 2: Create a Double Pivot

You can work through this example using the Pivot2 fact build in the DS_Advanced catalog.

You can pivot more than one data source within a DataStream. In this example, there are four data source columns named Actual Sales, Actual Revenue, Forecast Sales, and Forecast Revenue and they pivot to the single column named Amount. The intersection of the Scenario and Measure columns records which data source column contained each value.

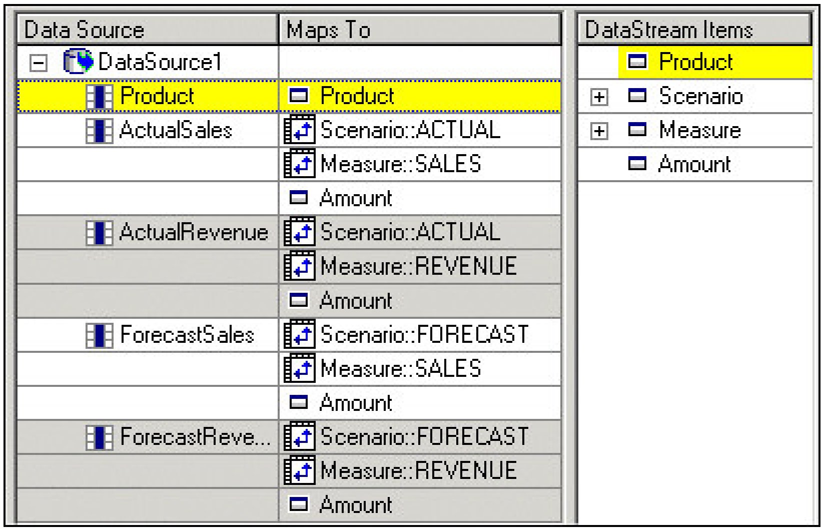
Product	**Actual Sales**	**Actual Revenue**	**Forecast Sales**	**Forecast Revenue**
Aloe Relief	598	2912	701	3407
Bear Edge	1616	60525	1739	66992
Bear Survival Edge	1184	94238	1118	90601

To define this double pivot, you must create three DataStream items, Scenario, Measure, and Amount.

The Scenario DataStream item has two pivot values, and records whether each value in the Amount column is actual data or forecast data.

The Measure DataStream item also has two pivot values and records whether each value in the Amount column relates to sales or revenue.

On the DataStream Items tab of the DataStream Properties window, the pivot looks like this.

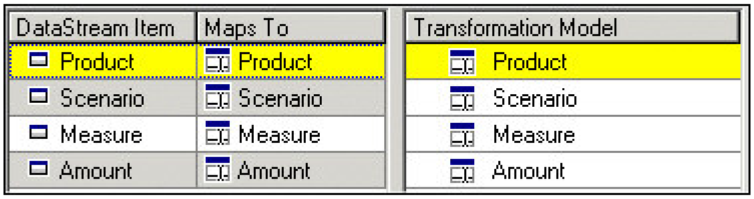


Data Source	Maps To
⊟ DataSource1	
Product	Product
ActualSales	Scenario::ACTUAL
	Measure::SALES
	Amount
ActualRevenue	Scenario::ACTUAL
	Measure::REVENUE
	Amount
ForecastSales	Scenario::FORECAST
	Measure::SALES
	Amount
ForecastReve...	Scenario::FORECAST
	Measure::REVENUE
	Amount

DataStream Items
Product
⊞ Scenario
⊞ Measure
Amount

You must then map the DataStream items to four transformation model elements.

The Transformation Model window, should look like this.



DataStream Item	Maps To
Product	Product
Scenario	Scenario
Measure	Measure
Amount	Amount

Transformation Model
Product
Scenario
Measure
Amount

Simultaneously performing two pivots results in this table.

Product	Scenario	Measure	Amount
Aloe Relief	ACTUAL	SALES	598
Aloe Relief	FORECAST	SALES	701
Aloe Relief	ACTUAL	REVENUE	2912
Aloe Relief	FORECAST	REVENUE	3407
Bear Edge	ACTUAL	SALES	1616
Bear Edge	FORECAST	SALES	1739
Bear Edge	ACTUAL	REVENUE	60525
Bear Edge	FORECAST	REVENUE	66992
Bear Survival Edge	ACTUAL	SALES	1184
Bear Survival Edge	FORECAST	SALES	1118
Bear Survival Edge	ACTUAL	REVENUE	94238
Bear Survival Edge	FORECAST	REVENUE	90601

Example 3: Create Multiple Pivot Values

You can work through this example using the Pivot3 fact build in the DS_Advanced catalog.

No limit exists on the number of pivot values that you can define for a DataStream item. However, when you define a data source pivot, you must declare all the values that can exist in the corresponding DataStream item.

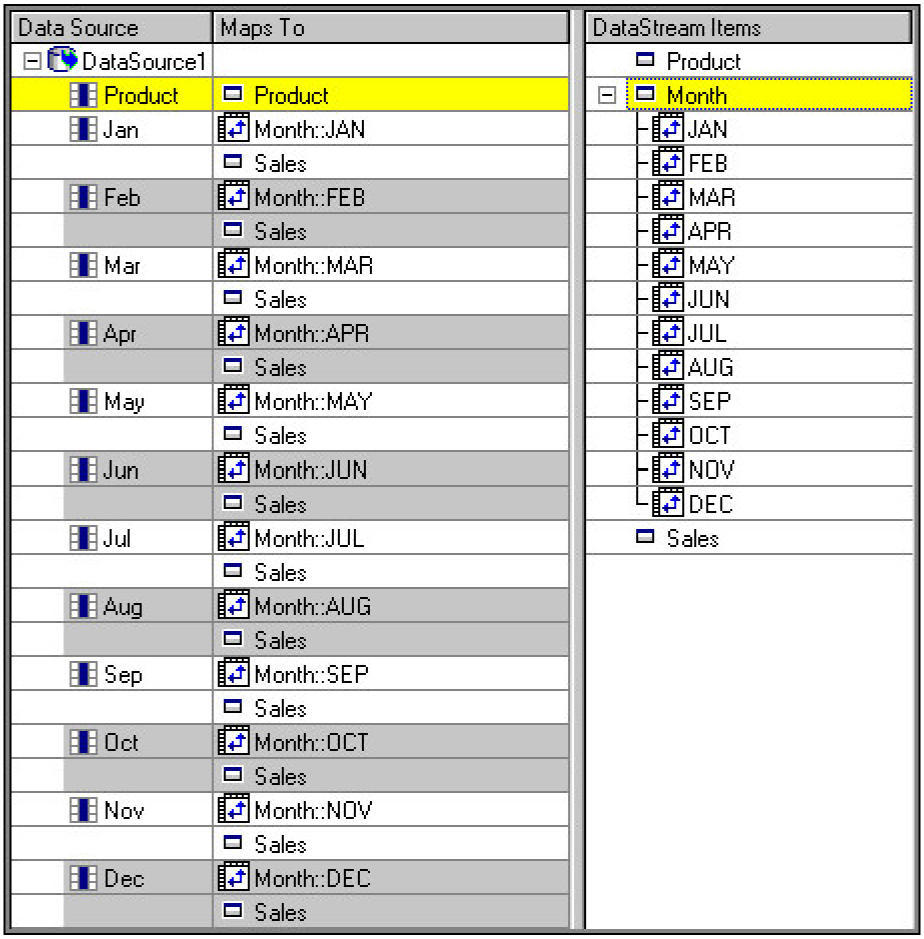
For example, the following table shows sales by month for each product.

Prod	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Star Dome	2.0	1.76	4.17	5.24	3.53	4.81	4.29	3.40	3.96	9.38	2.81	2.68

To pivot this information to a table that has one column for the name of the month and another to record the sales value, you must create three DataStream items, Product, Month, and Sales.

The Month DataStream item has twelve pivot values, one for each month.

On the DataStream Items tab of the DataStream Properties window, the pivot looks like this.

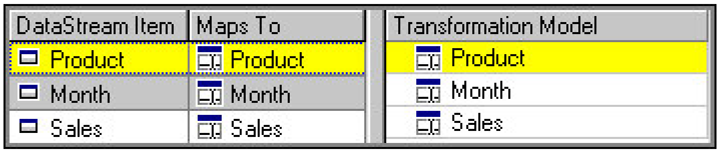


Data Source	Maps To
DataSource1	
Product	Product
Jan	Month::JAN
	Sales
Feb	Month::FEB
	Sales
Mar	Month::MAR
	Sales
Apr	Month::APR
	Sales
May	Month::MAY
	Sales
Jun	Month::JUN
	Sales
Jul	Month::JUL
	Sales
Aug	Month::AUG
	Sales
Sep	Month::SEP
	Sales
Oct	Month::OCT
	Sales
Nov	Month::NOV
	Sales
Dec	Month::DEC
	Sales

DataStream Items
Product
Month
JAN
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
Sales

You must then map the DataStream items to three transformation model elements.

The Transformation Model window, should look like this.



DataStream Item	Maps To
Product	Product
Month	Month
Sales	Sales

Transformation Model
Product
Month
Sales

This table shows the result of pivoting the original data to create Month and Sales columns.

Product	**Month**	**Sales**
Star Dome	Jan	2.20
Star Dome	Feb	1.76
Star Dome	Mar	4.17
Star Dome	Apr	5.24
Star Dome	May	3.53
Star Dome	Jun	4.81
Star Dome	Jul	4.29
Star Dome	Aug	3.40
Star Dome	Sep	3.96
Star Dome	Oct	9.38
Star Dome	Nov	2.81
Star Dome	Dec	2.68

Chapter 28. Handling Data Rejected by a Fact Build

IBM Cognos Data Manager can reject data that it is acquiring or delivering. You control how Data Manager handles this rejected data.

You can specify that rejected source data is saved to a text file and a SQLTXT definition file, or a relational table. You can also choose to send rejected fact data to a fact delivery where it can be saved.

Rejected Source Data

During data acquisition, IBM Cognos Data Manager can reject a row of source data if either of these is true:

- the data row duplicates the combination of dimension values of a data row acquired earlier during data acquisition.
- the value of a dimension element does not match a member of the corresponding hierarchy.

Reject Duplicate Source Data

Duplicate data source rows have identical dimension elements. The first data row is accepted and any subsequent duplicate data rows are rejected.

You can save the rejected data rows, edit them, and then execute the data rows again. You can treat this file as a data source using IBM Cognos Data Manager SQLTXT Designer.

Note: You can also accept or merge duplicate source data rows. For information see “Handle Duplicate Data” on page 112.

Procedure

1. Click the required fact build.
2. From the **Edit** menu, click **Properties** .
3. Click the **Input** tab.
4. Select **Reject records with duplicate keys**.
5. If you want to change the default name and location of the reject file, in the **File name** box, enter the full directory path and file name that you require.

   Tip: Click the browse button to open the **Select Reject File** dialog box where you select the required file.
6. Click **OK**.

   The Visualization pane is refreshed to show graphically that duplicate records are to be rejected.
7. Manually edit the data in the reject file to eliminate the problem that caused the data to be rejected, and then execute the build again. You can do this either by editing the reject file directly, or by using SQLTerm.

   You can use the Data Manager SQLTXT DBMS driver to further process the rejected data.

Reject Unmatched Members

A dimension element is rejected in the transaction data if it is associated with a hierarchy or lookup, but its ID cannot be found in the corresponding reference structure. You can stop this automatic rejection in one of two ways. You can

- map the dimension element to the hierarchy or lookup

  For information, see "Associate a Dimension or Derived Dimension Element with a Reference Item" on page 146.
- specify that you want to include the unmatched member

  For information, see "Accept Source Data Containing Unmatched Dimension Values" on page 154.

Saving Rejected Source Data

When IBM Cognos Data Manager rejects records during build execution, the rejected data can be saved to a file or a table.

If you do not want to save rejected records, click **Don't write reject records**.

Save rejected data to a file

The first time a build that creates reject records is executed, Data Manager creates a simple, tab-delimited text file. If the build has been executed previously, the existing data in the text file is replaced.

By default, the text file is named <build_name>.rej and is stored in the Data Manager Data directory.

Saving reject records to file is the default.

Procedure

1. Click the required fact build.
2. From the **Edit** menu, click **Properties** .
3. Click the **Input** tab.
4. In the **Reject Record Handling** box, click **Write reject records to file**.
5. If you want to change the default name and location of the reject file, in the **File name** box, enter the full directory path and file name.

   Tip: Click the browse button to open the **Select Reject File** dialog box where you select the required file.
6. Click **OK**.

Save rejected data to a table

This table can reside in any writeable Data Manager connection.

Procedure

1. Click the required fact build.
2. From the **Edit** menu, click **Properties** .
3. Click the **Input** tab.
4. In the **Reject Record Handling** box, click **Write reject records to table**.
5. In the **Connection** box, click the connection that contains the table where the rejected records are to be saved.

Tip: To create a connection, click **New**. For information on creating a connection, see “Create a Database Connection” on page 39.

6. In the **Table name** box, enter the name of the table to which you want the rejected data to be saved.

   Tip: You can also click the **Browse for table** button and then click the required table from the list in the **Select Table** dialog box.
7. Click **OK**.

Rejected Delivery Data

During data delivery, a row of fact data can be rejected if either of these is true

- the data row does not match any level filter
- the data row evaluates to FALSE in an output filter

Primarily, filters are used to restrict the fact data being delivered. However, they can also be used to reject unwanted data.

Use a Level Filter to Reject Data

You can use level filters to configure a delivery to accept only specific dimension and hierarchy levels. Data is rejected unless its dimension values match at least one of its level filters.

For more information, see “Use a Level Filter to Deliver Specific Dimensions and Hierarchy Levels” on page 190.

Use an Output Filter to Reject Data

An output filter rejects all rows for which the expression evaluates to FALSE. Each delivery may have no more than one output filter.

You can also use an output filter together with a derivation to reject data. For example, you could create a derivation that calculates profit margin. If the profit margin is less than 50%, IBM Cognos Data Manager sets the field to "<50%", if it is over 50% to ">50%", and if the margin is negative, to "Negative Margin". You can then create an output filter that has an expression that evaluates "Negative Margin" to FALSE and so rejects the data.

For more information, see “Use an Output Filter to Deliver Specific Data Rows” on page 192, and “Derivations” on page 128.

Save Data Rejected During Delivery

To save rejected data on delivery, create a fact delivery to send the rejected data to a suitable database in any supported DBMS.

Procedure

1. Determine an expression that evaluates to TRUE only for the rejected data. If necessary, you can create a derivation transformation model element against which to apply the filter expression.
2. Create a fact delivery to save the unwanted data. This fact delivery must be the topmost fact delivery in the Build tree.
3. Click the delivery, and from the **Edit** menu, click **Properties** .
4. Create an output filter that uses the expression you determined in step 1.

See “Use an Output Filter to Deliver Specific Data Rows” on page 192.

5. To ensure that subsequent deliveries are not offered the same data rows, click the **General** tab and then select the **Exclusive** check box.
6. Execute the build.
7. To prepare the rejected data for reprocessing, edit it in the DBMS to which it was delivered or in SQLTerm.

Specify the Encoding to Use for Reject Files

If you are using IBM Cognos Data Manager with multilingual or unicode data, you should specify unicode encoding for your reject files. By default, the encoding used is platform dependent.

You can specify the encoding using one of these methods:

- enter a command specifying the encoding to use in the Execute Build or Execute JobStream dialog box
- set an environment variable
- specify the encoding in the dm.ini file

For information on using the last two methods, see the IBM Cognos Data Manager *Installation and Configuration Guide*.

Procedure

1. Select the build or JobStream for which you want to specify the encoding.
2. From the **Actions** menu, click **Execute**.
3. In the **Additional options** box, use the following syntax to specify the file name (or variable) and the encoding to use:

   <filename|variable>!<encoding>

   For example

   ```
   -VREJECT_FILE={$DS_BUILD_FNAME}.rej!UTF-16
   ```

Chapter 29. Managing Memory

A fact build that merges or aggregates high volumes of data, and contains a significant number of objects, such as attributes, transformation model elements, and so on, may use a large amount of memory.

When performing aggregation, each accepted (input) row from the DataStream may contribute to many output rows for potential delivery. The maximum number of output rows is the product of the number of output levels of each hierarchy upon which aggregation occurs. For example, if a build uses seven dimensions, and each dimension uses a hierarchy with five output levels, then the maximum number of output rows is 5 x 5 x 5 x 5 x 5 x 5 x 5, or 78,125 per input row.

To better manage memory, you can limit the memory IBM Cognos Data Manager uses during fact build execution by using dimension breaking or by setting a memory limit.

Using dimension breaking allows Data Manager to clear the memory of all data associated with dimension values that will not reoccur in the input stream. Setting a memory limit within Data Manager prevents an operating system from terminating an application that requests more memory than the operating system can provide. It also ensures that Data Manager uses its own paging algorithm rather than relying on the operating system to provide this function.

Data Manager allocates memory for several purposes.

Structure Data

IBM Cognos Data Manager stores all the reference structure data, and caches it at the start of build execution. When built, this structure is static and is not subject to application paging.

You can reduce the memory used by caching reference data attributes in a temporary file rather than memory. You do this from the **Lookup Properties** dialog box by selecting the **Use file caching for attributes** box.

Dimension Domains

There are two domain types, dynamic and reference. When using a dynamic domain, new members are added as their values are detected in the transactional input stream. When using a reference domain, all the members from the associated reference structure, irrespective of whether they are referenced by the transactional input stream, are loaded into memory.

For more information, see "Dynamic and Reference Domains" on page 154.

The Hash Table

The hash table is an internal IBM Cognos Data Manager structure that is used to process and transform rows of data. Data Manager normally uses hashing to track memory required for aggregation and to merge duplicate rows.

The Page Pool

The page pool contains fixed-size data pages in which IBM Cognos Data Manager stores aggregated data and the associated hash table for indexing the data.

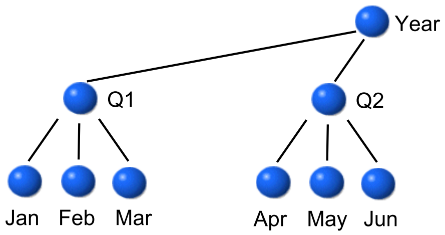
The Page Table

The page table provides IBM Cognos Data Manager with location and content details for all the data and hash pages currently in use.

Dimension Breaks

During data acquisition, partially built aggregates and all data that may contribute to future aggregations are stored. Breaking is the process of tracking changes in ordered dimensions to reduce memory usage. When a change is detected in a sorted input stream, all aggregates can be cleared from memory, releasing the memory that the data occupied.

For example, the following hierarchy contains year, quarter, and month levels.



When all the records for March have been processed, the memory is cleared of the March records, the hierarchy is checked, and all the records for Q1 are flushed. Processing continues until all the records for June have been processed, when memory is cleared of all the records for Q2 and Year.

In another example, suppose the acquired data is ordered on a City dimension, and data rows that have a city value of London are being processed. The first row to arrive with a city value other than London implies that all the data rows that have the value London have been received.

You can perform breaking on a single dimension or, more usually, on a combination of dimensions.

Note: You cannot perform breaking on a derived dimension or data aggregated using aggregate rules.

Breaking requires that the data is ordered consistently across all data sources that are being merged. When data is sourced from more than one data source and the databases are not using the same collation sequence, problems can occur. You must therefore ensure that a compatible collation sequence is used to order the data for each data source. For more information see, “Distributed Sorting” on page 40.

Determine the Dimensions on which to Break

Determining the dimensions on which to break requires a mixture of analysis, experience, and the evaluation of different options.

A dimension that is suitable for breaking

- is not aggregated, or is aggregated through few levels, with relatively few aggregated members generated
- has a large domain that can be broken into smaller segments
- is a balanced hierarchy, having an even distribution of members within parents

Set Dimension Breaks

When you have determined the dimensions on which to break, you are ready to set the dimension breaks and specify their behavior.

Procedure

1. Click the required fact build.
2. From the **Edit** menu, click **Properties** .
3. Click the **Breaks** tab.
4. Move the dimensions that you want to use for breaking from the **Dimensions** pane to the **Break On** pane.

   Tip: In addition to using the arrow buttons, you can also double-click a dimension to move it to the **Break On** pane.
5. Ensure that the order of the dimensions in the **Break On** pane is the order in which breaking is to be performed. You can change this order using the up and down arrow buttons.
6. Click **OK**.

Specify that Sorting is to be Performed

Breaking can only function correctly if each data source is sorted by the same set of dimensions, in the same sequence, and is of the equivalent character sets. Sorting is often performed in the database, for example, indexing often orders the data.

However, you can specify that sorting is performed by adding an ORDER BY clause to the source SQL or by selecting the Force Sort on Break Dimensions check box in the Fact Build Properties window.

An error occurs if the database does not support ORDER BY clauses, or if the SELECT statement is made invalid by adding an ORDER BY clause. For example, if there is already an ORDER BY clause or a FOR READ (db2) clause.

Note: Some data sources such as SQLTXT and Published FM Packages cannot have the sort applied and an error will occur. In these cases, you must manually add the sort to the data source or ensure that the data rows are pre-sorted.

You specify that sorting is to be performed after you set the required dimension breaks.

Procedure

1. Click the required fact build.
2. From the **Edit** menu, click **Properties** .
3. Click the **Breaks** tab.
4. Select the **Force Sort on Break Dimensions** check box.
5. Click **OK**.

Specify When to Break

Data should not necessarily be flushed on every break because identifying the completed row takes time.

You specify when breaking is to occur after you set the required dimension breaks.

Specifying when breaking is to occur can greatly improve break performance, especially when merging or aggregating on dimension elements with very high distinct cardinality. Using this setting, 60% is recommended, can reduce the number of breaks that occur and improve overall performance because memory is utilized more efficiently.

Procedure

1. Click the required fact build.
2. From the **Edit** menu, click **Properties** .
3. Click the **Breaks** tab.
4. Select the **Perform break processing every** check box, and then
   - select **Breaks** to specify a break every n dimension changes
   - select **Percent** to specify that the break is to be based on the percentage of the hash table that is used
5. In the box to the left of the option buttons, type a number to indicate the number of dimension changes or the required percentage (as a number between 1 and 100).
6. Click **OK**.

Processing Data Sources in Series or in Parallel

By default, data sources are read in series unless breaking is used, in which case they are read in parallel.

When data sources are read in parallel, IBM Cognos Data Manager orders the data sources using the order of the dimensions that you specified on the Breaks tab of the Fact Build Properties window. If there are multiple data sources, processing in parallel forces the ordering to apply to all the dimensions. This effects performance, because Data Manager compares the dimension values from the current row of each data source, to ascertain which is the next in order. However, if you know that there is no ordered overlap between the data from all the data sources, then you can avoid this performance overhead by executing the data sources in series. For example, if a FiscalPeriod dimension was first in the break on list and data source one contained data for periods 1995 to 1999 and data source two contained data for periods 2000 onwards, you could use execute in series. If you execute in series and there are overlaps in the data, then you will have duplicate data.

When data sources are processed in series, the order in which they appear in the Tree pane is used. You can change this order if you require. For information, see “Change the order of data sources” on page 132.

Procedure

1. Click the DataStream.
2. From the **Edit** menu, click **Properties** .
3. Click the **Input** tab.

4. Select the **Execute data sources in series** check box.
5. Click **OK**.

Memory Limits

You may want to limit the amount of memory that IBM Cognos Data Manager requests from the operating system for these reasons:

- Some operating systems terminate any application that requests more memory than the operating system can provide.
- If Data Manager must share resources with other applications running on the computer, you may want to prevent Data Manager from requesting memory that the other applications require.

Tips

- To obtain useful information for determining optimum memory settings, execute the build in check only mode.
- To obtain this information, you must specify that Internal details are included in the log file. For information, see “Specify Log File Details” on page 256.

Maximum Memory Usage

You can determine the maximum amount of memory that IBM Cognos Data Manager requested when executing a build by inspecting the log file. Search for the last entry of this format:

```
[INTERNAL - hh:mm:ss]Mem(M): x[Peak=y](details)
```

The value of y (the peak value) gives the maximum amount of memory in MB that Data Manager requested.

If Data Manager reaches the limit of available memory, it creates virtual memory by paging information to disk. You can determine whether this happens by inspecting the log file. Search for entries of this format:

```
[INTERNAL - hh:mm:ss]Paging: n1 pages, n2 in pool, n3 fault(s)
```

The presence of one or more such lines indicates that Data Manager has exhausted available real memory. The value of n3 indicates the number of times it occurred.

For information about how to specify the maximum amount of memory to be used, see “Set Memory Limits” on page 349.

Maximum Page Size

The page size determines how much paging will occur, and how many data rows IBM Cognos Data Manager stores on each page. The default page size is 16 KB. Data Manager adjusts this value (or any specified value) downwards to accommodate a whole number of data rows.

You can determine the page size that Data Manager used for the build by searching the log for an entry of this format:

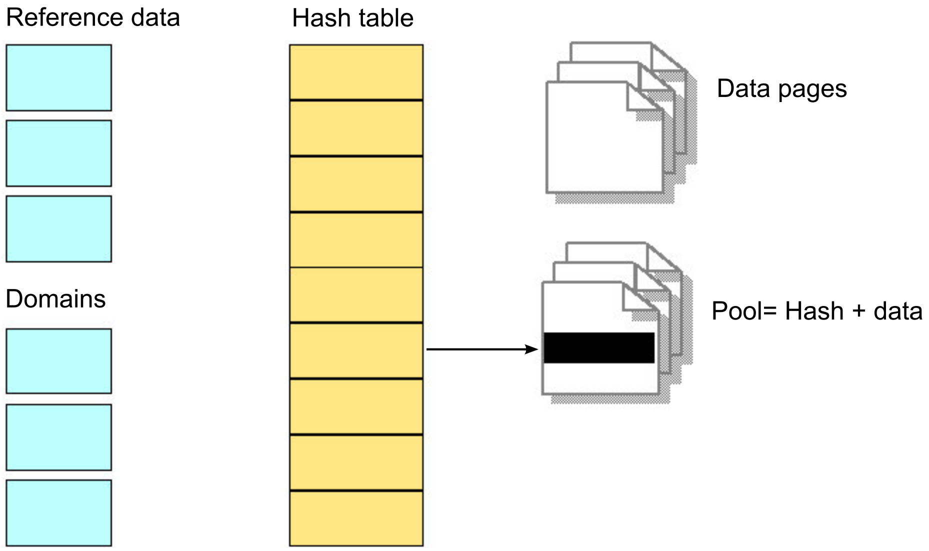
```
[INTERNAL - hh:mm:ss]Paging: page size nn
```

This entry appears near the top of the entries for the execution and expresses the page size in bytes.

For information about specifying the maximum page size, see “Set Memory Limits” on page 349.

Initial Size of the Page Table

By default, IBM Cognos Data Manager determines the initial size of the page table from the size of the hash table.



For information about changing this value, see “Set Memory Limits” on page 349.

Initial Size of the Hash Table

The hash table is an internal IBM Cognos Data Manager structure that is used to process and transform rows of data. Data Manager normally uses hashing to track memory required for aggregation and to merge duplicate rows.

At several times during execution, Data Manager shows statistics about the hash table. The following message indicates that 118 pages are used for the hash slots, 135173 of the 150000 slots of the page are in use, with none marked as reusable.

```
[INTERNAL - 11:14:05]Hash: 118 page(s), 135173 of 150000 slots
in use (90%), 0 reusables, avg 2.56 hits
```

The 'avg hits' value is the average number of probes (hits) into the hash table to find slots being searched for. The counters used to compute the value are reset each time it is reported. Hit rates higher than three may indicate that a hash table should be increased in size. When a build completes, the log reports the minimum hash table size that the build requires. In the following example, *nnn* is the table size in slots:

```
[INTERNAL - 11:14:05]Hash: minimum table size  nnn
```

Allocating a larger hash table at the start of the build reduces the overhead associated with resizing the hash table during build execution.

By default, the initial size of the hash table is 200,000 slots. However, Data Manager precalculates the maximum possible number of hash table slots and uses the precalculated value when this is smaller than the specified number of slots.

As a build progresses, the hash table may become full which causes Data Manager to extend the hash table. Data Manager resizes the table to 150% of the previous size. You can determine whether the hash table was restructured or resized by searching the log. Entries of this format indicate that resizing occurred:

```
[INTERNAL - hh:mm:ss]Hash: table full, restructuring
[INTERNAL - hh:mm:ss]Hash: resizing table from xx to yy
[INTERNAL - hh:mm:ss]Hash: table restructuring complete
```

Where possible, you should determine the minimum hash table size from the log, and specify this value on the Memory tab of the Fact Build Properties window. If you need to specify a lower value, then use a value that will result in the minimum hash table size after resizing. You can calculate suitable values by multiplying the minimum hash table size by (2/3)*n* where *n* is an integer.

For example, if the log indicates that the minimum hash table is 10,000 slots, then, rounding up to the nearest integer:

10,000 x (2/3) = 6,670 10,000 x (2/3)2 = 4,450 10,000 x (2/3)3 = 2,963

In this example, 6670, 4450, and 2963 make good starting values.

Set Memory Limits

This section describes how to set memory limits.

Procedure

1. Click the required fact build.
2. From the **Edit** menu, click **Properties**.
3. Click the **Memory** tab.
4. In the **Working memory limit (MB)** box, type a suitable value to specify the maximum amount of memory to be used.

   For more information, see “Maximum Memory Usage” on page 347.
5. In the **Max page size (bytes)** box, type a suitable value to specify the maximum page size.

   For more information, see “Maximum Page Size” on page 347.
6. In the **Initial page table size (slots)** box, type a suitable value to specify the initial size of the page table.

   For more information, see “Initial Size of the Page Table” on page 348.
7. In the **Initial hash table size (slots)** box, type a suitable value to specify the initial size of the hash table.

   For more information, see “Initial Size of the Hash Table” on page 348.
8. Click **OK**.

Chapter 30. IBM Cognos Data Manager SQLTXT Designer

You use IBM Cognos Data Manager SQLTXT Designer to configure the SQLTXT definition files used within IBM Cognos Data Manager to access text files as though they are in an SQL database.

Data Manager also supports direct access to data using named pipes. For more information, see "Use Named Pipes to Access Data" on page 369.

You can generate test data through the MAKEROWS SQLTXT database.

You access SQLTXT databases through the SQLTXT connection type.

Create an SQLTXT Definition File

An SQLTXT definition file (.def) defines a virtual database of text file tables. It associates each table with a text file in the database directory. It also specifies the structure of the tables.

The SQLTXT definition file contains a number of entries. The first line of each entry consists of the table name enclosed in square brackets. Following this are definition lines that define the contents of the associated text file. Each definition line starts with a keyword and includes a number of parameters. White space separates the keyword and parameters.

You can include comments within the SQLTXT definition file. These commence with two forward slashes like this //, and continue to the end of the current line. Blank lines are ignored.

An SQLTXT definition file must be stored using a unicode encoding, such as UTF-16, if any table or column name contains characters not in the current code page. For example, when using Japanese table names in a Microsoft Windows operating system-1252 environment.

Procedure

1. In IBM Cognos Data Manager SQLTXT Designer, from the **File** menu, click **New**.
2. From the **File** menu, click **Save As**.
3. In the **File name** box, type a name for the file.

   Note: The definition file must have the extension .def
4. Click **Save**.

   When you have created a definition file, you must add tables and columns to it. You can do this from Data Manager SQLTXT Designer using the Import wizard, or by adding a table and columns manually.

   For more information, see "Import a Table Definition using the Import Wizard" on page 363, or "Define a Table Manually" on page 367.

SQLTXT Text File Format

In addition to a definition file, a SQLTXT database requires a text file for each SQL table being accessed.

Text files can exist in a variety of character sets and contain any of these types of rows:

- Header rows provide metadata about the data rows in the text file and are required by some applications to provide names for the data columns in the table. Header rows are optional in a SQLTXT database because the definition file provides this information.
- Data rows.
- Trailer rows provide optional information. However, they limit the range of operations that IBM Cognos Data Manager can perform on the file. Trailer rows prevent Data Manager from appending data to the file, effectively limiting interaction to SELECT and DROP statements.

Blank data rows are ignored but blank header or trailer rows are not.

Wildcards in SQLTXT File Names

If the text file associated with a SQLTXT table has a wildcard in the file name, rows are taken from all the files that match the wildcard.

This is useful where

- different source systems have identically formatted but differently named files in a directory

  Using wildcards means that these can be treated collectively.
- files have names that contain dates, perhaps where a file arrives every day

  These files can be referred to with a single logical name.

SQLTXT Column Names

Column names cannot begin with a number, and should not contain characters other than letters and numbers. If a column name contains other characters they are removed. If this results in duplicate column names, the duplicate column name is appended with a sequential number.

SQLTXT Record Types

SQLTXT can read and write

- delimited records with either delimited or fixed width columns
- nondelimited records

SQLTXT Record Delimiters

When you are defining delimited records, you must specify the delimiter used to separate the individual records in the file. The following table describes each of the delimiters:

Delimiter	Description
NL	New line. This value is dependent on the platform on which IBM Cognos Data Manager is running, and the locale of the input file. The supported options are • for an ASCII file on a Microsoft Windows operating system platform, the byte sequence is 13 10 • for an ASCII file on a UNIX operating system platform, the byte sequence is 10 • for an EBCDIC file on any platform, the byte sequence is 21
{CR}	Carriage return. The byte sequence is ASCII 13.
{LF}	Line-feed character. The byte sequence is ASCII 10.
{CRLF}	Carriage return and line feed combination. The byte sequence is ASCII 13 10.
Other	Another delimiter is used. In the Other box, type the delimiter values. **Note:** You must specify delimiter values using byte sequences. Each byte must be introduced with a backslash, followed by the byte value specified as a 3 digit decimal value. For example, to use the carriage return delimiter, type 013\010.

Note: The options in { } are defined by byte sequences. These options are translated into the selected character set for the file. For example, if the file is ASCII, and {LF} is selected, byte sequence 10 is the line delimiter. If EBCDIC is chosen, the ASCII byte sequence is translated as follows:

ASCII 10 becomes EBCDIC 13 ASCII 13 becomes EBCDIC 13 EBCDIC NL becomes ASCII 21

SQLTXT Column Delimiters

When you are defining delimited records with delimited columns, you must specify the delimiter used to separate the columns. The following table describes each of the delimiters:

Delimiter	Description
Tab	Columns are tab-delimited.
Comma	Columns are comma-delimited.
White	Columns are delimited by at least two characters. These can be space, tab, or Enter. You cannot enter a single space.
Space	Columns are delimited by one or more space characters.
Other	Another delimiter is used. In the Other box, type the delimiter code.

Quotation Characters in SQLTXT Delimited Records

In delimited records with delimited columns, a value may be enclosed within quotation characters, for example, "text". The use of quotation characters is optional. However, by using quotation characters, you can include values that contain the inter-column delimiter. For example the text "31, Acacia Avenue" must be enclosed in quotation characters in a comma-delimited file.

If you want to include a quotation character in a value that is enclosed in quotation characters, you must escape the included quotation character with another. For example, if the quotation character is a quotation mark, the value 'te"xt' must appear as 'te""xt' if the string in which it appears is enclosed in quotation marks.

When defining delimited records, you must specify whether columns are enclosed in quotation characters. The following table describes each of the quotation characters:

Quotation character	Description
Double	Columns are enclosed by double quotation marks.
Single	Columns are enclosed by single quotations marks.
None	Columns are not enclosed or a closing quotation character is missing for one or more fields.
Other	Columns are enclosed by another character. In the Other box, type the character.

SQLTXT removes quotation characters during the import process.

Note: Quoted fields must start and end with the same quotation character. If a quotation character is missing, IBM Cognos Data Manager continues to search the record, ignoring the column delimiters.

Special Characters as Delimiters

To use a nonprinting or special character as a delimiter, type a backslash, followed by the decimal value. For example, to use ESC as a delimiter, type **\27**, and to use AB, type **\065\066**

SQLTXT Data Types

SQLTXT supports the following data types for SQLTXT columns:

- BINARY
- BLOB
- BOOLEAN
- CHAR
- CLOB
- DATE
- DATE WITH TIMEZONE
- FLOAT

- INTEGER
- INTERVAL DAY TO SECOND
- INTERVAL YEAR TO MONTH
- NUMBER
- TIME
- TIME WITH TIMEZONE

BINARY

BINARY can be used in delimited and fixed width columns.

This data type interprets binary values as hexadecimal string values.

When setting up a column type of BINARY, you must specify the length of the column. In delimited columns, the length equates to the maximum number of bytes permitted for the column. In fixed width columns, the length equates to the exact number of bytes of data. Binary values are held in hexadecimal format in the data file.

Note: The length of a BINARY column can be 0 to 8192. If you specify 0 as the length, IBM Cognos Data Manager sets the length to 8192 by default.

BLOB

BLOB can be used only in delimited columns.

This data type can be used to represent large binary objects. The BLOB itself is held in a file and the data file contains a reference to this BLOB file in the format "<< filename >>".

When setting up a column type of BLOB, you must specify the maximum length.

Note: There is no limit to the size of a BLOB column (other than available disk space). If you specify 0 as the length, Data Manager sets the length to 8192 by default.

BOOLEAN

BOOLEAN can be used in delimited and fixed width columns.

This data type interprets the Boolean values TRUE and FALSE as string values.

