DB2 Information Management Software

IBM

DB2 UDB Exploitation of the Windows Environment

- Detailed examples for leveraging DB2 UDB V8.1 in Windows 2000 environment
- Installing, security, monitoring, performance, high availability
- Application development guidelines

Whei-Jen Chen
Chris Fierros
Alexandre H. Guerra
Thomas Mitchell
Francis Reddington

Redbooks

ibm.com/redbooks

International Technical Support Organization

DB2 UDB Exploitation of the Windows Environment

March 2003

SG24-6893-00

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

First Edition (March 2003)

This edition applies to DB2 Universal Database Version 8.1 for use with Microsoft Windows 2000 operating system.

Contents

Figures

Tables

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 give 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.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AFS®
AIX®
Approach®
AS/400®
CICS®
DataJoiner®
DB2®
DB2 Connect™
DB2 Extenders™
DB2 Universal Database™
Distributed Relational Database Architecture™
DPI®
DRDA®
Everyplace™
IBM®
Informix®
Intelligent Miner™
iSeries™
Lotus®
MQSeries®
MVS™
MVS/ESA™
Net.Data®
Notes®
OS/2®
OS/390®
OS/400®
Perform™
QMF™
Redbooks™
Redbooks (logo)™ 
S/390®
SecureWay®
Sequent®
SmartSuite®
SOM®
SQL/DS™
SystemView®
Tivoli®
TME®
VisualAge®
WebSphere®
Word Pro®
z/OS™
zSeries™

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

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 Sun Microsystems, Inc. in the United States, other countries, or both.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Preface

This IBM Redbook offers you a broad understanding of how DB2 UDB V8.1 is integrated with Windows 2000 and Windows Server 2003. The book is organized as follows.

Chapter 1. Introduction
In this chapter we introduce DB2 Universal Database (UDB) and the IBM DB2 family of relational database management systems. We describe in detail the capabilities and features of DB2 UDB for Windows and how it integrates with the Microsoft Windows environments, and provide guidance on which DB2 products should be licensed for different environments.

Chapter 2. Installation and deployment
In this chapter we provide step-by-step procedures for installing DB2 server and client software, covering the full range from a single machine installation which requires user response, through automated software management. We also discuss basic Active Directory information and installation, and show how to integrate DB2 components into Lightweight Directory Access Protocol (LDAP).

Chapter 3. Post-installation tasks
In this chapter we discuss the post installation tasks that should be performed to have a functional, well-performing database, without having to get into detailed performance tuning. It is highly recommended that these tasks be done by every installation, since the default configuration parameters supplied by DB2 only satisfy the needs of a minimal system. We also discuss several methods on how to populate a database.

Chapter 4. Security
In this chapter we describe how DB2 Universal Database V8.1 exploits the security features built into the Windows 2000 operating system. We first introduce the Windows security model and point out the differences between Windows NT and Windows 2000, as we consider basic security concepts such as authentication and authorization. Then we describe the steps required to implement security for DB2 UDB in the Windows environment. Specifically, we consider DB2 security at the system, instance, and database levels.

Chapter 5. Performance
In this chapter we discuss the tasks involved in tuning the database and Windows environments to obtain optimal performance, especially the major items that experience has shown to have the largest impact on performance.

Chapter 6. Monitoring and management

In this chapter we describe the DB2 UDB V8.1 monitoring feature, including how DB2 integrates with the Windows system monitoring functions. We describe monitoring tool usage and data collection, and explain how to utilize the monitoring information with best-practice methods to manage failures or reduce possible damage.

Chapter 7. High availability

In this chapter we describe the high availability (HA) features of DB2 Universal Database V8.1 for Windows 2000 operating systems. We discuss not only the high availability features that are built into DB2 UDB, but also the integration of DB2 UDB into the high availability features of the Windows 2000 operating system. We begin by describing database availability features that can be implemented with relative ease and progress to the more highly available configuration feature that can provide for 99.999% uptime.

Chapter 8. Application development

In this chapter we describe the basic concepts and tools available to do application development with DB2. We cover the basic development tools available to the developer and the common languages available on the Windows platform, as well as providing recommendations, best practices, and application development tips.

Chapter 9. Windows scripting

In this chapter we cover the various means in which a DBA can create and execute a script. We highlight various languages, object models, and libraries available to a DBA to produce creative and productive scripts. We illustrate, through many coding example, how fundamental DBA tasks can be scripted in a multitude of scripting languages. We show the benefits of each language and explain where you can find installation and support for each language, as well as how a script can be conveniently scheduled to run at an optimal scheduled time.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Whei-Jen Chen is a Project Leader at the International Technical Support Organization, San Jose Center. She has extensive experience in application development, database design and modeling, and DB2 system administration. Whei-Jen is an IBM Certified Solutions Expert in Database Administration and Application Development as well as an IBM Certified IT specialist.

Chris Fierros is a DB2 Solutions Expert and principle consultant at Ten Digit Consulting, Inc. specializing in consulting, services, training, and support for DB2 Universal Database. Chris began his career with DB2 on distributed platforms in 1991 at IBM Corporation, where he worked in technical support, global services, and consulting practices. He has published articles, presented at conferences, and developed training courses about DB2. Chris is an IBM Certified Solutions Expert (ICSE) in DB2 UDB Administration and Development as well as an IBM Certified Advanced Technical Expert (CATE) in DB2 Clustering (EEE), DB2 Replication (DpropR), and DB2 Connect (DRDA). Chris has spent the last couple of years helping customers scale DB2 UDB on large 16-way and 32-way Intel servers running Windows 2000 and Windows Server 2003 Datacenter edition. Chris can be reached at chris@tendigit.com.

Alexandre H. Guerra is a DBA / Development Support Analyst in Brazil. He is currently a consultant at Orion/ZL Consulting working on an accounting system interconnecting all national financial institutes (SPB - Sistema de Pagamentos Brasileiro). Alex has 5 years experience in the Windows environment, development tools, and languages, including Visual Studio family and Delphi. His main areas of expertise are database planning, configuration and management, and tool software development, including data collecting systems using wireless specialized hardware, systems analysis, and programming. Alex can be reached at alexhguerra@hotmail.com.

Thomas Mitchell is a Certified Consulting Software IT Specialist for Data Management with IBM's Software Group in the United States, with responsibility for pre-sales and post sales technical support. He has been with IBM for nearly 30 years, working primarily with IBM database products across all platforms. He was one of the first two individuals in IBM to be Certified as a Data Management Specialist. He is the author of the tool OS2SPUFI that was widely used around the world until its features were included in the IBM distributed DB2 products.

Francis Reddington is an I/T Architect in IBM Global Services National Architects Center of Excellence (COE), who resides in the United States. He holds a Bachelor of Science degree from Rider University. He has over 16 years of application design and development experience as well as over 7 years pioneering Web-based applications since the origin of the World Wide Web. Francis specializes in emerging technologies and application integration. He currently shares seven patents pending in pervasive computing technologies. He has extensive knowledge of several programming languages, databases, design methodologies, and project management, and has worked on various UNIX, Windows NT, and Windows 2000 platforms. Francis has professional experience in a multitude of industries: government, military, aerospace, aviation, telecommunications, transportation, pharmaceutical, insurance, banking, finance, retail, mass media, health care, consumer products, and manufacturing. He has written and taught extensively on a multiple of I/T topics and is currently co-authoring a book, *The Enablement of Grid Computing*.

Thanks to the following people for their contributions to this project:

Michael R. Logan
IBM DB2 Product Management and Marketing

Dale Hagen, Andrew Hilden, Gene Kligerman, Leon Katsnelson, Desmond Lam, Dale McInnis, Aslam Nomani, Hiep Phuong, Karl Rempel, Scott Walkty, Shili Yang
IBM Toronto Labortory

Stefan Dörsam
IBM Böblingen Lab

Monty Wright
IBM Advanced Technical Support

Nagraj Alur, Emma Jacobs, Yvonne Lyon, Deanna Polm
International Technical Support Organization, San Jose Center

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. QXXE Building 80-E2
  650 Harry Road
  San Jose, California 95120-6099

1

Introduction

In this chapter we introduce DB2 Universal Database (UDB) and the IBM DB2 family of relational database management systems. We describe in detail the capabilities and features of DB2 UDB for Windows and how it integrates with the Microsoft Windows environments. We also provide guidance on which DB2 products should be licensed for different environments.

This chapter contains the following sections:

- DB2 UDB overview
  - DB2 family
  - DB2 UDB for Windows, UNIX, and Linux
- DB2 UDB products on Windows
- Planning considerations
- DB2 UDB Version 8 highlights
- DB2 UDB integration with Microsoft Windows

1.1 DB2 UDB overview

In this section we provide a detailed description of IBM DB2 Universal Database for Windows, UNIX, and the Linux product family, and discuss their features.

1.1.1 DB2 family

The IBM DB2 database software family is the worldwide leader in the industry and marks the next stage in the evolution of the relational database. DB2 is the industry's first multimedia, Web-ready relational database management system delivering leading capabilities in reliability, performance, and scalability. DB2 is the database of choice for customers and partners developing and deploying critical solutions. The DB2 family is a consistent set of relational database management systems (RDBMS) utilizing shared technologies and a common application programming interface.

These are the specific RDBMS products that make up the DB2 family:

- **DB2 Universal Database for z/OS and OS/390:** The premier IBM enterprise RDBMS for use on the mainframe to run powerful enterprise applications, and make large scale e-commerce a reality
- **DB2 Universal Database for iSeries:** An advanced, 64-bit relational database system that provides leading-edge performance in e-business and data warehousing environments (the iSeries and DB2 UDB for iSeries in combination provide the flexibility and adaptability to support any type of workload, small or large)
- **DB2 Server for VSE and VM:** A full-function RDBMS supporting production and interactive IBM VM and VSE environments for your company
- **DB2 Everyplace:** DB2 relational database and enterprise synchronization architecture for mobile and embedded devices
- **DB2 Universal Database V8.1:** An IBM relational database management system for AIX, Linux, HP-UX, Sun, and Windows (discussed in more detail in the next section)

1.1.2 DB2 UDB for Windows, UNIX, and Linux

In this section we consider the particular characteristics of DB2 UDB on the various platforms.

Overview

DB2 Universal Database (UDB) is the IBM object-relational database solution for the UNIX, Linux, and Windows operating environments. It is built on a solid foundation, bringing together a client/server database product with IBM's leadership in mission-critical relational database technology. The result is a highly scalable, highly extensible, very easy-to-use and manage database that can be trusted with your most critical database applications. DB2 UDB is available on the following platforms:

- Windows NT/2000/XP/Server 2003, Windows 95/98/ME
- Linux (on Intel)
- Linux (on 390)
- AIX
- HP-UX
- Sun Solaris

Key capabilities and benefits

We now consider the main capabilities and benefits of DB2 UDB.

Superior scalability

One of the major benefits of DB2 UDB is scalability. It can run on a laptop supporting a mobile worker, or it can run on a massively parallel machine supporting multiple terabytes of data and thousands of users — and, of course, it can support various configurations of SMPs (symmetric multiprocessors) and clusters of SMPs in between.

It is important to note that this scalability includes the same object-relational function up and down the line. This makes DB2 UDB the perfect database platform for small to medium sized businesses that are growing; for large enterprises that need to deploy 2-tier or 3-tier applications from desktop to department to enterprise; and for ISVs and Business Partners who are servicing all of these customers.

You will probably not outgrow DB2 UDB as your applications and businesses grow. You will be able to add data and still maintain query performance; or add users and still maintain response time; or just add hardware to speed up your queries.

DB2 UDB's scalability features can be grouped into three major capabilities:

- Advanced parallel processing
- High-performance computing
- Efficient large database operations

Advanced parallel processing

DB2 Universal Database uses parallel processing to speed up both transaction processing (OLTP) and query processing (OLAP, data warehousing, data mining) — or mixed workloads involving both. It supports both parallel transactions and parallel query on all major hardware architectures, including symmetric multiprocessors (SMP), clusters, and massively parallel processors (MPP).

- **SMP parallel support:**

  UDB will automatically execute multiple transactions (SQL statements) in parallel by dispatching them to the multiple processors in an SMP machine. UDB can also automatically execute a single query (SQL statement) in parallel by breaking the query into subtasks and dispatching each subtask to a different processor. Further, if the data for an SQL statement is spread across multiple disk locations, UDB can also use parallel I/O to execute the data retrieval in parallel.

- **Cluster and MPP parallel support:**

  When the Database Partitioning feature is used, a UDB database can be spread across multiple servers in a cluster or multiple nodes on an MPP. UDB will automatically execute multiple transactions (SQL statements) in parallel by dispatching them to the multiple nodes. UDB can also automatically execute a single query (SQL statement) in parallel by breaking the query into subtasks and dispatching each subtask to a different node.

  The key to efficient parallel processing across nodes is intelligent partitioning and parallel optimization. UDB will automatically spread (partition) the data across the nodes such that the optimizer will “know” where each row is located and will be able to dispatch the processing to the node where the data is located — minimizing the movement of data between nodes. This is known as a shared-nothing architecture and it is the best approach for ensuring scalability across multiple nodes, particularly when very large databases of 100's of gigabytes or multiple terabytes are involved.

High-performance computing

In addition to advanced parallel processing, UDB supports a number of important high-performance computing features that significantly enhance performance of both queries and transactions. The combination of these and other features make DB2 UDB particularly good at handling mixed workloads. These high-performance features include:

- **64-Bit memory support**

  Support now exists for very large physical memory (64-bit memory support). DB2 exploits 64-bit systems and 32-bit systems capable of supporting greater than 4GB of real memory. The buffer pool can be used to keep data available in this additional memory, thus reducing I/O and significantly improving performance.

- **Multiple buffer pools**

  You can create multiple buffer pools of various sizes (number of pages) and assign table spaces to them using the CREATE BUFFERPOOL SQL statement. This provides the database administrator greater control of the availability of data in memory. For example, operational transaction tables could be given their own bufferpool so they would not have to compete with ad hoc queries for memory. This could help maintain OLTP performance while still providing good response to end-user queries.

- **Asynchronous page cleaner**

  The ability to offload buffer page writing operations from a task executing the SQL query to another task can significantly improve query response time. The asynchronous page cleaning tasks ensure that sufficient space exists in the database buffers for query data. This capability can eliminate the need for query to wait synchronously until modified pages are written from the buffer to disk in order to make room for query data.

- **Sequential prefetch**

  The performance of queries, which require a large amount of sequential disk I/O is significantly improved due to the overlapping of I/O and CPU processing. This is particularly useful for query processing.

- **List prefetch**

  The support for list prefetch improves the performance of queries that access data randomly or non-sequentially.

- **Media-spanning through table spaces**

  With UDB, database administrators can partition a database into parts called table spaces. When creating a table, the name of the base, index, and long table space can be specified. Using index and long table spaces to store indexes and LOB data respectively allows these structures to be kept separate from the rest of table data. This flexibility can be used to increase database performance and availability. A table space can be spread over one or more physical storage device, providing the media-spanning capability.

- **Direct media access (raw device support)**

  UDB has the ability to store data directly on a device, without incurring the overhead of using a file system, for improved performance and data integrity. The administrator can choose to define a table space that uses devices, or uses the native file system.

- **Big block reads**

  This capability enables the reading of several disk pages using a single I/O operation, reducing the CPU usage and, in the process, improving query response time.

- **REORG utility**

  UDB includes a REORG utility that will resequence rows in a clustered order and eliminate fragmenting due to updates and deletes. This can significantly enhance performance of tables with a high rate of change. A REORGCHK utility is also provided to determine if tables are in need of reorganization.

- **Connection concentrator**

  The connection concentrator allows a physical database agent to be used by multiple logical agents, each representing a client connection. The restriction on the number of client connections is now solely based on the transaction load and the machine's ability to handle the workload. The connection concentrator uses scheduler technology to run individual transactions for multiple clients over the same physical database connection.

- **Multidimensional clustering (MDC)**

  Multidimensional clustering (MDC) provides an elegant method for flexible, continuous, and automatic clustering of data along multiple dimensions. This results in significant improvement in the performance of queries, as well as significant reduction in the overhead of data maintenance operations, such as reorganization, and index maintenance operations during insert, update, and delete operations. Multidimensional clustering is primarily intended for data warehousing and large database environments, and it can also be used in online transaction processing (OLTP) environments.

  MDC enables a table to be physically clustered on more than one key (or dimension) simultaneously. Prior to the introduction of MDC, DB2 only supported single-dimensional clustering of data, through clustering indexes. Using a clustering index, DB2 maintains the physical order of data on pages in the key order of the index, as records are inserted and updated in the table. Clustering indexes greatly improve the performance of range queries that have predicates containing one or more keys of the clustering index. With good clustering, only a portion of the table needs to be accessed and, when the pages are sequential, more efficient prefetching can be performed.

  With MDC, these benefits are extended to more than one dimension, or clustering key. In terms of query performance, range queries involving any combination of specified dimensions of the table will benefit from clustering. Not only will these queries access only those pages having records with the correct dimension values, these qualifying pages will be grouped by extents. Furthermore, although a table with a clustering index can become unclustered over time as space fills up in the table, an MDC table is able to maintain its clustering over all dimensions automatically and continuously, thus eliminating the need to reorganize the table to restore the physical order of the data.

Efficient large database operations

Scalability requires much more than performing transactions and queries faster using parallel processing or other high-performance features. It also requires that maintenance operations like data loading, backup and recovery, and indexing are scalable such that large data volumes can be efficiently managed. DB2 UDB provides the following in support of large database operations:

- **High speed LOAD utility:** DB2 UDB includes a high-speed LOAD utility that significantly increases the speed of doing data loads while ensuring recoverability of the data being loaded. It can take input from a number of file formats or an SQL SELECT statement and directly build table space pages without the overhead of logging or SQL INSERT processing. It can also build indexes and collect statistics during the load. The LOAD utility can significantly reduce the amount of time it may take refresh or add data to a data warehouse.
- **SMP support in utilities:** DB2 UDB not only exploits SMP architectures in transaction and query processing, but in its utility operations as well:
  - **Parallel data loading:** Loading data is a heavily CPU-intensive task. The LOAD utility takes advantage of multiple processors for tasks such as parsing and formatting data. Also, the LOAD utility can use parallel I/O servers to write the data to the containers in parallel. The LOAD utility has been shown to scale linearly as the number of processors in increased.
  - **Parallel index creation:** DB2 exploits I/O parallelism and CPU parallelism when creating an index. This helps to speed up index creation when a CREATE INDEX command is issued during restart (if an index is marked invalid) and during REORG processing.
  - **Parallel backup and restore:** Backing up and restoring data are heavy I/O-bound tasks. DB2 exploits I/O parallelism and CPU parallelism when performing backups and restores. DB2 takes advantage of multiple CPUs by assigning buffer manipulators among the CPUs. You can also backup to or restore multiple devices (e.g. tapes) in parallel.
- **Cluster/MPP support in utilities:** DB2 Universal Database with the Database Partitioning Feature can also exploit cluster and MPP architectures in its utility operations:
  - **Parallel split:** As mentioned above, the key to MPP parallelism is intelligent partitioning. The db2split utility is provided to split data into partitions prior to loading. It can run in parallel on multiple nodes thus significantly speeding up the overall load process.
  - **Parallel data loading:** Once the data is split, the LOAD utility can run in parallel on each node. If each node is an SMP, it takes advantage of multiple processors for tasks such as parsing and formatting data., and can use parallel I/O servers to write the data to the containers in parallel.

- **Autoloader:** This simplifies the process of splitting and loading multiple partitions of a table.
- **Parallel index creation:** The CREATE INDEX command runs in parallel on each node.
- **Parallel backup and restore:** Backup and restore operations can be run in parallel across nodes. DB2 exploits I/O parallelism and CPU parallelism when performing backups and restores. DB2 takes advantage of multiple CPUs by assigning buffer manipulators among the CPUs.

These advanced parallel processing, high-performance computing, and efficient large database operation features make DB2 Universal Database the most scalable, object-relational, and Web-enabled database in the industry. If you start out on a small NT server, it's easy to upgrade your hardware or move your data to a larger UNIX server if you need more capacity. If you're building a large data warehouse, you can start out on a single node and easily add additional nodes to support more data, more users or improved response times. Whether your needs today are large or small, you can rest assured that you won't get boxed in if you grow.

Multimedia extensibility

Another major benefit of DB2 UDB is extensibility. By extensibility, we mean the ability to store and manage not only traditional relational tables with characters and numbers — but also multimedia, complex objects such as documents, images, audio, video, spatial information, time-series data, etc. This may also include industry-specific objects such as x-rays, fingerprints, engineering drawings, insurance claim forms, etc.

The key to this capability is what is called object-relational technology. DB2 UDB's object-relational capabilities allow you to add your own data types and business functions to the database — effectively tailoring the database to fit your specific business or application requirements. IBM was first to deliver this capability in DB2 Common Server Version 2 in 1995. DB2 UDB's implementation is more advanced and more robust than any other “Universal Server” on the market and it has been thoroughly field tested making it very reliable.

DB2 UDB's extensibility features can be grouped into the following categories:

- Universal Data
- Business Rules
- Advanced SQL
- DB2 Extenders

These extended object-relational capabilities open up a whole new world of potential applications for you — and a very productive and efficient way to deliver them.

Universal Database (object-relational)

DB2 Universal Database supports the following key object-relational features. These are implemented according to SQL3 standards so they represent an open, non-proprietary approach.

- **User-Defined Types (UDTs):** UDTs allow users to define new data types, which are represented in the database using built-in types. For example, a user can define two currency types CDOLLAR for Canadian Dollars and USDOLLAR for U.S. Dollars. These types are distinct in the sense that they should not be directly compared to one another or to the decimal type, although the decimal type might be chosen for the internal representation of both UDTs in the database. UDTs, like built-in types, can be used for columns of tables as well as parameters of functions, including User-Defined Functions (UDFs). For example, a user can define a data type such as ANGLE (which varies between 1 and 360) and a set of UDFs to act on it, such as SINE, COSINE, and TANGENT.
- **User-Defined Functions (UDFs):** UDFs allow queries to contain powerful computation and search predicates to filter irrelevant data close to the source of the data. UDFs enable users to provide a set of functions that interpret a UDT, thereby defining the semantics of the UDT. Support for UDFs enable the creation of function libraries which can be developed by IBM, third party vendors, or customers and then integrated directly in the database. The SQL optimizer takes into account the semantics and execution cost of UDFs, thus treating User Defined Functions exactly as the built-in functions such as SUBSTR and LENGTH. Applications may be developed using different application language environments, such as C, C++, COBOL and FORTRAN, while sharing a set of UDTs and UDFs via SQL.
- **Large Objects (LOBs):** LOBs allow users to store very large (multi-gigabyte) binary or text objects in a database. Binary LOBs can be used to store multimedia objects such as documents, video, images, and voice. LOBs can also be used to store small structures whose semantics are defined by UDTs and UDFs. A powerful set of built-in functions, such as search, substring and concatenation, are supported for LOBs. Additional functions can be defined at any time by means of UDFs. More than one LOB column can be defined on a table.
- **User-Defined Table Functions (Table UDFs):** SQL users can now access data not stored in the relational format and make full use of the query capabilities of the relational database. It is often difficult, if not impossible, to subject data from non-relational data stores to relational operations. User-defined table functions are an extension to SQL that solves this problem. A table function is an external user-defined function that constructs a derived table. The program for the function can access data from the various sources and format it into a tabular form returned from the table function. Once the table function is written, it can be used in the FROM clause

of queries. Table functions can be used not only to subject this external data to the power of SQL, but also to capture external data permanently into relational tables.

- **OLE User-Defined Functions (UDFs):** OLE (Object Linking and Embedding) automation is part of Microsoft's architecture on Windows platforms. It enables applications (OLE automation servers) to expose objects and methods that can then be accessed by other applications (OLE automation controllers) including: Lotus SmartSuite (R), Lotus Approach, MS Office/Backoffice, SAP R/3, QMF (TM) HPO/Shuttle, many Web tools, many Visual Basic applications, and others. DB2 provides OLE automation controller support for accessing OLE automation server data through UDFs. These external UDFs allow data from OLE automation servers to be returned to SQL queries through DB2.

Business rules

Business rules enable the definition of complex integrity rules which can be used to enforce the correctness of a database. They enhance the power of the other object-oriented features. They augment object-code-only libraries (whose methods cannot be modified) to support additional specific object attributes and constraint checking. They also help to enforce inter-object data integrity rules. The key features supporting business rules include:

- **Defaults:** Defaults allow you to set default values for columns that are not assigned values explicitly in INSERT statements.
- **Generated Columns:** Generated columns let you specify an SQL function that will be used to calculate a value for a column that is not assigned a value in an INSERT statement. This might be used to generate a unique number for each new row, or to assign a tax code based on other columns such as salary or marital status.
- **Check Constraints:** Check constraints are generally used to enforce business rules not covered by key uniqueness or referential integrity constraints. For example, a user may define constraints on the EMPLOYEE table which specify that the job of an employee can only be one of 'Sales', 'Mgr', or 'Clerk', and that every employee that has been with the company for more than 8 years must have a salary of more than $40,000.
- **Referential Integrity:** Referential integrity (RI) lets you define required relationships between and within tables. Referential constraints are declared when a table is defined, and ensure the consistency of data values between related columns in different tables. DB2 maintains these relationships, so you do not need to program this function into applications.
- **Triggers:** Triggers may be used to maintain complex cross-table business rules, automatically generate a value for a newly inserted row, read from other tables for cross-referencing purposes, write to other tables for audit-trail

purposes and/or provide an alert capability, by adding a trigger which invokes a User-Defined Function (for example, to send an electronic mail message). A user might define a trigger that increments the number of employees each time a new row is inserted into the EMPLOYEE table.

- **Stored Procedures:** DB2 clients can call stored procedures that execute on the DB2 server. Stored procedures push processing back from the client to the server and thus provide support for thinner clients and for execution of the logic at the server-side. They also cut back significantly on network traffic, resulting in better response time. DB2 UDB stored procedures are somewhat unique in that they are not written in some proprietary language, but can be written in Java, C, C++, etc. They can also be written in the stored procedure language defined by the database standards committee. Stored procedures can return multiple rows and multiple result sets for greater efficiency.

Advanced SQL

Advanced SQL queries allow you to encapsulate a great deal of processing logic into a single, non-procedural statement. Examples of this include:

- **Recursive SQL queries:** Changes to the SQL optimizer will enable DB2 server databases to support not only the bills-of-material queries, but also the more powerful form of recursive queries such as path expression queries. Examples of queries that become possible with support for recursion are:
  - Bills-of-material queries where a user wants to return subparts of parts, and subparts of the subparts, and so on.
  - Path expression queries, where a user wants to calculate the lowest cost plane fares on multi-hop routes. For example, the following query can be formulated using recursive SQL: Return all possible flights from Toronto to Perth without making a stopover in London or Chicago and with no more than 3 plane changes.
  - The optimizer is capable of making sophisticated transformations and optimizations for recursive queries and non-recursive queries, resulting in selection of better access plans for improved performance.
- **Outer join support:** Left, right, and full outer join operations are supported using standard SQL92 syntax; that is, a join operation whose result includes unmatched rows in addition to matching rows.

DB2 Extenders

The DB2 Extenders build on the object-relational infrastructure of DB2. Each extender is a package of predefined UDTs, UDFs, triggers, constraints, and stored procedures that satisfies a specific application domain. With the Extenders, the user can store text documents, images, videos, and audio clips in DB2 tables by adding columns of the new data types provided by the Extenders. The actual data can be stored inside the table or outside in external files.

These new data types also have attributes that describe aspects of their internal structures, such as “language” and “format” for text data. Each extender provides the appropriate functions for creating, updating, deleting, and searching through data stored in its data types. The user can now include these new data types and functions in SQL statements for integrated content searching across all types of data.

The following four DB2 Extenders are provided as part of the DB2 Developer's Editions.

DB2 Text Extender

Text Extender adds the power of full-text retrieval to SQL queries by making use of features available in DB2 that let you store unstructured text documents of up to 2GB in databases. Text Extender offers DB2 users and application programmers a fast, versatile, and intelligent method of searching through such text documents. Text Extender's strength lies in its ability to search through many thousands of large text documents at high speed, finding not only what you directly ask for, but also word variations and synonyms. You are not restricted to searching only in text documents stored in DB2 databases, you can also search in text documents stored in files.

Text Extender can access various kinds of text documents, including word-processing documents in their original native form, and offers a rich set of retrieval capabilities including word, phrase, wild card, and proximity searching using Boolean logic (“and/or” logic).

At the heart of Text Extender is IBM's high-performance, advanced search engine. It allows your applications to access and retrieve text documents in a variety of ways. Your applications can:

- Search for documents that contain specific text, synonyms of a word or phrase, or sought-for words in proximity, such as in the same sentence or paragraph.
- Do wild card searches, using front, middle, and end masking for word and character masking.
- Search for documents of various languages in various document formats.
- Make a fuzzy search for words having a similar spelling as the search term. This is useful for finding words even when they are misspelled.
- Make a free-text search in which the search argument is expressed in natural language.
- Search for the names of people, places, or organizations.
- Search for words that sound like the search term.

You can integrate your text search with business data queries. For example, you can code an SQL query in an application to search for text documents that are created by a specific author, within a range of dates, and that contain a particular word or phrase. Using the Text Extender programming interface, you can also allow your application users to browse the documents. By integrating full-text search into DB2's SELECT queries, you have a powerful retrieval function that combines attribute and full-text search.

DB2 Image Extender

The DB2 Image Extender defines a data type and functions for images using DB2 UDB's built-in support for user-defined types and user-defined functions. It also exploits DB2 UDB's support for large objects of up to 2GB and uses DB2 UDB triggers to automatically store and maintain attribute information for images. With the DB2 Image Extender, you can:

- **Import and export images:** You can import and export images and image attributes into and out of a database. When an image is imported, the DB2 Image Extender stores and maintains image attributes such as size in bytes, format, height, width, and number of colors.
- **Control access to images:** This can be done with the same level of protection as traditional business data.
- **Convert the format of images:** You have the flexibility of importing or exporting an image in its source format or converting the format of the image when importing or exporting. You can also convert an image in other ways, such as rotating it or changing its scale.
- **Secure and recover images:** Images and their attributes stored in a DB2 database are afforded the same security and recovery protection as traditional business data.
- **Query images:** This can be done based on related business data or by image attributes. Images can be searched based on data that the user maintains, such as a name, number, or description; or by data that the DB2 Image Extender maintains, such as the format of the image or its distribution of colors.
- **Generate and display image thumbnails and full images:** A thumbnail is a miniature version of an image. When an image is imported into a database, the DB2 Image Extender creates and stores a thumbnail of the image. You can use the DB2 Image Extender to retrieve a thumbnail or a full-size image, and then use the DB2 Image Extender to invoke a favorite browser to display the thumbnail or full-size image.

The DB2 Image Extender supports a wide variety of image formats, such as GIF (including animated GIFs), JPEG, BMP, and TIFF.

DB2 Audio Extender

The DB2 Audio Extender defines a new data type and functions for audio using DB2 UDB's built-in support for user-defined types and user-defined functions. It also exploits DB2 UDB's support for large objects of up to 2GB and uses DB2 UDB triggers to automatically store and maintain attribute information for audio objects.

With the DB2 Audio Extender, you can:

- **Import and export audio clips:** You can import and export audio clips and audio attributes into and out of a database. When an audio clip is imported, the DB2 Audio Extender stores and maintains audio attributes such as number of audio channels, transfer time, and sampling rate.
- **Secure and recover audio data:** Audio clips and their attributes stored in a DB2 database are afforded the same security and recovery protection as traditional business data.
- **Query audio clips:** This can be done based on related business data or by audio attributes. You can search for audio clips based on data that you maintain, such as a name, number, or description; or by data that the DB2 Audio Extender maintains, such as the format of the audio or the date and time that it was last updated.
- **Play audio clips:** You can use the DB2 Audio Extender to retrieve an audio clip, and then use the DB2 Audio Extender to invoke a favorite audio browser to play the audio clip.

The DB2 Audio Extender supports a variety of audio file formats, such as WAVE, MIDI, MPEG1, and AU, and can work with different file-based audio servers.

DB2 Video Extender

The DB2 Video Extender defines a new data type and functions for video using DB2 UDB's built-in support for user-defined types and user-defined functions. It also exploits DB2 UDB's support for large objects of up to 2GB and uses DB2 UDB triggers to automatically store and maintain attribute information for video objects.

With the DB2 Video Extender, users can:

- **Import and export video clips:** You can import and export video clips and their attributes into and out of a database. When a video clip is imported, the DB2 Video Extender stores and maintains video attributes such as frame rate, compression format, and number of video tracks.
- **Secure and recover video data:** Video clips and their attributes stored in a DB2 database are afforded the same security and recovery protection as traditional business data.

- **Query video clips:** This can be done based on related business data or by audio attributes. You can search for video clips based on data that you maintain, such as a name, number, or description; or by data that the DB2 Video Extender maintains, such as the format of the video or the date and time that it was last updated.
- **Build storyboards:** You can use the DB2 Video Extender to automatically segment a video clip into shots based on scene changes. A scene change is a point in a video clip where there is a significant difference between two successive frames. When the DB2 Video Extender detects a scene change, it records data for the associated shot, including representative frames. This makes it easy to build summaries of a video, called storyboards.
- **Play video clips:** You can use the DB2 Video Extender to retrieve a video clip, and then use the DB2 Video Extender to invoke a favorite video browser to play the video clip.

The DB2 Video Extender supports a variety of video file formats, such as MPEG1, MPEG2, AVI, and QuickTime, and can work with different file-based video servers.

DB2 XML Extender

The IBM DB2 XML Extender implements the IBM overall XML technical strategy. That is, to be the world leader in e-business. The XML Extender as a part of the IBM business to business(B2B) server offering, makes the DB2 server to be XML enabled. It provides new technology for emerging market of early XML adoption and evaluation. The Extender offers the capability of XML storage and data interchange. By storage, the XML Extender provides mechanism of storing and retrieving XML documents in DB2, and searching the content of XML with high performance. By data interchange, the XML Extender provides mapping between new and existing DB2 relational tables and XML formatted documents. Together, the XML Extender allows DB2 customers to do e-business anywhere, enabling XML with their B2B and business to consumer(B2C) applications.

With the content of your structured XML documents in the DB2 database, you can combine structured XML information with your traditional relational data. Based on the application, you can choose whether to store entire XML documents in DB2 as a nontraditional distinct data type, or you want to map the XML content as traditional data in relational tables. You can decide how structured XML documents be stored or created through the Document Access Definition(DAD).

For non-traditional XML data types, the XML Extender adds the power to search rich data types of XML element or attribute values, in addition to the structural-text search provided by the Text Extender. For traditional SQL data which are either decomposed from incoming XML documents, or in existing

relational tables to be used to create outgoing XML documents, the XML extender provides a custom mapping mechanism to allow the transformation between XML documents and relational data.

Net.Data, which connects Web applications to DB2, now has built in XML exploitation. This allows you to generate XML tags as output from your Net.Data macro, instead of manually entering the tags. You can also specify an XML style sheet (XSL) to be used to format and display the generated Output.

DB2 Spatial Extender

DB2 supports the notion of Spatially Enabled SQL Queries. You can now integrate spatial data (locations via coordinates) with normal SQL data. The combination of these two technologies gives users access to new types of queries that they could not previous run. The DB2 Spatial Extender is used to achieve this functionality. The Spatial extender will store and index the spatial data (coordinate information) and allow specific spatial queries against this data.

The following features are available with the DB2 UDB Spatial Extender:

- Infrastructure to model and manipulate spatial data: spatial types, catalog views, spatial functions, support for spatial reference systems and spatial “layers”
- Object types and operations that conform with OpenGIS Consortium (OGC) and ISO SQLMM standards
- A multi-level grid index implementation modelled after the ESRI proprietary grid index technology
- Stored procedure API to create and manage spatial resources
- Geocoding is performed either incrementally or in batch mode. The Spatial Extender supports not only a default geocoder (shipped with the product), but also geocoders supplied by users and vendors.
- Spatial Extender supports DB2 UDB spatial data. But you can use DB2 UDB to create joins between this data and data outside of DB2 UDB. You can do this if you install Spatial Extender on the server that includes a federated database, and if you then enable this database for spatial operations. Once this database is so enabled, you can create joins between its spatial data and data in federated data sources. Currently, federated data sources are relational only (Oracle and DB2 family)
- Superior ease of use and ease of administration capabilities relative to other relational implementations
- Import, Export, and Geocode operations
- A sample application to help customers to set up and use Spatial Extender

DB2 Net.Search

The DB2 Net Search Extender contains a DB2 stored procedure that adds the power of fast full-text retrieval to DB2 applications. It offers application programmers a variety of search functions, such as fuzzy search, stemming, Boolean operators, and section search. Searching using DB2 Net.Search Extender can be particularly advantageous in the Internet, when search performance on large indexes and scalability according to concurrent queries are important factors.

The Net.Search extender in DB2 UDB allows very fast text search with:

- Word or phrase searches
- Stemmed searches
- Boolean operations
- Fuzzy searches
- Wildcard searches
- Fielded searches
- Presorted searches
- Cursor capability
- High speed indexing
- Creation of multiple indexes at the same time
- Only read access to the user data is required

Net.Search uses main memory database technology to reach high performance levels. Load tests resulted in 90 million Web site hits per day without degradation in the search performance.

DB2 Universal Database's support for universal data, business rules, advanced SQL and multimedia data objects and methods makes it among the most extensible object-relational database in the industry. Among the key differentiators is that this is built according to SQL standards and that it is built into the optimizer. This makes the implementation more open and allows for better performance than other implementations.

Web enablement

Customers need to be able to make the data stored in their DB2 database systems accessible to employees of the company, and selectively to their suppliers and customers through private network (intranet) and public network (Internet) applications. DB2 Universal Database is fully integrated with Web technology so that data can be easily accessed from the Internet or from your intranet with complete security. The following facilities included with UDB allow you to Web-enable your database applications right out of the box:

- DB2 Java Support
- Net.Data — A Webserver Database Gateway

DB2 Java support

Java applications are very attractive to customers who would like to develop a single application running on any operating system and who would like to reduce the cost of application distribution and maintenance.

Java Database Connectivity (JDBC) is a database access interface for Java applications that is being delivered with DB2. The DB2 JDBC database interface supports this API.

DB2 provides native support for Java at client workstations and DB2 servers. Java is supported on the client workstations in three ways:

- A Java application uses the DB2 Client Application Enabler, which must be installed on the client workstation, to communicate with the DB2 server. Customers with existing DB2 client/server configurations can now use Java as a database application development tool.
- Java applets allow applications to be developed that access DB2 servers without requiring DB2 Client Application Enabler code to be installed on client workstations. Java applets can be automatically downloaded to the client workstation at application invocation time.
- The new type 4 JDBC driver is a two-tier pure Java JDBC driver, which allows a Java client to communicate directly with DB2 servers via DRDA protocol. This driver is designed to replace the type 3 driver. You should migrate applets that use the type 3 JDBC driver to the type 4 driver, in preparation for the end of type 3 driver support.

Java support in DB2 servers consists of the ability to create native Java-based, user-defined functions and stored procedures.

Net.Data

Included in the server product boxes is a copy of Net.Data, a Web-enabling tool for interactive database-to-Web applications. Net.Data is IBM's strategic product for enabling Internet/intranet access to relational data on a variety of platforms. It provides open data access to DB2 and other data sources including Oracle, Sybase and any ODBC data server. With Net.Data, your application can use DB2 to build dynamic Web pages to support electronic commerce applications.

Net.Data provides for high-performance, robust, application development function and exploitation of existing business logic through open programming interfaces. Net.Data tightly integrates with Web-server APIs such as those from IBM, Lotus, Netscape, and Microsoft, providing higher performance than common gateway interface (CGI) applications.

Net.Data provides connection management to your key relational databases for optimum performance. Net.Data can establish a continuous connection to specified databases. Net.Data maintains the connection throughout the Web application and across invocations of Web applications. Since the database connection is continuous, the application does not experience the overhead of repeated connects to the database. The result is peak performance of your interactive Web application.

The Net.Data application is a macro with a rich macro language, variable substitution, conditional logic, and optional function calls. Net.Data supports client-side processing with Java applets and Java Script. Server-side processing includes Java applications and REXX, Perl, and C/C++ applications.

Net.Data and Java support make DB2 Universal Database internet ready right out of the box. For the most complete Web-enabled database solution on the market, DB2 UDB teams with WebSphere for a complete e-business environment.

Partner solutions

In order to implement a data management solution that meets the changing needs of your business, you'll need a robust relational database, and carefully chosen applications to run against it. DB2 Universal Database provides the rock-solid foundation required to successfully deploy enterprise solutions, while thousands of ISVs (Independent Software Vendors) offer a diverse range of sophisticated applications that support DB2.

In close partnership with IBM, leading software developers like Siebel, I2, SAP, Baan, Dassault, and PeopleSoft, develop enterprise resource planning (ERP) solutions and CRM solutions as well as B2B solutions and other applications that deliver the functionality your user base needs. These premier ISVs recognize that DB2 Universal Database has the performance, reliability, and flexibility to meet the demands of today's operational and business intelligence applications.

So, if you're looking for a data management solution, IBM's many business partners offer an extraordinary range of applications designed to run on DB2. From the ERP solutions mentioned above, to the business intelligence tools offered by Andyne, Brio, Business Objects, and Cognos, to the DB2 Extenders created by Consistency Point Technologies, Environmental Systems Research Institute, Inc. (ESRI), The Fillmore Group, Formida Software, Innovus Corp., IsoQuest, Inc., OG Software (Oxford Group), and Prime Factors, Inc., you'll find that our business partners offer a constantly expanding universe of solutions.

For the most up-to-date information on solutions and to look through the on-line IBM Solutions Catalog, go to these two sites:

http://www.software.ibm.com/data/partners/
http://www.software.ibm.com/solutions/isv

Business intelligence powerhouse

By Business Intelligence (BI) we mean applications like data warehousing, data mining, on-line analytical process (OLAP), and decision support. Many customers are looking for ways to mine and analyze their operational data for competitive advantage. These are among the most important uses of data management technology today because they provide customers with excellent returns on their investment.

Several factors make DB2 UDB an outstanding data store for BI applications:

- These applications often involve large volumes of data. Typical small to medium size warehouses and datamarts might contain 50 GB to 300 GB. Large installations can contain several terabytes. UDB addresses these requirements with its outstanding scalability, including advanced parallel query and VLDB operations.
- Queries against the database tend to be very complex. UDB has the most advanced query optimizer in the industry, providing excellent query performance with a minimum of DBA time required for tuning.
- DB2 UDB has some features specifically designed to assist with on-line analytical processing (OLAP) support.
- DB2 also offers the Warehouse Manager feature, an integrated set of tools for building, managing, and maintaining data warehouses and datamarts. The Warehouse Manager tool works with the Warehouse Center, the integrated GUI tool which co-ordinates and automates the activities needed to extract, clean, and populate the data into your informational data store.

Advanced optimizer

The DB2 SQL optimizer strengthens DB2's leadership position in query optimization for traditional applications and also extends this robust support to new application domains enabled by the object-oriented extensions described earlier. This new optimization technology provides the function and performance needed by customers to analyze and exploit vast amount of valuable information stored in their databases.

The increasing importance of decision support applications represents an important customer trend, which can result in very complex database queries. These queries may have been written by end-users, generated by automatic tools, or produced as a result of many point-and-click application interfaces popular in today's DOS, Windows, OS/2, and UNIX environments.

The optimizer incorporates a very sophisticated query rewrite phase that automatically transforms a complex query into a simpler query that is easy to optimize. As a result, the end user will realize the best possible performance regardless of the way a query is structured.

The optimizer also looks at a greater number of alternatives in its search for the best query execution plan, as well as employing more sophisticated techniques of modeling the cost of different ways of fetching the data from disk. All these techniques can result in orders of magnitude of performance improvement for complex queries as compared to existing SQL optimizer technology.

A related technology is DB2's support of Materialized Query Tables (MQT) or Automated Summary Tables (AST). Using AST queries that might take minutes or hours to complete can be shortened dramatically, often to seconds or even sub-second response times. This is done by precalculating summary information into a summary table, then using the power of optimizer's query rewriting to change the submitted query so that it retrieves the information from the summary table, rather than recalculating it. Users don't have to change their queries to take advantage of this performance improvement, it's handled automatically by the DB2 optimizer once the administrator defines the AST.

OLAP support

Data warehouse and OLAP applications are characterized by the use of a special design technique, which is called star schema, to model relational data for multidimensional analysis (MDA). Under this schema it is common for a large “fact” table to be joined to multiple “dimension” tables in a very complex SQL join query. Any optimization techniques that can be used to improve the performance of these joins can significantly improve the performance of the overall application. DB2 Universal Database has special optimization techniques to do this:

- **Index ANDing using dynamic bit maps:** UDB uses dynamic bit-map technology to efficiently combine multiple indexes. Performance is improved for queries that use columns that are key columns of different indexes over the same table. This includes the use of indexes for multiway joins.
- **Star joins:** DB2's star join algorithm exploits dynamic bit maps to join a large fact table with a series of relatively small dimension tables, thus minimizing data I/O.

OLAP and MDA applications are also characterized by queries involving complex grouping and aggregation of data. With DB2 UDB the GROUP BY clause has been extended to support “super groups”. One type of super group is a ROLLUP group, a result set that contains “sub-total” and “overall total” rows in addition to the regular grouped rows. Another type of super group is a CUBE group, a result set that contains “cross-tabulation” rows in addition to all the rows that would be in a ROLLUP group for the same columns.

The ROLLUP and CUBE aggregations are useful in OLAP (on-line analytical processing) decision support applications to aggregate base data in multiple dimensions and over roll-up/drill-down hierarchies. An example of a common usage of this is analyzing data in three dimensions, such as looking at sales by product, by store, and by month.

DB2 Universal Database's scalability and parallel query performance, advanced query optimization and unique OLAP support make it the best database engine to power any Business Intelligence application.

DB2 UDB has particular strengths in supporting business intelligence applications such as data warehousing and on-line analytical processing (OLAP). DB2 leads the industry in parallel database technology and query optimization resulting in the proven ability to help customers find competitive advantage, better customer service or reduced costs by mining their data for the knowledge required to make better decisions. Furthermore, this does not require the additional expense of a specialized database; DB2 UDB provides a single database that can be used across an enterprise for all data management requirements from OLAP to OLTP.

Ease of use and management

DB2 Universal Database is one of the easiest databases in the industry to use and manage. It includes a complete suite of graphical tools to satisfy the needs of:

- Database administrators (DBAs)
- Application programmers

It also includes tools to assist with client set-up and ad hoc query and reporting for end users.

Administering databases with the DB2 Administration Tools

You can administer local or remote servers using the DB2 Administration Tools. Use the Control Center to perform administration tasks such as configuring DB2 instances and databases, backing up and recovering data, scheduling jobs, and managing media, all from a graphical interface.

Managing instances and database objects using the Control Center

The Control Center displays instances and database objects (such as table spaces, tables, and packages) and their relationships to each other. Using the Control Center, you can manage local and remote servers from a single point of control.

From the Control Center, you can perform operations on database objects. You can:

- Create and drop a database
- Create, alter, and drop a table space or table
- Create, alter, and drop an index
- Back up and recover a database or table space
- Define replication sources and subscriptions to replicate data between systems
- Monitor resources and events on a server

You can also control DB2 instances by:

- Maintaining communication protocols
- Setting database manager and database configuration values that affect performance

SmartGuides or Wizards are provided to help you perform complex tasks. For example, SmartGuides are available to tune the performance of your system or to recommend what indexes you need (and it will build them for you too).

The Control Center provides additional functionality to assist you in managing your servers:

- **Control Center:** Use the Control Center to start another session of the Control Center to administer a server.
- **Satellite Center:** Use the Satellite Center to manage the Satellites that are served by a particular DB2 Control Server. It provides create, remove, modify, and manage functions for Satellites and Groups. You can also create and manage scripts to administer the Satellites.
- **Command Center:** Use the Command Center to enter DB2 commands and SQL statements in an interactive window and see the execution result in a result window. You can scroll through the results and save the output to a file.
- **Task Center:** Use the Task Center to create scripts, which you can store and invoke at a later time. These scripts can contain DB2 commands, SQL statements, as well as operating system commands. Scripts can be scheduled to run unattended. These jobs can be run once or set up to run on a repeating schedule; a repeating schedule is particularly useful for tasks like backup.
- **Health Center:** Use the Health Center to monitor your system for early warnings of potential problems or to automate actions to correct problems discovered.
- **Journal:** Use the Journal to view all available information about jobs that are pending execution, executing, or that have completed execution. You can also

view the recovery history log, the alerts log, and the messages log; and review the results of jobs that are run unattended.

- **Tools Setting:** Use the Tools Setting to change the settings for the DB2 Administration Tools.
- **License Center:** Use the License Center to manage licenses and display license status and usage of any DB2 products installed on your system. You can also use the License Center to configure your system for proper license monitoring.
- **Data Warehouse Center:** Use the Warehouse Center to manage data warehouses and datamarts. You can define data sources, cleansing steps, and target systems. You can set up a schedule for when data extractions are run on the data sources and when data will be propagated to the target warehouses.
- **Information Center:** The Information Center provides quick access to DB2 product information. This product information includes such items as: database tasks, reference material, DB2 documentation, troubleshooting aids, sample programs for application development, and DB2 Web-related URLs.
- **Development Center:** The Development Center provides an easy-to-use interface for developing routines such as stored procedures and user-defined functions (UDFs). A set of wizards makes it easy to perform your development tasks. The Development Center provides a single development environment that supports the entire DB2 family ranging from the workstation to z/OS. You can launch the Development Center as a stand-alone application from the IBM DB2 Universal Database program group or from a DB2 Universal Database center, such as the Control Center, the Command Center, or the Task Center.

  With the Development Center, you can:

  - Create, build, and deploy Java and SQL stored procedures.
  - Create, build, and deploy user-defined functions:
    - SQL table and scalar UDFs
    - UDFs that read MQSeries messages
    - UDFs that access OLE DB data sources
    - UDFs that extract data from XML documents
  - Debug SQL stored procedures using the integrated debugger.
  - See the contents of the server for each database connection that is in your project or that you have explicitly added to the Server View. You can also view and work with other database objects such as tables, triggers, and views.
  - Export and import routines and project information.

The Development Center also provides a DB2 development add-in for each of the following development environments:

- Microsoft® Visual C++
- Microsoft Visual Basic
- Microsoft Visual InterDev

With the add-ins, you can easily access the features of the Development Center and other DB2 centers from your Microsoft development environment, making it easy for you to develop and incorporate stored procedures and UDFs into your DB2 application development.

DB2 autonomic technology

IBM has put a lot of research and development effort into providing advanced automation functionality within DB2. The results of this effort is known as SMART ("Self-Managing And Resource Tuning") Technology. The purpose of SMART is to reduce the complexity of managing DB2 for inexperienced users, and to reduce the total cost of ownership by enabling many administrative and tuning functions to be carried out automatically by the DBMS.

These are some examples of SMART in DB2 UDB:

- **Configuration Advisor Wizard:** This provides an easy-to-use GUI tool to tune the performance of a database with just minimal input about the anticipated workload.
- **Design Advisor Wizard:** This provides recommendations on adding new indexes or removing existing indexes based on your input of workload characteristics.
- **DB2 Health Center:** This constantly monitors the overall health of a system. It can automatically diagnose and fix many potential problems. In addition, it is able to contact an administrator by E-mail or pager, either to notify them that corrective action has been taken, or because a problem has been identified which is deemed by the administrator to be too risky to attempt an automated fix, or which cannot be fully automated because a hard limit has been reached or the task requires some form of manual intervention.
- **Memory Tracker and Visualizer:** This shows all memory in all pools at a particular instant. It provides a graphical display of memory usage and also supplies information to the DB2 monitoring facilities for ongoing analysis.
- **Storage Management:** This is a tool available through the Control Center. From this tool you can access the Storage Management view that displays a snapshot of the storage for a particular database, database partition group, or table space. Statistical information can be captured periodically and displayed depending on the object chosen:

- For table spaces, information is displayed from the system catalogs and database monitor for tables, indexes and containers defined under the scope of the given table space.
- For databases or database partition groups, information is displayed for all the table spaces defined in the given database or database partition group.
- For databases, information is also collected for all the database partition groups within the database.

You can use the information displayed in this view to monitor various aspects of your storage, such as space usage for table spaces, data skew (database distribution) for database partition groups, and capture cluster ratio of indexes for database partition groups and table spaces. From the Storage Management view you can also set thresholds for data skew, space usage, and index cluster ratio. A warning or alarm flag will let you know if a target object exceeds a specified threshold.

Data access and replication

DB2 Universal Database is a distributed, fully networked RDBMS. It provides you with the flexibility of placing data anywhere in your network required for optimum service and productivity, and provides efficient means for clients to access that data over the network. Further, DB2 provides the most efficient and seamless integration of data on mainframe and midrange data servers in the industry allowing you to reduce costs and improve cycle-times by leveraging your current investments in data, hardware, software, and skills.

DB2 UDB's support for distributed data can be grouped into 3 categories:

- Client/server data access
- Data replication
- Host data integration through DRDA

Client/Server data access

DB2 UDB supports a number of features that provide for efficient access to UDB database servers from client workstations over a network:

- **Stored procedures:** DB2 clients can call stored procedures that execute on the DB2 server. Stored procedures push processing back from the client to the server and thus provide support for thinner clients. They also cut back significantly on network traffic resulting in better response time. UDB stored procedures are somewhat unique in that they are not written in some proprietary language, but can be written in Basic, Java, C, C++, etc. They can also be written in the standards based stored procedure language defined by the database standards committee. Stored procedures can return multiple rows and multiple result sets for greater efficiency.

- **Compound SQL:** DB2 client applications can batch multiple SQL statements into a single Compound SQL statement that is transmitted over the network and executed as one at the server. This can cut down on network traffic and is particularly good for mass inserts, updates or deletes over the network.
- **Row blocking:** DB2 UDB can reduce the number of network transmissions by buffering rows at both the client and server ends. The buffer sizes are tunable parameters.
- **2-phase commit:** DB2 clients can access and update multiple UDB servers in a single transaction (unit-of-work). UDB servers have built-in transaction monitors that log the transaction phases and handle rollback in case of failure. The 2-phase commit process is coordinated from the client.

Data replication

Data Propagator Relational is built into DB2 Universal Database (and remains available as separate products or features for DB2 products on MVS, OS/390, OS/400, and VSE and VM). This feature is based on an advanced change-capture technology. It provides a highly efficient way for automatically maintaining consistent copies of relational data in the DB2 family of databases without requiring any batch processing windows and explicit knowledge of, or changes to, business applications. Data Propagator includes:

- **Easier and intuitive administration:** The Control Center includes replication administration, provides many ease-of-use features, and acts as a single-point-of-control for the entire DB2 replication network.
- **Subscription sets for transaction consistency:** With subscription sets, updates to all related target tables are committed in a single unit of work, supporting referential integrity requirements.
- **Update-anywhere replication:** Robust, update-anywhere replication is enabled with rigorous conflict detection, and automatic compensation for a new breed of distributed applications.
- **On-demand replication for occasionally connected and mobile systems:** With support for on-demand replication, auto connect and disconnect, and minimization of connection time, Mobile or occasionally disconnected systems running DB2 Universal Database Personal Edition on Windows NT/2000/XP, or Windows 98/ME are enabled to participate fully and efficiently in read-only and update-anywhere replication scenarios.
- **DB2 source view replication:** With Support for view-based replication, you can subset and distribute data efficiently from normalized databases.
- **Event-driven propagation:** This is an easy mechanism for you to control the timing of propagation, for example, at the end of day or after the completion of a particular batch run; or the content of propagation only on updates performed before 6:00 p.m.

Distributed database access through DRDA

Many customers have a need to distribute data across relational databases from different vendors or on different hardware platforms. IBM has developed the Distributed Relational Database Architecture (DRDA) to support this. It has become the standard for integrating client/server databases and client-based tools with mainframe and midrange databases on OS/390, MVS, VSE, VM and OS/400. DB2 UDB uses DRDA to provide the most complete and efficient data integration with these platforms in the industry.

There are two major subsets of DRDA function: the client or requester side, and the server-side:

- **DRDA Application Requester (DRDA-AR):** This allows DB2 client workstations to transparently access data in DB2 for OS/390, DB2 for MVS, DB2 for OS/400 and/or DB2 for VSE & VM. The DRDA-AR is provided with DB2 Connect or is built into DB2 UDB Enterprise Server Edition.
- **DRDA Application Server (DRDA-AS)**: This allows other DRDA-ARs to access data stored in DB2 Universal Database. In particular, this capability allows DB2 for MVS, DB2 for VSE & VM (SQL/DS), and DB2/400 applications (or any other application that implements the DRDA Application Requester functionality) to access data located in the UDB server databases. With this capability, thousands of existing database applications running on the MVS, VM, and OS/400 platforms are able to access data stored in DB2 UDB databases. The DRDA-AS is included in DB2 UDB Workgroup Server Edition and Enterprise Server Edition.

Federated systems

A DB2 federated system is a special type of distributed database management system (DBMS). A federated system consists of a DB2 instance that operates as a server, a database that serves as the federated database, one or more data sources, and clients (users and applications) who access the database and data sources. With a federated system you can send distributed requests to multiple data sources within a single SQL statement. The power of a DB2 federated system is in its ability to:

- Join data from local tables and remote data sources, as if all the data are local.
- Take advantage of the data source processing strengths, by sending distributed requests to the data sources for processing.
- Compensate for SQL limitations at the data source by processing parts of a distributed request at the federated server.
- Write capability to perform INSERT, UPDATE, and DELETE actions on the data sources
- Ability to create remote tables on relational data sources.

Federated database systems provide the middleware functionality for outstanding information integration. Built into DB2 Enterprise Server Edition is the ability to federate relational data across IBM's family of databases, including the DB2 family and Informix IDS.

Multi-platform support
DB2 Universal Database is one of the most open database platforms available. It runs on the most popular UNIX and Intel server platforms including AIX, HP-UX, Solaris, Linux and Windows NT/2000/Server 2003. It supports all major industry standards relevant to distributed data so that it can be accessed using thousands of existing tools and applications, and can be easily managed within an open, network computing environment. These capabilities allow you to reduce costs and improve cycle-times by leveraging your current investments in data, hardware, software, and skills.

Bullet-proof reliability
DB2 Universal Database is setting the standard for quality and reliability in the client/server database industry. As more mission-critical applications are implemented on UNIX and Intel platforms, IBM's ability to bring mainframe-level reliability to this environment has become a major factor in choosing DB2. Better reliability and availability can reduce your costs, while scalability both within and across platforms can reduce the risk of dead-end projects.

1.2 DB2 UDB products on Windows

DB2 UDB for Windows consist of several different product offerings based on a single code base that provides you with a great deal of flexibility in licensing depending on your requirements. We describe the offerings below, with recommendations on which of the products to use under what circumstances later in the section Planning considerations.

1.2.1 Product descriptions

In this section we provide brief descriptions of the various DB2 products.

DB2 UDB Enterprise Server Edition

DB2 UDB Enterprise Server Edition (ESE) meets the database server needs of midsize to large businesses. It can be deployed in Linux, UNIX, and Windows server environments on any size server. ESE is the ideal foundation for building data warehouses, transaction processing, or Web-based solutions as well as a back end for packaged solutions like ERP, CRM, and SCM. Additionally, ESE offers connectivity and integration for other enterprise DB2 and Informix data sources.

DB2 Database Partitioning Feature

The DB2 Database Partitioning Feature (DPF) allows DB2 UDB Enterprise Server Edition customers to partition a database within a single system or across a cluster of systems. The DPF capability provides the customer with multiple benefits including scalability to support very large databases or complex workloads and increased parallelism for administration tasks.

DB2 UDB Workgroup Server Edition

DB2 UDB Workgroup Server Edition (WSE) is the database server designed for deployment in a departmental or small business environment that involves a small number of internal users. WSE uses a licensing model designed to provide an attractive price point for smaller installations while still providing a full function database server. WSE can be deployed in Linux, UNIX, and Windows server environments on systems with up to four CPUs.

DB2 UDB Workgroup Server Unlimited Edition

DB2 UDB Workgroup Server Unlimited Edition (WSUE) product offers a simplified per processor licensing model for deployment in a departmental or small business environment that has Internet users or number of users that makes per processor licensing more attractive than the DB2 UDB Workgroup Server Edition licensing model. The WSUE can be deployed in Linux, UNIX, and Windows environments on systems with up to four CPUs.

DB2 UDB Personal Edition

DB2 UDB Personal Edition (PE) provides a single user database engine ideal for deployment to PC-based users. PE can be remotely managed, making it the perfect choice for deployment in occasionally connected or remote office implementations that don't require multi-user capability.

DB2 Universal Developer's Edition

DB2 UDB Universal Developer's Edition (UDE) offers a low-cost package for a single application developer to design, build, or prototype applications for deployment of any of the DB2 client or server platforms. It includes all client and server DB2 editions, DB2 Connect, DB2 Extenders, DB2 Warehouse Manager, and Intelligent Miner. The software in this package cannot be used for production systems.

DB2 Personal Developer's Edition

DB2 UDB Personal Developer's Edition (PDE) enables a developer to design and build single user desktop applications. This offering includes Windows and Linux versions of the DB2 Personal Edition products as well as the DB2 Extenders.

DB2 Warehouse Manager

DB2 Warehouse Manager extends the scalability, manageability, and accessibility of DB2 data warehouses and data marts built using the Data Warehouse Center. It provides an integrated and flexible set of tools for key steps of designing, populating, and managing data warehouses.

DB2 Spatial Extender

DB2 Spatial Extender enables users to leverage the power of the Relational Database model for managing their location-based data just as easily as traditional business data, and use industry-standard SQL for spatial data analysis, which can add competitive advantages to existing and new applications.

The DB2 Spatial Extender is available free of charge in the DB2 Personal Edition and DB2 Workgroup Server Edition for up to five users. It is also available as a separate program for DB2 UDB Workgroup Server Unlimited Edition and DB2 UDB Enterprise Server Edition.

DB2 Net Search Extender

DB2 Net Search Extender enables a solution to include a powerful in-memory search capability for text-based data. The integration of these advanced searching capabilities directly into the database saves the time and expense of integrating third-party, text-based solutions while providing the speed and flexibility in text searching demanded by e-commerce applications.

The DB2 Net Search Extender is available free of charge in the DB2 Personal Edition and DB2 Workgroup Server Edition for up to five users. It is also available as a separate program for DB2 UDB Workgroup Server Unlimited Edition and DB2 UDB Enterprise Server Edition

DB2 Datalinks Manager

DB2 Datalinks Manager can be used to manage data files that are not normally found in a database, for example, engineering blueprints or medical x-rays. These data files can be on a file system outside of the database.

DB2 provides a predefined data type called DATALINK. A DATALINK value in a database represents an object stored in a storage system outside of the database system. DB2 will treat the DATALINK value as if it were stored in the database, even though it is not. This provides robust support in terms of integrity, access control, and recovery. A DB2 extension, called the DB2 File Manager, enables the asset management of files stored on file servers outside of the database management system.

Using DATALINKs and the DB2 File Manager means that external files can be backed up with the database and SQL Data Control Language statements can be used to control permission to those files (for example, GRANT and REVOKE). Users can create indexes on text, images, and videos, and store those attributes in relational tables along with the DATALINK value. The DATALINK value is a pointer or a uniform resource locator (URL) to an external file.

DB2 Connect Personal Edition

DB2 Connect Personal Edition (CPE) provides the application programming interface (API) drivers and connectivity infrastructure to enable direct connectivity from Windows and Linux desktop applications to mainframe and iSeries database servers. CPE is specifically designed and is licensed for enabling two-tier client-server applications running on individual workstations and as such is not appropriate for use on servers.

DB2 Connect Enterprise Edition

DB2 Connect Enterprise Edition (CEE) is a combination of DB2 Connect server and DB2 client software designed to address the needs of organizations that require robust connectivity from a variety of desktop systems to mainframe and iSeries database servers. DB2 client software is deployed on desktop systems and provides API drivers that connect client-server applications running on these desktop systems to a DB2 Connect server.

Designed to provide connectivity for client-server applications in large scale demanding environments, DB2 Connect server provides connection pooling and connection concentrator functions to maximize application availability while minimizing mainframe resource usage. DB2 Connect Enterprise Edition licensing terms are geared towards addressing the needs of two-tier client-server applications. License charges for the DB2 Connect Enterprise Server Edition product are based on the number of users of the product. Two types of licensing users are provided: concurrent users and registered users.

DB2 Connect Application Server Edition

DB2 Connect Application Server Edition product is identical to the DB2 Connect Enterprise Server in its technology. Just like the DB2 Connect Enterprise Edition it is designed for large scale demanding environments. However, its licensing terms and conditions are meant to address specific needs of multi tier client-server applications as well as applications that utilize Web technologies. DB2 Connect Application Server Edition license charges are based on the size of the number of processors available to the application servers where the application is running. License charges are not affected by the number of users of the application, size of the DB2 Connect server itself, or the size of the mainframe database server.

DB2 Connect Unlimited Edition

DB2 Connect Unlimited Edition product is ideal for organizations with extensive usage of DB2 Connect, especially where multiple applications are involved. This product provides program code of the DB2 Connect Personal Edition as well as program code identical to the DB2 Connect Application Server Edition for unlimited deployment throughout an organization.

DB2 Connect Unlimited Edition license fees are based on the size of the mainframe database server (measured in MSUs) and are not affected by either the number of users nor the number of processors available to the application servers. This makes DB2 Connect Unlimited Edition an ideal choice for organizations where multiple applications are utilizing DB2 Connect or organizations with a mix of two-tier, multi tier client-server, Web-based applications.

1.2.2 Try and buy product availability

IBM provides sixty day evaluation copies of most of the above products for download over the internet. The evaluation code can be obtained by following the instructions at this Web page:

http://www-3.ibm.com/software/data/db2/udb/

1.3 Planning considerations

In this section we discuss when you should use specific DB2 products over others on Windows platforms. We address this from the perspective of providing you all the functionality you need for your workload at the lowest cost.

1.3.1 Product selection guidelines

In this section we offer selection guidelines for the various DB2 products.

DB2 UDB Personal Edition

DB2 UDB Personal Edition (PE) provides a single user database engine ideal for deployment to PC based users that require a local database on their system. PE can be remotely managed with the Satellite Administration Center available with DB2 UDB Enterprise Server Edition, making it the perfect choice for deployment in occasionally connected systems like laptops or remote office implementations that don't require multi-user capability. PE installed systems can also be a client to a central server with one of the server versions of DB2 UDB installed. PE is licensed by system, one license for every system it is installed on.

DB2 UDB Workgroup Server Edition

DB2 UDB Workgroup Server Edition (WSE)) is the database server designed for deployment in a departmental or small business environment that involves a small number of internal users. WSE uses a licensing model designed to provide an attractive price point for smaller installations while still providing a full function database server on systems with up to four CPUs. WSE licensing includes a base server charge and a charge for each concurrent or registered user.

WSE should be considered when your concurrent or registered users is small and you don't require one of the following that require the use of either DB2 UDB Enterprise Server Edition or DB2 UDB Workgroup Server Unlimited Edition:

- Support for more than 4 CPUs
- Number of concurrent/registered users exceed 25-30
- Satellite Administration of distributed PE installations
- Federated database access
- Database Partitioning Feature
- Spatial Extender support of more than 5 users
- Net Search Extender support of more than 5 users
- Host system connectivity required and with the addition of DB2 Connect Enterprise Edition cost exceeds that of DB2 Enterprise Server Edition.

DB2 UDB Workgroup Server Unlimited Edition

DB2 UDB Workgroup Server Unlimited Edition (WSUE) offers the same capabilities provided with DB2 UDB Workgroup Server Edition, but offers a simplified per processor licensing model.

WSUE should be considered when DB2 UDB Workgroup Server Edition can't be used and you don't require one of the following that require the use of DB2 UDB Enterprise Server Edition:

- Support for more than 4 CPUs
- Satellite Administration of DB2 UDB Personal Edition installations
- Federated database access
- Database Partitioning Feature
- Host system connectivity required and with the addition of DB2 Connect Enterprise Edition cost exceeds that of DB2 Enterprise Server Edition.

DB2 UDB Enterprise Server Edition

DB2 UDB Enterprise Server Edition (ESE) should be considered when either DB2 UDB Workgroup Server Edition or DB2 UDB Workgroup Server Unlimited Edition cannot be used. ESE offers a per processor licensing model. If host system connectivity requirements exceed five concurrent users, additional host user access licenses will be required.

DB2 UDB Database Partitioning Feature

The Database Partitioning Feature (DPF) allows DB2 UDB Enterprise Server Edition users to partition a database within a single system or across a cluster of systems. The DPF capability provides the customer with multiple benefits including scalability to support very large databases or complex workloads and increased parallelism for administration tasks.

DPF should be considered for business intelligence (data warehouse) workloads when one or more of the following exist:

- Requirement to spread data across clustered systems
- More than 8-12 CPUs within the SMP system
- Speed of backup/recovery utility operations is critical (as your data volume approaches 100GB)
- Any single table exceeds 64GB using 4K pages, 128GB using 8K pages, 256GB using 16K pages, or 512GB using 32K pages.

DB2 UDB Universal Developer's Edition

DB2 UDB Universal Developer's Edition (UDE) offers a low-cost package for a single application developer to design, build, or prototype applications for deployment of any of the DB2 client or server platforms. It includes all client and server DB2 editions, DB2 Connect, DB2 Extenders, DB2 Warehouse Manager, and Intelligent Miner. The software in this package cannot be used for production systems. UDE is licensed per developer.

UDE should be considered for use on development/test systems where the cost of licensing the developers using the system doesn't exceed that of the licensing of one of the server versions of DB2 UDB.

DB2 UDB Personal Developer's Edition

DB2 UDB Personal Developer's Edition (PDE) enables a developer to design and build single user desktop applications. The software in this package cannot be used for production systems. It is available free of charge when downloaded from the internet or for a small fee when ordered on CD.

PDE should only be considered for developers that are developing applications to be implemented on DB2 UDB Personal Edition.

DB2 Connect Personal Edition

DB2 Connect Personal Edition (CPE) provides the application programming interface (API) drivers and connectivity infrastructure to enable direct connectivity from Windows or Linux desktop applications to mainframe and iSeries database servers. This product is specifically designed and is licensed for enabling two-tier

client-server applications running on individual workstations and as such is not appropriate for use on servers. CPE can also be installed and coexist on an individual workstation that has DB2 UDB Personal Edition installed.

CPE should be considered when an individual workstation requires direct access to DB2 for z/OS and OS/390, DB2 for VM/VSE or DB2 UDB for iSeries data.

DB2 Connect Enterprise Edition

DB2 Connect Enterprise Edition (CEE) DB2 Connect Enterprise Edition (CEE) is a combination of DB2 Connect server and DB2 client software designed to address the needs of organizations that require robust connectivity from a variety of desktop systems to mainframe and iSeries database servers. DB2 client software is deployed on desktop systems and provides API drivers that connect client-server applications running on these desktop systems to a DB2 Connect server. The functionality of CEE is also available as part of DB2 UDB Enterprise Server Edition. License charges for the CEE product are based on the number of users of the product. Two types of licensing users are provided: concurrent users and registered users.

