Version 4 Release 2

IBM i2 Analyze
Information Store Data Ingestion Guide

Note

Before you use this information and the product that it supports, read the information in "Notices" on page 47.

This edition applies to version 4, release 2, modification 0 of IBM® i2® Analyze and to all subsequent releases and modifications until otherwise indicated in new editions. Ensure that you are reading the appropriate document for the version of the product that you are using. To find a specific version of this document, access the Configuring section of the IBM Knowledge Center, and ensure that you select the correct version.

Contents

Information Store data ingestion

About this guide

This documentation provides an overview of the Information Store in IBM i2 Analyze, and the consequences of its structure for loading and manipulating data from external data sources. Later sections describe how to ingest and update data in the Information Store through staging tables that the deployment toolkit can create.

Intended audience

This guide is intended for users who want to add data from external data sources into the Information Store, and to keep that data up to date. Populating the staging tables requires users to be familiar with IBM DB2® database management systems.

Important: Before you ingest data with correlation identifiers, you must install the i2 Analyze 4.2.0.1 Fix Pack or Enterprise Insight Analysis 2.2.0.1 Fix Pack. For more information about downloading and installing the Fix Packs, see Release Material.

Contacting IBM Support

IBM Support provides assistance with product defects, answers FAQs, and helps users to resolve problems with the product.

About this task

After trying to find your answer or solution by using other self-help options such as technotes, you can contact IBM Support. Before contacting IBM Support, your company or organization must have an active IBM software subscription and support contract, and you must be authorized to submit problems to IBM. For information about the types of available support, see the Support portfolio topic in the *Software Support Handbook*.

Procedure

To contact IBM Support about a problem:

1. Define the problem, gather background information, and determine the severity of the problem. For more information, see the Getting IBM Support topic in the *Software Support Handbook*.
2. Gather diagnostic information.
3. Submit the problem to IBM Support in one of the following ways:
    - Online through the IBM Support Portal at Support Portal. You can open, update, and view all of your service requests from the Service Request portlet on the Service Request page.
    - By phone. For the phone number to call in your region, see the Directory of worldwide contacts web page at https://www.ibm.com/planetwide/

Results

If the problem that you submit is for a software defect or for missing or inaccurate documentation, IBM Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM Support website daily, so that other users who experience the same problem can benefit from the same resolution.

The i2 Analyze data model

Before the Information Store can ingest your data, you must prepare it according to the rules of the i2 Analyze data model. i2 Analyze models data in terms of entities, links, and properties (ELP).

- An *entity* represents a real-world object, such as a person or a car.
- A *link* associates two entities with each other. For example, a Person entity might be associated with a Car entity through an Owns link.
- A *property* stores a value that characterizes an entity or a link. For example, a Person entity might have properties that store their given name, surname, date of birth, and hair color.
- Properties can be grouped together in *property groups* to organize your data in ways that help analysis.

The data model for an i2 Analyze deployment states what entities and links can appear in the data. It also states what properties they can have, and defines the relationships that can exist between entities and links of different types.

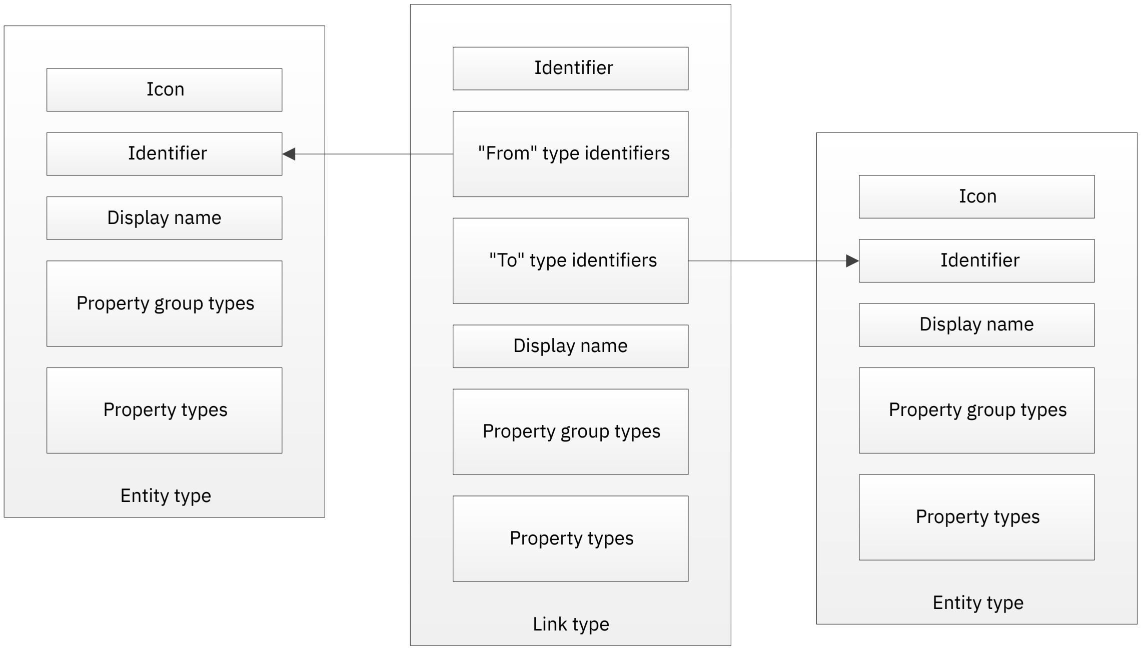
In i2 Analyze, a deployment-specific *schema* defines the data model for that deployment. As a result, the schema determines the structure of the Information Store and the shape that your data must have so that the Information Store can ingest and process it.

Entity types, link types, and property types

The *entity types* and *link types* in the data model for an i2 Analyze deployment determine what entities and links can appear in the data. The *property types* determine what properties the entities and links can have. The i2 Analyze schema defines all these elements of the data model.

Entity types and link types

In an i2 Analyze schema, entity types and link types have similar definitions. Among several common features, entity types and link types both contain identifiers, display names, and the definitions of property types:

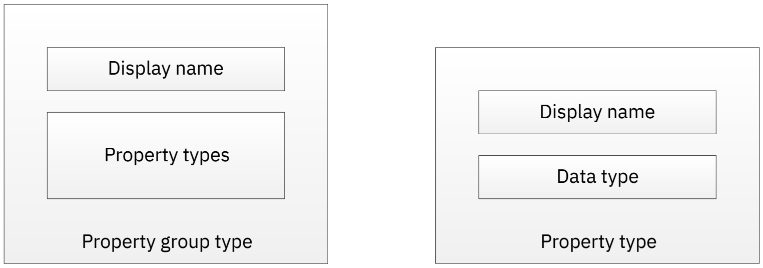


In addition to the common features, each entity type contains the icon that represents entities with that type in visualizations. Link types do not contain icons, but they do contain lists of "from" and "to" entity type identifiers. For a link that has a particular link type, these lists determine what entity types the entities at each of the link can have.

In a valid i2 Analyze schema, the identifiers that appear in the "from" and "to" lists of link types must also appear as the identifiers of entity types.

Property types and property group types

In an i2 Analyze schema, entity types and link types both contain property types. For an entity or a link that has a particular type, the property types specify the names and the logical types of the properties that the entity or link can have.



Note: These representations are simplified. For example, property types can also specify lists of possible property values, and declare whether a property is mandatory for an entity or a link that has the containing type.

In an i2 Analyze schema, property types can appear in an entity type or a link type individually, or as members of a property group type. The purpose of a property group type is to indicate that any properties with the property types that it contains are to be considered as a single unit.

For example, an entity type with the name Person might contain a property group type with the name Body Mark. That property group type might then contain property types with the names Category, Appearance, and Location.

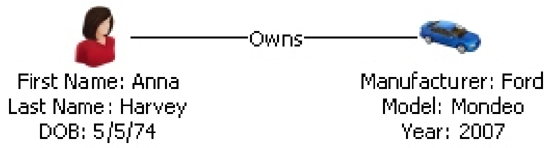
Note: The data stores in an i2 Analyze deployment do not have to provide specific storage for property group types, and the Information Store does not do so. The Information Store implements the data model by treating the property types in a property group type as if the entity type or link type contained them directly.

ELP relationships in i2 Analyze

Depending on the nature of the data that you want the Information Store to ingest, you might need to shape it to fit the typical structure of ELP data. Putting data into ELP format enables many of the analytical tools that i2 Analyze provides.

The simplest ELP relationship involves two entities that are connected with a single link. These kinds of relationships are the building blocks for networks that contain groups and chains of entities with any number of links between them.

In i2 Analyze, a simple relationship that involves entities, links, and properties can be visualized like this example:



Note: Because of the way that these relationships appear in visualizations, the structure is sometimes called a *dumbbell*.

Some of the information that users see in a relationship like this one comes from records in the Information Store:

- For the entity on the left, the data in the Information Store includes the property values "Anna", "Harvey", and "5/5/74".
- Similarly, for the entity on the right, the stored values include "Ford", "Mondeo", and "2007".
- The stored data for the link includes a way of identifying the two entities that it connects.

The remainder of the information in the example comes from definitions in the i2 Analyze schema:

- The default icons for the entities, and the names ("First Name", "Manufacturer") and logical types of their properties, are all defined in the i2 Analyze schema.
- The default label for the link ("Owns") is also defined in the i2 Analyze schema.

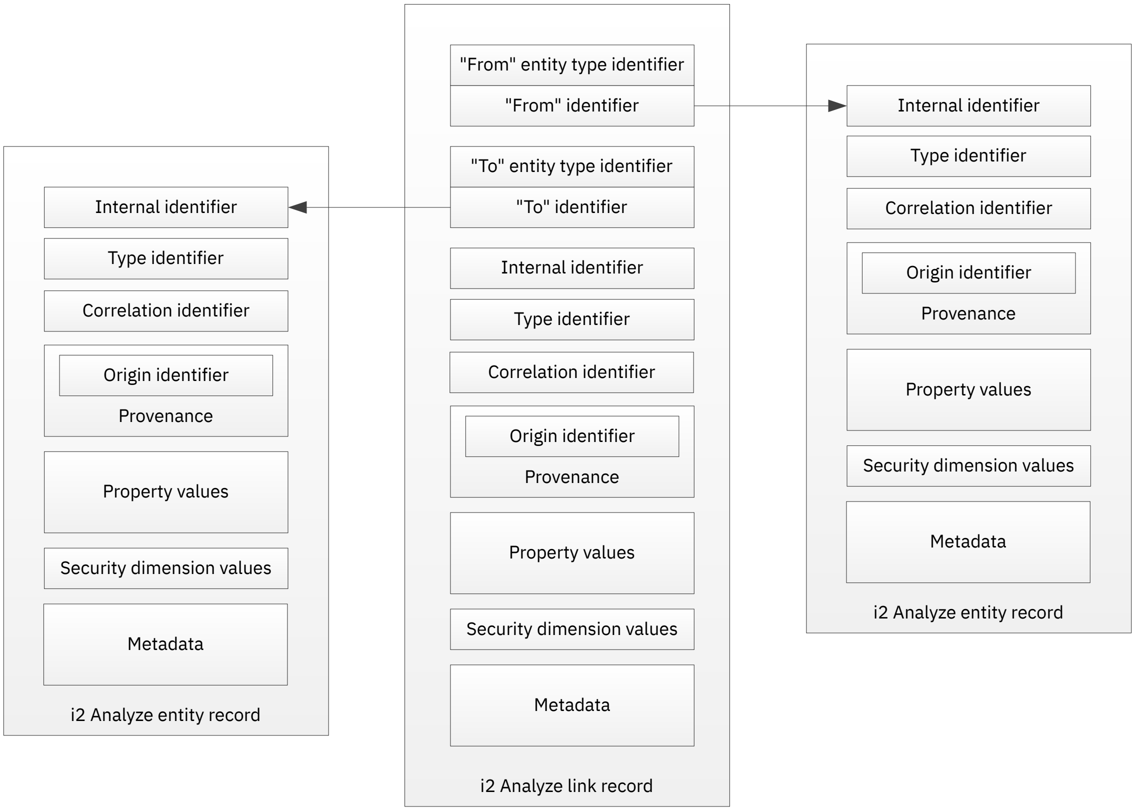
To enable ingestion into the Information Store, you need to resolve your data into the component parts of ELP relationships. If your source is a relational database, for example, it is possible that your tables correspond to entity types, while foreign key relationships form the basis for link types. If your source is a delimited text file, it is possible that rows in the file contain the data for one or more entities and links.

Data in i2 Analyze records

i2 Analyze deployments that store data in and interact with the Information Store use *i2 Analyze records* to realize the data model. i2 Analyze records contain the property data for entities and links, plus the metadata that enhances the analysis that users can carry out.