When setting up a fixed width column type of BOOLEAN, you must specify the length of the column. The length equates to the exact number of bytes of data. For example, the value TRUE consists of four bytes.

Data Manager interprets the value as T, t and 1 as TRUE, and F, f and 0 as FALSE. All other values are interpreted as TRUE.

Note: In fixed width columns, the length of a BOOLEAN column can be 0 to unlimited. If you specify 0 as the length, Data Manager sets the length to 1 by default.

CHAR

CHAR can be used in delimited and fixed width columns.

This data type interprets letters and numbers as string values.

When setting up a column type of CHAR, you must specify the length of the column. In delimited columns, the length equates to the maximum number of bytes of data permitted for the column. In fixed width columns, the length equates to the exact number of bytes of data. For example, the value MYVALUE consists of seven bytes.

Note: The length of a CHAR column can be 0 to unlimited. If you specify 0 as the length, Data Manager sets the length to 30 by default.

CLOB

CLOB can be used only in delimited columns.

This data type can be used to represent large string objects. The CLOB itself is held in a file in UTF-16 encoding and the data file contains a reference to this CLOB file in the format "<< filename >>".

When setting up a column type of CLOB, you must specify the maximum length.

Note: There is no limit to the size of a CLOB column (other than available disk space). If you specify 0 as the length, Data Manager sets the length to 8192 by default.

DATE

DATE can be used in delimited and fixed width columns.

This data type interprets dates as string values.

When setting up a column type of DATE, you must specify the format of the date. In delimited and fixed width columns, the format must equate to the format of the date in text file.

Note: The format can be any valid date format. If no format is given, Data Manager sets the format to syyyy-mm-dd hh:mi:ss.fffffffff by default.

For more information, see “Date String Formats in SQLTXT Files” on page 360.

DATE WITH TIMEZONE

DATE WITH TIME ZONE can be used in delimited and fixed width columns.

This data type interprets dates with a time zone as string values.

When setting up a column type of DATE WITH TIME ZONE, you must specify the date format. In delimited and fixed width columns, the format must equate to the format of the date in the data file.

Note: The format can be any valid date format followed by a valid time zone format. If no format is given, Data Manager sets the format to syyyy-mm-dd hh:mi:ss.fffffffff stzh:tzm by default.

For more information, see "Date String Formats in SQLTXT Files" on page 360.

FLOAT

FLOAT can be used in delimited and fixed width columns.

This data type interprets floating point numbers as string values.

When setting up a fixed width column type of FLOAT, you must specify the length of the column. The length equates to the exact number of bytes of data. For example, the value 4.2000 consists of six bytes.

Note: In fixed width columns, the length of a FLOAT column can be 0 to 30. If you specify 0 as the length, Data Manager sets the length to 30 by default.

INTEGER

INTEGER can be used in delimited and fixed width columns.

This data type interprets integer values as string values.

When setting up a fixed width column type of INTEGER, you must specify the length of the column. The length equates to the exact number of bytes of data. For example, the value 12345 consists of five bytes of data.

Note: In fixed width columns, the length of an INTEGER column can be 0 to 19. If you specify 0 as the length, Data Manager sets the length to 19 by default.

INTERVAL DAY TO SECOND

INTERVAL DAY TO SECOND can be used in delimited and fixed width columns.

This data type interprets day to second intervals as string values.

When setting up a column type of INTERVAL DAY TO SECOND, you must specify the interval format. In delimited and fixed width columns, the format must equate to the format of the interval in the data file.

Note: The format can be any valid day to second interval format. If no format is given, Data Manager sets the format to sddddddddd hh:mi:ss.fffffffff by default.

For more information, see "Date String Formats in SQLTXT Files" on page 360.

INTERVAL YEAR TO MONTH

INTERVAL YEAR TO MONTH can be used in delimited and fixed width columns.

This data type interprets year to month intervals as string values.

When setting up a column type of INTERVAL YEAR TO MONTH, you must specify the interval format. In delimited and in fixed width columns, the format must equate to the format of the interval in the data file.

Note: The format can be any valid year to month interval format. If no format is given, Data Manager sets the format to syyyyyyyyy-mm by default.

For more information, see "Date String Formats in SQLTXT Files" on page 360.

NUMBER

NUMBER can be used in delimited and fixed width columns.

This data type interprets precise numbers as string values.

When setting up a column type of NUMBER, you must specify the length and scale of the column. In delimited columns, the length equates to maximum number of bytes of data permitted for the column. In fixed width columns, the length equates to the exact number of bytes of data. The scale represents the number of digits after the decimal point.

For example, the value 12345.67890 consists of 11 bytes of data, and has a scale of five.

The length of a NUMBER column can be 1 to 77 and the scale can be 0 to 77. The length must be greater than the scale.

The precision (number of digits) is assumed to be the length.

TIME

TIME can be used in delimited and fixed width columns.

This data type interprets times as string values.

When setting up a column type of TIME, you must specify the time format. In delimited and in fixed width columns, the format must equate to the format of the time in the data file.

Note: The format can be any valid time format followed by a valid time zone format. If no format is given, Data Manager sets the format to hh:mi:ss.fffffffff stzh:tzm by default.

For more information, see "Date String Formats in SQLTXT Files" on page 360.

TIME WITH TIMEZONE

TIME WITH TIME ZONE can be used in delimited and fixed width columns.

This data type interprets times with a time zone as string values.

When setting up a column type of TIME WITH TIME ZONE, you must specify the date format. In delimited and in fixed width columns, the format must equate to the format of the time in the data file.

Note: The format can be any valid time format followed by a valid time zone format. If no format is given, Data Manager sets the format to hh:mi:ss.fffffffff stzh:tzm by default.

For more information, see "Date String Formats in SQLTXT Files" on page 360.

PACKED

PACKED can only be used in fixed width columns.

This data type interprets decimal numbers with two digits per byte. For example, the value 123.45 consists of five digits and is represented as follows:

Byte	Byte	Byte
1 2	3 4	5 +

The last byte only ever contains one digit, the remainder of the byte being filled with the sign (positive or negative).

When setting up a column type of PACKED, you must specify the length and scale of the column. The length represents the exact number of bytes of data. The scale represents the number of digits after the decimal point.

For example, the value 123.45 consists of three bytes of data (as shown above), and has a scale of two.

The scale must be less than or equal to (length *2)-1

Because the last byte can only contain one digit, the precision (number of digits) is calculated as (length *2)-1

This data type does not support null values. Data Manager replaces null values with zero when writing data of this type.

You can only read packed data from an EBCDIC file, not an ASCII file.

ZONED

ZONED can only be used in fixed width columns.

This data type interprets decimal numbers with one digit per byte. The final byte also contains the sign (positive or negative).

For example, the value 1234.56 consists of six digits and therefore six bytes.

When setting up a column type of ZONED, you must specify the length and scale of the column. The length represents the exact number of bytes of data. The scale represents the number of digits after the decimal point.

For example, the value 1234.56 consists of six bytes of data, and has a scale of two.

Note that:

- The length of a ZONED column can be 1 to 77 and the scale can be 0 to 77. The length must be greater than the scale.
- The precision (number of digits) is assumed to be the length.
- This data type does not support null values. Data Manager replaces null values with zero when writing data of this type.
- You can read zoned data from an EBCDIC or ASCII file.

Date String Formats in SQLTXT Files

IBM Cognos Data Manager requires that date strings containing text months (for example, Jan, Feb), are in English. If you attempt to import another language, an error appears.

Error Checking in SQLTXT

The Action on row error feature allows you to specify the action that IBM Cognos Data Manager should take if format errors are detected in data rows when reading files in a table during a SELECT operation.

Note: Only those columns specified in the SELECT statement have their data validated against the column data type specification. Any other columns included in the SQLTXT definition file are ignored.

The different actions are described below.

Action	**Description**	**Use when**
Error	Immediately abort the current SQLTXT operation and issue an error message.	You want to be sure that the file being read has absolute integrity.
Warn	Ignore the data row, and continue processing the file. Issue any warnings about this row when processing is complete. A summary warning message, which includes the number of data rows ignored, is always issued if errors are detected. Warnings about individual data rows are issued according the limit specified. For more information, see "Define a Table Manually" on page 367.	You want to process the file, but require warning for any potential problems.
Ignore	Ignore the data row and continue processing the file. No warnings are issued.	You are completely certain of the integrity of the file, or you want to process it irrespective of any errors.

You should take into consideration the following points when deciding which action to use.

Consideration	**Description**
Too many columns (delimited files only)	Error detected unless Ignore extra columns is selected. For more information, see "Define a Table Manually" on page 367.
Too few columns (delimited files only)	Error detected.

Consideration	Description
Value longer than column definition (CHAR only)	Error detected unless Allow truncation is selected. For more information, see "Define a Table Manually" on page 367.
Invalid or incorrectly formatted data	Error detected.
Null values in non nullable fields	Error detected.

For information on using Action on row error, see "Define a Table Manually" on page 367.

Null Values in SQLTXT

If a field in a file contains a null value, the way in which this value is handled depends upon the data type and nullability setting of the column in the table definition.

For information about each of the data types, see "SQLTXT Data Types" on page 354. For information about nullability settings, see "Add Columns to a Table" on page 368.

Data type	Outcome
CHAR	For a field containing a value that is empty or contains all spaces (after being trimmed if Remove trailing whitespace from fields is selected): • if the column is not nullable, the value is unchanged • if the column is nullable, the value is null
PACKED and ZONED	These data types cannot represent null values.
All other data types	If a field is empty or contains all spaces, the value is null.
DATE	For a date field that contains the value 0000-00-00 00:00:00, or a variant of this (depending on the format selected): • if the column is not nullable, an error occurs • if the column is nullable, the value is null
TIME	For a time field that contains the value 00:00:00, or a variant of this (depending on the format selected): • if the column is not nullable, an error occurs • if the column is nullable, the value is null
DATE WITH TIME ZONE	For a date with time zone field that contains the value 0000-00-00 00:00:00 00:00, or a variant of this (depending on the format selected): • if the column is not nullable, an error occurs • if the column is nullable, the value is null

Data type	Outcome
TIME WITH TIME ZONE	For a time with time zone field that contains the value 00:00:00 00:00, or a variant of this (depending on the format selected): • if the column is not nullable, an error occurs • if the column is nullable, the value is null
INTERVAL DAY TO SECOND	For a day to second interval field that contains the value 0 00:00:00, or a variant of this (depending on the format selected): • if the column is not nullable, an error occurs • if the column is nullable, the value is null
INTERVAL YEAR TO MONTH	For a year to month interval field that contains the value 0-0, or a variant of this (depending on the format selected): • if the column is not nullable, an error occurs • if the column is nullable, the value is null

Change the Font in IBM Cognos Data Manager SQLTXT Designer

You can change the font used in IBM Cognos Data Manager SQLTXT Designer.

Procedure

1. From the **Tools** menu, click **Options**.
2. Click the appropriate **Browse** button ... for the setting that you want to change
   - **Application font** applies to the **Tree** pane, and all window and dialog boxes
   - **Fixed width Font** applies to places where a fixed width font is used

   The **Select Font** dialog box appears.
3. From the drop down lists, select the appropriate font.

   Tip: If you are changing fixed width fonts, you can restrict the list of fonts to fixed width fonts only by selecting the **Fixed width only** check box.
4. Click **OK** to close the **Select Font** dialog box.

   Tip: To return to the default fonts, click **Reset**.
5. Click **OK**.

Open a SQLTXT Database to Access Text Files

You access text files using a SQLTXT database that you define. You must open a database within IBM Cognos Data Manager SQLTXT Designer before you can work with its contents. If there is a database open, a message appears informing you that IBM Cognos Data Manager will disconnect from the current database.

Procedure

1. From the **File** menu, click **Open**.

   The **Open SQLTXT Definition File** dialog box appears.
2. Locate the required database.

   Tip: If you must first create a database, from the **File** menu, click **New**.
3. Click **Open**.

Import a Table Definition using the Import Wizard

IBM Cognos Data Manager SQLTXT Designer can read and write

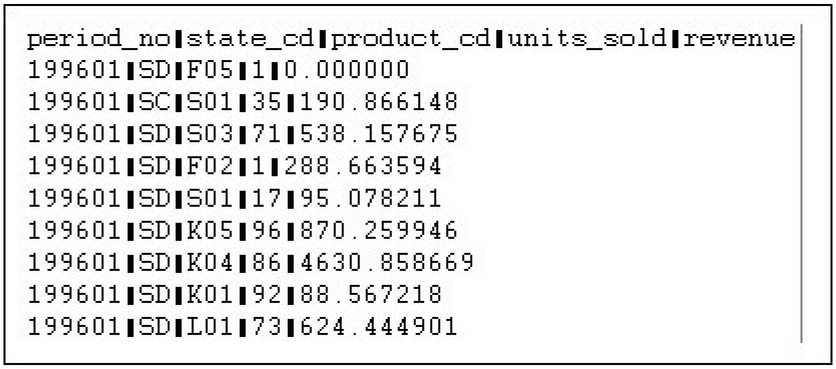
- nondelimited records
- delimited records with either delimited columns, or fixed width columns

You can define a table using the Import wizard, or manually by entering table and column definitions.

For information about the manual method, see “Define a Table Manually” on page 367.

Import Delimited Records with Delimited Columns

In the following example, each column is delimited by a tab character (indicated by a black vertical bar), and each record is delimited by a carriage return (indicated by the vertical gray line on the right).



```
period_no▮state_cd▮product_cd▮units_sold▮revenue
199601▮SD▮F05▮1▮0.000000
199601▮SC▮S01▮35▮190.866148
199601▮SD▮S03▮71▮538.157675
199601▮SD▮F02▮1▮288.663594
199601▮SD▮S01▮17▮95.078211
199601▮SD▮K05▮96▮870.259946
199601▮SD▮K04▮86▮4630.858669
199601▮SD▮K01▮92▮88.567218
199601▮SD▮L01▮73▮624.444901
```

Procedure

1. Open the appropriate SQLTXT database.
2. From the **Edit** menu, click **Import Table Data**.

   The **Import SQLTXT Table Data File** dialog box appears.
3. Select the text file to import, and click **Open**.

   Tip: You can change the font used in the **Import** wizard by clicking the **Font** button.
4. Click **Delimited records**.
5. In the **Record delimiter** box, click the delimiter used to separate individual records in the file.

   For information on delimiter types, see “SQLTXT Record Delimiters” on page 352.
6. In the **Number of header rows** box, enter the number of header rows in the file.

   IBM Cognos Data Manager SQLTXT Designer uses these header rows to name the columns.
7. In the **Decimal symbol** box, type a single character that is used for the decimal point in numeric values. For example, the value 123.45 uses a period (.) as the decimal point.
8. In the **Character set** box, enter the encoding used in the text file.

   Tip: You can click an encoding type from the list, or type the code page into the box.

9. In the **Digit grouping symbol** box, type the character that is used for the thousands separator in numeric values. For example, the value 12,345 uses a comma as the thousands separator.

   You can only specify a single character.
10. In the **Preview rows** box, enter the number of rows that you want to view in the wizard.

    Tip: Click **Refresh Preview** to reflect your selections.
11. Click **Next**.
12. Select **Delimited columns**.
13. In the **Delimiter type** box, click the delimiter used to separate the columns in the table.

    For information on column delimiter types, see "SQLTXT Column Delimiters" on page 353.
14. In the **Quote character** box, click the quotation character used to enclose individual columns within a record.

    For information on quote characters, see "Quotation Characters in SQLTXT Delimited Records" on page 354.
15. Click **Next**.

    IBM Cognos Data Manager automatically names and attempts to specify the data type of each column. You can override these selections.
16. To rename a column, click the column name and type a new name.
17. To change a data type, double-click the appropriate column, and in the **Data Type** dialog box, click a different data type.

    If you select NUMBER, you must also specify the number of digits expected after the decimal point in the **Scale** box.

    If you select a date or time, you must also specify the format in the **Format** box.

    For information about each of the data types, see "SQLTXT Data Types" on page 354.
18. Click **OK** to close the **Data type** dialog box.
19. Click **Process All Records** to ensure that all record lengths and data types are correct, and then click **Close**.
20. Click **Finish**.

    The wizard adds the table to the database.

    You can now customize the definition file as required. For information, see "Define a Table Manually" on page 367.

Import Delimited Records With Fixed Width Columns

In the following example, each column is of a fixed width, and each record is delimited by a carriage return (indicated by the vertical gray line on the right).

Procedure

1. Open the appropriate SQLTXT database.
2. From the **Edit** menu, click **Import Table Data**.

   The **Import SQLTXT Table Data File** dialog box appears.
3. Select the text file to import, and click **Open**.

   Tip: You can change the font used in the Import wizard by clicking the **Font** button.
4. Click **Delimited records**.

5. In the **Record delimiter** box, click the delimiter used to separate individual records in the file.

   For information on delimiter types, see “SQLTXT Record Delimiters” on page 352.

6. In the **Number of header rows** box, enter the number of header rows in the file.

   IBM Cognos Data Manager SQLTXT Designer uses these header rows to name the columns.

7. In the **Decimal symbol** box, type a single character that is used for the decimal point in numeric values. For example, the value 123.45 uses a period (.) as the decimal point.

8. In the **Character set** box, click the encoding used in the text file.

   Tip: You can click an encoding type from the list, or type the code page into the box.

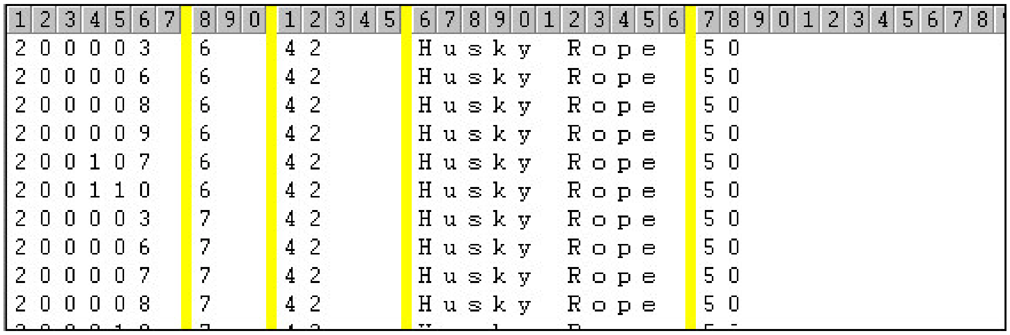
9. In the **Digit grouping symbol** box, type the character that is used for the thousands separator in numeric values. For example, the value 12,345 uses a comma (,) as the thousands separator.

   You can only specify a single character.

10. In the **Preview rows** box, enter the number of rows that you want to view in the wizard.

    Tip: Click **Refresh Preview** to reflect your selections.

11. Click **Next**.



```
1234567890123456789012345678
200003 6  42   Husky Rope 50
200006 6  42   Husky Rope 50
200008 6  42   Husky Rope 50
200009 6  42   Husky Rope 50
200107 6  42   Husky Rope 50
200110 6  42   Husky Rope 50
200003 7  42   Husky Rope 50
200006 7  42   Husky Rope 50
200007 7  42   Husky Rope 50
200008 7  42   Husky Rope 50
```

12. Select **Fixed width columns**.

13. Click on the first character in each column to add a delimiter to separate the columns.

    IBM Cognos Data Manager adds a vertical bar to indicate the position of the delimiter.

    Tip: You can move a delimiter by dragging it to a new position. To delete a delimiter, click it.

14. Click **Next**.

    Data Manager automatically names and attempts to specify the data type of each column. You can override these selections.

15. To rename a column, click the column name and type a new name.

16. To change a data type, double-click the appropriate column, and in the **Data type** dialog box, click a different data type.

    If you select NUMBER, you must also specify the number of digits expected after the decimal point in the **Scale** box.

If you select a date or time, you must also specify the format in the **Format** box.

For information about each of the data types, see “SQLTXT Data Types” on page 354.

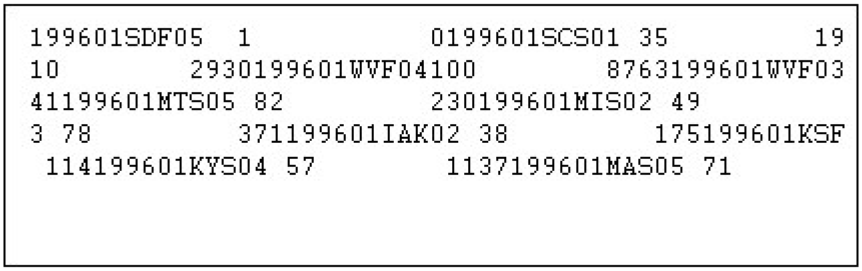
17. Click **OK** to close the **Data type** dialog box.
18. Click **Process All Records** to ensure that all record lengths and data types are correct. Click **Close**.
19. Click **Finish**.

    The wizard adds the table to the database.

    You can now customize the definition file as required. For information, see “Define a Table Manually” on page 367.

Import Nondelimited Records

The following example shows a nondelimited record which is a single string that does not have delimiters or fixed columns.



```
199601SDF05  1           0199601SCS01 35         19
10        2930199601WVF04100        8763199601WVF03
41199601MTS05 82         230199601MIS02 49
3 78         371199601IAK02 38         175199601KSF
 114199601KYS04 57        1137199601MAS05 71
```

Procedure

1. Open the appropriate SQLTXT database.
2. From the **Edit** menu, click **Import Table Data**.

   The **Import SQLTXT Table Data File** dialog box appears.
3. Select the text file to import, and click **Open**.

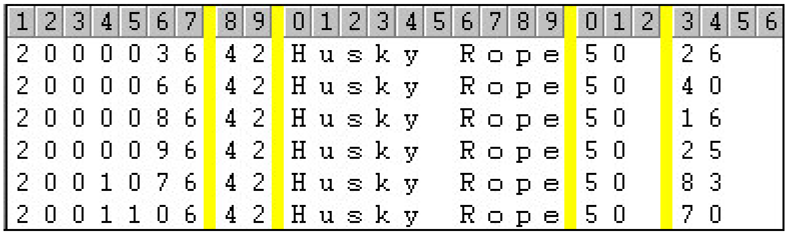
   Tip: You can change the font used in the Import wizard by clicking the **Font** button.
4. Click **Non-delimited records**.
5. In the **Record length** box, type the number of characters in the record.
6. In the **Number of header rows** box, enter the number of header rows in the file.
7. In the **Decimal symbol** box, type a single character that is used for the decimal point in numeric values. For example, the value 123.45 uses a period (.) as the decimal point.
8. In the **Character set** box, click the encoding used in the text file.

   Tip: You can click an encoding type from the list, or type the code page into the box.
9. In the **Digit grouping symbol** box, type the character that is used for the thousands separator in numeric values. For example, the value 12,345 uses a comma (,) as the thousands separator.

   You can only specify a single character.
10. In the **Preview rows** box, enter the number of rows that you want to view in the wizard.

    Tip: Click **Refresh Preview** to reflect your selections.

11. Click **Next**.



```
1234567 89 0123456789 012 3456
2000036 42 Husky Rope 50  26
2000066 42 Husky Rope 50  40
2000086 42 Husky Rope 50  16
2000096 42 Husky Rope 50  25
2001076 42 Husky Rope 50  83
2001106 42 Husky Rope 50  70
```

12. Click on the first character in each column to add a delimiter to separate the columns.

    IBM Cognos Data Manager adds a vertical bar to indicate the position of the delimiter.

    Tip: You can move a delimiter by dragging it to a new position. To delete a delimiter, click it.
13. Click **Next**.

    Data Manager automatically names and attempts to specify the data type of each column. You can override these selections.
14. To rename a column, click the column name and type a new name.
15. To change a data type, double-click the appropriate column, and in the **Data type** dialog box, click a different data type.

    If you select NUMBER, you must also specify the number of digits expected after the decimal point in the **Scale** box.

    If you select a date or time, you must also specify the format in the **Format** box.

    For information about each of the data types, see "SQLTXT Data Types" on page 354.
16. Click **OK** to close the **Data type** dialog box.
17. Click **Process All Records** if you want to ensure that all record lengths and data types are correct. Click **Close**.
18. Click **Finish**.

    The wizard adds the table to the database.

    You can now customize the definition file as required. For information, see "Define a Table Manually."

Define a Table Manually

Instead of using the Import wizard, you can define a table manually by adding an empty table, selecting the file that holds the data, and then adding the required columns.

Procedure

1. Open the appropriate SQLTXT database.
2. Click the **Database** icon.
3. From the **Edit** menu, click **Insert Table**.

   An empty table appears in the tree.
4. Click the new table and, from the **Edit** menu, click **Rename**.
5. Type the new name for the table.
6. In the **Data file** box, enter the name of the text file.

7. In the **Header lines** box, type the number of header rows in the file.
8. In the **Trailer lines** box, type the number of trailer lines in the file.

   SQLTXT ignores these rows when it accesses data in the file.
9. In the **Character set** box, enter the encoding used in the text file.

   Tip: You can click an encoding type from the list, or type the code page into the box.
10. In the **Action on row error** box, click the action to apply when faulty data is read.

    For information, see "Error Checking in SQLTXT" on page 360.
11. If you selected **Warn** in the **Action on row error** box, specify the maximum number of warnings to report in **Limit** box.

    Tip: For unlimited warnings, set the limit to -1.
12. Select or clear the **Remove trailing whitespace from fields** check box as appropriate.
13. Specify whether columns in the input text file are of fixed length or delimited by clicking either **Fixed length records** or **Delimited records**.
14. In the **Record delimiter** box, click the delimiter used to separate individual records in the file.

    For information on the delimiter types, see "SQLTXT Record Delimiters" on page 352.
15. If you selected **Delimited records**, in the **Column delimiter** box, specify the delimiter used to separate the columns in the table.

    For information on the delimiter types, see "SQLTXT Column Delimiters" on page 353.
16. If you selected **Delimited records**, in the **Quote character** box, click the quotation character used to enclose individual columns within a record.

    For information on quote characters, see "Quotation Characters in SQLTXT Delimited Records" on page 354.
17. If you selected **Delimited records**, select the **Ignore extra columns** check box to ignore any excess fields in a single record.

    Note: SQLTXT does not perform any error checks on the excess columns.
18. If you selected **Delimited records**, select the **Allow truncation** check box to truncate data in a column that exceeds the specified column length.
19. In the **Decimal symbol** box, type a single character that is used for the decimal point in numeric values. For example, the value 123.45 uses a period (.) as the decimal point.
20. In the **Digit grouping symbol** box, type the character that is used for the thousands separator in numeric values. For example, the value 12,345 uses a comma (,) as the thousands separator.

    You can only specify a single character.

Add Columns to a Table

This section describes how to add columns to add table.

Procedure

1. Click the table to which you want to add a column.
2. From the **Edit** menu, click **Insert Column**.

   A new column appears beneath the selected table.

3. In the **Data type** box, click the data type for the column.

   If you have specified that columns are delimited and you select

   - BINARY, BLOB, CHAR, or CLOB, enter the maximum column width found in the text file in the **Length** box
   - a date or time, enter the format of the date in the **Format** box

   If you have specified that columns are of fixed width and you select

   - NUMBER, PACKED or ZONED, enter the required decimal value in the **Length.Scale** box
   - a date or time, enter the format of the date in the **Format** box
   - any other data type, enter the required value in the **Length** box

   For information about each of the data types, see “SQLTXT Data Types” on page 354.
4. If the column can accept null values, select the **Nullable** check box.
5. If the column forms part of the primary key of the table, select the **Key** check box.

Create a Similar Table

If you want to create a table that is similar to an existing table, duplicate the table and then make the required changes. This is often quicker and simpler than creating a completely new table.

Procedure

1. Click the table to duplicate.
2. From the **Edit** menu, click **Duplicate Table**.

   IBM Cognos Data Manager SQLTXT Designer duplicates the table and assigns a unique name to it. The columns in the original table are also duplicated, but the original column names are retained.

Edit a Column or a Table

You can edit an existing column or table.

Procedure

1. Click the table or column that you want to edit.
2. In the right pane, edit the values as required.

Delete a Column or a Table

You can delete a column or table that is no longer required.

Procedure

1. Click the table or column that you want to delete.
2. From the **Edit** menu, click **Delete**.
3. In the message box, click **Yes**.

Use Named Pipes to Access Data

You can read data from, and write data to, pipes using SQLTXT. When reading data, IBM Cognos Data Manager continues to read from the pipe until it is closed, or an end-of-file marker (EOF) is read.

Define a Pipe using IBM Cognos Data Manager SQLTXT Designer

To define a pipe in IBM Cognos Data Manager SQLTXT Designer, you must manually create a SQLTXT definition file (.def).

For more information, see “Define a Table Manually” on page 367.

The definition file must include the name of the pipe in the **Data file** box. The syntax for pipes in SQLTXT is

```
"[PIPE!]/<filename>"
```

Although it is preferable to always include the pipe syntax, SQLTXT uses named pipes by default in the following circumstances:

- In the Microsoft Windows operating system, if the specified file name takes the form
  - `"//./pipe/<name>"` for a local pipe
  - `"//<machine_name>/pipe/<name>"` for a network pipe
- In the UNIX operating system, if the specified file name already exists, and it is a pipe (FIFO) device.

It is also possible to specify a pipe, using the pipe syntax, where the file name does not include any directory information. For example:

```
"PIPE!test"
```

In Windows, SQLTXT assumes that this is a named pipe on a local machine. Using the example above, this would be

```
"PIPE!//./pipe/test"
```

In UNIX, SQLTXT assumes that this is a named pipe located in the same directory as the SQLTXT definition file.

Notes

- Header lines are supported when reading pipes, and ignored when writing pipes.
- Trailer lines are not supported, and cause a run-time error if specified. Therefore, you should ensure that the value in the **Trailer lines** box is always set to zero.

Example SQLTXT Definition Files for Pipes

The following is an example SQLTXT definition file for a Microsoft Windows operating system pipe.

```
[Input]
FILENAME '//./pipe/input'
VERSION 8
CHARSET 'GB2312'
HEADERLINES 4
TRAILERLINES 0
ROWERROR ERROR
QUOTECHAR DOUBLE
DECIMALSEP '.'
THOUSANDSEP NONE
LINEDELIMITER CR
```

```
RECORD TAB
COLUMN 'col1' CHAR 3 'NOT NULL' 'NOT KEY'
COLUMN 'col2' CHAR 3 'NOT NULL' 'NOT KEY'
```

The following is an example SQLTXT definition file for a UNIX operating system pipe.

```
FILENAME 'pipe!/output'
VERSION 8
CHARSET 'ASCII'
HEADERLINES 1
TRAILERLINES 0
ROWERROR ERROR
QUOTECHAR DOUBLE
DECIMALSEP '.'
THOUSANDSEP ','
LINEDELIMITER CR
RECORD TAB
COLUMN 'C1' INTEGER 10 'NOT NULL' 'NOT KEY'
COLUMN 'C2' CHAR 33 'NOT NULL' 'NOT KEY'
```

Create an External Pipe

IBM Cognos Data Manager can read data from pipes that have been created by external applications. When you create an external pipe, there are certain restrictions that you should be aware of.

Data Manager creates named pipes using the CreateNamedPipe() Microsoft Windows operating system API function with the following parameter settings:

- PIPE_ACCESS_DUPLEX is used for the dwOpenMode parameter. If the client end uses the SetNamedPipeHandleState() Windows API function on the pipe handle, it requires GENERIC_WRITE access to the named pipe.

  Note: When Data Manager is the client, it will use the SetNamedPipeHandleState() function if the pipe handle is not in PIPE_READMODE_BYTE state, therefore external programs creating pipes for Data Manager should use PIPE_ACCESS_DUPLEX.
- PIPE_TYPE_MESSAGE, PIPE_READMODE_BYTE, and PIPE_WAIT are used for the dwPipeMode parameter.

Data Manager writes an explicit EOF to the pipe when it has sent all the data. This is done by flushing the pipe, writing a zero byte message and then flushing the pipe again before disconnecting, as shown by the following C language code sample:

```
FlushFileBuffers( hPipe );
WriteFile( hPipe, (LPVOID)NULL, (DWORD)0, &dwWritten, (LPOVERLAPPED)NULL);
FlushFileBuffers( hPipe );
DisconnectNamedPipe( hPile );
```

If Data Manager is acting as a client end of a named pipe (using SELECT to obtain data from it), it detects an EOF either by a successful call to ReadFile() when a zero byte read is returned, or when ReadFile() returns an error and the last error code is ERROR_NO_DATA.

Test Pipe Access

If you are reading data from a pipe, you can test the pipe access using SQLTerm or by executing the DataStream used to read the data.

For information on using SQLTerm, see Chapter 6, “SQLTerm and SQL Helper,” on page 31. For information on executing a DataStream, see “Execute a DataStream” on page 133.

When you execute the SQL query or DataStream, if the pipe cannot be found, there is a short delay while IBM Cognos Data Manager waits for a pipe to be created. During this time it is not possible to interrupt the execution. If no pipe is created, Data Manager informs you after the delay period times out.

Note: By default, the delay timeout period is 30 seconds. You can change this value by setting the DS_PIPE_CREATE_WAIT variable.

SQL Support for SQLTXT Files

A SQLTXT database supports a limited level of SQL. It is possible to select, filter, and deliver data, to and from SQLTXT files.

Where possible, SQLTXT supports the functions and operators that are available in IBM Cognos Data Manager. Some functions, such as member functions, are meaningless outside the context of a Data Manager build and SQLTXT does not support them.

For information about Data Manager functions and operators, see the IBM Cognos*Function and Scripting Reference Guide*.

Avoid using VALUES Clauses in Expressions

SQLTXT does not support expressions within VALUES clauses of INSERT statements, such as the following:

```
INSERT INTO progress(timestamp, message)
VALUES (datetimenow(), 'Starting')
```

However, you can use a dummy table, which has a single data row, to avoid using a VALUES clause.

Procedure

1. Create a dummy table, for example
   ```
   CREATE TABLE dummy (dummy CHAR(1));
   ```
2. Into the dummy table, insert a single data row, for example
   ```
   INSERT INTO dummy VALUES ('1');
   ```
3. Insert the required data into the table; for example
   ```
   INSERT INTO progress(timestamp, message)
   SELECT datetimenow(), 'Starting'
   FROM dummy;
   ```

Examples

These are examples of SQL statements supported by IBM Cognos Data Manager SQLTXT Designer.

SELECT Statements

-
  ```
  SELECT DISTINCT * FROM sales;
  ```

  This example retrieves all data from the table named sales and removes any duplicate data rows.

-
  ```
  SELECT prod_cd, unit_price*units_sold
  FROM sales
  WHERE prod_cd LIKE 'aa%';
  ```

  This example retrieves prod_cd and calculates the corresponding product of unit_price and units_sold for all data rows in the table named sales that have a prod_cd value starting with aa.

-
  ```
  SELECT DISTINCT *
  FROM product
  WHERE unit_price BETWEEN 1.0 AND 2.5;
  ```

  This example removes all duplicate data rows, retrieves all data from the table named product where the unit_price value is between 1.0 and 2.5 inclusive.

-
  ```
  SELECT DISTINCT prod_cd, unit_price
  FROM sales
  WHERE sales.prod_cd IN ('S01', 'S03', 'S05');
  ```

  This example removes all duplicate data rows, retrieves the prod_cd and unit_price from all data rows in the table named sales where prod_cd is one of S01, S02, or S03.

INSERT Statements

-
  ```
  INSERT INTO product (prod_cd, name, unit_price)
  VALUES ('p001', 'Widget', 2.25);
  ```

  This example inserts into the table named product a single data row having the value called prod_cd equal to 'p001', the value named name value equal to 'Widget', and the value named unit_price equal to 2.25. Any other fields contain null.

-
  ```
  INSERT INTO sales_copy SELECT * FROM sales;
  ```

  This example copies all rows from the table named sales to the table named sales_copy.

DELETE Statements

-
  ```
  DELETE FROM sales WHERE prod_cd='p001';
  ```

  This example deletes all rows from the table named sales where prod_cd is 'p001'.

-
  ```
  DELETE FROM sales;
  ```

  This example deletes all rows from the table named sales.

CREATE TABLE Statements

-
  ```
  CREATE TABLE product (
  prod_cd CHAR(3) 'NOT NULL' 'KEY',
  name CHAR(16) 'NULL' 'NOT KEY',
  unit_price FLOAT 'NULL' 'NOT KEY');
  ```

  This example creates a table named product that has a column named prod_cd of type CHAR with length 3 that is the primary key of the table, a column named name of type CHAR with length 16, and column named unit_price of type FLOAT.

DROP TABLE Statements

- `DROP TABLE product;`

  This example removes the table named product from the database and deletes the associated text file.

Special Columns

ROWNUM, FILEROWNUM, and FILENAME are special, read-only columns automatically added to SQLTXT data tables. ROWNUM returns the ordinal number of the row within the returned data; FILEROWNUM returns the line number of the row within the entire table; FILENAME returns the name of the file containing the current row.

For example, this is a Sales table.

Period	Location	Product	Sales	Revenue
200001	SD	F05	1	0
200001	SC	S01	35	191
200001	SD	S03	71	538
200001	SD	F02	1	289
200001	SD	S01	17	95

This SQL statement can be run against the Sales table:

```
SELECT ROWNUM, FILEROWNUM, Location, Product
FROM Sales
WHERE Product LIKE 'K%'
```

ROWNUM	FILEROWNUM	Location	Product
1	6	SD	K05
2	7	SD	K04
3	8	SD	K01

Generating Test Data

MAKEROWS is a virtual SQLTXT database that IBM Cognos Data Manager provides to generate test data. It does not have a corresponding definition file.