CEE should be considered when a single connection from a gateway to your mainframe data is desirable and you can identify the users that require the connectivity.

DB2 Connect Application Server Edition

DB2 Connect Application Server Edition has identical functionality as DB2 Connect Enterprise Edition (CEE) described above. It is licensed by processor instead of by users.

DB2 Connect Application Server Edition should be considered when it is not possible to identify the users that require mainframe connectivity like internet users or when the cost of licensing users on CEE exceeds the per processor cost. That would be about 50 users on a single processor system.

DB2 Connect Unlimited Edition

DB2 Connect Unlimited Edition product is ideal for organizations with extensive usage of DB2 Connect, especially where multiple applications are involved. This product provides program code of the DB2 Connect Personal Edition as well as program code identical to the DB2 Connect Application Server Edition for unlimited deployment throughout an organization. DB2 Connect Unlimited Edition license fees are based on the size of the mainframe database server (measured in MSUs) and are not affected by either the number of users nor the number of processors available to the application servers.

DB2 Connect Unlimited Edition should be considered in environments that are planning on massive deployment of DB2 Connect where the tracking of users or processors is not possible or desirable.

DB2 Warehouse Manager

DB2 Warehouse Manager provides an integrated and flexible set of tools for key steps of designing, populating, and managing data warehouses.

DB2 Warehouse Manager should be considered when the requirement exists to design, populate and manage data warehouses or data marts where the primary data source is from the DB2 family.

DB2 Spatial Extender

DB2 Spatial Extender enables users to leverage the power of the Relational Database model for managing their location-based data just as easily as traditional business data, and use industry-standard SQL for spatial data analysis, which can add competitive advantages to existing and new applications.

The DB2 Spatial Extender as a separate product should be considered when you have the requirement to spatially enable your database and one of the following conditions exists:

- Requirement for more than 5 users on DB2 Personal Edition or DB2 Workgroup Server Edition.
- DB2 UDB Workgroup Server Unlimited Edition or DB2 UDB Enterprise Server Edition is installed.

DB2 Net Search Extender

DB2 Net Search Extender enables a solution to include a powerful in-memory search capability for text-based data.

DB2 Net Search Extender should be considered for use when the need for high speed textual search is a requirement like that found in e-commerce applications. It needs to be ordered as a separate product when the number of users exceed 5 when running DB2 UDB Personal Edition or DB2 UDB Workgroup Server Edition. It also needs to be ordered as a separate program when used with DB2 UDB Workgroup Server Unlimited Edition and DB2 UDB Enterprise Server Edition

DB2 Datalinks Manager

DB2 Datalinks Manager can be used to manage data files that are not normally found in a database, for example, engineering blueprints or medical x-rays. These data files can be on a file system outside of the database.

The DB2 Datalinks Manager should be considered for use when your application needs to integrate with data that resides in a file system where it is not practical or possible to move the file system data into the database and the need exists to have common security, referential integrity and integrated backup/recovery with that data.

1.3.2 Sample scenarios

In this section we describe some sample workload requirements and explain what DB2 UDB products would be required to satisfy the requirements at the lowest cost.

Small departmental application

This application would be used for a standalone departmental system running on a Windows 2000 Server with less than four processors and has fewer than 20 users with no requirement to access mainframe data. DB2 UDB Workgroup Server Edition would satisfy this requirement. If for some reason access to mainframe or iSeries would also be required, DB2 Connect Enterprise Edition could also be included.

Occasionally connected laptops to departmental server

This application requires that users with laptop workstations running Windows XP maintain a local database with a subset of information that is synchronized with data maintained on a departmental server at a central location running Windows 2000 Server with two processors. The requirement also exists that the databases on the laptops be centrally maintained.

For the laptop users we would need to have DB2 UDB Personal Edition on their workstations. For the departmental server we would require the use of DB2 UDB Enterprise Server Edition (ESE). ESE is required to provide the Satellite Administration support to centrally manage the laptop databases. If the requirement for centrally management didn't exist we could satisfy the departmental server needs with DB2 UDB Workgroup Server Edition or DB2 Workgroup Server Unlimited Edition.

Distributed datamart with federated access to mainframe data

This application is a distributed datamart on a Windows 2000 Server with two processors to handle end user ad hoc queries with occasional transparent drill-thru to an enterprise warehouse located on a mainframe DB2 system.

The requirement for transparent drill-thru dictates that we use the federated system support only available with DB2 UDB Enterprise Server Edition. If the requirement was for direct access instead of transparent access to mainframe DB2 data we could use DB2 Workgroup Server Edition along with DB2 Connect Enterprise Edition

Large scale data warehouse

This application is a large data warehouse that exceeds the capacity that can be provided on a single Windows 2000 Server system.

In this example, since the data needs to be partitioned across multiple system we require the use of DB2 Enterprise Server Edition along with the Database Partitioning Feature.

Development and test server for five developers

To support the development and testing environment for multiple projects we need to setup a separate development system on Windows 2000 Server to isolate that activity from the production environment.

This most cost effective way to meet this requirement is by using the DB2 Universal Developer's Edition. It provides the database and the tools to support development/test environment.

1.4 DB2 UDB Version 8 highlights

IBM DB2 UDB V8 for Linux, UNIX and Windows marks the next stage in the evolution of the relational database. DB2 is the database of choice for the development and deployment of critical solutions such as:

- e-business
- Business intelligence
- Content management
- Enterprise Resource Planning
- Customer Relationship Management

We introduce the highlights of the Version 8 enhancements. For more detail on the specific enhancements see the IBM DB2 Universal Database What's New Version 8 manual available at the following Web site:

http://www-3.ibm.com/software/data/db2/udb/pdfs/db2q0.pdf

Highlights of Version 8 include:

- **Innovative manageability:**
  - Configuration Advisor puts the knowledge of a seasoned DBA at your fingertips
  - Health Center/Monitor keeps your database functioning
  - Memory Visualizer let's you dynamically see and control DB2's memory usage
  - Advisors to deliver expert advice on index and materialized query tables
  - Simplified management of large scale partitioned databases
- **New levels of integrated information:**
  - Federated Web Services allows combination of data from Web service providers
  - XML Productivity Tools simplify integrating XML
- **Robust e-business foundation:**
  - Connection Concentrator for more user scalability
  - Dynamic configuration
  - In-place online reorg
  - Online load
  - Online storage management
  - Null and default compression
  - Replication enhancements
  - New client architecture
- **Integrated business intelligence:**
  - Multidimensional data clustering improves performance of complex queries
  - Real-time and bulk scoring of data
- **Enhanced Application Development Productivity:**
  - Development Center
  - WebSphere integration
  - Microsoft integration

1.5 DB2 UDB integration with Microsoft Windows

When providing DB2 on a platform, IBM seeks to offer more than just another port. Wherever possible, IBM customizes DB2 to exploit the capabilities of a platform. This was true of DB2 for Windows NT and Windows 2000 and will continue with the introduction of Windows Server 2003. In this section we discuss the capabilities of DB2 on Windows 2000 and the extensive integration that has been achieved.

IBM's DB2 UDB on Windows was designed to exploit all the key features of Windows NT and Windows 2000 and to integrate seamlessly into the Windows environment. The result of this effort was that DB2 passed the "Designed for Microsoft Back Office" logo certification. DB2 has also passed the "Certified for Microsoft Windows" logo qualifications. In fact it was the first database to do so, even before Microsoft's own databases

1.5.1 Built for the Windows environment

To deliver DB2 for Windows, IBM did more than just port our UNIX database server to the Windows platform. Wherever possible, IBM has customized DB2 to exploit the capabilities of Windows. Due to the platform-independent design of DB2, this turned out to be a relatively easy task.

By modifying less than 10% of the code, IBM was able to deliver a product that had all the functionality, reliability, and scalability that have become standards on other platforms. With changes to such a small portion of the product, many wonder what degree of integration IBM was able to achieve. The answer is a very high degree of integration.

In the following sections we discuss some of the key integration points that are offered by DB2.

System integration

DB2 integrates with Windows and key administrative tools. As an example, the DB2 engine and other key components all run as Windows services. This provides a number of benefits like automated startup and additional administrative capabilities. DB2 also integrates well with the Windows security model. All user authentication and group enumeration is done using the users and groups defined in Windows. This has the obvious advantages of not requiring duplicate userids and administration.

One of the most attractive aspects of the Windows platform has been the ease-of-administration offered by its user interface and administrative tools. DB2 not only offers its own rich set of GUI based administration tools, but where appropriate also integrates with key Windows tools. An excellent example of this is DB2's support of the Windows Performance Monitor. Even though DB2 provides its own performance monitoring tools and capabilities, DB2 exposes its performance counters to the Windows Performance Monitor. This allows a single interface to monitor performance of key OS performance indicators and key DB2 performance indicators.

Another key tool that DB2 integrates with is the Event Log. This tool provides the single point of information about Windows and running applications. The information recorded by this tool includes status information like DB2 start/stop history. DB2 also records major problems it encounters to this log. Generally, the Event Log will contain a subset of the information available in DB2's own db2diag.log, but provides a graphical interface into the information, offering users an alternative way to diagnose problems.

Windows Active Directory maintains a list of resources such as systems, users, and shares that are available in the Windows 2000 network. It provides better support for the needs of large customers and greatly simplifies administration of Windows and its resources. DB2 integrates into the Active Directory by listing DB2 servers and databases. This eliminates the need for cataloging DB2 nodes and databases, making it much easier to deploy and use DB2.

Windows 2000 added the Kerberos authentication protocol. This protocol has a number of advantages over the proprietary NTLM protocol used by previous Windows versions. By supporting this protocol, DB2 will be able to deliver a secure single logon. If both client and server are running Windows 2000, Windows XP or Windows Server 2003, DB2 will be able to authenticate the user based on the Kerberos credentials obtained at the client, eliminating the need to have the user logon again for DB2.

Many of the key administrative tools in Windows 2000 have been implemented as Microsoft Management Console (MMC) snap-ins. MMC provides a framework for delivering tools with a consistent interface and single point of invocation. To continue the tradition of merging DB2 capabilities with the Windows administrative tools, DB2 provides a snap-in for MMC that will allow administrators to launch the DB2 tools for administering DB2.

Windows environment

IBM understands that successful integration into the Windows environment means more than just running well on the operating systems. It is also necessary to integrate with other key Microsoft products and development interfaces. DB2 meets both of these criteria.

One challenge for companies with widespread Windows implementations is the distribution and maintenance of software to servers and desktop machines. To aid with this effort DB2 supports both remote or unattended installation and thin clients. The unattended installation option allows DB2 server or client code to be installed on a machine with no user intervention. This is done by use of a response file that customizes the install of DB2. When combined with software distribution packages like Microsoft's Systems Management Server (SMS), unattended installation will allow automated distribution of DB2 and its updates to any number of Windows machines. The thin client support only requires a minimum of DB2 software to be installed on a particular client. Configuration information and much of the code is stored on a centralized server allowing for both a smaller footprint and simplified administration of clients. This capability is also appealing to customers using Microsoft Terminal Services.

Microsoft promotes a number of programming interfaces for database access. Initially they introduced Open Database Connectivity (ODBC) as a database independent programming interface. As the Microsoft programming tools and needs evolved, they added other interfaces designed to simplify programming or add new capabilities. Among these are OLE DB, and ADO. DB2 fully supports the ODBC interface and provides a native ODBC driver. DB2 also provides an OLE DB Provider to permit OLEDB applications native access to DB2 data. In addition to these Microsoft interfaces, DB2 also provides access via its own embedded SQL interface as well as the Java interfaces JDBC and SQLJ.

In addition to supporting the key development interfaces, DB2 also supports the major development tools from Microsoft. This includes Microsoft Access and Microsoft Visual Studio which includes Visual Basic and Visual C++. This support includes a free tool for migrating Access applications to DB2, and Visual Studio Add-Ins which integrate key components of DB2 (like the Stored Procedure Builder) into the Visual Studio development environment.

Another point of integration required for DB2's success on Windows lies in working with Microsoft Transaction Services (MTS). MTS implements the transaction services portion of Microsoft's Component Object Model (COM). DB2 supports MTS both as a coordinator for distributed units of work and for the development of transactional components for applications.

Performance

One of DB2's key strengths has always been its ability to deliver high performance. DB2 maintains this strength on the Windows platform. In addition to the many features that enable DB2 to deliver high performance across all platforms, such as advanced optimizer technology, DB2 exploits some features of Windows that enable higher performance. This performance has been demonstrated with a number of leading benchmark results.

The first of these performance features is exploitation of the threaded nature of the Windows OS. Windows allows a single process to execute multiple concurrent tasks. Each of these tasks is referred to as a thread. DB2 exploits this architecture by implementing a single process, multi threaded model. The DB2 engine runs one process which in turn dispatches numerous threads to perform its work. This model, and DB2's ability to break complicated queries into multiple concurrent tasks, allows DB2 to fully exploit both the CPU and I/O capabilities of a given system with minimal overhead.

As on other platforms, DB2 on Windows supports both the Systems Managed Space (SMS) and Database Managed Space (DMS) storage models. This allows for balancing the needs of administrative simplicity with overall performance. For DMS on Windows, DB2 supports the raw I/O interface. This allows DB2 to write directly to a device or partition without the overhead of a file system. Bypassing the file system in this manner provides the most efficient I/O path possible for DB2.

DB2 on Windows 2000 exploits the Address Windowing Extensions (AWE, sometimes referred to as PAE) APIs. These APIs allow applications to address more than 3 gigabytes of RAM. DB2 uses these APIs to implement its extended storage cache much like it does on other platforms. This allows DB2 to fully exploit the memory capabilities of today's large Intel servers.

Scalability

Perhaps the key advantage of DB2 over other Windows DBMS is its scalability. DB2 easily scales from the smallest implementations with a few megabytes of data and a few users, to very large systems that support terabytes of data or thousands of users. DB2 intelligently exploits the SMP architecture found in many Intel based servers, but goes beyond that to a very scalable, clustered implementation.

The DB2 Enterprise Server Edition with the Database Partitioning Feature allows a single DB2 database to span 2 or more machines. This is done with a shared-nothing, partitioned architecture. This means that each node has its own CPU, RAM, and disk that is not used by the other database nodes. Each node is interconnected via either TCP/IP, over your choice of media, or for very high performance and scalability DB2 can use a switch that implements Intel's Virtual Interconnect Architecture (VIA). This eliminates any single point of contention and allows DB2 to scale better than any other Windows based relational database.

This architecture does not require any special hardware, software, or drivers, which makes it both simple to implement and inexpensive. The partitioned architecture means that each node contains only a portion, or partition, of the data that makes up the database. DB2 automatically controls the distribution of

the data to the appropriate node. When executing queries the very sophisticated DB2 optimizer determines which nodes actually contain data needed to resolve the query and automatically distributes the workload to those nodes. This eliminates the need for any customized application or SQL coding to exploit this environment. It is completely transparent to both the application and the end user.

High availability

Although there have been a number of improvements in the reliability and availability of Intel servers, like RAID storage, redundant components, and hot-swap capability, there are still cases where a single Intel based solution will or must come down. If you've ever experienced the infamous blue-screen-of-death, then you understand. To address the availability needs of the Windows platform, Microsoft provides fail over capability in the Windows 2000 Advanced Server.

With this solution, when one machine fails another takes over its function with minimal downtime or interruption to the user. DB2 exploits this capability to provide highly available configurations in either a hot-standby or mutual-takeover scenario.

With hot-standby support, only one of the machines in a fail over pair would actively service DB2. In the event of a failure, DB2 would fail-over to the other machine and resume processing. As an alternative, each of the machines could be running DB2 or other services and pick up the tasks of the other server in the event of a failure. This allows for better utilization of the equipment during up-time, but may result in performance degradation after a failure. DB2 supports both of these scenarios in either a partitioned or non-partitioned environment. For DB2 Database Partitioning Feature clusters that span more than 2 machines, DB2 can span multiple of these availability clusters to achieve high availability.

Windows 2000 Data Center introduced improved high availability support. Where other versions of Windows 2000 only support fail over pairs, Data Center will support fail over for clusters of up to four machines. This will allow for much greater flexibility when defining a high availability solution. DB2 supports these extended capabilities in both partitioned and non-partitioned environments.

Summary

DB2 is a very tightly integrated solution for the Windows environment that exploits the key features of the operating system to provide excellent stability, reliability, and scalability. DB2 is uniquely positioned to provide your Windows-based database solutions regardless of the size or type of your implementation. IBM has been and will continue to be the leader in database technology across all platforms.

2

Installation and deployment

In today's companies, from small businesses to large enterprises, the burden of software maintenance is one of the most expensive and time consuming tasks.

This chapter provides step-by-step procedures for installation of the DB2 server and client software — ranging from single machine installation requiring user response, through automated software management.

The following types of installation are covered:

- Installation Wizard (single installation)
- Installation Profile (multiple installation)

We also discuss basic Active Directory information and installation, and explain how to integrate DB2 components into Lightweight Directory Access Protocol (LDAP).

2.1 Installation preparation and considerations

In this section we discuss what preparation steps are necessary before you begin the installation of DB2 UDB on a Windows system. The following topics are covered:

- Installation overview for DB2 servers (Windows)
- Installation requirements
- Authorization considerations
- FixPak considerations
- Migration considerations

2.1.1 Installation overview for DB2 servers on Windows

In this section we provide an installation overview for DB2 Enterprise Server Edition (single-partition) and DB2 Workgroup Server Edition on Windows.

Preparing your environment for installation

Before you install, you need to prepare your computer for installation, as follows:

- Verify that your computer meets the necessary installation requirements.
- Ensure that your system has enough memory to run DB2.
- Ensure that your system has enough disk space for a DB2 installation.
- Ensure that you have the necessary user accounts for installation and setup You require one user account for installation and two user accounts for setup. The user accounts required for setup can be created before you install — or you can have the DB2 Setup wizard create them for you.

If you are installing on Windows 2000 and are planning to use Lightweight Directory Access Protocol (LDAP) to register the DB2 server in LDAP, you will extend the Windows 2000 directory schema so that it can contain DB2 object classes and attribute definitions.

Installing DB2

After preparing your environment, you can install DB2 using the DB2 Setup wizard, which has the following features:

- A DB2 Setup Launchpad, from which you can view installation notes and release notes, and which you can use to learn about DB2 Version 8 features
- Typical, Compact, and Custom installation types
- The option to install support for multiple languages
- DB2 Administration Server setup (including DAS user setup)

- Administration contact and Health Monitor notification setup
- Instance setup and configuration (including instance user setup)
- DB2 tools metadata and data warehouse control database setup
- Response file creation capability

Applying the latest FixPaks

After you install DB2 using the DB2 Setup wizard, it is recommended that you apply the latest DB2 Version 8 FixPak. DB2 FixPaks are available on the IBM support site.

Verifying the installation

After you install DB2 using the DB2 Setup wizard and have applied the latest DB2 FixPak, it is recommended that you verify the installation, as follows:

1. Create a sample database using the **db2sampl** command. You can also create a sample database using the First Steps utility, if you choose to install it.
2. Once the sample database has been created successfully, you will run SQL commands to retrieve sample data.

2.1.2 Installation requirements

In this section we discuss the operating system, hardware, and software requirements necessary for the installation of DB2.

Installation requirements for DB2 servers (Windows)

To install DB2, the following operating system and software requirements must be met:

Operating system requirements

DB2 Workgroup Server Edition runs on:

- Windows NT Version 4 with Service Pack 6a or higher
- Windows 2000. Service Pack 2 is required for Windows Terminal Server
- Windows XP (32–bit)
- Windows Server 2003 (32–bit)

DB2 Enterprise Server Edition runs on:

- Windows NT Version 4 with Service Pack 6a or higher
- Windows 2000. Service Pack 2 is required for Windows Terminal Server
- Windows Server 2003 (32–bit and 64–bit)

Hardware requirements

For 32–bit DB2 products, a Pentium or Pentium compatible CPU is required. For 64–bit DB2 products, an Itanium or Itanium compatible CPU is required.

Software requirements

These are the software requirements:

- If you plan to use the Tivoli Storage Manager facilities for backup and restore of your databases, you require the Tivoli Storage Manager Client Version 4.2.0 or later. If you are running in a 64-bit environment, you need Tivoli Storage Manager Client Version 5.1 or later.
- Java Runtime Environment (JRE) Version 1.3.1 is required to run DB2 servers and DB2's Java-based tools, such as the Control Center. The DB2 Setup wizard will install the Java Runtime Environment (JRE) Version 1.3.1 if you choose to install DB2 Java-based tools.
- A browser is required to view online help.
- If you plan to use LDAP (Lightweight Directory Access Protocol), you require either a Microsoft LDAP client or an IBM SecureWay LDAP client V3.1.1.
- If you plan to use the Simple Network Management Protocol (SNMP) subagent, you require DPI 2.0 provided by IBM SystemView Agent. SNMP is not supported with DB2 offerings on Windows 64-bit platforms.
- If you plan to create Stored Procedures, the Microsoft Visual C++ Version 7.0 or Microsoft Visual C++ Version 6.0 must be installed.

Windows (64–bit) considerations

Here are some considerations you need to keep in mind:

- Local 32–bit applications are supported.
- 32–bit UDFs and stored procedures are supported.
- SQL requests from remote 32-bit downlevel clients are supported.
- DB2 Version 8 Windows 64-bit servers support connections from DB2 Version 6 and Version 7 32-bit clients only for SQL requests. Connections from Version 7 64-bit clients are not supported.

Windows 2000 Terminal Server installation limitations

You cannot install DB2 Version 8 from a network mapped drive using a remote session on Windows 2000 Terminal Server edition. The available workaround is to use Universal Naming Convention (UNC) paths to launch the installation, or run the install from the console session.

For example, if the directory *c:\pathA\pathB\...\pathN* on serverA is shared as serverdir, you can open \\serverA\serverdir\filename.ext to access the file c:\pathA\pathB\...pathN\filename.ext on server.

Memory requirements for DB2 servers (Windows)

At a minimum, DB2 requires 256 MB of RAM. Additional memory may be required. When determining memory requirements, be aware of the following:

- Additional memory may be required for non-DB2 software that may be running on your system.
- Additional memory is required to support database clients.
- Specific performance requirements may determine the amount of memory needed.
- Memory requirements will be affected by the size and complexity of your database system.
- Memory requirements will be affected by the extent of database activity and the number of clients accessing your system.

Disk requirements for DB2 servers (Windows)

The disk space required for DB2 Enterprise Server Edition (ESE) (single partition) or Workgroup Server Edition (WSE) depends on the type of installation you choose. The DB2 Setup wizard provides Typical, Compact, and Custom installation types. Table 2-1 provides approximate disk space requirements for each installation type.

Table 2-1 Disk space requirements for each installation type

Installation type	Minimum disk space
Typical	350 MB
Compact	100 MB
Custom	100 MB

Exact disk space requirements depend on the features installed and the type of disk drive. You may require significantly more space on FAT drives with large cluster sizes.

Typical installation

DB2 is installed with most features and functionality, using a typical configuration. Typical installation includes graphical tools such as the Control Center and Configuration Assistant. You can also choose to install a typical set of data warehousing or satellite features.

Compact installation

Only the basic DB2 features and functions are installed. Compact installation does not include graphical tools.

Custom installation

A custom installation allows you to select the features you want to install. The DB2 Setup wizard will provide a disk space estimate for the installation options you select. Remember to include disk space allowance for required software, communication products, and documentation. In DB2 Version 8, HTML and PDF documentation is provided on separate CD-ROMs.

2.1.3 Authorization considerations

In this section we consider some requirements for authorization of user accounts.

User accounts required for installation of DB2 servers

If you are installing on Windows NT, Windows 2000, Windows XP, or Windows Server 2003, you require three DB2 server user accounts, an installation user account, and two user accounts for setup. The installation user account must be defined prior to running the DB2 Setup wizard. The setup user accounts (DB2 Administration Server user, and DB2 instance user) can be defined prior to installation or you can have the DB2 Setup program create them for you.

All user account names must adhere to your system naming rules and to DB2 naming rules.

Installation user account

A local or domain user account is required to perform the installation. The user account must belong to the Administrators group on the machine where you will perform the installation and must have the following user rights:

- Act as part of the operating system
- Create a token object
- Increase quotas
- Replace a process level token

DB2 Administration Server user account

A local or domain user account is required for the DB2 Administration Server (DAS). You can create the DAS user account before installing DB2 or you can have the DB2 Setup wizard create it for you. If you want to have the DB2 Setup wizard create a new domain user account, the user account you use to perform the installation must have authority to create domain user accounts. The user account must belong to the Administrators group on the machine where you will perform the installation. This account will be granted the following user rights:

- Act as part of the operating system
- Create token object
- Log on as a service
- Increase quotas

- Replace a process level token

The DB2 Administration Server (DAS) is a special DB2 administration service used to support the GUI tools and assist with administration tasks on local and remote DB2 servers. The DAS has an assigned user account that is used to log the DAS service on to the computer when the DAS service is started. It is recommended that the DAS user have SYSADM authority on each of the DB2 systems within your environment so that it can start or stop other instances if required. By default, any user that is part of the Administrator group has SYSADM authority.

DB2 instance user account

A local or domain user account is required for the DB2 instance. You can create the DB2 instance user account before installing DB2 or you can have the DB2 Setup wizard create it for you. If you want to have the DB2 Setup wizard create a new domain user account, the user account you use to perform the installation must have authority to create domain user accounts. The user account must belong to the Administrators group on the machine where you will perform the installation. This account will be granted the following user rights:

- Act as part of the operating system
- Create token object
- Increase quotas
- Log on as a service
- Replace a process level token

Every DB2 instance has one user that is assigned when the instance is created. DB2 logs on with this user name when the instance is started.

2.1.4 FixPak considerations

In this section we discuss the various considerations you need to keep in mind when applying a FixPak.

Applying the latest FixPak

Applying the latest FixPak (optional) is a part of the larger task of installing DB2 products. A DB2 FixPak contains updates and fixes for bugs (Authorized Program Analysis Reports, or APARs) found during testing at IBM, as well as fixes for bugs reported by customers.

Every FixPak is accompanied by a document, called APARLIST.TXT, that describes the bug fixes it contains. FixPaks are cumulative. This means that the latest FixPak for any given version of DB2 contains all of the updates from previous FixPaks for the same version of DB2. We recommend that you keep your DB2 environment running at the latest FixPak level to ensure problem-free

operation. When installing a FixPak on a partitioned ESE system, all participating computers must have the same FixPak installed while the system is offline.

Prerequisites

Each FixPak may have specific prerequisites. See the FixPak README that accompanies the FixPak for more information.

Procedure

1. Download the latest DB2 FixPak from the IBM DB2 UDB and DB2 Connect Online Support Web site at:

   `http://www.ibm.com/software/data/db2/udb/winos2unix/support`

2. Each FixPak contains a set of Release Notes and a README. The README provides instructions for installing the FixPak.

2.1.5 Migration considerations

DB2 UDB Version 8 supports the migration of previous versions of DB2 with the restrictions listed below.

Migration is required if you have a previous version of DB2 or DataJoiner installed and you want to use the instances, databases, and directories from the older version with DB2 UDB Version 8.

For details on migration, please see Part 2 of the manual, *IBM DB2 Universal Database Quick Beginnings for DB2 Servers Version 8 Part 2.*

Migration restrictions

Be aware of the following restrictions before you migrate to DB2 Version 8:

- Migration is only supported from:
  - DB2 Version 6.x or Version 7.x (All platforms supported in Version 6.x and Version 7.x; Linux must be at Version 6 FixPak 2).
  - DB2 DataJoiner V2.1.1 32-bit (AIX, Windows NT, and Solaris Operating Environment)
- Issuing the **migrate database** command from a DB2 Version 8 client to migrate a database to a DB2 Version 8 server is supported; however, issuing the migration command from an DB2 Version 6 or Version 7 client to migrate a database to a DB2 Version 8 server is not supported.
- When migrating from DB2 DataJoiner V2.1.1, DB2 Relational Connect is required to support non-IBM data sources.
- Migration between platforms is not supported. For example, you cannot migrate a database from a DB2 server on Windows to a DB2 server on UNIX.

- Migrating a partitioned database system that has multiple computers requires that database migration be performed after DB2 Version 8 is installed on all participating computers.
- Windows allows only one version of DB2 to be installed on a computer. For example, if you have DB2 Version 7 and install DB2 Version 8, DB2 Version 7 will be removed during the installation. All instances are migrated during DB2 installation on Windows operating systems.
- User objects within your database cannot have DB2 Version 8 reserved schema names as object qualifiers. These reserved schema names include: SYSCAT, SYSSTAT, and SYSFUN.
- User-defined distinct types using the names BIGINT, REAL, DATALINK, or REFERENCE must be renamed before migrating the database.
- You cannot migrate a database that is in one of the following states:
  - Backup pending
  - Roll-forward pending
  - One or more table spaces not in a normal state
  - Transaction inconsistent
- Restoration of down-level (DB2 Version 6.x or Version 7.x) database backups is supported, but the rolling forward of down-level logs is not supported.
- Database transactions executed between database backup time and the time DB2 Version 8 migration is completed are not recoverable.

2.2 Installation wizard (single installation)

In this section we discuss the installation wizard type of installation. In each of the steps, the user response will be required to proceed to the next step. It is important that you have planned and have at hand all the information needed before you start the installation procedure.

The step-by-step installation procedures can help system maintenance personnel to easily develop procedures and documentation for the installation procedures.

In this section we also provide information which is supplementary to the next chapter, through install profile generation during the installation procedure.

DB2 Setup Wizard

DB2 Setup Wizard is the tool for installation and configuration tasks on DB2. In the effort to help DB2 installation easiness, DB2 Setup Wizard is designed to be straightforward and handful. It can do basic installation without being hassled

with all the parameters that are used to have at installation step. If you desire, it is possible to leave a majority of the parameters and configurations to be done or customized just after the file packages are installed. Also, its possible to add or remove packages and features after the installation procedure is finished.

DB2 Setup Wizard packages

Along with the specific engine and programs for each type of installation, DB2 Setup Wizard provides additional packages for each type of install. They are grouped into server and client tools. DB2 Setup Wizard will display them accordingly to the type of setup (client or server, run-time or development clients). If you are not sure about the features that you are installing, choose the typical installation procedure. To get the best performance in production servers and avoid unnecessary packages, we recommend that you choose carefully the packages that you are installing.

Administration Tools

The following basic and advanced tools are used to manage and configure DB2 objects:

- **Control Center:** Java-based tool for browsing and administration of DB2 objects
- **Client Tools:** Basic client tools and functionality
- **Command Center:** Java-based tool for issuing DB2 commands and statements
- **Configuration Assistant:** Java-based tool for configuring DB2 servers, clients and database access
- **Database Tools:** Includes a number of database tools and executables
- **Event Analyzer:** Used in conjunction with event monitors to trace performance data
- **DB2 Web Tools:** Allows you to run the command center and health center as Web server applications providing DB2 server access through a Web browser

Application development tools

The following tools and sample programs are used to develop DB2 applications:

- **Development Center:** Java-based tool to develop, build and test stored procedures and UDFs
- **Warehouse Samples:** Sample applications to show how to integrate applications with Data Warehouse Center
- **Spatial Extender Samples:** Samples for using Spatial Extender
- **Information Catalog Manager Sample:** Samples used with Information Catalog Center Tutorial.

- **Java Development Kit:** IBM's enhanced JDK
- **Sample Applications:** Sample applications in DB2-supported languages
- **SQLJ Application Development Tools:** Tools to build Java applications that contain embedded SQL (SQLJ) using JDBC driver
- **SQLJ Samples**

Server support

The following are required DB2 server components and additional components for extending server functionality:

- **Apply:** Replication component for copying changes to target tables
- **Capture:** Replication component for Capture changes from source tables
- **Connect Support:** Provides ability to connect to host, AS/400, and i/Series systems
- **Satellite Control Server:** Provides administrative and status reporting support for Satellite systems
- **Relational Connect for Informix Data Sources:** Enables users and applications to submit distributed requests for data managed by Informix systems
- **Communication Protocols:** Contain support components for various protocols. Protocol support is required for client-server communication. DB2 supports the following protocols:
  - NetBIOS Listener
  - Named Pipes Listener
  - TCP/IP Listener

Client support

Required DB2 client components and additional components for extending client functionality:

- **Interfaces:** Interface and support components that allow you to work with different types of applications and data access methods:
  - JDBC Support
  - MDAC 2.7
  - ODBC Support
  - OLEDB Support
  - SQLJ Support
- **Base Client Support:** Components required for database connection, SQL and DB2 command functionality
- **System Bind Files:** Used for such things as database creation and accessing remote host databases

- **Satellite Synchronization:** Allows for a connection information exchange with the DB2 Satellite Control Server
- **Spatial Extender Client:** Client component required for communicating with a Spatial Extender Server
- **Java Runtime Environment:** IBM's enhanced JRE
- **LDAP Exploitation:** Component that allows DB2 to use and LDAP directory to store database directory and configuration information
- **XML Extender:** Provides new data types, functions and stored procedures that assist you in working with XML data in DB2 databases
- **Communication Protocols:** Contain support components for various protocol; DB2 supports the following protocols, which are required for client-server communication:
  - NetBIOS Listener
  - Named Pipes Listener
  - TCP/IP Listener

Business Intelligence

The Business Intelligence group contains components that provide additional functionality for performing business intelligence tasks:

- **Data Warehouse tools:** Components that help you work with and manage a data warehouse:
  - **Data Warehouse Center:** Java-based tool used for data warehouse administration
  - **Warehouse Server:** Controls the interaction of warehouse components, controls and monitors warehouse agents, and provides scheduling functions
- **Information Catalog Manager tools:** Components that help you work with and manage an information catalog:
  - **Information Catalog Center:** Graphical tool used to manage a catalog of your business data
  - **Information Catalog Center for the Web:** Allows you to access information catalogs and view data from a Web browser

Getting started

Here we discuss the First Steps application and sample databases contained in the Getting Started package:

- **First Steps:** Graphical tool that will help familiarize you with DB2 features and functions
- **Sample Database:** Source code to create DB2 sample database objects

- **Warehouse Sample Database Source:** Sample data and metadata that you can use to create a sample data warehouse using the First Steps application
- **XML Extender Samples:** Sample programs that you can use to learn about XML Extender technology

DB2 documentation

DB2 documentation is a separate product from DB2 product installation.
It is also based on DB2 Setup Wizard, and the same procedures can be applied.
It includes an automatic documentation update from the Internet. Since its contents can be updated and changed very easy, we recommend that you go to IBM's Web site to get the latest documentation.

2.2.1 Server installation

In this section we discuss a basic DB2 installation procedure. Some features are left unspecified intentionally to demonstrate the capability of using DB2 Setup Wizard to add more features and configure some options after DB2 is installed.

More information about the packages can be found in the DB2 manuals *Quick Beginnings for DB2 servers* and *DB2 Installation and Configuration Supplement*. Read these manuals to get more in-depth information about specific topics.

DB2 server default packages

DB2 default installation without Data Warehouse and Satellite administration capability includes:

- **DB2 engine files:** All the DB2 engine files.
- **Administration Tools:** Control Center, Client Tools, Command Center, Configuration Assistant, Database Tools, Event Analyzer and DB2 Web Tools.
- **Application Development Tools:** Development Center, Warehouse Samples and Spatial Extender Samples.
- **Java Development Kit:** SQLJ Application Development Tools and SQLJ Samples.
- **Server support:** Apply, Capture, Connect Support, Relational Connect for Informix Data Sources, Communication Protocols (NetBIOS Listener, Named Pipes Listener, TCP/IP Listener).
- **Client support:** Interfaces (JDBC Support, MDAC 2.7, ODBC Support, OLEDB Support, SQLJ Support), Base Client Support, System Bind Files, Satellite, Synchronization, Spatial Extender Client, Java Runtime Environment, LDAP Exploitation, XML Extender and Communication Protocols (NetBIOS Listener, Named Pipes Listener, TCP/IP Listener).

- **Getting started:** First Steps, Sample Database, Warehouse Sample Database Source and XML Extender Samples.

Our environment

The following are hardware and software used in our Lab to install DB2 using the Wizard.

- IBM NetFinity Server 5500 with 1 GB RAM memory and 9 GB HDD
- Windows 2000 Advance server

Pre-installation requirements

To start the server installation, you should have met the following requirements, and you need to have the necessary information at hand:

- Directory path where the DB2 files will be placed
- A minimum of 256 MB RAM memory, up to the requirements of your environment and databases
- At least 350 MB of disk space for a typical installation
- Conformance with DB2 requirements of the latest FixPacks for Windows 2000 and Windows Server 2003
- A common account or a User account for each purpose:
  - Installation user account
  - DB2 Administration Server user account
  - DB2 instance user account

Centralizing task scheduling

In our installation we follow a different approach for the tools database. We have a separate DB2 server for scheduling tasks and procedures. So, the steps shown below won't install the tools database. With this approach, we can lower CPU and memory consumption and provide a centralized way to manage DB2 servers.

However, if you have only one server in your environment, you will not be able to apply this approach. Instead, you can rely on the Windows task scheduling to run some specified scripts. But you will need to control and design specialized scripts that do error reporting and management.

Create db2 administration account

Note: This step is only required if you don't have an account with the privileges mentioned previously. If you desire, the account's name should have a maximum of 8 characters for compatibility issues with UNIX environments.

At the primary domain controller, create a new domain user. Select **Start –> Programs –> Administrative Tools –> Active Directory Users and Computers**, right-click **Users**, and choose **New –> User** (Figure 2-1).

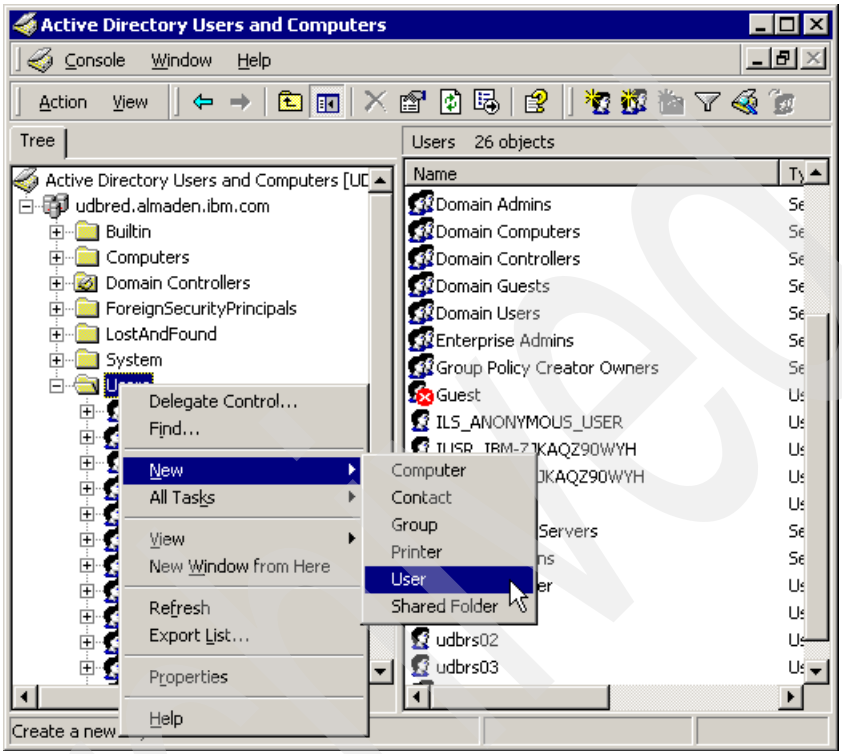

Figure 2-1 Create new user account for DB2 services

Fill in the user name and the required information on the fields and click **Next** (Figure 2-2).

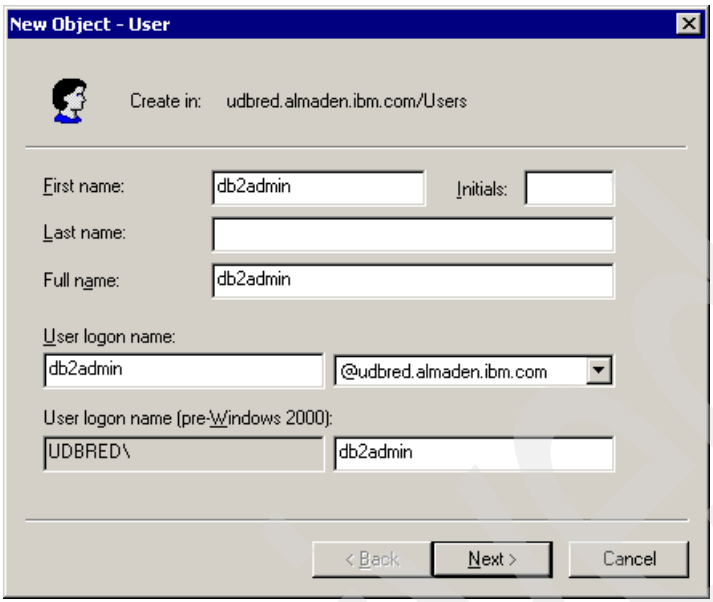

Figure 2-2 Filling in user information and network identification

On the next screen, enter the user's password. Initially, you should apply the policies, **User cannot change password** and **Password never expires**. Refer to your company's policies of handling system accounts. Click **Next**, review the information, and end the new user procedure by clicking **Finish**.

Now add the created user to the domain administration group. Go to **Active Directory Users and Computers**, select the **Users** folder, and right-click the previous created user. Select **Properties**.

Select the **Member of** tab, and click **Add**. Select **Domain Admins** and the **Administrators** group from the domain and click **OK**. Click **OK** to confirm the user modification (Figure 2-3).

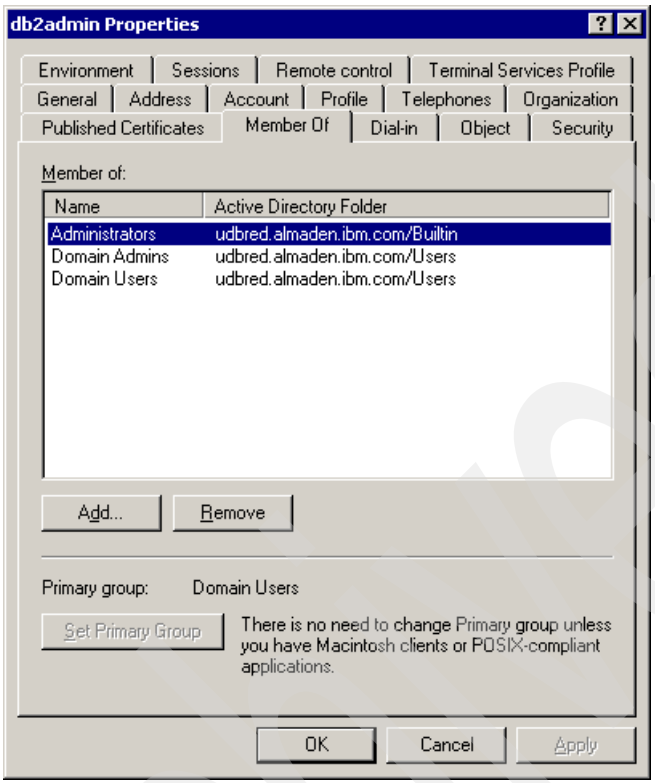

Figure 2-3 User properties and domain membership

Add DB2 server to the domain

Note: This step is only required if the server isn't member of the domain.

On DB2 server, log-on locally using a domain administrator account, right-click the **My Computer** icon and select **Properties**. Select **Network Identification** and click **Properties**.

On the **Member of** panel, choose **Domain**, and enter the domain that the server will be member of. Click **More...** .Leave the option **Change the primary DNS suffix when domain membership changes** checked and/or fill in the **Primary DNS suffix of this computer** with the DNS suffix that this server will have. Click **OK** to proceed (Figure 2-4).

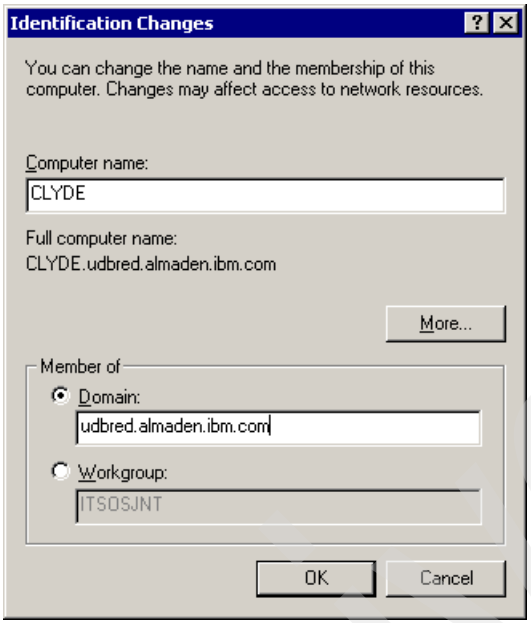

Figure 2-4 Changing server's network information to join the domain

The next dialog will ask for an account which has the permission to add this computer into the domain. The *db2admin* account will be used in this case. Click **OK** after filling in the user name and password. A welcome to the domain pop-up will be shown. Just click **OK** and a pop-up will inform that server rebooting is required. Click **OK** and click **OK** on the **Identification Changes** dialog. Click **Yes** to reboot the server.

Installing DB2

If you try to run DB2 Setup Wizard from a mapped drive on Terminal Services, the following message dialog will appear (Figure 2-5)

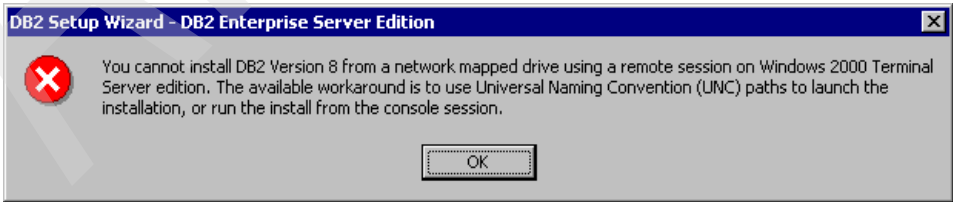

Figure 2-5 DB2 Setup Wizard running from mapped drive on Terminal Services

Note: On some dialogs on the DB2 Setup Wizard, when user-id and password are required to be validated, DB2 will search locally for the user. To validate domain user accounts, add the domain name plus a slash before user name. Example: On Data Warehouse authentication dialogs, enter **`UDBRED\db2admin`**.

Note: DB2 requires the English language to work. On a **Typical** installation it includes English plus the language defined locally. If you need support on another language, choose the **Custom** installation type.

Step-by-step installation procedure

Follow these steps to perform the installation:

1. Place the DB2 Server (ESE or WSE) Install CD in the drive and wait for the setup initial dialog. If the dialog is not shown, or the install files are on the network, search and run **setup.exe** at the directory of db2 installation. On the **IBM DB2 Setup Launchpad** dialog, click the **Install Products** tab (Figure 2-6).

Figure 2-6 DB2 Setup Launchpad

2. On **DB2 Setup Launchpad** dialog, make sure that the **DB2 UDB Server** option is selected and click **Next**.

3. On **DB2 Setup Wizard Welcome** dialog of DB2 Setup Wizard, just click **Next**.
4. After you read the license agreement on the **DB2 Setup Wizard License Agreement** dialog, choose **I accept the terms of the license agreement** option and click **Next**.
5. The **DB2 Setup Wizard installation type** dialog is now shown requiring the type of the installation desired. We continue with **Typical** selection.

   Depending on the product you are installing, the following options may be available:

   – Data warehousing
   – Satellite administration capability

   We leave both unchecked. If you need them afterwards, just rerun DB2 Setup Wizard and check it to install all required data warehousing features and tools. Click **Next** to proceed to the next step.
6. A warning dialog may be shown, advising you about changes in APPC protocol. Refer to the *Release Notes regarding changes to DB2 UDB and DB2 Connection Version 8 APPC Support* documentation if you need more information. Proceed by clicking **OK**.
7. The **DB2 Setup Wizard installation action** dialog is shown.If you want to generate a script from the installation procedure, leave **Save your settings in a response file** checked. Make sure that **Install DB2 Enterprise Server edition on this computer** checked. Otherwise, only the script file will be generated. Click **Next**.
8. The **DB2 Setup Wizard server partitioner** dialog is shown. If you need setup information about server partitioning, refer to the *Quick Beginnings for DB2 servers* documentation. Leave **Single-Partition database environment** option checked and click **Next**.
9. The **DB2 Setup Wizard select installation folder** dialog is shown. Select the drive that has enough disk space to install all DB2 files. You can check the amount of free disk space available on each drive of the system clicking **Disk Space....** You can change the path in the drive where the root directory of DB2 will be placed by clicking Change.... Click **Next** to advance.
10. DB2 Setup Wizard now asks for a user account that will start the DB2 administration service (DAS). Enter the domain, user name, and password, and confirm the password defined previously. Leave the **Use the same user and password for the remaining DB2 services** check box checked if you want to do so. Click **Next**.
11. A warning is displayed if you are installing DB2 server on Windows 2000, because it may not possible to validate the user account if the domain controller is running Windows XP or Windows Server 2003 operating systems. If certain that the user name and password are correct, click **No**.

12. DB2 Setup Wizard now asks where the contact list should be placed. In our case it will be created locally. Select **Local - Create a contact list on this system**, uncheck **Enable notification** and proceed by clicking **Next**.
13. A warning is displayed advising that there are no SMTP server specified to send notifications. Just click **OK**. We set it after the installation procedure.
14. If you leave **Use the same user and password for the remaining DB2 services** unchecked**,** DB2 will repeat the steps 10 and 11 in order to ask you for DB2 instance user account. Also, it will display a dialog where you can configure the protocols selected and change the service start behavior. Proceed on them clicking **Next**.
15. Since we haven't selected yet any SMTP server, leave **Defer the task until the installation is complete** checked. Click **Next**.
16. Click **Install** to start the file copy. At the end of installation process, just click **Finish**. Depending on the system configuration, it could be necessary to reboot the machine. If DB2 Installation Wizard detects anything requiring a reboot, it will ask you to do so. Just reboot your machine.

For information on errors encountered during installation, see the db2.log file, which stores general information and error messages resulting from the install and uninstall activities. By default, the db2.log file is located in the *\MyDocuments\DB2LOG* directory. The location of the *MyDocuments* directory will depend on the settings on your computer.

2.2.2 Client installation

In this section we discuss a basic installation procedure for each client. We show the minimum requirements and steps of the client installation here and provide the post installation steps in the next chapter.

Our environment

The following are hardware and software used in our Lab to demonstrate the client installation process:

- IBM Personal Computer 300 GL with 512 MB RAM memory and 9 GB HDD
- Window 2000 Server

Tip: DB2 clients can be installed using accounts with User privileges. To do this, add write permission to the User group on the HKEY_LOCAL_MACHINE \Software Registry branch.

Run-Time Client installation

On DB2 Run-Time Client, the first thing that you note is the absence of CCA, which is substituted by the ODBC Driver Manager in the task of node and

database cataloging and binding. On the other hand, DB2 Run-Time client now relies on LDAP to do system configuration.

DB2 Run-Time client has these components:

- **DB2 client engine files:** All the DB2 client engine files.
- **Client support:** Interfaces (JDBC Support, MDAC 2.7, ODBC Support, OLEDB Support, SQLJ Support), Base Client Support, System Bind Files, Satellite, Synchronization, Spatial Extender Client, Java Runtime Environment, LDAP Exploitation, XML Extender and Communication Protocols (NetBIOS Listener, Named Pipes Listener, TCP/IP Listener).
- **Administration Tools:** Client Tools, Command Center, Configuration Assistant, Database Tools, Event Analyzer and DB2 Web Tools.

The default installation includes all the components mentioned above except MDAC 2.7. If MDAC 2.7 is required, precede the installation procedure with the Custom option.

If you are planning to use older versions of Windows (NT/98/95), verify with your system administrator if the domain controller accepts pre-Windows 2000 authentication.

Step-by-step installation procedure

Follow these steps to perform the installation:

1. Place the DB2 Run-Time Client Install CD (RTCL) in the drive and wait for the setup initial dialog. If the dialog is not shown or the install files are on the network, search and run **setup.exe** at the directory of db2 installation. On the **DB2 Setup Wizard Welcome** dialog of DB2 Setup Wizard, just click **Next**.
2. On **DB2 Setup Wizard Welcome** dialog of DB2 Setup Wizard, just click **Next**.
3. After you read the license agreement on the **DB2 Setup Wizard License Agreement** dialog, choose **I accept the terms of the license agreement** option and click **Next**.
4. The **DB2 Setup Wizard installation type** dialog is now shown, requesting the type of the installation. There are three types of installations:

   Typical This option installs the client features that are used most often, including all required features, ADO, OLEDB, and ODBC support. Also, it will configure with default values.

   Compact This option installs only required client features and basic libraries to connect to DB2 servers.

   Custom This option lets you select the required client features.

Now we continue the setup using the Custom option. Click **Custom** and click **Next** to proceed to the next step.

5. The **DB2 Setup Wizard installation action** dialog is shown. If you want to generate a script from the installation procedure, leave **Save your settings in a response file** checked. Make sure that **Install DB2 Run-Time on this computer** is checked. Otherwise, only the script file will be generated. Click **Next**.
6. The **DB2 Setup Wizard feature select** dialog is shown. To add MDAC 2.7, open **Run-Time Client–>Client Support->Interfaces**, right-click **MDAC 2.7** and click **Add this package**. Proceed adding or removing packages as required. Select the drive that has enough disk space to install all DB2 files by clicking **Drive**. You can check the amount of free disk space available on each drive of the system clicking **Disk Space...**. You can change the path in the drive where the root directory of DB2 will be placed by clicking **Change...**. Click **Next** to advance.

Note: ADODB and OLEDB components require MDAC 2.7. If MDAC 2.7 is not installed and is not selected to install, a dialog is shown (Figure 2-7), advising you that DB2 will not install MDAC 2.7 automatically. Click **OK** to proceed. To return and add MDAC 2.7, click **Cancel** on the next dialog.

Figure 2-7 MDAC 2.7 message box

7. A warning dialog will be shown, advising about changes in APPC protocol. Refer to *Release Notes regarding changes to DB2 UDB and DB2 Connection Version 8 APPC Support*. Proceed for details, clicking **OK**.
8. A NetBIOS configuration dialog is shown if NetBIOS protocol was selected. To configure it, just leave **Configure NetBIOS for connections to DB2 servers** checked; fill in workstation name and adaptor number. Click **Next**.
9. The **DB2 Setup Wizard language selection** dialog is shown. The English language is required to install DB2 products. If you need support on another language, left-click it on the **Available Languages** list to select and click **>** to add it to the **Selected Languages** list. This list is alphabetically sorted. Verify that the drive and path are correctly entered. Proceed by clicking **Next**.

10. Click **Install** to start the file copy. At the end of installation process, just click **Finish**. Depending on the system configuration, it could be necessary to reboot the machine. If DB2 Installation Wizard detects anything requiring a reboot, it will ask you to do so. Just reboot your machine.

For information on errors encountered during installation, see the db2.log file, which stores general information and error messages resulting from the install and uninstall activities. By default, the db2.log file is located in the *\MyDocuments\DB2LOG* directory. The location of the *MyDocuments* directory will depend on the settings on your computer.

Administration Client installation

The differences between Administration Client and Run-Time client are the Client Configuration Assistant (CCA), Java Administration Tools, and Business Intelligence tools.

DB2 Administration client has these components:

- **DB2 client engine files:** All the DB2 client engine files.
- **Administration Tools:** Control Center, Client Tools, Command Center, Configuration Assistant, Event Analyzer and DB2 Web Tools.
- **Application Development Tools:** Development Center, Warehouse Samples, and Spatial Extender Samples.
- **Java Development Kit:** SQLJ Application Development Tools and SQLJ Samples.
- **Business Intelligence:** Data Warehouse tools (Data Warehouse Center) and Information Catalog Manager tools (Information Catalog Center and Information Catalog Center for the Web).
- **Client support:** Interfaces (JDBC Support, MDAC 2.7, ODBC Support, OLEDB Support, SQLJ Support), Base Client Support, System Bind Files, Satellite, Synchronization, Spatial Extender Client, Java Runtime Environment, LDAP Exploitation, XML Extender and Communication Protocols (NetBIOS Listener, Named Pipes Listener, TCP/IP Listener).
- **Getting started:** First Steps application.

Step-by-step installation procedure

Follow these steps to perform the installation:

1. Place the DB2 Administration Client Install CD (ADMCL) in the drive and wait for the setup initial dialog. If the dialog is not shown or the install files are on the network, search and run **setup.exe** at the directory of DB2 installation. On the **IBM DB2 Setup Launchpad** dialog, click the **Install Products** tab.

2. On the **DB2 Setup Launchpad** dialog, make sure that the **DB2 UDB Administration Client** option is selected and click **Next**.
3. After you read the license agreement on the **DB2 Setup Wizard License Agreement** dialog, choose **I accept the terms of the license agreement** option and click **Next**.
4. The **DB2 Setup Wizard installation type** dialog is now shown requiring the type of the installation desired. There are three types of installation:

   Typical This option installs the administration tools and the client features that are used most often, including all required features, ADO, OLEDB, ODBC support, Control Center and CCA. Also, it will configure with default values. Leave **Data warehousing** checked if you need data warehousing tools.

   Compact This option installs only required client features and basic libraries to administrate and connect to DB2 servers.

   Custom This option lets you select the required client features.

   We continue the setup using the Custom option. Click **Custom** and click **Next** to proceed to the next step.
5. The **DB2 Setup Wizard installation action** dialog is shown.If you want to generate a script from the installation procedure, leave **Save your settings in a response file** checked. Make sure that **Install DB2 Administration Client on this computer** is checked. Otherwise, only the script file will be generated. Click **Next**.
6. The **DB2 Setup Wizard feature select** dialog is shown. Install MDAC 2.7 by opening **Administration Client–>Client Support->Interfaces**. Right-click **MDAC 2.7** and click **Add this package**. Proceed with adding or removing packages as required. Select the drive that has enough disk space to install all DB2 files by clicking **Drive**. You can check the amount of free disk space available on each drive of the system by clicking **Disk Space...**. You can change the path in the drive where the root directory of DB2 will be placed by clicking **Change...**. Click **Next** to advance.

   Note: ADODB and OLEDB components require MDAC 2.7. If MDAC 2.7 is not installed and if not selected to install, a dialog is shown, advising you that it will not install MDAC 2.7 if not installed if required. Just click **OK**. If you want to return and add MDAC 2.7, click **Cancel** on the next dialog**.**

7. A warning dialog will be shown, advising that the server will start to send and receive information to and from many ports during the datawarehouse setup. Just make sure that the ports (the default are 11000 to 11002) will allow connection to and from the server. Also, DB2 Setup Wizard advises about changes in APPC protocol. Refer to *Release Notes regarding changes to*

DB2 UDB and DB2 Connection Version 8 APPC Support. for details. Proceed by clicking **OK**.

8. A NetBIOS configuration dialog is shown if NetBIOS protocol was selected. To configure it, just leave **Configure NetBIOS for connections to DB2 servers** checked and enter the workstation name and adaptor number. Click **Next**.
9. The **DB2 Setup Wizard language selection** dialog is shown. The English language is required to install DB2 products. If you need support on another language, left-click the **Available Languages** list to select the language and click **>** to add it to the **Selected Languages** list. This list is alphabetically sorted. Verify if the drive and path are correctly entered. Proceed by clicking **Next**.
10. Click **Install** to start the file copy. At the end of installation process, just click **Finish**. Depending on the system configuration, it could be necessary to reboot the machine. If DB2 Installation Wizard detects anything requiring a reboot, it will ask you to do so. Just reboot your machine.

For information on errors encountered during installation, see the db2.log file, which stores general information and error messages resulting from the install and uninstall activities. By default, the db2.log file is located in the *\MyDocuments\DB2LOG* directory. The location of the *Documents* directory will depend on the settings on your computer.

Application Development Client installation

The Application Development Client is composed of the Administration Client and the add-ins for Visual Studio and the Development Center.

DB2 Application Development Client has these components:

- **DB2 client engine files:** All the DB2 client engine files.
- **Administration Tools:** Control Center, Client Tools, Command Center, Configuration Assistant, Database Tools, Event Analyzer and DB2 Web Tools.
- **Application Development Tools**: Development Center, Warehouse Samples, Spatial Extender Samples, Information Catalog Manager Sample, Java Development Kit, Sample Applications, SQLJ Application Development Tools, SQLJ Samples.
- **Java Development Kit:** SQLJ Application Development Tools and SQLJ Samples.
- **Business Intelligence:** Data Warehouse tools (Data Warehouse Center) and Information Catalog Manager tools (Information Catalog Center and Information Catalog Center for the Web).

- **Client Support:** Interfaces (JDBC Support, MDAC 2.7, ODBC Support, OLEDB Support, SQLJ Support), Base Client Support, System Bind Files, Satellite, Synchronization, Spatial Extender Client, Java Runtime Environment, LDAP Exploitation, XML Extender, and Communication Protocols (NetBIOS Listener, Named Pipes Listener, TCP/IP Listener).
- **Getting Started:** First Steps and XML Extender Samples.

Step-by-step installation procedure

Follow these steps to perform the installation:

1. Place DB2 Application Development Client Install CD (ADCL) in the drive and wait for the setup initial dialog. If the dialog is not shown, or the install files are on the network, search and run **setup.exe** at the directory of db2 installation. On **IBM DB2 Setup Launchpad** dialog, click the **Install Products** tab.
2. On **DB2 Setup Launchpad** dialog, make sure that the **DB2 UDB Application Development Client** option is selected and click **Next**.
3. After you read the license agreement on the **DB2 Setup Wizard License Agreement** dialog, choose **I accept the terms of the license agreement** option and click **Next**.
4. The **DB2 Setup Wizard installation type** dialog is now shown requiring the type of the installation desired. There are three types of installation:

   Typical This option installs the visual studio add-ins and the stored procedure builder, plus the administration tools to navigate through DB2 servers and objects, and the client features that are used most often, including all required features, ADO, OLEDB, ODBC support, Control Center and CCA. Also, it will configure with default values. Leave **Data warehousing** checked if you need data warehousing tools.

   Compact This option installs only the development basic features and client features to query and connect to DB2 servers.

   Custom This option lets you select the required client features.

   We continue the setup using the Custom option. Click **Custom** and click **Next** to proceed to the next step
5. The **DB2 Setup Wizard installation action** dialog is shown.If you want to generate a script from the installation procedure, leave **Save your settings in a response file** checked. Make sure that **Install DB2 Application Development Client on this computer** checked. Otherwise, only the script file will be generated. Click **Next**

6. The **DB2 Setup Wizard feature select** dialog is shown. Choose all the packages that you need. Select the drive that has enough disk space to install all DB2 files by clicking **Drive**. You can check the amount of free disk space available on each drive of the system by clicking **Disk Space...**. You can change the path in the drive where the root directory of DB2 will be placed by clicking **Change...**. Click **Next** to advance.

Note: ADODB and OLEDB components require MDAC 2.7. If MDAC 2.7 is not installed and if not selected to install, a dialog is shown, advising you that it will not install MDAC 2.7 if not installed if required. Just click **OK**. If you want to return and add MDAC 2.7, click **Cancel** on the next dialog.

7. A warning dialog will be shown, advising that the server will start to send and receive information to and from many ports during the Data Warehouse setup. Make sure that the ports (the default are 11000 to 11002) allow connection to and from the server. Also, DB2 Setup Wizard advise about changes in APPC protocol. Refer to *Release Notes regarding changes to DB2 UDB and DB2 Connection Version 8 APPC Support*. for details. Proceed clicking **OK**.
8. A NetBIOS configuration dialog is shown if NetBIOS protocol was selected. To configure it, leave **Configure NetBIOS for connections to DB2 servers** checked and enter workstation name and adaptor number. Click **Next**.
9. The **DB2 Setup Wizard language selection** dialog is shown. The English language is required to install DB2 products. If you need support on another language, left-click the **Available Languages** list to select the language and click **>** to add it to the **Selected Languages** list. This list is alphabetically sorted. Verify if the drive and path are correctly entered. Proceed by clicking **Next**.
10. Click **Install** to start the file copy. At the end of installation process, just click **Finish**. Depending on the system configuration, it could be necessary to reboot the machine. If DB2 Installation Wizard detects anything requiring a reboot, it will ask you to do so. Just reboot your machine.

For information on errors encountered during installation, see the db2.log file, which stores general information and error messages resulting from the install and uninstall activities. By default, the db2.log file is located in the *\MyDocuments\DB2LOG* directory. The location of the *Documents* directory will depend on the settings on your computer.

2.3 Installation profile

Installation profile is a DB2 feature which facilitates middle to large environment deployment. It provides a standardized way to install DB2 components and configure services and instance's parameters. DB2 Version 8 implements this with response files.

The step-by-step installation procedures described in the previous section will be used as a base for the examples of each installation response file generation. It is important to develop a nomenclature standard to store your response files in an organized way. We suggest a initial nomenclature standard that you can use. It is up to you to extend or redesign this to fit your needs.

If you want more information on how to work with response files, refer to *DB2 Installation and Configuration Supplement*.

Note: Response files should be handled confidentially, since they contain the user name and password in plain text.

Response file basics

Response files can contain the following information to do automatic installation and configuration:

- DB2 installation type
- DB2 products path
- DB2 accounts information
- Global DB2 registry variables
- Instance variables
- Instance database manager configuration settings

Here is a sample of DB2 Enterprise Server Edition's (ESE) response file provided with the DB2 ESE Install CD:

```
*DB2_USE_JDK12                  = BLANK, YES or NO
*DB2_VI_ENABLE                  = BLANK, ON or OFF
*DB2_VI_DEVICE                  = BLANK or char()
*DB2_VI_VIPL                    = BLANK or char()

* General information for instance to be created
* ----------------------------------------------
INSTANCE                        = DB2
DEFAULT_INSTANCE                = DB2
DB2.NAME                           = DB2
```

Comments are made by placing either a * or a # at the start of a line, or by placing ** or ## after the start of a line to comment out the rest of the line.

For descriptions of DB2 registry variables, please see *Appendix F* in the *Administration Guide*.

For descriptions of configuration parameters, please see *Chapter 20* in the *Administration Guide*.

Do not uncomment selected components (the COMP keywords) unless you change the INSTALL_TYPE to CUSTOM.

If you specify a default instance (DEFAULT_INSTANCE) in the response file, the USERNAME and PASSWORD keywords for the instance will be used for all the other USERNAME and PASSWORD values unless they are explicitly specified. Some of the variables that would be applied are the following:

- DAS_USERNAME
- DAS_PASSWORD
- MD_DB.USERNAME
- MD_DB.PASSWORD

If there is a DB2 process up and running which prevents the installation, DB2 will offer the shutdown services. However, letting DB2 shut down the running process may result in data loss. Make sure that all the DB2 processes are shut down gracefully, using a DB2 command such as **DB2STOP**, to prevent data loss.

These are the current three modes available to generate response files:

- **Manual generation:** You can create response files as a plain text file. DB2 installation CDs provide a sample of each product installation on server and client machines that can be used as a base for more elaborate installation procedures. They are on db2/*platforms*/sample path, where *platform* refers to the platform that you are currently working with. In our case it is *Windows.* These response files with default entries are available on the respective CD:

db2adcl.rsp	DB2 Application Development Client
db2admcl.rsp	DB2 Administration Client
db2conee.rsp	DB2 Connect Enterprise Edition
db2conpe.rsp	DB2 Connect Personal Edition
db2dlm.rsp	DB2 Data Links Manager
db2ese.rsp	DB2 Enterprise Server Edition
db2gse.rsp	DB2 Spatial Extender Server
db2lsdc.rsp	DB2 Life Sciences Data Connect
db2pe.rsp	DB2 Personal Edition
db2rcon.rsp	DB2 Relational Connect
db2rtcl.rsp	DB2 Run-Time Client
db2wm.rsp	DB2 Warehouse Manager
db2wmc.rsp	DB2 Warehouse Manager Connectors
db2wse.rsp	DB2 Workgroup Server Edition

- **DB2 Setup Wizard generation:** Check **Save your settings in a response file** check box when DB2 Setup Wizard asks if you want to do so (Figure 2-8).

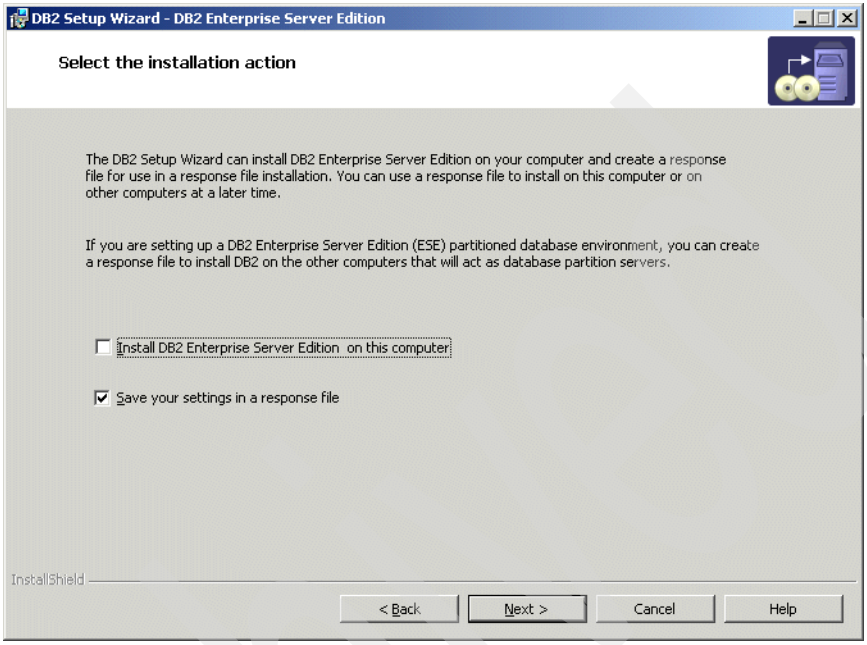

Figure 2-8 Generating a response file through DB2 Setup Wizard

Tip: Disabling product installation at the generate response file dialog in DB2 Setup Wizard is an easy way to generate the script with all the options selected.

- **Reverse-Engineer a current installation:** Use the command **`db2rspgn`** to reverse-engineer a specific installation. See chapter 6 of *DB2 Installation and Configuration Supplement Version 8* documentation for more details.

DB2 Setup Wizard parameters

DB2 Setup Wizard has the following parameters:

`/?`	Generates help message
`/f`	Force DB2 services to stop
`/i xx`	Install with preferred language, where xx means the language
`/l`	Full path and file name of log file
`/u`	Full path and file name of response file
`/t`	Creates a file with install trace information

Bundling software with DB2 packages

It is possible to bundle software installation with DB2 client and server tools by adding the DB2 Server and/or Client Install images with a response file to the software package. The response file included should use **`setup.exe`** (for client/server installation) or **`thnsetup.exe`** (for thin-client installation). Also, you can generate on-the-fly scripts according to your requirements while running server or client installers.