The schema for an i2 Analyze deployment defines what the i2 Analyze records in that deployment can represent. Every i2 Analyze record has one of the types that the i2 Analyze schema defines. If a record has a link type, then the record represents a link – it is a *link record*. If a record has an entity type, then it is an *entity record*.

This diagram shows how entity and link records compare, and how they are related to each other. It also introduces some other features of the data in i2 Analyze records.



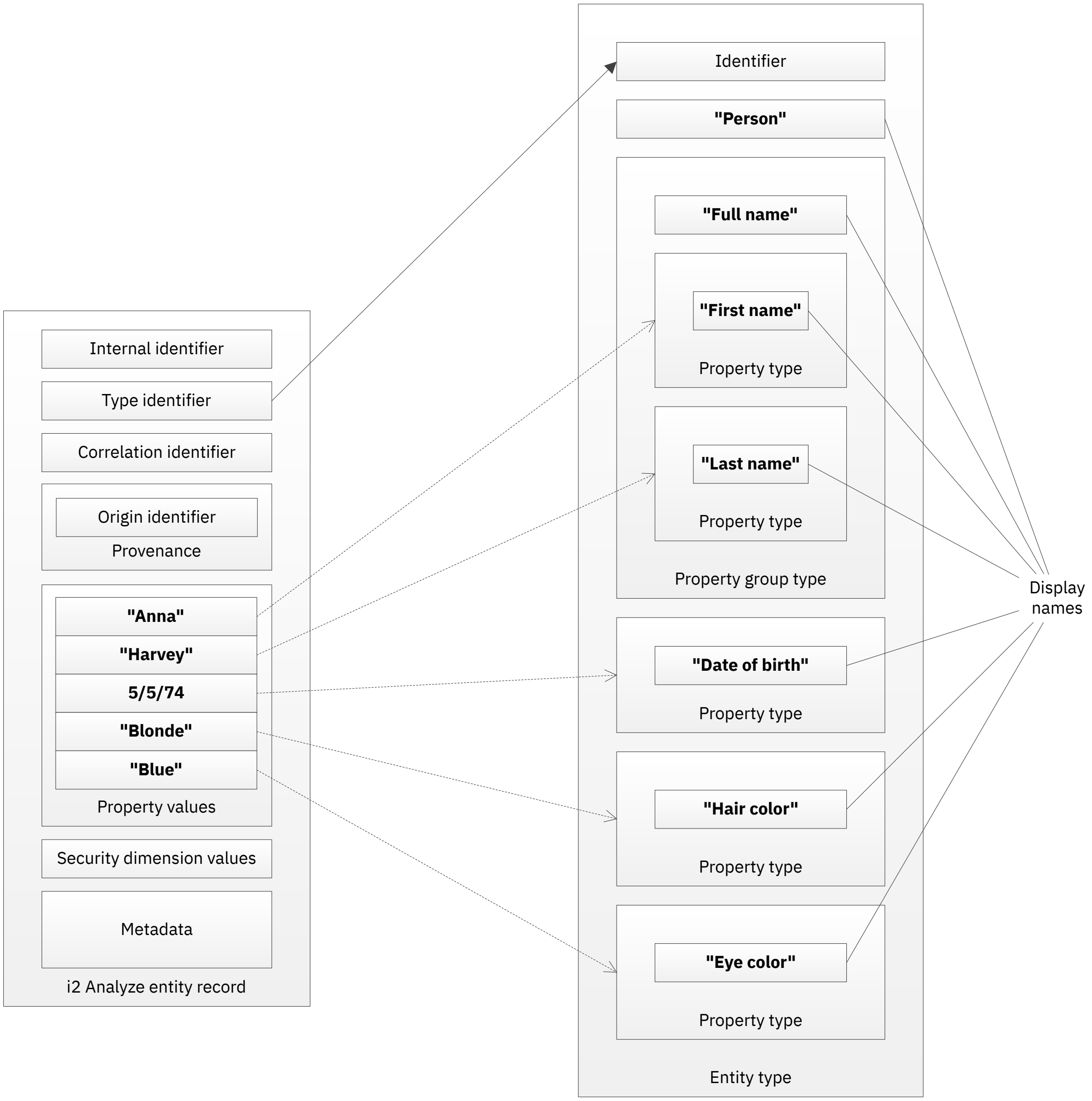
Note: The diagram contains some simplifications:

- *Metadata* refers to information that includes time stamps for each i2 Analyze record. For link records, the metadata also includes strength and direction information.
- *Provenance* is about the source information that contributed to a particular i2 Analyze record. An i2 Analyze record can have more than one piece of provenance, when multiple pieces of source information contributed to the record.

As an example of how to represent a simple entity in this system, consider the following information about a person:

Full Name	Anna Harvey
Date of Birth	5/5/74
Hair Color	Blonde
Eye Color	Blue

The following diagram shows one way to represent this information as an i2 Analyze record:



An i2 Analyze entity record can contain properties that have any of the property types that the entity type defines. However, one record can contain only one property of each defined type.

Note: i2 Analyze records do not take account of property group types in the i2 Analyze schema. The Opal services interpret all property types in the schema as if they are not inside property group types.

The diagram also shows how the property types in the schema only partially determine the contents of an i2 Analyze record. Some of the other contents are due to the security schema, while others still are about identification:

- All i2 Analyze records contain security dimension values, which i2 Analyze uses to determine the access level to the record that a particular user has.
- All i2 Analyze records can contain time stamps that specify when source information for the record was created or edited.
- All i2 Analyze records can contain a correlation identifier that specifies the real-world object the record represents.
- All i2 Analyze records contain one or more pieces of provenance. Each piece has an origin identifier that reproducibly references data for the record in its original source. No two records can have provenance with the same origin identifier, but one record can have provenance from more than one source.
- Additionally, i2 Analyze link records contain an indication of their direction. i2 Analyze considers links to go "from" one entity "to" another. The direction of a link can be with or against that flow, or it can run in both directions or none.

When i2 Analyze records are stored in the Information Store, they contain a few extra pieces of data:

- At creation, the Information Store provides i2 Analyze records with an internal identifier that distinguishes them uniquely within the store. As such, these identifiers are internal to the Information Store.
- In the Information Store, all i2 Analyze link records contain the internal identifiers of the entity records at the ends of the link.
- In the Information Store, each piece of provenance contains the identifier of a source for the record.
- In the Information Store, all i2 Analyze records state when they were first created or uploaded to the store; when they were most recently presented; and when they were last updated.

Your data sources are likely to contain some, but not all, of the data that i2 Analyze records and the Information Store require. To enable the Information Store to ingest your data, you must provide the extra information to the ingestion process.

Identifiers in i2 Analyze records

i2 Analyze records make extensive use of identifiers. Records refer to their type in the i2 Analyze schema, to their original source data, and to other records in ELP relationships. Preparing data for ingestion into the Information Store often involves creating or providing the identifiers that form the basis for the reference mechanisms.

Type identifiers

Every i2 Analyze record contains a type identifier, which is a reference to one of the entity types or link types that the i2 Analyze schema defines. When the Information Store ingests your data, you must provide each row of data with an identifier that matches the identifier of a type definition in the schema.

Every i2 Analyze *link* record contains two further type identifiers, which are references to the entity types of the records at the ends of the link. When the Information Store ingests the data for a link, you must also provide these identifiers.

This strong typing of records in the Information Store is key to the analytical functions that i2 Analyze provides. It allows users to consider not only the existence of relationships between records, but also the nature of those relationships. The schema defines exactly what relationships to allow between record types, and the Information Store enforces those rules during ingestion.

Origin identifiers

The roles of an origin identifier are to reproducibly reference data for a record in its original source, and to identify that data uniquely throughout an i2 Analyze deployment. The precise nature of the identifier depends on the nature of the source, and sometimes on whether the record is a link or an entity.

For example, if the original source is a relational database, then the entity data that you instruct the Information Store to ingest is likely to have ready-made unique identifiers. A table name and a primary key are often enough. Link data can also have ready-made identifiers, but it might not, especially if the relationship that the link represents exists only as a foreign key.

When it ingests data, the Information Store creates origin identifiers by combining a range of information that you specify. For example, you might decide to use the primary key of the record in the external data source, plus the name of that data source. For a link, if no obvious primary key exists in the source, then you must manufacture a reproducible origin identifier by combining other values.

During the ingestion process, the Information Store compares the origin identifier of an incoming row with the origin identifiers of existing data. It can then determine whether the new data creates a record for this data or updates an existing record with this data.

Correlation identifiers

The role of a correlation identifier is to indicate that data is for a specific real-world object. If multiple pieces of data are about the same specific real-world object, they should have the same correlation identifier. At ingestion time, the correlation identifier of incoming data informs the Information Store how to process that data. During ingestion, the Information Store compares the correlation identifiers of incoming data, and the correlation identifiers of existing i2 Analyze records to determine the correlation operation that occurs.

When you ingest data into the Information Store, you can provide a correlation identifier for each row of data in the staging table. You must have access to the correlation identifier when you populate the staging table.

For more information about correlation identifiers, and how to create them, see Correlation identifiers.

Internal identifiers

The Information Store augments every i2 Analyze record with an internal identifier that distinguishes it uniquely within the Information Store. For a link record, the Information Store also adds the internal identifiers of the entity records at the "from" and "to" ends to the link.

The Information Store manages the creation and assignment of internal identifiers automatically. For ingestion operations that the deployment and ETL toolkits enable, you do not have to interact directly with the internal record identifiers. However, they can be important in deletion-by-rule operations that target records based on their relationships with others.

Security of i2 Analyze records

i2 Analyze records are subject to the i2 Analyze security rules. The security schema defines the security model for your i2 Analyze deployment, and every i2 Analyze record must have at least one value from each security dimension in the schema.

When a user runs a query against the Information Store, i2 Analyze looks up which groups the user belongs to, and determines their security permissions. Then, it compares their security permissions to the security dimension values of the records in the query results. In this way, i2 Analyze calculates which records the user has access to.

You must add security information to data during ingestion to the Information Store. Each time the ingestion process runs, you can specify which security dimension values the ingested data receives:

- If you decide that all the data from a given external source must have the same security settings, you can specify the same dimension values for all types.
- Alternatively, you can dictate that all the data of a particular entity type or link type receives the same security dimension values.
- You can also configure the process so that each row of data receives security dimension values that you specify or determine programmatically.

Note: In this version of IBM i2 Analyze, you can set the security dimension values for a row of data during the ingestion process, but at no other time. To change the dimension values, you must arrange for the Information Store to ingest the data again.

Properties in i2 Analyze records

In an ingestion process, providing i2 Analyze records with property values typically involves describing which values in the source data to assign to which types of properties. However, date-and-time values require special handling, and you might need to decide how to handle multiple values for the same property type.

Dates and times

To store date-and-time values accurately, and to enable wide-ranging analysis, i2 Analyze records require you to provide information about time zones and Daylight Saving Time. If your data source contains date-and-time information, then you must manipulate it into the right form before the Information Store can ingest it.

Property groups

The Information Store provides no special representation for property groups, and it does not support storing more than one property with the same property type in the same record. The Information Store treats property types that appear inside property group types in the i2 Analyze schema as if the property group type does not exist.

Information Store staging tables

Arranging for the Information Store to ingest your data requires no understanding of the data tables in the Information Store database. To make ingestion easier, you can instruct i2 Analyze to create a set of staging tables that provide a simplified representation of the underlying structure.

When you supply data for the Information Store to ingest, it must match the structure of the staging tables. If you can load your data into the staging tables successfully, the Information Store can ingest your data. This two-part approach (loading, then ingesting) provides an opportunity for the system to validate data before ingestion takes place.

At your request, i2 Analyze generates an Information Store staging table that can contain data for i2 Analyze records of a single entity type or link type. To generate the staging table, it uses information from the i2 Analyze schema, which is the same starting point from which it generates the main data tables during deployment.

An entity type staging table contains:

- At least one column for each property type in the schema.
- A column for the source identifier of the data. During ingestion, i2 Analyze can use the source identifier to construct an origin identifier for the ingested data.
- Two columns to record when the data was created and updated in the source.
- Two columns to record the *correlation identifier type* and *correlation identifier key* of the data. During ingestion, i2 Analyze uses the information in these columns to construct the correlation identifier for the ingested data.
- A column for each security dimension that the security schema defines. During ingestion, i2 Analyze can use the information in these columns to implement per-record security.