To generate test data, write a special type of SELECT statement with the following where <expression> is a literal value or a function, and <number_of_rows> is the number of rows to generate.

```
SELECT [DISTINCT] <expression>{,<expression>} FROM <number_of_rows>;
```

Because the generated rows may contain duplicates, the number of returned rows might be less than this number if you use the DISTINCT option.

These examples generate rows of test data:

- `SELECT 'sample' FROM 1000;`

  This example generates 1,000 rows of data, each containing a single column having the text value, 'sample'.
- `SELECT RAND(1,1000,1), 'sample' FROM 1000;`

  This example generates 1,000 rows of data, each containing two columns. The first column contains a random number between 1 and 1,000. The second column contains the text value, 'sample'.
- `SELECT DISTINCT RAND(1,10000,1), 'sample' FROM 1000;`

  This is the same as the previous example except that IBM Cognos Data Manager removes duplicate data rows from the output. Thus, the number of output data rows might be less than 1,000.

Connect to the MAKEROWS Database

You can connect to the MAKEROWS database from IBM Cognos Data Manager Designer, from the Data Manager language, and from SQLTerm.

In the Designer, use the SQLTXT DBMS driver to create a connection to the MAKEROWS database, and then replace the name of the definition file with MAKEROWS.

In the Command Line, create an alias for the MAKEROWS database by using this syntax where <runenv> is the run environment context for the alias, and <aliasname> is your name for the alias.

```
<runenv> <aliasname> SQLTXT 'MAKEROWS'
```

In SQLTerm, use the following command:

```
CONNECT SQLTXT MAKEROWS
```

Chapter 31. Commands

The following table briefly describes all the IBM Cognos Data Manager commands.

Command	Description
databuild	Executes the fact build that you specify in command line parameters, lists available delivery modules, or lists delivery module properties.
dimbuild	Executes the dimension build that you specify in command line parameters.
rundsjob	Executes a JobStream.
dsremote	Executes a fact build, dimension build, or JobStream that is located on a remote computer.
catimp	Imports a component from a text file to a Data Manager catalog.
catexp	Exports a component from a Data Manager catalog to a text file.
catlist	Lists the contents of a Data Manager catalog.
catbackup	Performs a backup of a Data Manager catalog to a text file.
catrestore	Restores a backup of a Data Manager catalog from a text file.
catupgrade	Upgrades a Data Manager catalog.
catmanage	Manages a Data Manager catalog.
catdoc	This command documents the contents of a catalog.
showref	Lists the members of a reference structure.
dstofm	Exports a metadata collection to an XML file. This file can then be used to produce a model of your data mart or data warehouse in IBM Cognos Framework Manager, without the need to recreate the complete model.
sqlterm	Starts the SQLTerm application.
catpkgtoxml	Converts a Data Manager package file to a data movement specification XML file.

Tip: When using parameters that contain embedded spaces, you may need to enclose them in double quotation marks. For example, "DSN=Global Sales".

databuild

This command executes a fact build that you specify in command line parameters, lists available delivery modules, or lists the properties for the specified delivery module.

Execute a Fact Build

Syntax

```
databuild [<catdb_driver> <catdb_items>] <databuild_name
or uuid> [options]
```

When you run databuild with this syntax, it executes the named fact build that resides in the named catalog or package (specified using the option -p).

Symbol	Description
<catdb_driver>	Specifies the database driver to use to connect to the catalog.
<catdb_items>	Specifies the connection string to pass to the database driver. For example, to connect as a user named DBA, with password PASS, to a catalog that resides in an ODBC data source named Sample, you would specify `"DSN=Sample;UID=DBA;PWD=PASS"`
<databuild_name>	Specifies the name of the fact build to execute within the catalog or package.
<uuid>	Specifies the universally unique identifier for the component to execute.
<options>	For information, see “databuild Options” on page 379.

Example

```
databuild ODBC "DSN=SalesCat;UID=Sales;PWD=dm123" US_Sales
```

This command runs databuild to execute the fact build named US_Sales. To retrieve the build definition, databuild connects to the ODBC data source named SalesCat as user Sales, with password dm123.

```
databuild -p"c:\data files\sales.pkg" US_Sales
```

This example runs databuild to execute a fact build named US_Sales from the package named sales.pkg.

List All Delivery Modules

Syntax

```
databuild -D
```

When you run databuild with this syntax, it returns a list of all the delivery modules to which you have access.

Example

```
databuild -D
Available Delivery Modules are:
===============================
FACT Delivery
-------------
DB2LOAD     - DB2 LOAD
ESSDIRECT   - Essbase Direct Feed
ESSFILE     - Essbase File Feed
INXLOAD     - Informix LOAD
SSLOAD      - Microsoft SQL Server BCP Command
ORALOAD     - ORACLE SQL*Loader
RBLOAD      - Red Brick Loader (TMU)
TABLE       - Relational Table
BCPLOAD     - SQL Server Bulk Copy via API
FASTLOAD    - Teradata Fastload
MULTILOAD   - Teradata Multiload
TPUMP       - Teradata TPump
TEXT        - Text File
DIMENSION Delivery
------------------
STAR        - Relational Table
```

For more information on delivery modules, see Appendix A, “Fact Delivery Module Properties,” on page 413.

List the Properties of a Delivery Module

Syntax

```
databuild -D<module>
```

When you run databuild with this syntax, it returns a list of all the properties for the delivery module that you specify on the command line.

Symbol	Description
<module>	Specifies the name of the delivery module for which you want to list the properties.

For more information on delivery modules, see Appendix A, “Fact Delivery Module Properties,” on page 413.

databuild Options

Syntax

```
<options> ::=
 [-e<runenv>] [-p<package_file>] [-A<dbalias_file>] [-C]
 [-D<module_name>] [-L<logfile>] [-O] [-P] [-T<tracefile_root>]
 {-V<name>[=<value>]}[-X[F][D]
```

Symbol	Description
-e	Denotes the run environment for the fact build.
<runenv>	Specifies the run environment to apply to identify the correct database alias.
-p	Denotes that the fact build is executed from a package.
<package_file>	Specifies the name and path of the package file. **Note:** If you also specify catalog details, then logging and audit information is stored in the catalog, even if the catalog has no connection with the package.
-A	Denotes the text file that contains the database alias definitions to use for the fact build.
<dbalias_file>	Specifies the name and path of the file that contains the database alias definitions to use for the fact build.
-C	Executes the fact build in check only mode.
-D	Lists the available delivery modules, or the properties of the specified delivery module.
<module_name>	Specifies the name of the delivery module for which you want to list the properties.
-L	Logs progress to the specified file. You can specify the file name using the <logfile> argument.
<logfile>	Specifies the full path and name of the text file to which IBM Cognos Data Manager should log progress.
-O	Executes the build in object creation mode.
-P	Specifies that at the end of a build, a keystroke is required before terminating.
-T	Denotes the path and name for expression and script tracing log files.
<tracefile_root>	Specifies the path and name to use for the expression and script tracing log files. For more information, see “Create a Log File for Expression and Script Tracing” on page 264.
-V	Denotes that the parameter is a named variable and value pair.
<name>	Specifies the variable to use during item substitution.
<value>	Specifies the value that the corresponding variable is to use.

Symbol	Description
-X	Specifies the delivery type: • F delivers data using fact deliveries • D delivers data using dimension deliveries

dimbuild

Executes the dimension build that you specify in the command line parameters.

Syntax

```
dimbuild [<catdb_driver> <catdb_items>] <dimbuild_name>
[<options>]
```

When you run dimbuild with this syntax, it executes the named dimension build that resides in the named catalog or package (specified using the option -p).

Symbol	Description
<catdb_driver>	Specifies the database driver to use to connect to the catalog.
<catdb_items>	Specifies the connection string to pass to the database driver to connect to the catalog. For example, to connect as user DBA, with password PASS, to a catalog that resides in an ODBC data source named Sample, you would specify "DSN=Sample;UID=DBA;PWD=PASS"
<dimbuild_name>	Specifies the name of the dimension build to execute within the catalog or package.
<options>	For information, see “dimbuild Options.”

Example

```
dimbuild ODBC "DSN=SalesCat;UID=Sales;PWD=ds123"
Period
```

This command runs dimbuild to execute the dimension build named Period. To retrieve the dimension build definition, dimbuild connects to the ODBC data source named SalesCat as user Sales, with password ds123.

```
dimbuild -p"c:\data files\sales.pkg" Period
```

This example runs dimbuild to execute a dimension build named Period from the package named sales.pkg.

dimbuild Options

Syntax

```
<options> ::=
 [-e<runenv>] [-p<package_file>] [-A<dbalias_file>] [-C]
 [-L<logfile>] [-O] [-P] [-T<tracefile_root>] {-V<name>[=<value>]}
```

Symbol	Description
-e	Denotes the run environment for the dimension build.
<runenv>	Specifies the run environment to apply to identify the correct database alias.
-p	Denotes that the dimension build is executed from a package.
<package_file>	Specifies the name and path of the package file. **Note:** If you also specify catalog details, then logging and audit information is stored in the catalog, even if the catalog has no connection with the package.
-A	Denotes the text file that contains the database alias definitions to use for the dimension build.
<dbalias_file>	Specifies the name and path of the text file that contains database alias definitions to use for the dimension build.
-C	Executes the dimension build in check only mode.
-L	Logs progress to the specified file. You can specify the file name using the <logfile> argument.
-O	Executes the build in object creation mode.
-P	Specifies that at the end of a build, a keystroke is required before terminating.
-T	Denotes the path and name for expression and script tracing log files.
<tracefile_root>	Specifies the path and name to use for the expression and script tracing log files. For more information, see "Create a Log File for Expression and Script Tracing" on page 264.
-V	Denotes that the parameter is a named variable and value pair.
<name>	Specifies the name of the variable to use during item substitution.
<value>	Specifies the value that the corresponding named variable takes.

rundsjob

This command executes a JobStream.

Syntax

```
rundsjob [<catdb_driver> <catdb_items>] <jobstream_name
or uuid> [<options>]
```

When you run rundsjob with this syntax, it executes the named JobStream that resides in the named catalog or package file (specified using the option -p).

Symbol	Description
<catdb_driver>	Specifies the database driver to use to connect to the catalog containing the JobStream.
<catdb_items>	Specifies the connection string to pass to the database driver to connect to the catalog. For example, to connect as user DBA, with password PASS, to a catalog that resides in an ODBC data source named Sample, you would specify `"DSN=Sample;UID=DBA;PWD=PASS"`
<jobstream_name>	Specifies the name of the JobStream to execute within the catalog or package.
<uuid>	Specifies the universally unique identifier for the component to execute.
<options>	For information, see "rundsjob Options."

Example

```
rundsjob ODBC "DSN=Sales;UID=admin" example_jobstream
-R
```

This example runs rundsjob to restart the JobStream named example_jobstream. The IBM Cognos Data Manager catalog resides in the ODBC data source named Sales, to which Data Manager connects as user admin with no password.

```
rundsjob -p"c:\data files\sample.pkg" example_jobstream
```

This example runs rundsjob to execute a JobStream named example_jobstream from the package named sample.pkg.

rundsjob Options

Syntax

```
<options> ::=
 [-e<runenv>] [-p<package_file>] [-A<dbalias_file>] [-C]
[-D]
 [-L<logfile>] [-M] [-N<maxprocesses>] [-P] [-O]
[-R] [-S<seconds>]
 [-T<tracefile_root>] {-V<name>[=<value>]}
```

Symbol	Description
-e	Denotes the run environment for the JobStream.
<runenv>	Specifies the run environment to apply to identify the correct database alias.
-p	Denotes that the JobStream is executed from a package.

Symbol	Description
<package_file>	Specifies the name and path of the package file. **Note:** If you also specify catalog details, then logging and audit information is stored in the catalog, even if the catalog has no connection with the package.
-A	Denotes the text file that contains the database alias definitions to use for the JobStream. **Note:** You cannot combine this option with -D (execute build nodes as individual data movement tasks).
<dbalias_file>	Specifies the name and path of the text file that contains database alias definitions to use for the build.
-C	Executes the JobStream in check only mode.
-D	Executes fact build and dimension build nodes within the JobStream as individual data movement tasks. For more information, see "Configure a JobStream Execution to Use IBM Cognos Business Intelligence Load Balancing" on page 254. **Note:** You cannot combine this option with -A (local external alias files).
-L	Logs progress to the specified file. You can specify the file name using the <logfile> argument.
<logfile>	Specifies the full path and name of the text file to which IBM Cognos Data Manager should log progress.
-M	By default, user messages can only be included in the log file for the JobStream node that is being executed. Using this flag, you can include user messages for a JobStream node in the log file of the master JobStream. You must also specify USER values in the TRACE_VALUES variable for both the JobStream node and the master JobStream. For information, see "TRACE_VALUES" on page 300.
-N	Restricts the number of processes that a JobStream can execute concurrently.
<maxprocesses>	Specifies the number of processes that a JobStream can execute concurrently. Any subsequent master JobStream nodes are suspended until the process count for the currently executing JobStream falls below the <maxprocesses> value.
-P	Specifies that at the end of a JobStream, a keystroke is required before terminating.
-O	Executes builds in the JobStream in object creation mode.
-R	Restarts executing the JobStream at the point of a previous interruption. The JobStream restarts at the earliest node that did not successfully complete.

Symbol	Description
-T	Denotes the path and name for expression and script tracing log files.
<tracefile_root>	Specifies the path and name to use for the expression and script tracing log files. For more information, see “Create a Log File for Expression and Script Tracing” on page 264.
-V	Denotes that the parameter is a named variable and value pair.
<name>	Specifies the name of the variable to use during item substitution.
<value>	Specifies the value that the corresponding named variable takes.

dsremote

This command executes a fact build, dimension build, or JobStream that is located on a remote computer.

Syntax

To use this command, you must have the IBM Cognos Data Manager Network Services server installed and configured on the remote Data Manager server.

For more information, see the IBM Cognos Data Manager *Installation and Configuration Guide*.

```
dsremote [-n] [-P] [-32] [-64] <host> <command> {<command_arg>}
<command> ::=
 rundsjob | databuild | dimbuild
```

Symbol	Description
-n	Specifies that the client should not wait for the output of the remote command, but submit the command and then exit.
-P	Specifies that when remote execution is complete, a keystroke is required before terminating.
-32	Specifies the remote execution to use the 32-bit execution mode.
-64	Specifies the remote execution to use the 64-bit execution mode.
<host>	The computer on which remote execution is to take place.
<command>	Specifies the type of component that dsremote is to execute: • rundsjob executes a JobStream • databuild executes a data build • dimbuild executes a dimension build

Symbol	Description
<command_arg>	Specifies the arguments for the type of component that you specified in <command>. For more information, see "databuild" on page 378, "dimbuild" on page 381, or "rundsjob" on page 382.

Example

```
dsremote Europe dimbuild ODBC DSN=DS_Tutorial
Product
```

This command runs dsremote to execute the dimension build named Product on a remote computer named Europe. To retrieve the dimension build definition, databuild connects to the ODBC data source named DS_Tutorial that is defined on the remote computer.

catlist

This command lists the contents of an IBM Cognos Data Manager catalog. By default, catlist lists all database aliases, fact builds, dimension builds, and reference structures. However, you can list only one type of catalog component (or a single catalog component) by using the appropriate option.

Syntax

```
catlist <catdb_driver_name> <catdb_driver_items>
[<component_type>] [<component_namecomponent_uuid>]
-o<filename>] [-I[C][N][U]] [-P] {-V<name>[=value>]}
<component_type>::=
 A | B | D | F | H | J | K | L | M | S | T
```

Symbol	Description
<catdb_driver_name>	Specifies the database driver to use to connect to the catalog.
<catdb_driver_items>	Specifies the connection string to pass to the database driver. For example, to connect as user DBA, with password PASS, to a catalog in an ODBC data source named Sample, you would specify `"DSN=Sample;UID=DBA;PWD=PASS"`

Symbol	Description
<component_type>	Specifies the type of component that catlist is to list: • A specifies a database alias • B specifies a fact build • D specifies a reference dimension • F specifies a user-defined function • H specifies a hierarchy • J specifies a JobStream • K specifies a metadata conformed dimension • L specifies a lookup • M specifies a metadata collection • S specifies a dimension build • T specifies a template
<component_name>	Specifies the name of the individual component to list.
<component_uuid>	Specifies the universally unique identifier for the component to list.
-o	Denotes that the parameter is the name of a text file to which Data Manager should write the output.
<filename>	Specifies the name and path of the text file to which Data Manager should write the output.
-I	Shows additional information about the specified component: • C shows the business name • N shows the description • U shows the universally unique identifier
-P	Specifies that at the end of listing, a keystroke is required before terminating.
-V	Denotes that the parameter is a named variable and value pair.
<name>	Specifies the name of a variable.
<value>	Specifies the value that the associated variable takes.

catexp

This command exports a catalog item to an IBM Cognos Data Manager file-based specification.

Syntax

```
catexp <catdb_driver_name> <catdb_driver_items>
<export_filename> <component_type> [<component_nameuuid>]
[-a] [-L<logfile>] [-P]  {-V<name>[=<value>]}
<component_type> ::=
 A | B | D | F | H | J | K | L | M | S | T
```

Symbol or option	Description
<catdb_driver_name>	Specifies the database driver to use to connect to the catalog.
<catdb_driver_items>	Specifies the connection string to pass to the database driver to connect to the catalog. For example, to connect as user DBA, with password PASS, to a catalog that resides in an ODBC data source named Sample, you would specify `"DSN=Sample;UID=DBA;PWD=PASS"`
<export_filename>	Specifies the path and name of the file to which you want to export the catalog component.
<component_type>	Specifies the type of component to export: • A specifies a database alias • B specifies a fact build • D specifies a reference dimension • F specifies a user-defined function • H specifies a hierarchy • J specifies a JobStream • K specifies a metadata conformed dimension • L specifies a lookup • M specifies a metadata collection • S specifies a dimension build • T specifies a template
<component_name>	The name of the individual component to export.
<uuid>	Specifies the universally unique identifier for the component to export.
-a	Appends the exported component to the named file.

-L	Logs progress to the specified file. You can specify the file name using the <logfile> argument or you can use the following syntax:
	Microsoft Windows operating system syntax
	`-L %<variable_name>%\<file_name>`
	UNIX operating system syntax
	`-L $<variable_name>/<file_name>`

<logfile>	Specifies the name and path of the text file to which Data Manager should log progress.
-P	Specifies that at the end of the export, a keystroke is required before terminating.
-V	Denotes that the parameter is a named variable and value pair.
<name>	Specifies the name of a variable.

Symbol or option	Description
<value>	Specifies the value that the associated variable takes.

Notes

- If you export an object to a file in which that object already exists, the old and new definitions may conflict to produce an invalid definition. To avoid this, edit the target text file to remove the redundant definition before exporting the new definition to the file.
- Data Manager cannot import from definition files that contain more than one fact build or dimension build. To avoid this, always specify the component name or universally unique identifier when you export a fact build or dimension build.

catimp

This command imports a component from a text file to an IBM Cognos Data Manager catalog.

Syntax

```
catimp <catdb_driver_name> <catdb_driver_items>
<import_filename> <filetype> [<component_name>]
[-d<default_dbalias>] [-o] [-I] [-v<version>] [-r<revision>] [-u]
[-L<logfile>] [-P] {-V<name>[=<value>]}
<filetype> ::=
 B | D | F | J | K | M | R | S
```

Symbol or option	Description
<catdb_driver_name>	Specifies the database driver to use to connect to the catalog.
<catdb_driver_items>	Specifies the connection string to pass to the database driver to connect to the catalog. For example, to connect as user DBA, with password PASS, to a catalog that resides in an ODBC data source named Sample, you would specify "DSN=Sample;UID=DBA;PWD=PASS"
<import_filename>	Specifies the path and name of the text file that defines the component.
<filetype>	Specifies the type of component to import: • B specifies a fact build • D specifies a database alias • F specifies a user-defined function • J specifies a JobStream • K specifies a metadata conformed dimension • M specifies a metadata collection • R specifies a reference structure • S specifies a dimension build

Symbol or option	Description
<component_name>	Specifies the name of the component to import. This is mandatory for JobStreams, fact builds, dimension builds, metadata conformed dimensions, and metadata collections. It should not to be used with reference structures, database aliases, or functions.
-d	Denotes that the parameter is the name of the default database alias that Data Manager should use where none is otherwise specified.
<default_dbalias>	Specifies the name of the default database alias to use where none is otherwise specified.
-o	Overwrites any catalog components that have the same type and name as an imported component.
-I	Causes Data Manager to ignore any parse errors.
-v	Denotes that the parameter is the version number stored against a component.
<version>	Specifies the version number stored against a component. Version numbers follow this format: x.y.z
-r	Denotes that the parameter is the revision number to be stored against a component.
<revision>	Specifies the revision number to be stored against a component. Revision numbers follow this format: x.y.z The revision number is incremented each time the component is modified and saved within Data Manager Designer. **Note:** The revision number cannot be older than the version number.
-u	Indicates an upgrade procedure, in other words, the component being imported is from IBM Cognos DecisionStream Series 7.

-L	Logs progress to the specified file. You can specify the file name using the <logfile> argument or you can use the following syntax:
	Microsoft Windows operating system syntax
	`-L %<variable_name>%\<file_name>`
	UNIX operating system syntax
	`-L $<variable_name>/<file_name>`

<logfile>	Specifies the name and path of the text file to which Data Manager should log progress.
-P	Specifies that at the end of the import, a keystroke is required before terminating.

Symbol or option	Description
-V	Denotes that the parameter is a named variable and value pair.
<name>	Specifies the name of a variable.
<value>	Specifies the value that the associated variable takes.

Example

This command runs catimp to import the build named US_Sales from the text file named Sales.bld, into the catalog that resides in the ODBC data source named SalesCat, overwriting any build with the same name. In the catalog, catimp gives the build US_Sales the caption US Sales. catimp connects to the named data source as the default user, with no password.

```
catimp ODBC "DSN=SalesCat" SalesBld.bld B US_Sales
"US Sales" -o
```

Order of Import

Because IBM Cognos Data Manager validates objects that reside in a catalog, you must import text definitions in dependency order. For example, do not import a build until the catalog contains all the aliases and hierarchies that it uses. Normally, you should import text definitions in this order:

1. Aliases.
2. User-defined functions.
3. Reference dimensions.
4. Templates.
5. Reference structures.
6. Fact builds and dimension builds.
7. JobStreams and metadata dimensions.
8. Metadata collections.

Notes

- If you import an object from a file that defines the object more than once, the resulting definition may be invalid. To avoid this, edit the file to remove the redundant definition before importing from the file.
- Data Manager cannot import from definition files that contain more than one fact build, or dimension build. If a file contains more than one of these components, edit it to remove all but one component before importing from the file.

catbackup

This command performs a backup of an IBM Cognos Data Manager catalog to a text file.

Syntax

```
catbackup <catdb_driver_name> <catdb_driver_items> <filename>
[-L<logfile>] [-P] {-V<name>[=<value>]}
```

Symbol or option	Description
<catdb_driver_name>	Specifies the database driver to use to connect to the catalog.
<catdb_driver_items>	Specifies the connection string to pass to the database driver to connect to the catalog. For example, to connect as user DBA, with password PASS, to a catalog that resides in an ODBC data source named Sample, you would specify `"DSN=Sample;UID=DBA;PWD=PASS"`
<filename>	Specifies the path and name of the text file to which you want to backup the catalog.

-L	Logs progress to the specified file. You can specify the file name using the <logfile> argument or you can use the following syntax:
	Microsoft Windows operating system syntax
	`-L %<variable_name>%\<file_name>`
	UNIX operating system syntax
	`-L $<variable_name>/<file_name>`

<logfile>	Specifies the name and path of the text file to which Data Manager should log progress.
-P	Specifies that at the end of the backup, a keystroke is required before terminating.
-V	Denotes that the parameter is a named variable and value pair.
<name>	Specifies the name of a variable.
<value>	Specifies the value that the associated variable takes.

catrestore

This command restores a backup of an IBM Cognos Data Manager catalog from a text file.

Syntax

```
catrestore <catdb_driver_name> <catdb_driver_items> <filename>
[-L<logfile>] [-P] {-V<name>[=<value>]}
```

Symbol or option	Description
<catdb_driver_name>	Specifies the database driver to use to connect to the catalog.
<catdb_driver_items>	Specifies the connection string to pass to the database driver to connect to the catalog. For example, to connect as user DBA, with password PASS, to a catalog that resides in an ODBC data source named Sample, you would specify `"DSN=Sample;UID=DBA;PWD=PASS"`

Symbol or option	Description
<filename>	Specifies the path and name of the text file that defines the catalog backup.

-L	Logs progress to the specified file. You can specify the file name using the <logfile> argument or you can use the following syntax:
	Microsoft Windows operating system syntax
	`-L %<variable_name>%\<file_name>`
	UNIX operating system syntax
	`-L $<variable_name>/<file_name>`

<logfile>	Specifies the name and path of the text file to which Data Manager should log progress.
-P	Specifies that at the end of the restore, a keystroke is required before terminating.
-V	Denotes that the parameter is a named variable and value pair.
<name>	Specifies the name of a variable.
<value>	Specifies the value that the associated variable takes.

catupgrade

This command upgrades an IBM Cognos Data Manager catalog.

Syntax

```
catupgrade <catdb_driver_name> <catdb_driver_items>
[-c] [-d] [-h] [-l<migrate_level>] [-s<migrate_since_timestamp>]
[-L<logfile>] [-P] {-V<name>[=<value>]}
```

Symbol or option	Description
<catdb_driver_name>	Specifies the database driver to use to connect to the catalog.
<catdb_driver_items>	Specifies the connection string to pass to the database driver to connect to the catalog. For example, to connect as user DBA, with password PASS, to a catalog that resides in an ODBC data source named Sample, you would specify `"DSN=Sample;UID=DBA;PWD=PASS"`
-c	Specifies that the catalog tables should be created, if necessary.
-d	Specifies that the old catalog tables are dropped after a successful upgrade.
-h	Specifies that the run history be migrated.

Symbol or option	Description
-l	Denotes that the parameter is the level of run history to migrate.
<migrate_level>	Specifies the level of run history to migrate: • 1 the minimum. • 2 core run details (default). • 3 full run details and audit trail.
-s	Denotes that the parameter is the date/time from which to migrate the run history.
<migrate_since_timestamp>	Specifies the date/time from which to migrate run history. You specify the date/time using this format YYYY-MM-DD [HH:MI:SS]. By default Data Manager migrates all the available run history.

-L	Logs progress to the specified file. You can specify the file name using the <logfile> argument or you can use the following syntax:
	Microsoft Windows operating system syntax
	`-L %<variable_name>%\<file_name>`
	UNIX operating system syntax
	`-L $<variable_name>/<file_name>`

<logfile>	Specifies the name and path of the text file to which Data Manager should log progress.
-P	Specifies that at the end of the upgrade, a keystroke is required before terminating.
-V	Denotes that the parameter is a named variable and value pair.
<name>	Specifies the name of a variable.
<value>	Specifies the value that the associated variable takes.

catmanage

This command allows you to manage an IBM Cognos Data Manager catalog.

Syntax

```
catmanage <catdb_driver_name> <catdb_driver_items> {email to}
[-C] [-P] [-L<logfile>] [-M<email_profile>] [-s<email_subject>]
[<catalog_options>]
<options> ::=
 -c | -i | -g | -d | -k
```

Symbol or option	Description
<catdb_driver_name>	Specifies the database driver to use to connect to the catalog.

Symbol or option	Description
<catdb_driver_items>	Specifies the connection string to pass to the database driver to connect to the catalog. For example, to connect as user DBA, with password PASS, to a catalog that resides in an ODBC data source named Sample, you would specify `"DSN=Sample;UID=DBA;PWD=PASS"`
emailto	Specifies the name of the email recipient.
-C	Connects to the catalog.
-P	Specifies that a keystroke is required before terminating.

-L	Logs progress to the specified file. You can specify the file name using the <logfile> argument or you can use the following syntax:
	Microsoft Windows operating system syntax
	`-L %<variable_name>%\<file_name>`
	UNIX operating system syntax
	`-L $<variable_name>/<file_name>`

<logfile>	Specifies the name and path of the text file to which Data Manager should log progress.
-M	Specifies that the value is an email profile.
<email_profile>	Specifies the email profile.
-s	Specifies that the value is an email subject.
<email_subject>	Specifies the email subject.
<catalog_options>	Select from the following options: • -c creates the catalog tables. • -i creates indexes for the catalog tables. • -g grants all permissions to all users for all tables in the schema. However, this may not apply to all databases. • -d deletes the contents from all the catalog tables in the schema. • -k drops all the catalog tables, removing the catalog from the database.

catdoc

This command documents the contents of a catalog.

Syntax

```
catdoc <catdb_driver_name> <catdb_driver_items>
<filename>
[-T<file type>][-L<logfile>][-P]{-V<name>[=<value>]}
```

Symbol or option	Description
<catdb_driver_name>	Specifies the database driver to use to connect to the catalog.
<catdb_driver_items>	Specifies the connection string to pass to the database driver. For example, to connect as user DBA, with password PASS, to a catalog that resides in an ODBC data source named Sample, you would specify `"DSN=Sample;UID=DBA;PWD=PASS"`
<filename>	Specifies the path and name of the text file that defines the catalog.
-T	Denotes the path and name for tracing the log files.
<file type>	Specifies the log file type.

-L	Logs the progress to the specified file. You can specify the file name using the <logfile> argument or you can use the following syntax:
	Microsoft Windows operating system syntax
	`-L`
	`%<variable_name>%\<file_name>`
	UNIX operating system syntax
	`-L $<variable_name>\<file_name>`

<logfile>	Specifies the name and path of the text file to which IBM Cognos Data Manager should log progress.
-P	Specifies that when documenting is complete, a keystroke is required before terminating.
-V	Denotes that the parameter is a named variable and value pair.
<name>	Specifies the name of a variable.
<value>	Specifies the value that the associated variable takes.

showref

This command shows details of the members of a reference structure defined in a catalog.

Syntax

```
showref <catdb_driver> <catdb_items> <refdim_name|uuid>
<refdata_name|uuid> [<options>]
```

Symbol	Descriptions
<catdb_driver>	Specifies the database driver to use to connect to the catalog.

Symbol	Descriptions
<catdb_items>	Specifies the connection string to pass to the database driver to connect to the catalog. For example, to connect as user DBA, with password PASS, to a catalog that resides in the ODBC data source named Sample, you would specify "DSN=Sample;UID=DBA;PWD=PASS"
<refdim_name>	Specifies the reference dimension that contains the reference structure.
<uuid>	Specifies the universally unique identifier for the reference dimension that contains the reference structure.
<refdata_name>	Specifies the name of the reference structure.
<uuid>	Specifies the universally unique identifier for the reference structure.
<options>	For information, see "showref Options."

Example

```
showref ODBC "DSN=Example Catalog" Fiscal -b
[PROGRESS - 13:19:45] Accessing catalog
Year -- {Year} 4 members [0 leaf nodes]
    Quarter -- {Quarter} 13 members [1 leaf node]
        Period -- {Period} 36 members [36 leaf nodes]
```

This example runs showref to produce a brief description of the Fiscal hierarchy defined in the catalog that resides in the ODBC data source named Example Catalog.

showref Options

Syntax

```
<options> ::=
 [-e<runenv>] [-A<dbalias_file>]
 [-b] [-h] [-l] [-m<mindepth>] [-O[<output_file>]] [-x<maxdepth>]
 [-F] [-M] [-P] {-V<name>[=<value>]}
```

Symbol or option	Description
-e	Denotes that the parameter is the name of a run environment.
<runenv>	Specifies the name of the run environment to apply when deriving the reference structures.
-A	Denotes that the parameter is the name of a text file that contains the database alias definitions that the reference structures require.

Symbol or option	Description
<dbalias_file>	Specifies the name and path of the text file that contains the database alias definitions that the reference structures require.
-b	Produces a brief summary only.
-h	Specifies that the reference structure is a hierarchy. (default)
-l	Specifies that the reference structure is a lookup. By default, IBM Cognos Data Manager assumes that the reference structure is a hierarchy.
-m	Denotes that the parameter gives the minimum depth (in hierarchy levels) for the listing.
<mindepth>	Specifies the minimum depth (in number of hierarchy levels) to list.
-o	Denotes that the parameter is the name of a text file to which Data Manager should write the reference structure listing.
<output_file>	Specifies the name and path of the text file to which Data Manager should write the reference structure listing.
-x	Denotes that the parameter gives the maximum depth (in hierarchy levels) for the listing.
<maxdepth>	Specifies the maximum depth (in number of hierarchy levels) to list.
-F	Removes any unused foster parents from the reference structure listing.
-M	Reports all messages when caching the reference data item. This overrides any reporting limit set on the Features tab of the reference structure Properties window.
-P	Specifies that a keystroke is required before terminating.
-V	Denotes that the parameter is a named variable and value pair.
<name>	Specifies the name of the variable to use during item substitution.
<value>	Specifies the value that the corresponding named variable takes.

dstofm

This command exports a conformed model description to an XML file. This file can then be used to produce a model of your target data mart or data warehouse in IBM Cognos Framework Manager, without the need to recreate the complete model.

Syntax

```
dstofm <catdb_driver> <catdb_items> <output_file> {<collection_name>}
[-D<description] [-N<name>] [-P] [-U] {-V<name>[=<value>]}
```

Symbol or option	Description
<catdb_driver>	Specifies the database driver to use to connect to the catalog.
<catdb_items>	Specifies the connection string to pass to the database driver.
<output_file>	Specifies the name and path of the metadata export file (.xml).
<collection_name>	Specifies the names of the metadata collections to export.
-D	Denotes a description for the exported metadata file.
<description>	Specifies the description for the exported metadata file.
-N	Denotes the model name for the exported metadata file.
<name>	Specifies the model name for the metadata export.
-P	Specifies that at the end of a build, a keystroke is required before terminating.
-U	Specifies that unreferenced metadata dimensions are to be included.
-V	Denotes that the parameter is a named variable and value pair.
<name>	Specifies the variable to use during item substitution.
<value>	Specifies the value that the corresponding variable is to use.

Example

```
dstofm ODBC "DSN=Example Catalog" ExampleExport.xml -N ExampleModel
```

This example runs dstofm to export the example conformed model description that resides in the Example Catalog, to the metadata export file named ExampleExport.xml.

sqlterm

This command starts the SQLTerm application (the IBM Cognos Data Manager terminal for SQL databases). Using SQLTerm, you can

Syntax

- connect to and query SQL databases
- obtain information about database objects, DBMS drivers, and data sources
- run SQL scripts

You can work with SQLTerm in either of these modes:

- from the command line to execute an SQL script

- in the SQLTerm shell

```
sqlterm [<driver_name> <item_string>]
[-d<dbalias>] [-e<runenv>] [-A<dbalias_file>]
[-f<sql_file>] [-o<output_file>] [-w<num_width>]
[-C] [-E] [-P] {-V<name>[=<value>]}
```

Symbol	Description
<driver_name>	Specifies the name of the DBMS driver to use to connect to the data.
<item_string>	Specifies the connection string to pass to the DBMS driver to establish the data connection.
-d	Denotes the default database alias.
<dbalias>	Specifies the name of the alias that defines the database to which Data Manager should connect. For more information, see "Use an Alias to Connect to a Database" on page 44.
-e	Denotes that the parameter is the name of a Data Manager run environment context.
<runenv>	Specifies the run environment context to use to determine the database to which each alias refers.
-A	Denotes the text file that contains the database alias.
<dbalias_file>	Specifies the name and path of the file that defines database connection parameters for each alias and run environment.
-f	Denotes that the parameter is the name of a script file.
<sql_file>	Specifies the path and name of the script file that SQLTerm should execute. When you choose this option, SQLTerm executes the script and then terminates.
-o	Denotes that the parameter is the name of a text file to which Data Manager should write the output.
<output_file>	Specifies the name and path of the text file to which Data Manager should write the output.
-w	Denotes that the parameter is the width at which SQLTerm should wrap its output.
<num_width>	Specifies the width in number of characters at which SQLTerm should wrap its output.
-C	Specifies that Cognos SQL is to be the default.
-E	Causes SQLTerm to echo SQL statements to the console immediately before it executes them.

Symbol	Description
-P	Specifies that a keystroke is required before terminating.
-V	Denotes that the parameter is a variable name and value pair.
<name>	Specifies the name of the variable.
<value>	Specifies the value that the named variable takes.

Example

- `sqlterm ODBC "DSN=Sales;UID=user;PWD=pass"`

  This example runs SQLTerm to automatically connect to the ODBC data source named Sales as a user named user with the password pass.
- `sqlterm -fmyscript.sql`

  This example runs SQLTerm to execute the script named myscript.sql. SQLTerm terminates when it has completed processing the script.
- `sqlterm -dSales -eDEVELOP -Astdalias.def`

  This example runs SQLTerm to connect to the database listed in the file named stdalias.def as alias Sales for the run environment context DEVELOP.

For information on SQL scripts, see "Work with SQL Scripts" on page 409.