DB2 clients are free and can be downloaded from the IBM FTP site:

```
ftp://ftp.software.ibm.com/ps/products/db2/fixes
```

For server code, please check with the IBM marketing group regarding licensing issues.

Node and database configuration

DB2 response file installation doesn't do node and database client configuration. DB2 clients rely on LDAP to do so. But if you don't have an LDAP environment to configure your client, these are your options:

- **Replace the db2cli.ini file:** This is the easiest way to configure node and database information, but you will lose all customized information deleting db2cli.ini.
- **Do an import:** Run **`db2cfimp`** with a file that contains profile information.
- **Thin-client installation:** Provides automatic client updates without LDAP. When node and database configuration are modified at the server, the clients are automatically updated. Thin-client installation will be discussed next.

Detecting and killing a locked DB2 Setup Wizard process

DB2 Setup Wizard utilizes the Windows Installation Services. In the Windows environment, the process name of DB2 Setup Wizard is msiexec.exe. DB2 Setup Wizard can be locked if the installation script is not coded properly or the user accounts do not have proper authority. You can kill the process by TaskManager.

To prevent DB2 Setup Wizard from locking, do the following checks:

- Make sure you have a valid user account with the necessary authorities on the server where DB2 will be installed.
- Do a test install to see if some pop-up screen appear. Pop-up screens could freeze DB2 Setup Wizard during the installation. Most of them are usually about user authentication.

2.3.1 Server installation

In this section we provide a step-by-step procedure for response file type installation. As mentioned before, running DB2 Setup Wizard from a mapped drive will result in failure. For more information, refer to Chapter 7 in *Quick Beginnings for DB2 Servers, Version 8*.

Generating a response file script

Here are the steps you need to follow:

1. Execute steps 1 to 6 of server installation procedure on page 59. In step 7, choose carefully between typical and custom option, since the install profile are differently generated from these options. On our example, we proceed with the **Typical** option.
2. **Leave Save your settings in response file** checked and unchecked **Install DB2 Enterprise Server Edition on this computer** check box. Proceed to the next step by clicking **Next**.
3. Continue executing the steps from 8 to 15. In step 16, DB2 Setup Wizard will display a text box where you can enter the path and the file name of the script. We proceeded by entering c:\ese_typ_clean.rsp and then clicked **Finish**.

DB2 Setup Wizard generates only the necessary parameters in the response file. If you need to add more parameters, we recommend using response files provided in DB2 Install CDs, and modify them accordingly to your needs.

Performing installation using the response file script

Here are the steps you need to follow:

1. Place the DB2 ESE Install CD in the drive if DB2 install files are not available on the network. Open a new command-line prompt and execute:

   ```
   start /wait [PATH1]\setup.exe /U [PATH2]\db2_typ_clean.rsp /L
   [PATH2]\db2_typ_clean.log /T [PATH2]\db2_typ_clean.trc.
   ```

   PATH1 is the root directory of DB2 installation where setup.exe resides. PATH2 is the path where the response file is and where the log files will be placed. When done this way, the shell will wait for completion of the process, allowing you to add this command to a batch file and handle the return code in an adequate fashion. The default successful return code for DB2 Setup Wizard is 0.
2. During the install process, DB2 Setup Wizard will generate a temporary log file, db2wi.log. You can verify if DB2 Setup Wizard is working if this file grows. If something fails, open it and go to the last few lines to see what happened. After the return of the prompt, we recommend that you open it to see if there are any message codes or warnings. Remove it if you want to do so.

2.3.2 Client installation

In this section we provide a procedure for client installation using a response file. We also discuss another way to deploy DB2 clients: thin-client installation.

Generating a response file script

In this section we provide a step-by-step procedure for response file type installation. We take a DB2 application development client as a sample.

As mentioned before, trying to run DB2 Setup Wizard from a mapped drive will result in failure. For more information, refer to Chapter 7 in *Quick Beginnings for DB2 Servers, Version 8*.

Step-by-step procedure

Here are the steps you need to follow:

1. Execute steps 1 to 3 of the client installation procedure on page 67. In step 4, choose carefully between the Typical and Custom options, since the install profiles are differently generated from these options. In our example, we proceed with the **Custom** option.
2. Leave checked the check box **Save your settings in response file**, and leave unchecked **Install DB2 Application Development Client on this computer**. Proceed to the next step clicking **Next**.
3. Continue executing the steps from 6 to 10. In step 10, DB2 Setup Wizard will display a text box where you can inform the path and the file name of the script. We proceeded informing c:\adcl_cus_compl.rsp and them clicking **Finish**.

DB2 Setup Wizard generates only the necessary parameters in the response file. If you need to add more parameters, we recommend using the response files provided in DB2 Install CDs, and modify them accordingly to your needs.

Performing installation using the response file script

Here are the steps you need to follow:

1. Place the DB2 Application Development client Install CD in the drive if the DB2 install files are not available on the network. Open a new command-line prompt and execute:

```
start /wait [PATH1]\setup.exe /U [PATH2]\db2_typ_clean.rsp /L
[PATH2]\db2_typ_clean.log /T [PATH2]\db2_typ_clean.trc.
```

PATH1 is the root directory of DB2 installation where setup.exe resides. In our case this is \\Coral\udbsky\software\db2\v8_ga\nt\ese.

PATH2 is the path where the response file is and where the log files will be placed. In our case this is c:\ .

When done this way, the shell will wait for completion of the process, allowing you to add this command to a batch file and handle the return code in an adequate fashion. The default return code for DB2 Setup Wizard success is 0.

2. During the installation procedure, DB2 Setup Wizard will generate a temporary log file, db2wi.log. You can verify if DB2 Setup Wizard is working if this file grows. If something fails, open it and go to the few last lines to see what happened. After the return of the prompt, we recommend that you open it to see if there are any message codes or warnings. Remove it if you want to do so.

Thin-client installation

When you are dealing with lower capacity machines, local storage space is an important concern. Thin-client installation provides such users a DB2 client capability with small space footprint. When you use this option, you should be concerned with the initial applications startup, because the necessary files will be loaded from the network and you will experience a small delay.

Also, for some companies that don't have LDAP servers, the client configuration is still a burden to manage. Thin-client installation provides these users an option to centralize DB2 client configuration.

DB2 thin-clients load DB2 Administration Client from a code server and require a minimum amount of configuration and disk space on client machines. The code server can be updated, both the configuration and client version, and the clients will reflect this. The thin-client server is an option only available to the DB2 Administration client and will be displayed only if the **Custom** option is selected from the DB2 Setup Wizard installation type dialog.

The DB2 thin-client installation files are on the DB2 ADCL Install CD.

DB2 thin-client server only supports the following combination per each installation:

- Windows 98 and/or Windows 95
- Windows 2000, Windows XP, Windows Server 2003, and/or Windows NT

However, one server can serve both groups with a different path and installation. For more information on thin-client installation, refer to Chapter 7 in *Quick Beginnings for DB2 Servers, Version 8*.

2.4 Enterprise deployment with Microsoft SMS

The Microsoft Systems Management Server (SMS) software is designed to reduce the cost and administration associated with software deployment in large Enterprise environments with hundreds and thousands of computers. Using SMS to distribute DB2 UDB to client and server machines can significantly reduce your total cost of ownership (TCO). In this section we cover use of the Microsoft Systems Management Server software for large scale deployment of DB2 Universal Database in enterprise environments.

Before you can install DB2 UDB with SMS — or any software, for that matter — you must already have Microsoft SMS installed and configured on both your deployment SMS servers and the client workstations. The task of installing and configuring SMS will not be covered in this section, as it is beyond the scope of this book. The Microsoft *Systems Management Server Administrator's Guide* is a good reference to help you get started on this task.

2.4.1 Creating DB2 UDB packages

The first step is to create a package of the DB2 UDB software you plan to deploy. IBM has greatly simplified this task by providing a package definition file (pdf) for each edition of DB2 UDB on the product CD-ROM. You can create an SMS package using this supplied package definition file from the SMS Administrator Console by selecting **`Packages -> File -> New -> Package from Definition`**.

This will launch the Create Package from Definition Wizard (see Figure 2-9), and you will need to select the **Browse** button and browse your way to the provided package definition file (pdf) located in the \DB2\WINDOWS\SAMPLES directory on your product CD-ROM or hard drive.

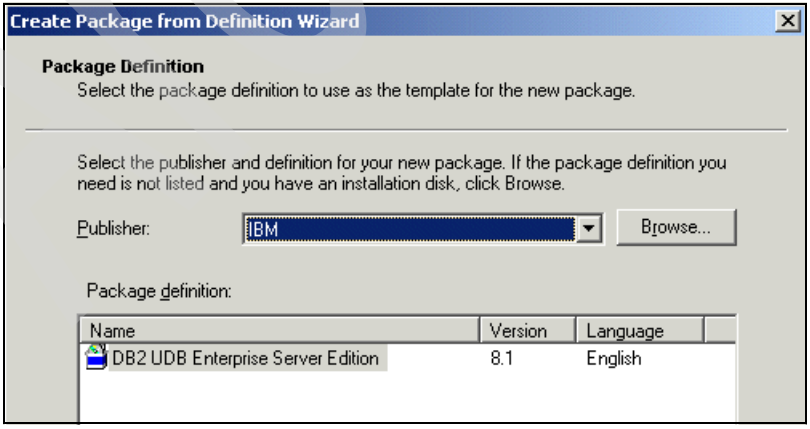

Figure 2-9 Create Package from Definition Wizard

Once the package has been created it will appear in the Packages container of SMS Administration Console. Figure 2-10 shows all of the DB2 UDB V8.1 packages that we have created using the package definition file shipped with each edition.

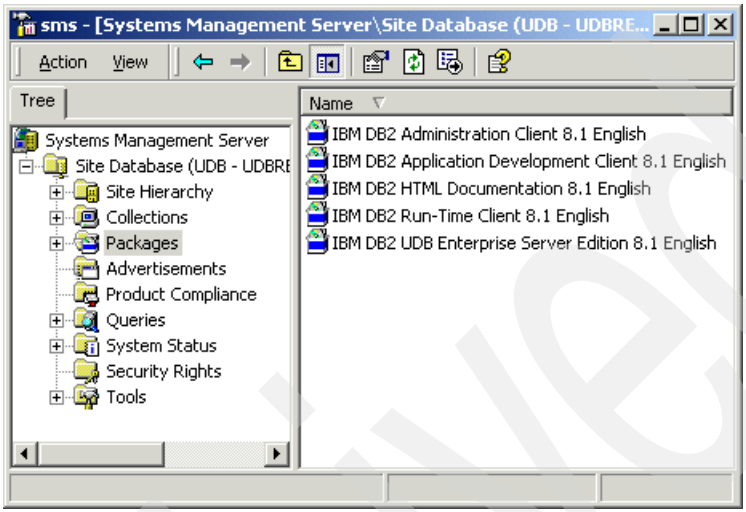

Figure 2-10 SMS Administration Console

You can find out more about each package by selecting the package and opening the Properties dialog; see Figure 2-11.

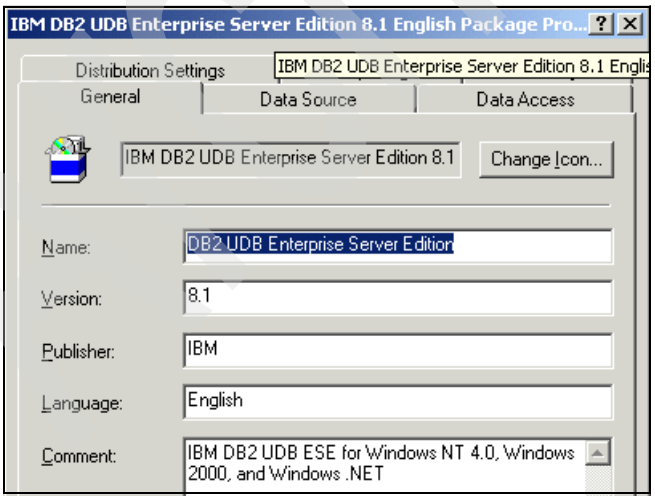

Figure 2-11 Package Properties

At this point you are ready to define Distribution Points, which can be easily accomplished using the SMS Administration Console's New Distribution Points Wizard. See Figure 2-12.

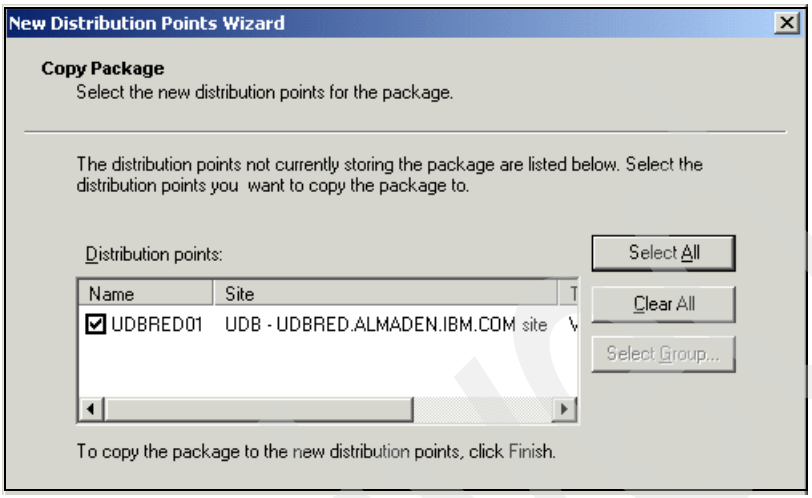

Figure 2-12 New Distribution Points Wizard

Once you have defined distribution points for all of your packages, you are ready to distribute the packages. This is accomplished using the SMS Administrators Console's Distribute Software Wizard. See Figure 2-13.

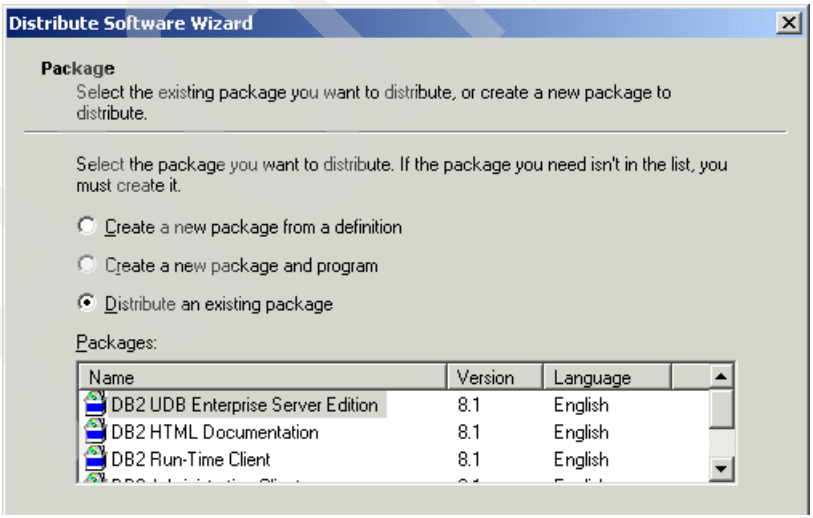

Figure 2-13 Distribute Software Wizard

2.5 Active Directory Services

The Windows 2000 Active Directory (AD) provides a central repository for managing network resources such as users, groups, and computer. In addition to these traditional network resources, the default Active Directory can be extended to store information about other types of resources on the network such as DB2 databases.

In this section we offer an overview of DB2 UDB's exploitation of the Windows 2000 Active Directory. We cover the following topics to completely implement DB2 UDB with Active Directory Services in a Windows 2000 Domain:

1. Extending the Active Directory for DB2
2. Installing the DB2 MMC Snap-In Extension
3. Enabling DB2 Active Directory support
4. Managing DB2 Active Directory resources

What's new: DB2 UDB V8.1 for Windows will provide a Microsoft Management Console (MMC) Snap-In Extension to the Active Directory Users and Computer MMC Snap-In. This MMC Snap-In Extension can be used to view and/or modify the DB2 node (ibm_db2Node) and database (ibm_db2Database) objects.

2.5.1 Active Directory Overview

Active Directory (AD) services provide a directory for resources in a Windows 2000 network. The goal of Active Directory services is to provide a central repository for managing network resources such as users, groups, and computer accounts. In addition to these traditional types of network resources, the Active Directory can be used to store information about other types of resources such as DB2 databases.

Active Directory services store information about network resources in a repository, simply called the directory, which can be published for easy access by users of the domain. It can be installed on Windows 2000 Server, Advanced Server or Datacenter Server editions. The process of installing the Active Directory services promotes the server to a Windows domain controller (DC).

In the Windows 2000 domain model, all servers running Active Directory services act as domain controllers. This is unlike the Windows NT domain model, in which a single primary domain controller (PDC) was supported by one or more backup domain controllers (BDC); see Figure 2-14.

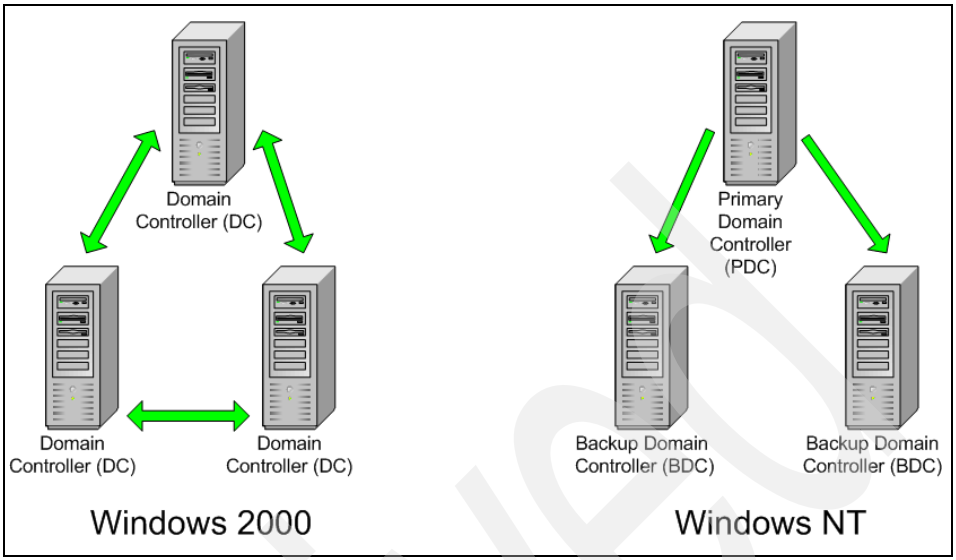

Figure 2-14 Active Directory Model

Another important distinction between the two domain models is that in the Windows 2000 domain model, all domain controllers maintain an updatable copy of the directory that is replicated to and from other domain controllers in the domain. This is an improvement over the Windows NT domain model, in which only the primary domain controller maintained an updatable copy and the backup domain controllers supported only a read-only copy.

The Active Directory provides support for industry standard protocols such as DHCP, DNS, SMTP, and LDAP. DB2 stores information in the Windows 2000 Active Directory using the Lightweight Directory Access Protocol (LDAP) to interface with Active Directory services.

The primary advantage in using Active Directory services to manage DB2 resources is that it provides a central repository for maintaining both database and instance directories. With Active Directory services it is no longer necessary to catalog individual databases or instances on each and every client on the network.

Additionally, the maintenance of database directories can be reduced substantially. For example, if a database is physically moved from one database server to another, only the DB2 database directory at the Active Directory server need be updated. After this simple procedure all DB2 clients will have access to the updated database directory.

Another unique feature that DB2 exploits with the Active Directory is the ability to store DB2 Registry Profiles at the user level. With user level DB2 Registry profiles, each individual user can have a customize DB2 environment.

2.5.2 Extending the Active Directory

In order to use the Windows Active Directory to store information about DB2 resources on the network, the Active Directory must first be modified to include DB2 object classes and attributes. This process is commonly referred to as extending the Active Directory schema or Active Directory schema extension.

The objects in the Active Directory schema are secured by access control lists (ACLs) so that only authorized users can alter the schema.

DB2 Schema Extension Utility

DB2 provides a command line based utility called the DB2 Schema Extension (*db2schex.exe*) to extend the Active Directory schema so that it includes DB2 object classes and attributes.

This utility can be found in the *%db2path%\bin* directory, if you have already installed DB2 UDB as well as on the DB2 UDB Enterprise Server Edition V8.1 for Windows Operating Environments product CD-ROM in the directory *\db2\windows\utilities*.

The *db2schex* has the flowing syntax:

```
Db2schex [ -b -w -u -k ]
```

Note: In order to extend the Active Directory schema, you must be a member of the Schema Admins group or have been delegated the rights to update the schema.

2.5.3 Installing the MMC Snap-In Extension

In order to use the Microsoft Management Console (MMC) Snap-In Active Directory Users and Computers to manage DB2 nodes and databases, DB2's extensions to this Active Directory Users and Computers Snap-In must be installed and registered on a Windows 2000 Domain Controller (DC).

The following command can be used to register the DB2's Snap-In Extension for the Active Directory Users and Computer MMC on a Windows 2000 Domain Controller (DC) that already has DB2 UDB V8.1 for Windows installed:

```
Regsvr32 %db2path%\bin\db2ads.dll
```

If you would like to use this snap-in extension for the Active Directory Users and Computer MMC from a domain controller that does not have DB2 UDB installed, you must first copy the DB2 Active Directory Services extension file (db2ads.dll) and resource file (db2adsr.dll) to a local directory and then register the DB2 Snap-In Extension. See Figure 2-15.

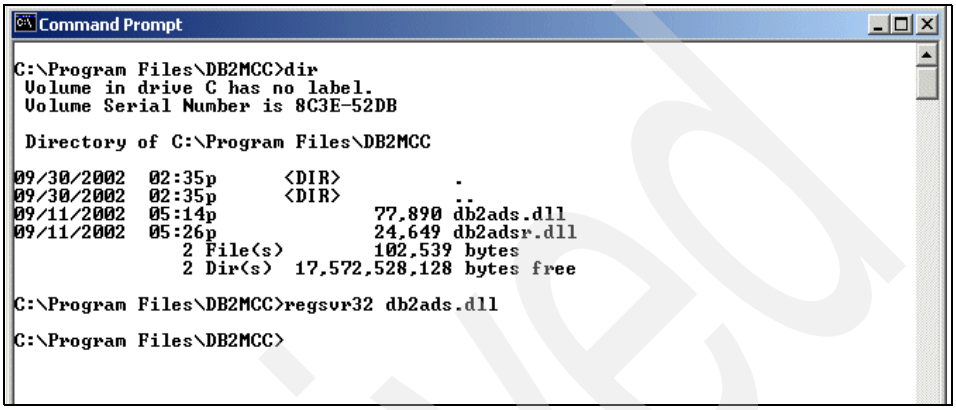

Figure 2-15 DB2 Snap-in Extension files

Figure 2-16 shows a screen capture of the Active Directory Users and Computers Microsoft Management Console (MMC) Snap-In after DB2's Snap-In Extension has been installed and registered. We discuss this MMC in more detail later in this section.

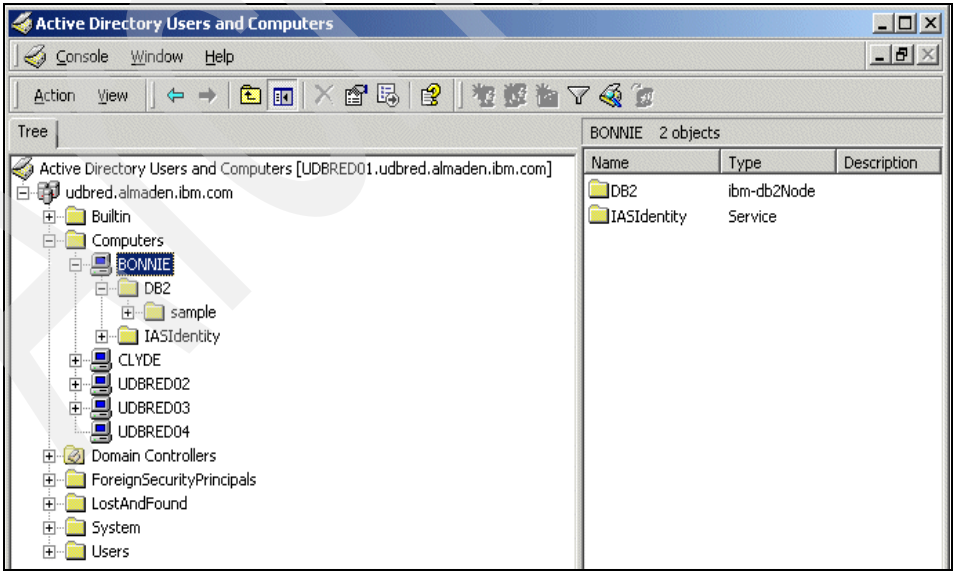

Figure 2-16 Active Directory with DB2 Sanp-in Extension installed

2.5.4 Enabling DB2 Active Directory support

In the previous section we discussed how to extend the Active Directory schema so that DB2 resources are defined within the directory schema. In this section we discuss how to enable both DB2 servers and DB2 clients to begin using the Active Directory services as a repository for DB2 resources.

In order for DB2 clients and servers to use the Active Directory they must (1) belong to a Windows 2000 domain, (2) have the Microsoft LDAP client installed, (3) have DB2 enabled for LDAP support, and finally (4) must logon to the Windows 2000 domain.

DB2 uses the Lightweight Directory Access Protocol (LDAP) to interface with Active Directory services. The procedure for enabling both DB2 clients and server to use the Active Directory services requires only one DB2 Registry Variable called DB2_ENABLE_LDAP to be defined.

The **DB2SET** command for this registry variable has the following syntax:

```
DB2SET DB2_ENABLE_LDAP=YES
```

On DB2 servers this registry variable must be set in order to register or de-register a DB2 server with the Active Directory. On DB2 clients this registry variable must be set in order to connect to a database that exists in the Active Directory.

Note: The installation wizard of DB2 UDB V8.1 for Windows will automatically set the DB2_ENABLE_LDAP registry variable for the default DB2 instance (DB2) on both clients and servers, if the wizard detects the presence of the Windows LDAP client support (wldap32.dll).

Active Directory cache

DB2 provides a mechanism for caching node and database directory information in a private memory area for each application. This cache is enabled or disabled with the instance (DBM) configuration parameter called *dir_cache*. By default the directory cache is enabled (*dir_cache=yes*) which minimizes the time required to search the directory by reducing directory file I/O.

The DB2 node and database catalog information from the Active Directory can also be stored in the local directory cache by enabling the DB2 Registry variable, DB2LDAPCACHE. The **DB2SET** command for this registry variable has the following syntax:

```
DB2SET DB2LDAPCACHE=YES
```

The DB2 node and database information obtained from Active Directory services can be manually refreshed by invoking the **DB2 REFRESH LDAP** command. The syntax to refresh DB2 node entries in the local directory cache is as follows:

```
db2 refresh ldap node directory
```

The syntax to refresh DB2 database entries in the local directory cache is as follows:

```
db2 refresh ldap database directory
```

Note that refreshing the local LDAP node and database directory removes these entries from the private directory cache, forcing them to be refreshed which can add additional overhead to an applications connect time if the entry resides on another Windows 2000 Domain.

Active Directory search

DB2 provides a registry variable which controls how extensive of an Active Directory search should be performed when attempting to locate a database or node resource maintained within the Active Directory. The syntax for the DB2LDAP_SEARCH_SCOPE registry variable is as follows:

```
DB2SET -gl DB2LDAP_SEARCH_SCOPE=search_scope
```

The values for the search_scope can be “local”, “domain”, or “global” and the default value is “domain” in which all domain trees are searched.

Note: DB2LDAP_SEARCH_SCOPE can only be set at the global LDAP level (-gl). This is the only registry variable that can and must be set at the global LDAP level.

Active Directory Profiles

DB2 UDB also exploits the Windows 2000 Active Directory to create user specific DB2 Registry Profiles. This provides for a customized DB2 environment for each individual user. DB2 Registry Variables can be stored in the directory at the user level by specifying the -ul option:

```
DB2SET -ul registry_variable=registry_value
```

The DB2 profile registry variables at the user level are cached on the local machine. The user level entries in this cache can be manually refreshed using the following command:

```
Db2set -ur
```

2.5.5 Managing the Active Directory

Once the Active Directory schema has been extended and DB2 LDAP support enabled as discussed in the previous sections, DB2 will automatically update the Active Directory. However, in some cases you may need to manually add, update, or remove a resource from the Active Directory.

Managing DB2 Servers

IBM provides several DB2 commands that can be used to manually update DB2 node and database information in the directory if needed. These commands are discussed in this section.

Note: If you elect to create the default DB2 instance (DB2) during the install process, the installation wizard may automatically register the node with Active Directory services using the computers name which will be truncated to 8 characters if needed.

In the likely event that DB2 instances have already been created on one or more servers prior to enabling DB2 to use the Active Directory services, IBM provides commands to register the DB2 instance with the Active Directory services:

- **REGISTER**
- **DEREGISTER**
- **UPDATE LDAP**
- **CATALOG LDAP**

Register DB2 Server

The **REGISTER** command can be used to register a DB2 instance in the Active Directory. The **REGISTER** command can be invoked for either the current DB2 instance or for a remote DB2 instance.

If invoking the **REGISTER** command for the current DB2 instance, the syntax is as follows:

```
db2 register db2 server in ldap as <ldap_node_name > protocol tpcip
```

If invoking the **REGISTER** command for a remote DB2 instance, the syntax is as follows:

```
db2 register db2 server in ldap as <ldap_node_name > protocol tpcip
hostname <host_name> svcename <service_name> remote <computer_name>
instance <instance_name>
```

Note: In order to register a DB2 instance, the DB2 configuration parameter svcename must be valid.

Deregister DB2 Server

The **DEREGISTER** command is used to remove a DB2 node entry from the Active Directory. The syntax for the **DEREGISTER** command is as follows:

```
db2 deregister db2 server in ldap node <ldap_node_name>
```

The **DEREGISTER** command will remove the DB2 node from the Active Directory as well as any databases that have been cataloged to this node.

Update LDAP Node

In the event that the registration information within the Active Directory needs to be updated, IBM provides a DB2 command to do so, using the **UPDATE LDAP NODE** command:

```
db2 update ldap node <ldap_node_name > hostname <host_name> svcename
<svcename>
```

Catalog LDAP Node

In the event that a node alias is required for a specific application, IBM provides the DB2 command as well using the **CATALOG LDAP NODE** command as follows:

```
CATALOG LDAP NODE node-name [AS node-alias] [USER username [PASSWORD
password]
```

Managing DB2 databases

For the most part, database entries in the Active Directory are automatically maintained by DB2 when the database is either created or dropped. However in the event that the database was created prior to enabled Active Directory support, IBM provides two commands to **CATALOG** and **UNCATALOG** database within the Active Directory.

Catalog LDAP database

Normally, the database will be registered in Active Directory when it is created, as long as the AD schema has already been extended, the DB2 LDAP support enabled, and the DB2 instance registered with Active Directory.

However, if during the process of creating the database, the registration of the database in Active Directory fails (perhaps the database name already exists in the Active Directory), you may need to manually register the database in the Active Directory using the **catalog ldap database** command, whose syntax is as follows:

```
catalog ldap database <database_name> at node <node_name> with "comment"
```

Uncatalog LDAP database

Under normal circumstances the database is automatically removed from the Active Directory when either the database is dropped or the instance it belongs to is deregestered, which can also happen when the instance is dropped. If needed, the database can be manually uncataloged using the **`uncatalog ldap database`** command, whose syntax is as follows:

```
db2 uncatalog ldap database <database_name>
```

Using the Microsoft Management Console

In the previous section we covered how to install and register DB2's MMC Snap-In Extension for the Active Directory for Users and Computers. In this section we explore the snap-in more detail.

Microsoft Management Console (MMC) provides a facility for hosting administration tools, called Snap-Ins, used to mange just about everything from user and group accounts to individual services on computers on the network. Some common examples of MCC Snap-Ins include: Event Viewer, Local Users and Groups, Performance Logs and Alerts, and Services. In fact there is even a MCC Snap-In that can be used to browse and extend the Active Directory, its called the Active Directory Schema Snap-In.

The screen capture in Figure 2-16 on page 88 shows the Active Directory Users and Computers MMC Snap-In after it has been extended using the DB2 Schema Extender utility (db2schex.exe) discussed in the previous section.

Note: In order to view the DB2 node and database objects in the Active Directory Users and Computers MCC Snap-In you must first change the view by selecting **View —> Users, Groups, and Computers as containers** and **View —> Advanced Features**.

The DB2 Node (ibm-db2Node) and DB2 Database (ibm-db2Database) object class properties can be viewed by selecting the object in the MCC, right clicking, and finally selecting the Properties pop-up menu item.

The **Object** tab of the DB2 Properties dialog in Figure 2-17 shows the date and time the DB2 node (ibm-db2Node) was created and last modified.

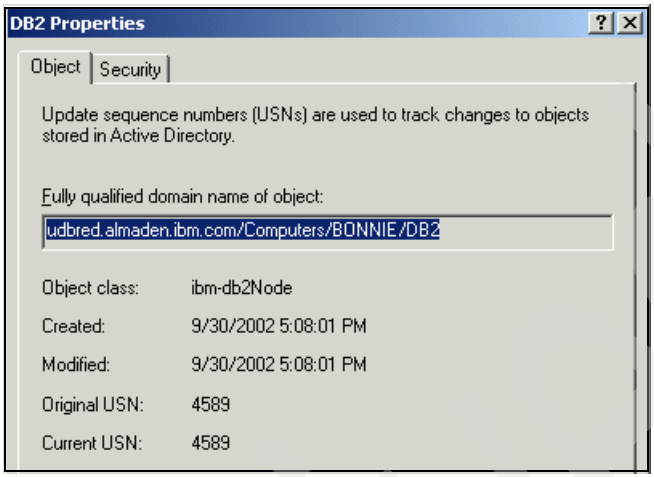

Figure 2-17 DB2 Node object properties

The **Security** tab of the DB2 Properties dialog in Figure 2-18 shows the individual permissions that each groups and/or user has on the DB2 node (ibm-db2Node) object. Notice that in this example, all “Authenticated Users” have only “Read” access to the object in the Active Directory.

.

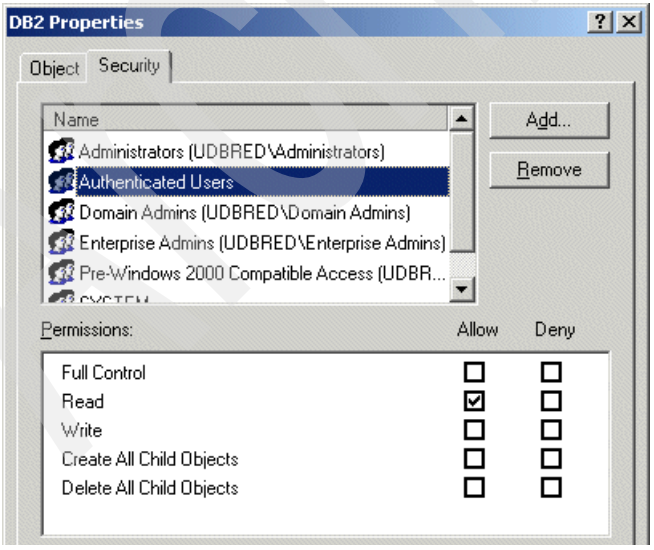

Figure 2-18 DB2 Node security properties

3

Post-installation tasks

In this chapter we discuss the post installation tasks that should be performed to have a functional, well-performing database, without having to get into detailed performance tuning as discussed in Chapter 5, “Performance” on page 191. We highly recommend that these tasks be done by every installation, since the default configuration parameters supplied by DB2 only satisfy the needs of a minimal system. We also discuss several methods for populating a database.

Setting proper values for database manager/database configuration parameters and creating appropriate indexes can achieve a significant performance improvement. We discuss how this can be done using the Configuration Advisor Wizard, and explain the role of the Control Center as the central point of administration for the DB2 Universal Database.

This chapter covers the following topics:

- Using the Control Center
- Database creation
- Configuration advisor
- Populating your database
- Design Advisor Wizard

3.1 Introduction

Setting proper values for database manager/database configuration parameters and creating appropriate indexes can achieve a significant performance improvement; however, this task is not easy if you are a new DBA. DB2 UDB provides the Configuration Advisor Wizard, which helps you to tune performance related configuration parameters by requesting information about the database, its data, and the purpose of the system. For creating indexes, DB2 provides the Design Advisor Wizard, which you can use to determine which indexes to create or drop for a given set of SQL statements.

These wizards or Smart Guides, as they are sometimes called, can be invoked from the Control Center. The Control Center is the central point of administration for the DB2 Universal Database. The Control Center provides the user with the tools necessary to perform typical database administration tasks. It allows easy access to other server administration tools, gives a clear overview of the entire system, enables remote database management, and provides step-by-step assistance for complex tasks.

If you will be running in an IBM WebSphere environment, we recommend that you follow the guidelines in the redbook, *DB2 UDB/WebSphere Performance Tuning Guide*, available from the IBM Redbook Web site at:

`http://www.ibm.com/redbooks`

3.2 Using the Control Center

The DB2 Control Center is the tool used for the remainder of this chapter. It is made available when it is selected as an option during custom installation or by default when typical installation is chosen.

You can start the Control Center in the following ways:

- Select **Control Center** from the Tools menu of another tool.
- Click the icon from the toolbar of another tool.
- Enter the **db2cc** command in Command Prompt Window.
- On Windows systems, click the **Start** button and select **Programs —> IBM DB2 —> General Administration Tools —> Control Center.**

You use the Control Center to manage systems, DB2 Universal Database instances, DB2 Universal Database for OS/390 and z/OS subsystems, databases, and database objects such as tables and views.

Figure 3-1 shows an example of the information available from the Control Center. In our example, the Control Center is started at a Windows 2000 server, with one local instance, and using the DB2 sample databases.

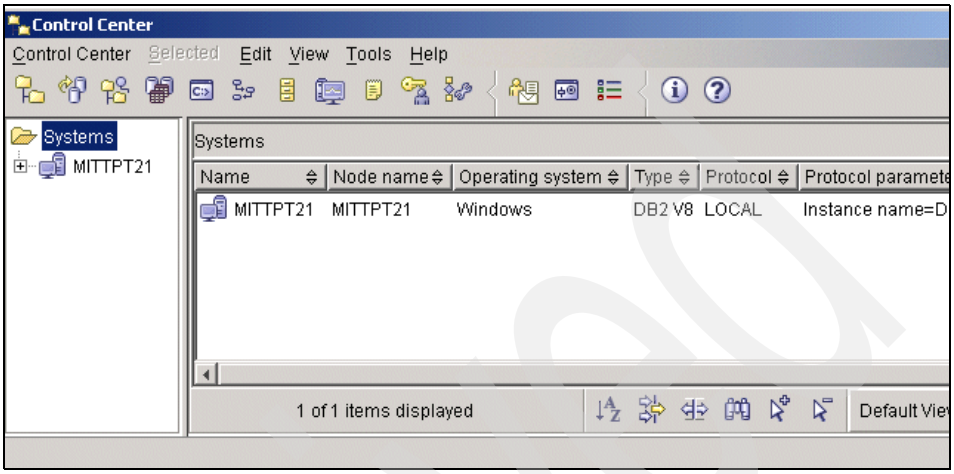

Figure 3-1 Control Center initial screen

The Systems object represents the local machine. To display all the DB2 systems that your system has cataloged, expand the object tree by clicking the plus sign (+) next to Systems. The left portion of the screen lists available DB2 systems. We can see from Figure 3-2 that the system MITTPT21 contains an instance, DB2. The instance DB2 has four databases. When Systems is highlighted, details about each system are shown in the Contents Pane. We can see that this is a Windows system.

The main components of the Control Center are listed below:

- **Menu Bar:** This is used to access the Control Center functions and online help.
- **Tool Bar:** This is used to access the other administration tools.
- **Objects Pane:** This is shown on the left-hand side of the Control Center window. It contains all the objects that can be managed from the Control Center as well as their relationship to each other.
- **Contents Pane:** This is found on the right side of the Control Center window and contains the objects that belong or correspond to the object selected on the Objects Pane.
- **Contents Pane Toolbar:** These icons are used to tailor the view of the objects and information in the Contents pane. These functions can also be selected in the View menu.

Hover Help is also available in the Control Center, providing a short description for each icon on the tool bar as you move the mouse pointer over the icon.

If you want to see each database object in detail, you can expand a database icon and list each database object by clicking the plus sign (**+**) next to a database icon.

3.3 Database creation

Before any activities can be performed on a database, it must first be created. With DB2 UDB, a database can be created one of the following ways:

1. Use the Command Line Processor CREATE DATABASE command.
2. Use the First Steps dialog that creates DB2's sample databases.
3. Use the Control Center to create a database from a database backup.
4. Use the Control Center Create Database wizard.

We now discuss the Control Center Create Database wizard. To use the wizard, perform the steps described in the following sections.

Start Create Database wizard

The following steps show you how navigate to Create Database wizard under Control Center.

Start the Control Center

Click the **Start** button and select **Programs -> IBM DB2 -> General Administration Tools -> Control Center**.

Expand objects to show databases

Figure 3-2 illustrates the expansion of the objects in the Control Center to show the database objects.

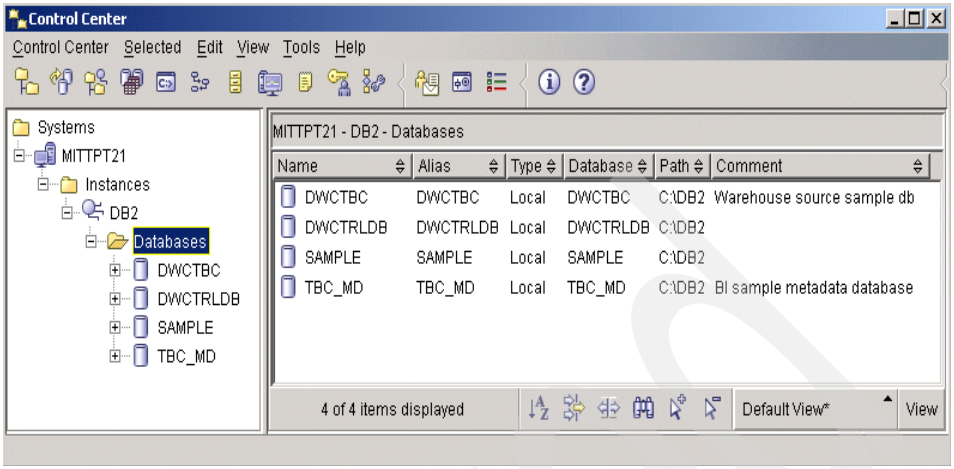

Figure 3-2 Expand objects to show databases

Select Create Database wizard

Figure 3-3 shows the steps needed to start the Create Database wizard. To start the wizard, perform the following:

Click **Databases** under the DB2 Instance, depress mouse button two, and select **create --> database using wizard**.

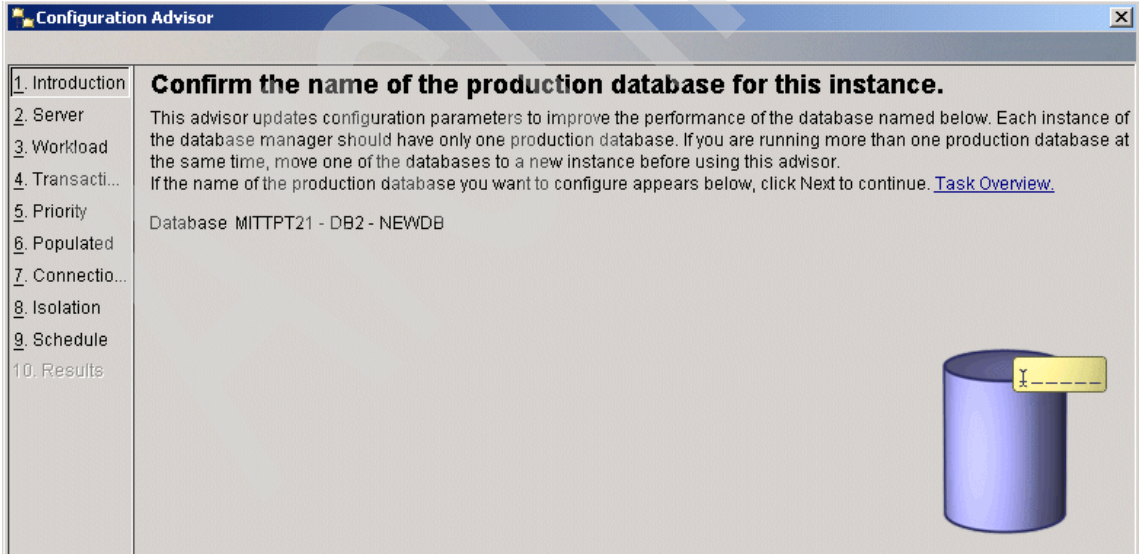

Figure 3-3 Select Create Database wizard

Provide information to Create Database wizard

Follow these steps to create a database using the Create Database wizard.

Specify a name for your new database

Figure 3-4 shows the initial information you need to provide to the wizard, such as the name of the database to be created. In our example we use the name NEWDB. We also need to provide the drive letter that the database will reside on, in our example drive C:. This drive sets the default where database objects will be created; this can be overridden later for specific database objects. The Alias parameter provides an alternate name for the database that is used by users or applications to connect to the database. In most cases this should be the same as the database name. The Comment parameter provides text to describe the database.

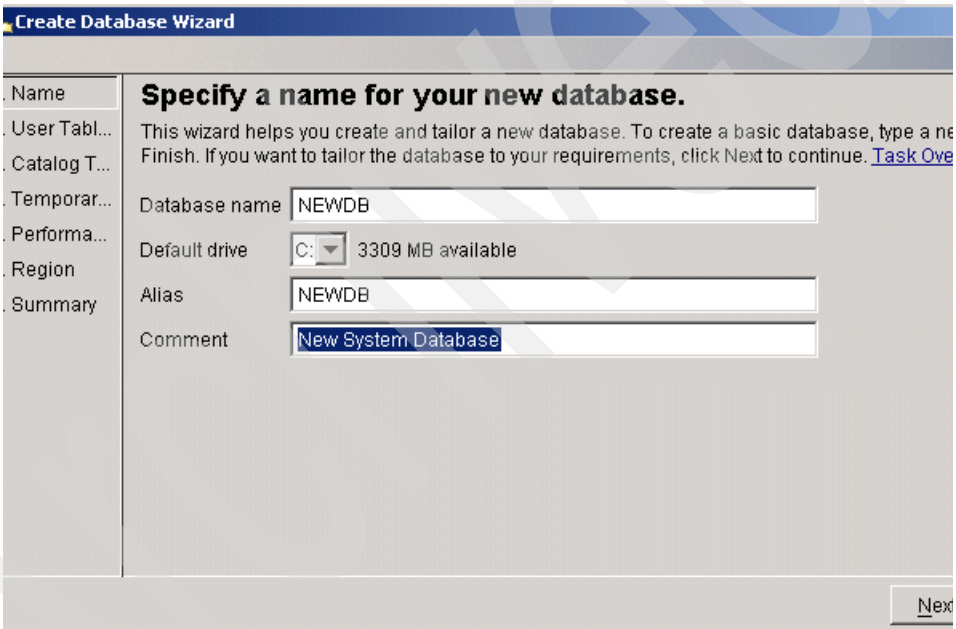

Figure 3-4 Specify a name for your new database

Specify how and where to store the user tables

Figure 3-5 shows how we can override the default set in Figure 3-4 on page 100 for the storage of user tables in the database. We can also select how the database manager should manage user tables, either System Managed or Database Managed. For our example we use the default of System Managed. Individual tables can be created later that use Database Managed Space.

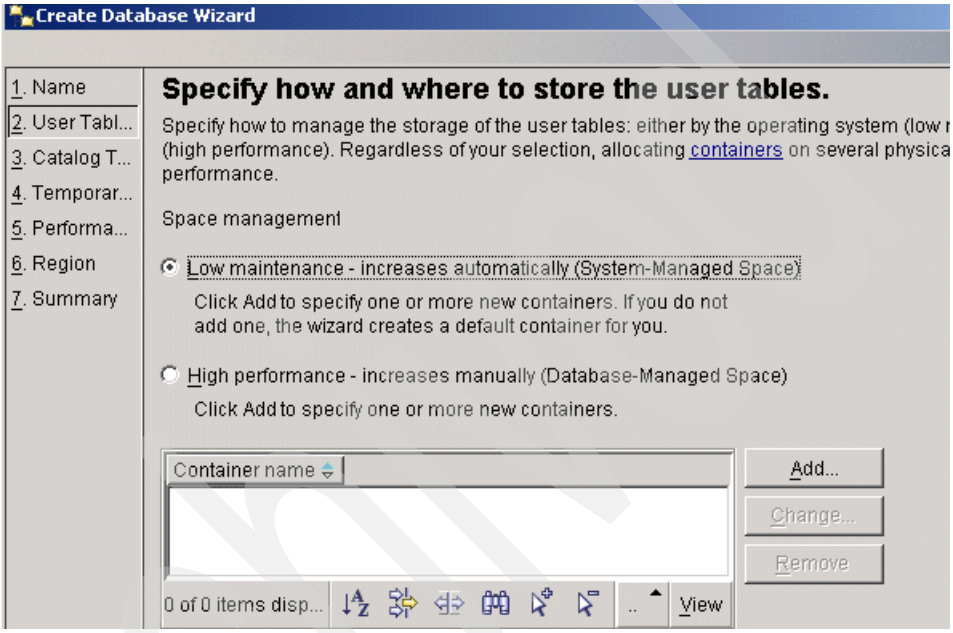

Figure 3-5 Specify how and where to store the user tables

Specify how and where to store the system catalog tables

Figure 3-6 shows how we can specify how to manage the storage of the system catalog tables. The system catalog tables contain information such as descriptions of indexes, tables, views, and packages. Since the system catalog will be used often, specify containers on the fastest drives available. For our example we continue with the default.

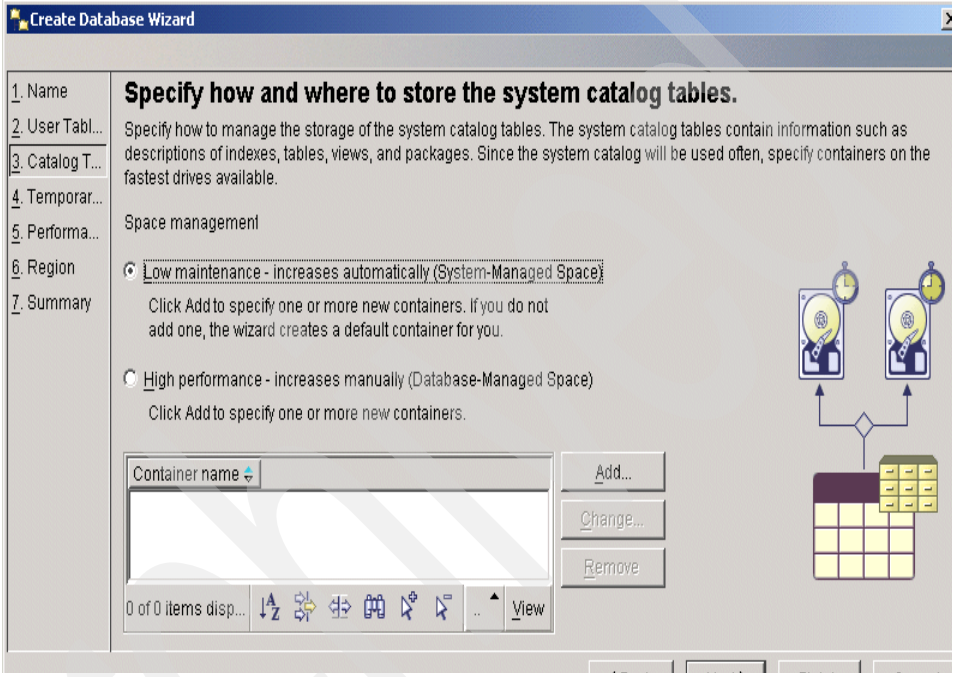

Figure 3-6 Specify how and where to store system catalog tables

Specify how and where to store system temporary tables

Figure 3-7 shows how to specify how to manage the storage of the system temporary tables. Temporary storage is used for activities such as sorting and reorganizing tables, creating indexes, and joining tables. The more space you allocate for temporary storage, the faster these tasks will run. If you need more space later, you can add it. If the applications using your database do a large amount of sorting or table joins, it is recommended that the system temporary tables be placed on drives that do not contain user data or the system catalog tables.

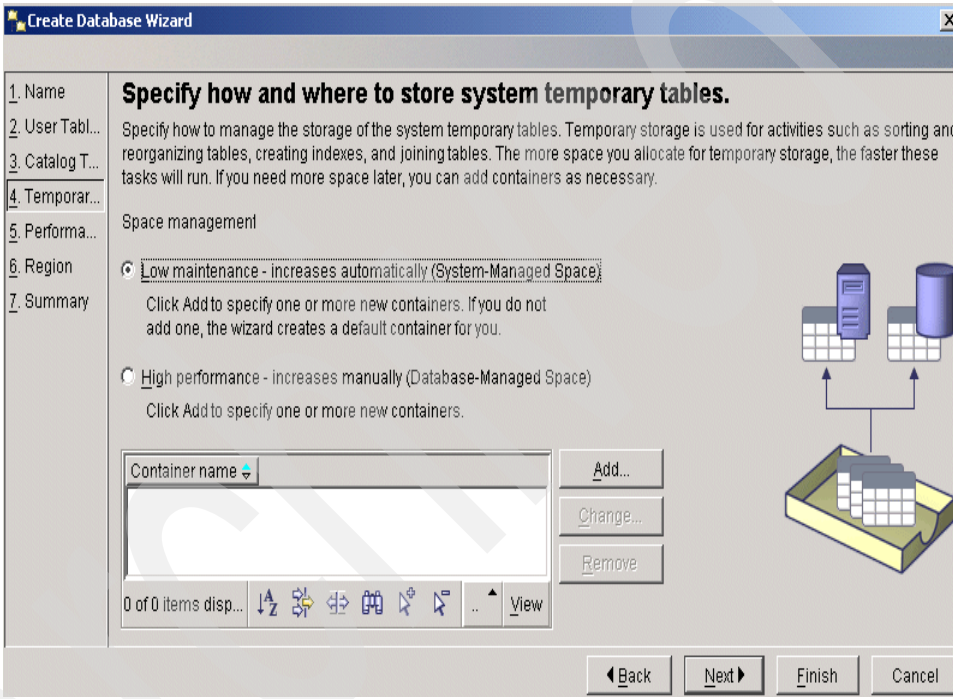

Figure 3-7 Specify how and where to store system temporary tables

Tune the performance of this database

Figure 3-8 shows how we can perform some initial tuning on how the database reads and writes data. The defaults are normally satisfactory, but can be changed for individual tables later if needed.

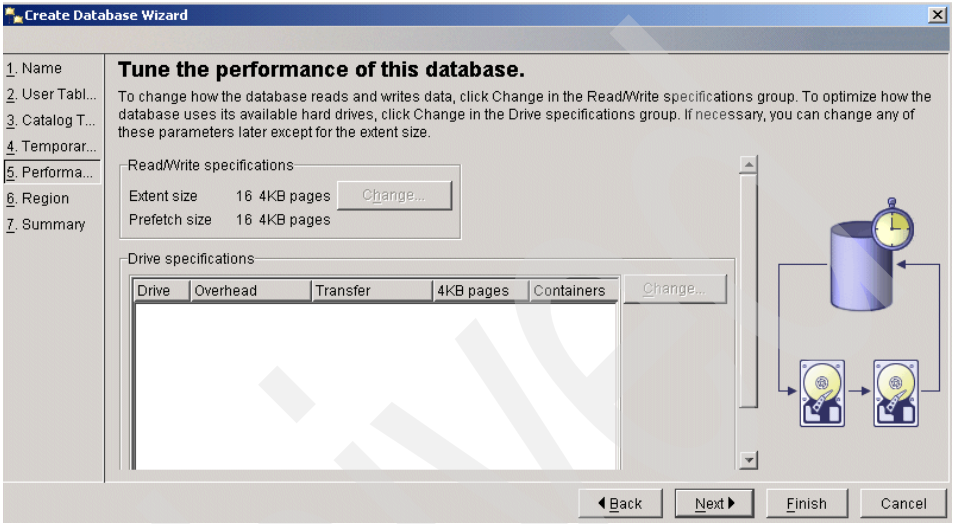

Figure 3-8 Tune the performance of this database

Specify the locale for this database

Figure 3-9 shows how we can override the operating system and database defaults for the territory/code set and collating sequence. We use the defaults, which are normally satisfactory.

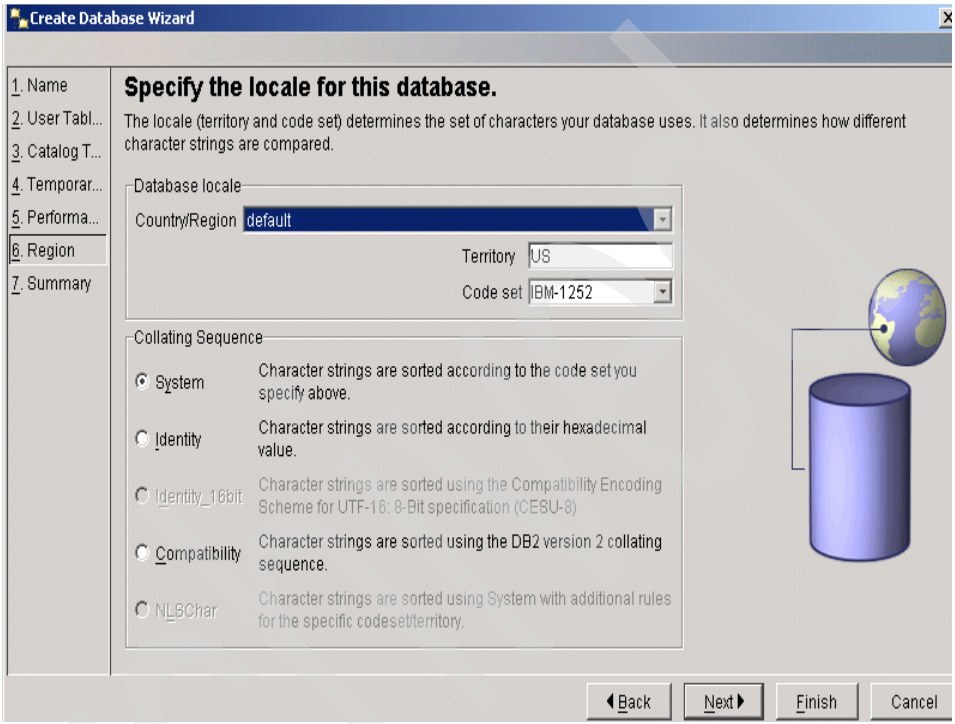

Figure 3-9 Specify the locale for this database

Review the actions that will take place when you click Finish

Figure 3-10 shows a review screen summarizing the actions that will be taken when creating the database. To create the database, select **Finish**.

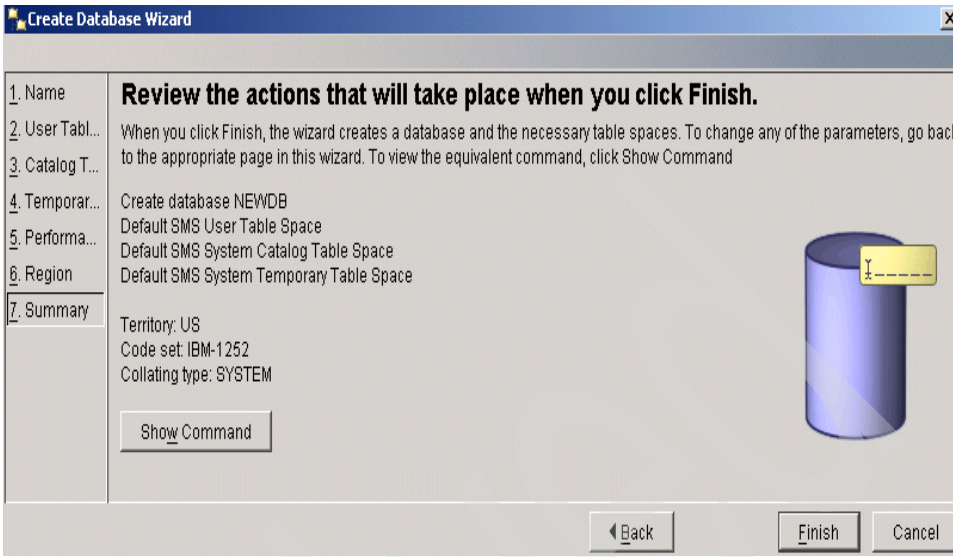

Figure 3-10 Review the actions that will take place when you click Finish

Proceed to Configuration Advisor

After the database is created with the Create Database wizard, you are presented with the option to execute the Configuration Advisor as shown in Figure 3-11. Select **Yes** to proceed with executing the Configuration advisor.

I

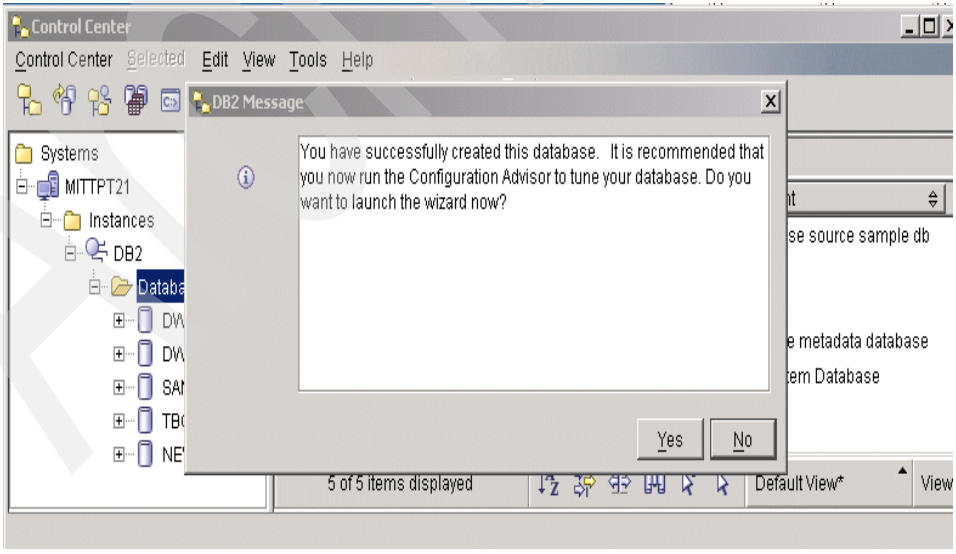

Figure 3-11 Proceed to the Configuration Advisor

3.4 Configuration advisor

The Configuration Advisor provides an easy-to-use tool to tune a DB2 UDB system with minimal input. It is a very good starting point for achieving satisfactory database performance for most workloads.

The Configuration Advisor updates configuration parameters to improve the performance of the named database. Each instance of the database manager should have only one production database. If you are running more than one production database at the same time, you need to move one of the databases to a new instance before using the Configuration Advisor.

We step you through the tool, providing some guidance on what input to provide. We assume you selected **Yes** to running the Configuration Advisor at the completion of the database creation step discussed previously.

Confirm the name of the production database for this instance

The first screen of the Configuration Advisor shown in Figure 3-12 confirms that the correct database has been chosen to base the tuning parameters on. For our example, it should specify the name NEWDB that we created above. If the name is correct, select **Next**.

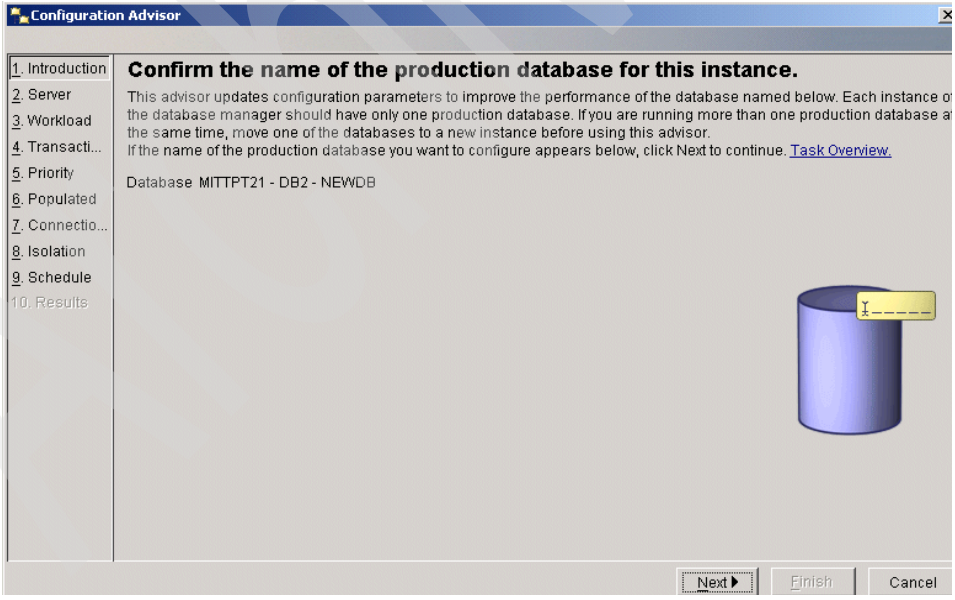

Figure 3-12 Confirm the name of the production database for this instance

Specify server memory for the database manager to use

Use the slider bar to set target values for server memory (RAM) as shown in Figure 3-13. If other applications are running on this server, set the slider bar to less than 100%. The currently available memory is affected by all applications that are running right now, including DB2.

Caution must be used when setting the percentage, since the parameter settings resulting from running the Configuration Advisor will try to utilize all the memory you specify. If the value is overstated, system performance can be severely impacted by causing system swapping from the over commitment of system memory.

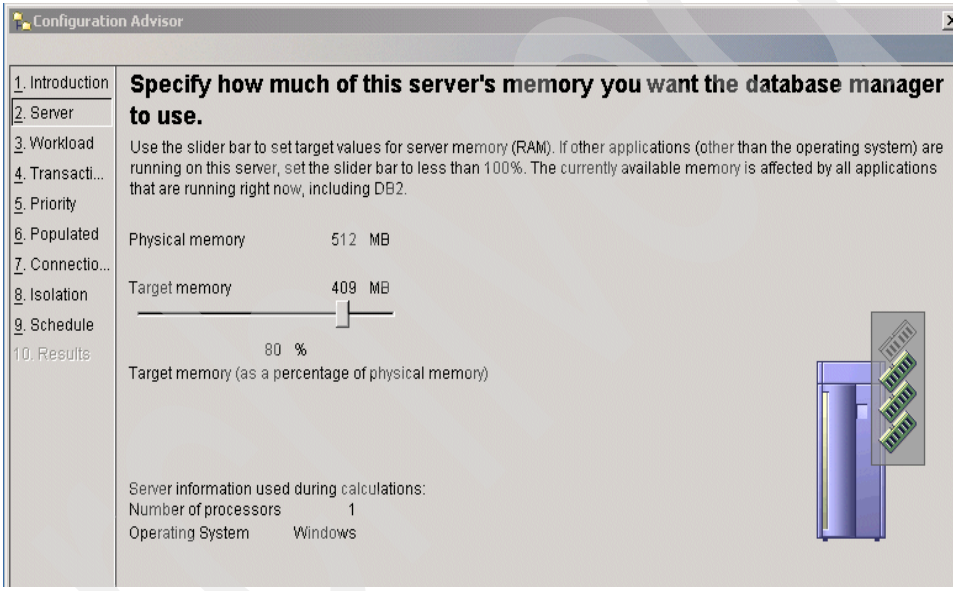

Figure 3-13 Specify server's memory for the database manager to use

Select the type of workload that reflects your database

Figure 3-14 shows how you can optimize your database for a particular type of workload. If your database is mostly used for queries, select **Queries**. If it is mostly used for transactions, select **Transactions**. If you are unsure, or it is mixed, select **Mixed**.

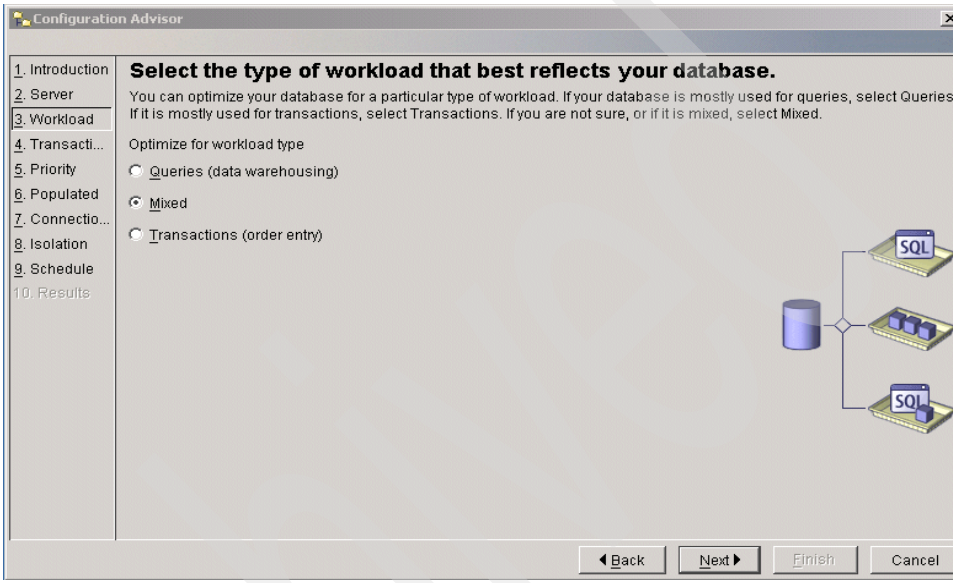

Figure 3-14 Select the type of workload that best reflects your database

Specify a typical database transaction

Figure 3-15 shows how you select the average number of SQL statements in a single unit of work (between commits) that best reflects the transactions running in your database. If you are unsure, select **More than 10**. Also select the estimated number of transactions per minute running in your database. This input will help in determining parameters for log sizing and memory allocation for locking.