For example, if the i2 Analyze schema contains this simplified entity type definition:

```
<EntityType Id="ET5" DisplayName="Person">
    <PropertyTypes>
        <PropertyType DisplayName="First (Given) Name"
                      LogicalType="SINGLE_LINE_STRING" Id="PER4"/>
        <PropertyType DisplayName="Middle Name"
                      LogicalType="SINGLE_LINE_STRING" Id="PER5"/>
        <PropertyType DisplayName="Family Name"
                      LogicalType="SINGLE_LINE_STRING" Id="PER6"/>
        <PropertyType DisplayName="Date of Birth"
                      LogicalType="DATE" Id="PER9"/>
        <PropertyType DisplayName="Date and Time of Death"
                      LogicalType="DATE_AND_TIME" Id="PER10"/>
    </PropertyTypes>
</EntityType>
```

This SQL statement is then the definition of a corresponding staging table:

```
CREATE TABLE "IS_STAGING"."E_PERSON" (
        "SOURCE_CREATED" TIMESTAMP,
        "SOURCE_LAST_UPDATED" TIMESTAMP,
        "SOURCE_ID" VARCHAR(50),
        "CORRELATION_ID_TYPE" VARCHAR(20)
        "CORRELATION_ID_KEY" VARCHAR(200)
        "SECURITY_LEVEL" VARCHAR(50),
        "SECURITY_COMPARTMENT" VARCHAR(50),
        "P_FIRST_GIVEN_NAME" VARCHAR(250),
        "P_MIDDLE_NAME" VARCHAR(250),
        "P_FAMILY_NAME" VARCHAR(250),
        "P_DATE_OF_BIRTH" DATE,
        "P0_DATE_AND_TIME_OF_DEAT" TIMESTAMP,
        "P1_DATE_AND_TIME_OF_DEAT" VARCHAR(250),
        "P2_DATE_AND_TIME_OF_DEAT" SMALLINT,
        "P3_DATE_AND_TIME_OF_DEAT" TIMESTAMP
    );
```

Note: Staging tables for link types additionally contain a column for the direction of the link, and two further columns for the source identifiers of the link ends.

The statement creates the staging table in a separate DB2 schema from the Information Store data tables. Many of the columns in the staging table have names that are derived from the display names of the property types in the i2 Analyze schema. In most cases, the relationship between the schema and the staging table is obvious, but some exceptions exist:

- The first two columns of the staging table are SOURCE_CREATED and SOURCE_LAST_UPDATED. You can use these columns to store information about when the data to be ingested was created and modified in its source.
- The third column of the staging table is named SOURCE_ID. When you populate the staging table, you can use this column to store values that reference the rest of the data in its original source.

  Note: If this staging table definition was for a link type, it would also contain FROM_SOURCE_ID and TO_SOURCE_ID columns, and a DIRECTION column.
- The next two columns of the staging table are CORRELATION_ID_TYPE and CORRELATION_ID_KEY. If you want to correlate data that is ingested into the Information Store, you can use these columns to store the values that are used to create the correlation identifier of the data.
- The next columns derive from the security schema rather than the i2 Analyze schema. One column exists for each security dimension that the security schema defines. You can use these columns if you want to give different dimension values to each i2 Analyze record that is created or updated as a result of ingestion.
- Finally, any property type in the i2 Analyze schema that has the logical type DATE_AND_TIME occupies four columns in the staging table. These columns always appear in the same order:
  - The "P0" column is for the local date and time as originally recorded, as a TIMESTAMP.
  - The "P1" column is for the time zone of the local date and time, as listed in the IANA database.
  - The "P2" column is for an indicator of whether Daylight Saving Time is (1) or is not (0) in effect.

    Note: i2 Analyze considers this value only when the time is ambiguous because it occurs during the hour that is "repeated" when Daylight Saving Time ends.

- The "P3" column is for the date and time as expressed in Coordinated Universal Time (UTC), as another `TIMESTAMP`.

The staging tables contain some, but never all, of the data for i2 Analyze records. They do not contain the type identifiers that Information Store records must have, and it is not mandatory to populate the columns for time stamps and security dimension values. When a staging table does not have all the data that the Information Store requires, you must supply the remainder in an ingestion mapping.

In the staging tables, it is not mandatory to populate the columns for the correlation identifier. However, if you do provide them you are effectively turning on correlation for that item type.

Ingestion mapping files

In the data ingestion process, ingestion mappings describe how to create records in the Information Store from data that you load into the staging tables. The mappings are defined in one or more ingestion mapping files, and each mapping deals with records of one particular entity or link type.

An ingestion mapping file is an XML document whose structure is validated during the ingestion process. Every time that you instruct the Information Store to ingest data, you specify both the mapping file to use, and the ingestion mapping within it. You can choose to put all your ingestion mappings in one file, or to spread them across several files.

Ingestion mappings have two complementary purposes. First, they make the association between an entity type or a link type in the i2 Analyze schema and a staging table in the DB2 database. Second, they provide any extra information that the Information Store requires but the staging tables do not contain.

For all record types, the extra information that an ingestion mapping can provide includes:

- The type identifier of the entity or link type that the mapping applies to
- The name of the data source that the data to be ingested comes from
- How to create an origin identifier for data of this type
- The security dimension values that all records of this type receive, if you do not use per-record security

Link type ingestion mappings provide further information that addresses the requirements of link records:

- The Information Store must be able to test that it already contains the entity records at the ends of an incoming link . The link type mapping must describe how to create the origin identifiers associated with those entity records so that the Information Store can look them up.
- To make the look-up more efficient, the link type mapping also contains the type identifiers of the entity records that appear at the "from" and "to" ends of the incoming links. A link type that can connect entities of several different types requires a separate mapping for each valid combination of end types.

Ingestion mapping syntax

The root element of an ingestion mapping file is an `<ingestionMappings>` element from the defined namespace. For example:

```
<ns2:ingestionMappings
    xmlns:ns2="http://www.i2group.com/Schemas/2016-08-12/IngestionMappings"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...
</ns2:ingestionMappings>
```

Within the ingestion mapping file, you use an `<ingestionMapping>` element to define a mapping for a particular entity type or link type. Each `<ingestionMapping>` element has a mandatory `id` attribute that must be unique within the mapping file. You use the value to identify the mapping when you start ingestion. For example:

```
<ingestionMapping id="Person">
    ...
</ingestionMapping>
```

Note: For examples of complete ingestion mapping files, search for files with the name `mapping.xml` in the i2 Analyze deployment toolkit. All of those files contain definitions that are similar to the definitions here.

Entity type ingestion mappings

When the mapping is for an entity type, the `<ingestionMapping>` element has the following children:

`stagingArea`

The `<stagingArea>` element specifies where the mapping gets its staged data from. In this version of i2 Analyze, the staged data is always in a staging table, and `<stagingArea>` always has a `<tableName>` child.

`tableName`

The value of `<tableName>` is the name of the staging table that contains the data to be ingested.

For example:

```
...
<stagingArea xsi:type="ns2:databaseIngestionSource">
    <tableName>IS_STAGING.E_PERSON</tableName>
</stagingArea>
...
```

`itemTypeId`

The value of the `<itemTypeId>` element is the identifier of the entity type (or the link type) to which the mapping applies, as defined in the i2 Analyze schema.

For example:

```
...
<itemTypeId>ET5</itemTypeId>
...
```

originId

The `<originId>` element contains a template for creating the origin identifier of each ingested row. `<originId>` has two mandatory child elements: `<type>` and `<keys>`.

type

The "type" of an origin identifier allows the services in an i2 Analyze deployment to determine quickly whether they are interested in (or how to process) a particular row of data. The value of the `<type>` element does not have to be meaningful, but data from different sources generally have different values.

keys

The "keys" of an origin identifier contain the information necessary to reference the data in its original data source. The `<keys>` element has `<key>` children. The number of child elements you need depends on how difficult it is to guarantee uniqueness.

For data that originates in relational sources, you might use keys whose values include the source name, the table name, and the unique identifier of the data within that table.

For example:

```
...
<originId>
    <type>OI.EXAMPLE</type>
    <keys>
        <key>EXAMPLE</key>
        <key>PERSON</key>
        <key>$(SOURCE_ID)</key>
    </keys>
</originId>
...
```

Here, `$(SOURCE_ID)` is a reference to the column named `SOURCE_ID` in the staging table to which this ingestion mapping applies. When the Information Store ingests the data, the value from the staging table becomes part of the origin identifier in the Information Store.

An alternative approach to the same goal is to populate the `SOURCE_ID` column with a value that is already a composite of the key values. This technique simplifies the ingestion mapping at the expense of more work when you populate the staging table.

Note:

The length of the value inside the `<type>` element must not exceed 100 bytes. This is equivalent to 100 ASCII characters.

The combined length of the values inside the `<keys>` element must not exceed 1000 bytes. This is equivalent to 100 ASCII characters.

`dataSourceName`

The value of the `<dataSourceName>` element identifies the data source from which the data in the staging table came. It must match the name of an ingestion source that you provide to the Information Store during the ingestion process.

For example:

```
...
<dataSourceName>EXAMPLE</dataSourceName>
...
```

`createdSource` and `lastUpdatedSource`

By default, the ingestion process automatically puts the values from the `SOURCE_CREATED` and `SOURCE_LAST_UPDATED` columns of the staging tables into the Information Store. If you want to use the same values for all ingested data, you can override that behavior by including the non-mandatory `<createdSource>` and `<lastUpdatedSource>` elements and specifying values in DB2 date-time string format.

For example:

```
...
<createdSource>2002-10-04 09:21:33</createdSource>
<lastUpdatedSource>2002-10-05 09:34:45</lastUpdatedSource>
...
```

`securityDimensionValues`

Every row that the Information Store ingests must have a security dimension value from each dimension in the security schema. The Information Store staging tables contain a column for each access security dimension that the security schema defines.

In your ingestion process, you can use the staging table columns to store dimension values on a per-row basis. Alternatively, you can specify that all the data that the Information Store ingests through the same mapping get the same security dimension values.

In the ingestion mapping file, the `<securityDimensionValues>` element has `<securityDimensionValue>` children. For per-row security, use the value of each `<securityDimensionValue>` element to reference a security dimension column.

For example:

```
...
<securityDimensionValues>
  <securityDimensionValue>$(SECURITY_LEVEL)</securityDimensionValue>
  <securityDimensionValue>$(SECURITY_COMPARTMENT)</securityDimensionValue>
</securityDimensionValues>
...
```

In the staging table, the referenced columns can contain either a single dimension value, or a comma-separated list of dimension values.

For per-mapping security, set the value of each `<securityDimensionValue>` element to a security dimension value.

For example:

```
...
<securityDimensionValues>
    <securityDimensionValue>HI</securityDimensionValue>
    <securityDimensionValue>UC</securityDimensionValue>
    <securityDimensionValue>OSI</securityDimensionValue>
</securityDimensionValues>
...
```

In either approach, the values that you specify must be present in the i2 Analyze security schema.

Link type ingestion mappings

When the ingestion mapping is for a link type, the `<ingestionMapping>` element has the same children that entity types require, plus the following ones:

`fromItemTypeId`

The value of the `<fromItemTypeId>` element is the type identifier of entity records that the schema permits at the "from" end of the link type to which this mapping applies.

For example:

```
...
<fromEntityTypeId>ET5</fromEntityTypeId>
...
```

`fromOriginId`

The `<fromOriginId>` element contains a template for creating the origin identifier of the entity record at the "from" end of each ingested link row. Its syntax is identical to the `<originId>` element.

The origin identifiers that result from `<fromOriginId>` must match the origin identifiers that result from the `<originId>` element for the entity type in question. The ingestion process uses this information to verify that the Information Store already ingested an entity record that has this origin identifier.

For example:

```
...
<fromOriginId>
    <type>OI.EXAMPLE</type>
    <keys>
        <key>EXAMPLE</key>
        <key>PERSON</key>
        <key>$(FROM_SOURCE_ID)</key>
    </keys>
</fromOriginId>
...
```

`toItemTypeId`

The value of the `<toItemTypeId>` element is the type identifier of entity records that the schema permits at the "to" end of the link type to which this mapping applies.

For example:

```
...
<toEntityTypeId>ET10</toEntityTypeId>
...
```

toOriginId

The <toOriginId> element behaves identically to the <fromOriginId> element, except for the obvious difference that it applies to the entity record at the "to" end of each ingested link row.

For example:

```
...
<toOriginId>
    <type>OI.EXAMPLE</type>
    <keys>
        <key>EXAMPLE</key>
        <key>ACCOUNT</key>
        <key>$(TO_SOURCE_ID)</key>
    </keys>
</toOriginId>
...
```

linkDirection

The <linkDirection> element is a non-mandatory child of the <ingestionMapping> element. When you include a <linkDirection> element in an ingestion mapping, you can either provide the same value for all links, or refer to the DIRECTION column of the staging table. Legal values for the element or the column are WITH, AGAINST, BOTH, and NONE.