Working in the SQLTerm Shell

SQLTerm makes available the following commands:

- ! (system command)
- COGNOS SQL
- CONNECT
- DBALIAS
- DISCONNECT
- DESCRIBE
- DIFF
- DRIVERS
- ECHO
- EXIT
- GO
- HELP
- ISOLEVEL
- NATIVE SQL
- NUMBER
- OBJECTS
- OUTPUT
- PROGRESS
- PROMPT
- QUIT
- SCHEMA

- SCREEN
- SCRIPT
- SOURCES
- STATISTICS
- TIME
- TYPES

! (System Command)

Use this to issue an operating system command in the SQLTerm shell.

Syntax

```
!<syscmd>
```

Symbol	Description
<syscmd>	Specifies the operating system command to execute.

Example

```
!dir
```

This example executes the Microsoft Windows operating system dir command to list the contents of the current directory.

COGNOS SQL

Use this command to use Cognos SQL rather than native SQL.

Syntax

```
COGNOS SQL
```

CONNECT

Use this command to directly connect to a data source. SQLTerm terminates any existing connection before establishing the connection to the required data source.

Syntax

```
CONNECT <driver> <items>
```

Symbol	Description
<driver>	Specifies the name of the DBMS driver to use to connect to the data.
<items>	Specifies the connection string that to pass to the DBMS driver to establish the connection.

Example

```
CONNECT ODBC DSN=Sales;UID=user;PWD=pass;
```

This example instructs SQLTerm to connect to the ODBC data source named Sales as a user named user with password pass.

DBALIAS

Use this command to connect to a data source using a database alias. SQLTerm terminates any existing connection before establishing the connection to the required data source.

Syntax

```
DBALIAS <dbalias>
```

Symbol	Description
<dbalias>	Specifies the name of the alias to which SQLTerm should connect.

To use this command, you must pass the name of a valid database alias definition file to SQLTerm. You can either do this using the -A command line option, or by setting the DS_DBALIAS_FILE environment variable. If you require a run environment context other than <DEFAULT>, you must also pass the run environment name by using the -e command line option or by setting the DS_RUNENV environment variable.

Example

```
DBALIAS Sales
```

This example instructs SQLTerm to connect to the database to which the alias Sales refers.

DISCONNECT

Use this command to terminate an existing connection to a database.

Syntax

```
DISCONNECT
```

DESCRIBE

On its own, the DESCRIBE command lists the tables in the current database.

Syntax

If you supply a valid table name, SQLTerm details each column in the table. Where possible, these are the column name, its data type and length, whether the column can accept null values, and whether the column forms part of the primary key of the table.

```
DESCRIBE [<table_name>]
```

Symbol	Description
<table_name>	Specifies the name of a table in the currently connected database.

Example

This example lists the tables in the currently connected database

```
: DESCRIBE
Table Name
==============================
ConversionRate
Country
EuroConversion
OrderDetail
OrderHeader
OrderMethod
Product
ProductForecast
ProductLine
ProductType
ReturnReason
ReturnedItem
SalesBranch
SalesStaff
SalesTarget
15 tables
```

DIFF

This command lists the differences between two tables that you specify. Both tables must reside in the database to which sqlterm is connected.

Syntax

```
DIFF <tab 1> <tab 2>
```

Symbol	Description
<table_1>	Specifies the first of the two tables to compare.
<table_2>	Specifies the second of the two tables to compare.

DRIVERS

This command lists the DBMS drivers for which IBM Cognos Data Manager is configured. Use the ALL argument to obtain a list of all the drivers known to Data Manager.

Syntax

```
DRIVERS [ALL]
```

ECHO

ECHO ON performs the same function as the -E command line argument. It causes SQLTerm to echo SQL statements to the console immediately before it executes them. ECHO OFF prevents SQLTerm from echoing SQL statements to the console. On its own, ECHO returns its status (either ON or OFF).

Syntax

This command is particularly useful if you use redirection to provide the standard input for SQLTerm. ECHO is OFF by default when SQLTerm takes input from the standard input. When you type an SQL statement at the keyboard, it appears on screen as you type and does not need to be echoed. However, input that is redirected from a text file does not appear on screen. Setting ECHO to ON allows you to see the statements that SQLTerm executes.

```
ECHO [ON | OFF]
```

EXIT

This command causes SQLTerm to terminate.

Syntax

```
EXIT
```

Synonyms for this command are QUIT, EX, and \q.

FETCHSIZE

This command sets or shows the fetch row count used to query the database.

Syntax

```
FETCHSIZE <N>
```

Symbol	Description
<N>	This setting controls the number of rows that are queried at once by using the bulk fetch facility for your database. You can use this to tune large extractions from a source database. This setting has no effect for databases that do not support bulk fetch operations. The default fetch row size is 50.

GO

This command causes SQLTerm to execute an SQL statement.

Syntax

Synonyms for the GO command are ;, \p\g, and \g. For example, `SELECT * FROM product;`

```
<statement>GO
```

Symbol	Description
<statement>	Any valid SQL statement.

Example

```
SELECT DISTINCT product_id, product_name FROM productsGO
```

This example instructs SQLTerm to execute the SQL statement, SELECT DISTINCT product_id, product_name FROM product.

HELP

This command returns a list of available SQLTerm commands and a brief description of each.

Syntax

```
HELP
```

ISOLEVEL

This command sets or shows the database isolation level defined for the database transaction you are using to query the data source.

Syntax

```
ISOLEVEL <N>
```

Symbol	Description
<N>	This setting can be used where you specifically want to control the transaction isolation level for a query transaction. This setting has no effect for database systems that do not support transaction isolation levels. The default value is determined by the source database system. The values you can set are • 0 (DBMS default) • 1 (Read uncommitted) • 2 (Read committed) • 3 (Cursor stability) • 4 (Repeatable read) • 5 (Phantom protection) • 6 (Serializable)

NUMBER

Use this command to set or show the width at which SQLTerm displays columns of numeric data.

Syntax

```
NUMBER [<width>]
```

Symbol	Description
<width>	Specifies the width in number of characters, at which SQLTerm should show columns of numeric data.

OBJECTS

Use this command to list objects in the database to which SQLTerm is currently connected.

Syntax

```
OBJECTS [T][V][S][*] [<schema>]
```

Symbol	Description
T	Lists tables.
V	Lists views.
S	Lists schemas.
*	Lists all objects in the database. (default)

Symbol	Description
<schema>	Specifies the name of the schema from which you want to list objects. If you omit this, sqlterm lists objects from all available schemas.

NATIVE SQL

Use this command to specify that native SQL is to be used rather than Cognos SQL.

Syntax

```
NATIVE SQL
```

OUTPUT

Use this command to determine whether SQLTerm should show the data that results from SQL SELECT statements.

Syntax

```
OUTPUT [ON | OFF]
```

On its own, this command returns the OUTPUT setting (either ON or OFF). OUTPUT ON causes SQLTerm to show data; OUTPUT OFF causes SQLTerm to suppress data display. This command is useful if you want to obtain a row count for a large data table.

Example

```
: DBALIAS 'Sales'
Connected to ODBC
: OUTPUT OFF
OUTPUT is OFF
: SELECT * FROM ds_sales;
10529 rows selected
:
```

PROGRESS

This command shows select progress every n rows.

Syntax

```
PROGRESS <N>
```

Symbol	Description
<N>	Specifies the number of rows.

PROMPT

This command shows a line of text.

Syntax

```
PROMPT <text>
```

Symbol	Description
<text>	Specifies the line of text to show. If the text contains spaces, then you must enclose the text in double quotation marks.

QUIT

This command causes SQLTerm to terminate.

Syntax

```
QUIT
```

SCHEMA

This command lists the schemas and user names from the current connection.

Syntax

```
SCHEMA
```

SCREEN

This command sets the width in number of characters, at which SQLTerm wraps its output. On its own, it returns the current width.

Syntax

```
SCREEN [<width>]
```

SCRIPT

This command causes SQLTerm to execute an SQL script.

Syntax

```
SCRIPT <filename>
```

For information about working with SQL scripts, see “Work with SQL Scripts” on page 409.

SOURCES

This command causes SQLTerm to return a list of available data sources.

Syntax

SOURCES lists data sources that are available using the specified driver. If you specify the optional ALL argument, SOURCES lists all data sources that are available using any DBMS driver.

```
SOURCES [<driver>ALL]
```

Symbol or option	Description
<driver>	Specifies the DBMS driver for which SOURCES should list available data sources. By default this is the current driver.
ALL	Specifies that SOURCES should return a list of all data sources whatever their driver.

STATEMENT

This command indicates the type of the next SQL statement.

Syntax

```
STATEMENT [SELECT|DML|DDL]
```

Symbol or option	Description
SELECT	Specifies that the next SQL statement is a SELECT statement.
DML	Specifies that the next SQL statement is a DML statement.
DDL	Specifies that the next SQL statement is a DDL statement.

STATISTICS

This command determines whether SQLTerm returns timing statistics after executing each SQL statement.

Syntax

On its own, STATISTICS returns its status (either ON or OFF). STATISTICS ON causes SQLTerm to return timing statistics after it executes each SQL statement. STATISTICS OFF turns this off.

This command is particularly useful in determining SQL statement efficiency.

```
STATISTICS [ON|OFF]
```

TIME

This command shows the system date and time.

Syntax

```
TIME
```

TYPES

This command shows information about the current connection. For ODBC, it also shows information about the data type conversion.

Syntax

```
TYPES
```

Work with SQL Scripts

You can create SQL scripts in any text editor. In addition to SQL statements, SQL scripts can contain any valid SQLTerm commands. You can run them on the command line by using the -f argument. You can also run them from the SQLTerm shell by using the SCRIPT command.

The following script connects to an ODBC data source, returns the number of data rows for a data table, disconnects from the data source, and terminates SQLTerm.

```
CONNECT ODBC 'DSN=GOSales'
SELECT COUNT(*) FROM SalesStaff;
DISCONNECT
EXIT
```

If the script is named example.sql then the following example runs it from the command line:

```
sqlterm -fexample.sql
```

If the script is named example.sql, then this example runs it from the SQLTerm shell:

```
SCRIPT example.sql
```

Redirection

Most operating systems support redirection. It allows you to specify that a program should take its input from a text file, rather than from the keyboard. You can also use redirection to send the output to a text file, instead of the computer monitor. On Microsoft Windows and UNIX operating systems, the syntax for this is:

```
sqlterm [<command_options>] [< <scriptname>] [> <outputfile>]
```

Symbol	Description
<command_options>	Specifies the command line options for SQLTerm. For more information, see "sqlterm" on page 399.
<scriptname>	Specifies the name and path of the file from which the operating system should supply input to SQLTerm.
<outputfile>	Specifies the name and path of the text file to which the operating system should send output from SQLTerm.

Example

```
sqlterm -E < sample.sql > results.log
```

This example runs SQLTerm with ECHO on, taking its input from the file named sample.sql, and saving the output to results.log.

Note: By default, ECHO is OFF when you use redirection. This means that SQL statements do not appear in the output stream. If you want the SQL statements to appear in the output stream, you should use the -E command line option. Alternatively, you can issue an ECHO ON command in the script.

catpkgtoxml

This command converts an IBM Cognos Data Manager package file to a data movement specification XML file.

Syntax

```
catpkgtoxml <pkgfile> <xmlfile> [<defaultRunnableName>]
[-d|-f|-j] [-P] {-V<name>[=<value>]}
```

Symbol or option	Description
pkgfile	Specifies the name and path of the package file that you want to convert.

Symbol or option	Description
xmlfile	Specifies the name and path of the XML file that you want to create.
defaultRunnableName	Specifies the default component to run for the data movement specification.
-d	Denotes that a dimension build should run.
-f	Denotes that a fact build should run.
-j	Denotes that a JobStream should run.
-P	Specifies that a keystroke is required before terminating.
-V	Denotes that the parameter is a variable name and value pair.
<name>	Specifies the name of the variable.
<value>	Specifies the value that the named variable takes.

Work with Catalogs

Using catalog-based projects has these advantages over using text-based projects:

- Your projects can access user-defined variables.
- You can use JobStreams to integrate catalog-based projects and other IBM Cognos Data Manager objects into a logical process.
- You can use auditing.

To work with catalog-based projects directly in the Data Manager language, you must use the catexp utility to export definitions from the catalog to text files, modify the definitions in a text editor, and use the catimp utility to import the definitions back to the catalog. The order in which you export definitions from a catalog is unimportant. However, because Data Manager validates catalog-based objects, you must import the definitions in dependency order. For more information, see “Order of Import” on page 391.

You can also work directly with package files without requiring that they be imported into a host catalog before executing a build or JobStream. The databuild, dimbuild, and rundsjob commands have command line options to allow a build or uuid to be specified and the package treated as the complete store of components to be executed. This can be important when executing builds and JobStreams from other applications when it is not possible to have the required relational database in place to hold the catalog components.

Appendix A. Fact Delivery Module Properties

In IBM Cognos Data Manager you use delivery modules to deliver the transformed data to the specified data repositories. Data Manager organizes the delivery modules into these groups:

- fact delivery modules that deliver data produced by a fact build
- dimension delivery modules that deliver data describing a single dimension

For more information on setting up delivery modules, see Chapter 16, "Delivering Fact Data Using a Fact Build," on page 179 and Chapter 17, "Delivering Dimension Data Using a Fact Build," on page 199.

Each fact delivery module has different module properties, and may also have element properties that you can set.

Note: Your Data Manager license may not include all the delivery modules described. Please contact your Data Manager software vendor to extend your license for additional delivery modules.

DB2 LOAD Delivery Module

The DB2 LOAD delivery module delivers data directly to IBM DB2 databases using the DB2 bulk loader.

Note: The bulk loader command for DB2 is designed for the DB2UDB loader for Microsoft Windows and UNIX operating systems. To deliver output for bulk loading into DB2/400 or DB2/MVS, in the Operation box, select Create Loader Files.

Module Properties

The table in this section describes the module properties.

Property	Description
Operation	Specifies whether IBM Cognos Data Manager should • **Run Loader** to deliver the text data and control files, load the data into the database, and clean up the control files and any other intermediate files. (default) • **Run Loader (leave files)** to deliver the text data and control files, load the data into the database, but leave the control files and any other intermediate files so that you can view them. • **Create Loader Files** to deliver the text data and control files, but not invoke the loader to load the data into the database. You should use this option to deliver output for bulk loading into DB2/400 or DB2/MVS. The control file generated uses options specific to DB2 UDB for Microsoft Windows and UNIX operating systems and generates a machine portable text data file.

Property	Description
File directory	Specifies the location in which to load data. By default, Data Manager loads data into the Data directory. If Data Manager is installed on a different computer to the database, and **Load from Client file** is not selected, then you must specify a shared server location. For example, on Windows use Uniform Naming Convention (UNC): \\*ComputerName**SharedFolder*
Filename prefix	If **Create Loader Files** is selected in the **Operation** box, specify the prefix to use to generate the file names. If this box is empty, a default value is used.
Dump log file	Specifies whether to print the log file that the loader generates.
No row count check	Specifies whether to check for successful delivery of the data. Clear this check box to prevent an error if the target DBMS does not report successful delivery in the expected format. Select this check box to use a newer version of the DBMS where the vendor changed the format of the completion message.
Log file success message	If **No row count check** is not selected, this property is used to determine whether the load was successful by searching for the specified substring in the log file or output file. Data Manager searches for this string and the number of rows delivered as substrings in each line of the log file or output file.
Use named pipes	Specifies that Data Manager should use named pipes where possible. By default, Data Manager generates an intermediate text file and runs the bulk loader on it after delivering all data.
Field delimiter	Specifies the character sequence to use to separate fields. Specify a character string or use one of these named characters: Tab (the default), Comma, Space, None, or NL (new line).
Load mode	Specifies one of these load modes: • **Insert** inserts the data in the target table. (default) • **Replace** deletes the contents of the target table but not the indexes before delivering the data. This is equivalent to a DELETE FROM statement.
Data file encoding	Specifies the data file encoding to use.
Load from client file	Specifies that data should be loaded from a client file indicating that the File directory path is relative to the database loader (client) rather than to the database server.
Savecount	An integer that specifies the number of rows at which the loader should establish a consistency point. This allows a failed load to be restarted from the last consistency point. The default value is zero, which means that no consistency points are established, unless necessary.

Property	Description
Warningcount	An integer that specifies the maximum number of warnings that can occur before the loader stops. The default is zero which means that the loader does not stop.
Additional load options	Specifies a string of extra load options to append to the command.
DB2 CLP command	Specifies the path and command for the DB2 loader process. **Note:** You must specify the exact path name and command, including any double quotation marks required. For more information, see “Specify Executables to Deliver Data” on page 462.

Element Properties

For this delivery module, the element properties are the same for all element types.

Property	Description
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example, to use a zero value enter 0
Perform escape processing	Specifies whether to use the escape character (\) before all backslash, tab, and new line characters in string values to ensure that the delivery module correctly interprets the statement.
Floating point scale	Specifies the number of decimal places for a numeric value that must be converted to text. By default, this value is three decimal places.

Essbase Direct Feed Delivery Module

The Essbase Direct Feed delivery module delivers data directly to an Oracle Essbase server.

If the Essbase server stores the data in an IBM DB2 database, this module gives automatic support for DB2 OLAP.

Module Properties

The table in this section describes the module properties.

Property	Description
Server login	Specifies the Essbase login details. This value is mandatory.

Property	Description
Application: database	Specifies the Essbase application and database to which you want to connect. This value is mandatory.
Rows per update	Specifies the number of rows to be written to Essbase before changes become permanent.
Clear existing data	Specifies whether to delete any existing data from the Essbase cube before starting the load process.
Calculate after load	Specifies the Essbase calculation to run on completion of the load process.
Update outline	Specifies whether to update the Essbase outline throughout the load process.

Element Properties

The element properties available depend on the type of element being defined. Note that attribute elements are not available for this delivery module.

Property	Description
Member name expression	An expression that specifies the member attribute to use as the Essbase member name. By default, IBM Cognos Data Manager uses ID. This value is an expression, for example, `concat(CAPTION,'(',ID,')')` This value is mandatory.
Member alias expression	An expression that specifies the member attribute to use as the Essbase member alias.
Dimension name	Specifies the name of the Essbase dimension that corresponds to the Data Manager dimensions.
Full outline update	Specifies whether to completely rebuild the Essbase outline for a dimension.
Member name	Specifies the name of the member in the measures dimension.
Floating point scale	Specifies the number of decimal places for a numeric value that must be converted to text. By default, this value is three decimal places.

Essbase File Feed Delivery Module

The Essbase File Feed delivery module delivers data to flat files that may then be loaded into Oracle Essbase.

If the Essbase server stores the data in an IBM DB2 database, this module gives automatic support for DB2 OLAP.

Module Properties

Property	Description
Data file	Specifies the name of the file to hold the data for loading into Essbase. This value is mandatory.
Generate dimension build files	Specifies whether to generate Essbase dimension files in addition to the fact data.

Element Properties

The element properties that are available depend on the type element being defined. Note that attribute elements are not available for this delivery module.

Property	Description
Member name expression	An expression that specifies the member attribute to use as the Essbase member name. By default, IBM Cognos Data Manager uses ID. This value is mandatory.
Member alias expression	An expression that specifies the member attribute to use as the Essbase member alias.
Dimension name	Specifies the name of the Essbase dimension that corresponds to the Data Manager dimensions.
Member name	Specifies the name of the member in the measures dimension.
Floating point scale	Specifies the number of decimal places for a numeric value that must be converted to text. By default, this value is three decimal places.

Informix LOAD Delivery Module

The Informix LOAD delivery module delivers data directly to an Informix database using the Informix DBACCESS command.

Module Properties

The table in this section describes the module properties.

Property	Description
Operation	Specifies whether IBM Cognos Data Manager should • **Run Loader** to deliver the text data and control files, load the data into the database and clean up the control files and any other intermediate files. (default) • **Run Loader (leave files)** to deliver the text data and control files, load the data into the database but leave the control files and any other intermediate files so that you can view them. • **Create Loader Files** to deliver the text data and control files, but not invoke the loader to load the data into the database.

Property	Description
File directory	Specifies the location in which to load data. By default, Data Manager loads data into its Data directory. **Tip:** To view the Data directory, from the **Tools** menu, click **Browse Data Files**.
Filename prefix	If **Create Loader Files** is selected in the **Operation** box, specify the prefix to use to generate the file names. If this box is empty, a default value is used.
Dump log file	Specifies whether to print the log file that the bulk loader generates.
No row count check	Specifies whether to check for successful delivery of the data. Clear this check box to prevent an error if the target DBMS does not report successful delivery in the expected format. Select this check box to use a newer version of the DBMS where the vendor changed the format of the completion message.
Log file success message	If **No row count check** is not selected, this property is used to determine whether the load was successful by searching for the specified substring in the log file or output file. Data Manager searches for this string and the number of rows delivered as substrings in each line of the log file or output file.
Use named pipes	Specifies that Data Manager should use named pipes where possible. By default, Data Manager generates an intermediate text file and runs the bulk loader on it after delivering all data.
Load mode	Specifies one of these load modes: • **Append** appends the data to the target table. (default) • **Truncate** deletes the contents of the target table together with indexes, before data delivery. This is equivalent to a DELETE FROM statement but is usually faster. • **Replace** deletes the contents of the target table but not the indexes before data delivery. This is equivalent to a DELETE FROM statement.
Informix 'dbaccess' command	Specifies the path and the command for the Informix loader process. **Note:** You must specify the exact path name and command, including any double quotation marks required. For more information, see “Specify Executables to Deliver Data” on page 462.

Element Properties

For this delivery module, the element properties are the same for all element types.

Property	Description
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example, to use a zero value, enter 0
Perform escape processing	Specifies whether to use the escape character (\) before all backslash, tab, and new line characters in string values to ensure that the delivery module correctly interprets the statement.
Floating point scale	Specifies the number of decimal places for a numeric value that must be converted to text. By default, this value is three decimal places.

Microsoft SQL Server BCP Command Delivery Module

The Microsoft SQL Server BCP Command delivery module delivers data directly to Microsoft SQL Server databases using the Microsoft SQL Server BCP bulk load utility.

The BCP command utility is a component of the Microsoft SQL Server client installation. It provides users with an intermediate load file that is passed to the BCP utility.

Note: To load to Microsoft SQL Server without using an intermediate load file use the SQL Server Bulk Copy via API delivery module.

Module Properties

The table in this section describes the module properties.

Property	Description
Operation	Specifies whether IBM Cognos Data Manager should • **Run Loader** to deliver the text data and control files, load the data into the database and clean up the control files and any other intermediate files. (default) • **Run Loader (leave files)** to deliver the text data and control files, load the data into the database but leave the control files and any other intermediate files so that you can view them. • **Create Loader Files** to deliver the text data and control files, but not to invoke the loader to load the data into the database.

Property	Description
File directory	Specifies the location in which to load data. By default, Data Manager loads data into its Data directory. **Tip:** To view the Data directory, from the **Tools** menu, click **Browse Data Files**.
Filename prefix	If **Create Loader Files** is selected in the **Operation** box, specify the prefix to use to generate the file names. If this box is empty, a default value is used.
Dump log file	Specifies whether to print the log file that the bulk loader generates.
No row count check	Specifies whether to check for successful delivery of the data. Clear this check box to prevent an error if the target DBMS does not report successful delivery in the expected format. Select this check box to use a newer version of the DBMS where the vendor changed the format of the completion message.
Log file success message	If **No row count check** is not selected, this property is used to determine whether the load was successful by searching for the specified substring in the log file or output file. Data Manager searches for this string and the number of rows delivered as substrings in each line of the log file or output file.
Record delimiter	Specifies the character sequence to use to separate records. Specify a character string or use one of these: Tab, Comma, Space, None, NL (new line), CRLF (carriage return and line feed combination) on a Microsoft Windows operating system, or CR (carriage return) on a UNIX operating system. NL is the default.
Field delimiter	Specifies the character sequence to use to separate fields. Specify a character string or use one of these named characters: Tab (the default), Comma, Space, None, or NL (new line).
Load mode	Specifies one of these load modes: • **Append** appends the data to the target table. (default) • **Truncate** deletes the contents of the target table, and drops all indexes, before delivering the data. This is equivalent to a DELETE FROM statement, but is usually faster. • **Replace** deletes the contents of the target table but not the indexes, before delivering the data. This is equivalent to a DELETE FROM statement. • **Insert** inserts the data in the target table.
Data file encoding	Specifies the data file encoding to use.
Treat empty string as null value	Specifies whether empty string values should be read as null values.

Property	Description
SQL Server BCP command	Specifies the path and the command for the loader process. **Note:** You must specify the exact path name and command, including any double quotation marks required. For more information, see "Specify Executables to Deliver Data" on page 462.
Additional command line options	Allows additional BCP command line options to be specified.

Element Properties

For this delivery module, the element properties are the same for all element types.

Property	Description
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example, to use a zero value, enter 0
Floating point scale	Specifies the number of decimal places for a numeric value that must be converted to text. By default, this value is three decimal places.

Netezza NZLoad Delivery Module

The Netezza NZLoad delivery module delivers data directly to a Netezza database using the NZLoad utility.

Module Properties

The table in this section describes the module properties.

Property	Description
Operation	Specifies whether IBM Cognos Data Manager should • **Run Loader** to deliver the text data and control files, load the data into the database and clean up the control files and any other intermediate files. (default) • **Run Loader (leave files)** to deliver the text data and control files, load the data into the database but leave the control files and any other intermediate files so that you can view them. • **Create Loader Files** to deliver the text data and control files, but not to invoke the loader to load the data into the database.

Property	Description
File directory	Specifies the location in which to load data. By default, Data Manager loads data into its Data directory. **Tip:** To view the Data directory, from the **Tools** menu, click **Browse Data Files**.
Filename prefix	If **Create Loader Files** is selected in the **Operation** box, specify the prefix to use to generate the file names. If this box is empty, a default value is used.
Dump log file	Specifies whether to print the log file that the bulk loader generates.
No row count check	Specifies whether to check for successful delivery of the data. Clear this check box to prevent an error if the target DBMS does not report successful delivery in the expected format. Select this check box to use a newer version of the DBMS where the vendor changed the format of the completion message.
Log file success message	If **No row count check** is not selected, this is used to determine whether the load was successful by searching for the specified substring in the log file or output file. Data Manager searches for this string and the number of rows delivered as substrings in each line of the log file or output file.
Use named pipes	Specifies that Data Manager should use named pipes where possible. By default, Data Manager generates an intermediate text file and runs the bulk loader on it after delivering all data.
Record delimiter	Specifies the character sequence to use to separate records. Specify a character string or use one of these: Tab, Comma, Space, None, NL (new line), CRLF (carriage return and line feed combination) on a Microsoft Windows operating system, or CR (carriage return) on a UNIX operating system. NL is the default.
Field delimiter	Specifies the character sequence to use to separate fields. Specify a character string or use one of these named characters: Tab (the default), Comma, Space, None, or NL (new line).
Load mode	Specifies one of these load modes: • **Append** appends the data to the target table. (default) • **Truncate** deletes the contents of the target table and indexes, before delivering the data. This is usually faster than a DELETE FROM statement. • **Replace** deletes the contents of the target table but not the indexes, before delivering the data. This is equivalent to a DELETE FROM statement. • **Insert** inserts the data in the target table. For more information, see the bulk loader documentation.
Host	Specifies the server name for the database connection.

Property	Description
Database	Specifies the database name.
Data file encoding	Specifies the data file encoding to use.
Use quotes	Specifies whether to use quotation marks to enclose data values. By default, no quotation marks are used.
Quote value	Specifies the type of quotation character to use for enclosing data values.
Null value indicator	Specifies the value to represent null data values. By default, this value is a blank string.
Additional command line options	Allows additional Netezza command line options to be specified.
Loader command	Specifies the path and the command for the loader process. **Note:** You must specify the exact path name and command, including any double quotation marks required. For more information, see “Specify Executables to Deliver Data” on page 462.

Element Properties

For this delivery module, the element properties are the same for all element types.

Property	Description
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example, to use a zero value, enter 0
Perform escape processing	Specifies whether to use the escape character (\) before all backslash, tab, and new line characters in string values to ensure that the delivery module correctly interprets the statement.
Floating point scale	Specifies the number of decimal places for a numeric value that must be converted to text. By default, this value is three decimal places.

ORACLE SQL*Loader Delivery Module

The ORACLE SQL*Loader delivery module delivers data directly to an Oracle database using the Oracle Bulk Load utility.

Module Properties

The table in this section describes the module properties.

Property	Description
Operation	Specifies whether IBM Cognos Data Manager should • **Run Loader** to deliver the text data and control files, load the data into the database and clean up the control files and any other intermediate files. (default) • **Run Loader (leave files)** to deliver the text data and control files, load the data into the database but leave the control files and any other intermediate files so that you can view them. • **Create Loader Files** to deliver the text data and control files, but not to invoke the loader to load the data into the database.
File directory	Specifies the location in which to load data. By default, Data Manager loads data into its Data directory. **Tip:** To view the Data directory, from the **Tools** menu, click **Browse Data Files**.
Filename prefix	If **Create Loader Files** is selected in the **Operation** box, specify the prefix to use to generate the file names. If this box is empty, a default value is used.
Dump log file	Specifies whether to print the log file that the bulk loader generates.
No row count check	Specifies whether to check for successful delivery of the data. Clear this check box to prevent an error if the target DBMS does not report successful delivery in the expected format. Select this check box to use a newer version of the DBMS where the vendor changed the format of the completion message.
Log file success message	If **No row count check** is not selected, this is used to determine whether the load was successful by searching for the specified substring in the log file or output file. Data Manager searches for this string and the number of rows delivered as sub-strings in each line of the log file or output file.
Use named pipes	Specifies that Data Manager should use named pipes where possible. By default, Data Manager generates an intermediate text file and runs the bulk loader on it after delivering all data.
Record delimiter	Specifies the character sequence to use to separate records. Specify a character string or use one of these: Tab, Comma, Space, None, NL (new line), CRLF (carriage return and line feed combination) on a Microsoft Windows operating system, or CR (carriage return) on a UNIX operating system. NL is the default.
Field delimiter	Specifies the character sequence to use to separate fields. Specify a character string or use one of these named characters: Tab (the default), Comma, Space, None, or NL (new line).

Property	Description
Load mode	Specifies one of these load modes: • **Append** appends the data to the target table. (default) • **Truncate** deletes the contents of the target table and indexes, before delivering the data. This is usually faster than a DELETE FROM statement. • **Replace** deletes the contents of the target table but not the indexes, before delivering the data. This is equivalent to a DELETE FROM statement. • **Insert** inserts the data in the target table. For more information, see the bulk loader documentation.
Data file encoding	Specifies the data file encoding to use.
Control file encoding	Specifies the control file encoding to use.
Direct load	Specifies whether Data Manager should use the Oracle DIRECT path instead of the Oracle CONVENTIONAL path for the load.
Unrecoverable (direct load only)	In conjunction with the **Direct load** property, specifies whether to invoke the Oracle Unrecoverable option. This option can reduce load time but, in case of load failure, may result in corruption of the target table.
CLOB data inline	Specifies whether to treat Oracle CLOB (Character Large Object) data as character data. By default, CLOB data is written to temporary files which are referenced from the loader data file, whereas character data is written directly into the data file. Writing CLOB data directly into the data file allows faster loading. All CLOB data be must less than 8000 characters in length. The CLOB data must not contain any special characters that are likely to conflict with load control characters, such as new line characters.
Partition name	For a table that is partitioned within Oracle, this property specifies to which partition the data is to be delivered.
Options for control file	Specifies the SQL loader specific options for the loader control file.
SQL*loader command	Specifies the path and the command for the loader process. **Note:** You must specify the exact path name and command, including any double quotation marks required. For more information, see “Specify Executables to Deliver Data” on page 462.

Element Properties

For this delivery module, the element properties are the same for all element types.

Property	Description
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example, to use a zero value enter 0
Floating point scale	Specifies the number of decimal places for a numeric value that must be converted to text. By default, this value is three decimal places.

Red Brick Loader (TMU) Delivery Module

The Red Brick Loader (TMU) delivery module delivers data directly to an IBM Red Brick database using the Red Brick Bulk Load utility.

Module Properties

The table in this section describes the module properties.

Property	Description
Operation	Specifies whether IBM Cognos Data Manager should • **Run Loader** to deliver the text data and control files, load the data into the database, and clean the control files and any other intermediate files. (default) • **Run Loader (leave files)** to deliver the text data and control files, load the data into the database, but leave the control files and any other intermediate files so that you can view them. • **Create Loader Files** to deliver the text data and control files, but not to invoke the loader to load the data into the database.
File directory	Specifies the location in which to load data. By default, Data Manager loads data into its Data directory. **Tip:** To view the Data directory, from the **Tools** menu, click **Browse Data Files**.
Filename prefix	If **Create Loader Files** is selected in the **Operation** box, specify the prefix to use to generate the file names. If this box is empty, a default value is used.
Dump log file	Specifies whether to print the log file that the bulk loader generates.
Use named pipes	Specifies that Data Manager should use named pipes where possible. By default, Data Manager generates an intermediate text file and runs the bulk loader on it after delivering all data.

Property	Description
TMU <database>	Specifies the database for the TMU connection.
Load mode	Specifies one of these load modes: • **Append** mode appends the data to the target table. (default) • **Replace** mode deletes the contents of the target table but not the indexes, before delivering data. This is equivalent to a DELETE FROM statement. • **Insert** mode inserts the data in the target table. • **Modify** mode updates rows where the delivered rows have the same key as a row that exists in the target table. When there is no match, Data Manager appends the row to the target table. • **Update** mode updates rows where delivered rows have the same key as a row that exists in the target table. When there is no match, Data Manager discards the rows that do not correspond to an existing row in the target table. For more information, see the bulk loader documentation.
Optimized load	Specifies whether to perform a Red Brick optimized load.
Red Brick TMU command	Specifies the command to run the TMU process. **Note:** You must specify the exact path name and command, including any double quotation marks required. For more information, see "Specify Executables to Deliver Data" on page 462.

Element Properties

For this delivery module, the element properties are the same for all element types.

Property	Description
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example, to use a zero value, enter 0
Floating point scale	Specifies the number of decimal places for a numeric value that must be converted to text. By default, this value is three decimal places.

Relational Table Delivery Module

The Relational Table delivery module delivers fact build data to relational tables. These tables can then be accessed by client-server analysis applications, or can form the basis of standard reporting systems.

Module Properties

The table in this section describes the module properties.

Property	Description
Refresh type	Specifies the method by which rows should be applied to the output table: • **Append** appends the data to the target table. (default) • **Truncate** deletes the contents of the target table, together with all indexes, before delivering data. This is equivalent to a DELETE FROM statement but is usually faster. • **Replace** deletes the contents of the target table but not the indexes, before delivering data. This is equivalent to a DELETE FROM statement. • **Update/Insert** updates or inserts data in the target table. The value selected in the Update detection method box determines the way this is performed. For more information, see "Update/Insert Delivery Method" on page 429 • **Update** updates rows where delivery records have the same key as a row that exists in the target table. When there is no match, IBM Cognos Data Manager discards the records that do not correspond to an existing row in the target table. For more information, see "Update Delivery Method" on page 437
Incoming record keys are unique	If you are using the Update/Insert or Update refresh type, selecting this option indicates that there is a unique key for every record contained in the delivery data. For more information, see "Unique Incoming Record Keys for Update/Insert" on page 432.
Update detection method	If you are using the Update/Insert or Update refresh type, the way in which Data Manager updates or inserts data in the target table depends upon the update detection method you choose: • **Update** (default) • **Select** • **Select/Compare** • **Key Cache** For more information, see "Update Detection Method for Update/Insert" on page 430 and "Update Detection Method for Update" on page 438.
Key Cache max memory (MB)	If you are using the Update/Insert or Update refresh type, selecting this option specifies the maximum size of the key cache and its location. Enter • a negative value to store the key cache in memory • zero to store the key cache on disk • a positive value <*n*> to indicate that memory is to be used if the calculated size is less than or equal to <*n*>, otherwise use disk **Note:** By default, the key cache is stored in memory.

Property	Description
Key cache SELECT restriction	If you are using the Update/Insert or Update refresh type, selecting this option specifies the WHERE clause to restrict the number of key value combinations cached. By default, Data Manager reads the key value combinations for all rows in the target table. For very large tables this can be problematic, so you can specify a restriction to limit the values read. For example, you could enter a date range or a key range.
Key cache Cognos SQL restriction	If you are using the Update/Insert or Update refresh type, and have entered a restriction clause in the **Key cache SELECT restriction** box, selecting this check box specifies that the restriction is Cognos SQL.
Commit interval	Specifies (in terms of number of rows updated/inserted), the interval between commits of data in the target table. By default, Data Manager does not issue a COMMIT instruction until it has delivered all the data rows. That is, it executes the entire delivery as a single transaction. You must decide whether this represents an unacceptably large or long transaction.
Share database connection	If a fact build includes more than one relational table delivery that delivers to the same database connection, then selecting this option specifies that each delivery should share the database connection and transactions. Sharing a database connection may be required to reduce the number of connections to a database, or to deliver related tables where a database constraint exists between those tables. **Note:** The lowest specified commit interval on the deliveries is used for the transactions on the shared database connection.

Update/Insert Delivery Method

The Update/Insert delivery method allows you to control the way in which existing fact records are updated and new fact records are inserted in the target table. This provides more efficient delivery of fact records where it is not known if a row with the same key exists in the target table.