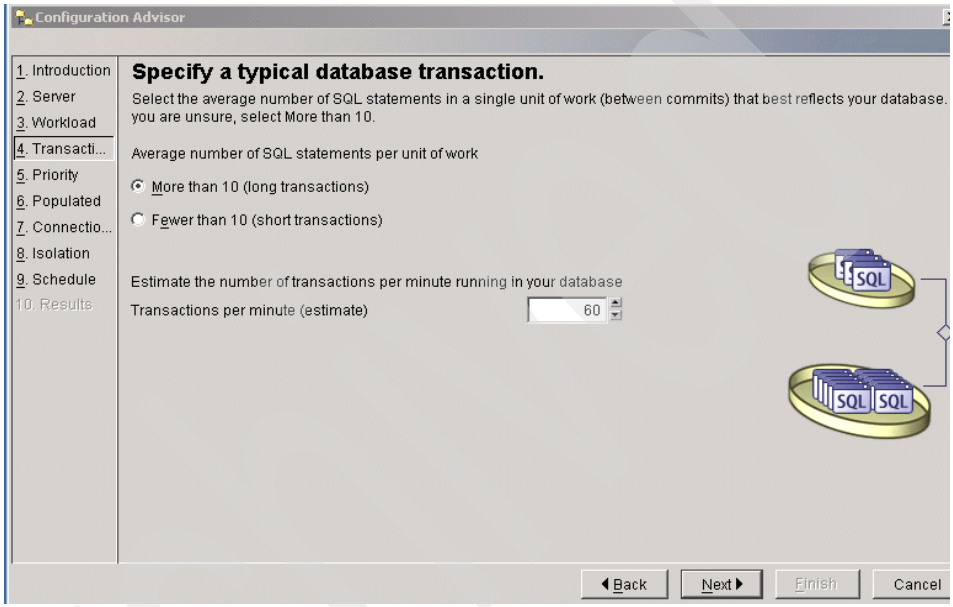

Figure 3-15 Specify a typical database transaction

Specify a database administration priority

Figure 3-16 shows how to specify whether you want to optimize for transaction performance or for shorter database recovery time. If you are not sure, or if both are equally important, select **Both**. The general recommendation is to specify both, since the impact on transaction performance is minimal.

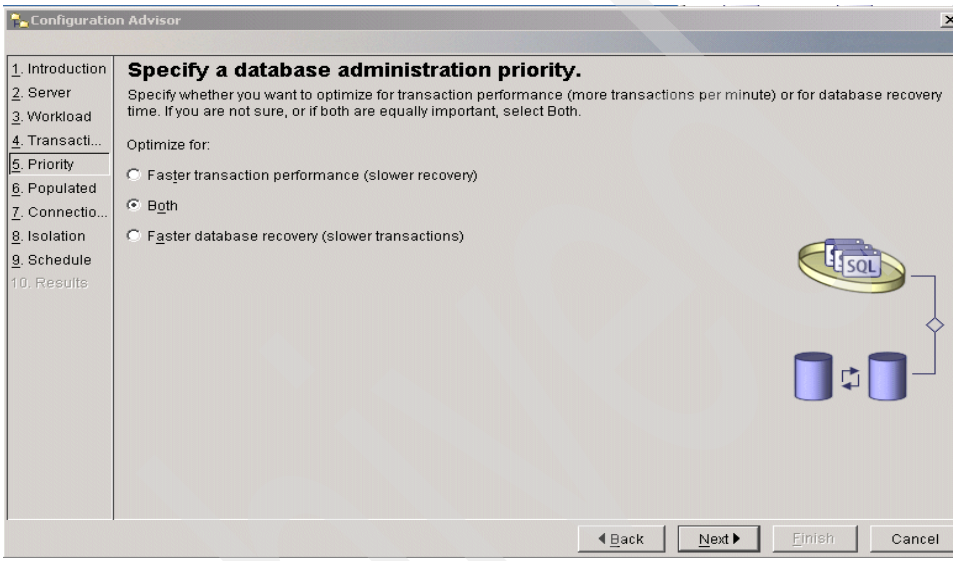

Figure 3-16 Specify a database administration priority

Specify whether the database is populated with data

The volume of data in the database can affect the recommended values. For our purposes, with this initial execution, specify **No** as shown in Figure 3-17. After the database is populated with data, it is recommended that you run the Configuration Advisor again to get more accurate values.

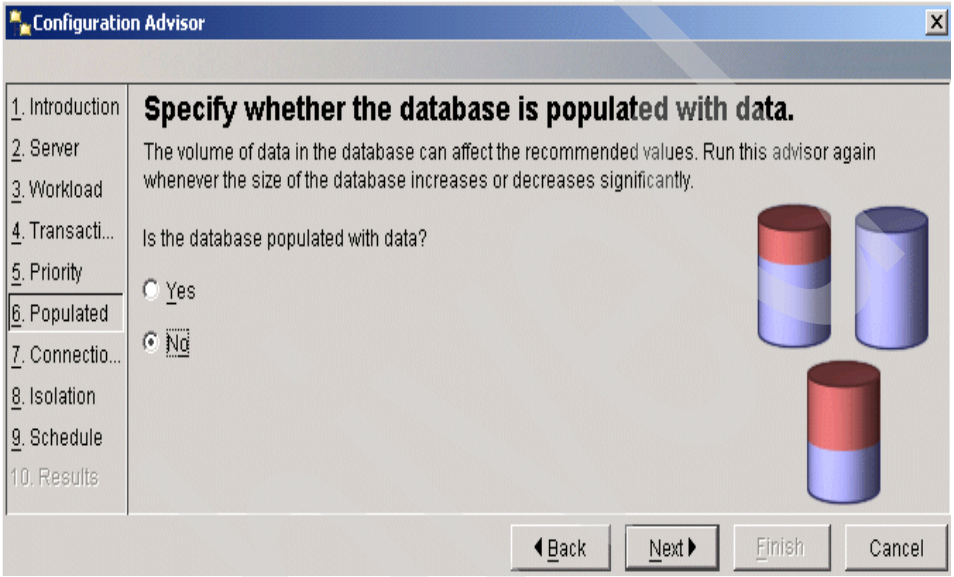

Figure 3-17 Specify whether the database is populated with data

Estimate number of applications connected to this database

Allocating enough connections ensures so that your users never have to wait for a currently connected user to disconnect. However, since each allocated connection requires more system memory, you might waste resources if you allocate too many connections. The average number of connected applications equals the number of users multiplied by the number of connections per user. If you are not sure what value to use, accept the default of zero local and 10 remote connections. Figure 3-18 shows how to specify these values.

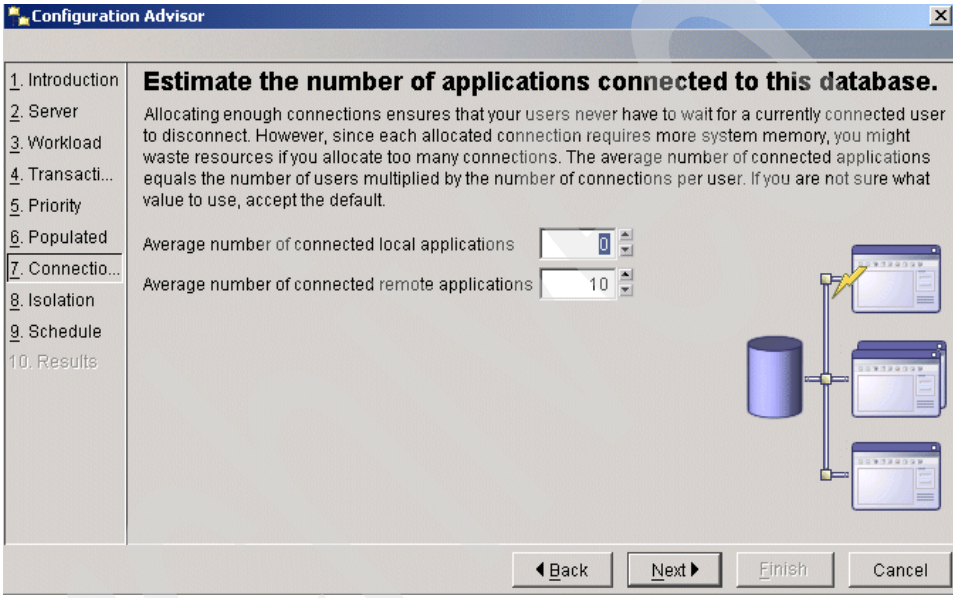

Figure 3-18 Estimate number of applications connected to this database

Select an isolation level that reflects your applications

Each application program running on your database can have a different isolation level. The level determines the number of locked rows and the locks' duration when a user either reads or changes data. Do not select an isolation level that creates more locks than necessary, since more memory is required for each lock. Most applications will use Cursor Stability as the isolation level. If you don't know what should be used, specify **Cursor Stability** as shown in Figure 3-19.

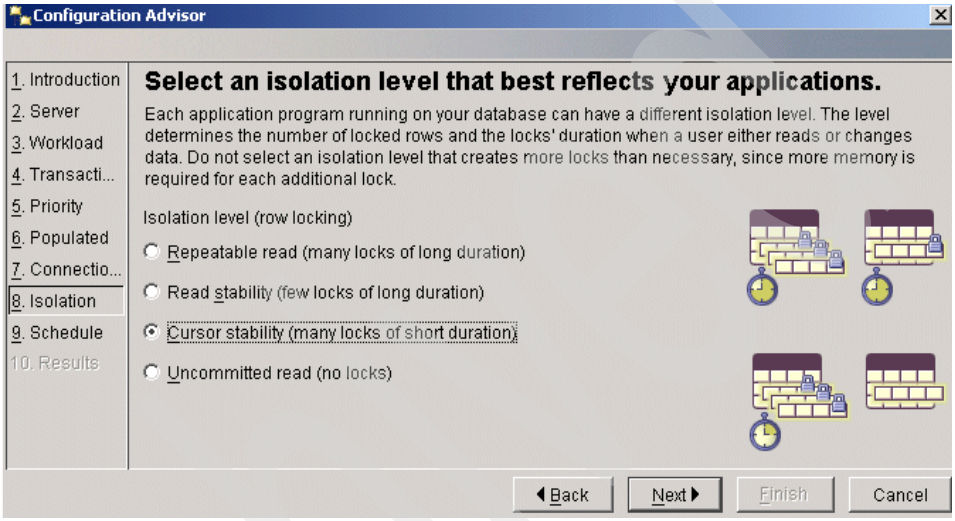

Figure 3-19 Select isolation level that best reflects your applications

Scheduling task execution

You can select whether to execute the commands immediately or create a task in the DB2 UDB Task Center. Creating a task allows you to schedule task execution and maintain its history. For our purposes we run the changes now as shown in Figure 3-20.

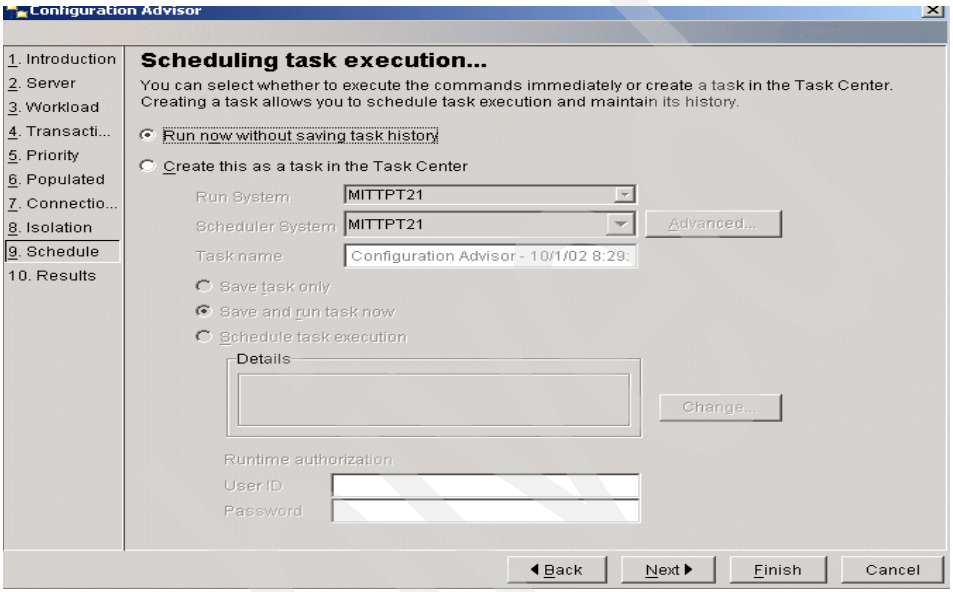

Figure 3-20 Scheduling task execution

Review the performance configuration recommendations

Figure 3-21 shows the recommended configuration parameter changes based on the input we provided above. You can see that compared to the default values, significant changes were recommended. If you select **Finish**, the recommended changes will be applied. Remember to run the Configuration Advisor again after you populate the database with data.

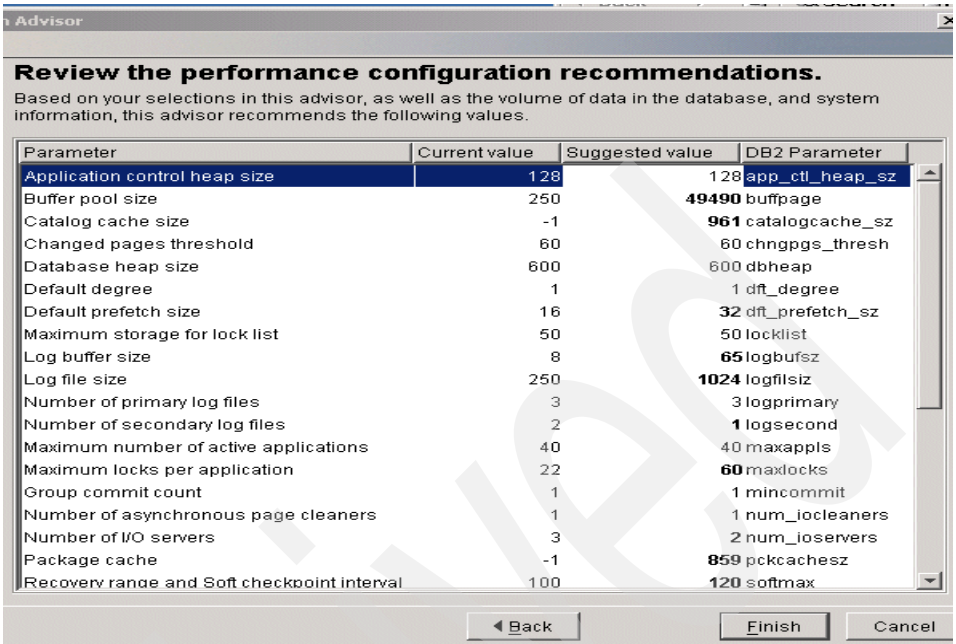

Figure 3-21 Review performance configuration recommendations

Configuration Advisor summary

The best way to think of the recommendations made by the Configuration Advisor Wizard is to use them as a starting point when you create a new database, or when an existing database has been changed significantly and you need to find better values for configuration parameters. Use the suggested values as a base on which to make further adjustments as you try to optimize the performance of your database. The remainder of this book will aid you in this regard.

3.5 Populating your database

In this section we discuss techniques for populating your database with data. The following topics are discussed:

- "Table creation" on page 117
- "Loading data" on page 128
- "Moving data" on page 129

3.5.1 Table creation

Tables are the main repository of data within databases. Creating the tables and entering data to fill in the tables will occur when you are creating a new database. Before creating tables in your database, you may also want to create table spaces for your tables as discussed in the manual, *DB2 UDB V8 Administration Guide: Implementation*, SC09-4820, Chapter 2, in the section “Creating a table space”. For our example, we use the default table space.

After you determine how to organize your data into tables, the next step is to create those tables, by using the CREATE TABLE statement. The table descriptions are stored in the system catalog of the database to which you are connected.

The CREATE TABLE statement gives the table a name, which is a qualified or unqualified identifier, and a definition for each of its columns. You can store each table in a separate table space, so that a table space contains only one table. If a table will be dropped and created often, it is more efficient to store it in a separate table space and then drop the table space instead of the table. You can also store many tables within a single table space.

The table does not contain any data at first. To add rows of data to it, use one of the following:

- The SQL INSERT statement
- The LOAD or IMPORT commands

A table consists of one or more column definitions. A maximum of 500 columns can be defined for a table. Columns represent the attributes of an entity. The values in any column are all the same type of information.

Note: The maximum of 500 columns is true when using a 4 KB page size. The maximum is 1012 columns when using an 8 KB, 16 KB, or 32 KB page size.

A column definition includes a column name, data type, and any necessary null attribute, or default value (optionally chosen by the user).

The column name describes the information contained in the column and should be something that will be easily recognizable. It must be unique within the table; however, the same name can be used in other tables.

The data type of a column indicates the length of the values in it and the kind of data that is valid for it. The database manager uses character string, numeric, date, time, and large object data types. Graphic string data types are only available for database environments using multi-byte character sets. In addition, columns can be defined with user-defined distinct types.

The default attribute specification indicates what value is to be used if no value is provided. The default value can be specified, or a system-defined default value used. Default values may be specified for columns with, and without, the null attribute specification.

The null attribute specification indicates whether or not a column can contain null values.

To create a table using the command line, enter:

```
CREATE TABLE <NAME>
(<column_name> <data_type> <null_attribute>)
IN <TABLE_SPACE_NAME)
```

The following is an example of a CREATE TABLE statement that creates the EMPLOYEE table. This table is defined in the sample database:

```
CREATE TABLE EMPLOYEE
( EMPNOCHAR(6) NOT NULL PRIMARY KEY,
FIRSTNMEVARCHAR(12) NOT NULL,
MIDINITCHAR(1) NOT NULL WITH DEFAULT,
LASTNAMEVARCHAR(15) NOT NULL,
WORKDEPTCHAR(3),
PHONENOCHAR(4),
PHOTOBLOB(10M) NOT NULL)
```

To create a table using the Control Center:

1. Expand the object tree until you see the Tables folder.
2. Right-click the **Tables** folder, and select **Create** —> **Tables Using Wizard** from the pop-up menu.
3. Follow the steps in the wizard as shown below to complete your tasks

Create Table wizard

The Create Table wizard provides an easy-to-use interface to create your tables. In the following examples we step through the wizard, creating the same EMPLOYEE table shown above created with SQL.

Figure 3-22 shows the first screen displayed when using the Create Table Wizard. It requests the schema name, table name and optionally a comment for the new table. The name for a table consists of two parts in the form of *schemaname.tablename*. The schema name is used as a high level qualifier for the table. A schema is an excellent way to group tables that are related. The table name is the name you would like to use for the table, but it should be something meaningful. The optional comment provides a means of describing the table.

For our example in Figure 3-22 we use the schema name of NULLID and the table name of EMPLOYEE. After completing the information, select **Next**..

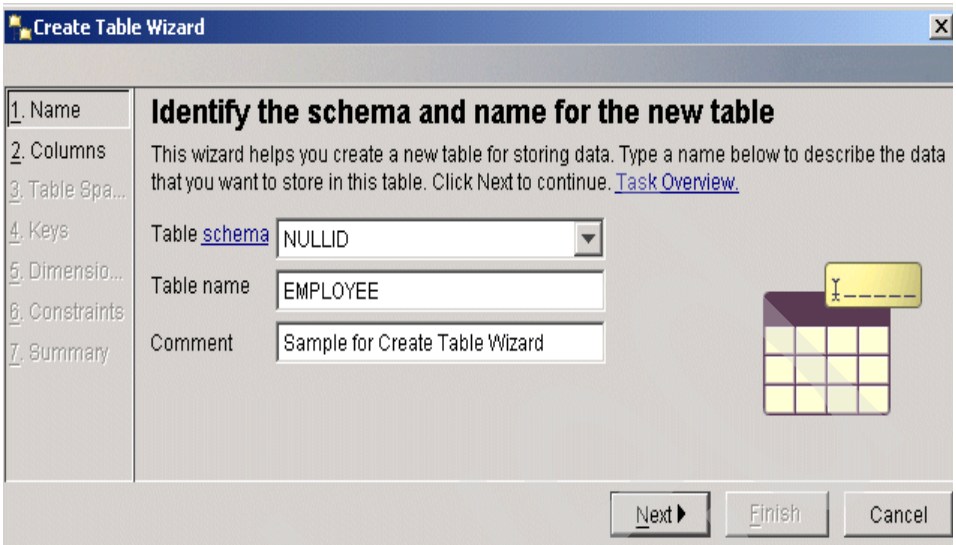

Figure 3-22 Create Table Wizard — Identify table

The next step in table creation is to define the columns in the table as shown in Figure 3-23, select **Add** to bring up the Add Columns dialog.

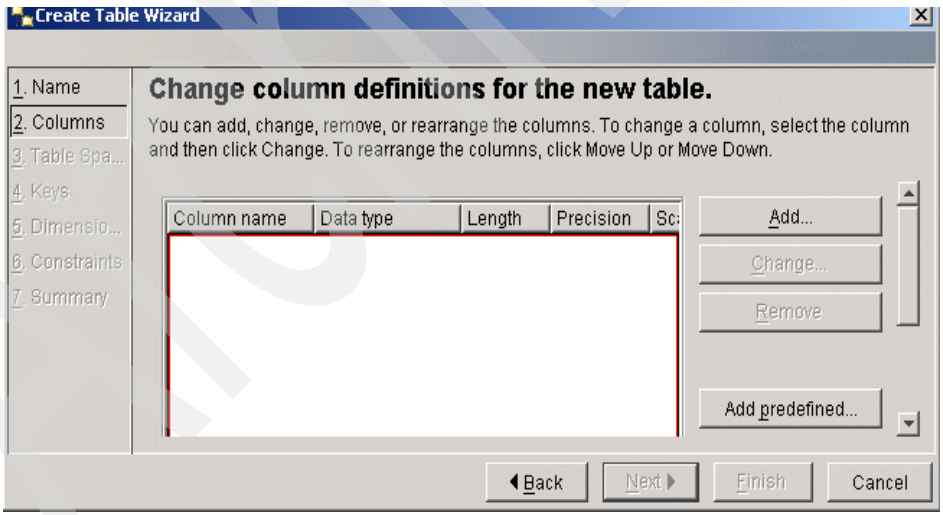

Figure 3-23 Create Table Wizard — Define columns

Figure 3-24 shows the Add Columns dialog, where you define individual columns and their attributes. The attributes that can be defined are as follows:

- **Column name:** The column name is the name you want to assign to the column. It can be up to 128 characters in length and should be meaningful for those using it.
- **Datatype:** The data type defines the type of data the column contains. The drop-down menu will show you the valid selections.
- **Datatype characteristics:**
  - Length: Defines the length column
  - Precision: Defines the decimal positions for decimal column
  - Bit data: Defines if the column contains binary data
  - Lob options: Defines if lob column is logged or compact
- **Nullable:** Defines if the column can contain Null values
- **Default:** Defines if default values are to be used for the column if values are not provided when a row is inserted. You can optionally specify what the default value is to be. If you don't specify your own default value, blanks will be used for character columns, zeros for numeric columns or current date/time/timestamp for DATE/TIME/TIMESTAMP columns.
- **Comment:** Provides descriptive text for the column
- **System default compression:** Specifies if you want DB2 to compress the column data if the system default value for the column is present.
- **Generate column contents:** Allows you to have the value for the column to be generated with either a formula or a sequential identity.

For our example, we define the column EMPNO as a character column 6 bytes in length that cannot contain Nulls. To add additional columns, select **Apply**. When all columns are defined, select **OK**. See Figure 3-24.

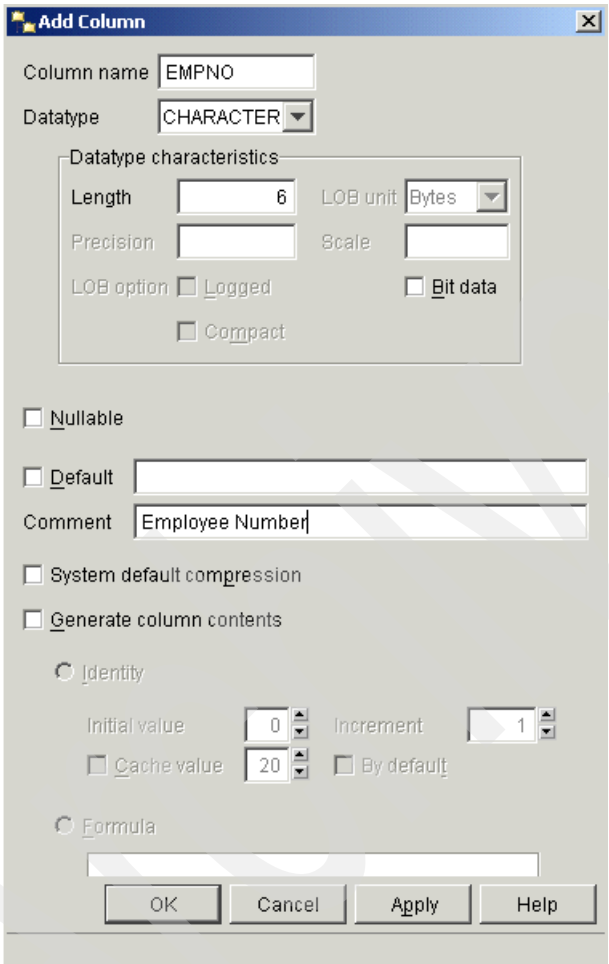

Figure 3-24 Create Table Wizard — Add column

Figure 3-25 shows a review of the columns that were defined by completing the Add Column dialog. Select **Next** to proceed to the next step.

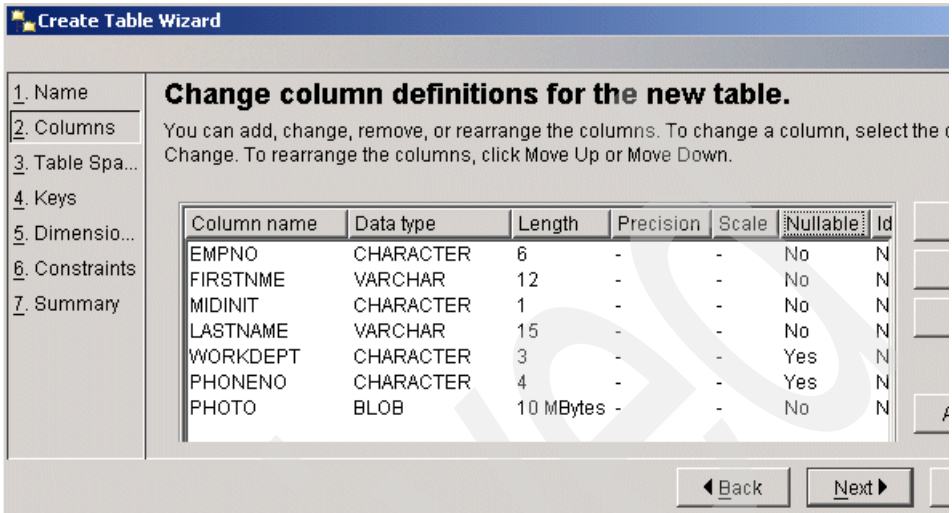

Figure 3-25 Create Table Wizard — Column review

Figure 3-26 shows the dialog to specify the table space to place the table in. The dialog permits you to specify an existing table space or to create a new table space. For our example we use the default table space. To proceed to the next step, select **Next**.

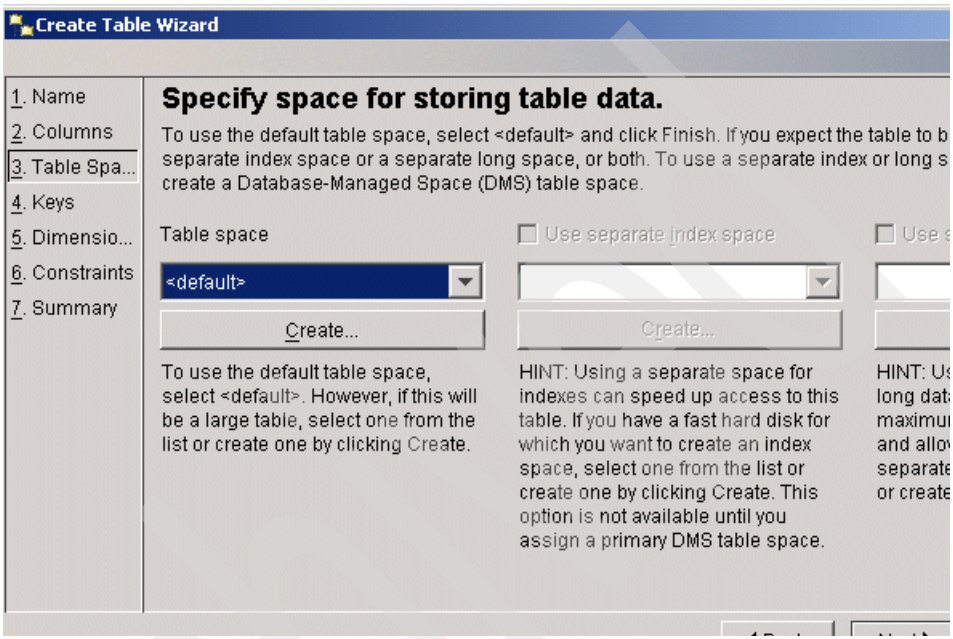

Figure 3-26 Create Table Wizard — Specify table space

Figure 3-27 shows the dialog where you can define keys for the table being created. The following types of keys can be defined:

- **Primary key:** A primary key identifies the columns to provide a unique identifier for each row in the table. A table can only have one primary key.
- **Unique key:** A unique key identifies the columns of the table on which you want DB2 to enforce uniqueness.
- **Foreign key:** Foreign keys ensure that a combination of one or more column values exist in another table.
- **Partitioning key:** A partitioning key determines the columns from which values are used to map a data row to a specific database partition. This option is only available if the Database Partitioning Feature is installed.

For our example we only define a primary key. Select **Add Primary** to proceed.

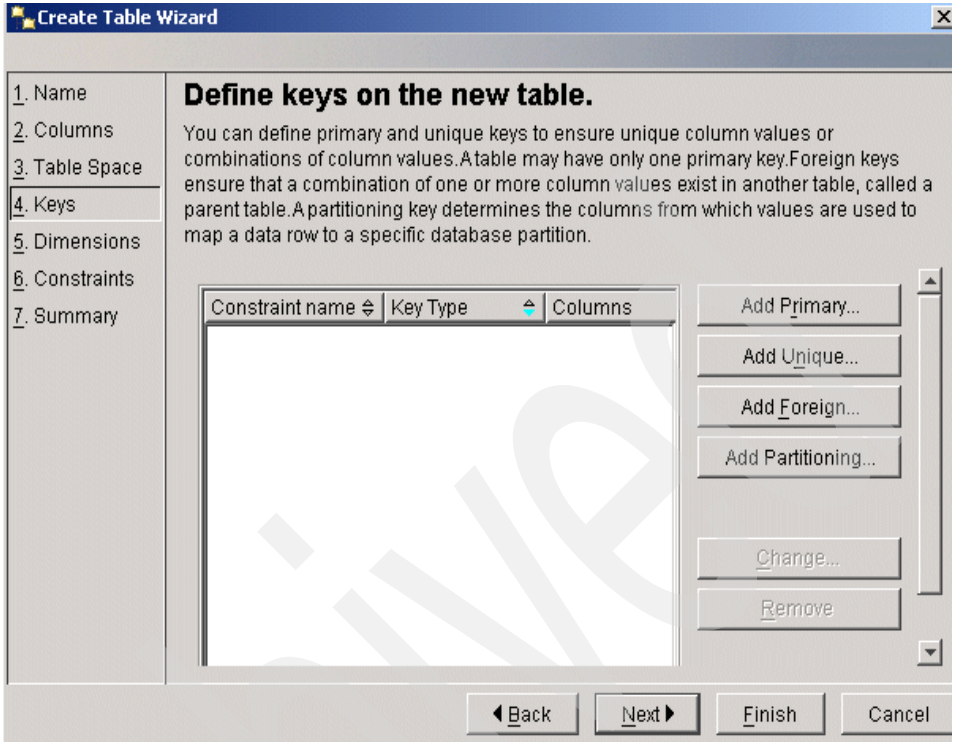

Figure 3-27 Create Table Wizard — Define keys

Figure 3-28 shows the Define Primary Key dialog. You can select one or more columns that are not nullable for your primary key and provide a name for the primary key constraint. The combination of columns selected must be unique within the table. Defining a primary key will result in an unique index being created on the table using the columns selected.

In our example we select the column **EMPNO** and use the name EMPPKEY for the constraint name. Select **OK** to define the primary key and return to the define keys dialog.

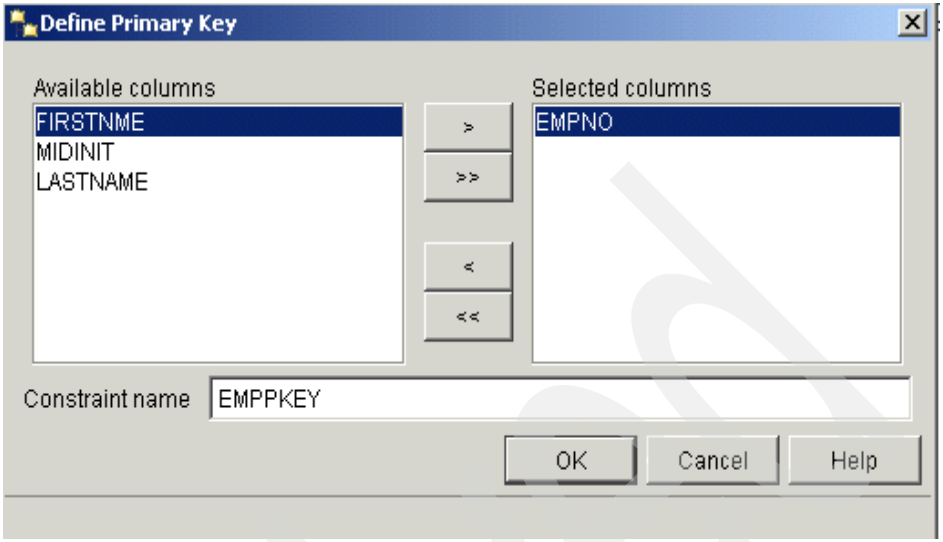

Figure 3-28 Create Table Wizard — Define primary key

Figure 3-29 shows the Define keys dialog after defining the primary key. Since we are only defining the primary key, select **Next** to proceed.

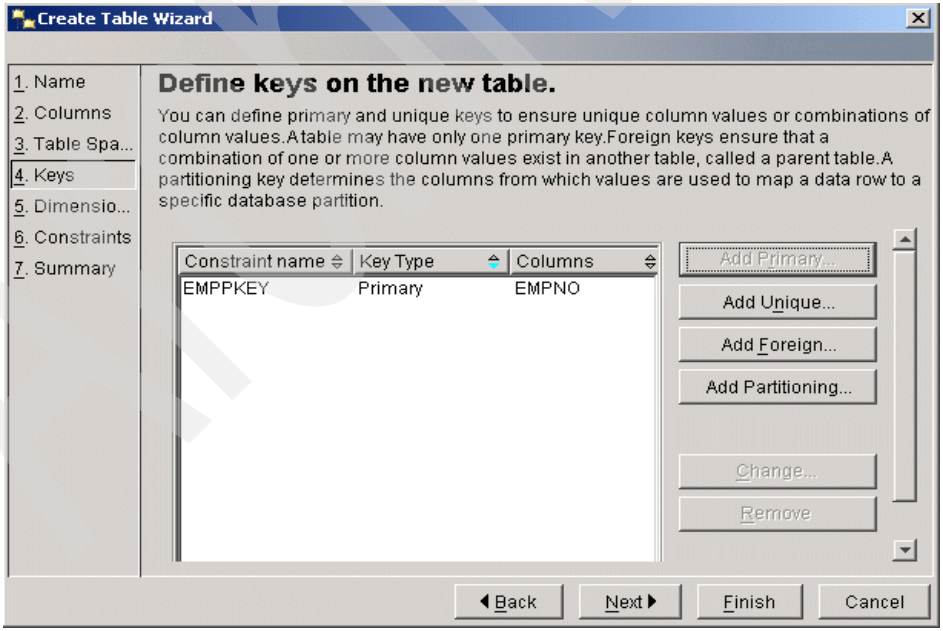

Figure 3-29 Create Table wizard — Keys review

Figure 3-30 shows the dialog for defining the columns to be used as dimensions when using a multi-dimensional clustering table. Defining one or more dimensions on a table enables multi-dimensional clustering. Each dimension, which can contain one or more table columns, represents an axis along which the table data will be clustered.

In our example we are not be using multi-dimensional clustering. Select **Next** to proceed.

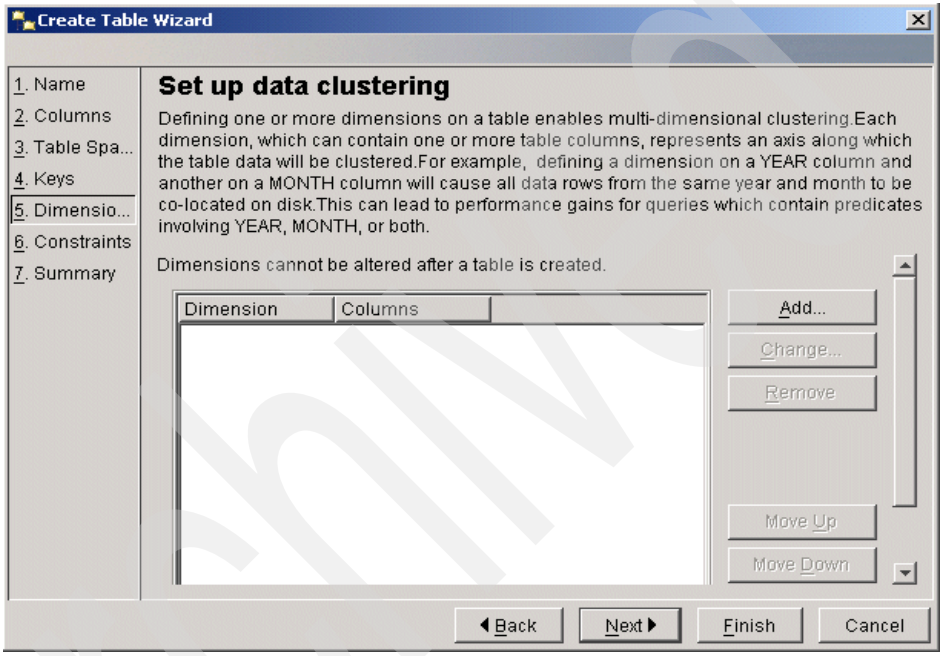

Figure 3-30 Create Table Wizard — Define clustering dimensions

Figure 3-31 shows the dialog where you can define additional check constraints on the table. Table check constraints are rules that specify the allowed values for every row in a table. They are checked whenever data is inserted or updated.

However, in our example we are not using constraints. Select **Next** to proceed.

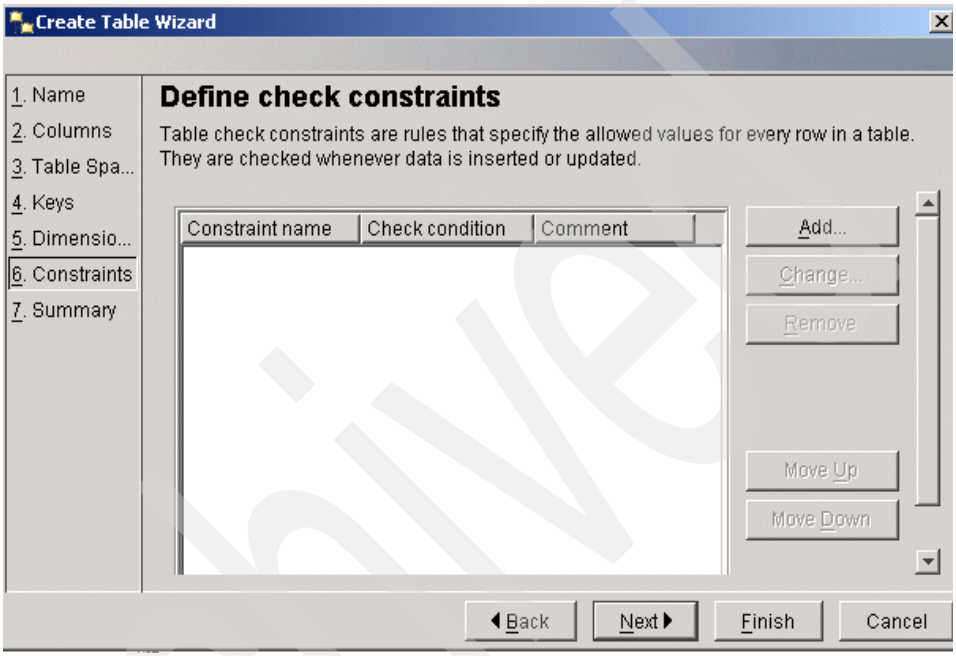

Figure 3-31 Create Table Wizard — Define check constraints

The dialog shown in Figure 3-32 provides a review of the actions that will take place when you click **Finish**. When you click **Finish**, the wizard creates a table based on your previous input. To change any of the parameters, go back to the appropriate page in this wizard. To show the SQL generated, click the **Show SQL.** The generated SQL can be saved by cutting and pasting it into a text file.

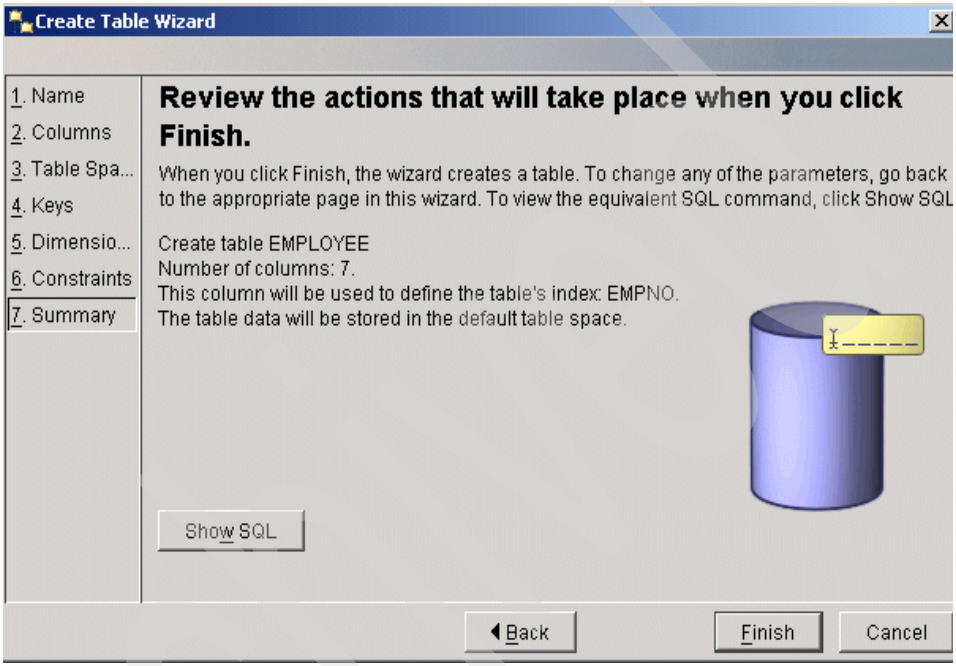

Figure 3-32 Create Table Wizard — Summary

3.5.2 Loading data

After creating a table it does not contain any data. To populate your table you can use one of the following approaches:

- The SQL INSERT statement
- The IMPORT command
- The LOAD command

For information on the SQL INSERT statement see the DB2 UDB V8 SQL Reference manual. For information on the IMPORT command see the DB2 UDB V8 Command Reference manual. In this section we discuss the LOAD command, since it is the preferred method for populating tables.

Load command

The LOAD command is a high-speed method of populating a table with data from several different input sources. It supports input data from the following sources:

- ASC (non-delimited ASCII format)
- DEL (delimited ASCII format)
- IXF (integrated exchange format, PC version), exported from the same or from another DB2 table
- CURSOR (a cursor declared against a SELECT or VALUES statement).

It can be used for the initial population of a table or to append additional data to a table. When appending data it can be used in an online mode that still permits access to existing data in the table while it is executing. It will also build any indexes that have been defined on the table it is loading.

Additional details on the LOAD command can be found in the DB2 UDB Command Reference manual publication number SC09-4828.

3.5.3 Moving data

A common requirement in a database management system is to move data from different data sources into the database. DB2 UDB provides a number of powerful techniques for this data movement. In this section we discuss the following techniques:

- Moving data from a DB2 table to another DB2 table in the same database
- Moving data from an OLE DB data source into a DB2 table
- Moving data from a DB2 table in another database into a DB2 table

Moving data from one table to another in the same database

In this section we show how an existing DB2 table can be duplicated in a database, using another name, and using DB2 Command Line Processor statements.

We first need to connect to the database are working with. In our case, we use the DB2 provided SAMPLE database:

```
CONNECT TO SAMPLE
```

We then need to create the new table that we will be duplicating. We use the “CREATE TABLE LIKE” SQL statement that will create a new table with the same attributes as another table, in our case we use the sample table EMPLOYEE as a model to create the new table TEST.EMPLOYEE.

```
CREATE TABLE TEST.EMPLOYEE LIKE EMPLOYEE
```

After creating the new table, we need to populate it with the data from the original table. We utilize a capability of the LOAD command that can use a SQL SELECT statement as input to the load process. The input data is defined by using the DECLARE CURSOR statement that defines the input data. Any valid SELECT statement can be used. In our example we select all columns and rows from the source table:

```
DECLARE LCUR CURSOR FOR SELECT * FROM EMPLOYEE
```

Once the input is defined with the DECLARE CURSOR, we invoke the LOAD command, specifying the cursor defined above and the table the data is to be loaded into, in our example TEST.EMPLOYEE:

```
LOAD FROM LCUR OF CURSOR INSERT INTO TEST.EMPLOYEE
```

After the load is complete, we disconnect from the database:

```
CONNECT RESET
```

Another variation of this technique would be to use the SQL INSERT statement instead of the DECLARE CURSOR statement and the LOAD command as follows:

```
INSERT INTO TEST.EMPLOYEE SELECT * FROM EMPLOYEE
```

The disadvantage of this technique is that INSERT is normally much slower than using LOAD and all inserted rows will be logged.

Moving data from an OLEDB data source

One of the techniques DB2 UDB provides to access non DB2 data is through User Defined Table Functions (UDF). These UDFs can return any data as if it were a DB2 table.

The Development Center provided with DB2 UDB, provides an excellent development tool for developing DB2 Stored Procedures and User Defined Functions (UDF). As part of the tool, the Development Center can construct UDFs along with views over the UDFs and optionally move the data into a DB2 table for the following types of data sources:

- SQL
- MQSeries
- XML
- OLE DB

Since most Windows environments have some data available in OLEDB data sources, we provide an example of moving data from the sample Customer table in the Northwinds database provided with Microsoft Access into a DB2 table.

For our example the DB2 SAMPLE database must have been created with the First Steps at install or by issuing the command DB2SAMPL from a command prompt. You must also have Microsoft Access installed on your workstation.

To use the Development Center you must have selected it when you installed the DB2 server software or have installed the DB2 Application Development Client on your workstation.

To start the Development Center from the Windows Start menu Select:
Programs->**IBM DB2**->**Development Tools**->**Development Center**

When the Development Center comes up you will be presented with a screen titled Development Center Launchpad as shown in Figure 3-33. This screen presents us with options to Create Project, Add Connection, and Create Object. We are performing all three steps for our example. First select **Create Project**.

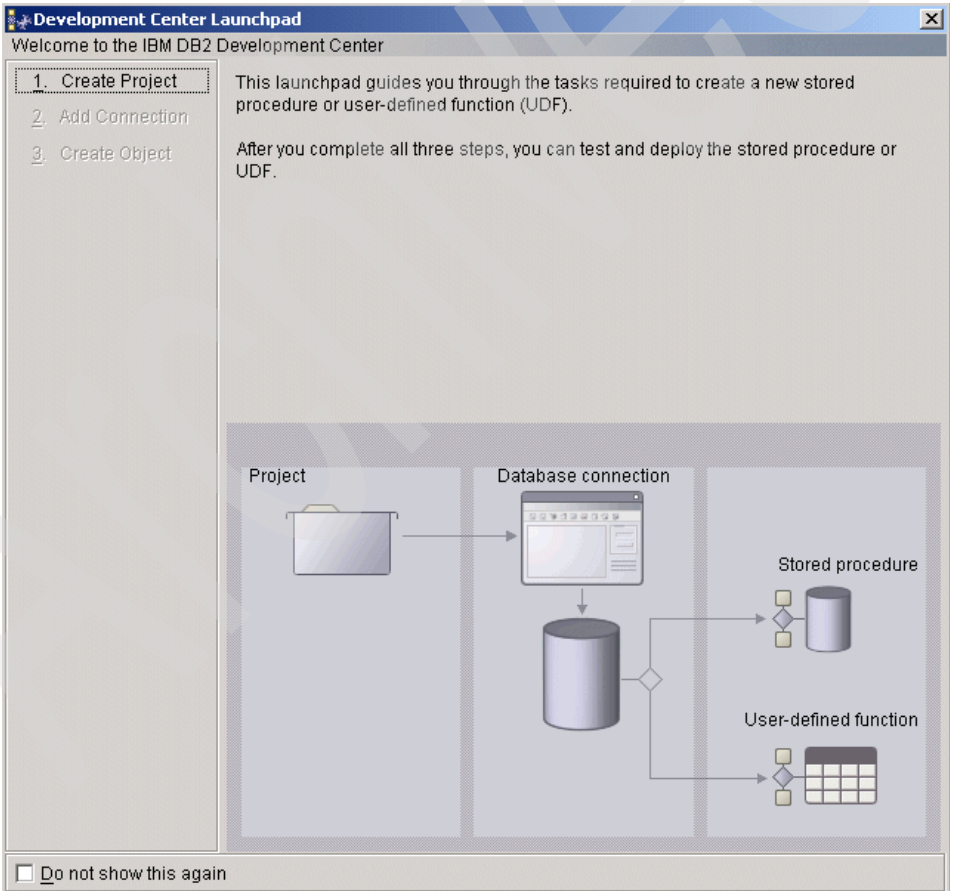

Figure 3-33 Development Center Launchpad

We are now presented with the Open Project dialog as shown in Figure 3-34. We create a new project for our example using the name NWIND. Select **OK.** The project is created, and we are returned to the Development Center Launchpad, as shown before in Figure 3-33 on page 131, where we will now add a database connection by selecting **Add Connection** (Figure 3-34).

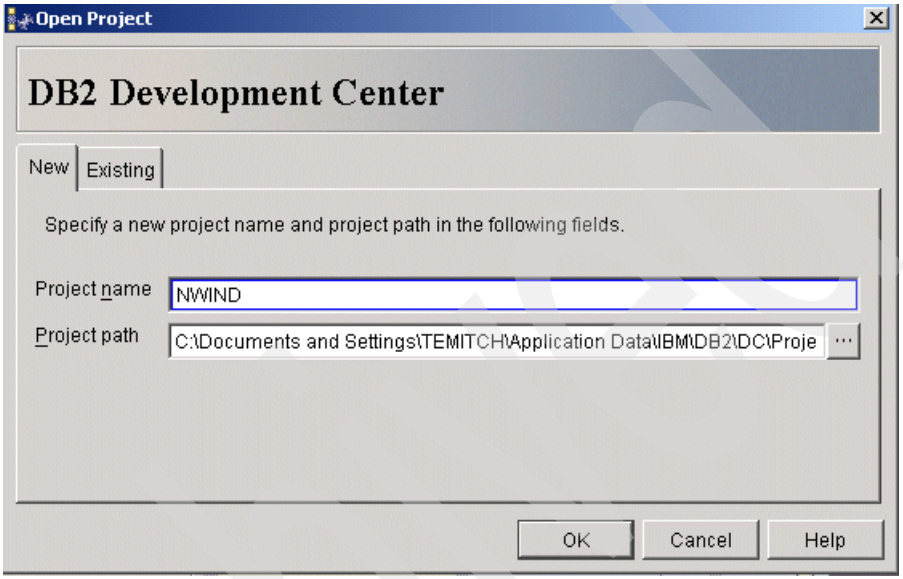

Figure 3-34 Development Center — Open Project

The Add Connection dialog as shown in Figure 3-35 requests information on the DB2 database to connect to for this project and the type of connection. For our example we want the connection to be online. Select **Online** and then select **Next** to proceed.

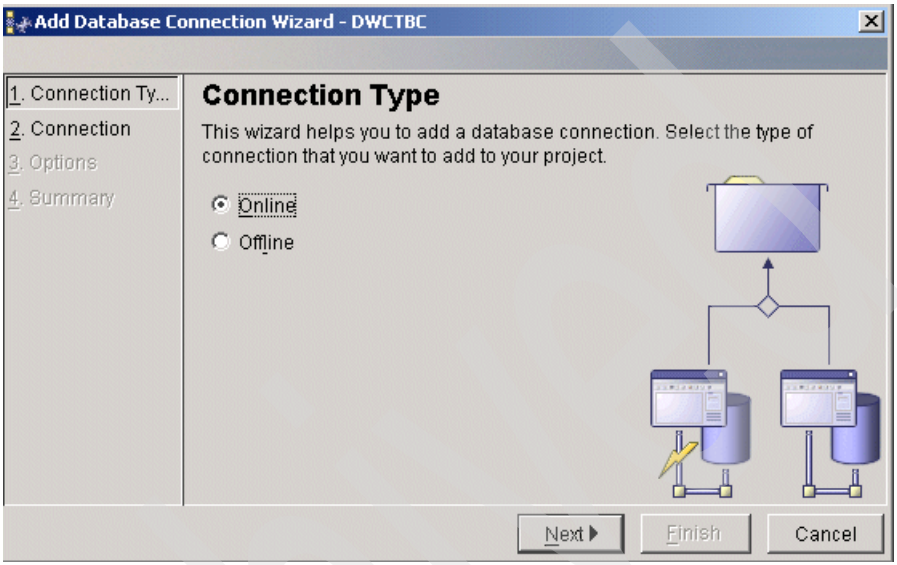

Figure 3-35 Development Center — Add Connection — Type

The dialog shown in Figure 3-36 requests the information about the database we would like to connect to for this project. For our example we want to use the DB2 provided sample database and to connect to it with our current userid and password.

The database Alias drop-down list will show the databases available on this workstation, select **SAMPLE** from the list. For the User information, check the checkbox, Use your current userid and password.

You can optionally test the connection to the selected database by selecting **Test Connection**.

Select **Next** to proceed to Options.

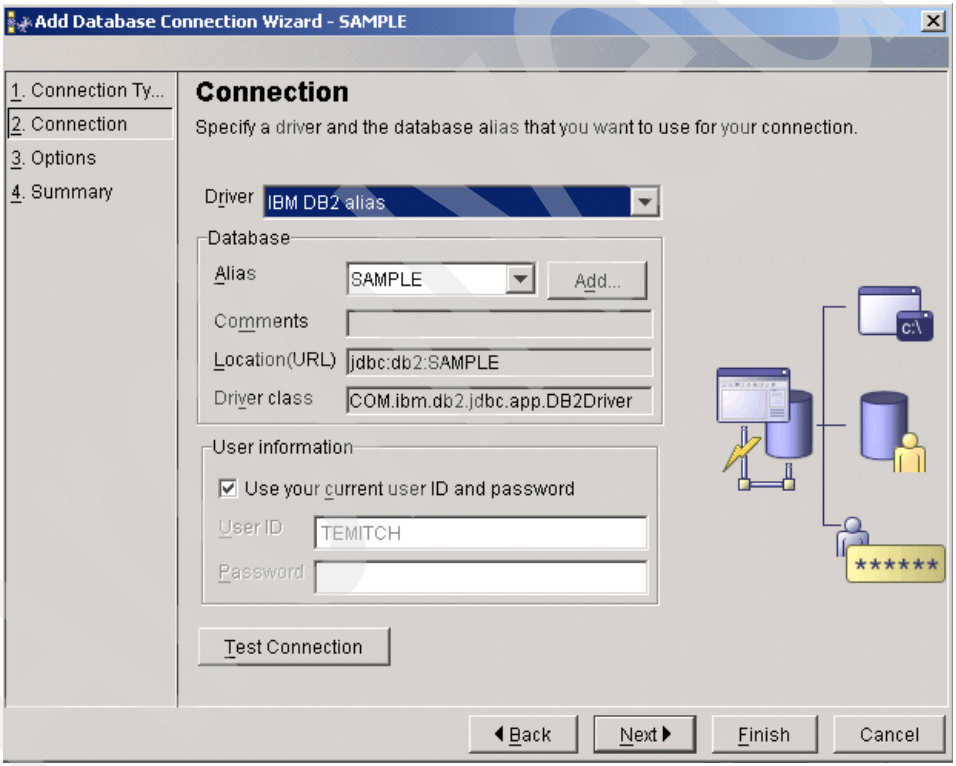

Figure 3-36 Development Center — Add Connection — Specify a Connection

The Options dialog shown in Figure 3-37 allows us to specify a schema name to use for this connection. For our example we use NWIND. Select **Next** to proceed to the Summary screen.

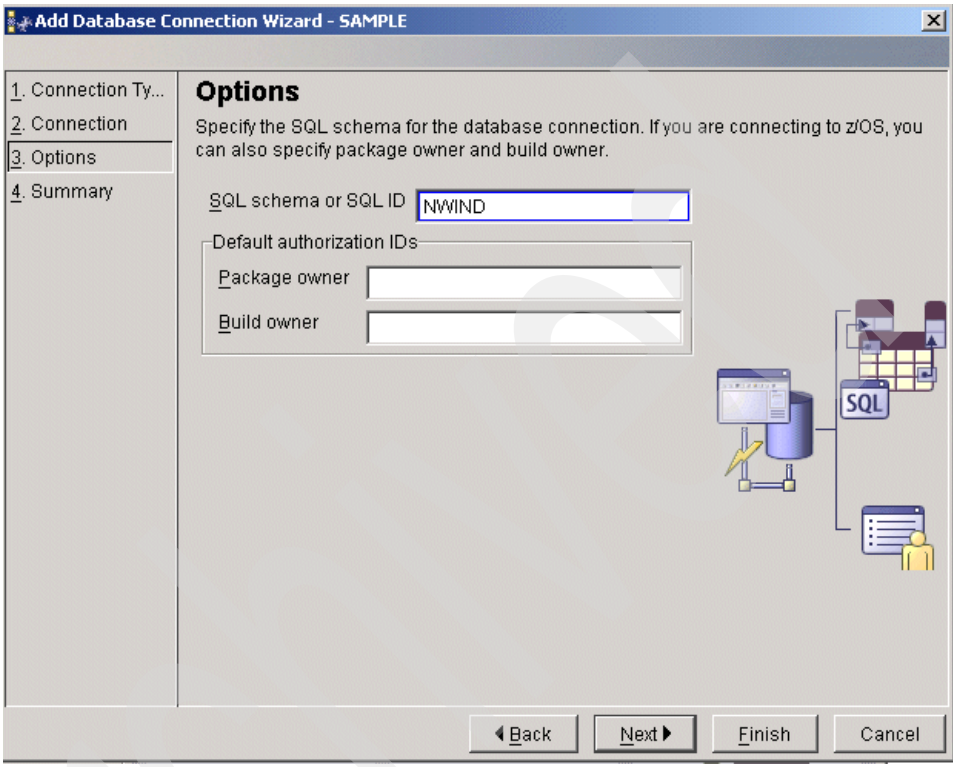

Figure 3-37 Development Center — Add Connection — Options

The Add Connection Summary screen shown in Figure 3-38 provides a review of the options we select for this connection. Select **Finish** to create the connection and return to the Development Center Launchpad dialog as shown in Figure 3-33 on page 131. Once you are back at the launchpad, select **Create Object** to proceed to creating our User Defined Table Function.

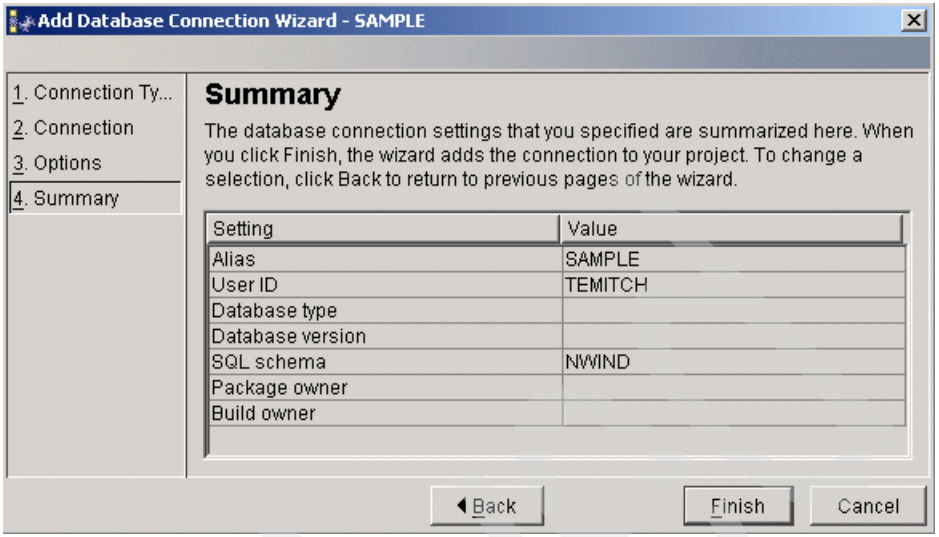

Figure 3-38 Development Center — Add Connection — Summary

The New Object dialog shown in Figure 3-39 requests information on the type of object we want to create. The object types available are Stored Procedure and User-Defined Function (UDF) with the additional options for SQL, MQSeries, XML and OLE DB. For our example, we highlight **User-Defined Function** and **OLE DB**, then select **OK** to proceed to the creation of the UDF.

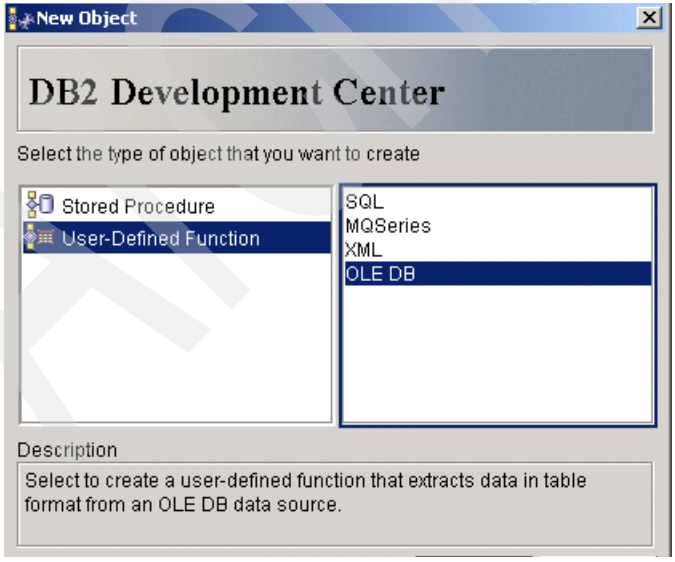

Figure 3-39 Development Center — New object

The specify UDF name dialog shown in Figure 3-40 requests what name we should use for the User-Defined Function (UDF) we are about to create. We name our UDF for our example NWIND.CUSTOMER and provide descriptive text describing the UDF. After entering the information, we select **Next** to proceed to the OLE DB Provider dialog.

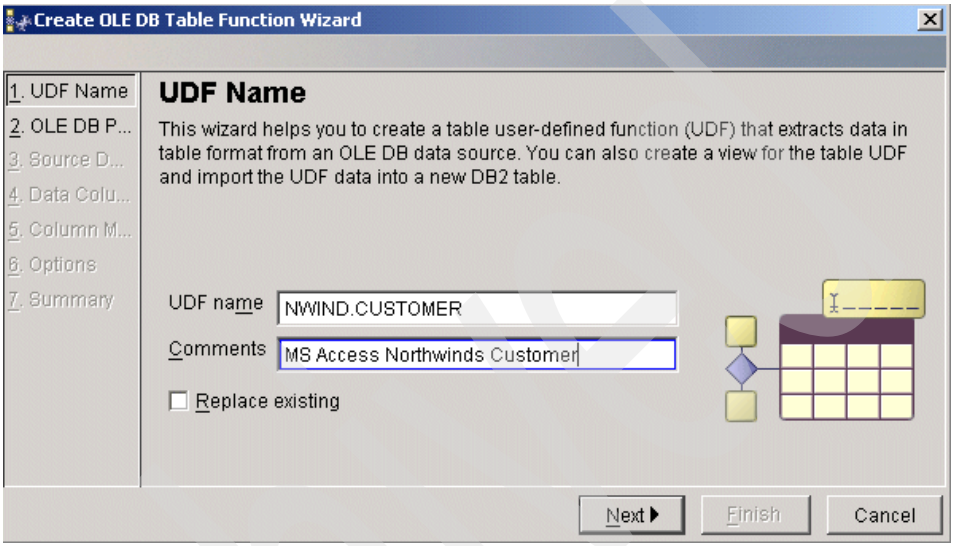

Figure 3-40 Development Center — Specify UDF name

The next five dialogs, starting with Figure 3-41, will request the information the OLE DB provider will be using for our UDF. The first step in providing this information is to build a connect string. Select **Build String** to proceed to the next dialog.

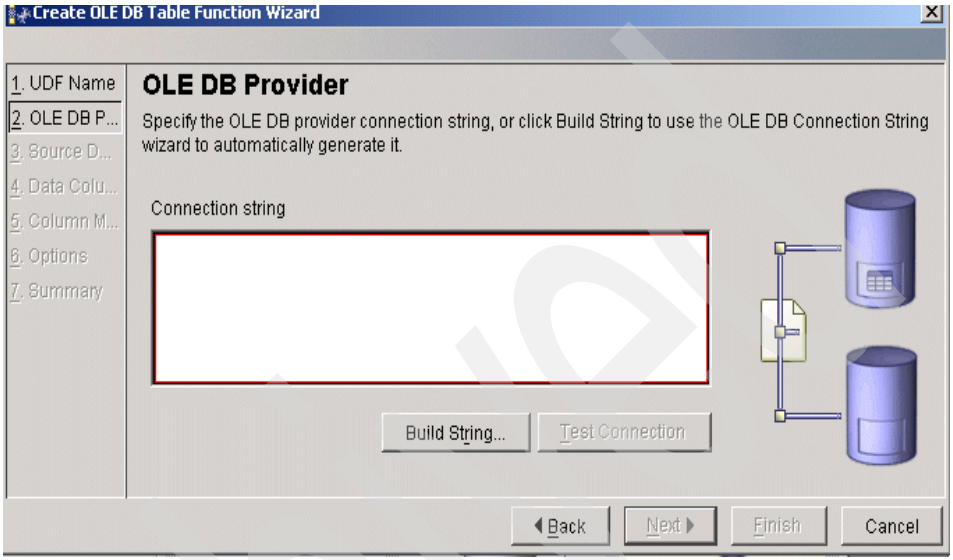

Figure 3-41 Development Center — Specify OLE DB Provider

The dialog shown in Figure 3-42 requests the OLE DB provider we will be using for our UDF. Your system may show a different list of providers, depending on what is installed on your workstation. Any of the providers listed could be used as a source for other UDFs you may want to create. For our example, we select the Microsoft Jet 4.0 OLE DB Provider to access the Microsoft Access Northwinds database. Highlight the provider, then select **Next** to proceed.

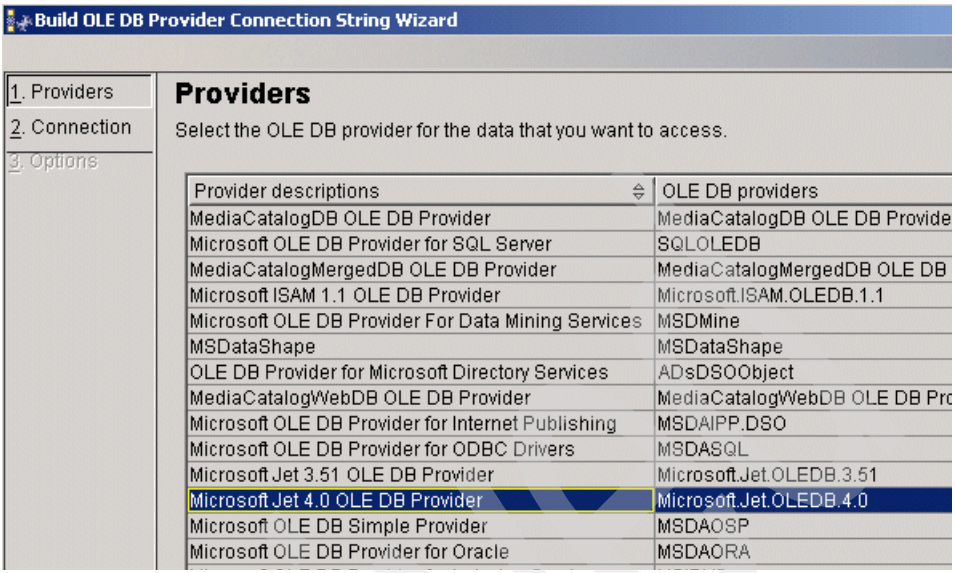

Figure 3-42 Development Center — Select OLE DB provider

The Connection dialog shown in Figure 3-43 requests the connection information for the data we want to access from the OLE DB provider. For our example, we need to specify the path and file information for the Microsoft Access Northwinds database which is located in the directory path, C:\Program Files\Microsoft Office\Office\Samples\Northwind.mdb. Depending on where you installed Microsoft Access on your workstation, it could in another location.

You can also specify a userid and password if needed by your system. For our example we use the current userid and password.

After entering the information you can select **Test Connection** to verify you can access the database.

Select **Next** to proceed to the OLE DB provider options dialog.

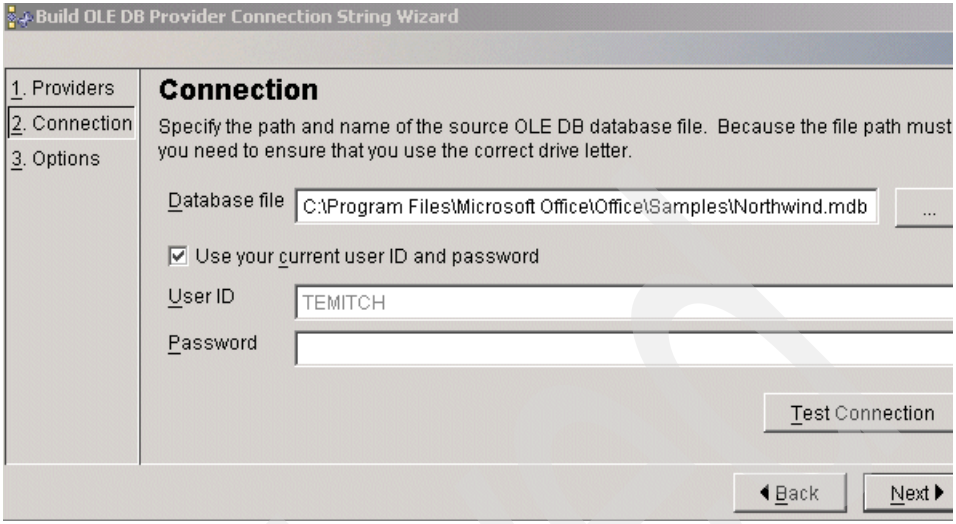

Figure 3-43 Development Center — OLE DB connection

The OLE DB provider Options dialog shown in Figure 3-44 permits you to change options specific to the OLE DB provider you will use. Refer to the documentation for the specific providers for details. For our example we don't require any changes. Select **Finish** to proceed to the next dialog.