For example, to use a fixed value:

```
...
<linkDirection>WITH</linkDirection>
...
```

Or, to use the value in the DIRECTION column:

```
...
<linkDirection>$(DIRECTION)</linkDirection>
...
```

If an ingestion mapping for a link type does not contain a <linkDirection> element, then any links that the Information Store ingests through the mapping have no direction.

References to columns and properties

In an ingestion mapping, you can use constants or references to specify values for i2 Analyze records. When you use a reference, the ingestion process retrieves a value from a staging table column or a property in a settings file. The settings file can also set system properties that control some aspects of ingestion into the Information Store.

Many of the pieces of information that you provide in an ingestion mapping are fixed for that mapping. Item types, end types, and some parts of the origin identifier do not change between the i2 Analyze records that one mapping is responsible for. The most appropriate way to specify this kind of information is to use constant values on a per-mapping basis.

The two main reasons for preferring references to constant values lie at opposite ends of the spectrum:

- To give different values for the same piece of data to records that are ingested through the same mapping, you can refer to a staging table column. This approach is appropriate for many non-property values that change from one record to the next.
- To use the same values across multiple ingestion mappings, refer to a property in a settings file. This approach might be appropriate when you want all the data from a source to get the same security dimension values. You can refer to the same property from every mapping that you write.

  A settings file that defines properties for the ingestion process is just a text file that contains a set of *name=value* pairs, with one pair on each line:

  ```
  SEC_LEVEL_VALUE=UC
  SEC_COMPARTMENT_VALUE=HI,OSI
  ```

  When you run one of the ingestion commands, you can supply it with the name of the properties file whose values you want to use.

To use a value by reference in an ingestion mapping, you use the `$(`*name*`)` syntax. *name* is the name of either a column in the staging table or a property in a settings file. For example, `$(SOURCE_ID)` and `$(DIRECTION)` refer to staging table columns, while in the previous example `$(SEC_LEVEL_VALUE)` and `$(SEC_COMPARTMENT_VALUE)` refer to properties.

Note: Since referring to columns and properties uses the same syntax, a clash can happen if a column and a property have the same name. In that case, the value of the property takes precedence.

System properties

As well as providing values for ingestion mappings, you can use the settings file to configure the behavior of the ingestion process. The file supports a handful of system properties that you can set in the same way as you create and set custom properties.

`IngestionFailureMode`

When the Information Store encounters a problem with a record during ingestion, its default behavior is to log the error and move on to the next record. Failure is *record-based*. Instead, you can specify that a problem with one record causes the Information Store not to ingest any of the records from that staging table. Failure then is *mapping-based*.

To configure mapping-based failure, create a settings file that contains the line `IngestionFailureMode=MAPPING`, and specify the file when you run the ingestion command. The default setting for `IngestionFailureMode` is `RECORD`.

`IngestionTempDir`

During the ingestion process, i2 Analyze creates some temporary files on the local file system. In some circumstances, these files can be large. The default location for the files is the Java temporary directory, but you can change it by providing a different path to the `IngestionTempDir` property in the settings file.

Preparing for ingestion

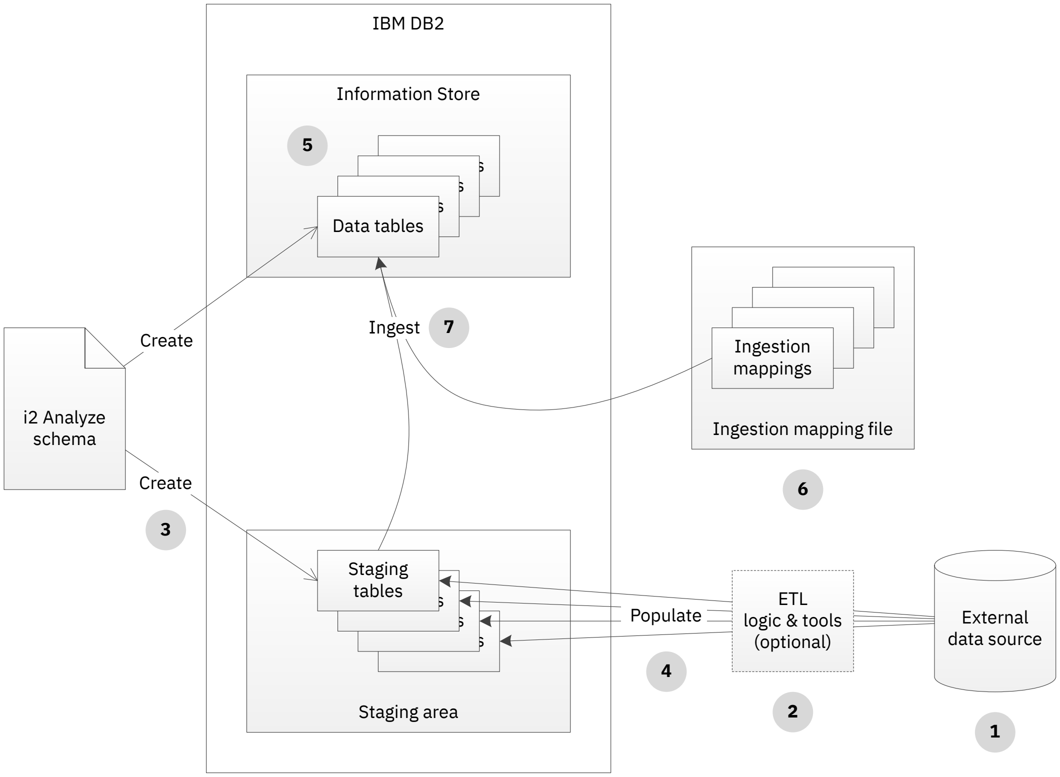
You must complete three tasks before the Information Store can ingest data from an external source. You must identify exactly which data to load, transform the data to align with the active i2 Analyze schema, and augment the data with extra information that the Information Store requires.

About this task

The only way to add and edit large volumes of data in the i2 Analyze Information Store is to enable and then instruct the Information Store to ingest it. The enablement process involves creating and populating staging tables for the data, and then supplying the metadata that is crucial to the analytical capabilities of i2 Analyze.

Procedure

You can plan and execute the Information Store data ingestion process in a series of discrete steps. This diagram illustrates the approach.



1. Decide which entity types and link types in the active i2 Analyze schema best represent the data that you want the Information Store to ingest.
2. Understand the impact of your deployment architecture on the ingestion process in general, and on any logic or tools for transformation in particular.
3. Create staging tables in the DB2 database for the types that you identified. Create more than one staging table for some link types.

4. Use external tools, or any other appropriate technique, to transform your data and load the staging tables with the data for ingestion.
5. Add information about your data source to the list of ingestion sources that the Information Store maintains.
6. Write the ingestion mappings that govern the ingestion process and provide additional information that the Information Store requires.
7. Run the ingestion command separately for each of the ingestion mappings that you wrote.

Example

The `examples\data\law-enforcement-data-set-1` and `\signal-intelligence-data-set-1` directories in the deployment toolkit contain files that i2 Analyze uses when you run the `setup -t ingestExampleData` command to populate the Information Store during deployment. These files provide demonstrations of many of the steps in the standard approach to ingestion. The following topics describe those steps in more depth as they detail the Information Store ingestion process.

Identifying the data to be ingested

The detail of how you arrange for the Information Store to ingest your data varies according to how that data is stored in its source. However, the start of the process is always to consider what data you have, and work out how you can shape it to fit the i2 Analyze schema.

About this task

Usually, when you start thinking about adding data from an external source into the Information Store, there is an i2 Analyze deployment already in place. That deployment necessarily has an i2 Analyze schema that defines all of the entity types, link types, and property types that data in the system can have. Before you go any further, you must have a clear idea of how your data becomes i2 Analyze entity records and link records in the Information Store.

It is unlikely that the data in your external source has a one-to-one mapping with the entity types and link types in the i2 Analyze schema:

- Probably, your source does not contain data for all the entity types in the schema. As a result, you do not usually need to create a staging table for every possible entity type.
- The schema can define link types that connect several different entity types. In that case, each entity-link-entity type combination for which your source contains data requires a separate staging table.

For example, imagine an i2 Analyze schema that defines the entity types "Person", "Vehicle", and "Account", and the link type "Access to". In this situation, you might decide to create a staging table for each of the entity types. However, the data requires two staging tables for "Access to" links: one for links between people and vehicles, and the other for links between people and accounts.

Procedure

1. Open the schema for the i2 Analyze deployment in Schema Designer.
2. Go through the list of entity types, and determine which of them represent data in your source.
3. Make a note of the identifier of each entity type that represents your data.
4. Repeat steps 2 and 3 for the list of link types. Check the Link Ends tab, and make a note of all the combinations for which your source contains data.

Results

When you complete the steps above, you have a list of all the i2 Analyze schema types that your data contains. You also have a list of all the staging tables that you need to create. Before you create those tables, the next part of the process is to understand the impact on ingestion of the i2 Analyze deployment architecture.

Understanding the architecture

The physical architecture of your i2 Analyze deployment both affects and is affected by how you acquire and transform external data for the Information Store. Depending on the architecture, you might need to perform more deployment tasks before you can run data ingestion commands.

About this task

A complete solution for loading and ingesting data into the Information Store has four mandatory architectural components:

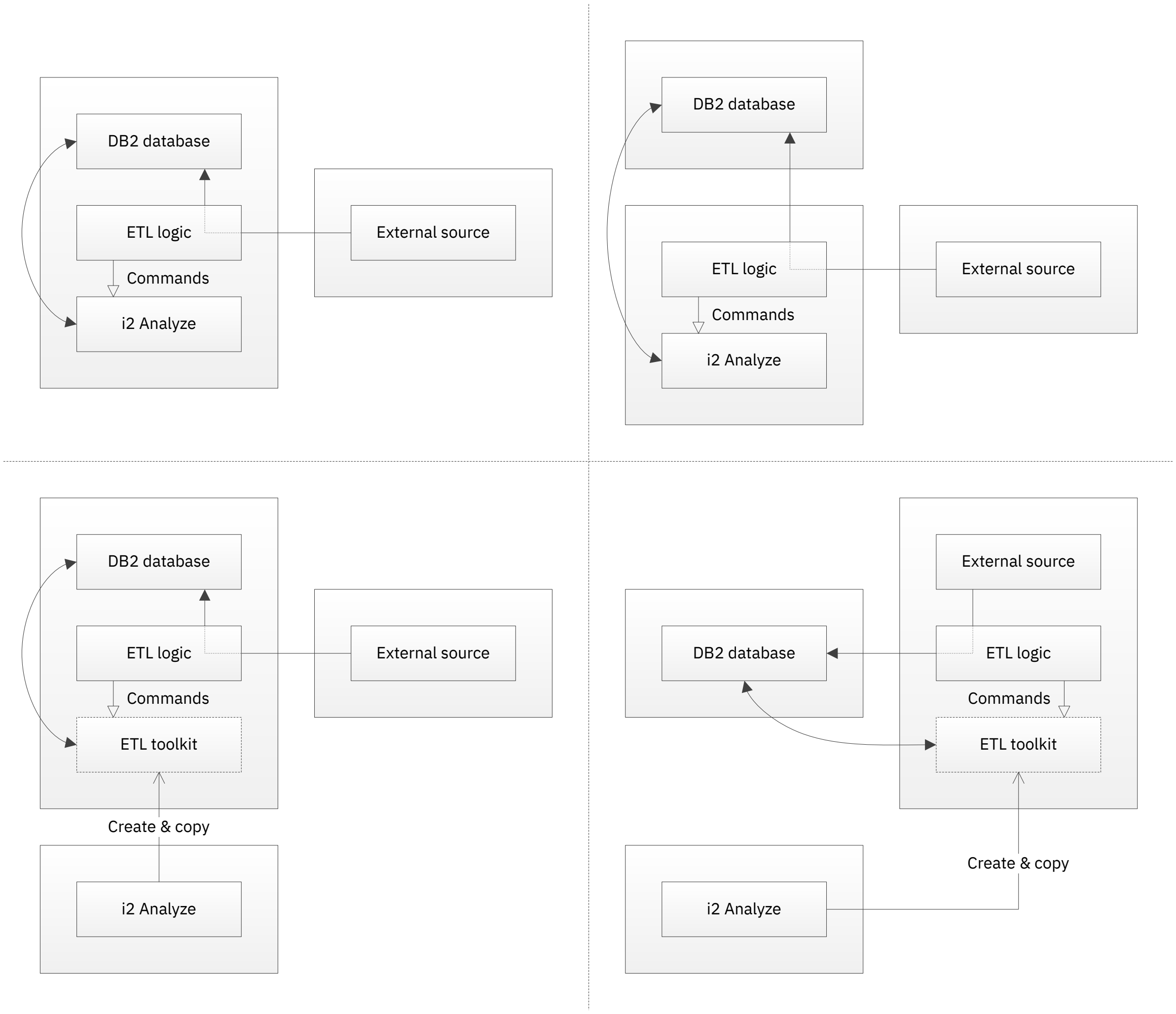
- The i2 Analyze server, and in particular the deployment toolkit that it contains
- The DB2 database that contains the Information Store and the staging tables
- An external data source
- The ETL logic that transforms the source data and loads it into the staging tables

In addition to the mandatory components, you can optionally include a component to prepare your data for correlation. To prepare your data for correlation, you might choose to use a matching engine or context computing platform. Matching engine and context computing platforms can support the identification of matches that enable you to identify when data that is stored in multiple sources represents a single entity. This preparation of your data might occur as part of your ETL logic, or between an external data source and the ETL logic.

i2 Analyze supports physical architectures in which the database is hosted on the same server as the application, or on a different one. You can also choose to locate your ETL logic on the same server as the i2 Analyze application, or on the same server as the database, or on an entirely separate server.