You can set up the processing method that best suits the profile of your data by configuring

- the update detection method
- whether or not incoming record keys are unique
- which columns should be updated in the target table
- the control attributes

Using different configurations can result in

- increased efficiency when writing records that already exist in the target table
- fewer unnecessary updates of records
- improved database performance due to fewer logged database operations during loading
- reduced complexity and maintenance overhead of update and insert loads

While it is possible to improve performance and efficiency when updating or inserting data, the overall performance also depends upon other factors, such as the logging options you choose when executing builds, and whether or not your tables are indexed.

Note: It is recommended that you use indexes on tables as this dramatically improves performance when using this delivery method.

Primary Key Settings for Update/Insert

IBM Cognos Data Manager uses the primary key settings to override the way in which the Relational Table delivery module identifies unique records. These key columns are then used to form the WHERE clause for an update.

Note: If null values are allowed in the dimension elements, then the update may fail to find records.

The update operation is constrained by an SQL WHERE clause formed as follows:

- Columns marked as keys are used.
- If no key is used and there are no dimension elements in the transformation model, then Data Manager always performs an insert.

For information on specifying key columns, see "Define the Properties for a Table Delivery" on page 181.

Update Detection Method for Update/Insert

The way in which IBM Cognos Data Manager updates or inserts data in the target table depends upon the update detection method you choose:

- **Update**

  Data Manager attempts to update every row in the target table.

  If the delivery record has the same key as a row that exists in the target table, Data Manager performs the update. If there is no match, Data Manager appends the delivery record to the target table.
- **Select**

  Data Manager searches the target table for a row with the same key as the delivery record.

  If a match is found, Data Manager updates the row. If there is no match, Data Manager appends the delivery record to the target table.
- **Select/Compare**

  Data Manager searches the target table for a row with the same key as the delivery record.

  If a match is found, Data Manager compares each column in the delivery record to the row that exists in the target table. If any columns differ, Data Manager updates them. Any identical columns remain unchanged. If no match is found, Data Manager appends the delivery record to the target table.
  - Select/Compare is only available if you have selected the Incoming record keys are unique check box.
  - If Select/Compare is selected, and you later clear the Incoming record keys are unique check box, the upgrade detection method is automatically changed to Select. The reason for this is that if the keys are not unique, when Data Manager updates a row, it does not attempt to compare individual columns. Instead it updates the entire row using the Select update detection method.
- **Key Cache**

Data Manager reads back the values of the key columns, and if appropriate, the record identity column, from the target table. It then caches them to either memory or to file depending on what you specified in the **Key Cache max memory (MB)** box. Data Manager then searches the cached keys for a row with the same key as the delivery record. If a match is found, Data Manager updates the row. If no match is found, the delivery record is appended to the target table.

For more information, see "Unique Incoming Record Keys for Update/Insert" on page 432.

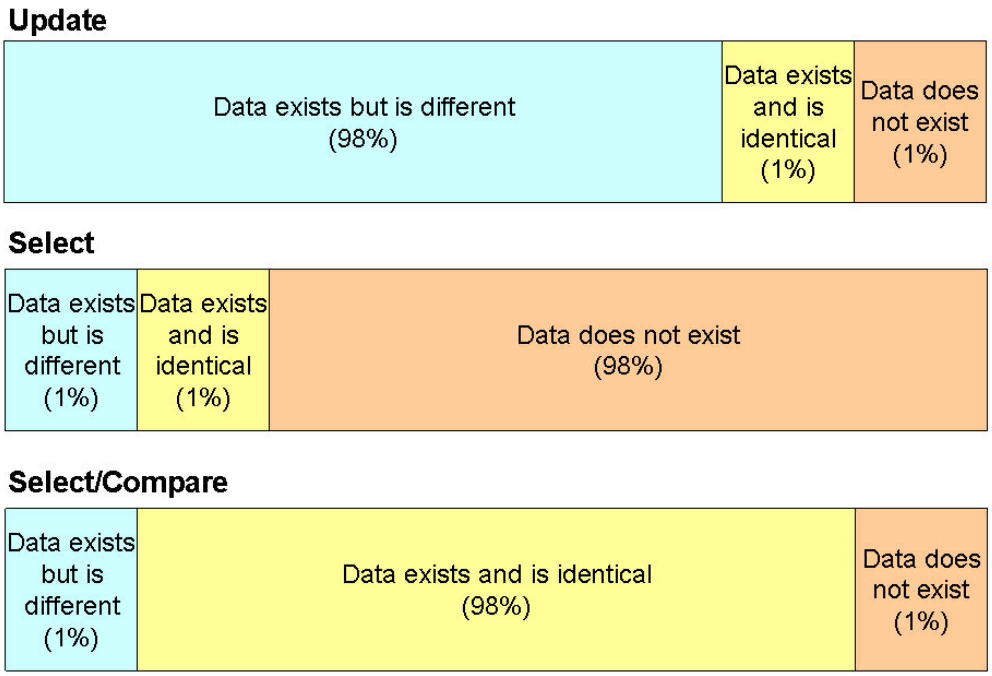
Guidelines For Deciding Which Update Detection Method to Use:

The decision on which update detection method to use is determined by the content of the data to be delivered. It is recommended that you use:

- Update if most delivery rows already exist in the target table, but need to be updated
- Select if most delivery rows contain new data so will need to be inserted rather than updated
- Select/Compare if
  - the delivery data contains unique incoming record keys
  - most delivery rows already exist in the target table and there are minimal changes to the data
  - you want individual columns to be updated rather than the whole row
- Key Cache in preference to Select if performance is an issue and either sufficient memory is available for a key cache in memory, or an appropriate restriction can be used to restrict the number of key value combinations for a key cache on disk.

For more information about unique incoming record keys, see "Unique Incoming Record Keys for Update/Insert" on page 432.

The following diagram illustrates, in percentage terms, approximately how your data should be split for each recommended method.


Update
Data exists but is different (98%)
Data exists and is identical (1%)
Data does not exist (1%)
Select
Data exists but is different (1%)
Data exists and is identical (1%)
Data does not exist (98%)
Select/Compare
Data exists but is different (1%)
Data exists and is identical (98%)
Data does not exist (1%)

Performance Considerations:

You should take into consideration how each update detection method can affect performance:

- Whilst Update and Select perform the same task, performance may be faster if you use Select because IBM Cognos Data Manager does not attempt to update every row.
- Select/Compare is generally slower than Select because Data Manager must check individual columns in matching rows. However, if you are executing builds across a network, performance is more likely to be improved using Select/Compare.
- Key cache should be used in preference to Select if performance is an issue and either sufficient memory is available for a key cache in memory, or an appropriate restriction can be used to restrict the number of key value combinations for a key cache on disk.

Unique Incoming Record Keys for Update/Insert

If your delivery data does not contain multiple records to update a single row in the target table, then the incoming record keys are unique. For example, in the following table, the incoming record keys are unique, because there is only a single delivery record for each product ID.

Product ID	Specification	Forecast Sales	Actual Sales
1	Red pen	500	150
2	Blue pen	400	200
3	Green pen	300	400

For databases that support array insert, if the incoming record keys are unique in the delivery data, selecting this option considerably improves performance when updating rows in the target table.

Notes

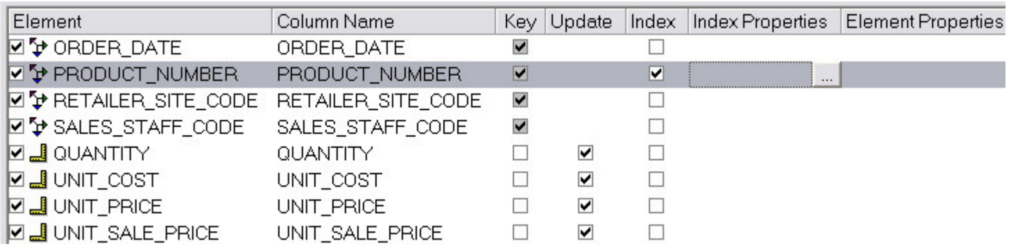
- If unique keys exist in the delivery data, and you want to use the Select/Compare update detection method, ensure that you select the Incoming record keys are unique option. For more information, see “Update Detection Method for Update/Insert” on page 430.
- If unique keys exist in the delivery data, and the target table is empty, IBM Cognos Data Manager does not attempt to update any rows, it just inserts them.

Updated Columns

By default, when IBM Cognos Data Manager performs an update, all columns in the target table are updated, with the exception of any columns identified as key columns.

There may be certain columns that you do not want to update in the target table. Excluding them from update reduces processing time and improves performance.

You can specify which columns to update, and which to exclude from update, in the Table Delivery Properties window.



Element	Column Name	Key	Update	Index	Index Properties	Element Properties
☑ ORDER_DATE	ORDER_DATE	☑		□		
☑ PRODUCT_NUMBER	PRODUCT_NUMBER	☑		☑	...	
☑ RETAILER_SITE_CODE	RETAILER_SITE_CODE	☑		□		
☑ SALES_STAFF_CODE	SALES_STAFF_CODE	☑		□		
☑ QUANTITY	QUANTITY	□	☑	□		
☑ UNIT_COST	UNIT_COST	□	☑	□		
☑ UNIT_PRICE	UNIT_PRICE	□	☑	□		
☑ UNIT_SALE_PRICE	UNIT_SALE_PRICE	□	☑	□		

Control Attributes

You can use the following control attributes in a fact delivery table to track when rows are updated or inserted into the target table:

- **Create Date** specifies the date on which a new row was inserted into the target table
- **Last Update Date** specifies the date on which an existing row was updated

The Create Date attribute is inserted once in the target table, and never changes. The way in which Last Update Date changes depends on the update detection method you are using:

- If you are using Update or Select detection methods, IBM Cognos Data Manager sets the Last Update Date each time it performs an update, even if no data has changed in a row. This happens because Data Manager refreshes the entire row without checking individual columns.
- If you are using Select/Compare, Data Manager only sets the Last Update Date when specific columns in a row have changed.

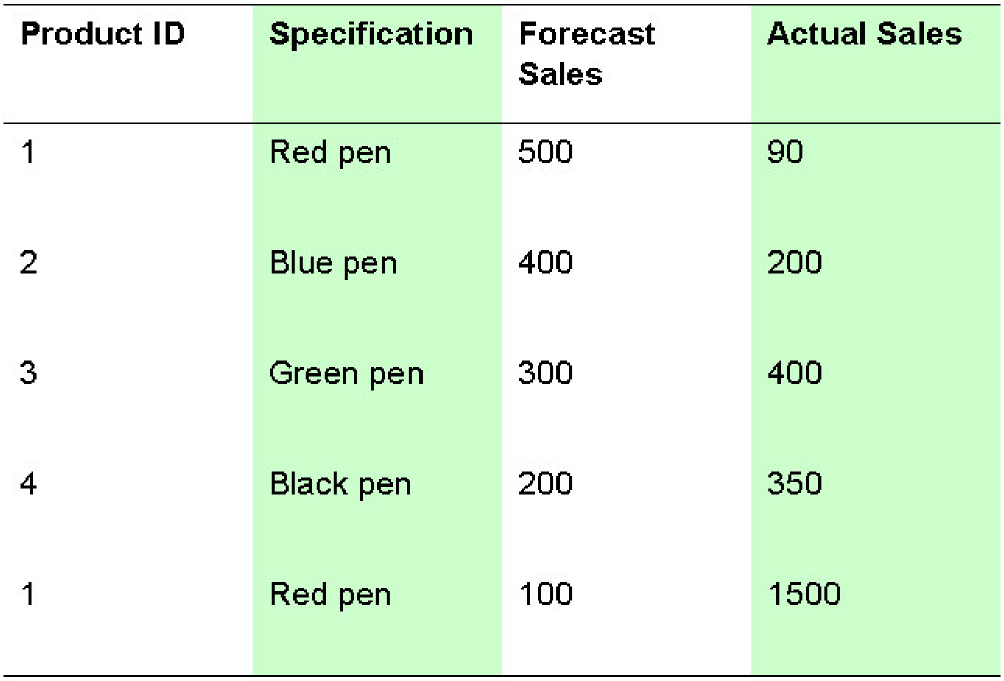
  Note: You must use this update detection method if you want to ensure the Last Update Date is always set accurately.

For more information on the update detection method, see “Update Detection Method for Update/Insert” on page 430.

For information on adding control attributes to a fact delivery table, see "Define the Properties for a Table Delivery" on page 181.

Update/Insert Delivery Example

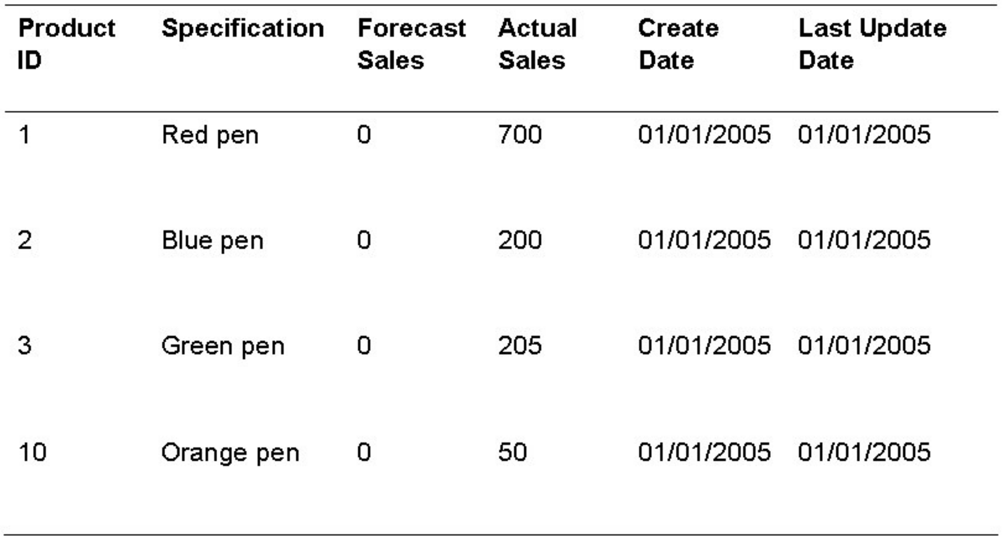
Suppose a fact delivery table contains the following data, where Product ID is defined as the key, and the shaded columns are selected for update.



Product ID	Specification	Forecast Sales	Actual Sales
1	Red pen	500	90
2	Blue pen	400	200
3	Green pen	300	400
4	Black pen	200	350
1	Red pen	100	1500

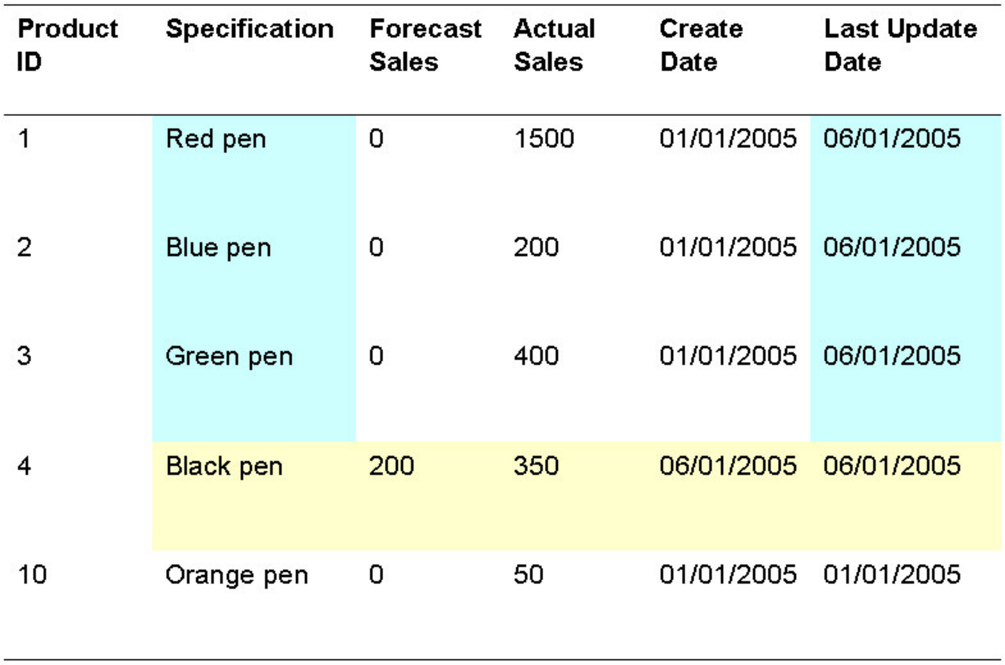
Note that there are two delivery records for product ID 1, in other words incoming record keys are not unique.

The target table into which the data will be delivered, currently contains the following rows. Control attributes are included in the target table.



Product ID	Specification	Forecast Sales	Actual Sales	Create Date	Last Update Date
1	Red pen	0	700	01/01/2005	01/01/2005
2	Blue pen	0	200	01/01/2005	01/01/2005
3	Green pen	0	205	01/01/2005	01/01/2005
10	Orange pen	0	50	01/01/2005	01/01/2005

When you load the fact data using the Update or Select update detection method, IBM Cognos Data Manager performs four updates (row 1 is updated twice) and one insert. The target table looks as follows.

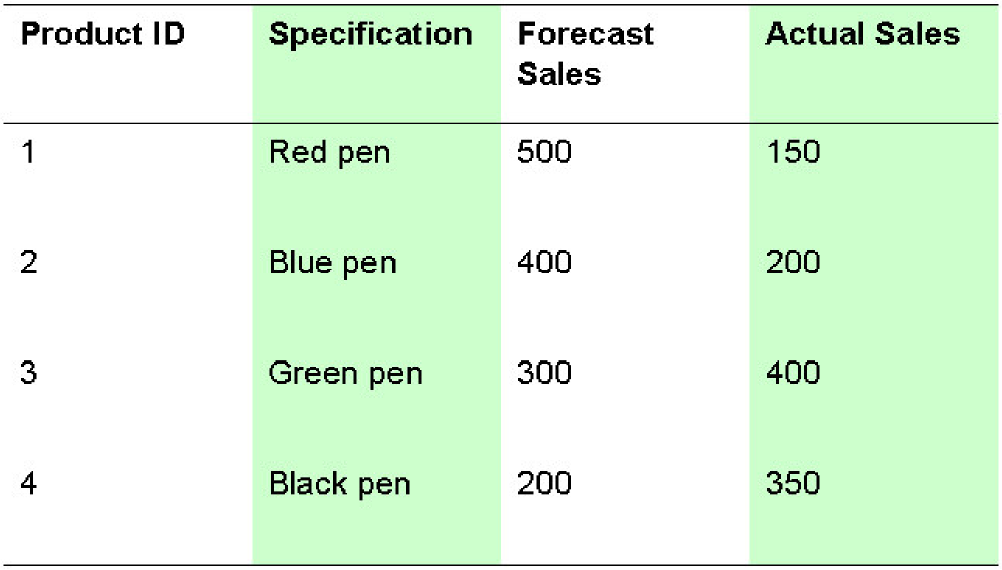


Product ID	Specification	Forecast Sales	Actual Sales	Create Date	Last Update Date
1	Red pen	0	1500	01/01/2005	06/01/2005
2	Blue pen	0	200	01/01/2005	06/01/2005
3	Green pen	0	400	01/01/2005	06/01/2005
4	Black pen	200	350	06/01/2005	06/01/2005
10	Orange pen	0	50	01/01/2005	01/01/2005

Notice that

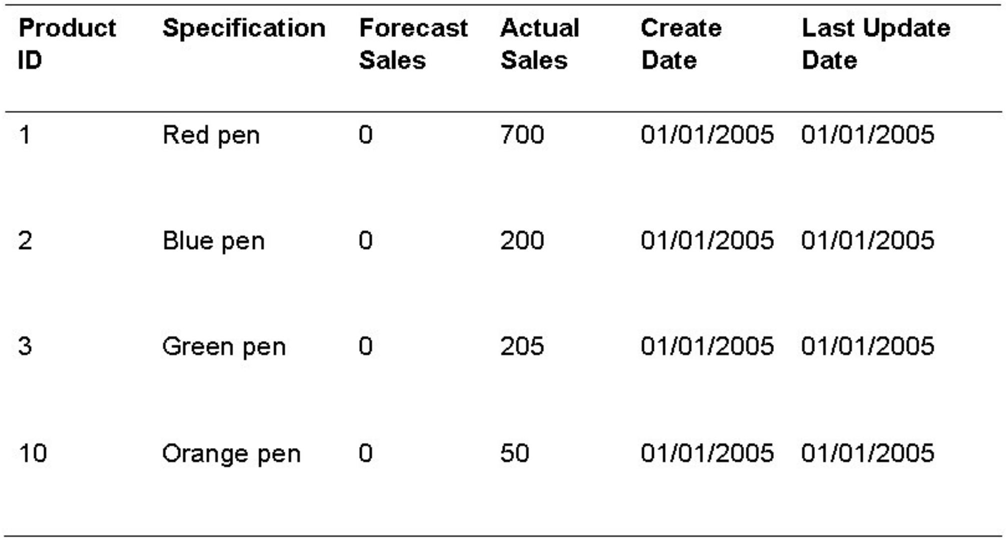
- The Forecast Sales column has not been updated for any existing row in the target table because it is excluded from update.
- Rows 1, 2, and 3 have all been updated (indicated by blue shading). Even though the data in row 2 has not changed, Data Manager has refreshed the row with data from the fact delivery table.
- Row 4 is a new row that has been inserted (indicated by yellow shading).
- Row 10 remains unchanged.
- The Last Update Date is set for every row that has been updated or inserted.

Now suppose a fact delivery table contains the following data. Product ID is defined as the key, shaded columns are selected for update, and incoming record keys are unique.



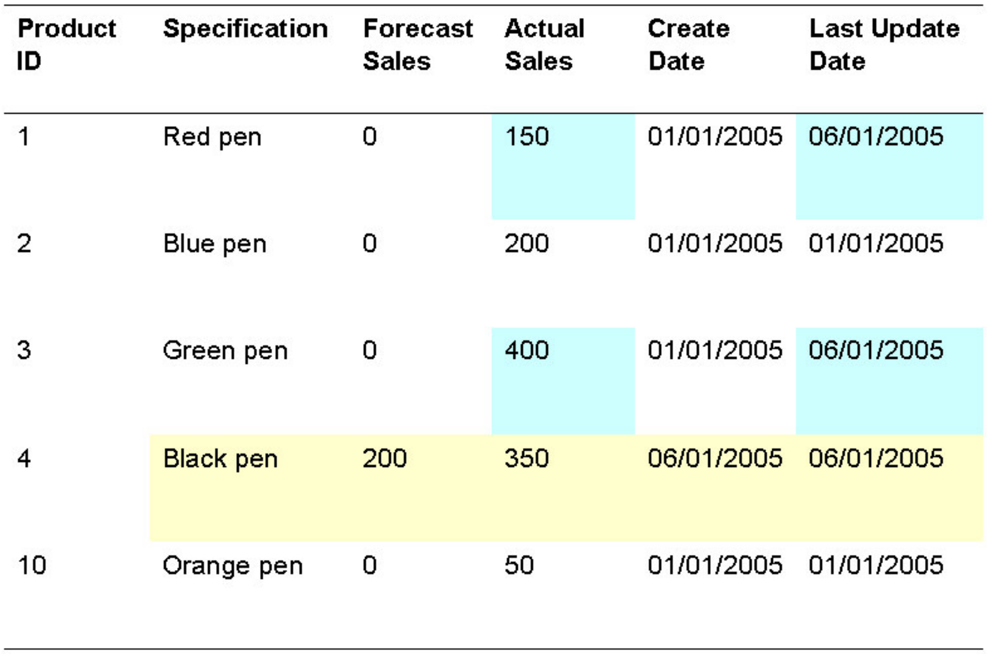
Product ID	Specification	Forecast Sales	Actual Sales
1	Red pen	500	150
2	Blue pen	400	200
3	Green pen	300	400
4	Black pen	200	350

The target table into which the data will be delivered, currently contains the following rows. Control attributes are included in the target table.



Product ID	Specification	Forecast Sales	Actual Sales	Create Date	Last Update Date
1	Red pen	0	700	01/01/2005	01/01/2005
2	Blue pen	0	200	01/01/2005	01/01/2005
3	Green pen	0	205	01/01/2005	01/01/2005
10	Orange pen	0	50	01/01/2005	01/01/2005

When you load the fact data into the target table using the Select/Compare update detection method (the Incoming record keys are unique option is selected), Data Manager performs two updates and one insert. The target table looks like this:



Product ID	Specification	Forecast Sales	Actual Sales	Create Date	Last Update Date
1	Red pen	0	150	01/01/2005	06/01/2005
2	Blue pen	0	200	01/01/2005	01/01/2005
3	Green pen	0	400	01/01/2005	06/01/2005
4	Black pen	200	350	06/01/2005	06/01/2005
10	Orange pen	0	50	01/01/2005	01/01/2005

Notice that

- In rows 1 and 3, Data Manager has only updated the columns that have changed (indicated by blue shading).
- Row 4 is a new row that has been inserted (indicated by yellow shading).
- Row 10 remains unchanged.
- The Last Update Date is set only for those rows that have been updated or inserted.

Update Delivery Method

The Update delivery method allows you to control the way in which existing fact records are updated in the target table.

You can improve the efficiency of updates to the target table by also specifying

- the update detection method
- which columns to update
- which control attributes to use

Using different configurations can result in

- increased efficiency when updating records
- fewer unnecessary updates of records
- improved database performance due to fewer logged database operations during loading
- reduced complexity and maintenance overhead of update loads

While it is possible to improve performance and efficiency when updating data, the overall performance also depends upon other factors, such as the logging options you choose when executing builds, and whether or not your tables are indexed.

Note: It is recommended that you use indexes on tables as this dramatically improves performance when using this delivery method.

Primary Key Settings for Update

IBM Cognos Data Manager uses the key settings to override how the Relational Table delivery module identifies unique records. These key columns are then used to form the WHERE clause on the update.

Note: If null values are allowed in the dimension elements, then the update may fail to find records.

The update operation is constrained by an SQL WHERE clause formed as follows:

- Columns marked as keys are used.
- If no key is used and there are no dimension elements in the transformation model, then Data Manager produces an error message and fails.

For information on specifying key columns, see "Define the Properties for a Table Delivery" on page 181.

Update Detection Method for Update

The way in which IBM Cognos Data Manager updates data in the target table depends upon the update detection method you choose:

- **Update**

  Data Manager attempts to update every row in the target table.

  If the delivery record has the same key as a row that exists in the target table, Data Manager performs the update. If there is no match, Data Manager discards the delivery record.
- **Select**

  Data Manager searches the target table for a row with the same key as the delivery record.

  If a match is found, Data Manager updates the row. If there is no match, Data Manager discards the delivery record.
- **Select/Compare**

  Data Manager searches the target table for a row with the same key as the delivery record.

  If a match is found, Data Manager compares each column in the delivery record to the row that exists in the target table. If any columns differ, Data Manager updates them. Any identical columns remain unchanged. If there is no match, Data Manager discards the delivery record.
  - Select/Compare is only available if you have selected the Incoming record keys are unique check box.
  - If Select/Compare is selected, and you later clear the Incoming record keys are unique check box, the upgrade detection method is automatically changed to Select. The reason for this is that if the keys are not unique, when Data Manager updates a row, it does not attempt to compare individual columns. Instead it updates the entire row using the Select update detection method.
- **Key Cache**

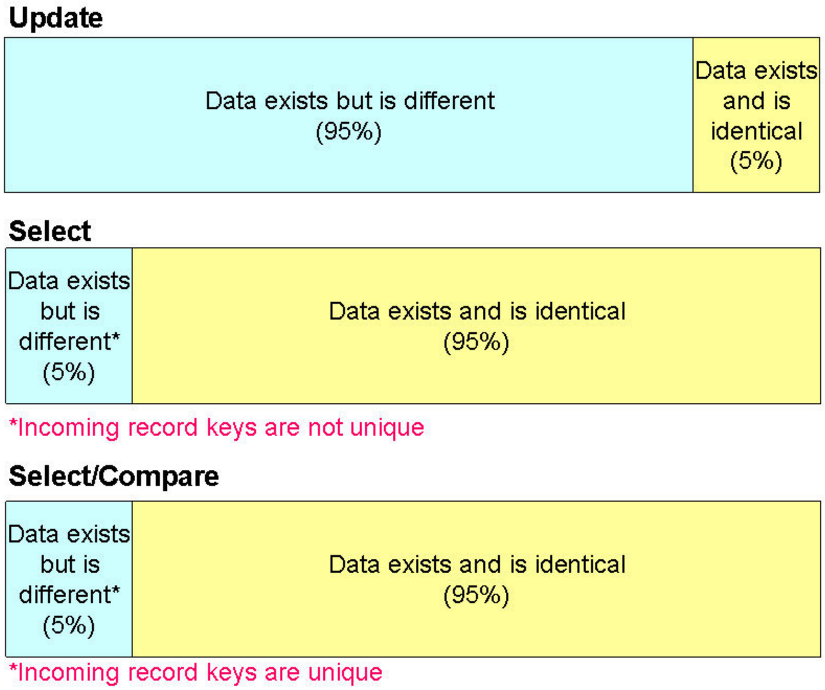
  Data Manager reads back the values of the key columns, and if appropriate, the record identity column, from the target table. It then caches them to either memory or to file depending on what you specified in the Key Cache max memory (MB) box. Data Manager then searches the cached keys for a row with the same key as the delivery record. If a match is found, Data Manager updates the row. If no match is found, Data Manager discards the delivery record.

Guidelines For Deciding Which Update Detection Method to Use

The decision on which update detection method to use is determined by the content of the data to be delivered. It is recommended that you use

- Update if most delivery rows that exist in the target table need to be updated
- Select if
  - most delivery rows already exist in the target table and there are minimal changes to the data
  - the delivery data does not contain unique incoming record keys
- Select/Compare if
  - most delivery rows already exist in the target table and there are minimal changes to the data
  - you want individual columns to be updated rather than updating the whole row
- Key Cache in preference to Select if performance is an issue and either sufficient memory is available for a key cache in memory, or an appropriate restriction can be used to restrict the number of key value combinations for a key cache on disk.

The following diagram illustrates, in percentage terms, approximately how your data should be split for each recommended method.


Update
Data exists but is different (95%)
Data exists and is identical (5%)
Select
Data exists but is different* (5%)
Data exists and is identical (95%)
*Incoming record keys are not unique
Select/Compare
Data exists but is different* (5%)
Data exists and is identical (95%)
*Incoming record keys are unique

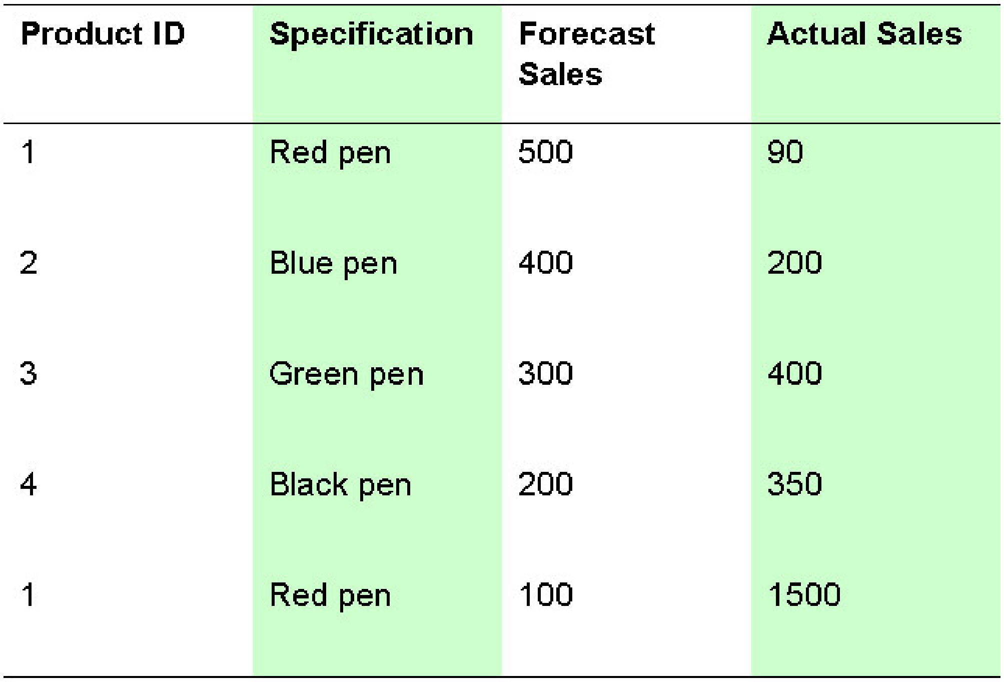
Performance Considerations

You should take into consideration how each update detection method can affect performance:

- While Update and Select perform the same task, performance may be faster if you use Select because IBM Cognos Data Manager does not attempt to update every row.
- Select/Compare is generally slower than Select because Data Manager must check individual columns in matching rows. However, if you are executing builds across a network, performance is more likely to be improved using Select/Compare.
- Key cache should be used in preference to Select if performance is an issue and either sufficient memory is available for a key cache in memory, or an appropriate restriction can be used to restrict the number of key value combinations for a key cache on disk.

Update Delivery Example

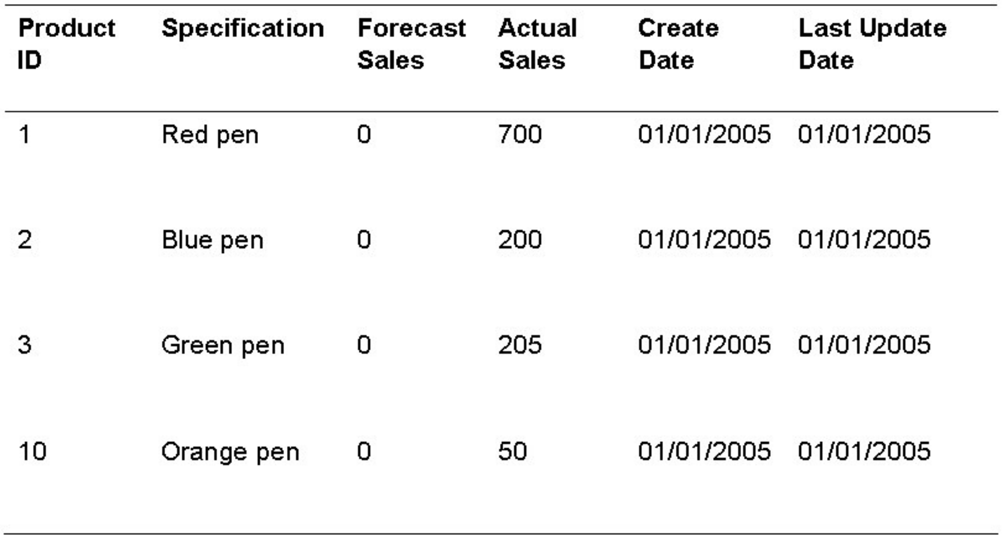
Suppose a fact delivery table contains the following data, where Product ID is defined as the key, and the shaded columns are selected for update.



Product ID	Specification	Forecast Sales	Actual Sales
1	Red pen	500	90
2	Blue pen	400	200
3	Green pen	300	400
4	Black pen	200	350
1	Red pen	100	1500

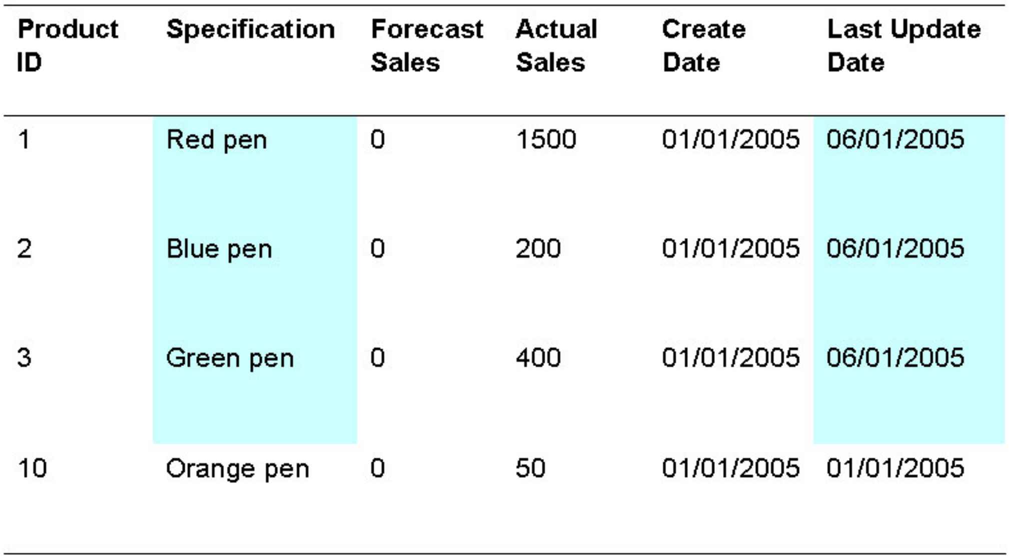
Note that there are two delivery records for product ID 1, in other words incoming record keys are not unique.

The target table into which the data will be delivered, currently contains the following rows. Control attributes are included in the target table.