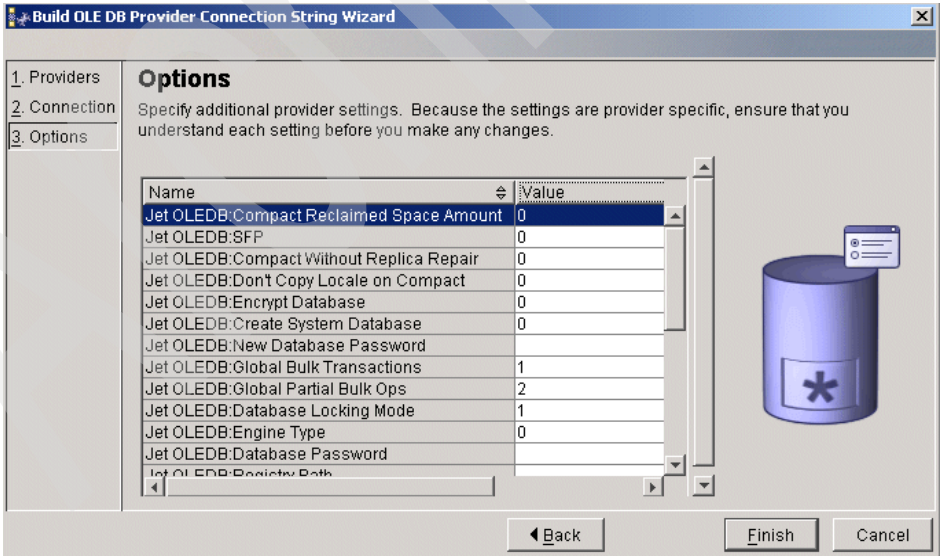

Figure 3-44 Development Center — OLE DB provider options

The screen shown in Figure 3-45 provides a review of the OLE DB provider connection string we built in the previous dialogs. Select **Next** to proceed to the dialog to specify the specific source data we want to use in our UDF.

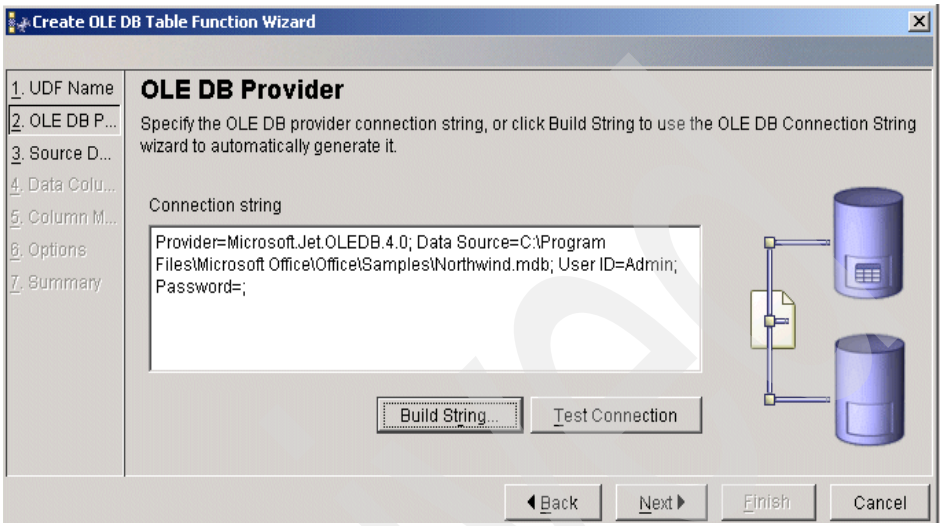

Figure 3-45 Development Center — OLE DB provider connect string

The dialog shown in Figure 3-46 requests the source data we want to access from our UDF. When the checkbox, Access source data using table, is checked, the drop-down list will show all the tables available for access in the database. You can also use an SQL query to access the source data. For our example we select the Customers table from the drop-down list. If you would like to look at a sample of the data content of table, you can select **Show Sample Content**.

Select **Next** to proceed to the data column selection dialog.

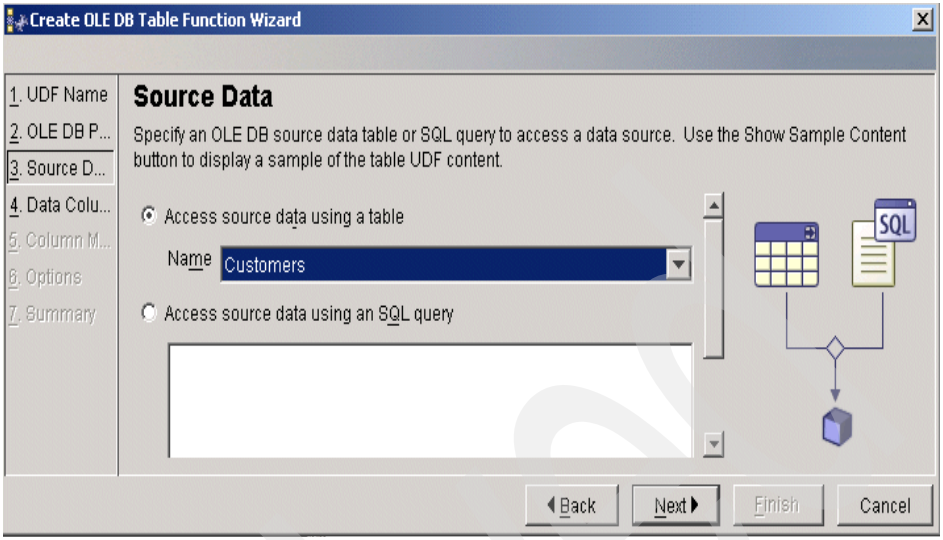

Figure 3-46 Development Center — Specify OLE DB source data

The Specify OLE DB data columns dialog shown in Figure 3-47 permits you to select specific data columns from the data source. By default, all columns from the table will be selected. To remove unwanted columns, highlight the columns you don't want and select the **< box**. For our example we use all columns in the table. To proceed to the Column Mapping dialog, select **Next**.

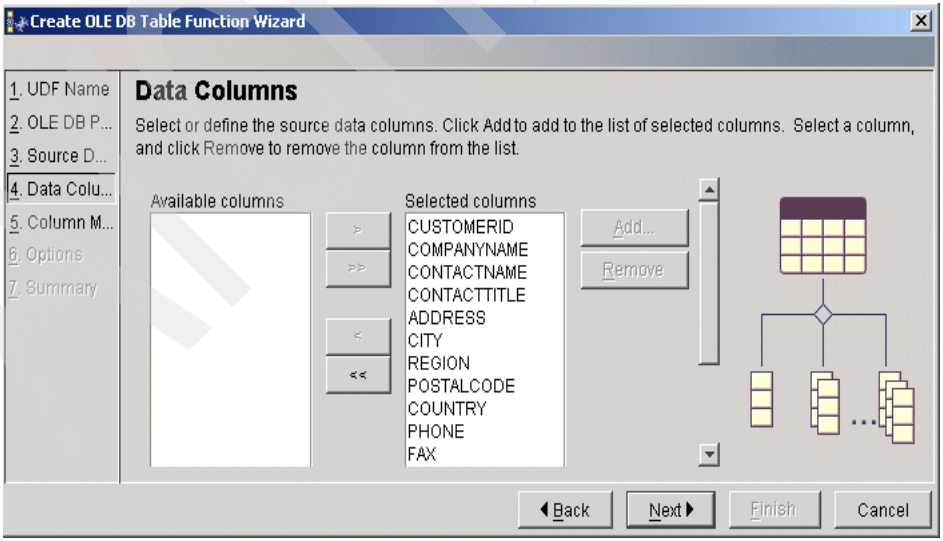

Figure 3-47 Development Center — Specify OLE DB data columns

The Column Mapping dialog shown in Figure 3-48 permits you to change the data type mapping of columns in the table. An example might be to change an OLEDB data type of DBTYPE_WSTR which would normally map to a DB2 VARCHAR data type to a CHAR data type. For our example we use the default column mapping. Select **Next** to proceed to UDF Options dialog.

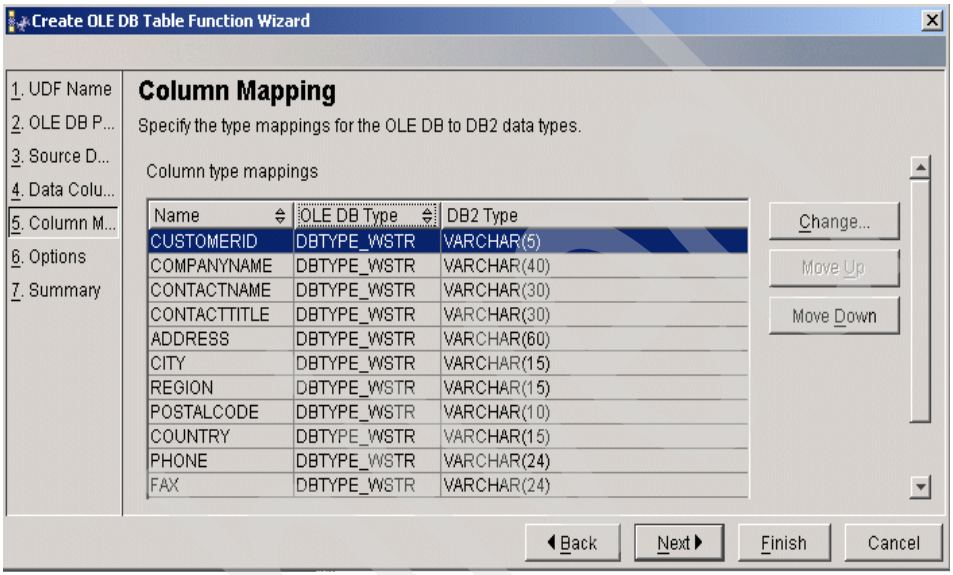

Figure 3-48 Development Center — OLE DB column mapping

The OLE DB UDF Options dialog shown in Figure 3-49 provides options in addition to creating the User-Defined Function. The options available are:

- **Create a corresponding table view:** This option will create a VIEW over the UDF we are creating, so the OLE DB data source can be accessed directly using SQL from a DB2 application as if it were a DB2 table. This option should be used if you don't want to move the data into a DB2 table. It should be noted that the performance of accessing the OLE DB data using this option is dependent on the OLE DB provider and generally should not be used for large volume data sources.
- **Create a new table and import the data into it:** This option will create a table with all the characteristics of the data source referenced by the UDF we are creating and copy the data into the new table. This provides a very easy method of converting data from an OLE DB data source like Microsoft Access, Microsoft Fox pro, or Microsoft SQL Server to DB2.

For our example we create a table named NWIND.CUSTOMERS and copy the data into it.

Select **Next** to proceed to the UDF Summary.

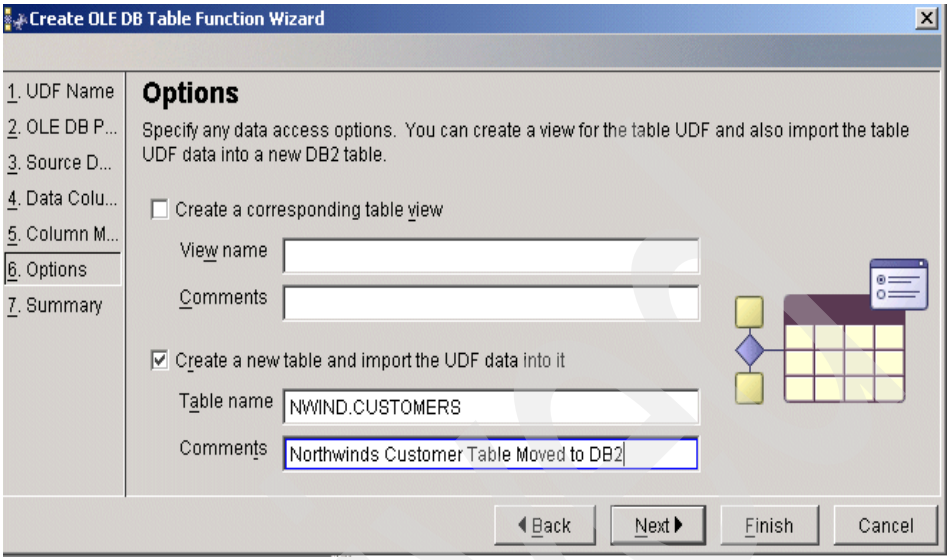

Figure 3-49 Development Center — OLE DB UDF options

The UDF Summary screen shown in Figure 3-50 provides a review of the actions to take place in the creation of the UDF. You can also review the SQL that will be generated and executed to create our UDF by selecting **Show SQL**. To create the UDF, create the new table and copy the data, then select **Finish**. After executing, you are returned to the Development Center, where you will see the newly created User-Defined Function. To see the table created, it can be viewed from the Control Center under Tables.

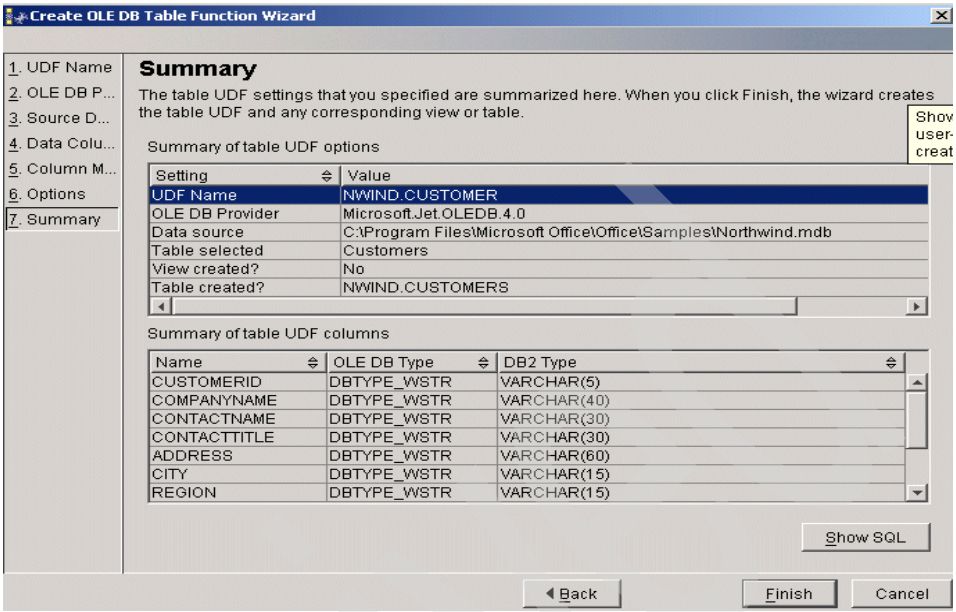

Figure 3-50 Development Center — OLE DB UDF summary

Moving data from another DB2 database

In this section we show how an existing DB2 table in another database can be duplicated in a database using DB2 Command Line Processor statements. The source table in our example is the sample table DSN8610.EMP on a DB2 for 390 V6 system named ZOSHOST with a location name of DB2zOS that is connected via TCPIP using a host name of zoshost. It could just as easily be a table on DB2 residing on a UNIX, Windows, or IBM iSeries as well as any data source supported by the federated database support in DB2 Enterprise Server Edition like Informix. The target table will be created in the SAMPLE database.

The first thing we need to do is to define the connections to the remote data source. This is a one-time step for each remote database management system. The CATALOG TCPIP NODE, CATALOG DATABASE and CATALOG DCS DATABASE statements below define this connection for our example. If you would prefer using a GUI tool for defining the connection, the DB2 Configuration Assistant could also be used instead of using the following commands:

- Set up the DB2 Connect connection to DB2 390 Named ZOSHOST.
- The REMOTE Parameter specifies the hostname of the MVS System.
- The SERVER Parameter specifies the TCPIP port number.

- The DB2 subsystem used, which by default is port 446:

```
CATALOG TCPIP NODE DB2zOS REMOTE zoshost SERVER 446
CATALOG DATABASE DB2zOS AS DB2zOS AT NODE DB2zOS
AUTHENTICATION DCS
```

- The locationname is the location assigned to the DB2 390 System:

```
CATALOG DCS DATABASE DB2zOS AS TSTDB201
```

If you haven't enabled federated database support in DB2 UDB Enterprise Server Edition the following database manager configuration change must be made followed by stopping and restarting the database manager. This step is only required one time and can also be done using the Configuration Assistant or the configure parameters for the instance with the Control Center.

Enable Federated Database Support in DB2 UDB:

```
UPDATE DATABASE MANAGER CONFIGURATION USING FEDERATED YES
STOP DATABASE MANAGER
START DATABASE MANAGER
```

Now that the connection has been defined we need to define the federated database capability to our target database, this is only required one time for each source database system. The following steps could also be performed using the Control Center under Federated Database Objects. We need to be connected to the target database SAMPLE to perform these steps.

```
CONNECT TO SAMPLE
```

The CREATE WRAPPER statement registers a wrapper to a federated database. A wrapper is a mechanism by which a federated server can interact with a certain category of data sources. In our example we use the DRDA wrapper.

Define DB2 on 390 as a Federated Database

```
CREATE WRAPPER DRDA LIBRARY 'db2drda.dll'
```

The CREATE SERVER statement defines a data source to a federated database. It assigns a name to the server and identifies the wrapper used by the server. In our example we give the server a name of DB2zOS that is a DB2/390 Version 6 system using the DRDA wrapper. The AUTHID identifies an authorized Userid and password to access the DB2 390 system, this may require a change for your system. The NODE parameter identifies the node we cataloged above. The DBNAME parameter identifies the name we used in the CATALOG DCS DATABASE command above.

```
CREATE SERVER "DB2zOS" TYPE DB2/390 VERSION 6
   WRAPPER "DRDA"
    AUTHID DB2ADM PASSWORD DB2ADM
 OPTIONS ( NODE 'DB2zOS', DBNAME 'TSTDB201' )
```

The CREATE USER MAPPING statement defines a mapping between an authorization ID that uses a federated database and the authorization ID and password to use at a specified data source. This will be required if the userid/password on your system needs to be mapped to another userid/password on the DB2 390 system.

```
CREATE USER MAPPING FOR "db2admin" SERVER "DB2zOS"
   OPTIONS( REMOTE_AUTHID 'DB2ADM',
   REMOTE_PASSWORD 'DB2ADM')
```

Up to this point in our example we have performed the one time steps necessary to connect to a remote database using federated database support. The following steps can be performed for each table you would want to copy.

For each table on the remote system we would like to access as if it were present in the database we are connected to we need to create a nickname to assign a name the table will be known as on this system. This is done with the CREATE NICKNAME SQL statement. You specify the name you want to give the table, the server it resides on and the name of the table on the remote system. Once the nickname is created the table can be referenced by the nickname given:

```
CREATE NICKNAME zOS.EMP FOR DB2zOS.DSN8610.EMP
```

Now that the table is accessible we can now use the same technique we used in “Moving data from one table to another in the same database” on page 129.

We then need to create the new table that we will be duplicating. We use the “CREATE TABLE LIKE” SQL statement that will create a new table with the same attributes as another table, in our case we use the nickname zOS.EMP as a model to create the new table NEW.EMP:

```
CREATE TABLE NEW.EMP LIKE zOS.EMP
```

After creating the new table we need to populate it with the data from the original table. We use a capability of the LOAD command that can use a SQL SELECT statement as input to the load process. The input data is defined by using the DECLARE CURSOR statement that defines the input data. Any valid SELECT statement can be used. In our example we select all columns and rows from the source table:

```
DECLARE LCUR CURSOR FOR SELECT * FROM zOS.EMP
```

Once the input is defined with the DECLARE CURSOR, we invoke the LOAD command specifying the cursor defined above and the table the data is to be loaded into, in our example NEW.EMP:

```
LOAD FROM LCUR OF CURSOR INSERT INTO NEW.EMP
```

After the load is complete we disconnect from the database:

```
CONNECT RESET
```

Another variation of this technique would be to use the SQL INSERT statement instead of the DECLARE CURSOR statement and the LOAD command as follows:

```
INSERT INTO NEW.EMP SELECT * FROM zOS.EMP
```

The disadvantage of this technique is that INSERT is normally much slower than using LOAD and all inserted rows will be logged.

3.6 Design Advisor Wizard

The Design Advisor Wizard will determine the best set of indexes for a given workload. A workload contains a set of weighted SQL statements that can include queries as well as updates. The wizard will recommend which new indexes to create, which existing indexes to keep, and which existing indexes to drop.

When the Design Advisor recommends indexes, these are not immediately created; instead they are stored in an Explain table called ADVISE_INDEX. Information can also be inserted into this table using the db2advis tool or manually using SQL.

The workload information which is considered by the Design Advisor Wizard is stored in an additional Explain table ADVISE_WORKLOAD.

The recommendations provided by the Design Advisor are only as good as the input you provide it. To ensure accurate recommendations the Design Advisor should only be run after your database is populated with data and the SQL you provide it is an accurate representation of your workload.

Before running the Design Advisor Wizard against a database, ensure that the Explain tables have been created for that database. Run *db2 -tf EXPLAIN.DDL* against the database to create these tables. EXPLAIN.DDL is found in the *sqllib/misc* directory.

3.6.1 Using the Design Advisor Wizard

In this section we show you how to use the Design Advisor Wizard.

Starting the Design Advisor wizard

To start the Design Advisor wizard from the Control Center, highlight the database NEWDB that we created before, then depress mouse button two and select **Design Advisor**, as shown in Figure 3-51.

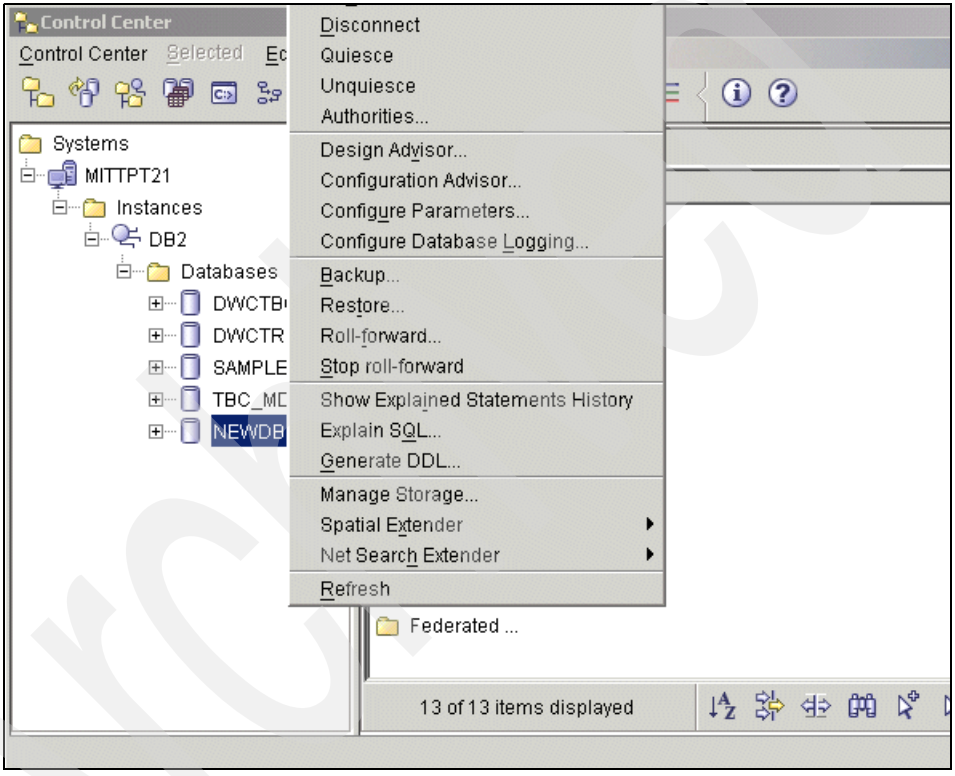

Figure 3-51 Starting the Design Advisor wizard

Design Advisor introduction

The Design Advisor Introduction screen shown in Figure 3-52 identifies the database we are working with and gives you an overview of the purpose of the wizard. Select **Next** to proceed.

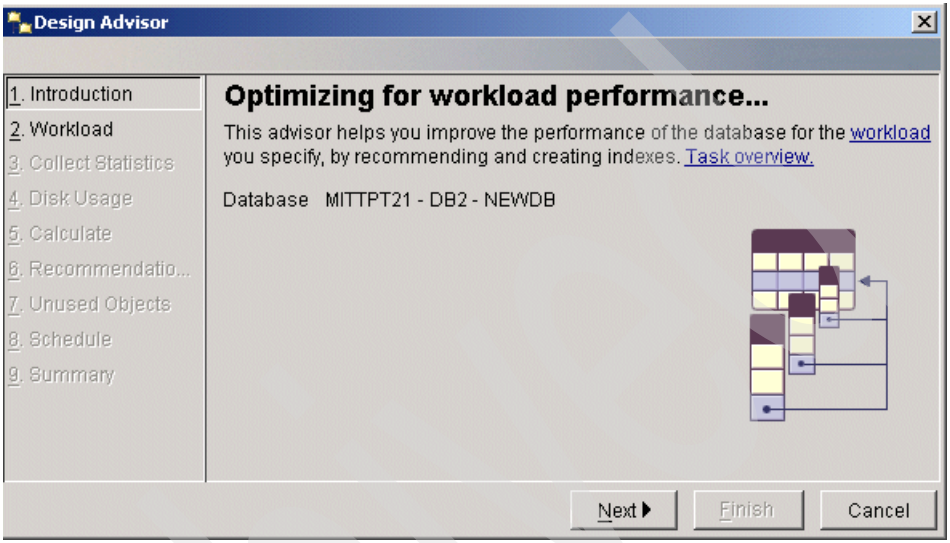

Figure 3-52 Design Advisor introduction

Defining a new workload

With the screen shown in Figure 3-53 we define the workload to be used by the Design Advisor Wizard, so that recommendations can be made on what indexes we could create to minimize the total workload cost.

You can add individual SQL statements or import SQL from existing packages which the database would normally have to process over a given period. Statements in the workload can be added or removed as desired.

More than one workload definition can be created; each is stored in the ADVISE_WORKLOAD table.

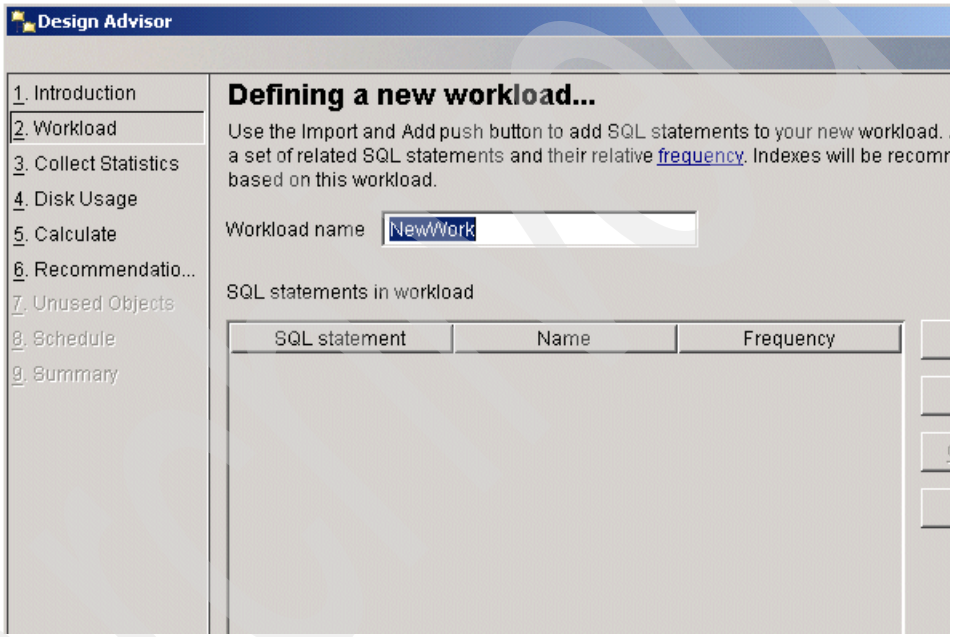

Figure 3-53 Defining a new workload

Updating catalog statistics

In order to provide the best recommendations, the table statistics in your database must be current. If you want to update the table statistics, select the tables required by the specified workload as shown in Figure 3-54. Updating the statistics will increase the calculation time.

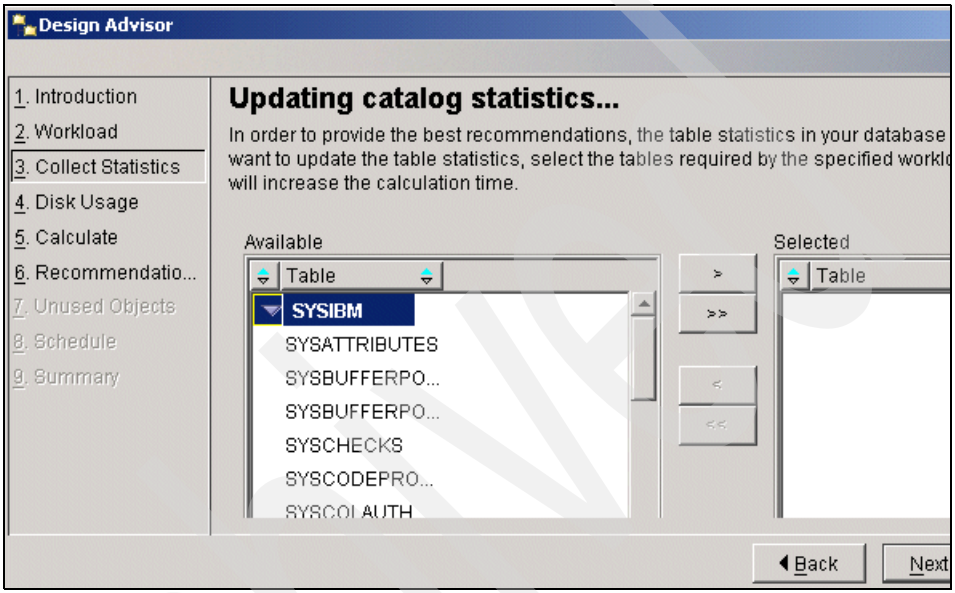

Figure 3-54 Updating catalog statistics

Setting the disk usage

Use the Setting disk usage screen shown in Figure 3-55 to specify the default table space and combined maximum disk space for the recommended objects. By setting a specific value for the maximum disk space you wish to allocate for any new indexes. It is recommended that a value be specified if disk space is scarce.

If you decide to set a disk space limit, be aware that the indexes recommended may not be the most optimal for your workload, as the Design Advisor Wizard may discard some indexes because they would exceed the available disk space limit. If you want to see what could be created regardless of disk space, do not set a limit.

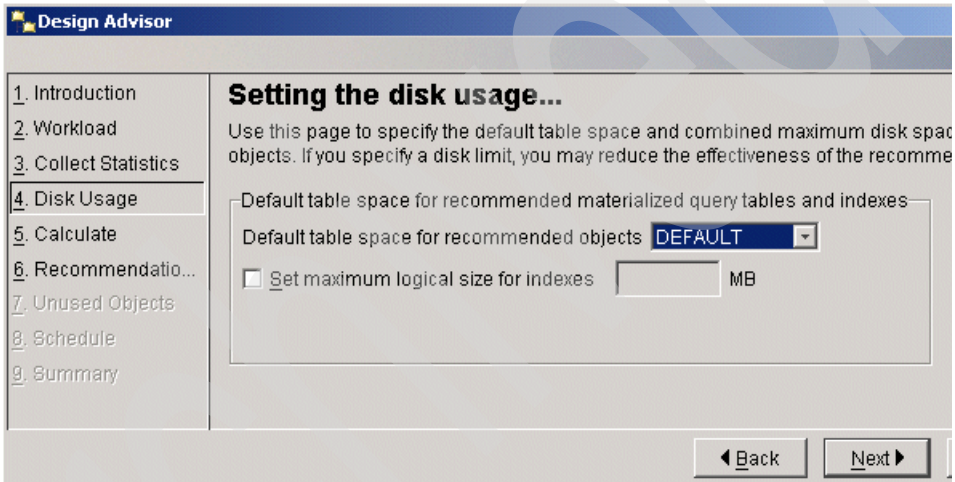

Figure 3-55 Setting the disk usage

Selecting when to calculate the recommendations

If your workload consists of many complex queries, then it may take some time to calculate the recommended indexes. The screen shown in Figure 3-56 can be used to determine not only when the calculations will be performed, but also the maximum amount of time they are allowed to run for. By using a scheduled time, you can allow the Design Advisor to run at a time when database resources are not being fully utilized (such as at night or on a weekend). You can also limit the length of time the wizard can run, thereby limiting execution time to a set window.

If you do specify a time limit in which the wizard can run, then the index recommendations made may not be the most optimal, as the wizard may not have been allowed the time required to calculate the best indexes for your workload.

However, if the results are returned before the time limit, then assume that these are the most optimal recommendations that the wizard was able to make.

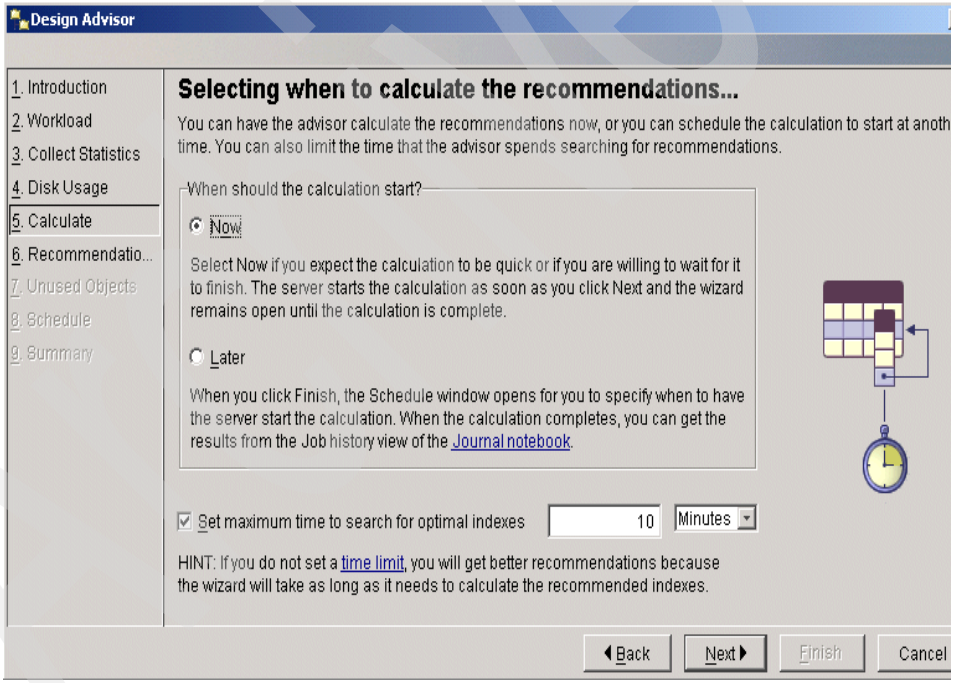

Figure 3-56 Selecting when to calculate recommendations

Selecting recommended objects for creation

Once the Design Advisor Wizard has completed, you will see a screen similar to that shown in Figure 3-57. The wizard will have recommended the indexes that can be created to help minimize the overall cost of this workload on the databases resources. For each index, you will see the table the index was created against, what columns were used to create the index, and what the disk space requirements to hold the index will be. We can also change the index name from this screen.

At the bottom of the screen is a section called Workload performance, which shows the improvement in workload duration time, if the new indexes are created. The values show are measured in timerons. Timerons do not correlate directly to any specific unit of elapsed time, but provide an estimate of the resources required by the database manager to execute the queries within the workload. Resources would include CPU (instructions required) and I/O (page transfers and seeks required).

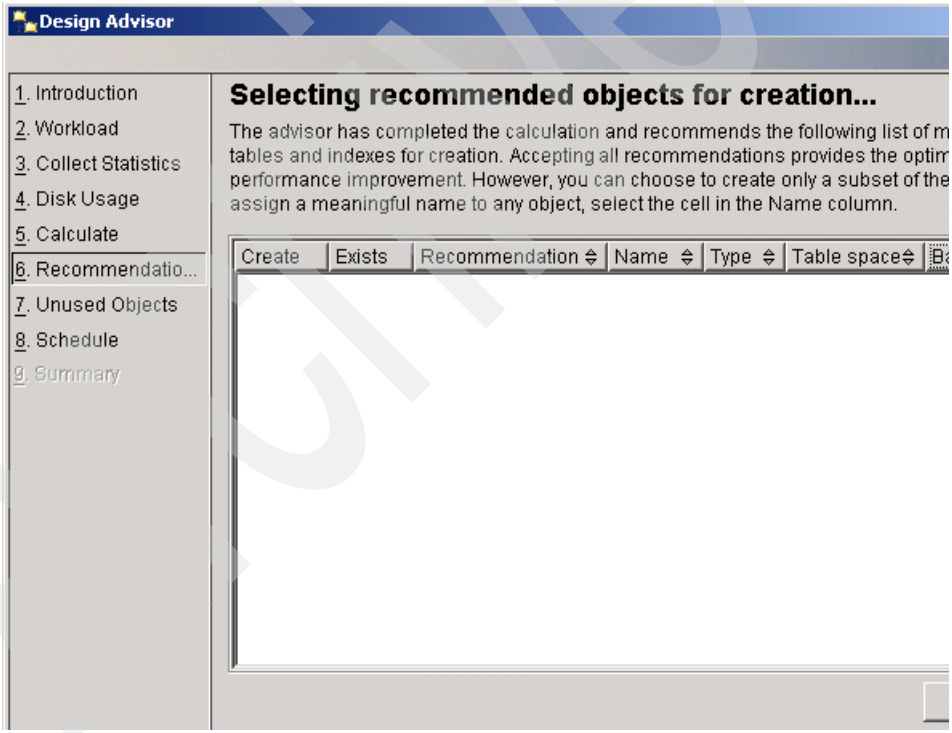

Figure 3-57 Selecting recommended objects for creation

Reviewing unused objects

The screen shown in Figure 3-58 shows which indexes the wizard flagged as being unused during the execution of the workload. Do not remove indexes unless you are sure that they are not required by other applications or workloads.

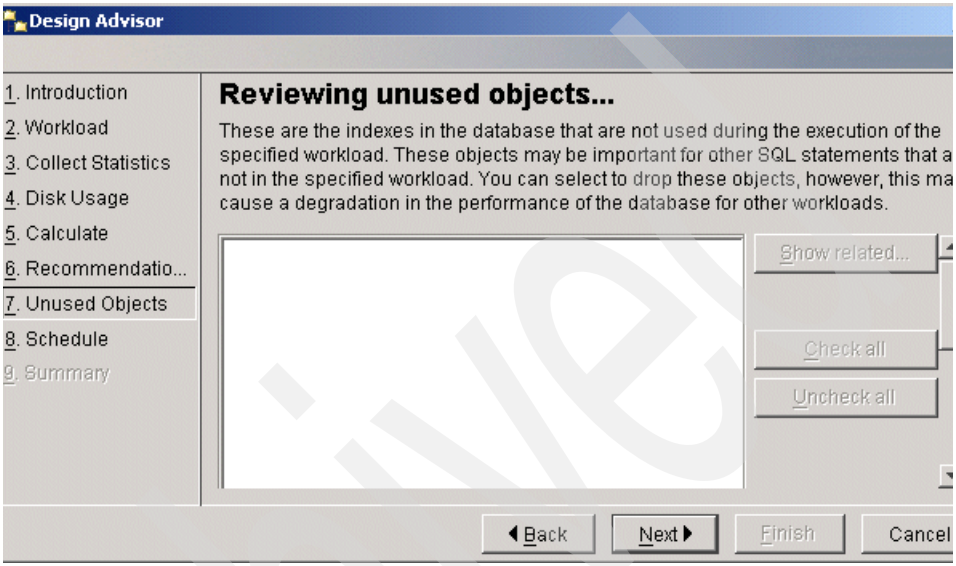

Figure 3-58 Reviewing unused objects

Scheduling task execution

Figure 3-59 shows how you can select whether to execute the commands immediately or create a task in the DB2 UDB Task Center. Creating a task allows you to schedule task execution and maintain its history.

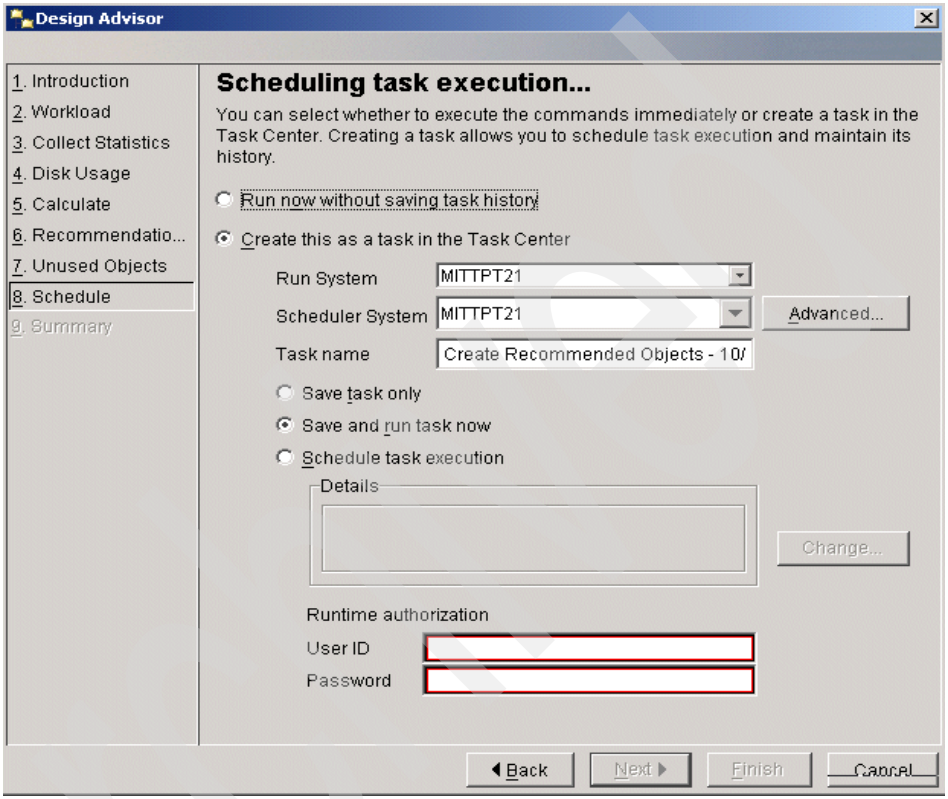

Figure 3-59 Scheduling task execution

Review summary

The next screen, Figure 3-60 will show a summary of the selections we have made in the previous screens; these include which indexes to create, existing indexes to keep, and indexes to remove. At this point you can save the actions to a script (shell script or DB2 script), and also carry them out immediately or schedule them to run later.

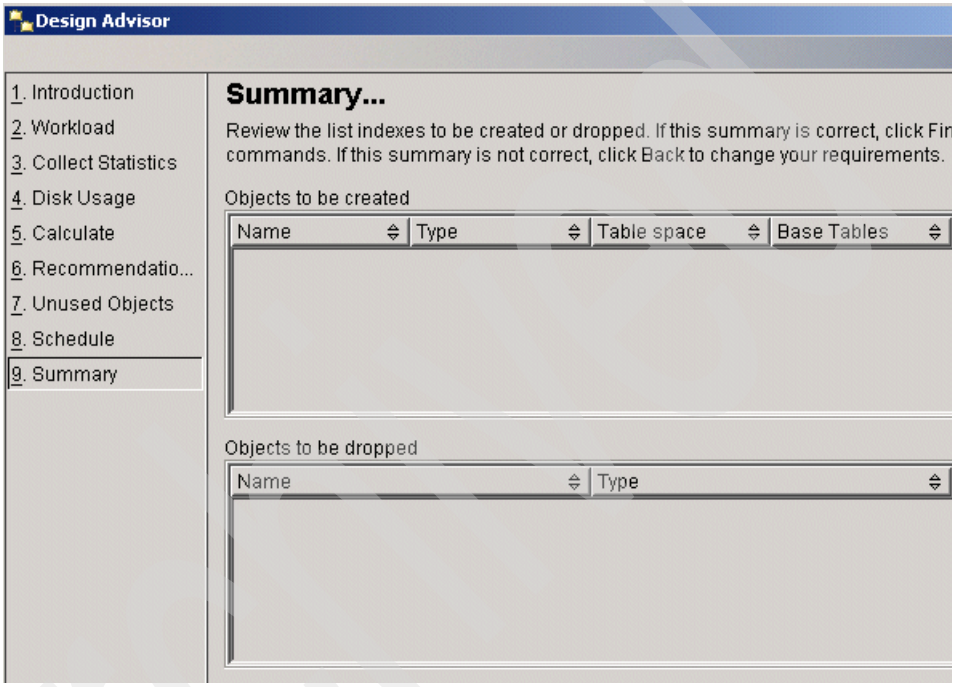

Figure 3-60 Design Advisor — summary

4

Security

In today's world, hackers, crackers, and spies present security threats from outside and within the enterprise. The best defense against security threats is a good offense. And implementing a good security offense requires a complete understand of how security works in your environment. In order to fully exploit the security features of the Windows operating system, we must first have a good understanding of how security works in this environment.

In this chapter we describe how DB2 Universal Database V8.1 exploits the security features built into the Windows 2000 operating system. We look specifically at the Windows 2000 security model and point out differences between Windows NT and Windows 2000 along the way. We first introduce the Windows security model, then take a brief look at the basic security concepts such as authentication and authorization. After laying the foundation of the Windows security model, we then describe in detail the steps required to implement security for DB2 UDB in the Windows environment. Specifically, we look at the DB2 security at the system, instance, and database levels.

This chapter consists of the following sections:

- Understanding Windows security
- System level security
- Instance level security
- Database level security

4.1 Understanding Windows security

The basic concepts of the Windows security model apply to both Windows NT and Windows 2000 operating systems, although the implementation of the model is slightly different between these two versions of the operating system. Unless otherwise indicated, we are talking specifically about Windows 2000.

4.1.1 Basic security concepts

The Windows security model is based on two simple concepts, authentication and authorization. Authentication is a concept that is best described with the question: Are you who you say you are? Authorization, on the other hand, is best described with the question: Are you able do what you want to do?

Authentication

User authentication in Windows 2000 typically occurs when a user logs on to a computer with a userid and password. The process of matching the userid with the password answers the question: Are you who you say you are?

The Windows security model provides for authentication by storing userid and password information in a security database called a Security Accounts Manager (SAM) database. On Windows 2000 the Security Accounts Manager service authenticates userid and password at logon or at any attempts to gain access to a network resource.

Authentication of the user account can take place either on the local machine or on the domain. For example, if authorized a user can logon locally at the Windows workstation and be authenticated by the local SAM database on the local machine. The same user, if authorized, can also logon to the domain, if the computer belongs to one, and be authenticated by the domain SAM database.

Note: User account information that was traditionally stored in a domain SAM database on Windows NT is now stored in the Active Directory on Windows 2000 domains.

Authorization

Once the user has been authenticated with a matching userid and password, the windows security services stop asking “Are you who you say you are?”, and start asking “Are you able (authorized) to do what you want to do?”

Authorization in the Windows security model is controlled by Access Control Lists (ACL). This process is for the most part transparent to users, that is, until the point in which the user attempts to perform some activity for which they are not authorized.

For example, computers are one of the most common resources in a Windows domain. When a user attempts to access a computer in the domain, the security accounts manager service checks ACL to determine if the user is explicitly authorized to access that computer. A user may also be implicitly authorized to access network resources by being a member of one or more groups.

Note: In Windows 2000 domains, Access Control Lists (ACL) are stored in the Active Directory as attributes called security descriptors.

4.1.2 Windows 2000 domains

The Windows 2000 domain model takes most of its core features from the Windows NT domain model, but Windows 2000 adds significant functionality to it.

Domain names

Like the Windows NT domains, Windows 2000 domains use a unique network name to define the domain. In Windows 2000, the domain name follows the Domain Name System (DNS) standards which implement a hierarchy of names separated by periods (.) or more appropriately called dots, for example, ibm.com.

This model also introduces the concept of domain trees. The domain tree is defined by one or more domains that share a common domain name space. For example, almaden.ibm.com is the parent domain of itso.almaden.ibm.com, which in turn is the parent domain to redbooks.itso.almaden.ibm.com. These domains all share a common domain name space in the domain tree. The hierarchy is established by adding child domains to existing parent domains. For example, almaden.ibm.com is a child domain to ibm.com and both domains share the same domain name space.

The domain forest is defined by one or more domain trees that share a common root domain, for example, the domain forest ibm.com can be established by creating the first domain within the organization and establishing a new forest called ibm.com. Subsequent domains such as almaden.ibm.com and toronto.ibm.com establish their own domain trees (toronto.ibm.com) but still belong to the same domain forest (ibm.com). The domain that establishes the domain forest is referred to as the root domain; see Figure 4-1.

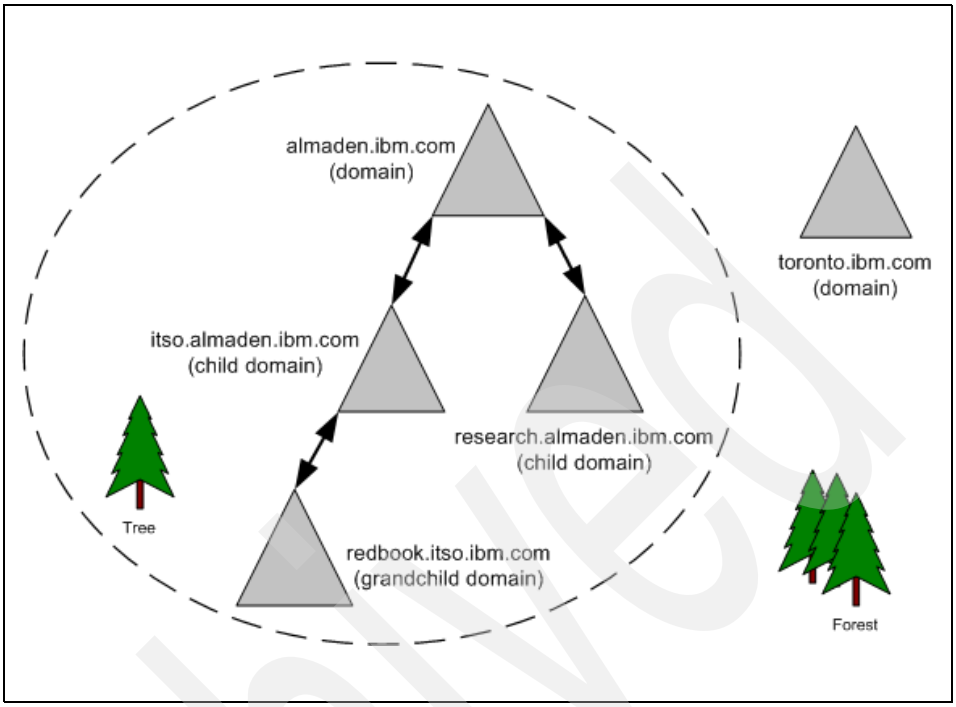

Figure 4-1 Windows 2000 domain forest and trees

Domain trusts

Windows supports the concept of single sign-on by establishing trust relationships between domains within a domain tree, and domain trees within a domain forest. These trust relationships allow users to authenticate at any domain within the forest and subsequently forage for resources without having to provide a userid and password more than once. Domain trusts can be managed using the MMC Snap-In called Active Directory Domains and Trusts.

Unlike the flat Windows NT domain model, the Windows 2000 domain model is based on a logical hierarchy of domain trees. All domains within a domain tree establish an implicit two-way trust relationship with each other. Child domains (itso.almaden.ibm.com) created within the domain tree automatically trust the parent domain (almaden.ibm.com) and vice versa. Additionally, domain trees added to a domain forest (ibm.com) also establish implicit trust relationships.

Domain modes

Windows 2000 can be created in two different modes: mixed-mode and native-mode. The default for Windows 2000 domains is mixed-mode, which allows Windows 2000 domain controllers to Windows NT backup domain controllers to co-exist within the same domain. Once all of the Windows NT backup domain controllers have been replaced, the domain mode can be upgraded to native-mode using the MMC Active Directory Domains and Trusts. This is important to point out, as mixed-mode domains do not have all of the same characteristics as do native-mode domains.

Domain controllers

In the Windows 2000 domain model, there is still the concept of domain controllers (DC), however the concept of the single Primary Domain Controller (PDC) and one or more Backup Domain Controllers (BDC) no longer applies. Windows 2000 domains can support Windows NT BDC if the domain is created in mixed-mode.

In Windows 2000, all domain controllers in the domain maintain an updateable copy of the SAM database, which is stored within the Windows 2000 Active Directory and replicated between all domain controllers in the domain.

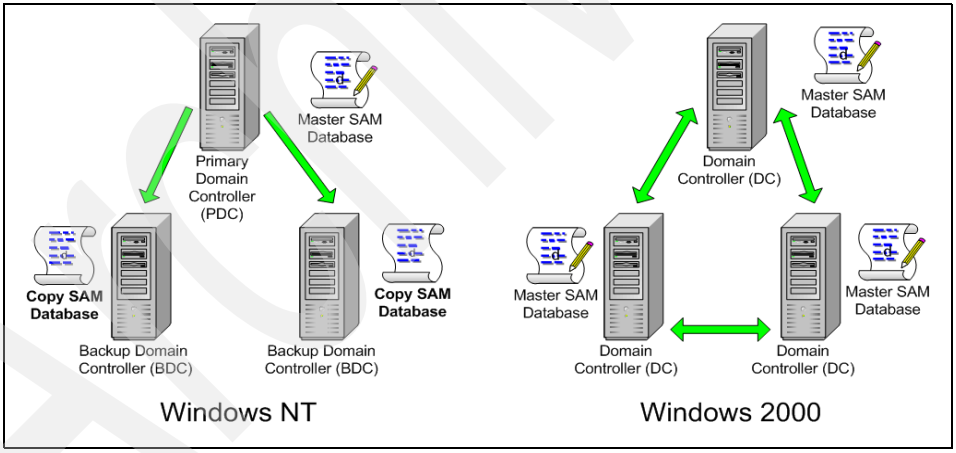

Figure 4-2 Comparison of windows domain models

One significant difference between the first domain controller defined within a Windows 2000 domain and the rest of the domain controllers is that the first domain controller becomes, by default, the operations master. The operations master can emulate the functions of the Windows NT primary domain controller in order to support pre-Windows 2000 clients.

Users and groups

The Windows 2000 domain model supports two basic user accounts: local users and domain users. Local user accounts can only be created in a local SAM database and therefore can only be created on Windows workstations, stand-alone servers, and additional servers (non-domain controllers) in a domain. On the other hand, domain user accounts can only be created in a domain SAM database which is stored within Active Directory on primary domain controllers within the domain.

The Windows security model simplifies the management of controlling access by implementing groups. A group account is simply a container for user accounts, called members. Group accounts simplifies the management process because a single ACL is maintained a group of users with similar security needs instead of ACL for each user account.

Windows NT groups

In Windows NT environments, there are three basic types of user groups: local groups, domain groups (sometimes called domain local groups), and global groups (sometimes called domain global groups). Local groups can only be created in local SAM databases, and domain or global groups can only be created in domain SAM databases. Global groups are a special type of domain group that can only have user accounts from the same domain as members. Global groups have no explicit privileges of their own, but rather are designed to be included in domain groups from the same domain or from other trusted domains.

In Windows NT, there is also a special group called *Everyone*. This group is controlled by the operating system or domain. In a Windows NT domain, every user account authenticated at the domain is a member in the *Everyone* group.

Windows 2000 groups

Although for the most part, groups behave the same in both Windows NT and Windows 2000, the latter adds a slight twist on the concept of groups. Groups in Windows 2000 have two possible attributes: scope and type. The group scope indicates the extent to which the group can travel among the trees and forest. The group type indicates the purpose of the group.

In Windows 2000, there are two group types and four different group scopes. The types of groups are security and distribution. The four different group scopes are local, domain, global, and universal. User accounts can be members of one or more of these groups.

The basic difference between a security group and a distribution group is the presence or absence of an access control list. Distribution groups do not have security enabled, because their primary purpose is for distribution applications such as e-mail.

In Windows 2000, the special group called *Everyone* does not exist, as it has been replaced by a group called *Authenticated Users*. The Authenticated Users group, much like the Everyone group, is controlled by the operating system or domain. The main difference between the *Everyone* group that existed in Windows NT and the Windows 2000 *Authenticated Users* group is that the latter does not have as members Anonymous or guests user accounts and it can't be used to assign permissions.

On Windows 2000 workstations, standalone servers, and additional servers (non-domain controllers in a domain), the local SAM database can be managed by the MMC Snap-In called Local Users and Groups (**`Start -> Programs -> Administration Tools -> Computer Management`**); see Figure 4-3.

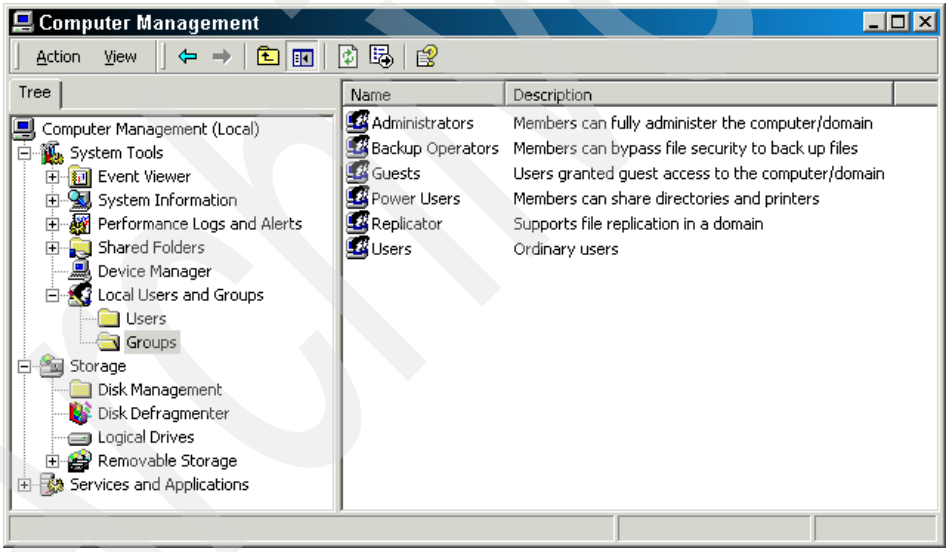

Figure 4-3 Computer Management — Local Users and Groups

On Windows 2000 Domain Controllers, this snap-in is disabled during the process of promoting the server to a domain controller and is replaced with the MMC Snap-In called Active Directory Users and Computers (**`Start -> Programs -> Administration Tools -> Active Directory Users and Computers`**); see Figure 4-4.

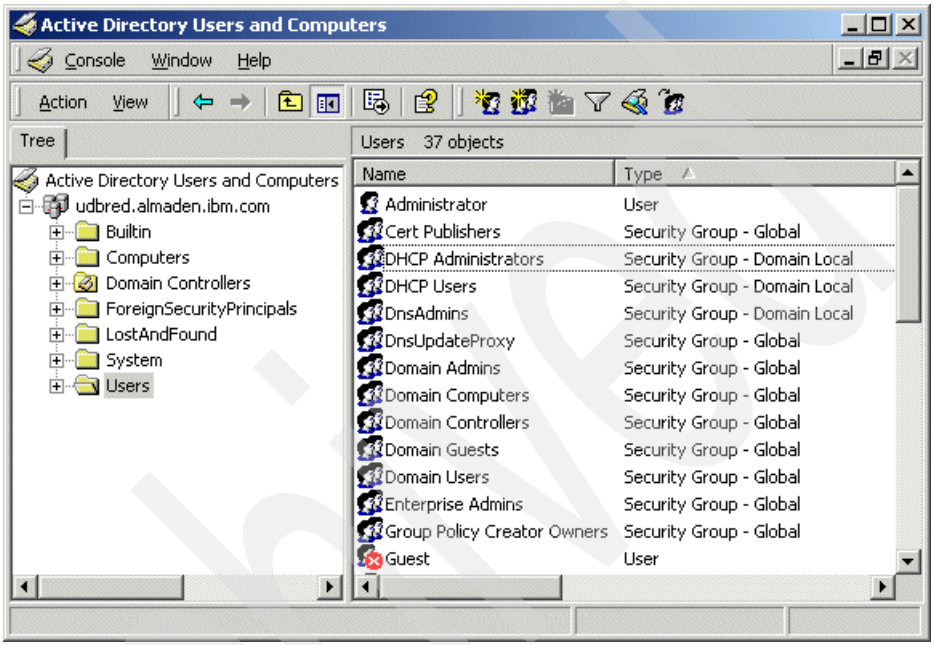

Figure 4-4 Active Directory Users and Computers

Because user accounts can belong to more than one group, and they usually do, the Windows security model must implement a mechanism for finding out all of the groups that an individual user account belongs to. This process is called group enumeration and is discussed later in this chapter.

Local groups

Local groups can only be created in local SAM databases. Since Windows 2000 Domain Controllers do not have a local SAM database, you cannot create a local group on a Windows 2000 domain controller. Only Windows workstations, stand-alone servers, and additional servers (non-domain controller servers in a domain) have a local SAM database, therefore local groups can only be created on these types of machines.

A local group can have as members both users and groups from the local SAM database. If the machine is a member of a domain, the local group can also have as members domain users and groups (users and groups from the domain SAM database) as well as any domain groups from any trusted domains.

Domain groups

Domain groups, sometimes called domain local groups, can only be created in domain SAM databases. Since only Windows 2000 Domain Controllers (DC) have domain SAM databases, domain groups can only be created on these Windows 2000 Domain Controllers. Windows 2000 domain controllers do not have local SAM databases, therefore local groups cannot be created on them.

In Windows 2000, native-mode domain groups can have as their members user accounts, global groups, and universal groups from any domain, as well as local groups from the same domain. They can be granted permissions only within the same domain. In Windows 2000 mixed-mode domain groups can have as their members user accounts and global groups from any Windows 2000 or Windows NT domain.

Global groups

In Windows 2000, native-mode global groups can have as their members accounts from the same domain and global groups from the same domain. They can be granted permissions in any domain in the forest. In Windows 2000 mixed-mode global groups can have as their members accounts from the same domain.

Universal groups

In Windows 2000, native-mode universal groups can have as their members accounts from any domain, global groups from any domain, and universal groups from any domain, and can be granted permissions in any domain in the domain tree or forest. In Windows 2000, native-mode universal groups can only be created as a distribution group, in other words, they cannot have permissions assigned.

Did you know that Windows 2000 is loosely synchronized?:
Windows 2000 operating systems implement the Kerberos v5 protocol for authentication. One important aspect of this security protocol is time synchronization among the servers in the domain. Windows 2000 implements a service called the Windows Time Service (w32time) that provides operating system time synchronization that is characterized as being “loosely synchronized”.

4.2 System level security

In this section we provide an introduction to DB2 security concepts at the system level. We look at how DB2 integrates with the security features at the operating system level.

Very much like the Windows security model, DB2's security implements the same basic concepts of authentication and authorizations. In fact, DB2 relies on the Windows security services to perform authentication of user accounts by calling upon the Security Accounts Manager (SAM) services to authenticate userid and passwords. Once the user has been authenticated, DB2 will then enumerate the account to obtain a complete list of groups the user is a member of and then cache this back on the DB2 servers.

We first consider security requirements for running DB2 services under local system, local user, and domain user accounts. We then explore how DB2 authenticates users in local, domain, and trusted domain Security Accounts Manager (SAM) databases. Finally, we take a look at how DB2 handles users that belong to more than one local, domain, and domain global group.

Installation note: In order to install DB2 on Windows, you must logon with a user account that belongs to either the local or domain Administrators group. Although not required, the advanced user right called *Act as part of the operating system* is necessary for the user account performing the installation to validate the user accounts used for DB2 to run as a service.

4.2.1 DB2 service accounts

During the installation of DB2 several services are created. Most of these services are configured to logon with a Local System account. Some of the DB2 services require security privileges beyond those available to the Local System account and therefore must be defined with user local or domain account.

What's new: The installation wizard for DB2 UDB V8.1 for Windows operating systems has a new feature that allows the installer to select a domain user account for running DB2 services such as the DB2 Administration Server and the default DB2 Instance.

System accounts

The LocalSystem account is a Windows operating system level account that is built into the Security Accounts Manager (SAM) database. The LocalSystem account is the most commonly used account for running DB2 services because:

- It is, by definition, a member of the local Administrators group.

- It already contains all of the advanced user rights, such as *Act as part of the operating system*.
- It does not require any password maintenance, that is, the LocalSystem account does not have a password that expires on a regular basis.

However, there are two items you should be aware of when using the LocalSystem account to run DB2 services:

- This account does not have access to LAN shares. This can cause problems if files to be accessed are on LAN drives or there are LAN drives in the PATH and DB2 is asked to search the path to find Stored Procedures, user defined functions, etc.
- Use of this account is generally discouraged on Domain Controllers because any code running under it has all the advanced rights.

Figure 4-5 shows some of the services created during the installation procedures as well as the *Log On As* accounts.

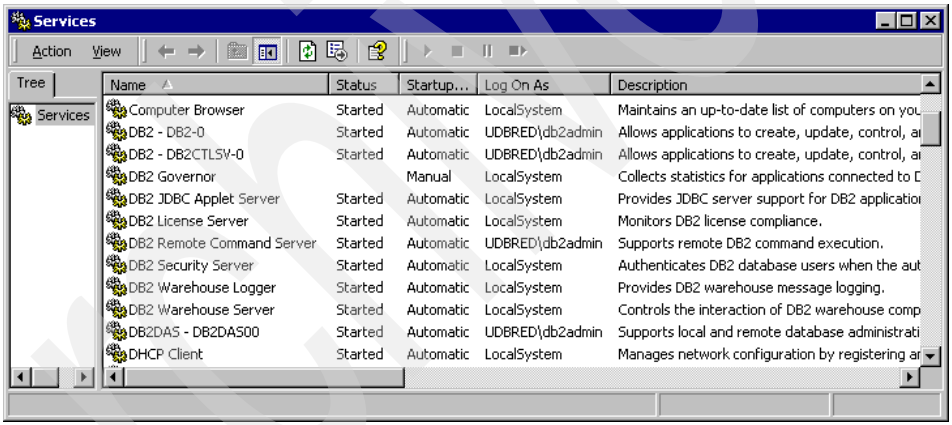

Figure 4-5 DB2 services

User accounts

A local user or domain user account may be required for the DB2 Administration Server (DAS), the DB2 database instances, and other DB2 services running on your server. If you are installing DB2 on a Windows 2000 domain controller, then this account will be a domain user account. If you are installing DB2 on an additional server in your Domain, then you have two options, a local user account or a domain user account.

What's new: In DB2 Universal Database V8.1, the Database Administration Server (DAS) is no longer an instance. In previous versions of DB2, the DAS instance name was DB2DAS00.

You can create the user account required for the various DB2 services before installing DB2, or you can simply elect to have the DB2 Setup Wizard create it for you. If you let the DB2 Setup Wizard create the user account, it will be added to the Administrators group on the machine where you are performing the installation. After installation is complete, you will most likely want to remove this service account from the Administrators group and place it in a more appropriate group.

If you create this account, you will need to temporarily add it to the Administrators group on the machine where you will perform the installation. Additionally, the account must have the following advanced user rights:

- Act as part of the operating system
- Create token object
- Increase quotas
- Logon as service
- Replace a process level token
- Lock pages in memory

After the installation of DB2, you may want to consider removing the DB2 service account from the Administrators group into a more appropriate group such as the Power Users. The primary purpose of the Database Administration Server (DAS) service is to support the administration tools to administer instances on local and remote DB2 servers. This service will require a user account that can start and stop other services. It should also be a member of the DB2 System Administrators group (SYSADM).

DB2 Security Server service

DB2 provides for single sign-on support by trusting that the authentication of the userid and password has already been performed. In order for DB2 to be assured that the authentication has already been performed, the DB2 Security Server service is installed by the DB2 Setup Wizard.

On DB2 servers, the code required to authenticate user accounts has been integrated into the DB2 System Controller Service (db2syscs.exe). This DB2 Security Server service is only required for DB2 clients that will connect to a DB2 server that is configured with authentication client (AUTHENTICATION=CLIENT) and with trust client authentication (TRUST_CLNTAUTH=CLIENT). A userid and password must also be passed as part of the connect statement, otherwise the server will assume the user has already been authenticated.

4.2.2 DB2 user authentication

As we mentioned at the beginning of this section, DB2 relies on the security services of the Windows operating system to authenticate user accounts. Before a user can access DB2 instance or database resources, the user's account must be authenticated.

Client or server

In DB2 Universal Database, there are several instance level configuration parameters that allow you to control where and how the authentication of user accounts is performed. The DB2 instance (dbm) configuration parameter called AUTHENTICATION provides support for both server-side and client-side authentication.

Server authentication

By default DB2 is configured for server-side authentication. That is DB2 will start the process of user account authentication on the server in which the DB2 instance is located. In the end, the actual authentication process may occur on another server in another domain.

If you are attaching to a DB2 instance or connecting to a DB2 database configured with server authentication from a remote DB2 client, you will always require a user account and password to be provided as part of the connection string. If you are accessing DB2 resources from a local client on the DB2 server itself, then a user account and password are not required, but will be authenticated if provided.

The authentication type for both DB2 clients and servers can be modified with the DB2 Control Center, if installed, or the DB2 command:

```
db2 update dbm cfg using AUTHENTICATION SERVER
```

Server encryption

In addition to server-side authentication DB2 also support server encrypted authentication. Server authentication does not provide support for encrypted passwords — meaning that unencrypted (clear text) passwords travel across the network from client to server, among trees throughout the forest. DB2 instances configured with authentication SERVER_ENCRYPT allows the server to authentication user accounts that have encrypted passwords with the attach or connect statements.

Here's how authentication SERVER_ENCRYPT works:

- If the client authentication is not specified, the client is authenticated using the method selected at the server.

- If the client authentication is SERVER, the client is authenticated by passing the userid and password to the server.
- If the client authentication is SERVER_ENCRYPT, the client is authenticated by passing an encrypted userid and encrypted password.
- If SERVER_ENCRYPT is specified at the client and SERVER is specified at the server, an error is returned because of the mismatch in the authentication levels.

The authentication type for both DB2 clients and servers can be modified with the DB2 Control Center, if installed, or the DB2 command:

```
db2 update dbm cfg using AUTHENTICATION SERVER_ENCRYPT
```

DCE MIA?: In DB2 UDB V8.1 support for DCE security has been removed in response to the industry move towards Kerberos as the mechanism for secure network authentication and single sign-on. DCS and DCS_ENCRYPT now have exactly the same meaning as SERVER and SERVER_ENCRYPT.

Kerberos authentication

The Kerberos security protocol performs authentication by creating a shared secret key. This key is used by the user to gain access to network resources much like we use key cards today. Once the key has been assigned to a user account, the user account no longer needs to provide a user account and password to gain access to network resources.

In order to implement KERBEROS authentication with DB2, you must be running both client and server operating systems that support the Kerberos security protocol. Among the Windows operating systems that support Kerberos are: Windows 2000, Windows XP, and Windows Server 2003.