The process of transforming source data can be demanding, especially if the requirements are complex or the volume is high. There are also scenarios in which you might want to automate the process of loading and then ingesting the external data. Ultimately, the architecture that you decide upon depends on the needs and constraints of your deployment.

The following diagram shows some of the permutations. The examples in the upper-left and upper-right quadrants represent deployments in which the ETL logic (implemented by a tool like IBM DataStage, for example) is co-hosted with the i2 Analyze application. The database can be on the same or a separate server; the solid arrows show data flow between the components during data load and ingestion.



The examples in the lower-left and lower-right quadrants represent deployments in which the ETL logic is on a separate server from the i2 Analyze application. (Typically, the ETL logic is hosted alongside the database or the external data source.) To enable the architecture, those deployments include the *ETL toolkit*, which is a cut-down version of the main deployment toolkit that targets only data ingestion.

When you need the ETL toolkit, you can generate it on the i2 Analyze server, and copy it to the server that hosts the ETL logic. When the ETL toolkit is properly configured, your ETL logic can run toolkit commands without reference to the rest of the deployment.

Procedure

As the diagrams show, the ETL toolkit is most likely to be useful in deployments where the i2 Analyze application and the ETL logic are on separate servers. As you plan your approach to ingestion, consider the following:

- If the ETL logic is relatively simple and data volumes are low, there are benefits to colocating as many components as you can, especially in a new deployment.
- If your deployment requires separate servers from the start, or as it evolves over time, determine where the bottlenecks are. Is it limited by server speed or network speed?

- If the ETL logic is taxing a server that hosts other components, consider moving the logic, but be aware of the increase in network traffic.
- If the volume of data is taxing the network, consider colocating components when you are able. (You might not have permission to deploy components to some servers, for example.)

Results

By acting as a proxy for the i2 Analyze deployment toolkit, the ETL toolkit provides for more flexibility in your choice of architecture. In some circumstances you can separate the database, the ETL logic, and the i2 Analyze application without incurring a networking penalty.

What to do next

The decision that you reach about where to locate the ingestion components does not have to be permanent. You can start with a simple plan and change it later. If you decide to use the ETL toolkit, the next step is to deploy it. If not, you can move on to creating staging tables in the DB2 database.

Deploying the ETL toolkit

If your deployment includes logic that extracts, transforms, and loads data on a different server from the i2 Analyze application or the Information Store, consider deploying the ETL toolkit. The ETL logic can then run ETL toolkit commands to automate loading and ingesting data into the Information Store.

About this task

In an i2 Analyze deployment that uses data from an external source, the ETL logic is the processing that transforms source data for loading into the Information Store staging tables. In mature deployments, it is common for the ETL process to be automated so that loading and ingesting data happen in sequence, on a schedule.

When your ETL logic is colocated with the standard i2 Analyze deployment toolkit, the logic can use that toolkit to drive the ingestion process automatically. When those components are on separate servers, you can deploy the ETL toolkit to the server that hosts the ETL logic. The ETL toolkit provides the ingestion functionality of the deployment toolkit in a stand-alone package.

Procedure

The ETL toolkit must be able to communicate with the Information Store with all the same credentials as the deployment toolkit. To enable this behavior, you use the deployment toolkit to create the ETL toolkit, and then copy it to the ETL logic server.

1. On the server that has the deployment toolkit, open a command prompt and navigate to the `toolkit\scripts` directory.
2. Run the `createEtlToolkit` command to generate the ETL toolkit:

   ```
   setup -t createEtlToolkit -p outputPath=output_path
   ```

   This command creates the ETL toolkit in a directory named `etltoolkit` in the output path that you specify.
3. Copy the ETL toolkit to the server that hosts the ETL logic.

If the ETL logic and the ETL toolkit are on the same server as the instance of DB2 that hosts the Information Store, then you do not need to modify the connection configuration. If DB2 is on a different server, then you must also enable the ETL toolkit to communicate with the remote database.

4. Install DB2 client software on the server that hosts the ETL logic.

5. Navigate to the `classes` directory of the ETL toolkit and open the file named `Connection.properties` in a text editor.

   You need to do two things with `Connection.properties`. First, you must provide the ETL toolkit with the location of the DB2 client that you installed. Second, you must extract some information from the file so that you can catalog the remote DB2 database.
6. Add a property named `db.installation.dir.db2` to the `Connection.properties` file, and set it to the path of the DB2 client.
   For example:

   ```
   db.installation.dir.db2=C:/Program Files/IBM/SQLLIB
   ```

7. Make a note of the value of the DBURL property, which has this form: `jdbc:db2://`*`host-name`*`:`*`port-number`*`/`*`instance-name`*. Then, save and close `Connection.properties`.
8. Run the following commands to enable the ETL toolkit to communicate with the Information Store:

   ```
   db2 catalog tcpip node node-name host-name server port-number
   db2 catalog database instance-name at node node-name
   ```

   Here, *host-name*, *port-number*, and *instance-name* are the values that you recorded from the DBURL property. *node-name* can be any value that you choose, provided that you use the same value in both commands.

If the instance of DB2 that hosts the Information Store is not using SSL, then the process is complete. If the DB2 instance is configured to use SSL, you must also enable the ETL toolkit to communicate with DB2 using SSL.

9. Create a truststore and import into the truststore the certificate that you exported from the database management system when you configured SSL for DB2.
   For example, run the following command:

   ```
   keytool -importcert -alias "dbKey" -file C:\IBM\etltoolkit\i2-db2-
   certificate.cer -keystore "C:\IBM\etltoolkit\i2-etl-truststore.jks" -
   storepass "password"
   ```

   Enter yes in response to the query, `Trust this certificate?`
10. Navigate to the `classes` directory of the ETL toolkit and open the file named `TrustStore.properties` in a text editor.
11. Populate the `DBTrustStoreLocation` and `DBTrustStorePassword` properties with the full path to the truststore that you created, and the password that is required to access it.
    For example:

    ```
    DBTrustStoreLocation=C:/IBM/etltoolkit/i2-etl-truststore.jks
    DBTrustStorePassword=password
    ```

12. You can use the Liberty profile `securityUtility` command to encode the password for the truststore.
    a) Navigate to the `bin` directory of your WebSphere® Application Server Liberty profile deployment that is configured by the deployment toolkit.
    b) In a command prompt, run the following command: `securityUtility encode` *`password`*.

       The encoded password is displayed in the command line. Use the entire value, including the `{xor}` prefix, for the `DBTrustStorePassword` property value. For more information about using the security utility, see securityUtility command.

Results

The ETL toolkit is ready for use by your ETL logic to modify the Information Store. At key points in the processes of preparing for and performing ingestion, you can use commands in the ETL toolkit in place of deployment toolkit functionality.

Creating the staging tables

The Information Store does not ingest data directly from your data source. Instead, ingestion takes place from staging tables that you create and populate. This abstraction makes it easier for you to align your data with the Information Store, and allows i2 Analyze to validate your data before ingestion.

About this task

The simplest approach to Information Store ingestion is to create a staging table for every entity type, and every entity-link-entity type combination, that you identified in your data. The i2 Analyze deployment toolkit and the ETL toolkit both have a command for creating one staging table at a time.

The deployment toolkit command looks like this:

```
setup -t createInformationStoreStagingTable
      -p schemaTypeId=type_identifier
      -p databaseSchemaName=staging_schema
      -p tableName=staging_table_name
```

While the ETL toolkit command looks like this:

```
createInformationStoreStagingTable
      -stid type_identifier
      -sn staging_schema
      -tn staging_table_name
```

In both cases, *type_identifier* is the identifier of one of the entity types or link types from the i2 Analyze schema that is represented in your data source. *staging_schema* is the name of the database schema to contain the staging tables. (The command creates the database schema if it does not exist.) *staging_table_name* is the name of the staging table itself, which must be unique, and must not exceed 21 characters in length.

Important: Many of the commands that are associated with the ingestion process modify the DB2 database that hosts the Information Store. By default, the commands use the DB2 credentials that you specified during deployment in the `credentials.properties` file.

To use different credentials in the deployment toolkit, add `importName` and `importPassword` parameters to the list that you pass to the command. To use different credentials in the ETL toolkit, modify the `DBUsername` and `DBPassword` settings in the `Connection.properties` file.

Procedure

1. If you are using the deployment toolkit, open a command prompt and navigate to the `toolkit\scripts` directory. If you are using the ETL toolkit, navigate to the `etltoolkit` directory.
2. For each entity type or link type that you identified for ingestion, run the `createInformationStoreStagingTable` command.

For example:

```
setup -t createInformationStoreStagingTable
      -p schemaTypeId=ET5 -p databaseSchemaName=IS_STAGING
      -p tableName=E_PERSON
```

By convention, you create all of the staging tables for the same source in the same database schema, which has the name `IS_STAGING` in this example. It is also conventional to name the staging table itself similarly to the display name of the entity type or link type to which the table corresponds. In this case, the staging table is for the Person entity type.

Note: When the i2 Analyze schema allows the same link type between several different entity types, create several staging tables for one link type:

```
setup -t createInformationStoreStagingTable
      -p schemaTypeId=LAC1 -p databaseSchemaName=IS_STAGING
      -p tableName=L_ACCESS_TO_PER_ACC
```

```
setup -t createInformationStoreStagingTable
      -p schemaTypeId=LAC1 -p databaseSchemaName=IS_STAGING
      -p tableName=L_ACCESS_TO_PER_VEH
```

This example illustrates an Access To link type (with identifier `LAC1`) that can make connections from Person entities to Account entities, or from Person entities to Vehicle entities. The commands create staging tables with different names based on the same link type.

Results

At the end of this procedure, you have a set of staging tables that are ready to receive your data before ingestion takes place. The next task is to make your data ready to populate the staging tables.

Preparing the external data

The staging tables that you create during the ingestion process have data structures that are similar to, but simpler than, the Information Store data tables. Whatever your source is, you must find a way to shape the data that it contains into a form that is compatible with the staging tables.

About this task

After you create the staging tables, you can view them in IBM Data Studio (or similar software) to see the definitions of their columns. You must make your data match these definitions before you can go on to populate the staging tables.

Procedure

Because all data sources and many i2 Analyze schemas are different, there is no single procedure that you can follow to prepare your data for ingestion. However, there are a number of common considerations.

- Each staging table can contain data that maps to only one entity type or link type. If your source data has rows or records that contain data for more than one of the types that you identified, then you must separate them during preparation or population.

For data in a relational source, this preparation might mean creating views on the original tables. If the data is in CSV files, then you might need to wait until you populate the staging tables to change its shape in this way.

- The Information Store does not support storing properties with multiple values in the same i2 Analyze record. The records that you create must contain values for a maximum of one property with each permitted property type.
- If you are dealing with date and time data, that data must meet extra requirements before the Information Store can ingest it. To retain information unambiguously, the staging tables use four columns to represent date and time data.

  Even if you know that your date and time data was recorded in Coordinated Universal Time, you must make that fact explicit in the data to be ingested. For example, if your source contains information about an event that started at 9 AM on October 6, 2002, then the values you need to prepare are:

  `2002-10-06 09:00:00` (the data and time originally entered)
  `UTC` (the time zone)
  `0` (daylight saving time is not in effect)
  `2002-10-06 09:00:00` (the date and time in Coordinated Universal Time)
- All staging tables contain a `SOURCE_ID` column that does not map to a property type in the i2 Analyze schema. You must prepare a value for this column that reproducibly references the data in its source. In some cases, this preparation might only involve copying a value from another part of the record.
- If your external source is a relational database, you might find that the only data for some links is the presence of a foreign key relationship between two tables. In that case, you must synthesize a reproducible reference for the link from the other data that you have available.