Product ID	Specification	Forecast Sales	Actual Sales	Create Date	Last Update Date
1	Red pen	0	700	01/01/2005	01/01/2005
2	Blue pen	0	200	01/01/2005	01/01/2005
3	Green pen	0	205	01/01/2005	01/01/2005
10	Orange pen	0	50	01/01/2005	01/01/2005

When you load the fact data using the Update or Select update detection method, IBM Cognos Data Manager performs four updates (row 1 is updated twice). The target table looks as follows.

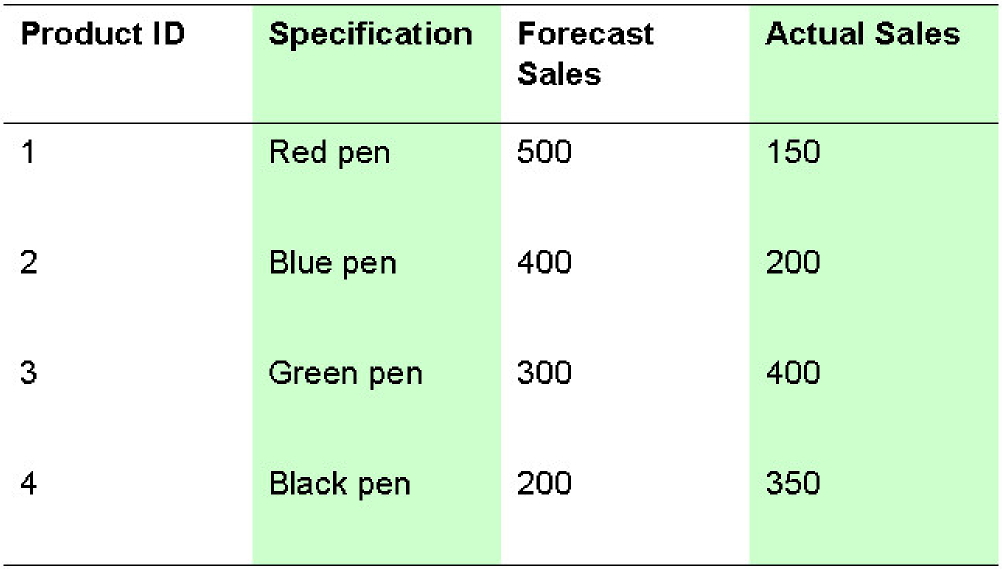


Product ID	Specification	Forecast Sales	Actual Sales	Create Date	Last Update Date
1	Red pen	0	1500	01/01/2005	06/01/2005
2	Blue pen	0	200	01/01/2005	06/01/2005
3	Green pen	0	400	01/01/2005	06/01/2005
10	Orange pen	0	50	01/01/2005	01/01/2005

Notice that

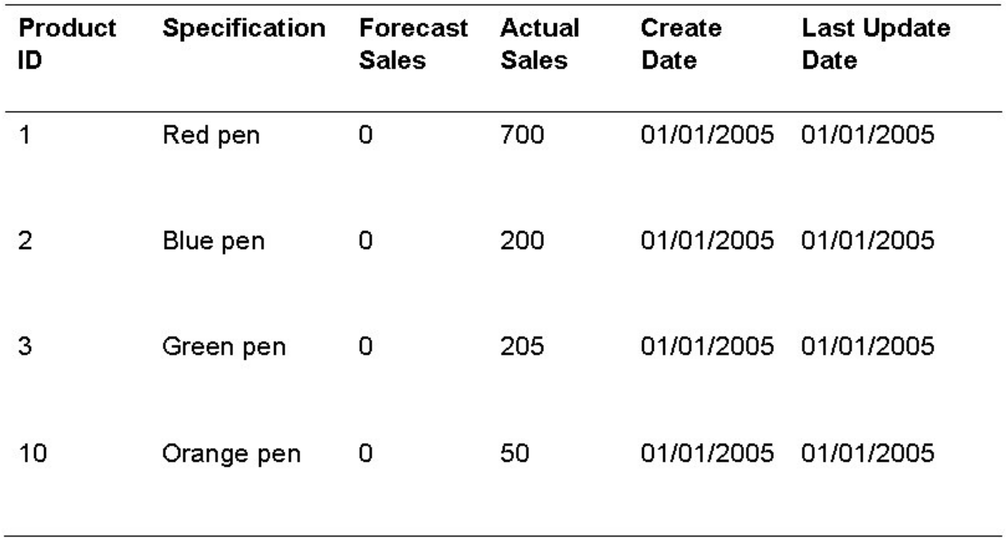
- The Forecast Sales column has not been updated for any existing row in the target table because it is excluded from update.
- Rows 1, 2, and 3 have all been updated (indicated by blue shading). Even though the data in row 2 has not changed, Data Manager has refreshed the row with data from the fact delivery table.
- Row 4 in the delivery data has been discarded because Data Manager could not find a match in the target table.
- Row 10 remains unchanged.
- The Last Update Date is set for every row that has been updated.

Now suppose a fact delivery table contains the following data. Product ID is defined as the key, shaded columns are selected for update, and incoming record keys are unique.



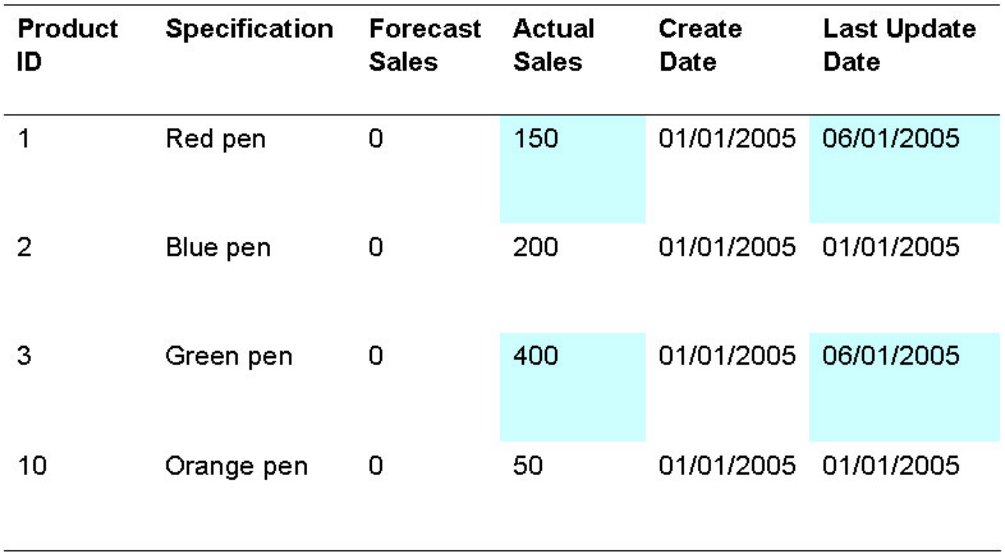
Product ID	Specification	Forecast Sales	Actual Sales
1	Red pen	500	150
2	Blue pen	400	200
3	Green pen	300	400
4	Black pen	200	350

The target table into which the data will be delivered, currently contains the following rows. Control attributes are included in the target table.



Product ID	Specification	Forecast Sales	Actual Sales	Create Date	Last Update Date
1	Red pen	0	700	01/01/2005	01/01/2005
2	Blue pen	0	200	01/01/2005	01/01/2005
3	Green pen	0	205	01/01/2005	01/01/2005
10	Orange pen	0	50	01/01/2005	01/01/2005

When you load the fact data into the target table using the Select/Compare update detection method (the Incoming record keys are unique option is selected), Data Manager performs two updates. The target table looks like this:



Product ID	Specification	Forecast Sales	Actual Sales	Create Date	Last Update Date
1	Red pen	0	150	01/01/2005	06/01/2005
2	Blue pen	0	200	01/01/2005	01/01/2005
3	Green pen	0	400	01/01/2005	06/01/2005
10	Orange pen	0	50	01/01/2005	01/01/2005

Notice that

- In rows 1 and 3, Data Manager has only updated the columns that have changed (indicated by blue shading).
- Row 10 remains unchanged.
- The Last Update Date is set for only those rows that have been updated.

Element Properties

The properties available depend on the type of element that you are defining.

Property	Description
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example,to use a zero value,just enter 0
Additive update	When updating a table, specifies whether to add new values to the values that exist in the table (rather than replace them). This property does not apply to dimensions.
Maximum UTF-8 bytes per character	When using Key Cache for key elements that comprise character data, this can be used to reduce the size of the cache. By default, IBM Cognos Data Manager calculates the size of such elements by multiplying the length of the target column by this value (default 4). For example, if it is known that the values of the element are always composed of ASCII or numeric characters, then the value should be specified as 1.

SQL Server Bulk Copy via API Delivery Module

The SQL Server Bulk Copy via API delivery module supports bulk copying to Microsoft SQL Server and Sybase SQL Server. It uses a programmatic interface supplied by your DBMS vendor.

To bulk copy to Microsoft SQL Server, use either a Microsoft SQL Server ODBC connection or a Microsoft SQL Server OLE-DB connection. To bulk copy to Sybase SQL Server, use a Sybase SQL Server connection.

Note: Depending on the type of database connection you are using, this delivery module uses different interfaces to the underlying database. For example, API delivery to Microsoft SQL Server via ODBC is not the same as API delivery to Microsoft SQL Server via OLE-DB. You should consider any performance implications as a result of changing the interface type.

Module Properties

The table in this section describes the module properties.

Property	Description
Load mode	Specifies one of these load modes: • **Append** appends the data to the target table. (default) • **Truncate** deletes the contents of the target table and drops all indexes before delivering the data. This is equivalent to a DELETE FROM statement but is usually faster. • **Replace** deletes the contents of the target table but not the indexes, before delivering the data. This is a DELETE FROM statement.
Batch size	Specifies the number of rows to be copied in a batch. By default, the delivery commits all the rows in one batch.
Bulk copy hints	Use to set load hints ROWS_PER_BATCH, KILOBYTES_PER_BATCH, TABLOCK, CHECK_CONSTRAINTS, and ORDERS.
Treat empty string as null value	Specifies whether empty string values should be read as null values.

Element Properties

For this delivery module, the element properties are the same for all element types.

Property	Description
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example, to use a zero value, enter 0

Sybase ASE BCP Command Delivery Module

The Sybase ASE BCP Command delivery module delivers data directly to Sybase ASE databases using the Sybase ASE Server BCP bulk load utility.

The BCP command utility is a component of Sybase ASE client installation. It provides users with an intermediate load file that is passed to the BCP utility.

Module Properties

The table in this section describes the module properties.

Property	Description
Operation	Specifies whether IBM Cognos Data Manager should • **Run Loader** to deliver the text data and control files, load the data into the database and clean up the control files and any other intermediate files. (default) • **Run Loader (leave files)** to deliver the text data and control files, load the data into the database but leave the control files and any other intermediate files so that you can view them. • **Create Loader Files** to deliver the text data and control files, but not to invoke the loader to load the data into the database.
File directory	Specifies the location in which to load data. By default, Data Manager loads data into its Data directory. **Tip:** To view the Data directory, from the **Tools** menu, click **Browse Data Files**.
Filename prefix	If **Create Loader Files** is selected in the **Operation** box, specify the prefix to use to generate the file names. If this box is empty, a default value is used.
Dump log file	Specifies whether to print the log file that the bulk loader generates.
No row count check	Specifies whether to check for successful delivery of the data. Clear this check box to prevent an error if the target DBMS does not report successful delivery in the expected format. Select this check box to use a newer version of the DBMS where the vendor changed the format of the completion message.
Log file success message	If **No row count check** is not selected, this is used to determine whether the load was successful by searching for the specified substring in the log file or output file. Data Manager searches for this string and the number of rows delivered as substrings in each line of the log file or output file.
Record delimiter	Specifies the character sequence to use to separate records. Specify a character string or use one of these: Tab, Comma, Space, None, NL (new line), CRLF (carriage return and line feed combination) on a Microsoft Windows operating system, or CR (carriage return) on a UNIX operating system. NL is the default.

Property	Description
Field delimiter	Specifies the character sequence to use to separate fields. Specify a character string or use one of these named characters: Tab (the default), Comma, Space, None, or NL (new line).
Load mode	Specifies one of these load modes: • **Append** appends the data to the target table. (default) • **Truncate** deletes the contents of the target table, and drops all indexes, before delivering the data. This is equivalent to a DELETE FROM statement, but is usually faster. • **Replace** deletes the contents of the target table but not the indexes, before delivering the data. This is equivalent to a DELETE FROM statement. • **Insert** inserts the data in the target table.
Data file encoding	Specifies the data file encoding to use.
Sybase ASE BCP command	Specifies the path and command for the loader process. **Note:** You must specify the exact path name and command, including any double quotation marks required. For more information, see "Specify Executables to Deliver Data" on page 462.
Additional command line options	Allows additional BCP command line options to be specified.

Element Properties

For this delivery module, the element properties are the same for all element types.

Property	Description
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example, to use a zero value, enter 0
Floating point scale	Specifies the number of decimal places for a numeric value that must be converted to text. By default, this value is three decimal places.

Sybase IQ LOAD Delivery Module

The Sybase IQ LOAD delivery module delivers data directly to Sybase IQ databases.

Module Properties

The table in this section describes the module properties.

Property	Description
Operation	Specifies whether IBM Cognos Data Manager should • **Run Loader** to deliver the text data and control files, load the data into the database and clean up the control files and any other intermediate files. (default) • **Run Loader (leave files)** to deliver the text data and control files, load the data into the database but leave the control files and any other intermediate files so that you can view them. • **Create Loader Files** to deliver the text data and control files, but not to invoke the loader to load the data into the database.
File directory	Specifies the location in which to load data. By default, Data Manager loads data into the Data directory. If Data Manager is installed on a different computer to the database, and **Load from Client file** is not selected, then you must specify a shared server location. For example, on Windows use Uniform Naming Convention (UNC): `\\ComputerName\SharedFolder`
Filename prefix	If **Create Loader Files** is selected in the **Operation** box, specify the prefix to use to generate the file names. If this box is empty, a default value is used.
Dump log file	Specifies whether to print the log file that the bulk loader generates. **Note:** Log files are stored on the Sybase IQ server.
Use named pipes	Specifies that Data Manager should use named pipes where possible. By default, Data Manager generates an intermediate text file and runs the bulk loader on it after delivering all data. **Note:** You can only use named pipes when Data Manager is installed on the same computer as the database.
Record delimiter	Specifies the character sequence to use to separate records. Specify a character string or use one of these: Tab, Comma, Space, None, NL (new line), CRLF (carriage return and line feed combination) on a Microsoft Windows operating system, or CR (carriage return) on a UNIX operating system. NL is the default.
Field delimiter	Specifies the character sequence to use to separate fields. Specify a character string or use one of these named characters: Tab (the default), Comma, Space, None, or NL (new line).
Load mode	Specifies one of these load modes: • **Insert** inserts the data in the target table. • **Replace** deletes the contents of the target table but not the indexes, before delivering the data. This is equivalent to a DELETE FROM statement.

Property	Description
Data file encoding	Specifies the data file encoding to use.
Load from client file	Specifies that data should be loaded from a client file, indicating that the File directory path is relative to the database loader (client) rather than to the database server.
Additional options	Specifies a string of extra load options to append to the command.
Sybase CLP command	Specifies the path and command for the Sybase IQ loader process. **Note:** You must specify the exact path name and command, including any double quotation marks required. For more information, see "Specify Executables to Deliver Data" on page 462.
Sybase quotes option	Specifies whether to use quotation marks to enclose data values. By default, no quotation marks are used.

Element Properties

The element properties available depend on the type of element being defined.

Property	Description
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example, to use a zero value enter 0
Perform escape processing	Specifies whether to use the escape character (\) before all backslash, tab, and new line characters in string values to ensure that the delivery module correctly interprets the statement.
Floating point scale	Specifies the number of decimal places for a numeric value that must be converted to text. By default, this value is three decimal places.

Teradata Fastload Delivery Module

The Teradata Fastload delivery module delivers data directly to Teradata Fastload databases.

Note: You can access Teradata on a UNIX operating system using an ODBC driver. However, you must ensure that, in the ODBC.ini file, you use the data source name that you used on a Microsoft Windows operating system.

Module Properties

The table in this section describes the module properties.

Property	Description
Operation	Specifies whether IBM Cognos Data Manager should • **Run Loader** to deliver the text data and control files, load the data into the database and clean up the control files and any other intermediate files. (default) • **Run Loader (leave files)** to deliver the text data and control files, load the data into the database but leave the control files and any other intermediate files so that you can view them. • **Create Loader Files** to deliver the text data and control files, but not to invoke the loader to load the data into the database.
File directory	Specifies the location in which to load data. By default, Data Manager loads data into its Data directory. **Tip:** To view the Data directory, from the **Tools** menu, click **Browse Data Files**.
Filename prefix	If **Create Loader Files** is selected in the **Operation** box, specify the prefix to use to generate the file names. If this box is empty, a default value is used.
Dump log file	Specifies whether to print the log file that the bulk loader generates.
No row count check	Specifies whether to check for successful delivery of the data. Clear this check box to prevent an error if the target DBMS does not report successful delivery in the expected format. Select this check box to use a newer version of the DBMS where the vendor changed the format of the completion message.
Log file success message	If **No row count check** is not selected, this is used to determine whether the load was successful by searching for the specified substring in the log file or output file. Data Manager searches for this string and the number of rows delivered as substrings in each line of the log file or output file.
Use named pipes	Specifies that Data Manager should use named pipes where possible. By default, Data Manager generates an intermediate text file and runs the bulk loader on it after delivering all data.
Field delimiter	Specifies the character sequence to use to separate fields. Specify a character string or use one of these named characters: Tab (the default), Comma, Space, None, or NL (new line).

Property	Description
Load mode	Specifies one of these load modes: • **Append** appends the data to the target table. (default) • **Truncate** deletes the contents of the target table, and drops all indexes, before delivering the data. This is equivalent to a DELETE FROM statement, but is usually faster. • **Replace** deletes the contents of the target table but not the indexes, before delivering the data. This is equivalent to a DELETE FROM statement.
Data file encoding	Specifies the data file encoding to use.
Max sessions	Specifies the maximum number of loader sessions to use. If no value is specified, the Fastload default is used.
Min sessions	Specifies the minimum number of loader sessions to use. If no value is specified, the Fastload default is used.
Error file name	Specifies an alternate file name for Fastload error messages. This produces a duplicate record of all Fastload error messages.
Error limit	Specifies the maximum number of records that can be rejected before the loader stops. If no value is specified, the Fastload default is used.
Verbose	Specifies whether to print every request sent to the Teradata Database.
Encrypt	Specifies whether the session is to be encrypted.
Additional options	Specifies a string of extra load options to append to the command. ERRORFILES specifies the file names to use for error tables. The default file names are fload_error1 and fload_error2. This option is mandatory.
Teradata fastload command	Specifies the path and command for the Fastload loader process. **Note:** You must specify the exact path name and command, including any double quotation marks required. For more information, see “Specify Executables to Deliver Data” on page 462.

Element Properties

The element properties available depend on the type of element being defined.

Property	Description
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example, to use a zero value, enter 0
Perform escape processing	Specifies whether to use the escape character (\) before all backslash, tab, and new line characters in string values to ensure that the delivery module correctly interprets the statement.
Floating point scale	Specifies the number of decimal places for a numeric value that must be converted to text. By default, this value is three decimal places.

TM1 Turbo Integrator Delivery Module

The TM1 Turbo Integrator delivery module can deliver data directly to IBM Cognos TM1 cubes. Alternatively, it can be used to create flat files for loading into TM1 cubes. In this instance, it also creates the TM1 process required to load data and runs the process, if required.

You can deliver fact data and dimension data.

IBM Cognos Data Manager dimension elements are delivered as dimensions into a TM1 cube. You can also deliver other Data Manager transformation model elements as elements of a TM1 measures dimension. In this case, you must specify the name of the TM1 measures dimension to use for delivery.

Alternatively, a single non-dimensional transformation model element must be delivered. In this case, you should not specify any TM1 measures dimension name.

Installation Considerations

The TM1 client and Data Manager must be installed on the same computer. For a distributed installation, the TM1 client, Data Manager, and the Data Manager engine must be installed on the same computer.

For Data Manager to find the required libraries, you must add a TM1 installation bin directory. To install the TM1 server locally, use TM1 9.4 and accept the default values.

When the installation is complete, ensure that the TM1 service has started, and then start Data Manager.

Notes

- Currently, you can only deliver to TM1 using Data Manager on a Microsoft Windows operating system.

- You can ignore specified minor errors when delivering data to TM1 by setting the variable DM_TM1_LOAD_SUPPRESS_ERROR_CODES. For more information, see "DM_TM1_LOAD_SUPPRESS_ERROR_CODES" on page 295.
- If you decide to ignore minor errors, information about the number of rows successfully delivered may not be accurate.

Module Properties

The table in this section describes the module properties.

Property	Description
Operation	Specifies whether IBM Cognos Data Manager should • **Run TM1 Process** to deliver the data files and the TM1 process, load the data into the cube and clean up the process and any other intermediate files. (default) **Note:** If the **Direct delivery**check box is selected, data files are purged but not deleted from the TM1 server. • **Run and Leave TM1 Process** to deliver the data files and the TM1 process, load the data into the cube but leave the process and any other intermediate files so that you can view them. • **Create TM1 Process** to create the data files and the TM1 process, but not to load the data into the cube.
Connection	Specifies the connection for the TM1 server.
File directory	Specifies the location in which to write the data files. By default, Data Manager writes data into its Data directory. **Tip:** To view the Data directory, from the **Tools** menu, click **Browse Data Files**. **Note:** If the **Direct delivery**check box is selected, you do not need to specify a file directory.
File directory on server	Specifies the server location of the file directory. This information is required only if Data Manager is installed on a different computer to the TM1 server. **Note:** If the **Direct delivery**check box is selected, you do not need to specify a file directory.
Filename prefix	If **Create TM1 Process** is selected in the **Operation** box, specify the file name to use for the data files and process. If you do not specify a file name, Data Manager uses a default name.
Direct delivery	Specifies whether to deliver data directly to the TM1 server rather than creating flat files for loading. Select this check box to deliver data directly to TM1.
Process name on server	Specifies the name to use for the TM1 process. If you do not specify a name, Data Manager uses a unique name by default.
Cube name	Specifies the name of the TM1 cube to create or update. If you do not specify a cube name, Data Manager ignores the specified cube action and data action.

Property	Description
Cube action	Specifies the cube action to take: • **Create** to create a cube for the delivered data. (default) **Note:** The specified cube name must not already exist. • **Recreate** to delete an existing cube and recreate it. • **Update** to deliver the data to an existing cube. When this is selected, for all dimensions (including a measures dimension), any dimension action that is set to **Create** or **Update** is treated as an update. • **NoAction** to not create or update a cube. • **Truncate** to reset all values in an existing cube to zero.
View name	If **Truncate** is selected in the **Cube actio**n box, specify which view name to truncate in the cube. If you do not specify a view name, by default, Data Manager truncates the whole cube using a default view name.
Data action	Specifies whether TM1 should • **Accumulate values** to deliver the delivery data values by accumulating them to existing data values. • **Store values** to deliver the delivery data values by overwriting existing data values. (default)
Measure dimension name	Specifies the name to use for the measure dimension in the cube. If you do not specify a name, Data Manager ignores the specified measures dimension action.
Measure dimension action	Specifies the dimension action to take for the measure: • **Create** to create a measures dimension for the delivered data. (default) • **Recreate** to delete an existing measures dimension and recreate it. • **Update** to deliver the data to an existing measures dimension. • **NoAction** to not create or update a measures dimension.
Floating point scale	Specifies the number of decimal places (for a numeric value) which must be converted to text. By default, this value is three decimal places.
TM1 prolog script	You can add to the TM1 prolog scripts if required. **Tip:** Click the browse button ... to type your script amendments in a text window.
TM1 epilog script	You can add to the TM1 epilog scripts if required. **Tip:** Click the browse button ... to type your script amendments in a text window.

Element Properties

The element properties available depend on the type of element being defined.

Property	Description
TM1 dimension name	Specifies the name of the TM1 dimension that corresponds to the IBM Cognos Data Manager dimensions.
Dimension action	Specifies the dimension action to take: • **Create** to create a dimension for the delivered data. (default) • **Recreate** to delete an existing dimension and recreate it. • **Update** to deliver the data to an existing dimension. • **NoAction** to not create or update a dimension.
TM1 element name expression	An expression that specifies how to construct the element names from the TM1 dimension. By default, Data Manager uses ID. This value is an expression, for example, `concat(CAPTION,'(',ID,')')` This value is mandatory.
Caption TM1 attribute name	Specifies the name of the TM1 attribute that corresponds to Data Manager dimension caption attribute (if one exists).
Caption TM1 attribute expression	An expression that specifies how to construct the TM1 attribute names from the TM1 dimension. By default, Data Manager uses CAPTION. This value is an expression, for example, `concat(CAPTION,'(',ID,')')`
Caption TM1 attribute type	Specifies the TM1 attribute type to use for captions: • **String** a string attribute. (default) • **Alias** a special attribute type.

Teradata Multiload Delivery Module

The Teradata Multiload delivery module delivers data directly to Teradata Multiload databases.

Note: You can access Teradata on a UNIX operating system using an ODBC driver. However, you must ensure that, in the ODBC.ini file, you use the data source name that you used on a Microsoft Windows operating system.

Module Properties

The table in this section describes the module properties.

Property	Description
Operation	Specifies whether IBM Cognos Data Manager should • **Run Loader** to deliver the text data and control files, load the data into the database and clean up the control files and any other intermediate files. (default) • **Run Loader (leave files)** to deliver the text data and control files, load the data into the database but leave the control files and any other intermediate files so that you can view them. • **Create Loader Files** to deliver the text data and control files, but not invoke the loader to load the data into the database.
File directory	Specifies the location in which to load data. By default, Data Manager loads data into its Data directory. **Tip:** To view the Data directory, from the **Tools** menu, click **Browse Data Files**.
Filename prefix	If **Create Loader Files** is selected in the **Operation** box, specify the prefix to use to generate the file names. If this box is empty, a default value is used.
Dump log file	Specifies whether to print the log file that the bulk loader generates.
No row count check	Specifies whether to check for successful delivery of the data. Clear this check box to prevent an error if the target DBMS does not report successful delivery in the expected format. Select this check box to use a newer version of the DBMS where the vendor changed the format of the completion message.
Log file success message	If **No row count check** is not selected, this is used to determine whether the load was successful by searching for the specified substring in the log file or output file. Data Manager searches for this string and the number of rows delivered as substrings in each line of the log file or output file.
Use named pipes	Specifies that Data Manager should use named pipes where possible. By default, Data Manager generates an intermediate text file and runs the bulk loader on it after delivering all data.
Field delimiter	Specifies the character sequence to use to separate fields. Specify a character string or use one of these named characters: Tab (the default), Comma, Space, None, or NL (new line).

Property	Description
Load mode	Specifies one of these load modes: • **Append** appends the data to the target table. (default) • **Truncate** deletes the contents of the target table, and drops all indexes, before delivering the data. This is equivalent to a DELETE FROM statement, but is usually faster. • **Replace** deletes the contents of the target table but not the indexes, before delivering the data. This is equivalent to a DELETE FROM statement.
Restart log table	Specifies a log table file name. Multiload uses the log table to restart a load from the last checkpoint if the load failed because the error limit was reached.
Data file encoding	Specifies the data file encoding to use.
Max sessions	Specifies the maximum number of loader sessions to use. If no value is specified, the Multiload default is used.
Min sessions	Specifies the minimum number of loader sessions to use. If no value is specified, the Multiload default is used.
Error file name	Specifies an alternate file name for Multiload error messages. This produces a duplicate record of all Multiload error messages.
Brief	Specifies that run-time output messages should be limited to the minimum required to determine whether the load is successful.
Encrypt	Specifies whether the session is to be encrypted.
Additional options	Specifies a string of extra load options to append to the command.
Route messages to (filename)	Specifies that run-time output messages should be routed to an alternative file.
Teradata multiload command	Specifies the path and command for the Multiload loader process. **Note:** You must specify the exact path name and command, including any double quotation marks required. For more information, see “Specify Executables to Deliver Data” on page 462.

Element Properties

The element properties available depend on the type of element being defined.

Property	Description
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example, to use a zero value enter 0
Perform escape processing	Specifies whether to use the escape character (\) before all backslash, tab, and new line characters in string values to ensure that the delivery module correctly interprets the statement.
Floating point scale	Specifies the number of decimal places for a numeric value that must be converted to text. By default, this value is three decimal places.

Teradata TPump Delivery Module

The Teradata TPump delivery module delivers data directly to Teradata TPUMP databases.

Note: You can access Teradata on a UNIX operating system using an ODBC driver. However, you must ensure that, in the ODBC.ini file, you use the data source name that you used on a Microsoft Windows operating system.

Module Properties

The table in this section describes the module properties.

Property	Description
Operation	Specifies whether IBM Cognos Data Manager should • **Run Loader** to deliver the text data and control files, load the data into the database and clean up the control files and any other intermediate files. (default) • **Run Loader (leave files)** to deliver the text data and control files, load the data into the database but leave the control files and any other intermediate files so that you can view them. • **Create Loader Files** to deliver the text data and control files, but not to invoke the loader to load the data into the database.
File directory	Specifies the location in which to load data. By default, Data Manager loads data into its Data directory. **Tip:** To view the Data directory, from the **Tools** menu, click **Browse Data Files**.

Property	Description
Filename prefix	If **Create Loader Files** is selected in the **Operation** box, specify the prefix to use to generate the file names. If this box is empty, a default value is used.
Dump log file	Specifies whether to print the log file that the bulk loader generates.
No row count check	Specifies whether to check for successful delivery of the data. Clear this check box to prevent an error if the target DBMS does not report successful delivery in the expected format. Select this check box to use a newer version of the DBMS where the vendor changed the format of the completion message.
Log file success message	If **No row count check** is not selected, this is used to determine whether the load was successful by searching for the specified substring in the log file or output file. Data Manager searches for this string and the number of rows delivered as substrings in each line of the log file or output file.
Use named pipes	Specifies that Data Manager should use named pipes where possible. By default, Data Manager generates an intermediate text file and runs the bulk loader on it after delivering all data.
Field delimiter	Specifies the character sequence to use to separate fields. Specify a character string or use one of these named characters: Tab (the default), Comma, Space, None, or NL (new line).
Load mode	Specifies one of these load modes: • **Append** appends the data to the target table. (default) • **Truncate** deletes the contents of the target table, and drops all indexes, before delivering the data. This is equivalent to a DELETE FROM statement, but is usually faster. • **Replace** deletes the contents of the target table but not the indexes, before delivering the data. This is equivalent to a DELETE FROM statement.
Restart log table	Specifies a log table file name. TPump uses the log table to restart a load from the last checkpoint if the load failed because the error limit was reached.
Data file encoding	Specifies the data file encoding to use.
Control file encoding	Specifies the control file encoding to use.
Error file name	Specifies an alternate file name for TPump error messages. This produces a duplicate record of all TPump error messages.
Config file name	Specifies the configuration file that contains configuration and tuning parameters for TPump.
Brief	Specifies that run-time output should be limited to the minimum required to determine whether the load is successful.

Property	Description
Verbose	Specifies whether additional statistical data is required.
Encrypt	Specifies whether the session is to be encrypted.
Additional options	Specifies a string of extra load options to append to the command. SESSIONS specifies the number of TPump sessions to use.
Route messages to (filename)	Specifies that run-time output messages should be routed to an alternative file.
Teradata TPump command	Specifies the path and command for the TPump loader process. **Note:** You must specify the exact path name and command, including any double quotation marks required. For more information, see "Specify Executables to Deliver Data" on page 462.

Element Properties

For this delivery module, the element properties are the same for all element types.

Property	Description
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example, to use a zero value enter 0
Perform escape processing	Specifies whether to use the escape character (\) before all backslash, tab, and new line characters in string values to ensure that the delivery module correctly interprets the statement.
Floating point scale	Specifies the number of decimal places for a numeric value that must be converted to text. By default, this value is three decimal places.

Text File Delivery Module

The Text File delivery module formats build data into simple text files. These files can then be imported into spreadsheets for crosstab (pivot table) analysis, or can be distributed to other systems.

Note: You can also deliver to text files that have been defined using IBM Cognos Data Manager SQLTXT Designer. This gives you more control over the format of the text file, although it is more time consuming to set up. For more information on using Data Manager SQLTXT Designer, see Chapter 30, "IBM Cognos Data Manager SQLTXT Designer," on page 351.

Module Properties

The table in this section describes the module properties.

Property	Description
Output filename	Specifies the path and name of the file to receive the data. If you enter only a file name, IBM Cognos Data Manager adds the file to its Data directory. This value is mandatory. **Tip:** To view the Data directory, from the **Tools** menu, click **Browse Data Files**.
Line delimiter	Specifies the character sequence to use to separate data rows in the output file. Specify a character string or use one of these named characters: Tab, Comma, Space, None, or NL (new line). NL is the default.
Default field delimiter	Specifies the character sequence to use to separate fields in the output file. Specify a character string or use one of these named characters: Tab (the default), Comma, Space, None, or NL (new line).
Default value if null	Specifies the value to represent null data values. By default, this value is a blank string for character values and 0 for numeric values.
Default date format	Specifies the format to use to write date/time values to the output file. If Data Manager recognizes a value as a date or time, it formats the value in its native format. Otherwise, it formats the value in the default format for the originating database. If the specified format is invalid for a particular value, Data Manager performs no formatting on that value. For information about date formats, see Appendix D, "Date, Time, and Interval Formats," on page 485.
Default scale	Specifies the number of decimal places to use for all numeric values in the output file.
Include field titles	Specifies whether to include field titles, containing the name of each element, as the first line of the output file. By default, field titles are not included.
Quote fields	Specifies whether to use double quotation marks to enclose field values. By default, no quotation marks are used.
Encoding	Specifies the encoding in which the text file is to be produced. **(Default)** is the local computer encoding, for example windows-1252 on a Microsoft Windows operating system in Western Europe. The others are standard Unicode encodings.

Element Properties

For this delivery module, the element properties are the same for all element types.

Property	Description
Field delimiter	Specifies the character sequence to use to separate fields in the output file. Specify a character string or use one of these named characters: Tab (the default), Comma, Space, None, or NL (new line).
Value if null	Specifies the value to represent null data values in the element. This is necessary when the column in the target table does not directly support null. To specify a blank string for character values, enter ' ' that is, a single quotation mark, a single space, and a single quotation mark. For numeric values, do not use single quotation marks. For example, to use a zero value, enter 0
Print format	Specifies the format used to print values for an element in the output file. The syntax is `[-][0]<precision>.][.<scale>]` The symbols represent the following: • - left justify the field • 0 leading zero padding for numeric values • <precision> the minimum number of characters to write • <scale> for numeric values, the maximum number of decimal places or character values to write By default, IBM Cognos Data Manager applies no special formatting to character and long values. However, Data Manager writes double-precision floating point values with this format: .6
Date Format	Specifies the format to use to write date/time values to the output file. If Data Manager recognizes a value as a date or time, it formats the value in its native format. Otherwise, it formats the value in the default format for the originating database. If the specified format is invalid for a particular value, Data Manager performs no formatting on that value. For information about date formats, see Appendix D, "Date, Time, and Interval Formats," on page 485.

Specify Executables to Deliver Data

For some delivery modules, you can specify the executable which IBM Cognos Data Manager uses to deliver the data. You may want to do this to use a non-default loader. For example, by default, Data Manager uses sqlldr to deliver to Oracle SQL*Net. However, Oracle 8 uses sqlldr80 and you would specify this loader for computers that use older versions of Oracle. You can specify executables for each

- delivery, by setting a module property
- session, by setting an environment variable

When delivering data, Data Manager searches for the executable to use as follows:

Procedure

1. Data Manager checks whether a delivery module command property is set. If found, this is used to deliver the data.
2. If no delivery module command property is set, Data Manager searches for the appropriate dm.ini entry or environment variable value. If found, the value is double-quoted as required, then used to deliver the data.

   Note: Environment variables take precedence over the dm.ini file entry.
3. If the two searches fail, Data Manager uses the default command.

   For a list of default commands, see “Set the Name of the Loader Executable in a Module Property.”

Set the Name of the Loader Executable in a Module Property

This section describes how to set the name of the loader executable in a module property.

Procedure

1. Open the appropriate **Delivery Properties** window.
2. Click the **Module Properties** tab.
3. In the relevant **Command** box, enter the name of the required loader executable.

   The following table contains default command names for each delivery module.

Delivery module	Default command
DB2 LOAD	Microsoft Windows operating system: db2cmd /c /w /i db2 UNIX operating system: db2
Informix LOAD	dbaccess
Microsoft SQL Server BCP Command	bcp
Netezza NZLoad	nzload
Oracle SQL*Loader	sqlldr
Red Brick Loader (TMU)	rb_tmu

Delivery module	Default command
Sybase ASE BCP Command	bcp
Sybase IQ LOAD	dbisql
Teradata Fastload	fastload
Teradata Multiload	mload
Teradata TPump	Windows: tpump UNIX: tpmump

Set the Name of the Executable in an Environment Variable

The following table gives the name of the environment variables and the associated delivery modules.