Here's how authenticating KERBEROS works:

1. A user logging on to the client machine using a domain account authenticates to the Kerberos key distribution center (KDC) at the domain controller. The key distribution center issues a ticket-granting ticket (TGT) to the client.
2. During the first phase of the connection the server sends the target principal name, which is the service account name for the DB2 server service, to the client. Using the server's target principal name and the target-granting ticket, the client requests a service ticket from the ticket-granting service (TGS) which also resides at the domain controller. If both the client's ticket-granting ticket and the server's target principal name are valid, the TGS issues a service ticket to the client.

3. The client sends this service ticket to the server via the communication channel (which may be, as an example, TCP/IP).
4. The server validates the client's server ticket. If the client's service ticket is valid, then the authentication is completed.

If a userid and a password are specified, the client will request the ticket-granting ticket for that user account and use it for authentication.

The authentication type for both DB2 clients and servers can be modified with the DB2 Control Center, if installed, or the DB2 command:

```
db2 update dbm cfg using AUTHENTICATION KERBEROS
```

Note: It is possible to catalog the databases on the client machine and explicitly specify the Kerberos authentication type with the server's target principal name. In this way, the first phase of the connection can be bypassed.

Kerberos / server encrypt authentication

The final type of server-side authentication supported by DB2 is a combination of both the Kerberos and the server encrypt authentication by setting the DB2 instance (dbm) configuration parameter AUTHENTICATION to KRB_SERVER_ENCRYPT.

This hybrid provides support to environments where all of the servers support the Kerberos security protocol, but not all of the client operating systems do — for example, a Windows 2000 mixed-mode domain that services Windows NT workstations.

Here's how authentication with KRB_SERVER_ENCRYPT works:

- If the client authentication is KERBEROS, the client is authenticated using the Kerberos security system.
- If the client authentication is not KERBEROS, or the Kerberos authentication service is not available, then the system authentication type is equivalent to SERVER_ENCRYPT.

The authentication type for both DB2 clients and servers can be modified with the DB2 Control Center, if installed, or the DB2 command:

```
db2 update dbm cfg using AUTHENTICATION KRB_SERVER_ENCRYPT
```

Client authentication

In addition to supporting authentication of user accounts at the server, DB2 also supports client-side authentication. Prior to the implementation of the Kerberos security protocol, client-side authentication was the only authentication that allowed for single sign-on. Client-side authentication is sometimes called single

sign-on security because once the user has been authenticated at the client workstation they no longer are required to present a user account and password to access network resources.

As you can see this presents a problem if we start trusting all clients on the network, as some clients supported by DB2 do not have reliable security systems built into the operating system. Clients such as Windows 9x are good examples of untrustworthy clients.

Trusted clients

To protect ourselves from these so-called untrustworthy clients, there are a couple of other DB2 instance (DBM) configuration parameters that can influence where authentication takes place. The first one is called TRUST_ALLCLNTS. This DB2 instance (dbm) configuration parameter indicates whether or not we will be trusting all clients when AUTHENTICATION is set to CLIENT.

The default value for this parameter is YES, which means that when the value for AUTHENTICATION is changed from SERVER to CLIENT, then DB2 server will trust all DB2 clients to authenticate user accounts. It is probably a good idea to trust only the trustworthy clients. If this is the case, then you should consider changing TRUST_ALLCLNTS to NO.

There is another DB2 instance (dbm) configuration parameter that influences where authentication take place. This parameter is called TRUST_CLNAUTH and the default value for this parameter is CLIENT.

This parameter indicates where authentication will take place in the event that AUTHENTICATION is set to CLIENT and TRUST_ALLCLNTS is set to YES, and a DB2 client attempts to attach or connect to a DB2 server and passes a userid and password as part of the connection statement. If no userid and password is contained within the attach or connect statement, then this parameter is ignored — otherwise, authentication, even for trusted clients, is performed based on the value of this parameter, either CLIENT (default) or SERVER.

Limiting domain scope

The DB2 Domain List (db2domainlist) registry variable can be used to limit the domain scope for client authentication. This DB2 registry variable specifies a list of one or more Windows domain names that the user must belong to in order to allow the user to attach or connect. DB2 will still attempt to authenticate the account in the local Security Account Manager (SAM) database, domain SAM database, or domain SAM databases of trusted domains, but in order for the user to establish an instance attachment or database connection, they must have been authenticated in one of the domains listed in this DB2 registry variable. Only users belonging to these domains will have their connection or attachment requests accepted:

```
db2set DB2DOMAINLIST=domain1,domain2,domainN
```

Note: This registry variable should only be used under a pure Windows NT domain environment with DB2 servers and clients running DB2 Universal Database Version 7.1 or later.

If this registry value is defined and the user's account does not belong to a domain specified in the variable, the user's attach or connect statement will return with an SQLCODE 1068N.

4.2.3 DB2 group enumeration

DB2, like Windows, simplifies the management of authorities and privileges by supporting groups. This is accomplished by first creating a group at the operating system level, adding users to the group, and then finally assigning the group to one or more DB2 authorities and/or privileges.

Individual user accounts may belong to one or more groups. In order to determine exactly the privileges of an individual user we must enumerate all of the groups. Group enumeration is the process of collecting all of the group names for an individual user account. User accounts can be explicit members of groups or they can be implicit members of groups by being in a group within a group. These groups can be local, domain, and/or global groups.

During the authentication of a user account, DB2 will first attempt to authenticate the account in the local SAM database. If the account is not found, then DB2 will attempt to authenticate the account in the domain SAM database. If the account is not found in the domain SAM database, DB2 will attempt to authenticate the account at domain SAM databases of trusted domains.

The process of group enumeration will by default occur in the SAM database where the user account is authenticated. Under most circumstances this is a good thing. However in some environments, it may be desirable to force the location of the enumeration to some other location, for example the local machine. This can be accomplished by setting the DB2 registry variable called db2_grp_lookup to either local or domain.

Group lookup

The DB2 Group Lookup registry variable can be used to force DB2 to perform group enumeration in on of three places. The default value for DB2 Group Lookup is null which indicates that the user's account should be enumerated where it is authenticated. As previously mentioned DB2 will first look in the local SAM, followed by the domain SAM, followed by any trusted domain SAM databases.

In the Windows 2000 domain model all domains within the domain tree establish implicit trusts between each other. In this model it is likely that a user that needs to access a database in a child domain is defined at the parent domain. However, the individual responsible for managing a particular group of database users can only do so in the child domain. In this case, you would want DB2 to perform group enumeration in the child domain.

If a user account exist in another trusted domain and you want DB2 to perform the actual group enumeration based on the groups the user is a member of in the current domain you will need to set this DB2 registry variable to domain.

To accomplish this, set the DB2 registry variable:

```
db2set db2_grp_lookup=domain
```

On the other hand, the user account may be defined in the same domain as the DB2 server, but you want DB2 to perform group enumeration based on the groups the user is a member of in the local SAM database. This might be the case if your DB2 System Administrator is also responsible for managing the groups of database users that access.

If a user account exists in the current domain and you want DB2 to perform the actual group enumeration based on the groups the user is a member of in the current local SAM database you will need to set this DB2 registry variable to local.

To accomplish this, set the DB2 registry variable:

```
db2set db2_grp_lookup=local
```

4.3 Instance level security

In this section we cover instance level security. DB2 restricts access to instance level resources by using groups that have been defined at the operating system. Each group is referred to by a name and defines a logical group of users. Assigning users with similar security needs to groups provides a more flexible method for controlling instance security.

DB2 currently restrict group names to eight characters. User and group names longer than eight characters can be defined at the operating system, but these must be added to local groups that follow the DB2 eight character naming standard. Additionally, on Windows, local group names, global group names and userids cannot have the same name.

In DB2, there are two basic categories of users that perform administration task. These are the DB2 system administrators and the DB2 database administrators.

The DB2 system administrators groups are groups of users that are involved in the administration, control, and maintenance of the DB2 system, instance, and databases. In this section we discuss the different types of DB2 system administration groups that are available for DB2.

The DB2 system groups that are controlled at the instance level are:

- DAS Administrators groups
- DB2 System Administrators groups
- DB2 System Controllers groups
- DB2 System Maintenance groups

What's new: DB2 UDB V8.1 introduces a new authority level designed to simplify administration of the Database Administration Server (DAS), called DASADM.

4.3.1 Default instance security

It is important to note that by default, any member of the Windows Administrators group has all of the authorities afforded to the DB2 System Administrator. In small shops this might be acceptable, as the DB2 System Administrator is probably the Windows System Administrator as well. In larger enterprises, where the roles of IT professionals are more specialized, it is common to see the roles of system administrators, DB2 administrators, DB2 operators, and database administrators segregated.

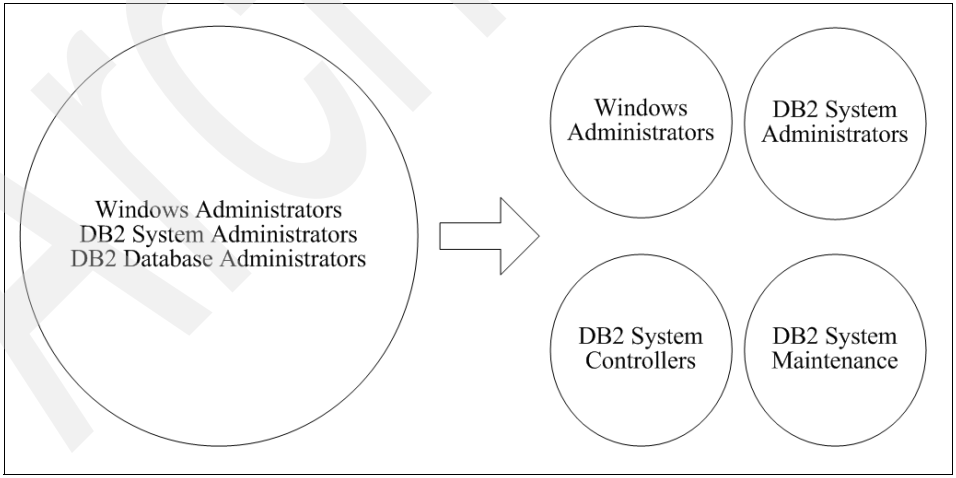

Figure 4-6 Security goals

4.3.2 DAS Administrator Authority (DASADM)

The Database Administration Server (DAS) Administrator Authority (DASADM) defines a group of users that perform DAS Administration (DASADM) tasks. This group has the highest level of authority within the DAS.

The DB2 Administration Server configuration parameter DASADM_GROUP can be updated using the Command Line Processor:

```
db2 update admin cfg using dasadm_group db2dasa
```

Prior to DB2 UDB V8.1, the DAS Administration authority did not exist. Rather, the DAS instance authority was controlled by the SYSADM, SYSCTRL, and SYSMAINT authorities that exist in database instances.

Note: The value for DASADM_GROUP can only be changed by a DB2 UDB V8.1 Command Line Processors interface.

4.3.3 DB2 System Administrators Authority (SYSADM)

The DB2 System Administration Authority (SYSADM) defines a group of users that perform DB2 system administration tasks. This group has the highest level of authority within the database instance.

Only a user with SYSADM authority can perform the following functions:

- Migrate a database
- Update the database manager configuration file
- Grant DBADM authority

The SYSADM_GROUP instance (dbm) parameter defines the group name with DB2 system administration (SYSADM) authority for the database instance. This parameter should be set to DB2SADMN after the DB2SADMN group has been created and populated with the appropriate users. The SYSADM_GROUP instance (dbm) parameter can be updated using either the Control Center or the Command Line Processor:

```
db2 update dbm cfg using sysadm_group db2sysa
```

Note: In order to start and stop a service on Windows, you must be a member of one of the following groups: Domain Administrators, Administrators, or Power Users.

4.3.4 DB2 System Control Authority (SYSCTRL)

The System Control Authority (SYSCTRL) defines a group of users that perform DB2 system control tasks. This group has privileges allowing operations affecting system resources, but not allowing direct access to data.

Only a user with SYSCTRL authority or higher can do the following:

- Update a database, node, or distributed connection services (DCS) directory
- Force users off the system
- Create or drop a database
- Drop, create, or alter a table space
- Restore to a new database.

The SYSCTRL_GROUP instance (dbm) parameter defines the group name with DB2 system control (SYSCTRL) authority for the database instance. This parameter should be set to DB2SCTRL after the DB2SCTRL group has been created and populated with the appropriate users.

The SYSCTRL_GROUP instance (dbm) parameter can be updated using either the Control Center or the Command Line Processor:

```
db2 update dbm cfg using sysadm_group db2sysc
```

4.3.5 DB2 System Maintenance Authority (SYSMAINT)

The System Maintenance Authority (SYSMAINT) defines a group of users that perform DB2 system maintenance tasks. This group has privileges to perform maintenance operations on all databases associated with an instance without having direct access to data.

Only a user with SYSMAINT or higher system authority can do the following:

- Update database configuration files
- Back up a database or table space
- Restore to an existing database
- Perform roll forward recovery
- Start or stop an instance
- Restore a table space
- Run trace
- Take database system monitor snapshots of a database manager instance or its databases

The SYSMAINT_GROUP instance (dbm) parameter defines the group name with DB2 system maintenance (SYSMAINT) authority for the database instance. This parameter should be set to DB2SMAIN after the DB2SMAIN group has been created and populated with the appropriate users.

The SYSMAINT_GROUP instance (dbm) parameter can be updated using either the Control Center or the Command Line Processor:

```
db2 update dbm cfg using sysadm_group db2sysm
```

4.3.6 DB2 directory security

One area that is often overlooked about the DB2 instance is the security of the DB2 node (instance) and database directories. In this section we look at ways to tighten and loosen security around the DB2 node and database directories.

Managing directory security

In DB2 UDB a user must be a member of the local administrators group, or if defined, must belong to either the SYSCTRL or SYSADM authority groups to catalog DB2 nodes or databases. If you plan on securing the DB2 node and database directories, you will need to add the appropriate user accounts to one or more of these domain groups.

In some cases it may be desired to remove these default security restrictions from the DB2 node and database directory. For example, you may consider doing this on workstations used in your development environments so that developers can add, update, or remove DB2 databases themselves.

If you want to allow users to catalog databases without such privileges, as in the case of power users or developers, you will need to change the database instance (dbm) configuration parameter called *catalog_noauth*. The default value for this parameter is NO, meaning that cataloging without authority is prohibited. Changing the value to YES will allow users to catalog DB2 nodes and databases without needing any specific authority to do so:

```
db2 update dbm cfg using catalog_noauth yes
```

Note that even if we remove the authority requirement to catalog a database, access to the database is still governed by authorities and privileges at the database server. Simply cataloging a database does nothing more than update the client's DB2 node and database directory.

DB2 Discovery feature

The DB2 Discovery feature is used by the DB2 Configuration Assistant and the DB2 Control Center to facilitate the cataloging of DB2 nodes and databases. In this section we discuss the instance (dbm) parameters that can be configured in order to change how the discovery process works.

Client Discovery

The primary instance (dbm) parameter that influences the discovery process is appropriately called *discover*. The discover instance parameter has three possible values. These are SEARCH, KNOWN, and DISABLE.

The default value for discover is SEARCH. The value of SEARCH for the discover configuration parameter allows the client to search the network for both unknown and known DB2 servers.

Unknown DB2 servers are servers in which the client does not know the name of the server. The client will search the network for a period of time defined by the db2discoverytime registry variable for any DB2 server that is on the network. A value of NULL for the DB2 registry variable db2discoverytime allows the client to search the network for 40 seconds.

If the discover instance (dbm) configuration parameter is set to KNOWN, then the client can only search the network for known DB2 servers. In this case the client is required, at the very least, to know the name of the system to search for existing instances and databases.

If the discover configuration parameter is set to DISABLE, the client will be unable to search the network for either unknown or known DB2 servers. Setting DB2 Discovery to DISABLE provides for the most security because clients are unable to browse the network in search of DB2 nodes and databases. However, doing this can create additional work for DBA's that might now have to manually catalog DB2 nodes and databases that users may have been able to find on their own with the help of DB2 Discovery. To address this issue, DB2 provides some additional configuration parameters.

Server Discovery

The Database Administration Server (DAS) also supports the discover configuration parameter. If you want to hide every instance on a DB2 server from being discovered, you can change the value of the discover parameter in the DAS instance to either DISABLE or KNOWN.

The default value for discover is SEARCH which allows the DB2 DAS to reply to both SEARCH and KNOWN discovery requests from DB2 clients. Setting the value of discover to KNOWN allows the DB2 DAS to reply to only KNOWN discovery request from DB2 clients. Finally, setting the value to DISABLE prevents the DB2 DAS from responding to any DB2 Discovery requests.

```
db2 update admin cfg using discover disable
```

Instance Discovery

In the event that you want to hide one or more instances on a DB2 server you can update the database instance (dbm) configuration parameter discover_inst. The default value for discover instance (discover_inst) is enabled, which allows the instance to be discovered by DB2 clients as long as the DB2 DAS for the server allows discovery.

```
db2 update dbm cfg using discover_inst disable
```

Database Discovery

You may also find yourself in a situation where you want to allow the DB2 server to be discovered and the DB2 instance to be discovered, but want to prevent one or more databases from being discovered. In this case you can disable DB2 Discovery for just the database.

```
db2 update db cfg for sample using discover_db disable
```

Active Directory Services security

As we discussed in the Active Directory Services section in Chapter 1, “Introduction” on page 1, DB2 UDB can be configured to integrate node (instance) and database directories in the Windows 2000 Active Directory. If you are planning on using the Active Directory to store this information, there are some security considerations you should address.

Once enabled to use the Active Directory, the DB2 UDB node (instance) and database objects are registered under the DB2 UDB server name found in the computer container in the Active Directory. In order to register a DB2 node or database in the Active Directory, you need to have the appropriate Active Directory privileges. These privileges can be obtained by membership in one of the following groups:

- Administrators
- Domain Admins
- Enterprise Admins

It should also be noted that, by default, all Authenticated Users in the domain have Read permissions on the DB2 node (instance) and database objects registered in the Windows Active Directory (see Figure 4-7). You may want to consider changing this default permission to only allow the appropriate user groups to Read this information. The Active Directory Users and Computer Management Console (MMC) is used to manage security of the Active Directory.

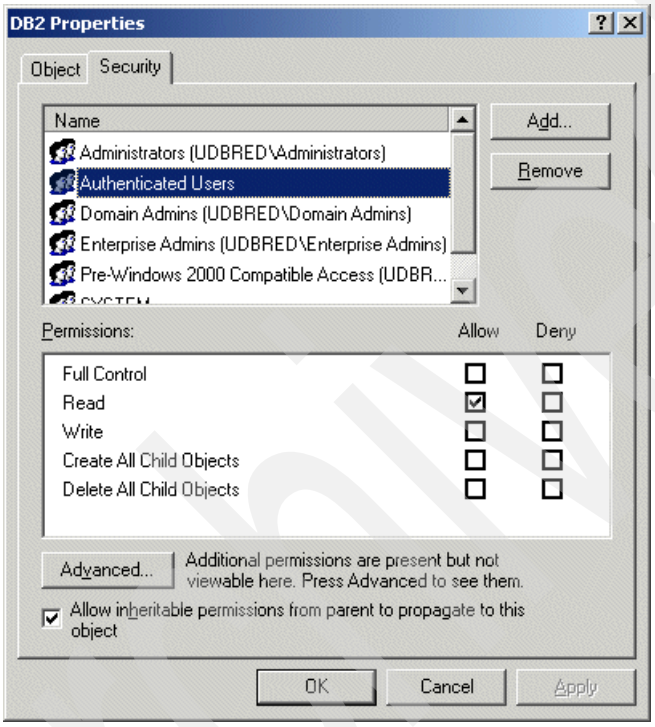

Figure 4-7 Default security for authenticated users

4.4 Database level security

In this section we cover database authorities and privileges. An important distinction between instance and database authorities is that database authority and database privileges are always granted or revoked to a user or group using an SQL statement. This can be performed via the DB2 Control Center or any other GUI that supports connecting to a database and invoking SQL statements.

Instance level security, as we discussed in our previous section, requires only membership in one of the three instance level security groups defined at the operating system and assigned to DB2 instances through instance (dbm) configuration parameters. These authorities cannot be assigned to individual user accounts.

DB2 restricts access to database level resources by granting authority and privileges to individual user accounts or groups that are defined at the operating system level. Each group is referred to by name and defines a logical group of users. Assigning users with similar security needs to groups provides a more flexible method for controlling database security.

The primary difference between a database authority and privilege is scope. Database authorities are granted at the database level. On the other hand, database privileges are granted on objects within the database.

4.4.1 Database authorities

Database authorities are granted and revoked to and from users or groups via the SQL grant or revoke statement. This can be performed via the DB2 Control Center or any other GUI that supports connecting to a database and invoking SQL statements.

Database Administration Authority (DBADM)

The Database Administration Authority (DBADM) is the second highest level of administrative authority. It applies only to a specific database, and allows the user to run certain utilities, issue database commands, and access the data in any table in the database.

Only a user with SYSADM authority can grant or revoke DBADM authority. Users with DBADM authority can grant privileges on the database to others and can revoke any privilege from any user regardless of who granted it. Table 4-1 lists the DB2 database authorities.

Table 4-1 Database authorities

Authority	Description
Create Packages (BINDADD)	The authority to create packages. The creator of a package automatically has the CONTROL privilege on that package and retains this privilege even if the BINDADD authority is subsequently revoked.
Connect (CONNECT)	The authority to access the database.
Create Table (CREATETAB)	The authority to create tables. The creator of a base table automatically has the CONTROL privilege on that table. The creator retains this privilege even if the CREATETAB authority is subsequently revoked.
Create External Routine (CREATE_EXTERNAL_ROUTINE)	The authority to register external routines. Care must be taken that routines so registered will not have adverse side effects.
Create Unfenced Routine (CREATE_NOT_FENCED_ROUTINE)	The authority to register routines that execute in the database manager's process.
Database Administration (DBADM)	The authority to administer the database. This authority implicitly includes all other database authorities.
Create Implicit Schemas (IMPLICIT_SCHEMA)	The authority to implicitly create a schema.
Load (LOAD)	The authority to load in this database. This authority gives a user the right to use the LOAD utility in this database.
Quiesce Connect (QUIESCE_CONNECT)	The authority to access the database while it is quiesced.

Public authorities

There are several DB2 database authorities that are granted by default to public, see Figure 4-8.These are:

- Connect (CONNECT)
- Create Tables (CREATETAB)
- Create Packages (BINDADD)
- Create Schemas Implicitly (IMPLICIT_SCHEMA)

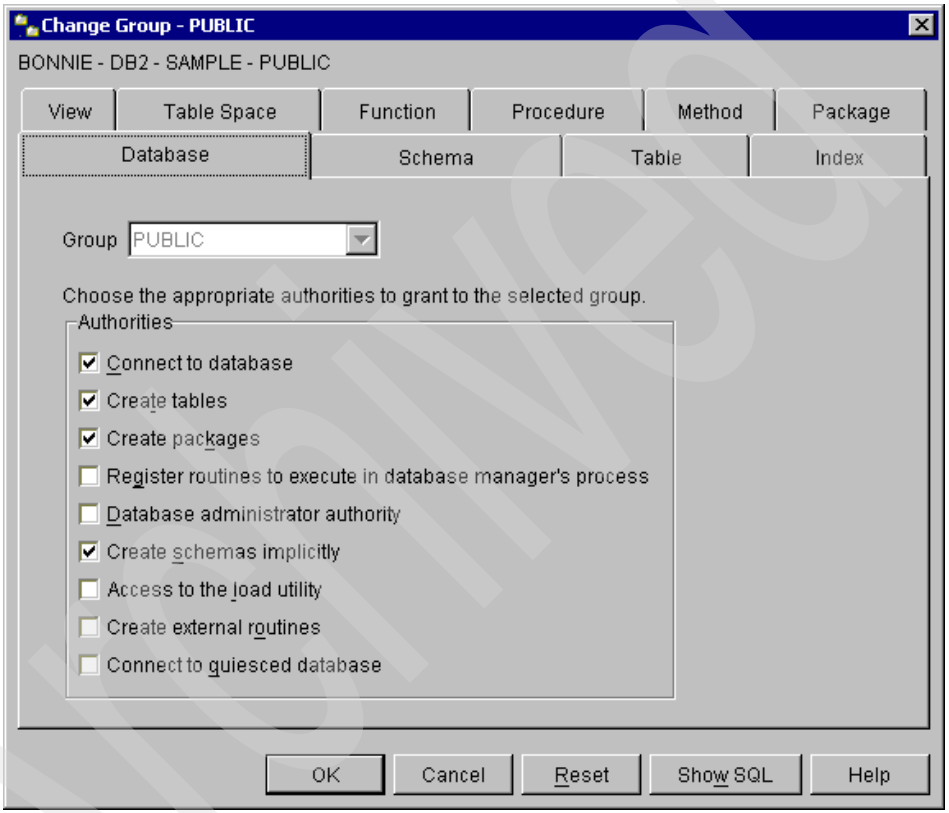

Figure 4-8 Public database authorities

4.4.2 Database privileges

Database privileges, like database authorities, are granted and revoked to and from users or groups via the SQL grant or revoke statement. Table 4-2 lists DB2 database privileges.

Table 4-2 Database privileges

Privilege	Description
Index	The privilege to CONTROL indexes.
Package	The privilege to BIND, CONTROL, EXECUTE packages
Routine	The privilege to EXECUTE routines.
Schema	The privilege to ALTERIN, CREATEIN, DROPIN schemas.
Sequence	The privilege to USAGE sequence.
Server	The privilege to PASSTHRU servers.
Table, View, Nickname	The privilege to ALTER, CONTROL, DELETE, INDEX, INSERT, REFERENCES, SELECT, UPDATE tables.
Tablespace	The privilege to USE table spaces.

Public privileges

Following are DB2 database privileges that are granted by default to public:

- EXECUTE WITH GRANT privilege to all functions in the SYSFUN schema
- EXECUTE privilege on all procedures in SYSIBM schema
- CONNECT, CREATETAB, BINDADD, and IMPLICIT_SCHEMA
- USE privilege on the USERSPACE1 table space
- SELECT privilege on each system catalog
- BIND and EXECUTE privilege to PUBLIC for each successfully bound utility.
- EXECUTE WITH GRANT privilege to PUBLIC on all functions in the SYSFUN schema.
- EXECUTE privilege on all procedures in SYSIBM schema.

To tighten the system or database security, the default public database authorities or privileges should be reviewed after the database is created to ensure that they meet the organization's security requirements.

4.4.3 Data encryption

In this section we discuss how you can provides additional security by encrypting data in your database. DB2 provides built-in user defined functions that can be used to encrypt and decrypt data stored in database tables. These functions allow you to store data in the database that can only be decrypted by users having the encryption password. In other words, nobody, not even users with DB2 System Administrator authority can access the data without the encyption password.

In the previous sections we discussed how DB2 UDB controls database security through authorities and privileges. These authorities and privileges are granted either directly to users or groups and stored in the database system catalog tables. Before DB2 checks these authorities and/or privileges the user account is authenticated by the operating system security. However, providing a user account at the operating system level for every possible user of your database may not be practical. In this situation you may have to store userid and password information within a database. If this is the case you may want to consider encrypting the password.

The DB2 UDB user defined functions for data encryption are:

- ENCRYPT
- DECRYPT_CHAR
- DECRYPT_BIN
- GETHINT

Note: Using encryption functions can result in unrecoverable data. If you choose to use these functions you should plan for both the eventuality of unrecoverable data and management of forgotten passwords.

Encrypting data

Data can be stored in database tables encrypted by using the ENCRYPT user defined function. This function allows you to encrypt data using an encryption password. Once encrypted, you must provide the encryption password in order to decrypt the data. You can optionally supply a password hint that is stored with the encrypted data. The password hint can be retrieved without the encryption password and is designed to help the original encryptor remember a forgotten password.

The syntax for the ENCRYPT function is:

```
ENCRYPT(data, [password],[hint] )
```

The password used for encryption can be supplied as a parameter to the ENCYRPT function call or set using the ENCRYPTION PASSWORD special register via the SQL statement SET ENCRYPTION PASSWORD.

Decrypting data

The DECRYPT_CHAR and DECRYPT_BIN user defined functions must be used to decyrpt data originally encrypted using the ENCRYPT function. The DECRYPT_CHAR function returns a VARCHAR data type. The DECRYPT_BIN function returns a VARCHAR FOR BIT DATA type. Neither of the two functions return a password hint, if one was stored with the encrypted data, instead the GETHINT function must be used.

As with the DECRYPT function, the password used for decryption can be supplied as a parameter to the DECYRPT_CHAR or DECRYPT_BIN functions or set using the ENCRYPTION PASSWORD special register via the SQL statement SET ENCRYPTION PASSWORD.

Forgotten passwords

In order to facilitate what would otherwise result in unrecoverable data, the ENCRYPT function supports the use of a password hint that can be supplied during as an optional parameter to the ENCRYPT function. In the event that the password used for encrypting the data is lost, the GETHINT user defined function can be used to retrieve the password hint. A password hint is a optional 32 byte phrase that may help the original encryptor remember the encryption password. For example, *What is the name of your pet?*.

4.4.4 Auditing database transactions

In many shops highly sensitive data is audited at the transaction level by capturing the how, who, what, when, and where characteristics of each individual database transactions. There are two common techniques that can be implemented to accomplish this level of data auditing.

The first technique involves creating insert and update triggers to capture information stored in the user, timestamp, and server special registers to populate the appropriate columns at the record level in the tables affected by the transactions. The second involves using DB2's built in Data Replication features to create replication subscriptions that populate audit or history tables as a result of the capture and apply processes.

Both of these techniques work well in a two tier client/server environment where the user connects directly to the database with his/her user account. However, they both fall short when used in a three tier architecture where the user connects to the middle tier with his/her user account, but the middle tier connects to the database with a generic account. In this situation still know when and where but we loose track of who.

In the past, addressing this issue meant moving this functionality from the database server (3rd tier) to the application server (2nd tier). It also meant that instead of using functionality already built into the database engine and paid for, you had to develop this functionality yourself. DB2 UDB V8.1 addresses this issue with a suite of four new special registers.

Client special registers

DB2 UDB V8.1 has extended the list of special registers to include four new special registers that can be set by the client application. The new special registers can help you keep track of accounting, application, user, and workstation information. All four of the special registers can be set using the Set Client Information (sqleseti) API. Once set, the values of the special registers can be retrieved using the SQL statement VALUES.

Client accounting

The CLIENT ACCTNG special register contains the value of the accounting string from the client information specified for this connection. The data type of the register is VARCHAR(255). The default value of this register is an empty string.

Client application name

The CLIENT APPLNAME special register contains the value of the application name from the client information specified for this connection. The data type of the register is VARCHAR(255). The default value of this register is an empty string.

Client user identification

The CLIENT USERID special register contains the value of the client userid from the client information specified for this connection. The data type of the register is VARCHAR(255). The default value of this register is an empty string.

Client workstation name

The CLIENT WRKSTNNAME special register contains the value of the workstation name from the client information specified for this connection. The data type of the register is VARCHAR(255). The default value of this register is an empty string.

5

Performance

DB2 UDB environments can range from stand-alone systems to complex combinations of database servers and clients running on multiple platforms. In all, the common key for successful applications is performance. When you plan, design, and build your database system, you need to understand various considerations about logical and physical database design, application design, and configuration parameters of DB2 so that your system can meet the performance requirements of your applications.

In this chapter we discuss tasks involved in tuning the database and Windows environments to obtain optimal performance. Additional details on performance tuning can be found in the manual, *DB2 UDB V8 Administration Guide: Performance*. Here, we cover the major items that experience has shown have the largest impact on performance:

- Performance tuning overview
- Primary Windows performance factors
- Primary DB2 performance factors
- Windows and DB2 system optimization

Note: Before using the information in this chapter, we highly recommend that you use both the Configuration Advisor and Design Advisor wizards discussed in Chapter 3, “Post-installation tasks” on page 95 of this book. These will provide you with a very good starting point in achieving excellent performance without having to get into the details covered here.

5.1 Performance tuning overview

While the performance of your database system may initially be good, as time goes on, it may need to serve more users, store more data, and process more complex queries. Consequently, the increased load level of your database server will affect its performance. This could be the time to upgrade to more powerful equipment. However, before investing in equipment, you may be able to improve performance by simply tuning the database system.

In this section we provide an overview of the performance tuning process and discuss a proven methodology on performance tuning. The following topics are discussed:

- Measuring system performance
- Determining when system tuning will be cost-effective
- Causes of performance problems
- Deciding when to tune the system
- Planning performance tuning

5.1.1 Measuring system performance

Performance is the capacity of your system to produce the desired results with a minimum cost of time or resources. It can be measured through response time, throughput, and availability.

Performance is not an absolute value. The performance of an information system can be rated as better or worse, compared to a reference value. First, the reference value needs to be established according to the requirements of the information system; then the results of tuning efforts can be compared against it. Those requirements, or the service level agreement, may include the throughput of the system, limits on the response time for a percentile of transactions, or any other issues relevant to the end user.

Units of measurement are usually based on the response time of a given workload. Other units of measurement may be based on transactions per second, I/O operations, CPU use, or a combination of the above.

Avoid setting limited performance goals such as “The response time of all transactions must be less than 3 seconds”. This is not a practical goal, because your application may submit a complex query which takes 10 minutes (even though this does not happen everyday). Rather, a more reasonable goal might be “This particular query must be completed in 2 minutes”. Once you have set a measurable performance goal, monitor your system’s actual performance and compare it with that goal. Then consider what you can do to fill in the gap.

5.1.2 Determining when system tuning will be cost-effective

Your database system is a complex data-processing environment that includes hardware resources, software components, and application programs. DB2 starts many processes that perform different functions in your database system, and it allocates the necessary memory areas, thus consuming hardware resources.

As system performance degrades, you may at first be tempted to upgrade your system with more powerful and expensive equipment. However, in the meantime, you may be able to improve the performance of your existing resources by simply tuning your operating system and databases. By carrying out a performance tuning project, you can balance your hardware resources to each part of the database system, including processes and the required memory areas.

Specific goals of tuning could include:

- Processing a larger or more demanding workload without buying new hardware
- Obtaining faster system response times, or higher throughput, without increasing processing costs
- Reducing processing costs without negatively affecting service to your users, and spending the money for other resources

Other benefits are intangible; for example, greater user satisfaction and productivity resulting from faster response times. If you manage an Internet business, higher performance — including quick response time and high availability — may prevent lost business opportunities. When weighing the cost of performance tuning against its possible benefits, all of these benefits need to be considered.

5.1.3 Causes of performance problems

There are many possible causes of performance problems. In the following sections we describe those most frequently encountered.

Poor application design

When you experience performance problems, in many cases these stem from poor application design and inefficient programs. The database itself may not have any problems at all. For example, SQL statements written inappropriately can degrade overall performance, even though your database is well designed and tuned.

You can also refer to *DB2 Universal Database V8 Administration Guide: Performance*, SC09-4821-00, Part 2, “Tuning application performance”, which describes tips you should take into consideration when designing and developing applications.

Poor system and database design

Poor design of your system and/or databases can be also a reason for performance problems. Inefficient disk storage layout, or failing to consider performance when designing and implementing your database, will certainly degrade performance, and can be very difficult to fix once your production system has been started.

See the manual, *IBM DB2 Universal Database Version 8 Administration Guide: Planning*, SC09-4822-00, which describes factors to consider when designing disk storage layout and databases.

System resource shortages

System resource shortages can cause bottlenecks in your database system:

- **Memory:** Every process will use some physical memory. If you have insufficient memory, you may find that applications will fail, or your system will start swapping.
- **Processor:** Too many users, or running applications on a CPU, may cause system degradation.
- **Storage:** I/O performance can play an important part in a database system. Too much activity on a single disk or I/O bus may cause performance problems.
- **Network:** Unless you are in a stand-alone environment, the network plays an important role. If the network is too slow, then this may appear to the client as a performance problem at the database server.

5.1.4 Deciding when to tune the system

As mentioned before, if the reason for poor performance exists in the system or database design, this will not be as easy to fix as simply tuning performance parameters of the operating system or the database manager. Therefore, you should be aware that performance tuning is not an add-on activity to be done in the production system when there is a problem. Performance needs to be a design goal taken into consideration in each step of the project life cycle:

- **Planning:** Choose the right hardware, software, and network for your system. Take care of future growth.
- **Design:** Choose an appropriate data model and programming interfaces.

- **Development:** Take performance into consideration when developing application programs. Choose the right locking strategy.
- **Testing/acceptance:** Use the same volume of data and configuration as the production system.
- **Production:** Perform proactive performance tuning.

In this chapter, we discuss not only tuning the Windows operating system and the database manager/database configuration parameters, but also various considerations associated with disk storage layout, database design, and application design.

5.1.5 Planning performance tuning

When you carry out a performance tuning project, the worst possible approach is to change the value of many performance tuning parameters without having any idea of what is causing the performance problem. Barring miracles, performance will only get worse, and you will never know which parameter was the cause. So, even if you are anxious for results, you will benefit from following a more methodical approach:

1. Find which queries are slow.
2. Measure the current performance and set the performance goal.
3. Monitor your system and identify where the bottleneck is.
4. Decide where you can afford to make trade-offs, and which resources can bear an additional load.
5. Change only one performance parameter to relieve the bottleneck.
6. Execute the queries again to monitor your system, and check if the performance meets your goal.
7. If the performance does not meet the goal, go back to step 3.

Locate problems and establish goals

When a user reports that response time is too slow, you first need to identify what the problem is. You may need to check:

- Whether this user is the only one who has experienced this problem, or whether others are also complaining about the same problem.
- Whether the poor performance was experienced only in a particular application or queries.
- Whether the problem is really one of response time. Sometimes, another problem (such as the application waiting for a response from the user) might be the cause of poor response time.

Also, you should check:

- Whether the user just noticed the poor performance today, or has been seeing it for a long time
- How slow it is — ten percent slower, or ten times slower usual — for example
- Whether someone had made significant changes on the system just before the performance problem was noticed

To carry out a performance tuning project, it is important to establish the objective. Find which queries are slow, and set a measurable performance goal such as “The response time of this particular query should be less than 30 seconds”.

Identify the cause

Once you have focused on particular queries, execute those queries and monitor the system by using the monitoring tools which the operating system and DB2 provide, and identify where the bottleneck is. The manual IBM DB2 Universal Database Version 8 System Monitor Guide and Reference publication number SC09-4847-00 is an excellent reference on the tools available for system monitoring. We also discuss monitoring in Chapter 6, “Monitoring and management” on page 295.

If possible, you should execute the queries in the system where the problem has been observed. If you have to execute them in another environment, it is important to replicate the original environment as carefully as possible. For example, if the user is running in client-server mode, then you should set up a client-server environment to test the queries.

This step is important, because if you tune resources that are not the primary cause of performance problems, your efforts will have little or no effect on response time until you have relieved the major constraints. Also, the changes you introduce can make subsequent tuning work more difficult.

Change one performance parameter at a time

Even if you find more than one possible cause and you are sure that tuning all of them will be beneficial, you should still change only one performance tuning parameter at a time. It will be very difficult to evaluate how much benefit each change has contributed if you change more than one parameter.

The appropriate values for parameters affecting performance can best be determined by performing tests and modifying the values of the parameters until the point of diminishing returns for performance is found. If performance versus parameter values were to be graphed, the point where the curve begins to plateau or decline would indicate the point at which additional allocation provides no additional value to the application, and is therefore simply wasting memory.

DB2 provides configuration parameters to balance your hardware resources to each part of the database system, including processes and required memory areas. These configuration parameters can be divided into two groups — the database manager configuration parameters and the database configuration parameters, depending on whether their settings affect instance level, or database level.

To update the database manager configuration parameters, you can use the Control Center GUI tools. Or, you can use the following command:

```
UPDATE DBM CFG USING parameter_name value
```

To update the database configuration parameters, you can use the Control Center GUI tools, or use the following command:

```
UPDATE DB CFG FOR dbname USING parameter_name value
```

Note: Once you have created a new database, we recommend that you use the Configure Advisor Wizard to obtain recommended values for database manager and database configuration parameters and set them as initial values rather than using default values. The Configuration Advisor Wizard will suggest suitable values for those parameters based on factors such as workload and server configuration (for example, the number of CPUs). We discuss this wizard in “Configuration advisor” on page 107 in .Chapter 3, “Post-installation tasks”

DB2 also provides the registry variables which control the behavior of the database manager. The registry variables can affect both instance level and machine level. We introduce some of the available registry variables which affect the system performance in this book.

To update the registry variables, you can use the following command:

```
db2set variable=value
```

You need to restart the database manager when you change the registry variables.

Before making any changes to performance parameters, be prepared to back out those changes if they do not have the desired effect or have a negative effect on the system. For example, the db2look utility with -f option extracts the current values of the configuration parameters and the DB2 registry variables. The output of the utility is a script file which you can execute to go back to the setting at the time when the utility was executed.

The db2look utility also extracts the required DDL statements to reproduce the database objects of a production database on a test database. This utility is very useful when testing against a production database is not possible. See the *DB2 UDB Command Reference*, for more information.

Note: The db2look utility with -f option only extracts configuration parameters and registry variables that affect the DB2 query optimizer. Not all registry variables might be extracted.

Apart from changing performance parameters, creating indexes may improve query performance. Defining appropriate indexes may reduce disk I/O and data sorts significantly. You can use the Explain tool to see whether indexes can be used for particular queries.

As we have stated, the major causes of performance problems are actually poor application design or poor database design, rather than the database configuration itself. Before tuning performance parameters, check to make sure that your application or database design is not a cause.

5.2 Primary Windows performance factors

We begin our discussion on system performance by taking into consideration the primary Windows performance factors. As with any system, performance begins by laying a good foundation of well balanced hardware resources that can be exploited by the operating system and eventually application specific software such as DB2 UDB.

In this section we focus on two primary Windows performance factors system hardware and system software. Specifically, we cover:

- System hardware:
  - Memory
  - Processor
  - Storage
  - Network
- System software:
  - Windows 2000 Servers
  - Windows Server 2003

5.2.1 System hardware

The performance goals of system hardware selection and configuration is to produce a well balanced system that can sustain high rates of overall system utilization. To achieve a well balanced system, we must adequately size individual hardware resources so that no single resource results in a system bottleneck. In this section we look at primary windows performance factors from a system hardware perspective.

You should take into consideration how you plan to scale your system, when planning for additional capacity requirements. Scaling out is achieved by adding more resources, such as memory, processors, and storage across multiple systems. DB2 UDB V8.1 ESE implements a shared-nothing architecture that partitions large databases into smaller more manageable “parts” called partitions that are managed by partitioned database servers. Scaling up is achieved by adding more resources, such as memory, processors, and storage to a single system.

In general, scaling-out is much easier in terms of hardware planning as you scale-out by simply adding additional hardware resources (processors, memory, etc.) by adding additional servers. A scale-up approach typically requires a little more planning as you may reach hardware system limitations in terms of the amount of resources that can be added to the server.

Memory

Since accessing data in memory is faster than accessing data on hard drives, the primary factor in terms of memory is quantity. Although memory speeds are also factors it is seldom a option when configuration a system, unless you are willing to select and configure another system altogether. The amount of physical memory is a critical system hardware resource that can have a huge impact on overall performance. In general the cost of memory on commodity servers is usually an insignificant factor when compared to the cost of other hardware resources.

Today most systems based on the 32-bit Intel Architecture (IA-32) support the Physical Address Extensions (PAE) capabilities of the IA-32. Physical Address Extensions provides operating system software with an instruction set to address physical memory above 4 GB. Operating systems that take advantage of the PAE can address up to 64 GB physical memory.

Given the memory extensions of the IA-32, the primary factor in memory selection will most likely not be the cost of the memory itself, but rather the incremental cost moving from one edition of the Windows operating system to the next in order to address more physical memory.

Processor

Most systems are limited by the total number of central processing units (CPU) they can support. Typically a 4-way cannot be upgraded to an 8-way unless it is indeed a true 8-way that was populated with only 4 processors. There are a few systems on the market today, such as the IBM x440 and the Unisys ES7000, that can be expanded beyond the total number of original processors by adding additional processor expansion modules.

Besides quantity and speed another important consideration in terms of processor selection is the size of the internal L2 cache. Slower processors with larger internal caches have shown significant throughput advantages for database applications over faster processors with smaller internal caches.

Another factor to consider when selecting the number of processors is the operating system software costs. There are incremental licensing costs associated with each Windows 2000 or Windows Server 2003 edition to support more processors.

Storage

The disk subsystem has been a area of much debate over the last several years. Most disk subsystems will implement some form of redundancy that has always favored recover ability over performance. In recent years improvements in technology has been able to overcome many of the performance limitations imposed by implementing redundant disk arrays.

Performance characteristics of disk controllers include speed, throughput, channels, and cache. Care should be taken in the placement of disk controllers in the system. Although most disk adapters are backwards compatible, it should go without saying that you want to match the disk controllers speed with that of the systems PCI bus. You should avoid placing faster 64-bit 66 Mhz disk controllers in slower 32-bit 33 Mhz PCI slots. You should also consider the number of disk controllers in your system as well. Attaching a single disk controller with several I/O channels might be capable of driving your subsystem, but can quickly saturate a single PCI bus, not to mention introduce a single point of failure into your system. If possible you should also avoid placing disk controllers on PCI buses populated with other I/O intensive resources.

Performance characteristics of disk subsystems include disk speed, size, cache, and the number of physical disks in the subsystem. You should favor a subsystem with a large number of small drives over a small number of large drives. If this is impractical, plan for growth by choosing a large number of large drives. Best performance will be achieved for database applications with a large number of physical disks (5-10) per processor. For example, a large 32-way SMP server should be attached to a disk subsystem with a minimum of 160 physical disks no including parity disks or hot spares. With an average18-GB disk you

would have almost 3 TB of total storage. At first this might seem impractical for a 1TB database, but consider space requirements for load files and storing the most recent backup image before copying to tape.

Hardware implementations of disk arrays are now common place on Intel based servers. Modern disk controllers support RAID levels 0,1,5, and 10 sometimes referred to as 0+1. As with most performance decisions there is always a give (cost) and take (performance) associated with choosing which RAID level to implement.

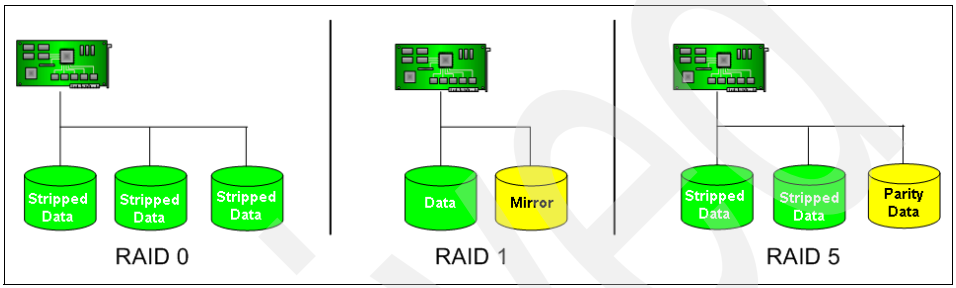

Figure 5-1 RAID 0, 1, 5

RAID Level 0

RAID level 0 provides no redundancy, put does provide better I/O performance by striping data across multiple physical disks. RAID level 0 appears to the operating system as single physical disk, however it is in reality a virtual drive composed of several smaller physical drives much the same way system memory appears as one contiguous memory space even though it is physically composed of several smaller memory units. In general, you should avoid using RAID level 0, as it provides no redundancy. A single hard drive failure to a RAID level 0 array will result in a complete failure of the array.

RAID Level 1

RAID level 1 provides redundancy by mirroring data across pairs of disks at the expense of one physical disk per mirrored pair and two write operations per I/O write request. Although typically only one mirrored disk is used, additional mirrored disks can be used to increase redundancy. RAID level 1 performs better at reading data than writing because it can choose from one of the two or more disk drives to service the read request based on disk utilization. RAID level 1 typically performs better at write request than RAID level 5 arrays.

Disk mirroring provides good performance for operating system and application files as I/O requests are minimal. It is also a good choice for operations that perform sequential I/O requests such as transaction log processing. Due to the mirroring of write operations you should consider using multiple RAID adapters with RAID level 1 to offset the impact of mirroring the drives.

RAID Level 5

RAID level 5 provides redundancy by striping data and parity information across multiple drives at the expense of one physical disk per array and four I/O (2 reads / 2 writes) operations per I/O write request. As the number of physical disks in the array increases the cost penalty for the array decreases but the write penalty remains the same. For example, in a RAID 5 array composed of 3 physical drives the storage reserved for parity is a penalty is 1 physical drive or 1/3 the total space. In a RAID 5 array composed of 6 physical drives the penalty is 1 physical drive or 1/6 the total space.

The process of striping data with parity information impacts the performance of writes requests as each write operation requires four I/O operations. The impact of this write penalty can be lessened by using a RAID controller with a large onboard memory cache. On the other hand, RAID level 5 provides better read performance over level 0 as read requests can be serviced by multiple drive spindles making it a better choice for read intensive applications.

RAID Level 10

RAID level 10, sometimes called RAID 0 + 1 because it is a hybrid of levels 0 and 1, provides redundancy by striping data across mirrored drives at the expense of one physical disk per striped mirrored disk. RAID level 10 provides the best overall performance and redundancy for read/write applications, although can be cost prohibitive. If RAID level 10 is cost prohibitive, consider using a combination of RAID level 1 and RAID level 5.

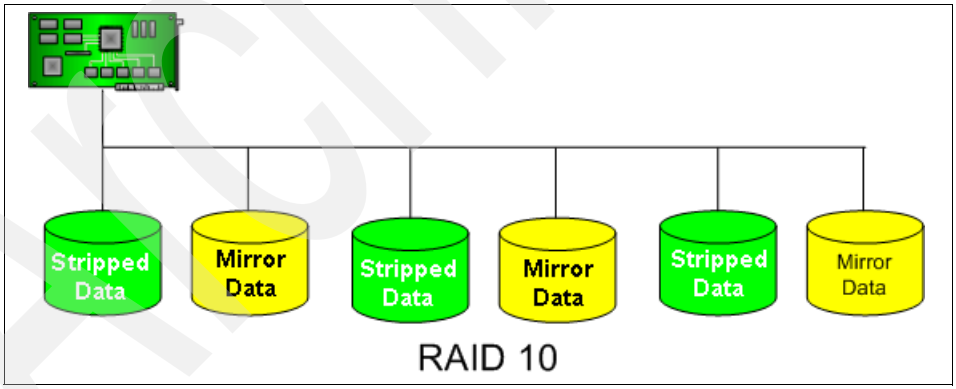

Figure 5-2 RAID 10

Network

Performance characteristics of network adapters include speed and throughput. As with disk controller, care should be taken in the placement of network adapters in the system. You should also consider the number of network

adapters in your system. If possible you should also avoid placing network adapters on PCI buses populated with disk controllers.

The speed of the network adapter can limit the total network throughput. You should consider using faster 64-bit 66 Mhz network adapters, especially when running on gigabit networks as slower 32-bit 33 Mhz network adapters are not capable of driving gigabit networks.

Most network adapters today support teaming. Teaming network adapters provides several benefits. First, it provides network redundancy by preventing a single network adapter that can be a single point of failure in your system. Second, it provides better performance as you can balance network traffic over two or more adapters and PCI buses.

5.2.2 Operating system software

In this section we look at primary windows performance factors from a system software perspective. We explore the Windows 2000 and Windows Server 2003 operating system offerings.

Windows 2000 servers

The Windows 2000 family of servers consists of Windows 2000 Server, Windows 2000 Advanced Server, and Windows 2000 Datacenter Server. All Window 2000 server products build upon the already reach features of Windows NT 4.0 Server and Windows NT 4.0 Server Enterprise Edition.

Table 5-1 Windows 2000 performance features

Feature	Server	Advanced Server	Datacenter Server
CPU	4	8	32
Memory (GB)	4	8	64

Windows 2000 Server Edition

Windows 2000 Server is the entry level edition of the operating system. It provides the same basic support functionality of its predecessor Windows NT 4.0 Server.

The Windows 2000 Server edition is licensed to support 32-bit Intel based server from one (1) to four (4) processors and can address up to 4 GB of physical memory. Although limited to 4 GB of addressable physical memory, Windows 2000 server as well as all other versions of the operating system support the Physical Address Extensions (PAE) provided by the IA32 architecture.

The Microsoft specific implementation provides a set of APIs called Address Windowing Extensions (AWE). This API allows 32-bit applications to address real physical memory above the 4 GB line using a windowing approach that will be discussed in detail later in this chapter.

Windows 2000 Advanced Server Edition

Windows 2000 Advanced Server provides the same basic server functionality of its predecessor, Windows NT 4.0 Server Enterprise Edition. It includes all of the existing features of the base Windows 2000 Server, is licensed to support 32-bit Intel servers with one (1) to eight (8) processors. It can address up to 8 GB of physical memory using the Physical Address Extensions (PAE) provided by the IA32 architecture.

Windows 2000 Advanced Server also includes 4 GB Tuning support. 4 GB Tuning, introduced by Microsoft in Windows NT 4.0 Enterprise Edition Service Pack 3, allows the operating system to make an additional 1 GB of memory available to applications.

Windows 2000 Advanced Server also include the Microsoft Clustering Service (MSCS). This optionally installed service provides fail over clustering support for up to two nodes in the cluster, as did Windows NT 4.0 Server Enterprise Edition.

Windows 2000 Datacenter Server Edition

Windows 2000 Datacenter Server includes all of the features of Windows 2000 Advanced Server. This edition of the Windows 2000 operating system can be licensed to support 32-bit Intel servers with 1-8, 1-16 or 1-32 processors. It can address up to 64 GB of physical memory using the Physical Address Extension provided by the IA32 architecture. Windows 2000 Datacenter Server extends the MSCS support to include up to 4 nodes in a single cluster.

Windows 2000 Datacenter Server also includes a feature called Winsock Direct. It enables unmodified Windows Sockets (Winsock) applications that use TCP/IP to exploit the performance benefits of system area networks (SANs). Winsock Direct enables efficient high-bandwidth, low-latency messaging that conserves processor time for application use. High-bandwidth and low-latency inter-process communication (IPC) and network system I/O allow more users on the system and provide faster response times and higher transaction rates.

Windows 2000 Server or Windows 2000 Advanced Server cannot be upgraded to Windows 2000 Datacenter Server. This is rather new and unique approach that Microsoft has taken with Windows 2000 Datacenter Server as the operating system cannot be purchased directly from Microsoft but instead must be acquired as a complete solution including hardware, software, and services from a Windows 2000 Datacenter Server OEM provider.

Windows Server 2003

Windows Server 2003 is the follow-on to the Windows 2000 family of operating systems. It is currently in beta and will be available, as its name implies, in the 2003 time frame. There are three editions of Windows Server 2003.

Table 5-2 Windows Server 2003 performance features

Feature	Standard	Enterprise	Datacenter
CPU	2	8	32
Memory	4 (32-bit)	32 (32-bit) 64 (64-bit)	64 (32-bit) 128 (64-bit)

For entry level systems, Windows Server 2003 Standard Edition will provide support for 32-bit Intel servers with up to 2 CPUs and 4 GB memory.

Windows Server 2003 Enterprise Edition will provide support for both 32-bit and 64-bit Intel servers with up to 8 CPUs. The 32-bit version will support 32 GB of memory and the 64-bit version will support up to 64 GB of memory. The Microsoft Cluster Service will be included and provide support for up to 8 nodes in a single cluster.

Windows Server 2003 Datacenter Edition will provide support for both 32-bit and 64-bit Intel servers with up to 32 CPUs. The 32-bit version will support 64 GB of memory and the 64-bit version will support up to 128 GB of memory. The Microsoft Cluster Service will be included and provide support for up to 8 nodes in a single cluster.

What's new with Windows Server 2003: Microsoft has announced that Windows Server 2003 Enterprise and Datacenter editions will provide support for NUMA or Non-Uniform Memory Access. Windows Server 2003 provides NUMA awareness to applications and can manage threads and memory via NUMA application program interfaces, sometimes referred to in the industry as NUMA nuggets.

5.3 Primary DB2 performance factors

In this section, we discuss how memory is used by DB2 UDB, and then introduce configurable parameters of DB2 UDB to optimize the performance of your database. You can see more than one hundred configurable parameters in the manual, *DB2 UDB V8 Administration Guide - Performance*, SC09-4821, although you do not have to tune all of them. You can focus on a few important parameters which highly impact the database performance only. Here we introduce some

parameters, grouped according to which areas of system resource usage these parameters affect.

We discuss the following topics:

- Configuration parameter introduction
- Memory
- Processor
- Storage
- Network
- Other performance factors

5.3.1 Configuration parameter introduction

DB2 was designed with tuning and configuration parameters that fall into two general categories:

- Database manager configuration parameters
- Database configuration parameters

Database manager configuration parameters

Each instance of the database manager has a set of the database manager configuration parameters (also called database manager parameters). These affect the amount of system resources that will be allocated to a single instance of the database manager. Also, they configure the setup of the database manager and the different communications subsystems (such as TCP/IP and APPC) based on environmental considerations. In addition, there are other database manager configuration parameters that serve informative purposes only and cannot be changed. All of these parameters effect the instance level and have global applicability independent of any single database stored under that instance of the database manager.

To view, set, and reset the database manager configuration parameters, you can use the following methods:

- **Control Center:** The Control Center provides the Configure Instance notebook which you can use to view, set, and reset the database manager configuration parameters.
- **Database parameters:** From the Command Line Processor or the Command Center, you can execute these commands:
  - `GET DBM CFG`
  - `UPDATE DBM CFG`
  - `RESET DBM CFG`

Database configuration parameters

Each database has a set of the database configuration parameters (also called database parameters). These affect the amount of system resources that will be allocated to that database. In addition, there are some database configuration parameters that provide descriptive information only and cannot be changed; others are flags that indicate the status of the database. To view, set, and reset the database configuration parameters, you can use the following methods:

- **Control Center:** The Control Center provides the Configure Database notebook which you can use to view, set, and reset the database configuration parameters.
- **Database Parameters:** From the Command Line Processor or the Command Center, you can execute these commands:
  - GET DB CFG FOR database_name
  - UPDATE DB CFG FOR database_name USING parameter_name value
  - RESET DB CFG FOR database_name

5.3.2 Memory

A primary performance tuning task is deciding how to divide the available memory among the areas within the database. You tune this division of memory by setting the key configuration parameters described in this section. Many of the configuration parameters available in DB2 affect memory usage on the system. We discuss some of them which have high impact on the database performance.

Before discussing key configuration parameters, we introduce you to the memory model of DB2, because many of the configuration parameters available in DB2 affect memory usage on the system. You should understand how memory is divided among the different heaps before tuning to balance overall memory usages on the system.

Types of memory used by DB2 UDB

Figure 5-3 shows that the database manager uses different types of memory. In this figure, we assume that intra-partition parallelism is enabled.

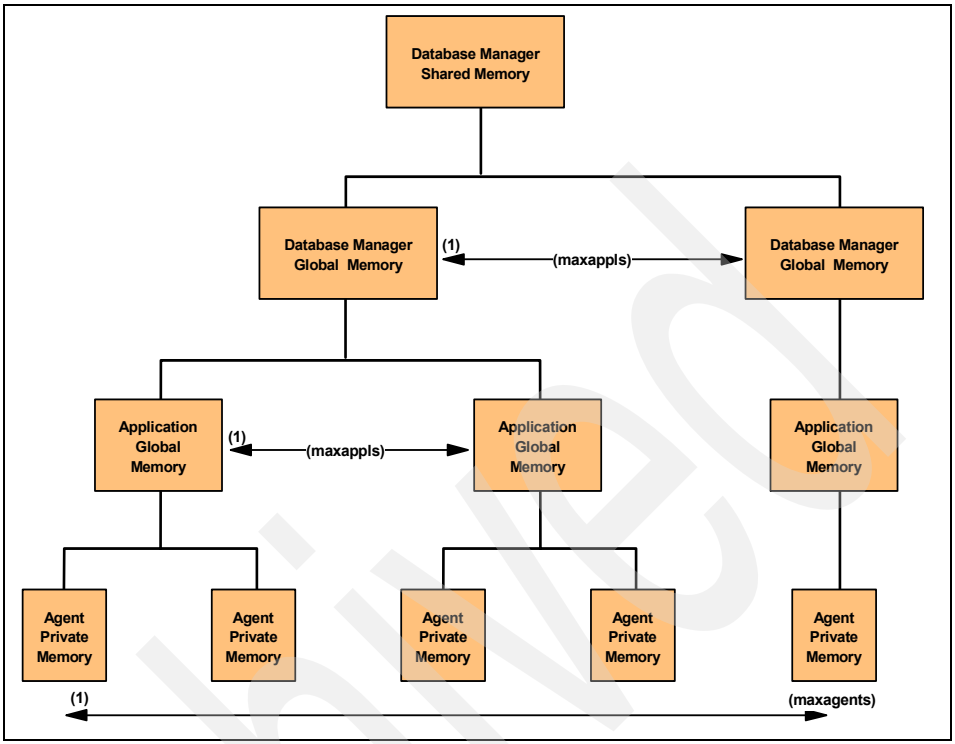

Figure 5-3 Memory segments used by DB2 UDB

The Database Manager Shared Memory is allocated when the database manager is started using the db2start command, and remains allocated until the database manager is stopped using the db2stop. This memory is used to manage activity across all database connections. From the Database Manager Shared Memory, all other memory is attached/allocated.

The Database Global Memory (also called Database Shared Memory) is allocated for each database when the database is activated using the ACTIVATE DATABASE command or the first application connects to the database. The Database Global Memory remains allocated until the database is deactivated using the DEACTIVATE DATABASE command or the last application disconnects from the database. The Database Global Memory contains memory areas such as buffer pools, lock list, database heap, and utility heap.

The database manager configuration parameter NUMDB defines the maximum number of concurrent active databases. If the value of this parameter increases, the number of Database Global Memory segments may grow depending on the number of active databases.

The Application Global Memory is allocated for each connection when the connection is established to a database and remains allocated until the connection is terminated. This memory is used by DB2 agents, including coordinator agents and subagents working on behalf of the application, to share data and coordinate activities among themselves.

The Agent Private Memory is allocated for each DB2 agent when the DB2 agent assigned to work for an application. The Agent Private Memory contains memory areas which will be used only by this specific agent, such as sort heaps and application heaps.

The Agent Private Memory remains allocated even after the DB2 agent completes tasks for the application and gets into idle state. However, if you set the DB2 registry variable DB2MEMDISCLAIM to YES, then DB2 disclaims some or all memory once freed, depending on the value given with the DB2 registry variable DB2MEMMAXFREE which defines the amount of the memory to be retained by each DB2 agent. For more information on these registry variables see the *DB2 UDB V8 Administration Guide: Performance* in Appendix A.

The database configuration parameter MAXAPPLS defines the maximum number of applications that can simultaneously connect to the database. The database manager configuration parameter MAXAGENTS defines the maximum number of DB2 agents including coordinator agents and subagents in the instance. If the value of this parameter increases, the number of the Application Global Memory segments and the Agent Private Memory segments may grow, depending on the number of connected applications and DB2 agents, respectively.

Note: Application Global Memory is allocated if you enable intra-partition parallelism, or if the database manager is in a partitioned database environment using the Database Partitioning Feature of DB2 UDB Enterprise Server Edition, which is beyond our discussion.

How memory is used

Figure 5-4 and Figure 5-5 show how memory is used to support applications. In the previous section we introduced some configuration parameters which may affect the number of memory segments. We now introduce the configuration parameters which allow you to control the size of each memory by limiting their size.

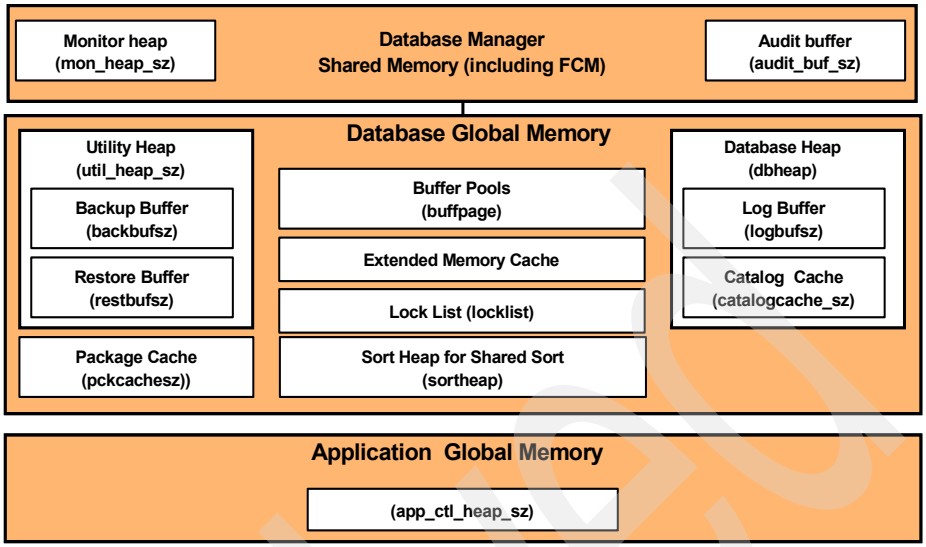

Figure 5-4 Database manager shared memory overview

The Database Manager Shared Memory is required for the database manager to run. The size of this memory is affected by the following configuration parameters:

- Database System Monitor Heap size (MON_HEAP_SZ)
- Audit Buffer Size (AUDIT_BUF_SZ)
- FCM Buffers (FCM_NUM_BUFFERS)
- FCM Message Anchors (FCM_NUM_ANCHORS)
- FCM Connection Entries (FCM_NUM_CONNECT)
- FCM Request Blocks (FCM_NUM_RQB)

The database manager uses the fast communication manager (FCM) component to transfer data between DB2 agents when intra-partition parallelism is enabled. Thus if you do not enable intra-partition parallelism, memory areas required for FCM buffers, message anchors, connection entries and request blocks are not allocated.

The maximum size of the Database Global Memory segment is determined by the following configuration parameters:

- Buffer Pool Size that were explicitly specified when the buffer pools were created or altered (the value of BUFFPAGE database configuration parameter is taken if -1 is specified)
- Maximum Storage for Lock List (LOCKLIST)
- Database Heap (DBHEAP)
- Utility Heap Size (UTIL_HEAP_SZ)

- Extended Storage Memory Segment Size (ESTORE_SEG_SZ)
- Number of Extended Storage Memory Segments (NUM_ESTORE_SEGS)
- Package Cache Size (PCKCACHESZ)

Application Global Memory is determined by the following configuration parameter:

- Application Control Heap Size (APP_CTL_HEAP_SZ)

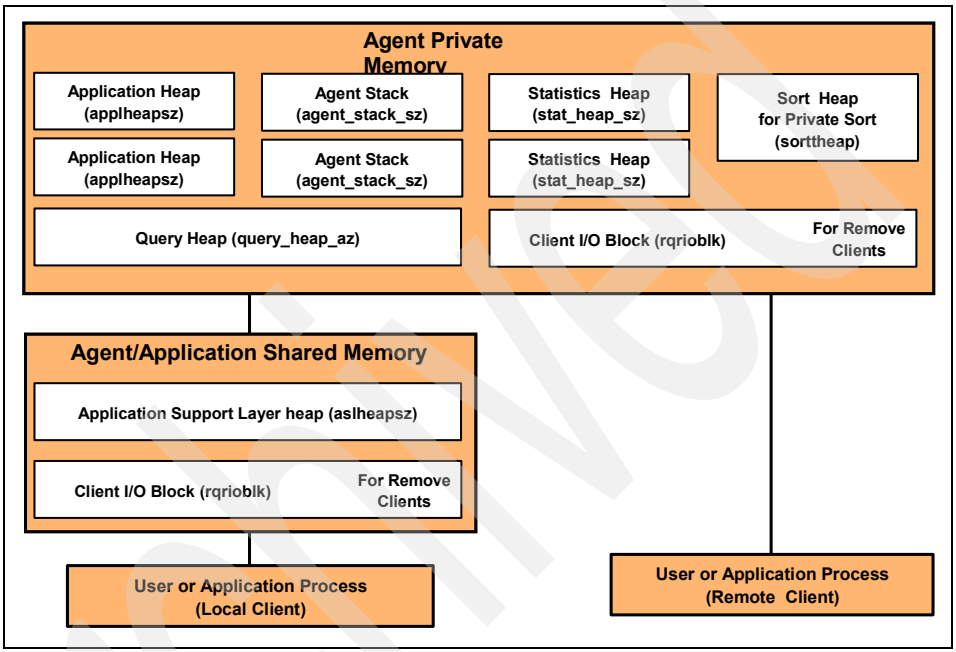

Figure 5-5 Database agent/application private/shared memory overview

The maximum size of Agent Private Memory segments is determined by the values of the following parameters:

- Application Heap Size (APPLHEAPSZ)
- Sort Heap Size (SORTHEAP)
- Statement Heap Size (STMTHEAP)
- Statistics Heap Size (STAT_HEAP_SZ)
- Query Heap Size (QUERY_HEAP_SZ)
- UDF Shared Memory Set Size (UDF_MEM_SZ)
- Agent Stack Size (AGENT_STACK_SZ)
- Client I/O Block Size (RQRIOBLK) (for remote clients)

The size of Agent/Application Shared Memory is affected by the following:

- Application Support Layer Heap Size (ASLHEAPSZ)
- Client I/O Block Size (RQRIOBLK) (for local clients)

For valid ranges and default values of these configuration parameters, see the manual, *DB2 UDB V8 Administration Guide - Performance*, SC09-4821.

Guidelines for tuning parameters that affect memory usage

The first rule for setting memory-allocation parameters is never to set them at their highest values unless such a value has been carefully justified. This rule applies even to systems with the maximum amount of memory. Many parameters that affect memory can allow the database manager easily and quickly to take up all of the available memory on a computer. In addition, managing large amounts of memory requires additional work on the part of the database manager and thus incurs more overhead.

Buffer pools

Buffer pools are a critically important memory component. In this section we describe buffer pools and provide information about managing them for good performance.

Buffer-pool management

A buffer pool is memory used to cache table and index data pages as they are being read from disk, or being modified. The buffer pool improves database system performance by allowing data to be accessed from memory instead of from disk. Because memory access is much faster than disk access, the less often the database manager needs to read from or write to a disk, the better the performance. Because most data manipulation takes place in buffer pools, configuring buffer pools is the single most important tuning area. Only large objects and long field data are not manipulated in a buffer pool.

When an application accesses a row of a table for the first time, the database manager places the page containing that row in the buffer pool. The next time any application requests data, the database manager looks for it in the buffer pool. If the requested data is in the buffer pool, it can be retrieved without disk access, resulting in faster performance.

Memory is allocated for the buffer pool when a database is activated or when the first application connects to the database. Buffer pools can also be created, dropped, and resized while the database is manager is running. If you use the IMMEDIATE keyword when you use ALTER BUFFERPOOL to increase the size of the buffer pool, memory is allocated as soon as you enter the command if the memory is available. If the memory is not available, the changed occurs when all applications are disconnected and the database is reactivated. If you decrease the size of the buffer pool, memory is deallocated at commit time. When all applications are disconnected, the buffer-pool memory is de-allocated.