  For example, you might be able to create a unique reference for a link by combining the identifiers of the entity records at its ends.
- All staging tables contain a `SOURCE_LAST_UPDATED` column that you can use to store information about when the data to be ingested was modified in its source.
- All staging tables contain `CORRELATION_ID_TYPE` and `CORRELATION_ID_KEY` columns. To correlate data that is ingested into the Information Store, use these columns to store the values that are used to create the correlation identifier for each row of data.

  If you specify values for a correlation identifier, you should also specify a value for the `SOURCE_LAST_UPDATED` column. The values of the `SOURCE_LAST_UPDATED` associated with the data is used during the correlation process.

  For more information about correlation, correlation identifiers, and the impact of the source last updated time see Overview of correlation.

  Important: Before you ingest data with correlation identifiers, you must install the i2 Analyze 4.2.0.1 Fix Pack or Enterprise Insight Analysis 2.2.0.1 Fix Pack. For more information about downloading and installing the Fix Packs, see Release Material.
- All staging tables contain columns for each of the access dimensions that the security schema defines. If your external source includes security information, then you can map that information to the security schema of your target deployment, and populate the staging table columns accordingly.

  Alternatively, you can leave the security columns blank, and provide security dimension values on a mapping- or source-wide basis later in the ingestion process.
- The staging tables for link types contain more columns that do not map directly to property types. Specifically, link type staging tables contain columns for the source identifiers of the entity records

at each end of the link, which you must populate. They also contain a column for the direction of the link.

The Information Store considers links to go "from" one entity "to" another. The direction of a link can be WITH or AGAINST that flow, or it can run in BOTH directions, or NONE.

- If your link data includes direction information, then you can add it to the staging table during the population process, and then refer to it from the mapping file.
- If your link data does not include direction information, then you can specify a value in the mapping file directly.

By default, if you have no direction information and you do nothing in the mapping file, the Information Store sets the direction of an ingested link to NONE.

Important: The Information Store places limits on the ranges of values that properties with different logical types can contain. If you attempt to use values outside these ranges, failures can occur during or after ingestion. For more information, see "Information Store property value ranges" on page 45.

Example

The `examples\data\law-enforcement-data-set-1` directory of the deployment toolkit contains a set of CSV files that were exported from a relational database.

In files like `event.csv`, you can see date and time data that meets the requirements of the staging tables. You can also see multiple files for "Access to" links, and how some staged link rows contain little more than a set of identifiers.

Populating the staging tables

The i2 Analyze deployment toolkit and the ETL toolkit create the staging tables for data ingestion in the same DB2 database as the Information Store data tables. After you prepare your data, but before you can instruct the Information Store to ingest it, you must populate the staging tables.

About this task

The approach that you take to populate the staging tables is likely to depend on the form that your source data is in, and the tools that you have available. For example, DB2 provides the ingest, import, and load utilities:

- If your data is in comma-separated value (CSV) files, then you can use the IMPORT or INGEST command to populate the staging tables.
- If your data is in the tables or views of another database, then you can use the IMPORT, INGEST, or LOAD command to do the work.

Alternatively, you can use IBM InfoSphere DataStage as a tool for transforming your data and loading it into the staging tables. You can specify the DB2 schema that contains the staging tables as the target location for the ETL output.

Example

The subdirectories of the `examples\data` directory in the deployment toolkit all contain a file with the name `LoadCSVDataCommands.db2`. In each case, this file is a DB2 script that populates the example staging tables from the prepared CSV files.

The script calls the IMPORT command repeatedly to do its work. In most instances, the command just takes data from columns in a CSV file and adds it to a staging table in a DB2 database schema.

Defining an ingestion source

The Information Store keeps a list of all the sources from which it has ingested data. Before it can ingest data, you must tell the Information Store about your source. In the ingestion mapping file, you then specify the data source name in the mapping definition for each entity type and link type.

About this task

The i2 Analyze deployment toolkit and the ETL toolkit both have a command for adding information about an ingestion source to the Information Store.

The deployment toolkit command looks like this:

```
setup -t addInformationStoreIngestionSource
      -p ingestionSourceName=src_name
      -p ingestionSourceDescription=src_display_name
```

While the ETL toolkit command looks like this:

```
addInformationStoreIngestionSource
      -n src_name
      -d src_display_name
```

In both cases, *src_name* is a unique name for the ingestion source, which also appears in the mapping file. *src_display_name* is a friendlier name for the ingestion source that might appear in the user interface of applications that display records from the Information Store.

Important: The value that you provide for *src_name* must be 30 characters or fewer in length. Also, do not use the word ANALYST as the name of your ingestion source. That name is reserved for records that analysts create in the Information Store through a user interface.

Procedure

1. If you are using the deployment toolkit, open a command prompt and navigate to the `toolkit\scripts` directory. If you are using the ETL toolkit, navigate to the `etltoolkit` directory.
2. Run the `addInformationStoreIngestionSource` command, specifying the short and display names of your ingestion source.
   For example:

   ```
   setup -t addInformationStoreIngestionSource
         -p ingestionSourceName=EXAMPLE
         -p ingestionSourceDescription="Example data source"
   ```

   If the Information Store already contains information about an ingestion source with the name EXAMPLE, this command has no effect.

Results

After you complete this task, you have performed all the necessary actions, and gathered all the necessary information, to be able to write ingestion mapping files. The next task is to create that file for your ingestion source.

Creating an ingestion mapping file

The mappings in an ingestion mapping file define how rows in staging tables become i2 Analyze records in the Information Store during the ingestion process. Each mapping that you write describes how to construct the origin identifiers for data of a particular type, and specifies the security dimension values that apply to records.

About this task

The Information Store ingestion mechanism makes it possible for you to develop and extend your ingestion mappings over time. You can test your approach to ingestion by writing and using a single (entity type) mapping, and then adding more entity type and link type mappings later. You can put all your mappings in one file, or put each mapping in a separate file, or anything between those two extremes.

Procedure

If you populated the staging tables successfully, then writing ingestion mappings can be straightforward. Eventually, you need a mapping for each staging table that you created, but you can approach the problem one mapping at a time.

1. Choose a populated staging table for an entity type that has links to other entity records in the data model.
2. Create an ingestion mapping file that contains an ingestion mapping for the staging table that you chose in step 1.

   If you prefer to start from an existing file, look at `mapping.xml` in the `examples\data\law-enforcement-data-set-1` directory of the deployment toolkit.
3. Run the ingestion command to test the mapping.

   If you are unhappy with the ingested data, edit the ingestion mapping and run the command again. If your changes do not affect the origin identifiers of the ingested data, the command overwrites the contents of any records that exist in the Information Store.

   Note: If the origin identifiers are the problem, follow the procedure to remove the data from the Information Store. Do not modify the mapping file or the staging table before you delete the data associated with that origin identifier.
4. Repeat steps 1, 2, and 3 for a second staging table whose entities have links to the records in the first one.
5. Repeat steps 1, 2, and 3 for a staging table that contains links that connect the entities from the first two tables.
6. Repeat all of the preceding steps for all the other staging tables that you populated.

Example

The `examples\data\law-enforcement-data-set-1` directory of the deployment toolkit contains an ingestion mapping file named `mapping.xml`. This file contains ingestion mappings for all the staging tables that the ingestion example creates. You can use `mapping.xml` as the basis for the ingestion mappings that you need for your data.

Running ingestion commands

After you populate the staging tables and write ingestion mappings, you can use toolkit commands to instruct the Information Store to ingest or update the records that represent external data. The

Information Store keeps a log of all such instructions that you can review to determine the success or failure of each one.

About this task

The commands in the i2 Analyze deployment and ETL toolkits enable you to create and update records in the Information Store. All three operation types are controlled by the data in the staging tables and the mappings in the ingestion mapping files.

After any operation that uses toolkit commands to change the contents of the Information Store, you can examine ingestion reports to determine how successful the operation was.

Adding data to the Information Store

After you create and populate your staging tables and write your ingestion mappings, the final part of the process is to run the ingestion command. It can be helpful to run the command twice for each mapping: first to validate your work, and then to instruct the Information Store to ingest your data.

Before you begin

Important: Before you ingest data with correlation identifiers, you must install the i2 Analyze 4.2.0.1 Fix Pack or Enterprise Insight Analysis 2.2.0.1 Fix Pack. For more information about downloading and installing the Fix Packs, see Release Material.

Note: To use correlation, you must understand the incoming data and confirm that it meets a number of conditions.

After you ingest data with correlation identifiers, the correlation identifiers and implicit discriminators that are ingested into the Information Store must not change. To ensure that your correlation identifiers and implicit discriminators do not change, the following must be true for the lifetime of that data in the Information Store:

- The method that you use to generate the correlation identifiers must not change. For example, if you used IBM InfoSphere Identity Insight to create the correlation identifiers, you must continue to do so.
- The property types and values that the method for generating correlation identifiers uses must not change. For example, if your correlation identifiers for people include data from their date of birth, then the value for their date of birth must not change and you must continue to use the date of birth property.
- The security dimension values for the data must not change. For example, if you ingest data with the security dimension values of `HI`, `OSI`, `CON`, you must always ingest that data with those security dimension values.
- If it is a link record, the link ends must not change. For example, if you ingest a link between entities A and B. Whenever you reingest that link, it must always have entities A and B as its link ends.
- If it is a link record, the link direction must not change. For example, if you ingest a link with direction of `W`, you must always ingest that data with direction `W`.

If you try to ingest data that causes the correlation identifier of a record to change, the data is rejected during the ingestion process.

If you later discover that the data does not meet these conditions, you must remove all of the data with correlation identifiers from the Information Store and reingest the data without using correlation.

About this task

When you instruct the Information Store to ingest the data that you loaded into the staging tables, you do it one ingestion mapping (and one staging table) at a time. The i2 Analyze deployment toolkit and the ETL toolkit both have a command for ingesting the data that is associated with a particular ingestion mapping in a particular mapping file.

The deployment toolkit command looks like this:

```
setup -t ingestInformationStoreRecords
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
      -p importLabel=ingestion_label
      -p importConfigFile=ingestion_settings_file
      -p importMode=STANDARD|VALIDATE|BULK
```

While the ETL toolkit command looks like this:

```
ingestInformationStoreRecords
      -imf ingestion_mapping_file
      -imid ingestion_mapping_id
      -il ingestion_label
      -icf ingestion_settings_file
      -im STANDARD|VALIDATE|BULK
```

Here, *ingestion_mapping_file* is the path to the XML file that contains the mapping that you want to use, and *ingestion_mapping_id* is the identifier of the mapping within that file. The latter is mandatory unless the file contains only one mapping.

The `importLabel`, `importConfigFile`, and `importMode` parameters are optional:

- When you specify `importLabel`, *ingestion_label* is a name that identifies a particular use of the ingestion command in the Information Store's `IS_PUBLIC.INGESTION_DELETION_REPORTS` view.
- When you specify `importConfigFile`, *ingestion_settings_file* is the path to a settings file that contains *name=value* pairs. You can refer to names in the settings file from references in the ingestion mapping file to use their values when you run the `ingestInformationStoreRecords` command.
- `importMode` is STANDARD by default. If you set it to `VALIDATE` instead, the command checks the validity of the specified mapping, but no ingestion takes place. In a limited range of circumstances, you can set it to BULK to accelerate the ingestion process.

Procedure

The procedure for instructing the Information Store to ingest your data is similar to many others in this process. You start with one type or one staging table, and build from there.

1. Choose an entity staging table that you populated with data and provided with an ingestion mapping.
2. Run the `ingestInformationStoreRecords` command in `VALIDATE` mode.
   For example:

   ```
   setup -t ingestInformationStoreRecords -p importMappingsFile=mapping.xml
         -p importMappingId=Person -p importMode=VALIDATE
   ```

The output to the console indicates whether the mapping you identified is valid, provides guidance when it is not valid, and gives a full list of column mappings. The command sends the same information to a log file that you can find at `toolkit\configuration\logs\importer\IBM_i2_Importer.log`.

3. Correct any problems in the ingestion mappings file (or any ingestion properties file that you specified) before you proceed with data ingestion.
4. Run the command again, without the `importMode` parameter, to instruct the Information Store to ingest your data.

   Note: You can improve the performance of entity ingestion by running `ingestInformationStoreRecords` for different entity types at the same time. *Do not* attempt to run the command for data of the same type at the same time.

   Note: Due to the increased number of operations and comparisons that are required as part of the correlation process, it can take longer to ingest data with correlation identifiers.
5. Repeat steps 1, 2, 3, and 4 for the other ingestion mappings that you created. Take care to run the command for entity types before you run it for link types.

   The ingestion process for links verifies that the entities at each end of the link are already ingested. If it fails to find them, the process fails.