Environment variable	Delivery module
DS_DB2_LOADER	DB2 UDB Microsoft Windows/UNIX operating system bulk loader command
DS_INFORMIX_LOADER	Informix bulk loader command
DS_MSSQLSERVER_BCP	Microsoft SQL Server BCP bulk loader command
DS_NETEZZA_LOADER	Netezza bulk loader command
DS_ORACLE_LOADER	Oracle bulk loader command
DS_REDBRICK_LOADER	Red Brick bulk loader command
DS_SYBIQ_LOADER	Sybase IQ bulk loader command
DS_SYBASE_LOADER	Sybase ASE BCP command
DS_TERADATA_FASTLOAD	Teradata Fastload bulk loader command
DS_TERADATA_MULTILOAD	Teradata Multiload bulk loader command
DS_TERADATA_TPUMP	Teradata TPump bulk loader command

For information about setting environment variables, see Chapter 24, “Variables,” on page 289.

Specify the Location of Executables

For each executable that you specify, you must set up its location on the computer on which you execute builds.

Note: For DB2LOAD delivery module, ensure you use db2.exe, not db2cmd.exe.

Procedure

1. Open the file named dm.ini which, by default, is located in Program Files\ibm\cognos\c10\datamanager
2. In the Program Locations section, type the locations of the required executables.
3. Save the file.

Appendix B. Database Drivers

The IBM Cognos Data Manager database drivers present a common interface to the Data Manager software. This allows you to treat all data sources in the same manner. In addition to performing common functions such as extracting data, Data Manager can

- connect to the required database
- convert the native data types of each database to and from its own (internal) data types
- convert the native date formats to and from the internal format that Data Manager uses

Data Manager supports these DBMS drivers:

- IBM Cognos Data Source
- DB2
- DTS Package (using OLE-DB)
- Essbase
- INFORMIX (using Connect)
- ODBC
- Oracle (using SQL*Net)
- Published FM Package
- Microsoft SQL Server (using OLE-DB)
- SAP R/3
- SQLTXT
- Sybase (using CT-Library)
- TM1

Data Manager uses connections to connect to the required data sources. For more information, see Chapter 7, “Connections,” on page 39.

For an up-to-date list of environments supported by IBM Cognos products, such as operating systems, browsers, Web servers, directory servers, database servers, OLAP servers and more, see http://www.ibm.com/software/data/support/cognos_crc.html.

Notes®

- Data Manager only enables those drivers that it detects are installed.
-

  Some databases impose limits on the length of table names. This may result in the names being truncated and possibly becoming duplicates.

Data Types

To provide consistency, Data Manager converts the native data types of each DBMS to its internal data types. This allows it to acquire, transform, and deliver data, irrespective of the source and destination DBMS.

For information on Data Manager internal data types, see Appendix E, “Data Types,” on page 491.

Data Manager can automatically convert most data types. However, unless mentioned in the DBMS Drivers section, other types are not supported unless they can be cast by the database into a type that Data Manager supports.

IBM Cognos Data Source Driver

The IBM Cognos Data Manager driver for IBM Cognos data sources provides a method of accessing any data source set up in IBM Cognos Business Intelligence that accepts Cognos SQL. When the IBM Cognos BI environment has been installed and configured, these data sources become available for consumption as Data Manager sources.

Data type conversion follows the rules of the driver that the data source represents.

DB2 DBMS Driver

This section includes information about data type conversions and date format conversions for this driver.

Data Type Conversions

The conversion between DB2 data types and IBM Cognos Data Manager data types varies slightly depending on the version of DB2 and the operating system on which it runs.

You can discover the conversions for a particular DB2 data source by connecting to the data source in SQLTERM (see CONNECT and DBALIAS) and issuing a TYPES command. For more information, see “sqlterm” on page 399.

Note: If you are using DB2version 8.2 client, to use the GRAPHIC, VARGRAPHIC, and DBGRAPHIC data types you must add the following line to the [COMMON] section of the db2cli.ini file:

```
Graphic=1
```

Example

```
UDA Driver Information
===================================================================
Driver is DB2
Data Source is SQL
Client Version is 03.01.0000
DBMS Version is 08.02.0002
Maximum Schema length is 30
Maximum Table length is 128
Maximum Column length is 30
Default Identifier case 'UPPER CASE', quote character '"'
Transactions supported.
Cursor COMMIT behavior DISRUPTIVE
DB/2 cli Information
===================================================================
ODBC version 3.52
```

```
DBMS is 'DB2/NT'
Driver version 08.01.0000, ODBC version 03.0
ODBC conformance is 'Level 2'
NOT NULLs supported, Transactions fully supported.
DBMS to ODBC type conversion :
DBMS Type (precision,min,max)     ( No) Parameters  Nullable?   ODBC Type
==============================================================================
BLOB  (2147483647,0,0)               -4       1     Nullable    BLOB
VARCHAR () FOR BIT DATA  (32672,0,0) -3       1     Nullable    VARBINARY
CHAR () FOR BIT DATA  (254,0,0)      -2       1     Nullable    BINARY
CLOB  (2147483647,0,0)               -1       1     Nullable    TEXT
CHAR  (254,0,0)                       1       1     Nullable    CHAR
NUMERIC  (31,0,31)                    2       2     Nullable    NUMERIC
DECIMAL  (31,0,31)                    3       2     Nullable    DECIMAL
INTEGER  (10,0,0)                     4       0     Nullable    INTEGER
SMALLINT  (5,0,0)                     5       0     Nullable    SMALLINT
FLOAT  (53,0,0)                       6       0     Nullable    DOUBLE
REAL  (24,0,0)                        7       0     Nullable    FLOAT
DOUBLE  (53,0,0)                      8       0     Nullable    DOUBLE
DATE  (10,0,0)                       91       0     Nullable    DATE2
TIME  (8,0,0)                        92       0     Nullable    TIME2
TIMESTAMP  (26,6,6)                  93       0     Nullable    TIMESTAMP2
VARCHAR  (32672,0,0)                 12       1     Nullable    VARCHAR
VARGRAPHIC  (16336,0,0)             -96       1     Nullable    NVARCHAR
GRAPHIC  (127,0,0)                  -95       1     Nullable    NCHAR
DBCLOB  (1073741823,0,0)            -97       1     Nullable    TEXT
DATALINK  (254,0,0)                 -400      0     Nullable    (none)
BIGINT  (20,0,0)                     -5       0     Nullable    QUAD
ODBC to DBMS type conversion :
ODBC Type         DBMS Type              Basis
===================================================
CHAR              CHAR                   ODBC Match
VARCHAR           VARCHAR                ODBC Match
NCHAR             GRAPHIC                ODBC Match
NVARCHAR          VARGRAPHIC             ODBC Match
BOOLEAN           SMALLINT               Best Match
SMALLINT          SMALLINT               ODBC Match
INTEGER           INTEGER                ODBC Match
DECIMAL           DECIMAL                ODBC Match
NUMERIC           NUMERIC                ODBC Match
FLOAT             REAL                   ODBC Match
DOUBLE            DOUBLE                 ODBC Match
QUAD              BIGINT                 ODBC Match
DATE              DATE                   Best Match
TIME              TIME                   Best Match
TIMESTAMP         TIMESTAMP              Best Match
INTERVAL          VARCHAR                Best Match
DATE2             DATE                   ODBC Match
```

TIME2	TIME	ODBC Match
TIMESTAMP2	TIMESTAMP	ODBC Match
TIMETZ	VARCHAR	Best Match
TIMESTAMPTZ	VARCHAR	Best Match
INTERVAL2	VARCHAR	Best Match
INTERVALYM	VARCHAR	Best Match
BINARY	CHAR () FOR BIT DATA	ODBC Match
VARBINARY	VARCHAR ()FOR BIT DATA	ODBC Match
TEXT	CLOB	ODBC Match
BLOB	BLOB	ODBC Match

Date Format Conversions

DB2 date formats are identical to IBM Cognos Data Manager date formats so no conversion is required.

DTS Package Driver

You can process data in DTS packages before connecting to it in IBM Cognos Data Manager to reuse it. This means that you do not have to stage the data in Data Manager.

You can connect to data stored in DTS packages, but you cannot deliver data to them as this is a read-only driver.

You can use the following SQL statements:

```
SELECT *
SELECT * FROM ALL
SELECT * FROM <packagename>
SELECT * FROM <stepname>
```

Note: When using DTS packages in Data Manager, in the Microsoft **DTS Workflow Properties**window, select **DSO Rowset**.

Essbase DBMS Driver

Essbase drivers do not require data type conversion or date format conversion.

INFORMIX (via Connect) DBMS Driver

This section includes information about data type conversions and date format conversions for this driver.

Data Type Conversion

The following table shows the INFORMIX data types that IBM Cognos Data Manager supports.

Data Manager	INFORMIX
BINARY	BOOLEAN if prec=1, otherwise CHAR
BLOB	BYTE

Data Manager	**INFORMIX**
BOOLEAN	BOOLEAN
CHAR (p)	CHAR (p>255) VARCHAR (p <= 255)
CLOB	TEXT
DATE	DATETIME YEAR TO SECOND DATETIME YEAR TO DAY
DATE WITH TIME ZONE	VARCHAR(75)
FLOAT	FLOAT

INTEGER (p)	NUMERIC(10) (p<=2)
	NUMERIC(19) (p>2)

INTERVAL DAY TO SECOND	INTERVAL DAY TO SECOND
INTERVAL YEAR TO MONTH	INTERVAL YEAR TO MONTH
NUMBER	NUMERIC(32)
TIME	DATETIME HOUR TO SECOND
TIME WITH TIME ZONE	VARCHAR(75)

Date Format Conversion

Specify the date format within the build or environment variable 'DS_DATEFMT_INFORMIX'.

ODBC DBMS Driver

IBM Cognos Data Manager supports ODBC data sources for Microsoft Windows and UNIX operating systems.

ODBC allows applications to interact with many DBMSs, using a single common interface, using SQL. DBMS-specific drivers provide an interface between client applications and the ODBC core. You can purchase these drivers from a number of vendors. In addition, individual software manufactures may provide ODBC drivers for their products.

An ODBC driver must conform to the ODBC 2.5 specification or higher.

SQL Syntax

Each ODBC driver should provide at least ANSI SQL compatibility. In many cases, an ODBC driver allows the native SQL dialect of the associated DBMS. Because

IBM Cognos Data Manager passes the SQL statements that you provide unchanged to the ODBC driver, you can use extensions. As with every other driver, Cognos SQL can also be used.

Network Data Sources

Some ODBC drivers require the existence of other software to establish their connections to the data. Some drivers have server and client portions that provide an alternative to the DBMS-specific network software.

Connecting to ODBC Data Sources

IBM Cognos Data Manager connects to ODBC data sources by reference to the appropriate ODBC data source name (DSN). How you configure a particular DSN depends upon the operating system and upon the particular driver that you use to perform the connection. Because there are many differences between the operating system and ODBC driver combinations, we do not attempt to describe the DSN configuration process. For this information, see the documentation that accompanies individual ODBC drivers and the ODBC core software.

Data Type Conversion

For data manipulation, ODBC reconciles disparate data types before the information reaches IBM Cognos Data Manager. Data Manager does not normally need to perform data type conversion on an ODBC data source. However, Data Manager must work with the native data types of the DBMS when it performs some data definition operations (for example, table creation). In such cases, Data Manager selects compatible native data types onto which to map its internal data types.

Sometimes the DBMS has more than one candidate for a single Data Manager internal data type. For example, a DBMS may support NUMBER, MONEY, and DECIMAL as floating-point data types and each is a suitable candidate onto which Data Manager can map its FLOAT data type. It is recommended that you do not use Data Manager to perform data definition operations where such ambiguity exists.

Date Format Conversion

ODBC date formats are identical to IBM Cognos Data Manager date formats, so date format conversion is not required.

Oracle (via SQL*Net) DBMS Driver

This section includes information about data type conversions and date format conversions for this driver.

Data Type Conversion

The following table shows the Oracle data types that IBM Cognos Data Manager supports.

Data Manager	**ORACLE**
BINARY	RAW
BLOB	BLOB

Data Manager	ORACLE
BOOLEAN	INTEGER
CHAR	CHAR NCHAR (Unicode) VARCHAR2 NVARCHAR2 (Unicode)
CLOB	CLOB
DATE	TIMESTAMP (date and time) DATE (date only)
DATE WITH TIME ZONE	TIMESTAMP WITH TIMEZONE
FLOAT	FLOAT

INTEGER (p)	NUMBER(5) (p<=2)
	NUMBER(10) (2<p<=4)
	NUMBER(15) (p>4)

INTERVAL DAY TO SECOND	INTERVAL DAY TO SECOND
INTERVAL YEAR TO MONTH	INTERVAL YEAR TO MONTH
NUMBER	NUMBER(38)
TIME	VARCHAR(12)
TIME WITH TIME ZONE	VARCHAR(75)

If Data Manager encounters other Oracle data types, it returns null.

Date Format Conversion

In previous versions of IBM Cognos Data Manager, by default, the returned value was set to YYYY-MM-DD HH24:MI:SS.

This default has been removed. If you want to set the returned value to YYYY-MM-DD HH24:MI:SS, you must now enable it using the variable named DM_NLS_DATE_FORMAT_COMPATIBILITY.

Published FM Package Driver

The IBM Cognos Data Manager driver for published IBM Cognos Framework Manager packages differs from other database drivers because it is a model that exists over many other data sources. Data Manager does not perform any data type and date conversion operations; these are performed in the source model.

The driver provides a method of accessing any data source that has been modeled and published by Framework Manager. When the IBM Cognos Business Intelligence environment has been installed and configured, published packages become available for consumption as Data Manager sources. This allows you to extend the reach of Data Manager to new data sources, such as OLAP sources and additional ODBC coverage.

You can interact with, and select, query items from the data source using a visual query-building environment within Data Manager Designer. This creates an XML-based query specification document that describes the data to be returned. Normally, Data Manager does this automatically, but you can choose to supply the query specification externally for Data Manager to use directly as a data source. This is an advanced capability for query users who are familiar with the IBM Cognos Software Development capability.

Note: Although it is possible to read any supported relational data source using published packages, such as Oracle sources, unless the sources have been modeled and published for other reasons, it is far more practical to use the Data Manager connection types directly. Data Manager provides a more direct access method for data sources. The driver for published packages is intended for situations where modeled data from relational or OLAP sources is required (such as dimensional data structures) or, for access to web-based data sources (such as web services data).

Microsoft SQL Server (via OLE-DB) DBMS Drivers

This section includes information about data type conversions and date format conversions for these drivers.

Data Type Conversions

The following table shows the Microsoft SQL Server 2005 and above data types that IBM Cognos Data Manager supports.

Data Manager	MSSQL
BINARY	BINARY VARBINARY TIMESTAMP UNIQUEIDENTIFIER
BLOB	IMAGE
BOOLEAN	SMALLINT
CHAR	CHAR NCHAR (Unicode) VARCHAR NVARCHAR (Unicode) SQL_VARIANT

Data Manager	MSSQL
CLOB	TEXT NTEXT
DATE	DATETIME SMALLDATETIME
DATE WITH TIME ZONE	VARCHAR(75)
FLOAT	FLOAT REAL
INTEGER (p)	TINYINT (p=1) SMALLINT (p=2) INTEGER (2<p<=4) BIGINT (4<p<=8) BIT
INTERVAL DAY TO SECOND	VARCHAR(30)
INTERVAL YEAR TO MONTH	VARCHAR(75)
NUMBER	NUMERIC(38) DECIMAL MONEY SMALLMONEY
TIME	VARCHAR(12)
TIME WITH TIME ZONE	VARCHAR(75)

Date Format Conversions

Specify the date format within the build or environment variable 'DS_DATEFMT_MSSQL'.

For more information, see “DBMS-related Variables” on page 304.

SAP R/3 DBMS Driver

This section includes information about data type conversions and date format conversions for this driver.

Data Type Conversion

The following table shows the SAP/ABAP data types that IBM Cognos Data Manager supports.

Data Manager	SAP
CHAR	ACCP CHAR CLNT CUKY LANG LCHR LRAW NUMC PREC RAW UNIT
NUMBER	CURR DEC QUAN
DATE	DATS
DOUBLE	FLTP
INTEGER	INT1 INT2 INT4

Date Format Conversion

No date format conversion is required.

SQLTXT DBMS Driver

The SQLTXT DBMS driver lets you access data in delimited text format using SQL.

For the SQLTXT format specification, see Chapter 30, “IBM Cognos Data Manager SQLTXT Designer,” on page 351.

Data Type Conversion

SQLTXT has three data types that correspond to the IBM Cognos Data Manager FLOAT data type. These are FLOAT, PACKED, and ZONED.

FLOAT

The FLOAT data type is valid in both delimited and fixed-width text data files. It is a text representation of a floating-point number, space-padded in fixed-length data, encoded as ASCII or EBCDIC to comply with the character set of the text data file.

PACKED

The PACKED data type is valid only in fixed-width text data files. It is a packed decimal type. This format stores each decimal digit (in hexadecimal) in one nibble (half a byte), with a further nibble storing the sign (plus or minus). This data type requires half the storage of the FLOAT data type. The low-order nibble stores the sign of the number. This format does not indicate the position of the decimal point. SQLTXT stores scale information in the SQLTXT catalog.

Because the least significant byte stores one decimal digit and the sign of the number, this format can only represent an odd number of decimal digits. If the number to be encoded has an even number of digits, then you must add a leading zero.

The following table gives the permitted values of the sign nibble, together with the meaning of each value.

Value (Hex)	Meaning
0x0A	Positive
0x0B	Negative
0x0C	Positive (preferred)
0x0D	Negative (preferred)
0x0E	Positive
0x0F	Positive

Example

The following table shows the PACKED encoding of some example decimal numbers.

Decimal	PACKED (Hex)	Comment
1,234,567.89	12 34 56 78 9C	Positive
123,456.789	12 34 56 78 9C	Positive
1,234,567.89	12 34 56 78 9F	Positive
123,456.789	12 34 56 78 9F	Positive
-1,234,567.89	12 34 56 78 9D	Negative
-123,456.789	12 34 56 78 9D	Negative

Each nibble, except the least significant, encodes a decimal digit as a hexadecimal number. The least significant nibble of each number indicates its sign. The examples use the preferred 0x0C for positive numbers, and 0x0D for negative numbers. This method of encoding does not indicate the position of the decimal point. Therefore, the packed representations of 1,234,567.89 and 123,456.789 are identical. Similarly, the representations of -1,234,567.89 and -123,456.789 are identical.

ZONED

The ZONED data type is valid only in fixed-width text data files. It uses one byte for each digit. Except for the least significant byte, the most significant nibble of each byte contains the zone code (0xF0). For the least significant byte, this nibble stores the sign. The least significant nibble of every byte contains the decimal digit, represented as a hexadecimal number. This format does not indicate the position of the decimal point. SQLTXT stores scale information in the SQLTXT catalog. This method encodes decimal digits as their EBCDIC character equivalents. Except for the least significant digit, you can view the decimal numbers in a standard EBCDIC text editor.

The following table gives the permitted sign values and their meanings.

Value (Hex)	Meaning
0xA0	Positive
0xB0	Negative
0xC0	Positive (preferred)
0xD0	Negative (preferred)
0xE0	Positive
0xF0	Positive (this is also the zone code for all but the least significant byte)

The following table shows the ZONED encoding of some example decimal numbers.

Decimal	ZONED (Hex)	Comment
1,234,567.89	F1 F2 F3 F4 F5 F6 F7 F8 C9	Positive
123,456.789	F1 F2 F3 F4 F5 F6 F7 F8 C9	Positive
-1,234,567.89	F1 F2 F3 F4 F5 F6 F7 F8 D9	Negative
-123,456.789	F1 F2 F3 F4 F5 F6 F7 F8 D9	Negative

The low-order nibble of each byte encodes a single digit as a hexadecimal number. For all but the least significant byte, the high-order nibble contains the zone code (0xF0). The high-order nibble of the least significant byte indicates the sign of the value. For the first two numbers, this is 0xC0, indicating positive numbers; for the last two, this is 0xD0, indicating negative numbers. Because this method of encoding does not indicate the position of the decimal point, the zoned representations of the two positive numbers are identical. Similarly, the zoned representations of the two negative numbers are identical.

BOOLEAN

SQLTXT also has a BOOLEAN data type. This encodes the Boolean values TRUE and FALSE, as string representations of numbers with zero values representing FALSE, and non-zero values representing TRUE.

Other Data Types

For other data types, SQLTXT uses the same data types as IBM Cognos Data Manager, so no conversion is required.

Date Format Conversion

Internally, SQLTXT handles dates in IBM Cognos Data Manager native format, so no conversion is required.

Sybase (via CT-Library) DBMS Driver

This section includes information about data type conversions and date format conversions for this driver.

Data Type Conversion

The following table shows the Sybase data types that IBM Cognos Data Manager supports.

Data Manager	**SYBASE**
BINARY	BINARY VARBINARY
BLOB	IMAGE
BOOLEAN	SMALLINT
CHAR	CHAR NCHAR (Unicode) VARCHAR NVARCHAR (Unicode)
CLOB	TEXT
DATE	DATETIME
DATE WITH TIME ZONE	VARCHAR(75)
FLOAT	DOUBLE PRECISION

Data Manager	**SYBASE**
INTEGER (p)	TINYINT (p=1) SMALLINT (p=2) INTEGER (2<p<=4) BIGINT (4<p<=8)
INTERVAL DAY TO SECOND	VARCHAR(30)
INTERVAL YEAR TO MONTH	VARCHAR(75)
NUMBER	NUMERIC(38)
TIME	DATETIME
TIME WITH TIME ZONE	VARCHAR(75)

If Data Manager encounters other Sybase data types, it produces a run-time error.

Date Format Conversion

You must specify the date format within the build or environment variable 'DS_DATEFMT_SYBASE'.

For more information, see “DBMS-related Variables” on page 304.

TM1 DBMS Driver

IBM Cognos TM1 drivers do not require data type conversion or date format conversion.

Appendix C. Audit Tables

The audit tables are used to provide summary information about the audit history, the audit trail, the audit messages, and the JobStream details.

There are the following audit tables:

- component run (dsb_component_run)
- context run (dsb_run_context)
- delivery history (dsb_delivery_hist)
- audit trail (dsb_audit_trail)
- audit message (dsb_audit_msg)
- audit message line (dsb_audit_msg_line)
- job node run (dsb_jobnode_run)

You can view the audit tables by creating a connection to the catalog and opening the connection in SQLTerm.

Component Run Table (dsb_component_run)

The component run table contains two records for each execution of a build or JobStream. IBM Cognos Data Manager creates a unique run_id for each type of component, and a unique component_uuid for each component executed.

At the start of the execution process, a record indicating the start time and the execution mode is created, and the status column is set to R (running).

On completion, the record is updated to add the end time and the number of rows rejected, and the completion status is set to S if execution was successful, or F if it failed.

Column	Description
audit_id	A number that is unique for each execution process. This is a surrogate key for the component_uuid and run_id natural keys.
component_uuid	A number that is universally unique for each component executed.
run_id	A number that is unique for each type of component in the build or JobStream.
start_timestamp	The start time of the execution process.
end_timestamp	The end time of the execution process.

Column	Description
status	The status of the execution process: • R indicates the process is running • S indicates the process reached a successful completion • F indicates the process failed due to errors • K indicates the process was successfully stopped manually • D indicates the process failed
process_info	A unique number to identify the process that ran the component.
component_type	The component type: • B indicates a fact build • S indicates a dimension build • J indicates a JobStream • X indicates that this is an additional record
component_name	The name of the component, or for type X components, the name of the computer on which the component was executed.

Run Context Table (dsb_run_context)

The run context table contains the context details of a build or JobStream execution.

Column	Description
audit_id	A number that is unique for each component executed.
run_mode	The execution mode and the delivery types: • N indicates normal execution mode • C indicates check only execution mode • O indicates object creation execution mode • F indicates a fact delivery • D indicates a dimension delivery
package_run	A Boolean value that indicates whether the execution process occurred from within a package: • Y indicates the execution process occurred from within a package • N indicates the execution process did not occur from within a package
hostname	The name of the computer on which the execution process occurred.

Delivery History Table (dsb_delivery_hist)

The delivery history table records the number of rows inserted or updated for each table targeted by the execution process. For each table, the name of the database, the table name, whether there were any existing rows in the table, the number of rows that were inserted, and the number of rows that were updated are also recorded.

Note: In the case of text file delivery, the file name is recorded in the table_name column and the dbalias column is left blank.

Column	Description
audit_id	A number that is unique for each component executed.
dbalias	The name of the database to which the data was delivered.
table_name	The name of the table into which the data was delivered.
existing_data	A Boolean value that indicates whether there was existing data in the table: • Y indicates there was existing data • N indicates there was no existing data
rows_inserted	The number of rows that were successfully inserted.
rows_updated	The number of rows that were successfully updated.

Audit Trail Table (dsb_audit_trail)

The audit trail table records all events from the fact build, dimension build, or JobStream execution process. Events are classified by group and item with a message for each. Some of the messages are text, and others are numbers that you can use for analysis.

You must specify in the relevant Properties window the events to record.

Column	Description
audit_id	A number that is unique for each component executed.
audit_timestamp	The start and end time that each row of data was written.
audit_group	The audit group to which information is written.
audit_item	The audit item of the group to which information is written.
audit_msg	The message from the audit item.

Audit Message Table (dsb_audit_msg)

The audit message table identifies a message that has been recorded when a fact build, dimension build, or JobStream is executed.

Column	Description
audit_id	A number that is unique for each component executed.
message_no	A unique number that identifies each message that is recorded.
message_code	The code for the message, as shown in the log file.
severity	The severity of the message: • F indicates a fatal message • E indicates an error • W indicates a warning

Audit Message Line Table (dsb_audit_msg_line)

The audit message line table records, for each message, the actual lines of message text.

Column	Description
audit_id	A number that is unique for each component executed.
message_no	A unique number that identifies each message that is recorded.
line_no	A unique number that identifies the line number of each message.
message_text	The message text.

Job Node Run Table (dsb_jobnode_run)

This table stores a record for each node in a JobStream. If the node is a fact build node, dimension build node, or JobStream node, it is allocated an audit_id and run_id. These IDs are recorded in the ds_component_run table.

Column	Description
job_audit_id	A number that is unique for the JobStream currently being executed.
node_id	The number of the JobStream node.

Column	Description
node_type	The node type: • B indicates a fact build node • D indicates a dimension build node • C indicates a condition node • S indicates an SQL node • P indicates a procedure node • E indicates an email node • A indicates an alert node • J indicates a JobStream node
node_caption	The name of the JobStream node.
start_timestamp	The start time of the execution process.
end_timestamp	The end time of the execution process.
status	The status of the process: • R indicates the process is running • S indicates the process reached a successful completion • F indicates the process failed due to errors • K indicates the process was successfully stopped manually • D indicates the process failed
masterjob_audit_id	A number that is unique for the master JobStream execution.
comp_audit_id	For fact build, dimension build, and JobStream node types, a number that identifies the component execution.

Appendix D. Date, Time, and Interval Formats

This section describes the date, time, and interval formats used by IBM Cognos Data Manager.

Date Formats

IBM Cognos Data Manager uses syyyy-mm-dd [hh:mi:ss[.fffffffff]] as the default date format.

English is the default language used to specify months and days of the week. However, you can customize Data Manager to use an alternative language for these names.

If you want to use French or German date formats, Data Manager includes both month and day files translated into French and German. For more information, see the IBM Cognos Data Manager *Installation and Configuration Guide*.

Date formats can contain any of these symbols, interspersed with format text.

Symbol	**Description**
- / . ; : "text"	Punctuation and text that is reproduced in the result **Note:** You must enclose literal text within double quotation marks.
am/pm	am/pm indicator
d	Day of the week (1 to 7)
day	Name of the day padded with blanks to a length of nine characters
dd	Day of the month (1 to 31)
ddd	Day of the year (1 to 366)
dy	Abbreviated name of the day of the week
fffffffff	Fractions of a second (maximum of 9)
hh	Hour of the day using the 24 hour clock
hh12	Hour of the day using the 12 hour clock
hh24	Hour of the day using the 24 hour clock
iw	Week of the year (1 to 52 or 1 to 53) based on the ISO standard
iyy	Last three digits of the ISO year
iy	Last two digits of the ISO year

Symbol	Description
i	Last digit of the ISO year
iyyy	Four digit year based on the ISO year
j	Julian day. That is, the number of days since January 1st, 4712 BC. This number must be an integer
m	Month number (1 to 12). Months 1 to 9 are not zero padded
mm	Month number (01 to 12). Months 1 to 9 are zero padded
mi	Minute (00 to 59)
mmm (or mon)	Lower case, three letter abbreviation of the month name (jan to dec)
Mmm (or Mon)	Proper case, three letter abbreviation of the month name (Jan to Dec)
MMM (or MON)	Upper case, three letter abbreviation of the month name (JAN to DEC)
month	Lower case month name (january to december). Where necessary months are zero padded
Month	Proper case month name (January to December). Where necessary months are zero padded
MONTH	Upper case month name (JANUARY to DECEMBER). Where necessary months are zero padded
Q	Quarter of year (1 to 4) where quarter 1 is January to March inclusive
ss	Seconds (00 to 59)
sssss	Seconds since midnight (0 to 86399)
stzh	Signed number of hour for the time zone difference (-12 to 13)
tzm	Number of minutes for the time zone difference (00 to 59)
w	Week of the month (1 to 5) where week 1 starts on the first day of the month and ends on the seventh day of the month
ww	Week of the year (1 to 53) where week 1 starts on the first day of the year and continues to the seventh day of the year
yyyy	Four digit year
syyyy	Four digit year. Dates BC are prefixed by -

Symbol	Description
yyy	Last three digits of the year
yy	Last two digits of the year
y	Last digit of the year

Examples

The following examples refer to a date and time of June 22 2006 17:35:29.123.

Format	Result
dd/mm/yy hh:mi:ss	22/06/06 17:35:29
hh:mi:ss.ffff	17:35:29.1230
dd-Mmm-yyyy hh12:mi pm	22-Jun-2006 05:35 pm

The following examples refer to a date and time of June 22 2006 17:35:29.123 -5:00.

Format	Result
dd/mm/yy hh:mi:ss.fff stzh:stzm	22/06/06 17:35:29.123 -5:00
dd-Mmm-yyyy hh12:mi pm stzhstzm	22-Jun-2006 05:35 pm -500

Default Century Behavior

If IBM Cognos Data Manager encounters a year without an explicit century (for example, 56), it assumes that the year is in the 1900s if it is 70 or greater, whereas if it less than 70 it assumes that it is in the 2000s.

Examples

Year value	Interpreted as
00	2000
56	2056
69	2069
70	1970

Year value	Interpreted as
99	1999

Time Formats

IBM Cognos Data Manager uses hh:mi:ss[.fffffffff] as the default time format.

Time formats can contain any of these symbols, interspersed with format text.

Symbol	Description
- / . ; : "text"	Punctuation and text that is reproduced in the result **Note:** You must enclose literal text within double quotation marks.
am/pm	am/pm indicator
fffffffff	Fractions of a second (maximum of 9)
hh	Hour of the day using the 24 hour clock
hh12	Hour of the day using the 12 hour clock
hh24	Hour of the day using the 24 hour clock
mi	Minutes (00 to 59)
ss	Seconds (00 to 59)
sssss	Seconds since midnight (0 to 86399)
stzh	Signed number of hour for the time zone difference (-12 to 13)
tzm	Number of minutes for the time zone difference (00 to 59)

Examples

The following examples refer to a time of 17:35:29.329.

Format	Result
hh:mi:ss	17:35:29
hhmiss.ff	173529.32
hh12:mi pm	05:35 pm

Interval Formats

IBM Cognos Data Manager supports two interval formats, day to second and year to month.

Day to Second Intervals

The default format for day to second intervals is sddddddddd hh:mi:ss[.fffffffff]

Symbol	Description
- / . ; : "text"	Punctuation and text that is reproduced in the result **Note:** You must enclose literal text within double quotation marks.
fffffffff	Fractions of a second (maximum of 9)
hh	Hour of the day using the 24 hour clock
mi	Minutes (00 to 59)
sddddddddd	Signed days in interval (maximum of 9)
ss	Seconds (00 to 59)
sssss	Seconds since midnight (0 to 86399)

Examples

The following examples refer to a day to second interval of 22 days 17 hours 35 minutes and 19.999 seconds.

Format	Result
sddd hh:mi:ss.fff	022 17:35:29.999
sdd hh:mi:ss.f	22 17:35:29.9
sdd hhmiss	22 173529
dd-hhmi	11-1705

Year to Month Intervals

The default format for year to month intervals is syyyyyyyyy-mm

Year to month intervals can contain any of these symbols, interspersed with format text.

Symbol	Description
- / . ; : "text"	Punctuation and text that is reproduced in the result **Note:** You must enclose literal text within double quotation marks.

Symbol	Description
mm	Months in interval
syyyyyyyyy	Signed years in interval (maximum of 9)

Examples

The following examples refer to a year to month interval of 100 years 11 months.

Format	Result
syyy-mm	100-11
syyyy/mm	0100/11

Appendix E. Data Types

To provide consistency, IBM Cognos Data Manager converts the native data types of each DBMS to its own (internal) data types.

This allows it to acquire, transform, and deliver data, irrespective of the source and destination DBMS.

The following table contains the Data Manager internal data types.

Data type	Description
Array	An array of values, which may be of mixed type. The index of the first item is 1. The maximum number of values is limited only by available memory.
BLOB	Binary values which exceed the limit of the BINARY data type may be held as a BLOB (Binary Large Object) data type. BLOB values which are less than DS_LOB_BUF_SIZE in length (default 8000 bytes) are held in memory and can be regarded as BINARY. BLOB values which are larger than DS_LOB_BUF_SIZE are held in temporary files.
Binary (<n>)	A binary value of fixed or variable length <n>, where 0 < <n> <= 8000.
Boolean	Boolean data types have a value of TRUE or FALSE. They are held internally as 0 or 1.
Char(<p>)	A character value of fixed or variable precision <p>, where 0 <= <p> <= 8000. Character data can be in any encoding which is supported by Data Manager.
CLOB	Character values which exceed the limit of the CHAR data type can be held as a CLOB (Character Large Object) data type. CLOB values which are less than DS_LOB_BUF_SIZE in length (default 8000 bytes) are held in memory and can be regarded as CHAR. CLOB values which are larger than DS_LOB_BUF_SIZE are held in temporary files.
	It follows from this that CLOB values in Data Manager can be manipulated in the same way as CHAR values providing they are less than DS_LOB_BUF_SIZE in length. CLOB is a string type so wherever a function takes a parameter of type 'string', a CLOB value may be substituted, but with the proviso above. Attempts to manipulate CLOB values which are more than DS_LOB_BUF_SIZE in length will generally result in a NULL value. Exceptions to this are the built in functions Length, which returns the CLOB length regardless, and Checksum, which calculates a CRC based on the actual data or on the temporary file name.

Data type	Description
	The precision of CLOB values is unknown until the value is fetched and this can lead to differences between CLOB and CHAR processing. CLOB columns are assigned the default maximum precision of 4000 on input and derived data items which return CHAR have a default precision of 255. This means that you may have to manually create tables when delivering derivations of CLOB values, as Data Manager cannot calculate the required precision.
	In SQLTerm, a maximum limit of 4000 characters is imposed for CLOB values. Any values containing more than 4000 characters are truncated.
Date	Can be used to represent either a date only with a default format of yyyy-mm-dd, or a date time with a default format of yyyy-mm-dd hh:mi:ss.fffffffff. For more information on formats, see Appendix D, "Date, Time, and Interval Formats," on page 485.
Float	Double precision, floating point numbers from 2.2250738585072014 x e -308 to 1.7976931348623158 x e308
Integer	Whole numbers, that is, those without a decimal point, both positive and negative, ranging from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Interval day to second	A signed interval value where the interval is expressed as days, hours, minutes, and seconds. Default format sddddddddd hh:mi:ss.fffffffff. For more information, see "Day to Second Intervals" on page 489.
Interval year to month	A signed interval value where the interval is expressed as years and months. Default format syyyyyyyyy-mm. For more information, see "Year to Month Intervals" on page 489.
Number(<p>,<s>)	Precise numbers with a precision <p> of up to 77 significant numbers with a scale <s> factor of up to the number of significant figures. Precision is the total number of digits included in the number. Scale is the number of digits after the decimal point. For information about operations on numbers, see "Rules for Performing Calculations on Numbers" on page 493.
Numeric	A combination of the number, integer, and float data types.
String	A combination of the char and CLOB data types.

Data type	Description
Time	Represents a time value. Default format hh:mi.ss.ffffffff. For more information, see "Time Formats" on page 488.

For descriptions of the conversions between the internal data types that Data Manager uses and the DBMS native data types, see Appendix B, "Database Drivers," on page 465.

Rules for Performing Calculations on Numbers

A number has a left side (digits prior to the decimal point) and right side (digits that follow the decimal point).

The left side (LS) is equal to the precision minus the scale.

The right side (RS) is equal to the scale.

For example, 12.349 is a number with the left side equal to 2, the right side equal to 3, and has a precision and scale of [5,3].

When you perform a calculation using two numbers (A and B), the length of the left and right sides of the resultant number (C) depend on

- the lengths of the left side and right side of the original numbers (A and B)
- the operation you are performing

For example C(LS.RS) = A(LS.RS) *op* B(LS.RS)

Having determined the lengths of the left and right side of a calculated number, the precision and scale can be set.