Pages remain in the buffer pool until the database is shut down, or until the space occupied by a page is required for another page. The following criteria determine which page is removed to bring in another page:

- How recently the page was referenced
- The probability that the page will be referenced again by the last agent that looked at it
- The type of data on the page
- Whether the page was changed in memory but not written out to disk (changed pages are always written to disk before being overwritten.)

In order for pages to be accessed from memory again, changed pages are not removed from the buffer pool after they are written out to disk unless the space is needed.

When you create a buffer pool, the default page size is 4 KB, but you can specify a page size of 4 KB, 8 KB, 16 KB, or 32 KB. Because pages can be read into a buffer pool only if the table-space page size is the same as the buffer-pool page size, the page size of your table spaces should determine the page size that you specify for buffer pools. You cannot alter the page size of the buffer pool after you create it. You must create a new buffer pool with a different page size.

Note: You can create large buffer pools if you have enabled Address Windowing Extensions (AWE) on Advanced Server and Data Center Server on Windows 2000.

Utilizing Windows AWE for buffer pools

On 32-bit platforms, virtual addressable memory is usually limited to between 2 GB and 4 GB. If your 32-bit machine has more real addressable memory than the maximum amount, you can configure any additional real addressable memory beyond virtual addressable memory for buffer pools. Any of the defined buffer pools can use AWE to improve performance.

You can allocate Windows 2000 Address Windowing Extensions (AWE) buffer pools using the DB2_AWE registry variable. Windows AWE is a set of memory management extensions that allow applications to manipulate memory above certain limits, which depend on the process model of the application. For information, refer to your Windows system documentation. Note, however, that if you use the memory for this purpose you cannot also use the extended storage cache.

Setting the DB2_AWE registry variable allows DB2 UDB on 32-bit Windows 2000 platforms to allocate buffer pools that use up to 64 GB of memory. Windows 2000 must be configured correctly to support Address Windowing Extensions (AWE)

buffer pools. This includes associating the “lock pages in memory”-right with the user, allocating the physical pages and the address window pages, and setting this registry variable. In setting this variable you need to know the buffer pool ID of the buffer pool that is to be used for AWE support. The ID of the buffer pool can be seen in the BUFFERPOOLID column of the SYSCAT.BUFFERPOOLS system catalog view. Buffer pools referenced with this registry variable must already exist in SYSCAT.SYSBUFFERPOOLS. To set the DB2_AWE variable use the DB2SET command from a command prompt as follows:

```
DB2SET DB2_AWE=<buffer pool ID>,<number of physical pages>,
<number of address windows>
```

Management of multiple database buffer pools

Although each database requires at least one buffer pool, you might create several buffer pools, each of a different size or with a different page size, for a single database that has table spaces of more than one page size. Each buffer pool has a minimum size, which depends on the platform.

A new database has a default buffer pool called IBMDEFAULTBP with a size determined by the platform and a default page size of 4 KB. When you create a table space with a page size of 4 KB and do not assign it to a specific buffer pool, the table space is assigned to the default buffer pool. You can resize the default buffer pool and change its attributes, but you cannot drop it.

Note: During normal database manager operation, you can use the ALTER BUFFERPOOL command to resize a buffer pool.

Page sizes for buffer pools

After you create or migrate a database, you can create other buffer pools. For example, when planning your database, you might have determined that 8 KB page sizes were best for tables. As a result, you should create a buffer pool with an 8 KB page size as well as one or more table spaces with the same page size. You cannot use the ALTER TABLESPACE statement to assign a table space to a buffer pool that uses a different page size.

Note: If you create a table space with a page size greater than 4 KB, such as 8 KB, 16 KB, or 32 KB, you need to assign it to a buffer pool that uses the same page size. If this buffer pool is currently not active, DB2 attempts to assign the table space temporarily to another active buffer pool that uses the same page size if one or to one of the default “hidden” buffer pools that DB2 creates when the first client connects to the database. When the database is activated again, and the originally specified buffer pool is active, then DB2 assigns the table space to that buffer pool.

When you create a buffer pool, you specify the size of the buffer pool as a required parameter of the SQL DDL statement CREATE BUFFERPOOL. To increase or decrease the buffer-pool size later, use the SQL DDL statement ALTER BUFFERPOOL.

Advantages of large buffer pools

Large buffer pools provide the following advantages:

- They enable frequently requested data pages to be kept in the buffer pool, which allows quicker access. Fewer I/O operations can reduce I/O contention, thereby providing better response time and reducing the processor resource needed for I/O operations.
- They provide the opportunity to achieve higher transaction rates with the same response time.
- They prrevent I/O contention for frequently used disk storage devices such as catalog tables and frequently referenced user tables and indexes. Sorts required by queries also benefit from reduced I/O contention on the disk storage devices that contain the temporary table spaces.

Advantages of many buffer pools

If any of the following conditions apply to your system, you should use only a single buffer pool:

- The total buffer space is less than 10,000 4-KB pages.
- People with the application knowledge to do specialized tuning are not available.
- You are working on a test system.

In all other circumstances, consider using more than one buffer pool for the following reasons:

- Temporary table spaces can be assigned to a separate buffer pool to provide better performance for queries that require temporary storage, especially sort-intensive queries.
- If data must be accessed repeatedly and quickly by many short update-transaction applications, consider assigning the table space that contains the data to a separate buffer pool. If this buffer pool is sized appropriately, its pages have a better chance of being found, contributing to a lower response time and a lower transaction cost.
- You can isolate data into separate buffer pools to favor certain applications, data, and indexes. For example, you might want to put tables and indexes that are updated frequently into a buffer pool that is separate from those tables and indexes that are frequently queried but infrequently updated. This change will reduce the impact that frequent updates on the first set of tables have on frequent queries on the second set of tables.

- You can use smaller buffer pools for the data accessed by applications that are seldom used, especially for an application that requires very random access into a very large table. In such a case, data need not be kept in the buffer pool for longer than a single query. It is better to keep a small buffer pool for this data, and free the extra memory for other uses, such as for other buffer pools.
- After separating different activities and data into separate buffer pools, good and relatively inexpensive performance diagnosis data can be produced from statistics and accounting traces.

Sort performance

In this section we discuss sort performance, which can have a major impact on memory usage.

Guidelines for sort performance

Because queries often require sorted or grouped results, sorting is often required, and the proper configuration of the sort heap areas is crucial to good query performance. Sorting is required when:

- No index exists to satisfy a requested ordering (for example a SELECT statement that uses the ORDER BY clause).
- An index exists but sorting would be more efficient than using the index
- An index is created.
- An index is dropped, which causes index page numbers to be sorted.

Sorting involves two steps:

1. **A sort phase:** A sort can be overflowed or non-overflowed. If the sorted data cannot fit entirely into the sort heap, which is a block of memory that is allocated each time a sort is performed, it overflows into temporary database tables. Sorts that do not overflow always perform better than those that do.
2. **Return of the results of the sort phase:** The return can be piped or non-piped. If sorted information can return directly without requiring a temporary table to store a final, sorted list of data, it is a piped sort. If the sorted information requires a temporary table to be returned, it is a non-piped sort. A piped sort always performs better than a non-piped sort.

Elements that affect sorting

The following elements affect sort performance:

- Settings for the following database configuration parameters:
  - Sort heap size, (sortheap), which specifies the amount of memory to be used for each sort

  - Sort heap threshold (sheapthres) and the sort heap threshold for shared sorts (sheapthres_shr), which control the total amount of memory for sorting available across the entire instance for all sorts
- Statements that involve a large amount of sorting
- Missing indexes that could help avoid unnecessary sorting
- Application logic that does not minimize sorting
- Parallel sorting, which improves the performance of sorts but can only occur if the statement uses intra-partition parallelism.

To find out if you have a sort performance problem, look at the total CPU time spent sorting compared to the time spent for the whole application. You can use the database monitoring tools as described in Chapter 6, "Monitoring and management" on page 295 to see this information, but the tools show total sort time by default, as well as other times such as I/O and lock wait.

If total sort time is a large portion CPU time for the application, then look at the following values, which are also shown by default:

- **Percentage of overflowed sorts:** This variable (on the performance details view of the Snapshot Monitor) shows the percentage of sorts that overflowed. If the percentage of overflowed sorts is high, increase the sortheap and/or sheapthres configuration parameters if there were any post-threshold sorts. To find out if there were any post threshold sorts, use the Snapshot Monitor.
- **Post threshold sorts:** If post threshold sorts are high, increase sheapthres and/or decrease sortheap.

In general, overall sort memory available across the instance (sheapthres) should be as large as possible without causing excessive paging. Although a sort can be performed entirely in sort memory, this might cause excessive page swapping. In this case, you lose the advantage of a large sort heap. For this reason, you should use an operating system monitor to track changes in system paging whenever you adjust the sorting configuration parameters.

Also note that in a piped sort, the sort heap is not freed until the application closes the cursor associated with that sort. A piped sort can continue to use up memory until the cursor is closed.

Techniques for managing sorting performance

Identify particular applications and statements where sorting is a significant performance problem:

- Set up event monitors at the application and statement level to help you identify applications with the longest total sort time.

- Within each of these applications, find the statements with the longest total sort time.
- Tune these statements using a tool such as Visual Explain.
- Ensure that appropriate indexes exist. You can use Visual Explain to identify all the sort operations for a given statement. Then investigate whether or not an appropriate index exists for each table accessed by the statement.

Note: You can search through the Explain tables to identify the queries that have sort operations.

You can use the database system monitor and benchmarking techniques to help set the sortheap and sheapthres configuration parameters. For each database instance and its databases:

- Set up and run a representative workload.
- For each applicable database, collect average values for the following performance variables over the benchmark workload period:
  - Total sort heap in use
  - Active sorts

  The performance details view of the Snapshot Monitor shows these performance variables.
- Set sortheap to the average total sort heap in use for each database.
- Set the sheapthres. To estimate an appropriate size:

  a. Determine which database in the instance has the largest sortheap value.
  b. Determine the average size of the sort heap for this database.

     If this is too difficult to determine, use 80% of the maximum sort heap.

  c. Set sheapthres to the average number of active sorts times the average size of the sort heap computed above.

  This is a recommended initial setting. You can then use benchmark techniques to refine this value.

Agent management

In this section we describe how the database manager uses agents and explain how to manage agents for good performance.

Database agents

For each database that an application accesses, various processes or threads start to perform the various application tasks. These tasks include logging, communication, and prefetching.

Database agents are engine dispatchable unit (EDU) processes or threads, which do the work in the database manager that applications request. In Intel-based operating systems such Windows, the agents run as threads.

The maximum number of application connections is controlled by the max_connections database manager configuration parameter. The work of each application connection is coordinated by a single worker agent.

A worker agent carries out application requests but has no permanent attachment to any particular application. The coordinator worker agent has all the information and control blocks required to complete actions within the database manager that were requested by the application.

There are four types of worker agents:

- **Idle agents:** This is the simplest form of worker agent. It does not have an outbound connection and it does not have a local database connection or an instance attachment.
- **Inactive agents:** An inactive agent is a worker agent that is not in an active transaction, does not have an outbound connection, and does not have a local database connection or an instance attachment. Inactive agents are free to begin doing work for an application connection.
- **Active coordinator agents:** Each process or thread of a client application has a single active agent that coordinates its work on a database. After the coordinator agent is created, it performs all database requests on behalf of its application, and communicates to other agents using inter-process communication (IPC) or remote communication protocols. Each agent operates with its own private memory and shares database manager and database global resources such as the buffer pool with other agents.

  When a transaction completes, the active coordinator agent may become an inactive agent. When a client disconnects from a database or detaches from an instance its coordinating agent will be:
  - An active agent; if other connections are waiting, the worker agent becomes an active coordinator agent.
  - Freed and marked as idle, if no connections are waiting and the maximum number of pool agents has not been reached.
  - Terminated and its storage freed, if no connections are waiting and the maximum number of pool agents has been reached.
- **Subagents:** In partitioned database environments and environments with intra-partition parallelism enabled, the coordinator agent distributes database requests to subagents, and these agents perform the requests for the application. After the coordinator agent is created, it handles all database requests on behalf of its application by coordinating the subagents that perform requests on the database.

Agents that are not performing work for any applications and that are waiting to be assigned are considered to be idle agents and reside in an agent pool. These agents are available for requests from coordinator agents operating for client programs or for subagents operating for existing coordinator agents. The number of available agents depends on the database manager configuration parameters maxagents and num_poolagents.

When an agent finishes its work but still has a connection to a database, it is placed in the agent pool. Regardless of whether the connection concentrator is enabled for the database, if an agent is not waked up to serve a new request within a certain period of time and the current number of active and pooled agents is greater than num_poolagents, the agent is terminated.

Agents from the agent pool (num_poolagents) are re-used as coordinator agents for the following kinds of applications:

- Remote TCP/IP-based applications
- Local applications on UNIX-based operating systems
- Both local and remote applications on Windows operating systems.

Other kinds of remote applications always create a new agent. If no idle agents exist when an agent is required, a new agent is created dynamically. Because creating a new agent requires a certain amount of overhead CONNECT and ATTACH performance is better if an idle agent can be activated for a client.

When a subagent is performing work for of an application, it is associated with that application. After it completes the assigned work, it can be placed in the agent pool, but it remains associated with the original application. When the application requests additional work, the database manager first checks the idle pool for associated agents before it creates a new agent.

Database-agent management

Most applications establish a one-to-one relationship between the number of connected applications and the number of application requests that can be processed by the database. However, it may be that your work environment is such that you require a many-to-one relationship between the number of connected applications and the number of application requests that can be processed.

The ability to control these factors separately is provided by two database manager configuration parameters:

- **The max_connections parameter:** This specifies the number of connected applications.
- **The max_coordagents parameter:** This specifies the number of application requests that can be processed.

The connection concentrator is enabled when the value of max_connections is greater than the value of max_coordagents.

Because each active coordinator agents requires global resource overhead, the greater the number of these agents, then the greater the chance that the upper limits of available database global resources will be reached. To prevent reaching the upper limits of available database global resources, you might set the value of max_connections higher than the value of max_coordagents.

Configuration parameters that affect the number of agents

The following database manager configuration parameters determine how many database agents are created and how they are managed:

- **Maximum Number of Agents (maxagents):** The number of agents that can be working at any one time. This value applies to the total number of agents that are working on all applications, including coordinator agents, subagents, inactive agents, and idle agents.
- **Agent Pool Size (num_poolagents):** The total number of agents, including active agents and agents in the agent pool, that are kept available in the system. The default value for this parameter is half the number specified for maxagents.
- **Initial Number of Agents in Pool (num_initagents):** When the database manager is started, a pool of worker agents is created based on this value. This speeds up performance for initial queries. The worker agents all begin as idle agents.
- **Maximum Number of Connections (max_connections):** This specifies the maximum number of connections allowed to the database manager system on each partition.
- **Maximum Number of Coordinating Agents (max_coordagents):** For partitioned database environments and environments with intra-partition parallelism enabled when the connection coordinator is enabled, this value limits the number of coordinating agents.
- **Maximum Number of Concurrent Agents (maxcagents):** This value controls the number of tokens permitted by the database manager. For each database transaction (unit of work) that occurs when a client is connected to a database, a coordinating agent must obtain permission to process the transaction from the database manager. This permission is called a processing token. The database manager permits only agents that have a processing token to execute a unit of work against a database. If a token is not available, the agent must wait until one is available to process the transaction. This parameter can be useful in an environment in which peak usage requirements exceed system resources for memory, CPU, and disk.

For example, in such an environment, paging might cause performance degradation for peak load periods. You can use this parameter to control the load and avoid performance degradation, although it can affect either concurrency or wait time, or both.

Connection-concentrator improvements for client connections

For Internet applications with many relatively transient connections, or similar kinds of applications, the connection concentrator improves performance by allowing many more client connections to be processed efficiently. It also reduces memory use for each connection and decreases the number of context switches.

Note: The connection concentrator is enabled when the value of max_connections is greater than the value of max_coordagents.

In an environment that requires many simultaneous user connections, you can enable the connection concentrator for more efficient use of system resources. With the connection concentrator, the active agent does not close its connection after a client disconnects, but is placed in the agent pool for the application, where it becomes a *logical subagent*, which is controlled by a *logical coordinator agent* with an active connection to the database manager.

Usage examples

We now offer four cases showing how these parameters are used:

- **Case 1:** Consider an ESE environment with a single database partition in which, on average, 1000 users are connected to the database. At times, the number of concurrent transactions is as high as 200, but never higher than 250. Transactions are short. For this workload, the administrator sets the following database manager configuration parameters:
  - max_connections is set to 1000 to ensure support for the average number of connections.
  - max_coordagents is set to 250 to support the maximum number of concurrent transactions.
  - maxagents is set high enough to support all of the coordinator agents and subagents (where applicable) that are required to execute transactions on the node.

  If intra_parallel is OFF, maxagents is set to 250 because in such an environment, there are no subagents. If intra_parallel is ON, maxagents should be set large enough to accommodate the coordinator agent and the subagents required for each transaction that accesses data on the node. For example, if each transaction requires 4 subagents, maxagents should be set to (4+1) * 250, which is 1250.

To tune maxagents further, take monitor snapshots for the database manager. The high-water mark of the agents will indicate the appropriate setting for maxagents:

- num_poolagents is set to at least 250, or as high as 1250, depending on the value of maxagents to ensure that enough database agents are available to service incoming client requests without the overhead of creating new ones. However, this number could be lowered to reduce resource usage during low-usage periods. Setting this value too low causes agents to be deallocated instead of going into the agent pool, which requires new agents to be created before the server is able to handle an average workload.
- num_init_agents is set to be the same as num_poolagents because you know the number of agents that should be active. This causes the database to create the appropriate number of agents when it starts instead of creating them before a given request can be handled.

The ability of the underlying hardware to handle a given workload is not discussed here. If the underlying hardware cannot handle X-number of agents working at the same time, then you need to reduce this number to the maximum that the hardware can support. For example, if the maximum is only 1500 agents, then this limits the maximum number of concurrent transactions that can be handled. You should monitor this kind of performance-related setting because it is not always possible to determine exact requests sent to other nodes at a given point in time.

- **Case 2:** In a system in which the workload needs to be restricted to a maximum 100 concurrent transactions and the same number of connected users as described in Case 1, you can set both of the following database manager configuration parameters to 100:
  - max_coordagents
  - num_poolagents

  With these settings, the maximum number of clients that can concurrently execute transactions is 100. When all clients disconnect, 100 agents are waiting to service new client connections. However, you should set maxagents, based on the type of workload, intra-query parallelism settings, the number of database partitions, and the underlying hardware.
- **Case 3:** Consider next an ESE installation with five database partitions, in which each partition has an average of 1000 user connections, and the concurrent transactions are as high as 200 but never higher than 250, set database configuration parameters as follows:
  - max_coordagents is set to 250, because, as in Case 1, at most, 250 clients execute transactions concurrently.

  - maxagents is set to 1500. Assuming data is distributed across the five partitions, each transaction may execute at least one subsection on each of the nodes in the system ((1 coordinator agent + 5 subagents) * 250 = 1500).
  - num_poolagents is set to 1200. Assuming an average of 200 concurrent transaction with a maximum of 250, then, as a result, the number of agents required, on average, will be (1+5)*200 = 1200).
  - num_init_agents is set to be the same as num_poolagents, as in Case 1.
- **Case 4:** In a system for which you do not want to enable the connection concentrator but want to allow for 250 connected users at one time, simply set both of the following database manager configuration parameters to 250:
  - max_connections
  - max_coordagents

5.3.3 Processor

In this section we discuss the configuration parameters which affect CPU usage. A database instance has several DB2 processes, and each of them consumes CPU time. Among the processes, the major consumers of CPU time are DB2 agents, including coordinator agents and subagents. They are one of the most crucial processes and facilitate the operations of applications with databases. If your database server has multiple CPUs and you enable intra-partition parallelism, the database manager can exploit those CPUs using multiple subagents to process one complex query.

Intra-partition parallelism

The database manager configuration parameter INTRA_PARALLEL controls whether or not intra-partition parallelism is enabled. We suggest that this parameter should be enabled or disabled depending on these considerations:

- Typical workload
- The number of users

For complex SQL type workloads with relatively few users (such as OLAP or DSS), then enable intra-partition parallelism (set to YES). For SQL which is simple, repetitive and the number of queries are large (such as OLTP), then do not enable intra-partition parallelism (set to NO).

If you do disable parallelism by setting the INTRA_PARALLEL database manager configuration parameter to NO, we advise that you set the MAX_QUERYDEGREE database manager configuration parameter or DEFAULT_DEGREE database configuration parameters to 1.

With intra-partition parallelism enabled, DB2 will allocate approximately 12 MB of memory (used for control information). This control area will be checked for each statement. So by setting MAX_QUERYDEGREE or DFT_DEGREE to try to avoid parallelism is meaningless as you still incur this overhead. This overhead will not be incurred with INTRA_PARALLEL set to NO.

Here are some additional considerations:

- **MAX_QUERYDEGREE and DFT_DEGREE:** Set this database manager configuration parameter to –1 (ANY) if you have set INTRA_PARALLEL to YES. By doing this, you allow the DB2 optimizer to decide the most suitable value. The value it chooses is based on complexity of query, database and instance configuration, and workload.
- **CPUSPEED:** Set this database manager configuration parameter to –1. This will be the most recommended value for DB2. By doing this, DB2 will calculate the value.

For new instances, make your decision on intra-partition parallelism before you create your database(s). If you change this value once a database exists, then you will need to rebind all packages defined in that database.

Controlling the number of DB2 agent processes

In addition to the impact on memory discussed in “Agent management” on page 218, agents management can impact CPU resources. Enabling intra-partition parallelism and having more concurrent users will increase the number of DB2 agent processes. For example, if you disable intra-partition parallelism, five concurrent query requests will be carried out by five DB2 agent processes respectively; however, if you enable intra-partition parallelism and the chosen query degree is four, twenty-five DB2 agent processes (one coordinator agents and four subagents for each request) will carry out these five query requests.

Having a huge number of DB2 agent processes may cause CPU constraints due to context switching, as well as memory constraints.

To control the number of DB2 agents, you have the following configuration parameters:

- The database manager configuration parameter MAXAGENTS defines the maximum number of database manager agents, whether it is coordinator agents or subagents, available at any given time to accept application requests.
- The database manager configuration parameter MAX_COORDAGENTS defines the maximum number of coordinator agents.

5.3.4 Storage

In this section we discuss some of the options available for improving the performance of input/output operations within DB2. We cover the following topics:

- Table space issues
- Prefetching concepts
  - Prefetching data into the buffer pool
  - Sequential prefetching
  - Sequential detection
  - Block-based buffer pools for improved sequential prefetching
- Parallel I/O management
- Multipage file allocation
- Logging

Table space issues

Table spaces in DB2 can be defined as either System Managed Storage (SMS) or Database Managed Storage (DMS), which we discuss in the following sections.

SMS table spaces

Containers are operating system directories; you can easily increase table space capacity by enlarging the underlying operating system file system. Data is striped across the container by extent; disk space is allocated on demand one page at a time (by default, see “Multipage file allocation” on page 243 to change this).

With SMS table spaces, data objects (table data, indexes, LONG VARCHARs, and LOBs) are located by operating system file name.

These are some advantages of using SMS table spaces:

- Space is not allocated until required.
- Initial database creation may be simpler (no DMS container definitions are required).
- Overall administration of space for the database is simpler.

DMS table spaces

Containers are either operating system files or raw devices. We recommend that, where possible, you should associate each container with a different disk or disks, as this enables parallel I/O, and gives the opportunity for larger table space capacity.

When using DMS device containers in a table space, you need to understand the following performance considerations:

- File system caching for SMS file containers is performed by the operating system in the file system cache (see “Bypass file system caching” on page 265 to change this).
- File system caching for DMS device containers is NOT done by the operating system.

In DMS table spaces, data is striped across the container(s) by extent. Extents are mapped in a round-robin fashion, and extents for different objects need not be contiguous.

Here are some advantages of using DMS table spaces:

- Containers can be added, extended and resized
- Large tables can be split up by data type (LOBs, indexes, data) across multiple table spaces.
- DMS, in most situations, will give better performance than SMS.

SMS versus DMS

In some situations, it can be difficult to decide whether to use SMS table spaces or DMS table spaces, as there are a number of different factors to consider. In general we recommend that you should use DMS table spaces with device (raw logical volume) containers if performance is your main priority. The database can do a much better job than the file system when it comes to managing its own disk blocks. However, in this section of the chapter we discuss situations where either SMS or DMS file containers may be a reasonable and sometimes more suitable alternative.

When using DMS raw devices, DB2 will ensure that pages are placed contiguously on the disk drives; with SMS containers, this is not the case, as the file system decides where to locate the pages (this can also apply to DMS file containers). Contiguous placement of pages is important, as pages make up DB2 extents. Contiguous pages will speed up operations like table scans.

Table space categories

In DB2 we have seen that table spaces can be either SMS or DMS. We can also break down table spaces into the following categories which represent the data type for a table space.

Regular table spaces

You can think of a regular table space as a table space for containing user data; the default table space for this data type is called USERSPACE1. Index data is also stored in regular table spaces, as are the system catalog tables. The default

table space for system catalogs is called SYSCATSPACE. Both USERSPACE1 and SYSCATSPACE are created when the CREATE DATABASE command is run, and both are created as SMS type table spaces by default.

You can drop USERSPACE1 if you wish, as its use for data is optional. Mostly, you will create additional table spaces for your data. However, the name SYSCATSPACE must be used for the table space which is holding the system catalogs.

Long table spaces
Any tables which contain long field data or long object data will occupy this type of table space.

When any application has a need to access data from a long table space (whether the data be LOBs, LONG VARCHAR, or LONG VARGRAPHIC), the database manager cannot use the buffer pools to cache the data. Therefore, every time a page is requested, the database manager must request it from disk.

Long table spaces must be DMS type table spaces. Use of long table spaces is optional, as the data type can reside in regular table spaces.

From a performance point of view, since LONG data is NOT stored in the buffer pools, then you can use DMS file containers instead of raw devices. The reason for this is that by using the file containers, you will benefit from the operating system's file system cache.

System temporary table spaces
System temporary table spaces are used during SQL operations for internal temporary tables, for sorts, to store intermediate results, table reorganizations, and other transient data. All databases in DB2 must have at least one system temporary table space. The default, called TEMPSPACE1, is created by default as an SMS table space.

User temporary table spaces
User temporary table spaces are used to store declared temporary tables once a user or application has defined such a table.

Choosing table space types for data tables
When deciding what type of table space to create to provide optimal performance (for example, to maximize I/O throughput), you need to be familiar with the following concepts before making your decision:

- Big block reads, a read where several pages (normally an extent are retrieved in a single request.
- Prefetching, reading pages in advance, in an attempt to reduce query response times.

- Page cleaning, the writing of modified buffer pool pages to disk to make room for a new page to be read in.

DB2 tries to perform big-block reads whenever possible to maximize the amount of data read in one operation. When using DMS device containers the data will in most cases be contiguous on the disk, which reduces seek time and latency. When using SMS or DMS file containers, then the operating system's file system may have broken the files up and stored the different parts of the file in different locations on the disk(s) which means the file has become fragmented. This obviously gives a read performance degradation.

Prefetching is discussed later in this chapter.

Data tables

Raw data and indexes are classed as regular user data. To maximize performance, choose DMS with raw containers. SMS has been thought of as giving a lower level of performance but superior (simpler) system administration. However, due to a number of enhancements to DMS, this argument is weakening slightly (features like the DMS table space resize option affect this).

Raw containers will, in nearly all cases, outperform file containers because the database manager can bypass the operating system's file system and also avoid unnecessary double buffering.

LOB or LONG VARCHAR data benefits from the use SMS or DMS table spaces with FILE containers, as they benefit from the operating system's file system cache. LOB and LONG VARCHAR are not buffered in DB2's buffer pools, as they use direct I/O. Any memory allocated to long table spaces should be kept to a minimum, or alternatively, just use the default buffer pool IBMDEFAULTBP.

Temporary table spaces

We suggest that for nearly all systems, SMS is the right table space type to choose for both system and user temporary table spaces.

The main advantage of using SMS for temporary table spaces is that data stored in temporary tables is mostly transient data (for example batch work). By using SMS you will only allocate disk space as and when required, unlike DMS, which will allocate all specified space when the table space is created.

If you have table spaces utilizing different page sizes, then our suggestion would be to create one temporary table space with a page size equal to that of the majority of your data tables. When selecting temporary table spaces, DB2 ignores page sizes which are too small to accommodate the operation which requires temporary space. For this reason, you will need to ensure that the page size of your temporary table space(s) can accommodate the row lengths and column used in your database.

The reason we suggest only having one temporary table space for a given page size is because when you have multiple temporary table spaces using same page sizes, the DB2 optimizer will generally choose a temporary table space size by selecting the temporary table space with the largest buffer pool. Let us say you had two 16 KB temporary table spaces defined called TEMP1 and TEMP2. Now assume TEMP1 has a large buffer pool and TEMP2 has only a small buffer pool; UDB will select TEMP1 as it has largest buffer pool. Once selection is made, UDB will then alternate between ALL temporary table spaces which have the chosen page size (16 KB).

This means that when performing an operation such as a sort, we alternate between TEMP1 and TEMP2 because they both use a 16 KB page size. The disadvantage here is that we only use TEMP1's large buffer pool half the time, because we are alternating between the two table spaces. A better solution in this example would be to have a single 16 KB temporary table space with one buffer pool.

Catalog table spaces

The system catalogs contain some LOB columns, as we described earlier in this section. When the database manager needs to retrieve a page of LOB data, it cannot use the buffer pools as a cache. The only way to improve performance by providing some buffering is to use SMS or DMS file containers. When these container types are utilized, the operating system's file system cache will provide some buffering.

For the catalog table space, use SMS or DMS with FILE containers and a small extent size (2 or 4 pages).

The reason for this is that the catalogs contain a lot of relatively small tables. DMS requires 2 overhead extents per table (Extent Map Page + 1st data extent). SMS requires only 1 page.

Deciding number of tables and table spaces

A common question asked is, "How many tables should I put in a table space?". This in turn affects another frequently asked design question, "How many table spaces should I create?" Whatever number is used, the decision can affect the performance of that particular table and table space.

Here are some considerations for placing tables, which might help you decide which tables should go:

- When recovering your table spaces, if you have a collection of tables related by Referential Integrity (RI) constraints or summary tables, then if you ever need to restore the tables, they should be rolled forward together to a point in time. If this is not done, then one table, or more, will be placed in a check pending state.

- If you are using DMS table spaces, you then have the option of creating what are known as “spanned” tables. A spanned table has data pages in more than one table space. The best example of this is a table which has its data in one table space, indexes in another, and LOB data in another. Only DMS supports this. An advantage of using spanned tables is that, by separating data and indexes, you can place the indexes in a table space with its own dedicated buffer pool, which can help ensure that index pages are kept in memory.
- How much raw data is in the tables is also important. If you have a number of small tables with a small amount of data which are not changed frequently, then it is feasible to group these tables together in a single table space. Larger base tables which are heavily accessed and frequently updated can justify having their own DMS table space. From a performance point of view, if a very large table has its own table space, then you can also assign it a dedicated buffer pool (see “Advantages of many buffer pools” on page 215 for details on this subject). By grouping tables in specific table spaces, you can perform flexible backup strategies as you have more granularity.
- Keeping important tables in their own table space will simplify administration of this table space and will speed recovery should the table space need to be restored (because you will only have one table to restore). When required, you can also create a dedicated buffer pool for these tables.
- Group infrequently accessed data in table spaces which use containers on slower devices.
- Minimize relationships (such as RI constraints) between table spaces; this simplifies recovery.
- Consider table space capacity limits: 64 GB (4K pages), 128 GB (8K pages), 512 GB (32K pages), 2 TB for long table spaces. These are the limits for DB2 UDB Enterprise Server (ESE). If you use the Database Partitioning Feature (DPF), they are the limits per partition.
- Consider leaving room for in-place table reorganization, which is faster than reorganization using a system temporary table space because data copied only once and not copied back and forth between temporary table space and source table space.
- Place tables that need to be monitored more in their own table spaces, since the LIST TABLESPACE SHOW DETAIL command gives the number of used pages. If the table space is a DMS table space, the command also reports the number of free pages and the high water mark information. For example, you can monitor the growth of a fact table of a star schema by putting it in its own table space.
- Try to minimize the overall number of table spaces.

Recoverability of related table spaces

It is worth remembering when assigning tables to table spaces that if you ever need to recover a table space, then “relationships” with tables in other table spaces will be a factor. For example, assume we have two table spaces TS1 and TS2; we recover TS1 from a backup, and then wish to roll forward to a point in time. We then realize that one of the tables within T1 is a summary table for one of the tables in T2. If we now simply roll forward TS1 and do not roll forward TS2 to the same point in time, the summary table in TS1 will be placed into a check pending state. The same would apply if the table in TS1 was the underlying table for the summary table contained in TS2.

Choosing table space containers

A container in DB2 is a physical storage device, and each container is assigned to one table space. Containers can be either files, directories, or devices; we look at these in this section. Whichever container type you decide to use for a table space, you should stay with it. You will NOT gain good performance by mixing container types within a table space.

Numerous benchmarks on device containers (raw logical volumes) versus directory and file containers have shown an overall improvement in disk I/O throughput of 10-35 percent when using device containers, as compared to file systems. However, the database administrator needs to take into account that actual gains will vary, depending on the I/O workload mix of the application, OLTP or DSS.

Directory containers

This is the only container type that can be used in SMS table spaces. When creating SMS table spaces, make sure you specify at creation time all the directories (containers) that you want to incorporate into the table space; additional containers CANNOT be added once the table space exists.

To improve performance with this container type, we recommend that each container be mapped to a different physical drive to improve parallel I/O.

File containers

File containers are used by DMS table spaces and are files of a pre-allocated size. DMS treats file and device containers in the same way. When choosing a size for the file, be aware that the space required, for the chosen size, is pre-allocated. When the file is created, DB2 will allocate the entire container (file) at table space creation time. Even though this allocation is done at creation time, the operating system’s file system may still fragment the file, so in some cases, the allocation of pages may not be contiguous.

If using DMS table spaces, then in most cases, file containers will give poorer performance when compared to raw device containers. The reason for this is that with file containers, the database manager has to interact with the operating system's file system layer, unlike raw device containers, which the database manager manages directly. An exception to this rule would be a DMS table space using file containers that was created to hold LOB data. In this case, performance might be greater than that of device container, since DB2 can take advantage of the file system cache. Remember that LOB data is NOT buffered by DB2, so using device containers would result in direct disk I/O operations when accessing LOB data.

Device containers

Device containers are commonly known as raw devices. This type of container can only be used by DMS table spaces. When using device containers, the space, as with file containers, is also allocated at table space creation time. However, in this case, DB2 interacts with the raw device directly, which ensures that page allocation is contiguous.

The recommendation would be to use device containers when maximum performance is your goal, the exception being DMS table spaces designed for LOB data, as mentioned earlier.

The DMS and device container combination avoid the double buffering scenario which can occur with DMS file containers or SMS directory containers. This occurs when pages are cached by the operating system in its file system cache and by DB2 in the buffer pools. This scenario is a waste of resources.

Configuring table space containers

Before you create your table spaces, you need to think about how the container configuration will affect the overall performance of the table space. We now look at the major factors affecting table space container configuration.

Disk considerations

The total number of disks you will require is determined by the amount of raw data you have to store. For performance, the main thing to remember is that by using multiple disks for your containers, you will enable DB2 to perform parallel I/O. For example, you would generally achieve better performance with a large number of low-capacity disk drives, allowing you to have a drive for each table space container, than you would from having a small number of high-capacity disk drives with multiple containers on each drive.

For random access workloads (such as OLTP), you will need as many disks in the table space as are necessary to support the required I/O. For this type of workload, writes will be the main concern, as they are more expensive than reads, therefore the read/write ratio of one physical write has to be considered.

For sequential access workloads (such as a data warehouse or DSS), some of the same points apply; you will need sufficient disks to support the required I/O. However, in most situations, DSS workloads will be more “disk-friendly”; this is because they have larger I/O sizes, which are generally sequential. This, along with other factors such as prefetching and disk caching, means that a DSS workload may be able to work efficiently with fewer disks than an OLTP system, with similar I/O requirements.

Other disk recommendations would be:

- Try to spread your table spaces over as many disks as possible; however, do NOT place more than one heavily accessed table space on a disk.
- If you only have a small number of disks available, then it is better to spread ALL the table spaces over ALL the disks rather than to assign only one disk to each table space.

Page size

The overall performance of your table space will be affected by the page size you choose when issuing the CREATE TABLESPACE statement. The type of workload you are going to put on the tables within the table space will help determine an optimal page size.

If your applications perform mainly random reads/writes, then we would recommend that a small page size should be used, as this will not waste space in the buffer pools. If, however, your workload is more DSS than OLTP, and accesses rows sequentially, then larger page sizes will provide better performance.

Note: Use large page sizes for temporary table spaces, as rows in tables that reside in this type of table space will be accessed sequentially.

However, there is an exception to this rule. When row size is smaller than page size/255, there will be wasted space on each page, as there is a maximum of 255 rows per page. If this applies to your system, then using a smaller page size may be more appropriate. Remember that the database manager allocates 76 bytes in each page (regardless of size) for control information. So using the default size of 4 KB as an example, this would leave 4020 bytes for data.

Larger page sizes may allow you to reduce the number of levels in the index, since more index keys will fit on each index leaf page. However, remember that each index page can have only 255 index entries. Therefore, if your index key size is small, a good percentage of the page might be wasted with a large page size.

Extent size
The extent size you choose for a table space determines the number of table data pages written to one container before DB2 writes to the next container; the type of table space used is a major factor. When using DMS table spaces, space is allocated one extent at a time; when one extent is full, a whole new extent is allocated. This differs from SMS table spaces which allocate space one page at a time (by default see “Multipage file allocation” on page 243).

This allocation policy will influence your table space performance. Let us assume you have a number of small tables which you want to group together in one table space. If you decide to use DMS, when a new table is created, two extents are immediately required for the table object. If you have many small tables in this table space, then the database manager would allocate a lot of disk space to store a small amount of data.

In this situation, the recommendation would be to use a small extent size, or if performance is not critical for these tables, then you can use an SMS table space which allocates pages one at a time.

You may notice when using DMS table spaces that a number of pages are used up when you first create the table space and then create the first object (table).

The default EXTENTSIZE of 32 pages is generally acceptable for most systems, but this should be reduced if your table space consists of many small tables. Use larger values for tables that are mostly scanned (query only) or have a very high growth rate. If your DMS table space does contain large tables, and you do not increase the value for EXTENTSIZE but instead use a small extent size, then you will incur an overhead when the database has to continually allocate these small extents (either when a table is expanding or new tables are being created).

When using “striped” containers (like RAID), extent size should be set to a multiple of the stripe size (such as RAID strip size).

Note: When using multidimensional clustering tables (MDC) the columns selected for the dimensions can have a significant impact on space utilization if the extent size is not properly set. See the manual, *DB2 UDB V8 Administration: Planning*, SC09-4822, Chapter 4, “Logical Database Design” under the section on multidimensional clustering.

Deciding how many containers to use
Once the decision has been made as to the configuration of the table space containers, the next set of questions that commonly arise are to do with the number of containers to use in a table space and what factors affect this decision.

SIngle or multiple containers

Other than the considerations we look at next in “Container sizes”, there are some other points to be made regarding having single or multiple containers in a table space and how this affects performance.

As long as the number of table spaces is small, then it is possible to have a one-to-one mapping of table space containers to physical volumes. This greatly simplifies the conceptual view of the physical database.

Normally you want to map each disk to its own container for the following reasons:

- Isolation from disk errors
- Ability to better isolate parts of a workload (or database) from one another.

  For example, separating tables scanned simultaneously into separate table spaces with separate disks can reduce disk head movement (thus seek time) when doing sequential scans.

  As another example, you may want to protect time-critical data from random/adhoc queries on other data (remember that there are usually still shared CPUs and buses, so this is not always effective).

Container sizes

One of the factors which will affect the parallel I/O performance is the size of the containers used. If you use multiple containers of different sizes in a table space, then the effectiveness of any parallel prefetches performed against that table space will be reduced, for example:

- You create a new table space containing one small container and one large container. Eventually the small container will become full and any additional data will be written to the large container. The two containers are now unbalanced. Having unbalanced containers reduces the performance of parallel prefetching to nothing, as a SELECT statement may result in some data being required from the small container, but a large amount may be required from the large container.
- You have a table space which contains a number of large containers; these start to fill up, so you add a small container. The table space will then be rebalanced and the small container will contain far less data than will now reside in the larger containers, and once again the database manager will not be able to optimize parallel prefetching.

In summary, if you must use multiple containers, keep them the same size for best performance, and keep the container types the same if performance is your goal.

Prefetching concepts

Prefetching data into the buffer pools usually improves performance by reducing the number of disk accesses and retaining frequently accessed data in memory.

Prefetching data into the buffer pool

Prefetching pages means that one or more pages are retrieved from disk in the expectation that they will be required by an application. Prefetching index and data pages into the buffer pool can help improve performance by reducing the I/O wait time. In addition, parallel I/O enhances prefetching efficiency.

There are two categories of prefetching:

- **Sequential prefetch:** A mechanism that reads consecutive pages into the buffer pool before the pages are required by the application.
- **List prefetch:** Sometimes called list sequential prefetch. Prefetches a set of non-consecutive data pages efficiently.

These two methods of reading data pages are in addition to a normal read. A normal read is used when only one or a few consecutive pages are retrieved. During a normal read, one page of data is transferred.

Prefetching and intra-partition parallelism

Prefetching is important to the performance of intra-partition parallelism, which uses multiple subagents when scanning an index or a table. Such parallel scans introduce larger data-consumption rates, which require higher prefetch rates.

The cost of inadequate prefetching is higher for parallel scans than serial scans. If prefetching does not occur for a serial scan, the query runs more slowly because the agent always needs to wait for I/O. If prefetching does not occur for a parallel scan, all subagents might need to wait because one subagent is waiting for I/O.

Because of its importance, prefetching is performed more aggressively with intra-partition parallelism. The sequential detection mechanism tolerates larger gaps between adjacent pages so that the pages can be considered sequential. The width of these gaps increases with the number of subagents involved in the scan.

Sequential prefetching

Reading several consecutive pages into the buffer pool using a single I/O operation can greatly reduce your application overhead. In addition, multiple parallel I/O operations to read several ranges of pages into the buffer pool can help reduce I/O wait time.

Prefetching starts when the database manager determines that sequential I/O is appropriate and that prefetching might improve performance. In cases such as table scans and table sorts, the database manager can easily determine that sequential prefetch will improve I/O performance. In these cases, the database manager automatically starts sequential prefetch. The following example, which probably requires a table scan, would be a good candidate for sequential prefetch:

```
SELECTNAME FROM EMPLOYEE
```

Implications of the PREFETCHSIZE for table spaces

To define the number of prefetched pages for each table space, use the PREFETCHSIZE clause in either the CREATE TABLESPACE or ALTER TABLESPACE statements. The value that you specify is maintained in the PREFETCHSIZE column of the SYSCAT.TABLESPACES system catalog table.

It is a good practice to explicitly set the PREFETCHSIZE value as a multiple of the number of table space containers and the EXTENTSIZE value for your table space, which is the number of pages that the database manager writes to a container before it uses a different container. For example, if the extent size is 16 pages and the table space has two containers, you might set the prefetch quantity to 32 pages.

The database manager monitors buffer-pool usage to ensure that prefetching does not remove pages from the buffer pool if another unit of work needs them. To avoid problems, the database manager can limit the number of prefetched pages to less than you specify for the table space.

The prefetch size can have significant performance implications, particularly for large table scans. Use the database system monitor and other system monitor tools to help you tune PREFETCHSIZE for your table spaces. You might gather information about whether there are I/O waits for your query, by using monitoring tools available for your operating system), or whether prefetch is occurring, by looking at the pool_async_data_reads (buffer pool asynchronous data reads) data element provided by the database system monitor.

If there are I/O waits and the query is prefetching data, you might increase the value of PREFETCHSIZE. If the prefetcher is not the cause of the I/O wait, increasing the PREFETCHSIZE value will not improve the performance of your query.

In all types of prefetch, multiple I/O operations might be performed in parallel when the prefetch size is a multiple of the extent size for the table space and the extents of the table space are in separate containers. For better performance, configure the containers to use separate physical devices.

Sequential detection

In some cases it is not immediately obvious that sequential prefetch will improve performance. In these cases, the database manager can monitor I/O and activate prefetching if sequential page reading is occurring. In this case, prefetching is activated and deactivated by the database manager as appropriate. This type of sequential prefetch is known as sequential detection and applies to both index and data pages. Use the seqdetect configuration parameter to control whether the database manager performs sequential detection.

For example, if sequential detection is turned on, the following SQL statement might benefit from sequential prefetch:

```
SELECTNAME FROM EMPLOYEE
WHERE EMPNO BETWEEN 100 AND 3000
```

In this example, the optimizer might have started to scan the table using an index on the EMPNO column. If the table is highly clustered with respect to this index, then the data-page reads will be almost sequential and prefetching might improve performance, so data-page prefetch will occur.

Index-page prefetch might also occur in this example. If many index pages must be examined and the database manager detects that sequential page reading of the index pages is occurring, then index-page prefetching occurs.

Block-based buffer pools for improved sequential prefetching

Prefetching pages from disk is expensive because of I/O overhead. Throughput can be significantly improved if processing is overlapped with I/O. Most platforms provide high-performance primitives that read contiguous pages from disk into non-contiguous portions of memory. These primitives are usually called scattered read or vectored I/O. On some platforms, performance of these primitives cannot compete with doing I/O in large block sizes.

By default, the buffer pools are page-based, which means that contiguous pages on disk are prefetched into non-contiguous pages in memory. Sequential prefetching can be enhanced if contiguous pages can be read from disk into contiguous pages within a buffer pool.

You can create block-based buffer pools for this purpose. A block-based buffer pool consist of both a page area and a block area. The page area is required for non-sequential prefetching workloads. The block area consist of blocks where each block contains a specified number of contiguous pages, which is referred to as the block size.

The optimal usage of a block-based buffer pool depends on the specified block size. The block size is the granularity at which I/O servers doing sequential prefetching consider doing block-based I/O. The extent is the granularity at which table spaces are striped across containers. Because multiple table spaces with different extent sizes can be bound to a buffer pool defined with the same block size, consider how the extent size and the block size interact for efficient use of buffer-pool memory. Buffer-pool memory can be wasted in the following circumstances:

- If the extent size, which determines the prefetch request size, is smaller than BLOCK_SIZE specified for the buffer pool.
- If some pages requested in the prefetch request are already present in the page area of the buffer pool.

The I/O server allows some wasted pages in each buffer-pool block, but if too much of a block would be wasted, the I/O server does non-block-based prefetching into the page area of the buffer pool. This is not optimal performance.

For optimal performance, bind table spaces of the same extent size to a buffer pool with a block size that equals the table-space extent size. Good performance can be achieved if the extent size is larger than the block size, but not when the extent size is smaller than the block size.

To create block-based buffer pools, use the CREATE and ALTER BUFFERPOOL statements. Block-based buffer pools have the following limitations:

- A buffer pool cannot be made block-based and use extended storage simultaneously.
- Block-based I/O and AWE support cannot be used by a buffer pool simultaneously. AWE support takes precedence over block-based I/O support when both are enabled for a given buffer pool. In this situation, the block-based I/O support is disabled for the buffer pool. It is re-enabled when the AWE support is disabled.

Note: Block-based buffer pools are intended for sequential prefetching. If your applications do not use sequential prefetching, then the block area of the buffer pool is wasted.

I/O server configuration for prefetching and parallelism

To enable prefetching, the database manager starts separate threads of control, known as I/O servers, to read data pages. As a result, the query processing is divided into two parallel activities: data processing (CPU) and data page I/O. The I/O servers wait for prefetch requests from the CPU processing activity. These prefetch requests contain a description of the I/O needed to satisfy the query.

The possible prefetch methods determine when and how the database manager generates the prefetch requests.

Configuring enough I/O servers with the num_ioservers configuration parameter can greatly enhance the performance of queries for which prefetching of data can be used. To maximize the opportunity for parallel I/O, set num_ioservers to at least the number of physical disks in the database.

It is better to overestimate the number of I/O servers than to underestimate. If you specify extra I/O servers, these servers are not used, and their memory pages are paged out. As a result, performance does not suffer. Each I/O server process is numbered. The database manager always uses the lowest numbered process, so some of the upper numbered processes might never be used.

To estimate the number of I/O servers that you might need, consider the following:

- The number of database agents that could be writing prefetch requests to the I/O server queue concurrently.
- The highest degree to which the I/O servers can work in parallel.

Parallel I/O management

If multiple containers exist for a table space, the database manager can initiate parallel I/O, in which database manager uses multiple I/O servers to process the I/O requirements of a single query. Each I/O server processes the I/O workload for a separate container, so that several containers can be read in parallel. Performing I/O in parallel can result in significant improvements to I/O throughput.

Although a separate I/O server can handle the workload for each container, the actual number of I/O servers that can perform I/O in parallel is limited to the number of physical devices over which the requested data is spread. For this reason, you need as many I/O servers as physical devices.

Parallel I/O is initiated differently in the following cases:

- **Sequential prefetch:** For sequential prefetch, parallel I/O is initiated when the prefetch size is a multiple of the extent size for a table space. Each prefetch request is then broken into many small requests along the extent boundaries. These small requests are then assigned to different I/O servers.
- **List prefetch:** For list prefetch, each list of pages is divided into smaller lists according to the container in which the data pages are stored. These smaller lists are then assigned to different I/O servers.

- **Database or table space backup and restore:** For backing up or restoring data, the number of parallel I/O requests are equal to the backup buffer size divided by the extent size up to a maximum value equal to the number of containers.
- **Database or table space restore:** For restoring data, the parallel I/O requests are initiated and split the same way as that used for sequential prefetch. Instead of restoring the data into the buffer pool, the data is moved directly from the restore buffer to disk.
- **Load:** When you load data, you can specify the level of I/O parallelism with the LOAD command DISK_PARALLELISM option. If you do not specify this option, the database manager uses a default value based on the cumulative number of table space containers for all table spaces associated with the table.

For optimal performance of parallel I/O, ensure that:

- There are enough I/O servers. Specify slightly more I/O servers than the number of containers used for all table spaces within the database.
- The extent size and prefetch size are sensible for the table space. To prevent over-use of the buffer pool, prefetch size should not be too large. An ideal size is a multiple of the extent size and the number of table space containers. The extent size should be fairly small, with a good value being in the range of 8 to 32 pages.
- The containers reside on separate physical drives.
- All containers are the same size to ensure a consistent degree of parallelism. If one or more containers are smaller than the others, they reduce the potential for optimized parallel prefetch. Consider the following examples:
  - After a smaller container is filled, additional data is stored in the remaining containers, causing the containers to become unbalanced. Unbalanced containers reduce the performance of parallel prefetching, because the number of containers from which data can be prefetched might be less than the total number of containers.
  - If a smaller container is added at a later date and the data is rebalanced, the smaller container will contain less data than the other containers. Its small amount of data relative to the other containers will not optimize parallel prefetching.
  - If one container is larger and all of the other containers fill up, it is the only container to store additional data. The database manager cannot use parallel prefetch to access this additional data.

- There is adequate I/O capacity when using intra-partition parallelism. On SMP machines, intra-partition parallelism can reduce the elapsed time for query by running the query on multiple processors. Sufficient I/O capacity is required to keep each processor busy. Usually additional physical drives are required to provide the I/O capacity.

The prefetch size must be larger for prefetching to occur at higher rates and use I/O capacity effectively. The number of physical drives required depends on the speed and capacity of the drives and the I/O bus and on the speed of the processors.

Multipage file allocation

When using SMS table spaces, DB2 uses the operating system file system to allocate space. By default this space allocation is acquired in small increments that can introduce a great deal of overhead when insert activity for a table space is high. Multipage file allocation is used to improve insert performance for SMS table spaces. If enabled, all SMS table spaces are affected: there is no selection possible for individual SMS table spaces.

The default for the parameter is No — multipage file allocation is not enabled.

Following database creation, the parameter may be set to **Yes**, which indicates that multipage file allocation is enabled. This is done using the db2empfa tool. The db2empfa tool can be executed from a command prompt, with the name of the database to enable multipage file allocation as a parameter as follows:

```
DB2EMPFA SAMPLE
```

Important: Once set to **Yes**, the parameter cannot be changed back to **No**.

Logging

All changes to data pages are logged, and the updated data page is not written to disk storage before its associated log record is written to the log. Therefore, improving the logging performance is very important to improve the overall performance if your application performs frequent updates. The following affect the logging performance.

Log file placement

When a database is created, the recovery log file for it is created in a subdirectory of the directory containing the database. The default is a subdirectory named SQLOGDIR under the directory created for the database.

To change the location of the log, the database configuration parameter NEWLOGPATH can be set to the location where the log files are to be stored. The parameter can point to either a path name or to a raw device. If it points to a path name, it must be a fully qualified path name, not a relative path name.

The new setting does not take effect until both of the following occur:

- The database is in a consistent state, as indicated by the database_consistent parameter.
- All users are disconnected from the database.

When the first new connection is made to the database, the database manager will move the logs to the new location specified by logpath.

There might be log files in the old log path. These log files might not have been archived. You might need to archive these log files manually. Also, if you are running replication on this database, replication might still need the log files from before the log path change. If the database is configured with the User Exit Enable (userexit) database configuration parameter set to Yes, and if all the log files have been archived either by DB2 automatically or by yourself manually, then DB2 will be able to retrieve the log files to complete the replication process. Otherwise, you can copy the files from the old log path to the new log path.

Recommendation: Ideally, the log files will be on a physical disk which does not have high I/O. For instance, avoid putting the logs on the same disk as the operating system or high volume databases. This will allow for efficient logging activity with a minimum of overhead such as waiting for I/O.

Log buffer size

Buffering the log records will result in more efficient logging file I/O because the log records will be written to disk less frequently and more log records will be written at each time. The number of write operations is also higher if there is not enough space in the log buffer.

The log records are written to disk when one of the following occurs:

1. A transaction commits, or a group of transactions commits (MINCOMMIT).
2. The log buffer is full.
3. As a result of some other internal database manager event.

Log Buffer, whose size is determined by LOGBUFSZ database configuration parameter, holds log records in storage until they are written to disk. Ensure that the log buffer should be large enough to hold the amount of space needed to perform a rollback of current uncommitted transactions without having to read data from the log files.

When you set the log buffer size, consider the size of database heap (DBHEAP), since LOGBUFSZ allocated from the database heap (DBHEAP). Also note that using larger log buffers will not reduce data integrity, as data base commits will still force log file writes.

For OLTP workloads, it is recommended to increase the default log buffer (logbufsz).

MINCOMMIT database configuration parameter

By default, write operations to log files will be performed every time a transaction is committed. These write operations to log files can be reduced by grouping commits together. This will reduce the number of writes, thus improving the performance of the database, as a compromise with the response time of small transactions, since they have to wait for other transactions to commit their work.

If there are not enough transactions to commit their work, the database manager will commit transactions every second.

When transactions are short, the log I/O can become a bottleneck due to the frequency of the flushing of the log at COMMIT time. In such environments, setting the MINCOMMIT configuration parameter to a value greater than 1 can remove the bottleneck. When a value greater than 1 is used, the COMMITs for several transactions are held or batched. The first transaction to COMMIT waits until (MINCOMMIT - 1) more transactions COMMIT; and then the log is forced to disk and all transactions respond to their applications. The result is only 1 log I/O instead of several individual log I/Os.

By increasing MINCOMMIT and grouping the COMMITs, an increase in efficient file I/O logging will be achieved, as it will occur less frequently and write more log records each time a write is required. In order to avoid an excessive degradation in response time, each transaction only waits up to 1 second for the (MINCOMMIT- 1) other transactions to COMMIT. If the 1 second of time expires, the waiting transaction will force the log itself and respond to its application.

Changes to the value specified for this parameter take effect immediately; you do not have to wait until all applications disconnect from the database.

Increase or change in this parameter from its default value when there are multiple or write applications typically request concurrent database COMMITs. This will result in writing more log records each time it occurs. If you increase MINCOMMIT, you may also need to increase the LOGBUFSZ parameter to avoid having a log-full buffer force a write during these heavy load periods.

For OLTP workloads, it is recommended to increase MINCOMMIT to a value of 10 or more.

SOFTMAX database configuration parameter

The SOFTMAX database configuration parameter represents the percentage of LOGFILSIZ when a softcheck point will either write the log control file to disk or call an asynchronous page cleaner.

If you use very large log files, consider lowering this parameter, as this will reduce the time required to restart a database in the event of a crash recovery. However, lowering the value can increase the database logging overhead, which may impact performance. Lowering the value is not recommended if you have relatively small log files, as the checkpoint will be reached when these logs become full.

LOGFILSIZ database configuration parameter

The LOGFILSIZ parameter defines the size of each primary and secondary log file. Increase the LOGFILSIZ database configuration parameter if the database has a large number of update/delete/insert transactions, because these transactions cause the log file to become full very quickly, for example, OLTP workloads. A log file that is too small can affect system performance because of the overhead of archiving old log files, allocating new log files, and waiting for a usable log file.

Setting LOGFILESIZ to an excessively high value has negative impact, such as:

- Taking a long time to create the log files
- Reducing your flexibility when managing archived log files and copies of log files, since some media may not be able to hold an entire log file

If the disk space is scarce, the value of the LOGFILSIZ should be reduced, since primary logs are preallocated at this size.

5.3.5 Network

In this section we discuss some configuration parameters that improve communication between the client and the server.

Number of TCP/IP connection managers

When working in a client-server environment where TCP/IP is used for communication, one or more connection managers can be used to process communication requests. More than one connection manager can be created using DB2TCPCONNMGRS registry variable. You can create up to a maximum of 8 connection managers. If the value is not set, the default number of connection managers will be created. The default value is 1 on single processor machines.

For SMP machines, the default number of the TCP/IP connection managers is calculated using:

Square root (# of processors) rounded up to the maximum value of 8

The default value can be overridden when the DB2TCPCONNMGRS registry value is set; then the number of TCP/IP connection managers specified (to the maximum of 8) will be created.

If the DB2TCPCONNMGRS value is less than 1, then the DB2TCPCONNMGRS value is set to 1, and a warning message is logged that the value is out of range.

If the DB2TCPCONNMGRS value is greater than 8, then the DB2TCPCONNMGRS value is set to 8, and a warning message is logged that the value is out of range.

If the DB2TCPCONNMGRS values are between 1 and 8, then they are used as given.

When there is greater than one connection manager created, connection throughput should improve when multiple client connections are received simultaneously.

Blocking

Row blocking is a technique that reduces database manager overhead by retrieving a block of rows in a single operation. These rows are stored in a cache, and each fetch request in the application gets the next row from the cache. When all the rows in a block have been processed, another block of rows is retrieved by the database manager.

The cache is allocated when an application issues an OPEN CURSOR request and is deallocated when the cursor is closed. The size of the cache is determined by a configuration parameter which is used to allocate memory for the I/O block. The parameter used depends on whether the client is local or remote:

- For local applications, the parameter ASLHEAPSZ is used to allocate the cache for row blocking
- For remote applications, the parameter RQRIOBLK (client I/O block size) on the client workstation is used to allocate the cache for row blocking. The cache is allocated on the database client

For local applications, you can use the following formula to estimate how many rows are returned per block:

Rows per block = ASLHEAPSZ * 4096 / ORL

Here, ASLHEAPSZ is in pages of memory 4096 is the number of bytes per page, and ORL is the output row length in bytes.

For remote applications, you can use the following formula to estimate how many rows are returned per block:

Rows per block = RQRIOBLK / ORL

Here, RQRIOBLK is in bytes of memory, and ORL is the output row length in bytes.

Note that if you use the FETCH FIRST n ROWS ONLY clause or the OPTIMIZE FOR n ROWS clause in a SELECT statement, the number of rows per block will be the minimum of the following:

- The value calculated in the above formula
- The value of n in the FETCH FIRST clause
- The value of n in the OPTIMIZE FOR clause

Refer to the DB2 UDB V8 SQL Reference manual for more information about cursors.

Application support layer heap size (ASLHEAPSZ)

The application support layer heap size is the communication buffer between the local application and its associated agent. For local applications, the ASLHEAPSZ parameter is used to allocate cache for row blocking. This buffer is allocated as shared memory by each database manager agent that is started.

By default, this value is 15 (4 KB) pages. This value can contain an average request or reply between the application and its agent. Its size can be increased if queries retrieving large amounts of data.

If the request or reply do not fit into the buffer, they will be split into two or more send-and-receive buffers. This size should be set appropriately, so that it is able to handle the majority of requests using a single send-and-receive buffer. The size of the request is based on the storage required to hold:

- The input SQLDA
- All of the associated data in the SQLVARs
- The output SQLDA
- Other fields which do not generally exceed 250 bytes

This parameter is also used to determine the I/O block size when a blocking cursor is opened and allocated in the application's private address space. So, this is an additional parameter to be considered while allocating an optimal amount of private memory for each application program. If the database client cannot allocate space for a blocking cursor out of an application's private memory, then a non-blocking cursor will be opened.

The ASLHEAPSZ parameter is used to determine the initial size of the query heap for both local and remote clients. The maximum size of the query heap is defined by the QUERY_HEAP_SZ parameter. QUERY_HEAP_SZ is the set of contiguous memory allocated by the database manager to receive the data sent by the local application. The query heap is used to store each query in the agent's private memory.

The information for each query consists of:

- Input SQLDA
- Output SQLDA
- The statement text
- SQLCA
- The package name
- Creator
- Session number
- Consistency token

This query heap parameter is provided to ensure that an application does not consume unnecessarily large amount of virtual memory within an agent, and it initially set with the value equal to ASLHEAPSZ.

The formula to calculate the ASLHEAPSZ value in number of pages is as follows:

```
aslheapsz >=( sizeof(input SQLDA) +
sizeof (each input SQLVAR)+sizeof(output SQLDA)+ 250 ) / 4096
```

You can reduce the size of the ASLHEAPSZ value when:

- The application requests size is small.
- Application is running on a memory constrained system.

You can increase the size of the ASLHEAPSZ value when:

- The queries are very large and require more than one send and receive request.
- The application is not running on a memory constrained system.

Larger record blocks may also cause more fetch requests than are actually required by the application. Control the fetch requests by using the OPTIMIZE FOR clause on the SELECT statement in your application.

Client I/O block size (RQRIOBLK)

This parameter specifies the size of the communication buffer between remote applications and their database agents on the database server. The client I/O block of memory is used to serve the request and replies from and to remote applications. When a database client requests a connection to a remote database, this communication buffer is allocated on the client.

On the database server, a communication buffer of 32767 bytes is initially allocated, until a connection is established and the server can determine the value of RQRIOBLK at the client. Once the server knows the RQRIOBLK value, it will reallocate its communication buffer if the value on the client is not 32767 bytes. It is independent of the protocol used to connect the client and server.

In addition to the communication buffer, the RQRIOBLK parameter is also used to determine the I/O block size at the database client when a blocking cursor is opened. This memory for blocked cursors is allocated out of application's private address space, so you should determine the optimal amount of private memory to allocate for each application program. If the database client cannot allocate space for a blocking cursor out of an application's private memory, a non-blocking cursor will be opened.

You need to increase the value of RQRIOBLK for the following conditions or scenarios:

- A single SQL statement transmits large data (for example, large object data).
- The number or size of rows being transferred is large (for example, if the amount of data is greater than 4096 bytes). However, this will increase the size of the working set memory for each connection, a trade-off has to be done.
- Larger record blocks may also cause more fetch requests than are actually required by the application. Control the fetch requests by using the OPTIMIZE FOR clause on the SELECT statement in your application.

5.3.6 Other performance factors

In this section we discuss other key factors that can have a significant positive impact on the performance of your DB2 system. The items we cover are:

- Indexing
- Locking
- Bypass file system caching
- Space compression for tables

Indexing

An index is a list of the physical locations of rows sorted by the contents of one or more specified columns of the base table. You can create indexes in order to:

- Avoid unnecessary table scans
- Ensure uniqueness
- Provide ordering
- Avoid sorts
- Speed up frequently executed queries

- Speed up join predicates and support referential integrity Indexes can reduce access time significantly; however, indexes can also have adverse effects on performance. Before creating indexes, consider the effects of multiple indexes on disk space and processing time:
- Each index takes up a certain amount of storage or disk space. The exact amount is dependent on the size of the table and the size and number of columns included in the index.
- Each INSERT or DELETE operation performed on a table requires additional updating of each index on that table. This is also true for each UPDATE operation that changes an index key.

Advantages and disadvantages of indexes

Although the optimizer decides whether to use an index to access table data, except in the following case, you must decide which indexes might improve performance and create these indexes. Exceptions are the dimension block indexes and the composite block index that are created automatically for each dimension that you specify when you create a multi-dimensional clustering (MDC) table.

You must also execute the RUNSTATS utility to collect new statistics about the indexes in the following circumstances:

- After you create an index
- After you change the prefetch size

You should also execute the RUNSTATS utility at regular intervals to keep the statistics current. Without up-to-date statistics about indexes, the optimizer cannot determine the best data-access plan for queries.

Note: To determine whether an index is used in a specific package, use the SQL Explain facility. To plan indexes, use the Design Advisor Wizard as described in “Design Advisor Wizard” on page 148 in Chapter 3, “Post-installation tasks” of this book or db2advis tool to get advice about indexes that might be used by one or more SQL statements.

Advantages of an index over no index

If no index exists on a table, a table scan must be performed for each table referenced in a database query. The larger the table, the longer a table scan takes because a table scan requires each table row to be accessed sequentially. Although a table scan might be more efficient for a complex query that requires most of the rows in a table, for a query that returns only some table rows an index scan can access table rows more efficiently.

The optimizer chooses an index scan if the index columns are referenced in the SELECT statement and if the optimizer estimates that an index scan will be faster than a table scan. Index files generally are smaller and require less time to read than an entire table, particularly as tables grow larger. In addition, the entire index may not need to be scanned. The predicates that are applied to the index reduce the number of rows to be read from the data pages.

Each index entry contains a search-key value and a pointer to the row containing that value. If you specify the ALLOW REVERSE SCANS parameter in the CREATE INDEX statement, the values can be searched in both ascending and descending order. It is therefore possible to bracket the search, given the right predicate. An index can also be used to obtain rows in an ordered sequence, eliminating the need for the database manager to sort the rows after they are read from the table.

In addition to the search-key value and row pointer, an index can contain include columns, which are non-indexed columns in the indexed row. Such columns might make it possible for the optimizer to get required information only from the index, without accessing the table itself.

Note: The existence of an index on the table being queried does not guarantee an ordered result set. Only an ORDER BY clause ensures the order of a result set.

Although indexes can reduce access time significantly, they can also have adverse effects on performance. Before you create indexes, consider the effects of multiple indexes on disk space and processing time:

- Each index requires storage or disk space. The exact amount depends on the size of the table and the size and number of columns in the index.
- Each INSERT or DELETE operation performed on a table requires additional updating of each index on that table. This is also true for each UPDATE operation that changes the value of an index key.
- The LOAD utility rebuilds or appends to any existing indexes. The indexfreespace MODIFIED BY parameter can be specified on the LOAD command to override the index PCTFREE used when the index was created.
- Each index potentially adds an alternative access path for a query for the optimizer to consider, which increases the compilation time.

Choose indexes carefully to address the needs of the application program.

Index planning tips

The indexes that you create should depend on the data and the queries that access it.

The following guidelines can help you determine how to create useful indexes for various purposes:

- To avoid some sorts, define primary keys and unique keys, wherever possible, by using the CREATE UNIQUE INDEX statement.
- To improve data-retrieval, add INCLUDE columns to unique indexes. Good candidates are columns that:
  - Are accessed frequently and therefore would benefit from index-only access
  - Are not required to limit the range of index scans
  - Do not affect the ordering or uniqueness of the index key.
- To access small tables efficiently, use indexes to optimize frequent queries to tables with more than a few data pages, as recorded in the NPAGES column in the SYSCAT.TABLES catalog view. You should:
  - Create an index on any column you will use when joining tables.
  - Create an index on any column from which you will be searching for particular values on a regular basis.
- To search efficiently, decide between ascending and descending ordering of keys depending on the order that will be used most often. Although the values can be searched in reverse direction if you specify the ALLOW REVERSE SCANS parameter in the CREATE INDEX statement, scans in the specified index order perform slightly better than reverse scans.
- To save index maintenance costs and space:
  - Avoid creating indexes that are partial keys of other index keys on the columns. For example, if there is an index on columns a, b, and c, then a second index on columns a and b is not generally useful.
  - Do not create indexes arbitrarily on all columns. Unnecessary indexes not only use space, but also cause large prepare times. This is especially important for complex queries, when an optimization class with dynamic programming join enumeration is used.
- Use the following general rule for the typical number of indexes that you define for a table. This number is based on the primary use of your database:
  - For online transaction processing (OLTP) environments, create one or two indexes
  - For read-only query environments, you might create more than five indexes

  - For mixed query and OLTP environments, you might create between two and five indexes.

- To improve performance of delete and update operations on the parent table, create indexes on foreign keys when using referential integrity.
- For fast sort operations, create indexes on columns that are frequently used to sort the data.
- To improve join performance with a multiple-column index, if you have more than one choice for the first key column, use the column most often specified with the "=" (equijoin) predicate or the column with the greatest number of distinct values as the first key.
- To help keep newly inserted rows clustered according to an index and avoid page splits, define a clustering index. A clustering index should significantly reduce the need for reorganizing the table.

  Use the PCTFREE keyword when you define the table to specify how much free space should be left on the page to allow inserts to be placed appropriately on pages. You can also specify the pagefreespace MODIFIED BY clause of the LOAD command.
- To enable online index defragmentation, use the MINPCTUSED option when you create indexes. MINPCTUSED specifies the threshold for the minimum amount of used space on an index leaf page as well as enabling online index defragmentation. This might reduce the need for reorganization at the cost of a performance penalty during key deletions if these deletions physically remove keys from the index page.

Consider creating an index in the following circumstances:

- Create an index on columns that are used in WHERE clauses of the queries and transactions that are most frequently processed.

  The WHERE clause will generally benefit from an index on WORKDEPT, unless the WORKDEPT column contains many duplicate values:

```
WHERE WORKDEPT='A01' OR WORKDEPT='E21'
```

- Create an index on a column or columns to order the rows in the sequence required by the query. Ordering is required not only in the ORDER BY clause, but also by other features, such as the DISTINCT and GROUP BY clauses.

  The following example uses the DISTINCT clause:

```
SELECT DISTINCT WORKDEPT
FROM EMPLOYEE
```

The database manager can use an index defined for ascending or descending order on WORKDEPT to eliminate duplicate values. This same index could also be used to group values in the following example with a GROUP BY clause:

```
SELECT WORKDEPT, AVERAGE(SALARY)
FROM EMPLOYEE
GROUP BY WORKDEPT
```

- Create an index with a compound key that names each column referenced in a statement. When an index is specified in this way, data can be retrieved from the index only, which is more efficient than accessing the table.