Results

At the end of this procedure, all the external data that you populated the staging tables with is in the Information Store. To add or update records you can repopulate the staging tables and rerun the `ingestInformationStoreRecords` command.

Using bulk mode for faster ingestion

The i2 Analyze commands for ingesting records have a setting that enables them to operate more quickly in a limited set of circumstances. If the data to be ingested is new and valid, and you are able to take the system offline while the operation proceeds, bulk mode might be appropriate.

Before you begin

Bulk ingestion has all the same requirements as standard ingestion. It uses the same staging tables, and the same ingestion mapping file. However, to use bulk mode you must be able to stop the i2 Analyze server, and you must also be prepared to restore the Information Store from backup if the command fails.

Important: If the Information Store in your i2 Analyze deployment contains records, you *must* make a backup before you attempt bulk mode ingestion. Any error or failure in the process can have serious consequences for your data.

About this task

Bulk mode can be useful when you are populating the Information Store for the first time with a large amount of data. When you use the mechanism for ingesting i2 Analyze records into the Information Store in bulk, you take on more responsibility for data integrity in exchange for higher throughput.

Bulk mode has no update facility, so you must be presenting all the data in the staging tables to the Information Store for the first time. For similar reasons, the data that you present must not contain repeated origin identifiers.

Bulk mode cannot process correlation identifiers, the data that you present must not contain correlation identifier type or key values.

Each time that you run the command, you must determine whether it completed correctly and what to do if it did not.

Procedure

1. Populate the staging tables, define an ingestion source, and create a mapping file as for standard ingestion.

   Note: Do not include correlation identifiers in the data that you ingest using bulk mode.
2. Ensure that the same origin identifier does not appear twice in the incoming data.

   It is not valid for two i2 Analyze records to contain the same origin identifier, but the bulk ingestion process does not perform checks to enforce the rule while the command is running.
3. Use the `stop` command from the deployment toolkit to stop the i2 Analyze application server.

   Bulk ingestion requires an exclusive lock on the Information Store database tables, so the application must not be running when you request it.
4. Unless you are certain that no existing i2 Analyze records contain the same origin identifier as any of the incoming data, use the `clearData` command to empty the Information Store.

   Bulk ingestion is best suited to when you populate the Information Store for the first time, or possibly when you first add data to the store from a new source.
5. Use the `ingestInformationStoreRecords` command in BULK mode to ingest data with your first entity mapping.
   For example:

   ```
   setup -t ingestInformationStoreRecords -p importMappingsFile=mapping.xml
         -p importMappingId=Person -p importMode=BULK
   ```

   The command has three possible outcomes. If the process was successful, you can move on to ingest records with the next mapping. Alternatively, the process can fail during ingestion or as a result of post-ingestion checks. If the process fails, you must repair the Information Store.
6. If the output from the command to the console contains errors, the Information Store is in an inconsistent state and you must resolve it:

   a) Use the `clearData` command from the deployment toolkit to empty the Information Store.

   b) If the Information Store contained i2 Analyze records before you started bulk ingestion, restore it from your backup.

   c) Address the causes of the errors, and then restart bulk ingestion from the beginning.
7. Repeat steps 4 and 5 for the other ingestion mappings that you created. Take care to run the command for entity records before you run it for link records. That requirement still applies, even in bulk mode.
8. Use the `start` command from the deployment toolkit to restart the i2 Analyze application server.
9. Revert to using the standard process rather than bulk mode for any future ingestion of data from the same source.

Updating the Information Store for changed data

The data that the Information Store ingests is fixed at the moment of ingestion, and changes to the data in its source do not automatically update the Information Store. However, you can update the

Information Store to reflect changes in an external source by running through the ingestion process again.

About this task

For most changes to the data in an external source, it is likely that you can reuse the work that you did to enable initial ingestion. If the changes to an external source are not significant enough to affect your method for generating reproducible origin identifiers, repeat ingestion follows the same process as initial ingestion.

Procedure

1. Examine the new data in the external source, and your ingestion mappings. Confirm that your configuration still generates origin identifiers that the Information Store can compare with their equivalents in existing ingested data.
2. Delete the contents of each staging table that you know to be affected by changes to the external data.
3. Populate the affected staging tables with the latest data from your external source.
4. Run the ingestion command for each ingestion mapping that refers to an affected staging table, taking care to process entity data before link data, as usual.

   The Information Store uses the origin identifier of each row that it attempts to ingest to determine whether the data is new:

   - If the origin identifier does not match the origin identifier of any data that is already in the Information Store, then the data is new to the Information Store and is ingested in the usual way.
   - If the origin identifier does match the origin identifier of any data that is already in the Information Store, then the staging table contains updated information. The Information Store clears its existing data and refills it with the new data.

   Note: If the correlation identifier or implicit discriminator is changed, the row is rejected with the `CORRELATION_CHANGES` error category. For more information about correlation changes, see "Troubleshooting the ingestion process" on page 42.

Results

After you follow this procedure, the Information Store contains new data that was added to an external source since the last ingestion. It also contains updated data that was changed in an external source since the last ingestion.

Updating the Information Store for deleted data

The data that the Information Store ingests is fixed at the moment of ingestion, and removal of the data in its source does not automatically delete it from the Information Store. However, you can remove the provenance from the Information Store to reflect the changes in the external source by using staging tables and the deployment toolkit.

Before you begin

When data changes in its original source, you can use the same pipeline that you used for initial ingestion to update the records in the Information Store. If data is deleted from its source, you can use the staging tables and the deployment toolkit to reflect that fact in the Information Store as well.

A single i2 Analyze record can represent data from multiple sources, which is results in the record containing multiple pieces of provenance. As a consequence, responding to source data deletion does not necessarily mean deleting records from the Information Store. When you use the toolkit to reflect deleted source data, the effect is to remove the provenance associated with that data. If the process removes a record's only provenance, the record is deleted. If not, the record remains in the Information Store.

Note: To delete records from the Information Store use the deletion-by-rule approach. You can write conditions to determine which records are deleted. For more information about deleting records in this way, see Deleting records by rule.

About this task

The commands to update the Information Store for deleted data use the same mapping file and the same staging tables as the commands for ingesting data, and you call them in a similar way. However, the only information that *must* be in the staging table is what the mapping file requires to generate the origin identifiers of the data that is no longer in the external source.

When running the commands to update the Information Store for deleted data, the rules are different from adding and updating data in several ways:

- Links do not have to be processed before entities, or vice versa.
- Links can be processed without specifying the origin identifiers of their ends.
- The process silently ignores any origin identifiers that are not in the Information Store.

If, as a result of deleting the last piece of provenance for a record, an entity record is deleted, all the link records that are connected to it are also deleted.

Because this process might cause significant numbers of i2 Analyze records to be deleted, IBM recommends running two commands. The first command previews the effect of running the second command before you commit to doing so. In the deployment toolkit, the two commands have different names but the same syntax:

```
setup -t previewDeleteProvenance
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
```

```
setup -t deleteProvenance
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
      -p importLabel=ingestion_label
      -p logConnectedLinks
```

In the ETL toolkit, you reuse the `ingestInformationStoreRecords` command with two new mode parameters:

```
ingestInformationStoreRecords
      -imf ingestion_mapping_file
      -imid ingestion_mapping_id
      -im DELETE_PREVIEW
```

```
ingestInformationStoreRecords
      -imf ingestion_mapping_file
      -imid ingestion_mapping_id
      -il ingestion_label
      -lcl true
      -im DELETE
```

In all cases, *ingestion_mapping_file* is the path to the XML file that contains the mapping that you want to use, and *ingestion_mapping_id* is the identifier of the mapping within that file. The latter is mandatory unless the file contains only one mapping.

When `logConnectedLinks` is specified, any links to the removed provenance are logged in `IS_PUBLIC.D`*`<import identifier><entity type id>`*`_`*`<link type id>`*`_LINKS` tables. For example, `IS_PUBLIC.D20180803090624143563ET5_LAC1_LINKS`. After you delete a piece of provenance from an i2 Analyze record with multiple pieces of provenance, any links to the provenance remain linked to the record. The links are only removed when the i2 Analyze record is removed. It is recommended to use this parameter when your Information Store contains data that has been correlated because you might want to update the Information Store to remove the remaining links. In the tables, if the value of the `DELETED` column is `N` that link remains in the Information Store. To remove the remaining links, you can complete the following procedure populating your staging tables with the required information to generate the origin identifiers of the links identified in the tables.

Previewing the delete operation does not create an entry in the `INGESTION_DELETION_REPORTS` view, so there is no need to specify a label in that case. The delete operation does populate that view, and *ingestion_label* is then an optional parameter.

Procedure

The procedure for updating the Information Store in this way starts with a staging table that contains information about the data that you no longer want to represent in the Information Store.

1. Run the `previewDeleteProvenance` command to discover what the effect of running `deleteProvenance` is.
   For example:

   ```
   setup -t previewDeleteProvenance -p importMappingsFile=mapping.xml
         -p importMappingId=Person
   ```

The output to the console window describes the outcome of a delete operation with these settings. High counts or a long list of types might indicate that the operation will delete more records than you expected.

```
>INFO [DeleteLogger] - Delete preview requested at 2017.12.08 11:05:32
>INFO [DeleteLogger] - Item type: Person
>INFO [DeleteLogger] - Number of 'Person' provenance pieces to be deleted: 324
>INFO [DeleteLogger] - Number of 'Person' i2 Analyze records to be deleted: 320
>INFO [DeleteLogger] - Number of 'Access To' provenance pieces to be deleted:
187
>INFO [DeleteLogger] - Number of 'Access To' i2 Analyze records to be deleted:
187
>INFO [DeleteLogger] - Number of 'Associate' provenance pieces to be deleted:
27
>INFO [DeleteLogger] - Number of 'Associate' i2 Analyze records to be deleted:
27
>INFO [DeleteLogger] - Number of 'Employment' provenance pieces to be deleted:
54
>INFO [DeleteLogger] - Number of 'Employment' i2 Analyze records to be
deleted: 54
>INFO [DeleteLogger] - Number of 'Involved In' provenance pieces to be
deleted: 33
>INFO [DeleteLogger] - Number of 'Involved In' i2 Analyze records to be
deleted: 33
>INFO [DeleteLogger] - Duration: 1 s
```

Note: When you run the command for entity records, the output can exaggerate the impact of the operation. If the staging table identifies the entities at both ends of a link, the preview counts the link record twice in its report.

2. Correct any reported problems, and verify that the statistics are in line with your expectations for the operation. If they are not, change the contents of the staging table, and run the preview command again.
3. Run the `deleteProvenance` command with the same parameters to update the Information Store.
   For example:

   ```
   setup -t deleteProvenance -p importMappingsFile=mapping.xml
         -p importMappingId=Person -p importLabel=DeletePeople
         -p logConnectedLinks
   ```

   Note: Do not run multiple `deleteProvenance` commands at the same time, or while data is being ingested into the Information Store.
4. Repeat steps 1, 2, and 3 for the types of any other records that you want to process.

Results

At the end of this procedure, the Information Store no longer contains the provenance for the data that you identified though the mapping file and the staging tables. Any records that have lost all their provenance, and any connected links, have been deleted as a result. Deleting data is permanent, and the only way to restore it to the Information Store is to add it again through the `ingestInformationStoreRecords` command.

Understanding ingestion reports

Every attempt to add or update data in the Information Store through the deployment or ETL toolkit adds rows to the IS_PUBLIC.INGESTION_DELETION_REPORTS view. You can use the contents of this view to track the history of all such operations, and to examine the impact of a particular operation.

About this task

Each time you run a command that might change the contents of the Information Store, you create a job in the DB2 database. Each job acts on one or more batches of i2 Analyze records. There is always one batch per item type that the command affects, but there can also be several batches for the same type if the number of affected records is large.

For example, consider a command that processes updates for deleted Person entity data. The first batch in the resulting job is for Person records, and there might be more such batches if there are many records to be deleted. If the Person records have links, then the job has further batches for each type of link that may get deleted as a result of the entity deletion.

The IS_PUBLIC.INGESTION_DELETION_REPORTS view contains information about every batch from every toolkit operation to create or update data in the Information Store.

Note: Deletion-by-rule operations also result in job and batch creation, and view population, according to the same rules. For more information, see the *Deletion Guide*.

The first few columns in the view have the same value for all batches within a job:

Column name	Description
LABEL	The value that you passed in the importLabel parameter of a toolkit command, or the value that a deletion-by-rule operation generates, or null.
JOB_ID	The server-assigned identifier for this ingestion or deletion job. This identifier is also a cross-reference to the DELETION_BY_RULE_LOG view if the job originated from a deletion-by-rule operation.
INGESTION_MODE	The value that you passed in the importMode parameter, or **Delete** for all deletion-by-rule operations.
VALIDATION_MODE	A description of how the job was configured to react to errors during the operation.
ERROR_THRESHOLD	The threshold that applies to some of the validation modes.
PRIMARY_ITEM_TYPE	The i2 Analyze schema ID of the item type that was specified at job creation.
PRIMARY_RECORD_COUNT	The number of records of the primary item type that were affected by the job. (Remember that deleting an entity record can affect link records too.)
START_TIME	The start time of the job as a whole.
END_TIME	The end time of the job as a whole.