The rules used to determine the lengths of the left and right side of a calculated number C(LS.RS) are described in the following tables.

Rule One

Operation	RS and LS	Description
Multiplication	No RS for A or B	LS (C) add LS lengths of A and B
	LS length <19 for A and B	Maximum precision of C is 19

Rule Two

Operation	RS and LS	Description
Addition, subtraction		LS (C) use maximum LS length (A or B) +1

Operation	RS and LS	Description
		Maximum precision of C is 19

Rule Three

Operation	RS and LS	Description
Any (except division)	LS length >19 for A or B	LS (C) use maximum LS length (A or B)

Rule Four

Operation	RS and LS	Description
Multiplication	RS length >0 for A or B	RS (C) use RS maximum length (A or B)
	Any LS length for A or B	LS (C) add LS lengths of A and B

Rule Five

Operation	RS and LS	Description
Addition, subtraction		LS (C) use maximum LS length (A or B) +1
		RS (C) use maximum RS length (A or B)

Rule Six

Operation	RS and LS	Description
Division		LS (C) add LS length (A) to RS length (B)
		RS (C) set to 16

Note: Unless stated otherwise, the maximum precision of a calculated Number is 77.

Example Calculations on Numbers

- This example uses rule 4 to determine the precision and scale of the calculated number X:

  123.456 [6,3] *100 [3,0] = X (Px, Sx)

  The LS of X is 3, and the RS is 3 The precision and scale of X is [6,3].
- This example uses rule 6 to determine the precision and scale of the calculated number Y:

123.456 [6,3] /100 [3,0] = Y (Py, Sy)

The LS of Y is 3, and the RS of Y is 16. The precision and scale of Y is therefore [19,16].

Appendix F. Buttons and Icons

This appendix provides a reference to the icons and buttons used in IBM Cognos Data Manager:

- toolbar buttons
- fact build visualization icons
- dimension build visualization icons
- reference structure visualization icons
- JobStream visualization icons
- metadata visualization icons
- source code control icons
- audit history icons

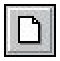
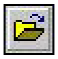
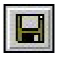
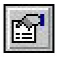
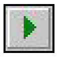
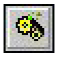
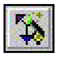
Toolbar Buttons

This section describes the following IBM Cognos Data Manager toolbars:

- Standard toolbar
- JobStream toolbar
- SQLTerm and data source toolbar
- Reference Explorer toolbar

Standard Toolbar Buttons

The following table shows and briefly describes the buttons on the Standard toolbar.

Button	Description
	Opens the **New Catalog** dialog box where you can create a catalog.
	Opens the **Open Catalog** dialog box where you can select the catalog that you want to open.
	Saves all the changes that you have made to the current catalog.
	Opens the **Properties** window for the selected component.
	Executes the current build using the default settings.
	Opens the Fact Build wizard which guides you through the creation of a fact build.
	Opens the Dimension Build wizard which guides you through the creation of a dimension build.

Button	Description
	Opens the Hierarchy wizard which guides you through the creation of a hierarchy.
16	Opens the Date Hierarchy wizard which guides you through the creation of a date hierarchy.
SQL	Opens SQLTerm, allowing you to examine data or construct SQL statements.
	Opens IBM Cognos Data Manager SQLTXT Designer which you can use to configure SQLTXT definition files that are used within IBM Cognos Data Manager to access text files as though they are in an SQL database.
	Opens the reference explorer that you can use to test and browse reference structures.
	Hides or reveals the Visualization pane.
	Enables you to choose the size of the objects shown in the Visualization pane.
	Restores the objects in the Visualization pane to their original size.
	Copies the contents of the Visualization pane to the clipboard, from where you can paste them into another application.
	Opens the navigator that you can use to navigate through the **Tree** pane, search for components, and ascertain component dependencies.
?	Opens the Data Manager online help.

JobStream Toolbar Buttons

The following table shows and briefly describes the buttons on the JobStream toolbar.

Button	Description
	Returns to select mode, from link mode.
	Adds a fact build node to the selected JobStream.
	Adds a dimension build node to the selected JobStream.

Button	Description
✿	Adds a JobStream node to the selected JobStream.
◆	Adds a condition node to the selected JobStream.
{☰}	Adds a procedure node to the selected JobStream.
SQL	Adds an SQL node to the selected JobStream.
▲	Adds an alert node to the selected JobStream.
✉	Adds an email node to the selected JobStream.
⤡	Changes to link mode so that you can link nodes in the JobStream.
⊞	Automatically positions nodes.
▦	Aligns nodes to a hidden grid.

SQLTerm and Data Source Toolbar Buttons

The following table shows and briefly describes the buttons on the SQLTerm toolbar and the Data Source Properties toolbar.

Button	Description
▶	Executes the current SQL query and returns all the selected rows in the database.
▶1	Executes the current SQL query and returns the first row in the database.
▶☰	Executes the current SQL statement and returns the result of the command. Used for the execution of commands that do not return result rows.
■	Interrupts the execution of an SQL query.
☰	Shows the SQL query that has been executed.

Button	Description
XML	Shows the generated XML for the query in the **Query Items** pane.
	Creates a filter in the **Filters** pane.
	Clears one of the following: • the results from the **Test** pane • the SQL query from the **Query** pane • all query items from the **Query Items** pane • all filters from the **Filters** pane

Reference Explorer Toolbar Buttons

The following table shows and briefly describes the buttons on the Reference Explorer toolbar.

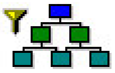
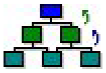
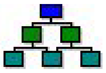
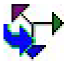
Button	Description
	Shows the elements at each level of the current reference structure.
	Shows the reference structure data as a true hierarchical structure.
$	Opens the **Reference Explorer** dialog box, allowing you to refresh the current reference structure information or select another reference structure.
	Hides or reveals the **Attributes** pane.
	Opens the **Find** dialog box where you can search for members by name.

Fact Build Visualization Icons

The following table shows and briefly describes the icons that may be used to show a fact build in the Visualization pane.

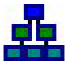
Icons	Description
	The database where IBM Cognos Data Manager acquires the source data, or delivers the transformed data.
	The data source that Data Manager is to use to extract the data from the source database.
	A data source column.

Icons	Description
	A data source derivation.
	A data source literal value.
	A DataStream.
	A DataStream to which a filter has been applied.
	A DataStream item.
	A private DataStream item.
	A DataStream derivation.
	A private DataStream derivation.
	A DataStream pivot value.
	The transformation model.
	An attribute element in the transformation model.
	A derivation element in the transformation model.
	A dimension element in the transformation model.
	A derived dimension element in the transformation model.
	A measure element in the transformation model.
	Allow records with duplicate keys.
	Reject records with duplicate keys.

Icons	Description
	Merge records with duplicate keys.
	Breaking is not performed.
	Breaking is performed.
	A hierarchy where unmatched members are not included.
	A hierarchy where aggregation is to be performed.
	A hierarchy where unmatched members are included, and no aggregation is to be performed.
	A fact delivery.
	A fact delivery where either a level filter or output filter has been applied.
	A dimension delivery.
	The table to which the fact delivery is to deliver the data.
	A primary partitioned table.
	A secondary partitioned table.

Dimension Build Visualization Icons

The following table shows and briefly describes the icons that may be used to show a dimension build in the Visualization pane.

Icons	Description
	The hierarchy to be delivered.
	The auto-level hierarchy to be delivered.

Icons	Description
	A balanced auto-level hierarchy.
	The lookup to be delivered.
	The dimension build.
	The dimension template.
	A table name.
	The database where IBM Cognos Data Manager delivers the transformed data.

Metadata Visualization Icons

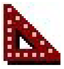
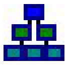
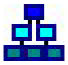
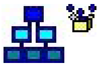
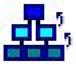
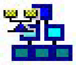
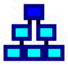
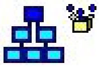
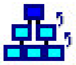
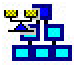
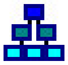
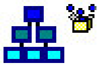
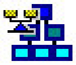
The following table shows and briefly describes the icons that may be used to show the metadata model of the conformed data mart or warehouse. This model can be output to an XML file so that it can be read by IBM Cognos Framework Manager.

Icons	Description
	A dimension within the conformed metadata model.
	A fact table within the conformed metadata model.

Reference Structure Visualization Icons

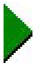
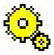
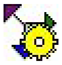
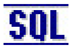
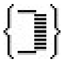
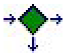
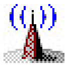
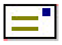
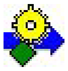
The following table shows and briefly describes the icons that may be used to show a reference structure in the Visualization pane.

Icons	Description
	The database where IBM Cognos Data Manager acquires the source data.
	The data source that Data Manager is to use to extract the data from the source database.
	A DataStream.

Icons	Description
	A DataStream to which a filter has been applied.
	A table.
	The template that Data Manager is to use to extract the data from the source database.
	A hierarchy.
	A hierarchy level.
	A hierarchy level with static members defined.
	A hierarchy level linked to the next available level.
	A balanced hierarchy level.
	A top level of a recursive hierarchy.
	A top level of a recursive hierarchy with static members defined.
	A recursive hierarchy linked to the next available level.
	A recursive hierarchy balanced at the top level.
	A dependent level of a recursive hierarchy.
	A dependent level of a recursive hierarchy with static members defined.
	A recursive hierarchy balanced at a dependent level.

JobStream Visualization Icons

The following table shows and briefly describes the icons that may be used to show a JobStream in the Visualization pane.

Icons	Description
	The starting point for a JobStream.
	A fact build node.
	A dimension build node.
	An SQL node.
	A procedure node.
	A condition node.
	An alert node.
	An email node.
	A JobStream node.

Source Code Control Icons

The following table shows and briefly describes the icons that may be used when a catalog is under source code control.

Icons	Description
	Component checked into source code control.
	Component checked out of source code control by current developer.
	Component checked out of source code control by another developer.

Audit History Icons

The following table shows and briefly describes the status icons that may be used in the audit history for a build or JobStream execution.

Icons	Description
▶	The node listed in the audit history is running.
✓	The node listed in the audit history has successfully completed.
✖	The node listed in the audit history has failed.
■	The execution process was manually stopped, and the process was successful.
!	The execution process was manually stopped, but the process failed prior to being stopped.

Appendix G. Cross Platform Deployment

You can create builds and JobStreams on one operating system for deployment on another.

For example, you can use IBM Cognos Data Manager Designer (on a Microsoft Windows operating system) to develop builds that you deploy to a UNIX operating system. Overall, this is a straightforward process. However, some conditions require additional consideration.

Note: You can execute builds and JobStreams on a remote server using IBM Cognos Data Manager Network Services. For more information, see "Execute a Build or JobStream on a Remote Computer" on page 250.

Use IBM Cognos Data Manager Designer to Deploy on a UNIX operating system

There are two main areas to consider when deploying on a UNIX operating system: file names and database connections.

Very few considerations are necessary before a build that is developed on one platform is deployed on another. This is because little environment-specific information is stored with a build. Where this information does exist however, it should be reviewed.

File Names to Use on UNIX

A file name (for example, for a text delivery or a reject file) specified for a Microsoft Windows operating system may be of the form C:\temp\file.txt.

This is not acceptable on a UNIX operating system platform. The simplest way to resolve this is to represent the file path as an IBM Cognos Data Manager variable, for example

```
{$FILE_PATH}file.txt
```

The variable resolves appropriately on each platform. For information about using variables, see Chapter 24, "Variables," on page 289.

Database Connections

IBM Cognos Data Manager uses a system of connection aliases to describe the connections to databases.

For example, a connection named Sales may resolve to an Oracle connection of username/password@service1. This system simplifies how connections are specified and reused throughout the design process. Unfortunately, while the connection string username/password@service1 may be the correct definition of the connection from the Microsoft Windows operating system computer where Data Manager Designer is running to a UNIX operating system host computer, it will not necessarily be correct if the same build is executed from a different computer.

For example, if the build were executed from a UNIX computer where the database resides, the connection may be username/password. If it were executed from a different UNIX computer, the connection may be username/password@service2.

You must create a database alias definition file that holds the connection details for each environment in which the connection is needed. The database alias definition file replaces the connection definitions defined as part of the Data Manager catalog. When the Data Manager engine is run, it is given the name of the database alias definition file and the environment in which it is running using this syntax

```
DATABUILD -A<dbalias_file> -e<runenv> <other parameters>
```

Create a Database Alias Definition File

Database alias definition files are text files that specify to which databases IBM Cognos Data Manager can connect.

A database alias definition file also specifies the parameters that Data Manager must use when connecting to the data. Each database alias definition file must define all the connections that a build requires.

You can create database alias definition files manually using a standard text editor, by exporting the connection configurations from a catalog, or by combining these two methods.

Create a Database Alias Definition File Manually:

Use a standard text editor to create database alias definition files manually. A database alias definition file consists of one line of text for each alias (or connection definition). The syntax for each line is

```
<runenv> <name> <driver> <driver_items> [<notes>]
```

Symbol	Description
<runenv>	The run environment context for the database alias. By default, this is <DEFAULT>. However, you can create additional run environments for special purposes.
<name>	The alias for the connection.
<driver>	The name of the database driver to use to connect to the data. For example, ORACLE.
<driver_items>	The information required by the database driver to establish the connection. For example, user ID and password.
<notes>	Optional notes to describe the alias. You can only view these notes within the alias definition file.

This is an example of a database alias definition file.

```
//******************
```

```
//* DEFAULT ALIASES *
//******************
<DEFAULT> 'Tutorial Catalog' ODBC 'DSN=DS Tutorial Catalog' ''
<DEFAULT> 'Tutorial DataMart' ODBC 'DSN=DS Tutorial Output' ''
<DEFAULT> 'Tutorial Reference' ODBC 'DSN=DS Tutorial Reference' ''
<DEFAULT> 'Tutorial Sales' ODBC 'DSN=DS Tutorial Sales' ''
<DEFAULT> 'Tutorial Stock' ODBC 'DSN=DS Tutorial Stock' ''
//******************
//* DEVELOPMENT ALIASES *
//******************
```

Run Environments:

Run environments help determine to which database IBM Cognos Data Manager connects when you specify a particular alias.

To establish valid run environment contexts, in the alias definition file, precede each alias definition with the name of the context to which that alias relates.

Within the default run environment (<DEFAULT>), you must provide an alias for each database to which Data Manager must connect. In addition, you may provide additional sets of database aliases if each run environment context and alias name combination is unique. If you do not define an alias for a run environment context, then Data Manager uses the alias from <DEFAULT>.

Create a Database Alias Definition File from a Catalog:

You can create a database alias definition file using CATEXP to export the connections from a catalog.

To do this, in a Microsoft Windows operating system command window, use this syntax

```
CATEXP <catdb_driver_name> <catdb_driver_items> <export_filename> A
[-a]
```

Symbol	Description
<catdb_driver_name>	Specifies the database driver to use to connect to the catalog.
<catdb_driver_items>	Specifies the connection string to pass to the database driver to connect to the catalog. For example, to connect as user DBA, with password PASS, to a catalog that resides in an ODBC data source named Sample, you would specify `"DSN=Sample;UID=DBA;PWD=PASS"`
<export_filename>	Specifies the path and name of the file to which you want to export the database aliases.

Symbol	Description
A	Specifies that you want to export database aliases.
-a	Appends the exported component to the named file.

The aliases in the resulting database alias definition file contain the same information as the connections in the catalog from which you exported them. This means that you cannot export the connections from the catalog that you use for Microsoft Windows operating system and then use the database alias definition file without modification. Therefore, you can either

- Edit the database alias definition file so that the aliases are valid for a UNIX operating system environment. That is, add new entries to the file, modifying the connection information and the environment.
- Create a temporary catalog in which you create the connections for the UNIX environment. Use CATEXP to export these connections to create the required database alias definition file.

Password Encryption:

When you create alias definitions manually, you specify database passwords in plain text. Therefore, anyone who can access the database alias definition files can discover the associated passwords. However, when you create a connection in a catalog, IBM Cognos Data Manager encrypts the passwords. When you create alias definitions by exporting these connections, Data Manager retains the password encryption.

To use password encryption, you must create the connection within a catalog and then export the connection to a database alias definition file.

Note: You can also use CATIMP to import a database alias component that contains an unencrypted password. CATIMP encrypts any unencrypted passwords on import.

Execute a Fact Build under UNIX

If IBM Cognos Data Manager can access the catalog from a UNIX operating system, then you can execute the fact build.

Procedure

Use the DATABUILD command with this syntax

```
DATABUILD<catdb_driver><catdb_items><databuild_nameuuid> -A<dbalias_file> -e<runenv>
```

When you invoke DATABUILD with this syntax, it executes the named fact build that resides in the named catalog.

Symbol	Description
<catdb_driver>	The database driver to use to connect to the catalog.
<catdb_items>	The information required by the database driver to connect to the catalog. For example, to connect as user DBA, with password PASS, to a catalog that resides in an ODBC data source named Sample, specify `"DSN=Sample;UID=DBA;PWD=PASS"`
<databuild_name>	Specifies the name of the fact build to execute.
<uuid>	Specifies the universally unique identifier for the component to execute.
-A	Denotes the text file that contains the database alias definitions to use for the fact build.
<dbalias_file>	Specifies the relative path and name of the alias definition file.
-e	Denotes the run environment for the fact build.
<runenv>	Specifies the run environment to apply to identify the correct database alias.

The following example invokes DATABUILD to execute the fact build named US_Sales. To retrieve the fact build definition, DATABUILD connects to the ODBC data source named SalesCat as user Sales, with password rm123. It obtains the connection specifications from the file alias.txt.

```
DATABUILD ODBC "DSN=SalesCat;UID=Sales;PWD=rm123" US_Sales -Aalias.txt -eunix
```

Appendix H. Problems When Using IBM Cognos Data Manager

This section provides solutions for problems you may encounter when using IBM Cognos Data Manager.

Source Code Control

When using a catalog that is under source code control, if the root of the catalog is checked out, other users are unable to add builds or JobStreams.

To avoid this, individual folders should be used, including one for each developer that is working on the project with new builds and JobStreams. When the builds and JobStreams are complete, move them into a permanent folder.

Writing of Large Objects to DB2 via Bulk Loader

The writing of CLOB and binary data to IBM DB2 is currently unsupported.

32 bit and 64 bit Database Clients

IBM Cognos Data Manager requires 32 bit database clients on UNIX operating system platforms.

Error Message Codes

The format used for error message codes has changed from DS-XXXXX-TNNN to DM-XXX-NNNN. As a result, the codes used in previous versions of IBM Cognos Data Manager are now invalid. If you use these codes for downstream or conditional processing, you must update them.

Note: Error message codes are stored in the audit logs.

Bind Error When Using SQL Server Bulk Copy via API Delivery Module

When using any load mode in the SQL Server Bulk Copy via API delivery module, when you execute a build, you may encounter the following bind error:

```
[S1000][Microsoft][ODBC SQL Server Driver]Not enough
columns bound
```

This occurs when a delivery element does not exist for every column in the target table.

To overcome this issue, you can manually create delivery elements for the required columns. Alternatively, you can use a Microsoft SQL Server (via OLE-DB) connection or change the delivery module to Microsoft SQL Server BCP Command.

Netezza ODBC 3.0 Driver Incompatible With IBM Cognos Data Manager

The Netezza ODBC 3.0 driver is not compatible with IBM Cognos Data Manager. You must install the Netezza ODBC 3.5 driver to save catalogs in Data Manager.

Limited Shared Memory on AIX

There is a default limit of 11 shared memory segments per 32 bit process on AIX®. This may cause issues with shared processes in a JobStream, and you may encounter error messages like this:

```
DM-IPC-0330 Failed to map/attach shared memory XXX
```

```
DM-LIB-0500 OS function 'shmat' produced error 24; The
process file table is full
```

To overcome this issue, you must set an AIX environment variable:

EXTSHM=ON

Note: You cannot set this variable programmatically using putenv().

Additional Permissions Required for DB2 Connection

You must set up user access to the SYSIBM schema when using an IBM DB2 connection in IBM Cognos Data Manager.

TM1 Turbo Integrator delivery via the Data Movement service fails when sharing a local folder

If you are running IBM Cognos Data Manager on your local computer, and you execute a build containing a TM1 Turbo Integrator delivery module via the Data Movement service installed on a server, the build may fail.

This occurs if you specify a local folder in the TM1 Turbo Integrator delivery module properties. The Data Movement service cannot create the required temporary file on your machine because it runs using a local system logon.

To overcome this problem, the IBM Cognos Business Intelligence server logon must be changed to an account that can access your local shared folder.

Alternatively, you can use the direct delivery option in the TM1 Turbo Integrator delivery module properties.

Notices

This information was developed for products and services offered worldwide.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. This document may describe products, services, or features that are not included in the Program or license entitlement that you have purchased.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group
Attention: Licensing
3755 Riverside Dr
Ottawa, ON K1V 1B7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, TM1, and Cognos are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “ Copyright and trademark information ” at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

- Netezza is a registered trademark or trademark of Netezza Corporation, an IBM Company.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.

Glossary

This glossary includes terms and definitions for IBM Cognos Data Manager.

The following cross-references are used in this glossary:

- See refers you from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
- See also refers you to a related or contrasting term.

To view glossaries for other IBM products, go to www.ibm.com/software/globalization/terminology (opens in new window).

A

aggregate rule
A separate aggregate function that can be set up for use in specific dimensions to override the regular aggregate. See also multidimensional aggregation, multilevel aggregation, regular aggregate.

aggregation
The process of reading data across one level in a hierarchy or auto-level hierarchy and summarizing it. See also multidimensional aggregation, multilevel aggregation.

ancestor
A member that exists at a higher level than another member in a hierarchy and is connected by a series of parent-child relationships.

attribute change tracking
A technique for managing historical data in a dimension build. The user can apply attribute change tracking to a business dimension in which one or more non-key values may change without a corresponding change in the key values. For example, this technique enables the business to make a minor change to a product specification without a corresponding change in product code and have the resulting data mart reflect the specification before and after the change.

attribute element
An element in the transformation model that holds non-additive information that is not a dimension or a measure but that may be of interest. For example, descriptive information is typically treated as an attribute.

audit To record information about build or JobStream execution for subsequent analysis.

auto-level hierarchy
A hierarchy that does not have a fixed number of levels. An auto-level hierarchy is structured solely in terms of parent-child relationships. For example, employees report to managers, who may report to other managers. See also recursive level.

B

build A specification for the acquisition, transformation, and delivery of data. See also dimension build, fact build.

bulk loader
A program supplied with a relational database management system that allows fast loading of records into a database.

business dimension
A category of data, such as products or time periods, that an organization might want to analyze. See also multidimensional.

business key
A level attribute or dimension table column that identifies each member of a hierarchy, level, or row of dimension data.

business name
A descriptive name for an object. Assigning a business name enables

analysis software to make reports more comprehensible by using the business name in place of the business key.

C

caption
A description associated with an identifier. The caption is often used in preference to the identifier in reports to make them more comprehensible.

catalog
A repository for storing specifications for builds, reference structures, connections, and other components.

child
In a hierarchy or auto-level hierarchy, a member that has at least one parent.

column token
A level attribute that has certain properties, irrespective of the reference structure level. For example, the level attribute that provides the identifier for each row of data is represented by the $ID column token.

component
A connection, build, reference dimension, reference structure, template, JobStream, user-defined function, metadata dimension, or metadata collection in a catalog.

component package
A utility that allows the import and export of components between catalogs. See also multi-developer support.

conformed dimension
A dimension with a single definition that can be reused or shared across multiple coordinated data marts.

connection
The information required to connect to a database. The actual information required varies according to the DBMS and connection method.

connection alias
A short name used to identify a database connection.

coordinated data mart
A data mart that shares its dimensions with many other data marts.

credential
A set of information that grants a user or process certain access rights.

D

database alias
A short name used to identify the parameters required to connect to a specific database.

database management system driver (DBMS driver)
Software that provides access to data in a specific type of database. Some examples of drivers are Oracle SQL*Net and various ODBC drivers.

database schema
See schema.

data mart
A subset of a data warehouse that contains data that is tailored and optimized for the specific reporting needs of a department or team. A data mart can be a subset of a warehouse for an entire organization, such as data that is contained in online analytical processing (OLAP) tools.

DataStream
An object that gathers together all the data sources from which a build or reference structure acquires data.

DataStream item
A Data Manager object that is required wherever mapping is performed. A DataStream item is used to map a data source column to a transformation model element in a fact build, or a level attribute in a reference structure.

DBMS driver
See database management system driver.

delivery module
A software module used to deliver data to a variety of target systems, such as relational databases. See also dimension delivery module, fact delivery module.

delivery module property
A property that can be used to configure a delivery module. For example, the relational table delivery module has a property that corresponds to the commit interval.

derivation
In a data source or DataStream, a value that is calculated rather than obtained directly from the source data, using an expression defined by the user.

derivation element
An element in the transformation model that is calculated rather than obtained directly from the source data, using an expression defined by the user.

derived dimension element
An element in the transformation model that allows the user to use a calculated expression as the source for a dimension attribute.

dimension
A broad grouping of descriptive data about a major aspect of a business, such as products, dates, or locations. Each dimension includes different levels of members in one or more hierarchies and an optional set of calculated members or special categories.

dimensional framework
The collection of reference dimensions in a catalog. The dimensional framework is used to provide information when merging data and delivering dimension data to a data mart.

dimension attribute
See level attribute.

dimension break
The process of tracking changes in the dimension sequence to improve engine performance and reduce memory usage.

dimension build
A build that delivers data to describe a single business dimension, such as a product or a customer. A dimension build has a source dimension table, a hierarchy for the data, a target database, and a target table. See also build.

dimension delivery
An object in a fact build that delivers data to describe a single business dimension, such as Product or Customer.

dimension delivery module
A delivery module used to deliver data, describing a single dimension, to a target data mart. See also delivery module.

dimension element
An element in the transformation model that is used to give context to a measure element. For example, a Product_Number dimension element, gives context to a Quantity measure element.

dynamic domain
A dimension in which the set of members is constructed from data sources as rows are processed.

dynamic member
A member of a reference structure that is defined by reference database tables.

F

fact build
A build that delivers fact data and dimension data so the user can construct a private data mart from within a single build. See also build.

fact delivery
An object in a fact build that delivers fact data.

fact delivery module
A delivery module used to deliver fact data to a target data mart. See also delivery module.

fact table
A database table, commonly located at the center of a star schema surrounded by dimension tables, that contains measures of a business process, and foreign keys from the dimension tables.

filter The process of restricting the data delivered by a fact build. See also level filter, output filter.

foster parent
In a hierarchy or auto-level hierarchy, an artificially introduced member that acts as a parent for members that either have no defined parent or whose defined parent cannot be found at the next highest level.

H

hierarchy
A particular view of a business dimension. A hierarchy contains the definition of related reference data that is organized into a tree structure of members related as parents and children.

I

ID See identifier.

identifier (ID)
A unique value that is used to represent a business entity within the data. For example, the identifier for a product may be its product code.

input level
In the transformation model, the hierarchy level at which source data is accessed.

J

JobStream
A specification of a process that Data Manager can execute, a JobStream contains a series of steps that can be executed in sequence or in parallel.

JobStream variable
A variable that is declared in the definition of a JobStream. The scope of a JobStream variable is the set of nodes that the JobStream contains. Where a process flow branches in a JobStream, each flow has its own copy of the variable. See also variable scope.

K

key A column or an ordered collection of columns that is identified in the description of a table, index, or referential constraint. The same column can be part of more than one key.

L

late arriving fact processing
The process of dealing with transaction records that arrive in the data warehouse out of sequence. These transactions are linked back to their dimension records that were current at the time of the transaction.

level A set of entities or members that form one section of a hierarchy in a dimension and represent the same type of object. For example, a geographical dimension might contain levels for region, state, and city. See also recursive level.

level attribute
A specific piece of information stored about each member of a level. For example, the color attribute could be used to store the color of a product. Some attributes can have special significance, such as when the attribute contains the business key for the member.

level filter
A filter used to restrict the delivery of data to specific dimensions and hierarchy levels. See also filter, output filter.

link A connection between two nodes in a JobStream.

literal Static data that can be returned by a DataStream. A literal remains constant for every row that the data source returns.

log file
A text file that records the execution of a fact build, dimension build, or JobStream.

logging
The recording of data about specific events on the system, such as errors.

lookup
A simple, single-level reference structure with no parent-child relationships. Members of a lookup are not arranged into hierarchical levels.

M

mapping
The process of defining the relationship between components to enable the flow of data from a data source to fact build or reference structure.

master table
A table that exists to define the range of an entity in a relational database. For example, a product master table would provide information about all the products of a company.

measure
A performance indicator that is quantifiable and used to determine how well a business is operating. For example, measures can be Revenue, Revenue/Employee, and Profit Margin percent.

measure element
An element in the transformation model that represents a business measure.

member
A node in a dimension or reference structure.

merge To consolidate source data in the transformation model using a specified merge method. Data can be merged for measure elements or attributes.

merge method
The type of calculation to perform when merging data.

message
Status information that the engine returns about a build or JobStream execution.

message frequency
The rate at which messages are generated as a build is executed.

metadata collection
A metadata component that groups together all the metadata star models to be included in a metadata export. A metadata collection can contain an entire data warehouse model, or a subset of a model, such as a single subject area.

metadata dimension
A metadata component that contains the description of a conformed dimension to include in a metadata export.

metadata export
A utility that allows the user to export descriptions of the target conformed model in a data mart or data warehouse to an XML file. This file can then be used to produce a model of the target data mart or data warehouse in Framework Manager, without the need to recreate the complete model.

metadata schema
A set of database tables that holds information about the dimensions of a model. In particular, it holds the members of each dimension, and information about how the members relate to each other. See also optimal snowflake schema, optimal star schema, parent-child schema, snowflake schema, star schema.

metadata star
A catalog object containing the description of the fact table to include in a metadata export, with reference to the metadata dimensions that have been set up.

model A system, consisting of fact data and metadata, that represents the aspects of a business.

multi-developer support
A concept that allows developers to share information in a catalog. Multi-developer support can be achieved in and across catalogs using source code control, component packages, and backing up and restoring catalogs. See also component package, source code control.

multidimensional
Pertaining to any system for which the dimension is the fundamental basis of data organization.

multidimensional aggregation
The process of reading data across one level of a hierarchy or auto-level hierarchy, and simultaneously summarizing it across a number of dimensions. See also aggregate rule, aggregation, multilevel aggregation, regular aggregate.

multilevel aggregation
The process of reading data across one level of a hierarchy or auto-level hierarchy, and summarizing it to lower levels. See also aggregate rule, aggregation, multidimensional aggregation, regular aggregate.

mutual exclusion
See semaphore.

N

namespace

1. For authentication and access control, a configured instance of an authentication provider that allows access to user and group information. In Framework Manager, namespaces uniquely identify query items and query subjects. Different databases are imported into separate namespaces to avoid duplicate names.
2. In XML and XQuery, a uniform resource identifier (URI) that provides a unique name to associate with the element, attribute, and type definitions

in an XML schema or with the names of elements, attributes, types, functions, and errors in XQuery expressions.

navigator
A utility that allows the user to navigate catalog components, determine component dependencies, and locate components.

O

operational key
The primary key of a table in an operational data store.

optimal snowflake schema
A snowflake schema, in which each dimension is represented by a table. Each row represents the leaf level members of the dimension and has columns that contain foreign keys to access all of the higher levels of the dimensions. See also metadata schema.

optimal star schema
A variant of the star schema that includes a main dimension table with keys to all the levels of the delivered hierarchy. The main dimension table contains the description of only the leaf level. To save storage space, the descriptions of the higher levels reside in supplementary tables. See also metadata schema.

orphan
A member of a hierarchy or auto-level hierarchy for which no explicit parent exists. Foster parents can be automatically provided for such orphans.

output filter
A filter used to restrict delivery of data to specific data rows. The output filter takes any expression that evaluates to TRUE or FALSE. See also filter, level filter.

output level
In the transformation model, the hierarchy level at which data is output.

P

parent
In a hierarchy or auto-level hierarchy, a member that has one or more child members at the level immediately below.

parent-child schema
A schema that has a single table per dimension with each row in the table representing a member from the dimension, together with its parent. The table may or may not have a column that identifies the level of each member. A variant of this schema uses one table for all dimensions and has an additional column to identify the dimension. See also metadata schema.

parse
To process an SQL statement to return the result set of columns as written in the SQL statement. See also prepare.

partitioning
A technique whereby infrequently used columns can be stored in separate, but linked, tables.

pivot
To treat multiple table columns as though they were a single column with multiple values. The specified table columns rotate through 90 degrees to form rows.

precision
An attribute of a number that describes the total number of binary or decimal digits.

prepare
To send an SQL query to the database, to verify that it is syntactically correct, and return the result set columns. See also parse.

procedure
A sequenced set of statements that may be used at one or more points in one or more computer programs, and that usually has one or more input parameters and yields one or more output parameters.

R

ragged hierarchy
A hierarchy that contains members that have parents at a level higher than the immediate parent level in the hierarchy.

recursive level
A hierarchy structured solely in terms of parent-child relationships, for which the user can explicitly name the levels contained in the recursive relationships. See also auto-level hierarchy, level.

reference dimension
A dimensional framework that holds reference structures and templates used to provide information when merging data and delivering dimension data into data marts.

reference domain
A dimension in which the set of members is constructed from all the members in a hierarchy or lookup, regardless of whether the fact data includes them.

reference explorer
A utility that can be used to explore the data and structure of reference structures.

reference structure
An object that helps define the dimensional framework. Reference structures include both hierarchies and lookups.

regular aggregate
A single aggregate function, used by default to aggregate data across all dimensions of the transformation model. See also aggregate rule, multidimensional aggregation, multilevel aggregation.

reject file
A text file in which records that the engine rejects are stored. In these files, each rejected record has the reason for rejection as its first field.

row limit
The maximum number of source data rows that a DataStream may process.

run environment
An environment where a database definition file collects alias definitions that the user can select, using a variable or command-line parameter.

S

scale The magnitude of a stored number. Storing numbers with precision and scale allows a wide range of numbers to be stored in the same amount of space, but large numbers are not stored as exact integers.

schema
A collection of database objects such as tables, views, indexes, or triggers that define a database. A schema provides a logical classification of database objects.

script A series of commands, combined in a file, that carry out a particular function when the file is run. Scripts are interpreted as they are run.

semaphore
A protected variable, used in a UNIX environment, that allows multiple program threads to share the same resource, such as file access, but not simultaneously.

snowflake schema
A schema that represents a dimension in a series of tables that correspond to the levels of the dimensions. The primary key of each table is the member identifier of each level. Each table has a foreign key to the level above. See also metadata schema.

source code control
A utility that allows the user to share catalogs between developers where concurrent access to the catalogs is required. See also multi-developer support.

SQL Helper
See SQLTerm.

SQL query
Any valid SQL statement in the dialect of the target database. This may be either data definition, such as CREATE TABLE, or data manipulation, such as SELECT.

SQLTerm
A utility that helps the user build SQL statements. It is available wherever SQL statements are required.

SQLTXT
An implementation of SQL over text files.

SQLTXT Designer
A utility that allows the user to configure the SQLTXT definition files to access text files as though they are in an SQL database.

star schema
A simple schema that represents each dimension in a single table. Levels of the dimension are represented as columns within this table. The primary key of this table is the member identifier of the lowest dimension level. See also metadata schema.

static member
A member of a reference structure that is defined as part of the specification, and not by reference to tables in a database.

structure data
Semi-static data that is used to provide context for other data within a relational database and is typically used to construct a reference structure. For example, a product master contains a list of products with which the company deals. The product master may also contain supporting information, such as product characteristics.

subscribed element
A transformation model element that has been selected for delivery to a target system.

substitution variable
A variable with a user-specified value that is used to perform a one-off replacement wherever a substitution is required. Even if the value of the variable changes throughout the process, this is not reflected in the substitution variable. See also variable scope.

surrogate ID
See surrogate identifier.

surrogate identifier (surrogate ID)
An artificial identifier used to replace the natural member identifier of a dimension. Whereas the natural member identifier may be text-based, the surrogate identifier is always an integer. Natural member identifiers may have meaning, such as a product code, however, surrogate identifiers do not.

surrogate key
See surrogate identifier.

T

template
A component that can be used to define reference structure attributes or dimension table columns together with their semantics.

transformation
The process of changing data from one format or structure to another format or structure.

transformation model
A model within a fact build that is used to manipulate the acquired source data. For example, the transformation model can merge data from different sources or aggregate data.

transformation model element
An element of the transformation model in a fact build. A transformation model element can be a dimension, a derived dimension element, a measure, an attribute, or a derivation.

tree pane
A pane in which objects of a catalog are arranged in a tree-like hierarchical fashion.

U

ultimate ancestor
An ancestor member that resides at the top level of a hierarchy.

unbalanced hierarchy
A hierarchy that has leaf nodes at more than one level. The parent of every member comes from the level immediately above.

unmatched member
A dimension member that does not relate to any member in the reference data.

user-defined function
An internal or external function created by the user. Internal functions are defined within a catalog. External functions are defined in an external, compiled module.

V

variable
A representation of a changeable value. See also variable scope.

variable scope
The collection of objects for which a variable is visible. See also JobStream variable, substitution variable, variable.

visualization pane
A pane that provides a graphical visualization of an object in the tree pane.

W

working memory
The amount of memory that the engine is allowed to use during processing.

Index