The remaining columns can have different values for different batches of records:

Column name	Description
BATCH_ITEM_TYPE	The i2 Analyze schema ID of the item type that was acted on in this batch. For at least one batch, the BATCH_ITEM_TYPE is the same as the PRIMARY_ITEM_TYPE.
BATCH_START_TIME	The start time of this batch, which is always later than the start time of the job.
BATCH_END_TIME	The end time of this batch, which is always earlier than the end time of the job.
INSERT_COUNT	The number of rows of data from this batch that were inserted to the Information Store, resulting in new i2 Analyze records.
UPDATE_COUNT	The number of rows of data from this batch that updated existing records in the Information Store.
MERGE_COUNT	The number of *merge* operations that occurred in the Information Store from this batch.
REJECT_CORRELATION_COUNT	The number of rows that were rejected from this batch, because they change the correlation identifier or implicit discriminators of a record.
DELETE_COUNT	The number of pieces of provenance deleted from the Information Store as a result of this batch.
DELETE_RECORD_COUNT	The number of records deleted from the Information Store as a result of this batch.
REJECT_COUNT	The number of rows that were rejected from this batch during processing because they are invalid.
STATUS	An indicator of the result of this batch, from success (all rows processed correctly) through partial success to failure (no rows processed).
REJECT_VIEW	The full name of the view that contains details of any rejected rows.
STACK_TRACE	If i2 Analyze generated a stack trace as a result of errors during ingestion or deletion, this column contains it.

Examples

Ingest example

For example, the (abbreviated) report for successful ingestion operations might look like this:

JOB_ID	1	2	2
INGESTION_MODE	Standard	Standard	Standard
PRIMARY_ITEM_TYPE	ET10	ET4	ET4
PRIMARY_RECORD_COUNT	62	5	1
BATCH_ITEM_TYPE	ET10	ET4	ET4
BATCH_START_TIME	2017-11-30 15:27:06.76	2017-11-30 15:27:09.45	2017-11-30 15:27:09.46

BATCH_END_TIME	2017-11-30 15:27:09.87	2017-11-30 15:27:09.63	2017-11-30 15:27:09.63
INSERT_COUNT	57	3	0
UPDATE_COUNT	0	0	0
MERGE_COUNT	5	2	0
REJECT_CORRELATION_COUNT	0	0	0
DELETE_COUNT	0	0	0
DELETE_RECORD_COUNT	0	0	1
REJECT_COUNT	0	0	0
STATUS	Succeeded	Succeeded	Succeeded

In this example, several commands to ingest entity records resulted in the creation of several jobs. Each job demonstrates different behavior that is possible during ingestion, including correlation operations:

JOB_ID 1

This job demonstrates what the ingestion report can look like when data in the staging table causes merge operations. In this example, five merge operations have been completed on the incoming rows of data, as shown in the MERGE_COUNT column. This results in 57 i2 Analyze records created from the 62 rows of data, as shown in the INSERT_COUNT and PRIMARY_RECORD_COUNT columns. This includes merging 5 rows of data with existing i2 Analyze records in the Information Store.

JOB_ID 2

This job demonstrates what the ingestion report can look like when the data in the staging table causes a merge operation. Two merge operations occurred, as shown in the MERGE_COUNT column. As part of one of the merge operations, an existing i2 Analyze link record would become circular because the each end of the link is now the same i2 Analyze record. This i2 Analyze link record is deleted from the Information Store, as shown in the DELETE_RECORD_COUNT column.

Delete example

For example, the (abbreviated) report for a successful delete operation might look like this:

JOB_ID	26	26	26	26	26
INGESTION_MODE	Delete	Delete	Delete	Delete	Delete
PRIMARY_ITEM_TYPE	ET5	ET5	ET5	ET5	ET5
PRIMARY_RECORD_COUNT	324	324	324	324	324
BATCH_ITEM_TYPE	ET5	LAC1	LAS1	LEM1	LIN1
BATCH_START_TIME	2017-11-30 15:27:06.76	2017-11-30 15:27:08.60	2017-11-30 15:27:08.60	2017-11-30 15:27:09.43	2017-11-30 15:27:09.45
BATCH_END_TIME	2017-11-30	2017-11-30	2017-11-30	2017-11-30	2017-11-30

	15:27:09.87	15:27:09.30	15:27:09.29	15:27:09.62	15:27:09.63
INSERT_COUNT	0	0	0	0	0
UPDATE_COUNT	0	0	0	0	0
MERGE_COUNT	0	0	0	0	0
REJECT_CORRELATION_COUNT	0	0	0	0	0
DELETE_COUNT	324	187	27	54	33
DELETE_RECORD_COUNT	320	187	27	54	33
REJECT_COUNT	0	0	0	0	0
STATUS	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded

In this example, a command to update the Information Store for deleted entity data (with item type ET5) resulted in the creation of a job with five batches. The first few columns of the INGESTION_DELETION_REPORTS view contain the same values for all batches in the same job. Later columns reveal how deleting entity records results in the deletion of connected link records (with item types LAC1, LAS1, LEM1, LIN1).

The DELETE_RECORD_COUNT value is less than the DELETE_COUNT value, this is because some of the provenance to be deleted was associated with an i2 Analyze record that had more than one piece of provenance. An i2 Analyze record is deleted only when the last associated provenance is deleted.

Troubleshooting the ingestion process

The commands that you run during the ingestion process send information about their progress to the command line and a log file. If any command encounters errors or does not run to completion, you can read the output to help you to diagnose the problem.

When an ingestion process runs to completion, the final output from the command is a report of what happened to the Information Store. The reports appear on the command line and in the ingestion log at toolkit\configuration\logs\importer\IBM_i2_Importer.log. The three possible end states are success, partial success, and failure.

Success

If the ingestion command processed all of the rows in the staging table without error, then the Information Store reflects the contents of the staging table. The command reports success like this example:

```
> INFO  [IImportLogger] - Total number of rows processed: 54
> INFO  [IImportLogger] - Number of records inserted: 0
> INFO  [IImportLogger] - Number of records updated: 54
> INFO  [IImportLogger] - Number of  merges: 0
> INFO  [IImportLogger] - Number of rows rejected: 0
> INFO  [IImportLogger] - Number of rows rejected because of correlation 
changes: 0
> INFO  [IImportLogger] - Duration: 5 s
> INFO  [IImportLogger] -
> INFO  [IImportLogger] - Result: SUCCESS
```

Partial success

If you ran the command in record-based failure mode, and it processed some of the rows in the staging table without error, then it reports partial success like this example:

```
> INFO  [IImportLogger] - Total number of rows processed: 34
> INFO  [IImportLogger] - Number of records inserted: 0
> INFO  [IImportLogger] - Number of records updated: 30
> INFO  [IImportLogger] - Number of  merges: 0
> INFO  [IImportLogger] - Number of rows rejected: 4
> INFO  [IImportLogger] - Number of rows rejected because of correlation 
changes: 0
> INFO  [IImportLogger] - Duration: 4 s
> INFO  [IImportLogger] -
> INFO  [IImportLogger] - Result: PARTIAL SUCCESS
> INFO  [IImportLogger] -
> INFO  [IImportLogger] - Total number of errors: 4
> INFO  [IImportLogger] - Error categories:
> INFO  [IImportLogger] - ABSENT_VALUE: 4
> INFO  [IImportLogger] -
> INFO  [IImportLogger] - The rejected records and errors are recorded in 
the database. For details, use the following view:
> INFO  [IImportLogger] - IS_STAGING.S20171204122426717092ET5_Rejects_V
```

The records in the Information Store reflect the rows from the staging table that the command successfully processed. The report includes the name of a database view that you can examine to discover what went wrong with each failed row.

Note: If a row is rejected because of correlation changes, the failed rows are added to a database table in addition to the current correlation data for the record in the Information Store that the row is updating. In the table, the columns prefixed with X_ contain the current correlation identifier and implicit discriminators for the record.

To ingest the row successfully, ensure that the correlation identifier or implicit discriminators of the row that you are ingesting match that of the associated record as listed in the table. For more information about the rules for correlation identifiers and implicit discriminators, and what to do if you must change them, see Correlation identifiers.

Failure

If you ran the command in mapping-based failure mode, then any error you see is the first one that it encountered, and the report is of failure:

```
> INFO  [IImportLogger] - Total number of rows processed: 1
> INFO  [IImportLogger] - Number of records inserted: 0
> INFO  [IImportLogger] - Number of records updated: 0
> INFO  [IImportLogger] - Number of  merges: 0
> INFO  [IImportLogger] - Number of rows rejected: 0
> INFO  [IImportLogger] - Number of rows rejected because of correlation
changes: 0
> INFO  [IImportLogger] - Duration: 0 s
> INFO  [IImportLogger] -
> INFO  [IImportLogger] - Result: FAILURE
```

When the process fails in this fashion, the next lines of output describe the error in more detail. In this event, the command does not change the contents of the Information Store.

Note: If a serious error occurs, it is possible for the ingestion command not to run to completion. When that happens, it is harder to be certain of the state of the Information Store. The ingestion process uses batching, and the records in the store reflect the most recently completed batch.

If the command reports partial success, you might be able to clean up the staging table by removing the rows that were ingested and fixing the rows that failed. However, the main benefit of record-based failure is that you can find out about multiple problems at the same time.

The most consistent approach to addressing failures of all types is to fix up the problems in the staging table and run the ingestion command again. The following sections describe how to react to some of the more common failures.

Link rows in the staging table refer to missing entity records

When the Information Store ingests link data, you might see the following error message:

```
Link data in the staging table refers to missing entity records
```

This message appears in the console output if the entity record at either end of a link is not present in the Information Store. To resolve the error, examine the console output for your earlier operations to check that the Information Store ingested all the entity records properly. Then, check that every link in the staging table includes the unique identifiers for the entities that it connects. Finally, rerun the ingestion command.

Rows in the staging table have duplicate origin identifiers

During any ingestion procedure, but especially when a staging table is large, you might see the following error message:

```
Rows in the staging table have duplicate origin identifiers
```

This message appears in the console output when several rows in a staging table generate the same origin identifier. (For example, more than one row might have the same value in the `SOURCE_ID` column.)

If you have incoming data that acts on the same provenance more than once, you must resolve those multiple actions into a single row before you populate the staging table. Alternatively, you can separate the actions so that they are not in the same staging table at the same time.

This problem is most likely to occur during an update to the Information Store that attempts to change the same provenance twice in the same batch. It might be appropriate to combine the changes, or to process only the last change. After you resolve the problem, repopulate the staging table and rerun the ingestion command.

Information Store property value ranges

The Information Store places limits on the ranges of values that properties can contain. Different logical types in the i2 Analyze schema imply different limits, which are not always the same as the restrictions on the underlying database. It is important to consider the limits when you prepare data for ingestion.

Logical type	Permitted values
SINGLE_LINE_STRING	Up to 250 bytes of UTF-8 characters
MULTI_LINE_STRING	Up to 32700 bytes of UTF-8 characters
SELECTED_FROM	Same as SINGLE_LINE_STRING
SUGGESTED_FROM	Same as SINGLE_LINE_STRING
DATE	From 1753-01-01 to 9999-12-30
TIME	From 00:00:00 to 23:59:59 *
DATE_AND_TIME	From 1753-01-01T00:00:00Z to 9999-12-30T23:59:59Z *
BOOLEAN	true or false
INTEGER	From -2^{31} to 2^{31} - 1
DOUBLE	From 4.9 x 10^{-324} to 1.79769313486231 x 10^{308} (Equivalent range for negative values. Maximum 15 digits of precision.)
DECIMAL	From -999999999999999999.9999 to 999999999999999999.9999 (Maximum 18 digits before the decimal mark. Maximum 4 digits after it.)

* The DB2 database that underlies the Information Store allows you to load time values that represent midnight as 24:00:00. When it stores such values, the database converts them to fit the ranges in the table.

In addition to the values in the table, you can set the value of any non-mandatory property to null. In the staging table for an item type that has a DATE_AND_TIME property type, all four columns that the value is spread across must be null in that case.

Notices

This information was developed for products and services offered in the U.S.A.

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.

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.

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 United Kingdom Limited
Hursley House
Hursley Park
Winchester
Hants SO21 2JN
U.K.

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, i2, and ibm.com 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.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.