  For example, consider the following SQL statement:

  ```
  SELECT LASTNAME
  FROM EMPLOYEE
  WHERE WORKDEPT IN ('A00','D11','D21')
  ```

  If an index is defined for the WORKDEPT and LASTNAME columns of the EMPLOYEE table, the statement might be processed more efficiently by scanning the index than by scanning the entire table. Since the predicate is on WORKDEPT, this column should be the first column of the index.

- Create an index with INCLUDE columns to improve the use of indexes on tables. Using the previous example, you could define a unique index as:

  ```
  CREATE UNIQUE INDEX x ON employee (workdept) INCLUDE (lastname)
  ```

  Specifying lastname as an INCLUDE column rather than as part of the index key means that lastname is stored only on the leaf pages of the index.

Index performance tips

Consider the following suggestions for using and managing indexes:

- **Specify parallelism at index creation:** When you create indexes on large tables hosted by an SMP machine, consider setting intra_parallel to YES (1) or SYSTEM (-1) to take advantage of parallel performance improvements. Multiple processors can be used to scan and sort data.

- **Specify separate table spaces for indexes:** Indexes can be stored in a different table space from the table data. This can allow for more efficient use of disk storage by reducing the movement of read/write heads during index access. You can also create index table spaces on faster physical devices. In addition, you can assign the index table space to a different buffer pool, which might keep the index pages in the buffer longer because they do not compete with table data pages.

  When you do not place indexes in separate table spaces, both data and index pages use the same extent size and prefetch quantity. If you use a different table space for indexes, you can select different values for all the characteristics of a table space.

Because indexes are usually smaller than tables and are spread over fewer containers, indexes often have smaller extent sizes, such as 8 and 16 pages. The SQL optimizer considers the speed of the device for a table space when it chooses an access plan.

Note: To specify separate table spaces for indexes, you must use database managed space (DMS).

- **Ensure the degree of clustering:** If your SQL statement requires ordering, such as ORDER BY, GROUP BY, and DISTINCT, even though an index might satisfy the ordering, the optimizer might not choose the index in the following cases:
  - Index clustering is poor. For information, examine the CLUSTERRATIO and CLUSTERFACTOR columns of SYSCAT.INDEXES.
  - The table is so small that it is cheaper to scan the table and sort the answer set in memory.
  - There are competing indexes for accessing the table.

  After you create a clustering index, perform a REORG TABLE in classic mode, which creates a perfectly organized index. To recluster the table, you might perform a sort and LOAD instead. In general, a table can only be clustered on one index. Build additional indexes after you build the clustering index. A clustering index attempts to maintain a particular order of data, improving the CLUSTERRATIO or CLUSTERFACTOR statistics collected by the RUNSTATS utility.

  To help maintain the clustering ratio, specify an appropriate PCTFREE when you alter a table before you load or reorganize that table. The free space on each page specified by PCTFREE provides space for inserts, so that these inserts can be clustered appropriately. If you do not specify PCTFREE for the table, reorganization eliminates all extra space.
- **Keep table and index statistics up-to-date:** After you create a new index, run the RUNSTATS utility to collect index statistics. These statistics allow the optimizer to determine whether using the index can improve access performance.
- **Enable online index defragmentation:** Online index defragmentation is enabled if the MINPCTUSED clause is set to greater than zero for the index. Online index defragmentation allows indexes to be compacted by merging leaf pages when the free space on a page falls at or below the specified level while the index remains available.

- **Reorganize indexes as necessary:** To get the best performance from your indexes, consider reorganizing your indexes periodically because updates to tables can cause index page prefetch to become less effective. To reorganize the index, either drop it and re-create it or use the REORG utility.

  To reduce the need for frequent reorganization, when you create an index specify an appropriate PCTFREE to leave a percentage of free space on each index leaf page as it is created. During future activity, records can be inserted into the index with less likelihood of causing index page splits. Page splits cause index pages not to be contiguous or sequential, which in turn results in decreased efficiency of index page prefetching.

  Dropping and re-creating or reorganizing the index also creates a new set of pages that are roughly contiguous and sequential and improves index page prefetch. Although more costly in time and resources, the REORG TABLE utility also ensures clustering of the data pages. Clustering has greater benefit for index scans that access a significant number of data pages.

  In a symmetric multi-processor (SMP) environment, if the intra_parallel database manager configuration parameter is YES or ANY, the “classic” REORG TABLE mode, which uses a shadow table for fast table reorganization, can use multiple processors to rebuild the indexes.

- **Analyze EXPLAIN information about index usage:** Periodically, run EXPLAIN on your most frequently used queries and verify that each of your indexes is used at least once. If an index is not used in any query, consider dropping that index.

  EXPLAIN information also lets you see if table scans on large tables are processed as the inner table of nested loop joins. If they are, an index on the join-predicate column is either missing or considered ineffective for applying the join predicate.

- **Use volatile tables for tables that vary widely in size:** A volatile table is a table that might vary in size at run time from empty to very large. For this kind of table, in which the cardinality varies greatly, the optimizer might generate an access plan that favors a table scan instead of an index scan.

  Declaring a table “volatile” using the ALTER TABLE...VOLATILE statement allows the optimizer to use an index scan on the volatile table. The optimizer will use an index scan instead of a table scan regardless of the statistics in the following circumstances:

  – All columns referenced are in the index
  – The index can apply a predicate in the index scan.

Index cleanup and maintenance

After you create indexes, performance can degrade when a large amount of insert/delete activity occurs unless you keep the index compact and organized. Consider the following suggestions to keep indexes as small and efficient as possible:

- Enable online index defragmentation. Create indexes with the MINPCTUSED clause. Drop and recreate existing indexes, if necessary.
- Perform frequent COMMITs or get X locks on tables, either explicitly or by lock escalation, if frequent COMMITS are not possible.

  Index keys marked deleted can be physically removed from the table after the COMMIT. X locks on tables allow the deleted key to be physically removed when it is marked deleted, as explained next.
- Use REORGCHK to help determine when to reorganize indexes or tables, or both, and when to use the REORG INDEXES with the CLEANUP ONLY option.

  To allow read and write access to the index during reorganization, run REORG INDEXES with the ALLOW WRITE ACCESS option.

> **Note:** In DB2 Version 8.1 and later, all new indexes are created as type-2 indexes. The one exception is when you add an index on a table that already has type-1 indexes. In this case only, the new index will also be a type-1 index. To find out what type of index exists for a table, execute the INSPECT command. To convert type-1 indexes to type-2 indexes, execute the REORG INDEXES command.

The primary advantages of type-2 indexes are as follows:

- An index can be created on columns whose length is greater than 255 bytes.
- The use of next-key locking is reduced to a minimum, which improves concurrency. Most next-key locking is eliminated because a key is marked deleted instead of being physically removed from the index page. For information about key locking, refer to topics that discuss the performance implications of locks.

Index keys that are marked deleted are cleaned up in the following circumstances:

- During subsequent insert, update, or delete activity:
  - During key insertion, keys that are marked deleted and are known to be committed are cleaned up if such a cleanup might avoid the need to perform a page split and prevent the index from increasing in size.

  - During key deletion, when all keys on a page have been marked deleted an attempt is made to find another index page where all the keys are marked deleted and all those deletions have committed. If such a page is found, it is deleted from the index tree.
  - If there is an X lock on the table when a key is deleted, the key is physically deleted instead of just being marked deleted. During this physical deletion, any deleted keys on the same page are also removed if they are marked deleted and known to be committed.
- When you execute the REORG INDEXES command with CLEANUP options:
  - The CLEANUP ONLY PAGES option searches for and frees index pages on which all keys are marked deleted and known to be committed.
  - The CLEANUP ONLY ALL option frees not only index pages on which all keys are marked deleted and known to be committed, but it also removes RIDs marked deleted and known to be committed on pages that contain some undeleted RIDs.
  - This option also tries to merge adjacent leaf pages if doing so results in a merged leaf page that has at least PCTFREE free space on the merged leaf page. The PCTFREE value is the percent of free space defined for the index when it is created. The default PCTFREE is ten percent. If two pages can be merged, one of the pages will be freed.
- During any rebuild of an index. Utilities that rebuild indexes include the following:
  - REORG INDEXES when not using one of the CLEANUP options
  - REORG TABLE when not using the INPLACE option
  - IMPORT with the REPLACE option
  - LOAD with the INDEXING MODE REBUILD option

Index reorganization

As tables are updated with deletes and inserts, index performance can degrade in the following ways:

- Fragmentation of leaf pages can occur. When leaf pages are fragmented, I/O costs increase because more leaf pages must be read to fetch table pages.
- The physical index page order no longer matches the sequence of keys on those pages, which is referred to as a badly clustered index. When leaf pages are badly clustered, sequential prefetching is inefficient and results in more I/O waits.
- The index develops more than its maximally efficient number of levels. In this case, the index should be reorganized.

If you set the MINPCTUSED parameter when you create an index, the database server automatically merges index leaf pages if a key is deleted and the free space is less than the specified percent. This process is called online index defragmentation. However, to restore index clustering, free space, and reduce leaf levels, you can use one of the following methods:

- Drop and recreate the index.
- Use the REORG INDEXES command to reorganize indexes online. You might choose this method in a production environment because it allows users to read from and write to the table while its indexes are being rebuilt.
- Use the REORG TABLE command with options that allow you to reorganize both the table and its indexes off-line.

Online index reorganization

When you use the REORG INDEXES command with the ALLOW WRITE ACCESS option, all indexes on the specified table are rebuilt while read and write access to the table is allowed. Changes made to the table during the reorganization process are applied after the indexes have been rebuilt. As a result, the rebuilt index might not be perfectly clustered. If PCTFREE is specified for an index, that percent of space is preserved on each page during reorganization.

Note: The CLEANUP ONLY option of the REORG INDEXES command does not fully reorganize indexes. The CLEANUP ONLY ALL option removes keys that are marked deleted and are known to be committed. It also frees pages in which all keys are marked deleted and are known to be committed. When pages are freed, adjacent leaf pages are merged if doing so can leave at least PCTFREE free space on the merged page. PCTFREE is the percentage of free space defined for the index when it is created. The CLEANUP ONLY PAGES option deletes only pages in which all keys are marked deleted and are known to be committed.

When you execute REORG INDEXES to reorganize an index, type-1 indexes are automatically converted to type-2 indexes, which use a method of deleting keys that avoids next-key locking and thus improves concurrency.

The REORG INDEXES command has the following requirements:

- SYSADM, SYSMAINT, SYSCTRL or DBADM authority, or CONTROL privilege on the indexes and table.
- An amount of free space in the table space where the indexes are stored equal to the current size of the index. Consider placing indexes subject to reorganization in a large table space when you issue the CREATE TABLE statement.

- Additional log space. REORG INDEXES logs its activity. As a result, the reorganization might fail, especially if the system is busy and other concurrent activity is logged.

Online index defragmentation

Online index defragmentation is enabled by the user-definable threshold for the maximum amount of free space on an index leaf page. When an index key is deleted from a leaf page and the threshold is exceeded, the neighboring index leaf pages are checked to determine if two leaf pages can be merged. If there is sufficient space on a page for a merge of two neighboring pages to take place, the merge occurs immediately in the background.

Online defragmentation is used when the MINPCTUSED value is set to less than one hundred (100). The recommended value for MINPCTUSED is less than 50 because the goal is to merge two neighboring index leaf pages. A value of zero for MINPCTUSED, which is also the default, disables online defragmentation.

Pages in the index are freed when the last index key on the page is removed. The exception to this rule occurs when you specify MINPCTUSED clause in the CREATE INDEX statement. The MINPCTUSED clause specifies a percent of space on an index leaf page. When an index key is deleted, if the percent of filled space on the page is at or below the specified value, then the database manager tries to merge the remaining keys with keys on an adjacent page. If there is sufficient space on an adjacent page, the merge is performed and an index leaf page is deleted.

Index non-leaf pages are not merged during an online index defragmentation. However, empty non-leaf pages are deleted and made available for re-use by other indexes on the same table. To free these non-leaf pages for other objects in a DMS storage model or to free disk space in an SMS storage model, perform a full reorganization of the table or indexes. Full reorganization of the table and indexes can make the index as small as possible. Index non-leaf pages are not merged during an online index defragmentation, but are deleted and freed for re-use if they become empty. The number of levels in the index and the number of leaf and non-leaf pages might be reduced.

For type-2 indexes, keys are removed from a page during key deletion only when there is an X lock on the table. During such an operation, online index defragmentation will be effective. However, if there is not an X lock on the table during key deletion, keys are marked deleted but are not physically removed from the index page. As a result, no defragmentation is attempted.

To defragment type-2 indexes in which keys are marked deleted, but remain in the physical index page, execute the REORG INDEXES command with the CLEANUP ONLY ALL option. The CLEANUP ONLY ALL option defragments the index, regardless of the value of MINPCTUSED.

If you execute REORG INDEXES with the CLEANUP ONLY ALL, two neighboring leaf pages are merged if such a merge can leave at least PCTFREE free space on the merged page. PCTFREE is specified at index creation time and defaults to ten percent.

Locking

To provide concurrency control and prevent uncontrolled data access, the database manager places locks on tables, table blocks, or table rows. A lock associates a database manager resource with an application, called the lock owner, to control how other applications can access the same resource.

Most problems associated with locking are application related issues. See the manua, *DB2 UDB V8 Administration: Performance*, in the chapter on Application Considerations, for details on application locking issues. In this section we discuss lock escalation issues.

The database manager uses record-level locking or table-level locking as appropriate based on:

- The isolation level specified at precompile time or when an application is bound to the database. The isolation level can be one of the following:
  - Uncommitted Read (UR)
  - Cursor Stability (CS)
  - Read Stability (RS)
  - Repeatable Read (RR)

  The different isolation levels are used to control access to uncommitted data, prevent lost updates, allow non-repeatable reads of data, and prevent phantom reads. Use the minimum isolation level that satisfies your application needs.
- The access plan selected by the optimizer. Table scans, index scans, and other methods of data access each require different types of access to the data.
- The LOCKSIZE attribute for the table. The LOCKSIZE clause on the ALTER TABLE statement indicates the granularity of the locks used when the table is accessed. The choices are either ROW for row locks, or TABLE for table locks. Use ALTER TABLE... LOCKSIZE TABLE for read-only tables. This reduces the number of locks required by database activity.
- The amount of memory devoted to locking. The amount of memory devoted to locking is controlled by the locklist database configuration parameter. If the lock list fills, performance can degrade due to lock escalations and reduced concurrency on shared objects in the database. If lock escalations occur frequently, increase the value of either locklist or maxlocks, or both.

In general, record-level locking is used unless one of the following is the case:

- The isolation level chosen is uncommitted read (UR).
- The isolation level chosen is repeatable read (RR) and the access plan requires a scan with no predicates.
- The table LOCKSIZE attribute is “TABLE”.
- The lock list fills, causing escalation.
- There is an explicit table lock acquired via the LOCK TABLE statement. The LOCK TABLE statement prevents concurrent application processes from either changing a table or using a table.

A lock escalation occurs when the number of locks held on rows and tables in the database equals the percentage of the locklist specified by the maxlocks database configuration parameter. Lock escalation might not affect the table that acquires the lock that triggers escalation. To reduce the number of locks to about half the number held when lock escalation begins, the database manager begins converting many small row locks to table locks for all active tables, beginning with any locks on large object (LOB) or long VARCHAR elements. An exclusive lock escalation is a lock escalation in which the table lock acquired is an exclusive lock. Lock escalations reduce concurrency. Conditions that might cause lock escalations should be avoided.

Correcting lock escalation problems

The database manager can automatically escalate locks from row or block level to table level. The maxlocks database configuration parameter specifies when lock escalation is triggered. The table that acquires the lock that triggers lock escalation might not be affected. Locks are first escalated for the table with the most locks, beginning with tables for which large object (LOBs) and LONG VARCHAR descriptors are locked, then the table with the next highest number of locks, and so on, until the number of locks held is decreased to about half of the value specified by maxlocks.

In a well-designed database, lock escalation rarely occurs. If lock escalation reduces concurrency to an unacceptable level, however, you need to analyze the problem and decide how to solve it.

Prerequisites

Ensure that lock escalation information is recorded. Set the database manager configuration parameter notifylevel to 3, which is the default, or to 4. At notifylevel of 2, only the error SQLCODE is reported. At notifylevel of 3 or 4, when lock escalation fails, information is recorded for the error SQLCODE and the table for which the escalation failed. The current SQL statement is logged only if it is a currently executing, dynamic SQL statement and notifylevel is set to 4.

Procedure

Follow these general steps to diagnose the cause of unacceptable lock escalations and apply a remedy:

1. Analyze in the administration notification log on all tables for which locks are escalated. This log file includes the following information:
   - The number of locks currently held
   - The number of locks needed before lock escalation is completed
   - The table identifier information and table name of each table being escalated
   - The number of non-table locks currently held
   - The new table level lock to be acquired as part of the escalation. Usually, an “S,” or Share lock, or an “X,” or eXclusive lock is acquired
   - The internal return code of the result of the acquisition of the new table lock level
2. Use the information in administration notification log to decide how to resolve the escalation problem. Consider the following possibilities:
   - Increase the number of locks allowed globally by increasing the value of the maxlocks or the locklist parameters, or both, in the database configuration file.

     You might choose this method if concurrent access to the table by other processes is most important. However, the overhead of obtaining record level locks can induce more delay to other processes than is saved by concurrent access to a table.
   - Adjust the process or processes that caused the escalation. For these processes, you might issue LOCK TABLE statements explicitly.
   - Change the degree of isolation. Note that this may lead to decreased concurrency, however.
   - Increase the frequency of commits to reduce the number of locks held at a given time.
   - Consider frequent COMMIT statements for transactions that require long VARCHAR or various kinds of large object (LOB) data. Although this kind of data is not retrieved from disk until the result set is materialized, the descriptor is locked when the data is first referenced. As a result, many more locks might be held than for rows that contain more ordinary kinds of data.

Bypass file system caching

In Windows NT and Windows 2000, by default files are cached in memory by the operating system when they are opened. 1 MB is reserved from a system pool for every 1 GB in the file. This caching is redundant with DB2's buffer management when the buffer pools are setup properly. This wastes memory that could be better utilized for additional DB2 buffer pools or sort heap. DB2 provides a registry variable to instruct it to open database files with the NOCACHE option.

If DB2NTNOCACHE is set to ON, file system caching is eliminated. If DB2NTNOCACHE is set to OFF, the operating system caches DB2 files. This applies to all data except for files that contain long fields or LOBs. To set this registry variable issue the following command from a command prompt:

```
DB2SET DB2NTNOCACHE=ON
```

Space compression for tables

There are two ways in which tables can occupy less space when stored on disk:

- If the column value is NULL, do not set aside the defined, fixed amount of space.
- If the column value can be easily known or determined (like default values) and if the value is available to the database manager during record formatting and column extraction.

DB2 UDB has an optional record format to allow for this type of space savings. Space savings can take place at the table level as well as at the column level.

Space compression for new tables

When creating a table, you can use the optional VALUE COMPRESSION clause to specify that the table is using the space saving row format at the table level and possibly at the column level.

When VALUE COMPRESSION is used, NULLs and zero-length data that has been assigned to defined variable-length data types (VARCHAR, VARGRAPHICS, LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, and DBCLOB) will not be stored on disk. Only overhead values associated with these data types will take up disk space.

If VALUE COMPRESSION is used then the optional COMPRESS SYSTEM DEFAULT option can also be used to further reduce disk space usage. Minimal disk space is used if the inserted or updated value is equal to the system default value for the data type of the column. The default value will not be stored on disk. Data types that support COMPRESS SYSTEM DEFAULT include all numerical type columns, fixed-length character, and fixed-length graphic string data types. This means that zeros and blanks can be compressed.

Space compression can have a significant impact on the amount of disk space required to store your data, if it contains a large number of any of the following:

- Nullable columns that contain null.
- VARCHAR, LONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, or DBCLOB columns with zero length.
- CHAR columns with all blanks
- DECIMAL, INTEGER, SMALLINT, BIGINT or FLOAT columns containing zeros.

Not only will space compression save on disk space, it has the potential to speed up queries that scan large amounts of data, since fewer data pages will have to be scanned due to the compression.

We highly recommend using space compression if your data meets the above criteria.

5.4 System optimization

In the first two sections of this chapter we laid the foundation for performance by discussing the primary performance factors for both system hardware, operating system software, and database software. In this section we discuss how to optimize these primary performance factors to configure your system for maximum database performance. We cover this for both the Windows 2000 operating system software and DB2 UDB V8.1.

5.4.1 Windows system optimization

At the beginning of this chapter we stated that the primary performance goals of a system hardware and software selection is to produce a well balanced system that can sustain high rates of overall system utilization. To achieve a well balanced system we must adequately size individual hardware and software resources so that no single resource results in a system bottleneck. In this section we look at how to optimize the primary windows performance factors for the role of a dedicated database server.

Tip: The Windows 2000 Server Resource Kit is an essential tool kit for Windows system optimization. It delivers in-depth operating system information and tools that enable you to understand, deploy, and make optimal use of their Windows 2000 operating systems. You should install the Windows 20000 Server Resource Kit as your first step to system optimization.

Memory optimization

The Windows 2000 family of server operating systems today scales physical memory from 4 GB with Windows 2000 Server, to 8 GB with Windows 2000 Advanced Server, all the way up to 64 GB with Windows 2000 Datacenter Server.

The primary goal of memory optimization in terms of the operating systems that will be dedicated as a database server is to maximize the amount of physical memory that will be available to the database server. This is accomplished by minimizing the use of memory by other applications such as file sharing and maximizing the total amount of memory available to the database server. These tools can help you optimize memory performance:

- Windows Task Manager
- Performance System Monitor
- Performance Logs and Alerts

Figure 5-6 shows the Windows System Monitor while benchmarking DB2 UDB on Windows 2000 Datacenter Server running on a 32-way Unisys ES7000 with 16 GB of physical memory. It is important to point out that the DB2 System Controller Service process (db2syscs.exe) has over 14 GB of committed memory.

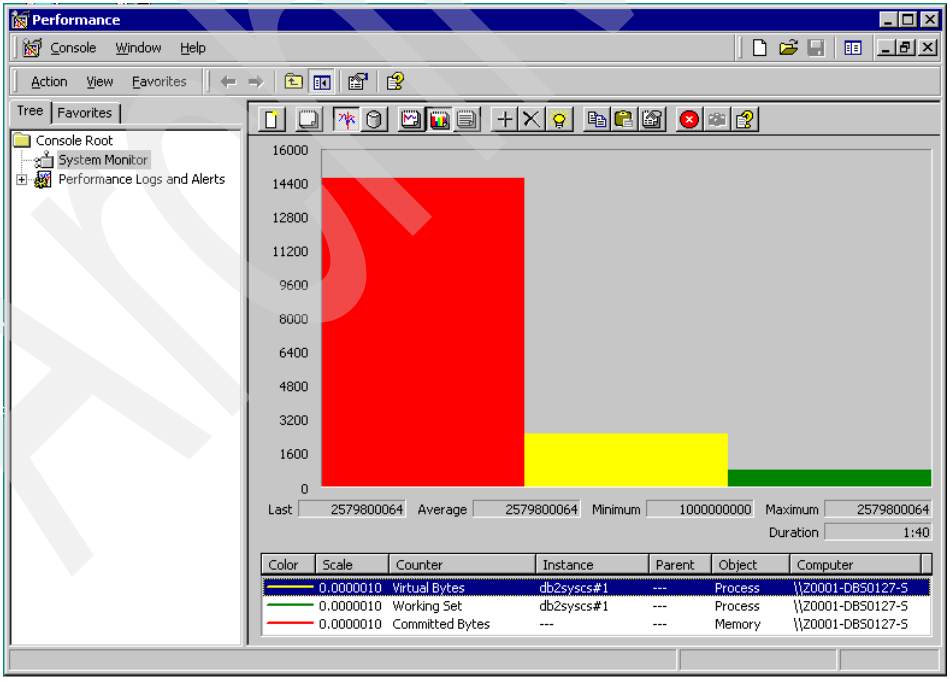

Figure 5-6 Performance System Monitor

Optimization system memory

The first order of business for memory optimization on a system that will function primarily as a database server is to minimize the use of memory by the Windows operating system for non-database related functions such as file caching. This can be accomplished by selecting the Maximize data throughput for network applications option found on the Server Optimization tab of the File and Printer Sharing for Microsoft Network properties dialog. See Figure 5-7.

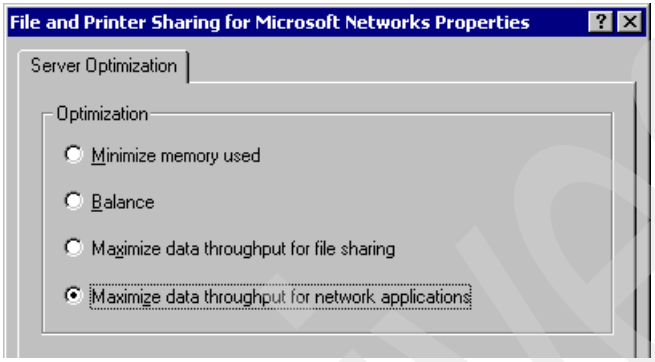

Figure 5-7 File and Printer Sharing for Microsoft Networks

Selecting this option has the same effect as using the Windows Registry Editor to change the value of the LargeSystemCache REG_DWORD to 0 on the Memory Management subkey.

Optimization system page file

Windows 2000 creates a single page file on the hard drive where the operating system was installed. On systems with less than 1 GB of physical memory the default page file is one and a half the size of the total physical memory. On systems with more that 1 GB of physical memory the default page file is 2 GB. On Windows 2000 the maximum size for a single page file is 4 GB.

For best performance, you should move the page file off the system drive and onto a dedicated page file drive. You can do this by selecting **`Start –> Settings –> Control Panel –> System –> Advanced –> Performance Options–>`** as shown in Figure 5-8.

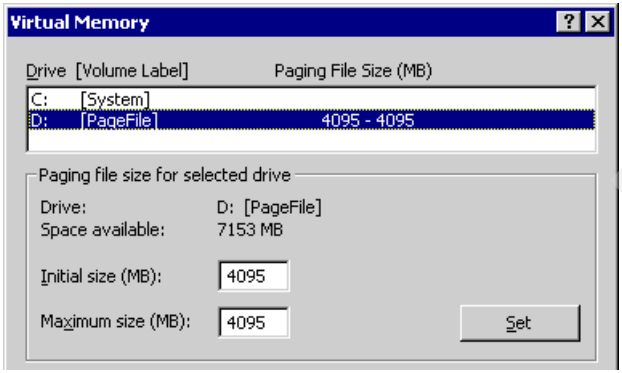

Figure 5-8 Windows Virtual Memory settings

If your system requires more than a 4 GB page file you have three options. First, you can create additional page files on additional physical hard drives. Second, you can create additional page files on a single physical hard drive that contains multiple logical partitions. Finally you can create multiple page file on a single physical drive with a single logical partition by modifying the memory management key in the Windows registry. To do this, open the Windows Registry Editor (regedt32.exe) select **HKEY_LOCAL_MACHINE -> System -> CurrentControlSet -> Control -> SessionManager -> MemoryManagement** and modify the memory management registry.

For example on a system with 8 GB of physical memory, you can create a 12 GB Windows page file key by adding additional page files in separate subdirectories on the same volume. First create the subdirectories, for example PageFile1, PageFile2, PageFile3. Second modify the memory management key using the Windows Registry Editor as show in Figure 5-9. Finally, reboot the system to allocate the multiple page files. If the original page file is not referenced in the changes made using the Windows Registry Editor then it should be deleted after the system reboots.

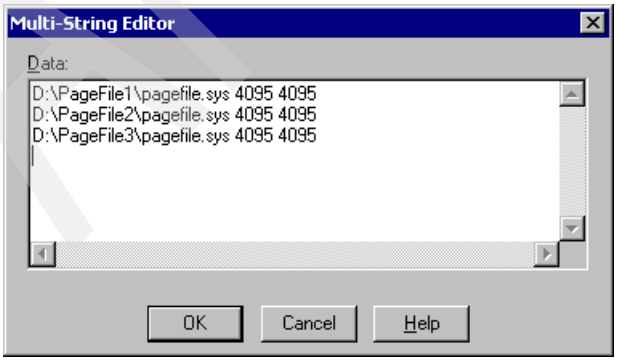

Figure 5-9 Enabling multiple page files

Warning: The Virtual Memory properties dialog does not support multiple page files. Multiple Page Files will be disabled if you open the Virtual Memory properties dialog and click **OK** to close it, and doing so will cause the system to use single page files.

Tuning 4 GB system memory

Windows 2000 Advanced Server and Datacenter Server both support 4 GB tuning. Microsoft introduced the concept of 4 GB tuning, with Windows NT 4.0 Enterprise Server (Service Pack 3). This memory tuning feature, which can be enabled with the /3 GB boot.ini switch, allows 32-bit applications that are aware of the /3 GB switch to increase there virtual address space by an additional 1 GB of memory for a total of 3 GB virtual address space. See Figure 5-10.

Figure 5-10 Windows 4 GB tuning

This memory optimization feature reduces the virtual address space of the Windows kernel to just 1 GB allowing memory intensive applications such as database servers to allocate an additional 1 GB of memory. After enabling 4 GB running you will be able to increase the database servers memory utilization up to 3 GB, allowing for much larger buffer pool, sort and utility heaps.

Note that although Windows 2000 Datacenter Server supports 4-GB Tuning, but only on systems with up to 16-GB of physical memory. This is because the Windows kernel requires a full 2 GB of virtual address space to support physical memory larger than 16 GB.

Extending beyond 4-GB limitations

The Microsoft Address Windowing Extensions (AWE) API set provides 32-bit windows applications the ability to address up to 64 GB of physical non-paged memory. Although the AWE API is available on all versions of Windows 2000, support is limited to 4 GB of memory with Windows 2000 Server. Its primary purpose on Windows 2000 Server is for development and testing of applications that use the AWE API.

The Address Windowing Extension (AWE) API is enabled using the /PAE boot.ini switch. PAE, which is a Intel acronym for Physical Address Extension as it is the Intel IA-32 architecture that provides the memory addressing capability, Microsoft simply offers the AWE API via a 32-bit PAE enabled kernel. See Figure 5-11.

Note that without the /PAE boot.ini switch Windows will not load the PAE enabled kernel and the total system memory reported by Task Manager will be about 4 GB regardless of how much physical memory is actually installed. This is because the Windows kernel itself uses the AWE API for memory management.

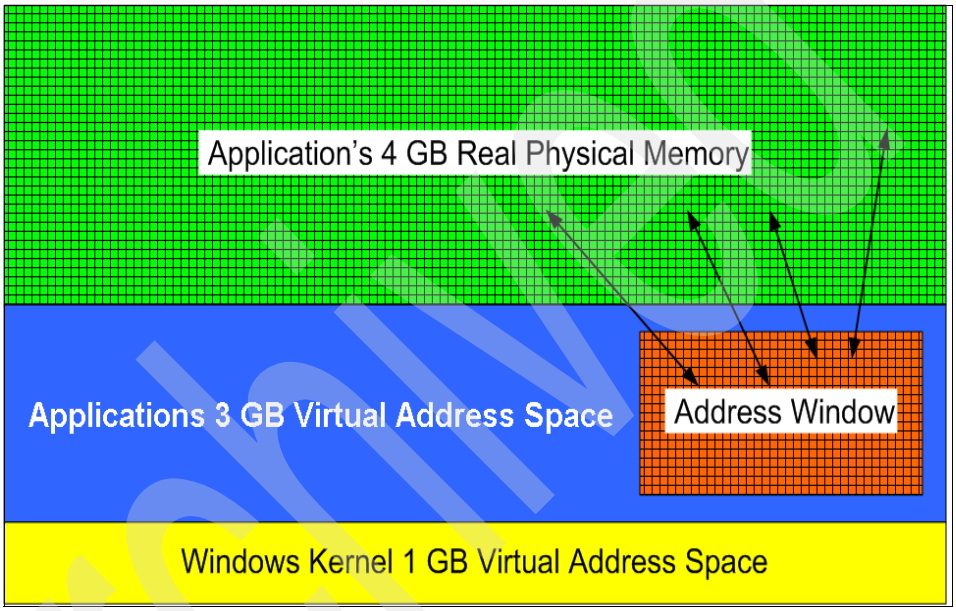

Figure 5-11 Windows Address Windowing Extension

It is important to note that not all software and/or drivers will function properly with the Windows PAE kernel on Windows 2000 Server or Windows 2000 Advanced Server. This is because AWE support is not required for certification on these versions of Windows, only on Windows 2000 Datacenter Server.

Table 5-3 Windows memory optimization summary

	Server	Advanced	Datacenter
/3GB		X	X (< 16GB)
/PAE	X	X	X

Tip: You can determine which boot.ini startup options are currently in effect by using the Windows Registry Editor and selecting **HKEY_LOCAL_MACHINE -> System -> CurrentControlSet -> Control -> SystemStartOptions.**

System memory metrics

The following Windows System Monitor performance metrics can be used to gauge your system performance in terms of memory utilization.

- Memory — Page Faults/sec
- Memory — Page Reads/sec
- Memory — Page Writes/sec
- Memory — Pages Input/sec
- Memory — Pages Output/sec
- Memory — Available Bytes
- Memory — Pool Nonpaged Bytes
- Process — Page Faults/sec
- Process — Working Set
- Process — Private Bytes
- Process — Page File Bytes

Processor optimization

The Windows 2000 family of server operating systems today scales from one to 4-way SMP servers with Windows 2000, from four to 8-way SMP servers with Windows 2000 Advanced Server, and all the way up to 32-way SMP servers with W2K Datacenter Server.

The primary goal of processor optimization in terms of the operating systems that will be dedicated as a database server is to maximize the amount of processors cycles that will be available to the database server. This is accomplished by minimizing the use of processing cycles by other applications. These tools can help you optimize processor performance:

- Windows Task Manager
- Windows System Monitor
- Performance Logs and Alerts
- Process Control Tool

Figure 5-12 shows a screen capture of the Windows Task Manager taken while benchmarking DB2 on Windows 2000 Datacenter Server running on a 32-way Unisys ES7000 with 16 GB of physical memory. It is important to note that the average CPU utilization is 91% and that over 14 GB of memory has been committed. Windows Task Manager also shows that the system is optimized as a database server as less than 100MB of memory is currently used for the Windows file system cache.

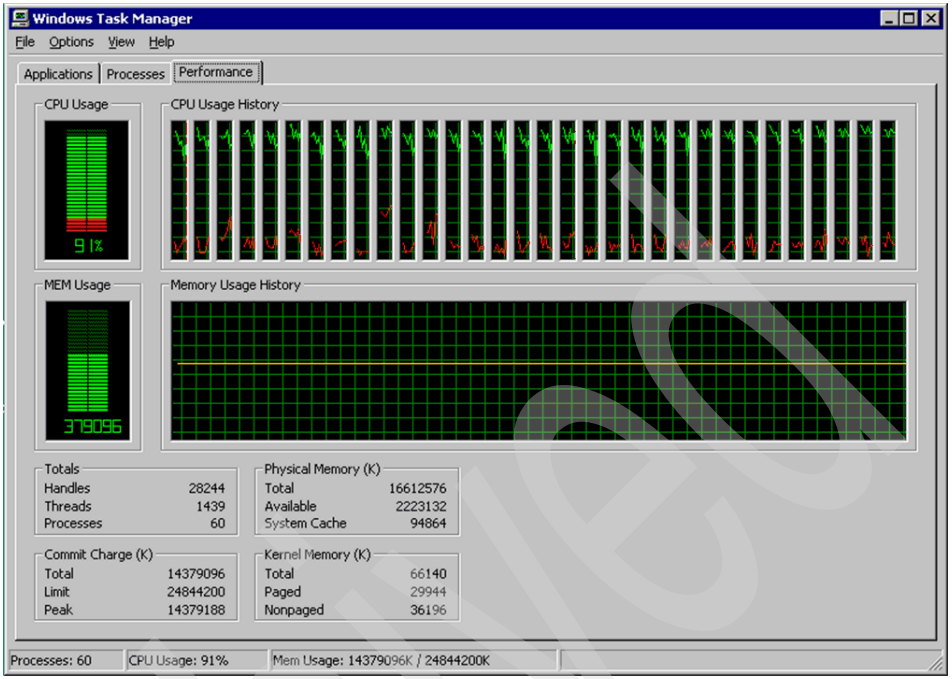

Figure 5-12 Windows Task Manager

Server process optimization

The first task at hand is to verify that the Windows operating system is optimized for background processes such as database servers. This is accomplished by selecting **`Start -> Settings -> Control Panel -> System -> Advanced -> Performance Options-> Background services`**. The background services option should already be set as this is the default setting on all Windows 2000 servers.

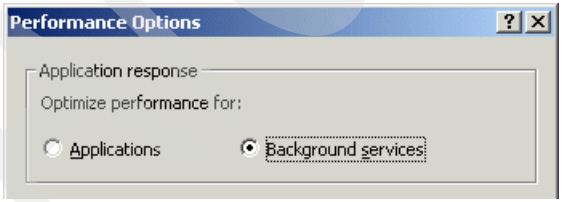

Figure 5-13 Windows Performance Options

Process Control Tool

Windows 2000 Servers extend the process model to include the concept of a job object. The purpose of the job object is to allow a process or group of processes to be managed. For example, without the Job Object Extensions most processes can modify scheduling priorities and/or processor affinity, a process defined within a Job Object cannot. In addition to the Job Object Extensions APIs that are included with all Windows 2000 server products, Windows 2000 Datacenter Server includes both a graphical user interface and command line interface into these APIs called the Process Control Tool. See Figure 5-14.

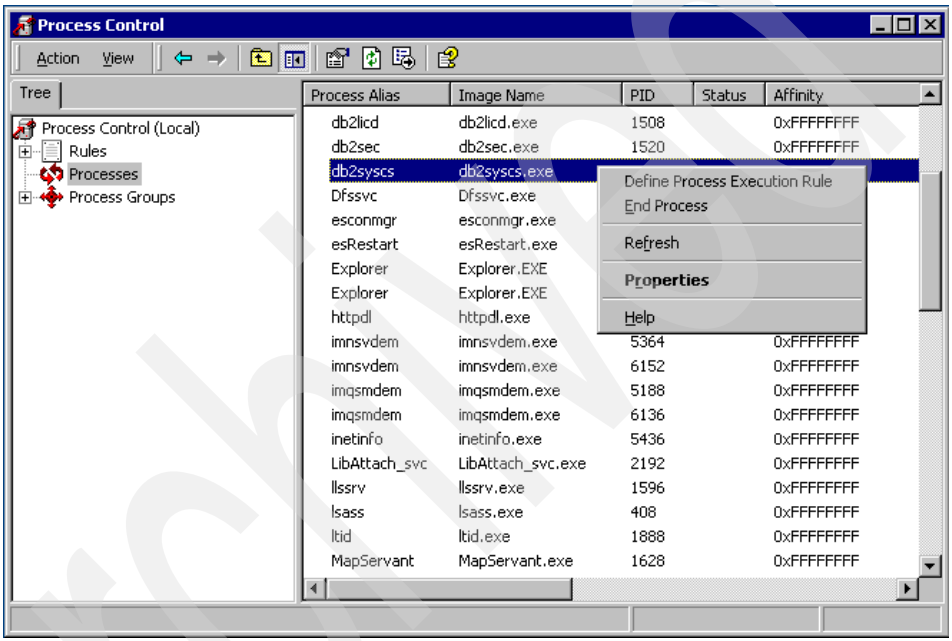

Figure 5-14 Process Control Tool

The Process Control Tool provides a graphical user interface into the Job Object Extension APIs. This system administration tool, located in the Administration Tools folder, can be used to control single processes or groups of processes as a single unit by limiting scheduling priority, processor affinity, processor time, and memory utilization. This tool only provides the system administrator with a graphical user interface into the JOE API. The process of actually managing job objects is performed by the Process Control service.

The Process Control tool can be used to manage the DB2 System Controller Service process (db2syscs.exe) as by default this process will run on all available processors on the system. On large SMP systems running more than one instance of the db2syscs.exe as in systems running multiple DB2 instances or database partitioned servers in a multiple logical node configuration setting processor affinity can increase overall system performance by increasing processor data locality and cache coherency.

System processor metrics

The following Windows System Monitor performance metrics can be used to gauge your system performance in terms of processor utilization.

- Processor — % Processor Time
- System — % Total Processor Time
- System — Processor Queue Length
- Process — % Privileged Time
- Process — % Processor Time
- Process — % User Time
- Process — Priority Base
- Thread — % Privileged Time
- Thread — % Processor Time
- Thread — % User Time
- Thread — Context Switches/sec
- Thread — Priority Base
- Thread — Priority Current
- Thread — Thread State

Storage optimization

The Windows 2000 family of server operating systems today scales from reliable and cost-effective internal ATA and SCSI disk storage, to high performance external SCSI and fibre channel attached storage, and all the way up to enterprise storage area networks.

The primary goal of disk optimization in terms of the operating systems that will be dedicated as a database server is to maximize I/O throughput. This can be accomplished by using multiple 64-bit 66 Mhz PCI disk controllers balanced among several PCI buses to drive a large number of physical disks (spindles) per disk array. These tools can help you optimize disk performance:

- Windows Task Manager
- Performance System Monitor
- Performance Logs & Alerts
- Windows Disk Management
- Windows Disk Defragmenter

Disk management

The primary tool for managing storage on Windows 2000 is Disk Management. This tool allows you to manage disks as they appear to the Windows operating system. In other words if you are using hardware implemented disk arrays that have been defined with the disk controllers setup utilities theses arrays will appear to the operating system as a single physical disk. Some third party vendors provide plugins that can be accessed directly from the Disk Management MMC.

Disk Management supports two types of disks, basic and dynamic. During the installation of Windows 2000 disks are automatically defined as basic and can subsequently be initialized as dynamic prior to defining any disk partitions. Windows 2000 support the use of both basic and dynamic disk within the same system, but you cannot mix and match these types of disk within a single volume. For example, you could not create a mirrored volume using a basic disk and a dynamic disk.

Basic disks follow a partition oriented scheme found in previous version of the Windows operating system. Basic disks partitions are created as primary or extended. Only four primary partitions can be defined on a physical disk and multiple logical drives can be defined inside an extended partition.

Dynamic disks are new with Windows 2000 and follow a volume oriented management scheme. Dynamic disks provide enhanced administration features that allow you to add disk and create, extend, or mirror volumes without restarting the operating system.

Volume management

The Disk Management tool in Windows 2000 allows you to create several different types of volumes on basic and dynamic disks. Basic volumes are created on basic disks and can be a primary or logical drive inside an extended partition. Basic volumes can be unformatted (raw) or formatted (cooked) with any of the file systems supported by Windows 2000. Support for basic volume management in Windows 2000 is limited to deleting volume, striped, mirror, or stripe sets with parity and breaking mirror sets.

On dynamic disks volumes can be created as simple, spanned, striped, mirrored, or RAID-5. Dynamic volumes can also be unformatted (raw) or formatted (cooked) with any of the file systems supported by Windows 2000.

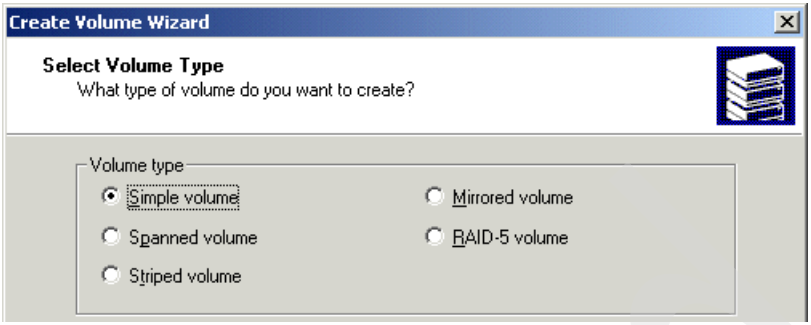

Figure 5-15 Disk Management — Create Volume Wizard

An unformatted volume can be created on either basic or dynamic disks (see Figure 5-16). These disks can have drive letters assignments, even though they are not formatted (see Figure 5-17). Direct disk access sometimes called raw I/O provides the best possible performance for database server as the Windows I/O Manager is completely bypassed for all database I/O requests to these disks. Direct disk access can be accomplished by referencing the physical disk number, the physical disk drive letter, and/or the physical disk GUID.

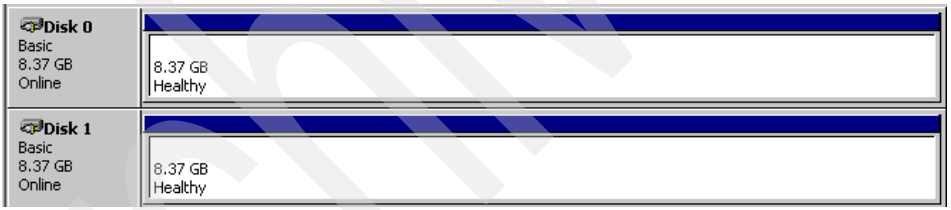

Figure 5-16 Disk Management — Unformatted (Raw) Disks

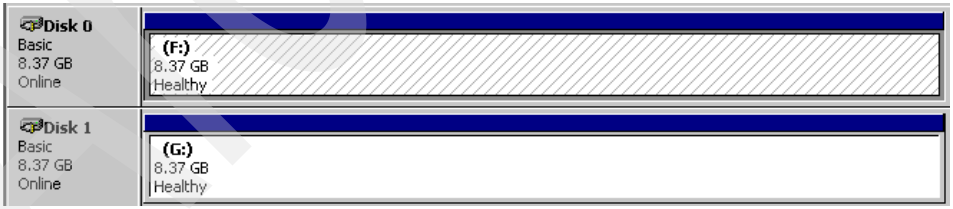

Figure 5-17 Disk Management — Unformatted (Raw) Disks with Drive Letters

Simple volume: A simple volume can only be defined on a single physical disk. It can be extend but only within the same physical disk. You can extend a simple volume across multiple disks and it will be converted to a spanned volume. A spanned volume is a volume that is made up of two or more physical disks. Spanned volumes ease disk administration because you can increase the volume size by simply adding more physical disks to the volume.

Striped volume: A striped volume can be defined on two or more physical disks. I/O requests against a striped volume alternate among the physical disks defined within the volume and can improve I/O performance as multiple disks (spindles) are used to access data within the volume. A striped volume is a software implementation of RAID level 0 and provides no data redundancy.

Mirrored volume: A mirrored volume can be defined on two physical disks. It provides redundancy as data is mirrored on both disks in the volume. If one of the physical disks fails, the data on the failed disk becomes unavailable, but the system continues to operate using the unaffected disk. You can break a mirrored volume which will result in two simple volumes. A mirrored volume is a software implementation of RAID level 1.

RAID-5 volume: A RAID-5 volume can be defined on three or more physical disks. It provides redundancy for data by striping with parity information. A RAID-5 volume is a software implementation of RAID level 1.

Mounted volume: You can use Disk Management to mount a local drive at any empty folder on a local NTFS volume. Prior to DB2 UDB V8.1 using mounted volumes was the only way to configure dual logging on physically separate disks. See Figure 5-18. You can also use the Mount Volume (mountvol.exe) system command to mount volumes.

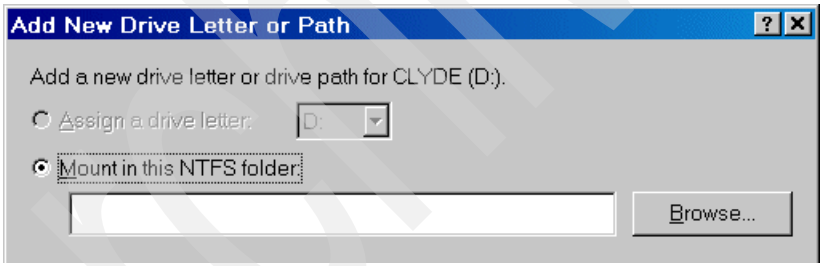

Figure 5-18 Using Disk Management to mount volumes

File system optimization

NTFS is the native file system for Windows 2000 operating systems. This file system is required to support large file systems and has several characteristics that make it more reliable and secure than other files systems supported by the Windows 2000 operating system. You can improve the overall performance of the NTFS file system by disabling some functionality that may not be required for a operating systems that will support a dedicated database server.

Cluster Size: On Windows 2000, the allocation unit size, sometimes called cluster size, can range from 512 bytes to 64k bytes. The allocation unit size removes any dependence on physical sector sizes allowing the NTFS to support for very large physical disks. The allocation unit size is defined when a volume is formatted and cannot be changed without re-formatting the volume. In general, larger allocation unit sizes will provide better disk space allocation performance and reduce disk fragmentation.

Disable Compression: On Windows 2000, NTSF supports file and folder compression on both basic and dynamic disks formatted with an cluster or allocation unit size of 4096 bytes or smaller. File compression requires additional overhead, as files must constantly be decompressed and recompressed, even when copied from one drive to another inside the system. In general, the compression overhead on Windows 2000 servers can result in a substantial performance degradation for database applications that perform significant I/O. You should avoid using file and folder compression on drives where you plan to place your database files.

Disable File Recycling: On Windows 2000, the Recycle Bin by default is configured with a global setting to allocate up to 10% of each drive for recycling. The recycle bin creates a hidden system folder on each drive and as files are deleted through Windows Explorer they are first moved to the Recycle Bin. Once the recycle bin becomes full or you empty the Recycle Bin the files are actually deleted. In general you should consider changing the default Recycle Bin settings to either significantly reduce the amount of space reserved or completely disable the recycling of files on drive that you will use to place database files. This will prevent transient files, such as database load files or your DBA's MP3 collection from consuming excessive disk space on your database server.

Disable 8.3 Name: On Windows 2000, the NTFS file system by default supports the 8.3 file naming convention to provide backward compatibility with MS-DOS and Windows 3.x client operating systems. As files are created on NTFS file system, the 8.3 name is by default always generating. You can disable this support, as it is unlikely that these types of clients will be accessing a database file system directly. Use the Windows Registry Editor. Select **HKEY_LOCAL_MACHINE -> SYSTEM -> CurrentControlSet -> Control -> Filesystem** and change the value of the NtfsDisable8dot3NameCreation key to 1.

Disable Last Access: The NTFS file system by default updates the last access date and time of each directory it traverses. This can have a direct negative effect of performance for large NTFS files systems. You can improve file system performance by disabling the updating of the directories last access. You can do this using the Windows Registry Editor by selecting **HKEY_LOCAL_MACHINE -> SYSTEM -> CurrentControlSet -> Control -> Filesystem** and adding the NtfsDisableLastAccessUpdate REG_DWORD with a value of 1.

Disk Metrics

The following Windows System Monitor performance metrics can be used to gauge your system performance in terms of disk utilization.

- PhysicalDisk — % Disk Time
- PhysicalDisk — Avg. Disk Queue Length
- PhysicalDisk — Current Disk Queue Length
- PhysicalDisk — Avg. Disk Sec/Read
- PhysicalDisk — Avg. Disk Sec/Write
- PhysicalDisk — Disk Read Bytes/sec
- PhysicalDisk — Disk Write Bytes/sec
- PhysicalDisk — Avg. Disk Bytes/Read
- PhysicalDisk — Avg. Disk Bytes/Write
- PhysicalDisk — Disk Reads/sec
- PhysicalDisk — Disk Writes/sec

Tip: The Windows Task Manager continuously collects metrics for process I/O requests. On large SMP systems running disk intensive I/O operations such as database servers, the collection of these metrics can impose considerable overhead on the system. You can improve performance by disabling I/O metric collection. The best way to do this is to use the Windows 2000 Server Resource Kit tool called Extensible Performance Counter List (exctrlst.exe).

Network optimization

The Windows 2000 family of server operating systems today supports a wide variety of network adapters from 10 Mbps to 1 Gbps over copper and fiber.

The primary goal of network optimization in terms of the operating systems that will be dedicated as a database server is to maximize I/O throughput. For optimal performance, the network adapters in a file server should be configured in full-duplex mode and connected to a switch. Use multiple network adapters and/or network adapters that are capable of taking advantage of the advanced TCP/IP off loading features built into Windows 2000. These tools can help you optimize disk performance:

- Windows System Monitor
- Performance Logs and Alerts
- Network Monitor

What's new with Windows Server 2003: Windows Server 2003 provides a Networking tab on Task Manager that can be used to view overall network utilization on individual network adapters installed on the system.

network Metrics

The following network metrics can be used to gauge your system performance in terms of network utilization.

- Network Interface — Bytes Total/sec
- Network Interface — Bytes Sent/sec
- Network Interface — Bytes Received/sec
- Protocol_layer_object — Segments Received/sec
- Protocol_layer_object — Segments Sent/sec
- Protocol_layer_object — Frames Sent/sec
- Protocol_layer_object — Frames Received/sec
- Server — Bytes Total/sec
- Server — Bytes Received/sec
- Server — Bytes Sent/sec
- Network Segment — % Network Utilization

Note: You must install the Network Monitor Driver in order to collect performance data using the Network Segment object counters.

5.4.2 DB2 system optimization

In this section we discuss additional DB2 specific items to assist in system optimization. We cover the following topics:

- Runstats utility
- Reorganization utility
- DB2 snapshot
- SQL Explain tools

Runstats utility

Statistical data stored in the system catalogs helps the optimizer choose the best access plan for queries. Make sure that you execute the RUNSTATS utility to update this statistical data:

- At frequent regular intervals for tables whose contents changes continually.
- After each operation that adds or changes data in a significant number of table rows. Such operations include batch updates and data loading that adds rows.

When the SQL compiler optimizes SQL query plans, its decisions are heavily influenced by statistical information about the size of the database tables and indexes. The optimizer also uses information about the distribution of data in specific columns of tables and indexes if these columns are used to select rows or join tables.

The optimizer uses this information to estimate the costs of alternative access plans for each query. When significant numbers of table rows are added or removed, or if data in columns for which you collect statistics is updated, execute RUNSTATS again to update the statistics.

Statistical information is collected for specific tables and indexes in the local database when you execute the RUNSTATS utility. The collected statistics are stored in the system catalog tables.

In addition to table size and data distribution information, you can also collect statistical information about the cluster ratio of indexes, the number of leaf pages in indexes, the number of table rows that overflow their original pages, and the number of filled and empty pages in a table. You use this information to decide when to reorganize tables and indexes.

Consider these tips to improve the efficiency of RUNSTATS and the usefulness of the collected statistics:

- Collect statistics only for the columns used to join tables or in the WHERE, GROUP BY, and similar clauses of queries. If these columns are indexed, you can specify the columns with the ONLY ON KEY COLUMNS clause for the RUNSTATS command.
- Customize the values for num_freqvalues and num_quantiles for specific tables and specific columns in tables.
- Collect DETAILED index statistics with the SAMPLE DETAILED clause to reduce the amount of background calculation performed for detailed index statistics. The SAMPLE DETAILED clause reduces the time required to collect statistics, and produces adequate precision in most cases.
- When you create an index for a populated table, add the COLLECT STATISTICS clause to create statistics as the index is created.

Distribution statistics are not collected:

- When the num_freqvalues and num_quantiles configuration parameters are set to zero (0)
- When the distribution of data is known, such as when each data value is unique.
- When the column is a data type for which statistics are never collected. These data type are LONG, large object (LOB), or structured columns.
- For row types in sub-tables, the table level statistics NPAGES, FPAGES, and OVERFLOW are not collected.
- If quantile distributions are requested, but there is only one non-NULL value in the column

- For extended indexes like those used by the DB2 Spatial Extender or declared temporary tables

Guidelines for collecting and updating statistics

The RUNSTATS command collects statistics on both the table and the index data to provide the optimizer with accurate information for access plan selection.

Use the RUNSTATS utility to collect statistics in the following situations:

- When data has been loaded into a table and the appropriate indexes have been created.
- When you create a new index on a table. You need execute RUNSTATS for only the new index if the table has not been modified since you last ran RUNSTATS on it.
- When a table has been reorganized with the REORG utility.
- When the table and its indexes have been extensively updated, by data modifications, deletions, and insertions (“extensive” in this case may mean that 10 to 20 percent of the table and index data has been affected).
- Before binding application programs whose performance is critical
- When you want to compare current and previous statistics. If you update statistics at regular intervals you can discover performance problems early.
- When the prefetch quantity is changed.

To improve RUNSTATS performance and save disk space used to store statistics, consider specifying only the columns for which data distribution statistics should be collected.

Ideally, you should rebind application programs after running statistics. The query optimizer might choose a different access plan if it has new statistics.

Collecting catalog statistics

You collect catalog statistics on tables and indexes to provide information that the optimizer uses to choose the best access plans for queries.

Prerequisites

You must connect to the database that contains the tables and indexes and have one of the following authorization levels:

- SYSADM
- SYSCTRL
- SYSMAINT
- DBADM
- CONTROL privilege on the table

Procedure

To collect catalog statistics:

1. Connect to the database that contains the tables and indexes for which you want to collect statistical information.
2. From the DB2 command line, execute the RUNSTATS command with appropriate options. These options allow you to tailor the statistics that are collected for the queries that run against the tables and indexes.

   For a complete list of RUNSTATS options, refer to the DB2 UDB V8 Command Reference manual.
3. When RUNSTATS is complete, issue a COMMIT statement to release locks.
4. Rebind packages that access tables and indexes for which you have regenerated statistical information.

To use a graphical user interface to specify options and collect statistics, use the Control Center.

Collecting distribution statistics for specific columns

For efficiency of both RUNSTATS and subsequent query-plan analysis, you might collect distribution statistics on only the table columns that queries use in WHERE, GROUP BY, and similar clauses. You might also collect cardinality statistics on combined groups of columns. The optimizer uses such information to detect column correlation when it estimates selectivity for queries that reference the columns in the group.

In the following steps, the database is assumed to be sales and to contain the table customers, with indexes custidx1 and custidx2.

Prerequisites

You must connect to the database that contains the tables and indexes and have one of the following authorization levels:

- sysadm
- sysctrl
- sysmaint
- dbadm
- CONTROL privilege on the table

Procedure

To collect statistics on specific columns:

1. Connect to the sales database.
2. Execute one of the following commands at the DB2 command line, depending on your requirements:

– To collect distribution statistics on columns zip and ytdtotal:

  RUNSTATS ON TABLE sales.customers
  WITH DISTRIBUTION ON COLUMNS (zip, ytdtotal)

– To collect distribution statistics on the same columns, but adjust the distribution defaults:

  RUNSTATS ON TABLE sales.customers
  WITH DISTRIBUTION ON
  COLUMNS (zip, ytdtotal NUM_FREQVALUES 50 NUM_QUANTILES 75)

– To collect distribution statistics on the columns indexed in custidx1 and custidx2:

  RUNSTATS ON TABLE sales.customer
  ON KEY COLUMNS

– To collect column statistics on the table only for specific columns zip and ytdtotal and a column group that includes region and territory:

  RUNSTATS ON TABLE sales.customers
  ON COLUMNS (zip, (region, territory), ytdtotal)

You can also use the Control Center to collect distribution statistics.

Collecting index statistics

Collect index statistics to allow the optimizer to evaluate whether an index should be used to resolve a query.

In the following steps, the database is assumed be sales and to contain the table customers, with indexes custidx1 and custidx2.

Prerequisites

You must connect to the database that contains the tables and indexes and have one of the following authorization levels:

- SYSADM
- SYSCTRL
- SYSMAINT
- DBADM
- CONTROL privilege on the table

Executing RUNSTATS with the SAMPLED DETAILED option requires 2MB of the statistics heap. Allocate an additional 488 4K pages to the stat_heap_sz database configuration parameter setting for this additional memory requirement. If the heap appears to be too small, RUNSTATS returns an error before it attempts to collect statistics.

Procedure

To collect detailed statistics for an index:

1. Connect to the sales database.
2. Execute one of the following commands at the DB2 command line, depending on your requirements:
   - To create detailed statistics on both custidx1 and custidx2:

     RUNSTATS ON TABLE sales.customers AND DETAILED INDEXES ALL
   - To create detailed statistics on both indexes, but use sampling instead of performing detailed calculations for each index entry:

     RUNSTATS ON TABLE sales.customers AND SAMPLED DETAILED INDEXES ALL
   - To create detailed sampled statistics on indexes as well as distribution statistics for the table so that index and table statistics are consistent:

     RUNSTATS ON TABLE sales.customers
     WITH DISTRIBUTION ON KEY COLUMNS
     AND SAMPLED DETAILED INDEXES ALL

You can also use the Control Center to collect index and table statistics.

Reorganization utility

After many changes to table data, logically sequential data may be on non-sequential physical data pages so that the database manager must perform additional read operations to access data. Additional read operations are also required if a significant number of rows have been deleted. In such a case, you might consider reorganizing the table to match the index and to reclaim space. You can reorganize the system catalog tables as well as database tables.

Note: Because reorganizing a table usually takes more time than running statistics, you might execute RUNSTATS as discussed earlier in this section under “Runstats utility” on page 281 to refresh the current statistics for your data and rebind your applications. If refreshed statistics do not improve performance, reorganization might help. For detailed information about the options and behavior of the REORG TABLE utility, refer to the DB2 UDB V8 Command Reference manual.

Consider the following factors, which might indicate that you should reorganize a table:

- A high volume of insert, update, and delete activity on tables accessed by queries

- Significant changes in the performance of queries that use an index with a high cluster ratio
- Executing RUNSTATS to refresh statistical information does not improve performance
- The REORGCHK command indicates a need to reorganize your table
- The trade-off between the cost of increasing degradation of query performance and the cost of reorganizing your table, which includes the CPU time, the elapsed time, and the reduced concurrency resulting from the REORG utility locking the table until the reorganization is complete.

Reducing the need to reorganize tables

To reduce the need for reorganizing a table, perform these tasks after you create the table:

- Alter table to add PCTFREE
- Create clustering index with PCTFREE on index
- Sort the data
- Load the data

After you have performed these tasks, the table with its clustering index and the setting of PCTFREE on table helps preserve the original sorted order. If enough space is allowed in table pages, new data can be inserted on the correct pages to maintain the clustering characteristics of the index. As more data is inserted and the pages of the table become full, records are appended to the end of the table so that the table gradually becomes unclustered.

If you perform a REORG TABLE or a sort and LOAD after you create a clustering index, the index attempts to maintain a particular order of data, which improves the CLUSTERRATIO or CLUSTERFACTOR statistics collected by the RUNSTATS utility.

Note: Creating multi-dimensional clustering (MDC) tables might reduce the need to reorganize tables. For MDC tables, clustering is maintained on the columns that you specify as arguments to the ORGANIZE BY DIMENSIONS clause of the CREATE TABLE statement. However, REORGCHK might recommend reorganization of an MDC table if it considers that there are too many unused blocks or that blocks should be compacted.

Determining when to reorganize tables

To determine how performance is related to changes in database tables and indexes, examine the statistics collected by RUNSTATS. Among other information, the statistics show the data distribution within a table, the number of used and empty pages, and RIDs marked deleted in index leaf pages. Statistics also provide information about prefetch efficiency. If you run RUNSTATS regularly

and analyze the statistics over a period of time, you can identify performance trends.

In particular, analyzing the statistics produced by RUNSTATS can indicate when and what kind of reorganization is necessary.

Procedure

To determine whether you need to reorganize tables, query the catalog statistics tables and monitor the following statistics:

- **Overflow of rows:** Query the OVERFLOW column in the SYSSTAT.TABLES table to monitor the overflow number. This column stores the number of rows that do not fit on their original pages. Row data can overflow when VARCHAR columns are updated with longer values. In such cases, a pointer is kept at the original location in the row original location. This can hurt degrades performance because the database manager must follow the pointer to find the contents of the row. This two-step process increases the processing time and might also increase the number of I/Os.

  As the number of overflow rows increases, the potential benefit of reorganizing your table data also increases. Reorganizing the table data will eliminate the overflow for rows.

- **Fetch statistics:** Query the three following columns in the SYSCAT.INDEXES and SYSSTAT.INDEXES catalog statistics tables to determine the effectiveness of the prefetchers when the table is accessed in index order. These statistics characterize the average performance of the prefetchers against the underlying table:

  - The AVERAGE_SEQUENCE_FETCH_PAGES column stores the average number of adjoining pages in sequence in the table. These pages would be eligible for prefetching. A small number indicates that the prefetchers cannot be effective because they never achieve the full prefetching size configured for the table space. A large number indicates that the prefetchers are performing effectively. This number should approach NPAGES for a clustered index and table.

  - The AVERAGE_RANDOM_FETCH_PAGES column stores the average number of pages that must be accessed randomly while scanning pages that are primarily in sequence. The prefetchers ignore small numbers of random pages when most pages are in sequence, and continue to prefetch to the configured prefetch size. As the number of random fetch pages increases, the table is becoming more disorganized. Such disorganization is usually caused by inserts that occur out of sequence, either at the end of the table or in overflow pages. This causes fetches that slow query performance when the index is used to access a range of values.

- The AVERAGE_SEQUENCE_FETCH_GAP column stores the average number of pages that interrupted prefetching. These occur when many pages are accessed randomly, which interrupts the prefetchers. A large number indicates a table that is disorganized or poorly clustered to the index. A small number indicates a clustered index and table.

- **Number of index leaf pages with RIDs marked deleted but not removed:** In type-2 indexes, RIDs are not usually physically deleted when the RID is marked deleted. In such indexes, useful space might be occupied by these logically deleted RIDs. Query the NUM_EMPTY_LEAFS column of the SYSCAT.INDEXES and SYSSTAT.INDEXES statistics tables to retrieve the number of leaf pages that contain on which all RIDS are logically deleted. For leaf pages on which not all RIDs are marked deleted, the total number of logically deleted RIDs is stored in the NUMRIDS_DELETED column. Use this information to estimate how much space might be reclaimed by executing REORG INDEXES with the CLEANUP ALL option. To reclaim only the space in pages on which all RIDs are marked deleted, execute REORG INDEXES with the CLEANUP ONLY PAGES options.

- **Cluster-ratio and cluster-factor statistics for indexes:** A cluster-ratio statistic between 0 to 100 is stored in the CLUSTERRATIO column of the SYSTCAT.INDEXES catalog table. If you collect DETAILED index statistics, a finer cluster-factor statistic between 0 and 1 is stored in the CLUSTERFACTOR column. Only one of these two clustering statistics can be recorded in the SYSCAT.INDEXES catalog table. In general, only one of the indexes in a table can have a high degree of clustering. A value of -1 indicates that no statistics for clustering are available. To compare the cluster-factor with the cluster-ratio values, multiply the cluster factor by 100 to obtain a percentage.

  Index scans that are not index-only accesses might perform better with higher cluster ratios. A low cluster ratio leads to more I/O for this type of scan, since after the first access of each data page, it is less likely that the page is still in the buffer pool the next time it is accessed. Increasing the buffer size might also improve the performance of an unclustered index.

  If table data was initially clustered in the order of a certain index, and the clustering statistics information indicates that the data is now poorly clustered for that same index, reorganize the table to cluster the data again.

- **Number of leaf pages:** Query the NLEAF column in the SYSCAT.INDEXES table to monitor number of leaf pages predicts how many index page I/Os are needed for a complete scan of an index.

  Random update activity can cause page splits that increase the size of the index beyond the minimum amount of space required. When indexes are rebuilt during the reorganization of a table, it is possible to build each index with the minimum amount of space.

Note: By default, ten percent free space is left on each index page when the indexes are rebuilt. To increase the free space amount, specify the PCTFREE parameter when you create the index. Whenever you reorganize the index, the PCTFREE value is used. Free space greater than ten percent might reduce frequency of index reorganization because the additional space can accommodate additional index inserts.

- **Comparison of file pages:** To calculate the number of empty pages in a table, query the FPAGES and NPAGES columns in SYSCAT.TABLES and subtract the NPAGES number from the FPAGES number. The FPAGES column stores the number of pages in use; the NPAGES column stores the number of pages that contain rows. Empty pages can occur when entire ranges of rows are deleted.

  As the number of empty pages increases, the need for a table reorganization also increases. Reorganizing the table reclaims the empty pages to compress the amount of space used by a table. In addition, because empty pages are read into the buffer pool for a table scan, reclaiming unused pages can also improve the performance of a table scan.

Choosing a table reorganization method

DB2 provides two methods of reorganizing tables: classic and in-place. In general, classic table reorganization is faster, but should be used only if your applications function without write access to tables during the reorganization. If your environment does not allow this restriction, although in-place reorganization is slower, it can occur in the background while normal data access continues. Consider the features of each method and decide which method is more appropriate for your environment.

To choose a table reorganization method, consider the features of the following methods:

- **Classic table reorganization:** This method provides the fastest table reorganization, especially if you do not need to reorganize LOB or LONG data. In addition, indexes are rebuilt in perfect order after the table is reorganized. Read-only applications can access the original copy of the table except during the last phases or the reorganization, in which the permanent table replaces the shadow copy of the table and the indexes are rebuilt.

  On the other hand, consider the following possible disadvantages:

  – There is a large space requirement. Because classic table reorganization creates the shadow copy of the table, it can require twice as much space as the original table. If the reorganized table is larger than the original, reorganization can require more than twice as much space as the original.

The shadow copy can be built in a temporary table space if the table's table space is not large enough, but the replace phase performs best in the same DMS table space. Tables in SMS table spaces must always store the shadow copy in temporary space.

- There is limited table access. Even read-only access is limited to the first phases of the reorganization process.
- This is an all or nothing process. If the reorganization fails at any point, it must be restarted from the beginning on the nodes where it failed.
- It is performed within the controller of the application that invokes it. The reorganization can be stopped only by that application or by a user who understands how to stop the process and has authority to execute the FORCE command for the application.

Recommendation: Choose this method if you can reorganize tables during a maintenance window.

- **In-place table reorganization:** The in-place method is slower and does not ensure perfectly ordered data, but it can allow applications to access the table during the reorganization. In addition, in-place table reorganization can be paused and resumed later by anyone with the appropriate authority by using the schema and table name.

Note: In-place table reorganization is allowed only on tables with type-2 indexes and without extended indexes like those used with the DB2 Spatial Extender.

Consider the following trade-offs:

- Imperfect index reorganization occurs. You might need to reorganize indexes later to reduce index fragmentation and reclaim index object space.
- A longer completion time is likely. When required, in-place reorganization defers to concurrent applications. This means that long-running statements or RR and RS readers in long-running applications can slow the reorganization progress. In-place reorganization might be faster in an OLTP environment in which many small transactions occur.
- This requires more log space. Because in-place table reorganization logs its activities so that recovery is possible after an unexpected failure, it requires more log space than classic reorganization.

  It is possible that in-place reorganization will require log space equal t: several times the size of the reorganized table. The amount of required space depends on the number of rows that are moved and the number and size of the indexes on the table.

Recommendation: Choose in-place table reorganization for 24x7 operations with minimal maintenance windows.

Refer to the REORG TABLE syntax descriptions in the DB2 UDB V8 Command Reference manual for detailed information about executing these table reorganization methods.

DB2 snapshot

The DB2 database manager maintains data about its operation, its performance, and the applications using it. This data is maintained as the database manager runs, and can provide important performance and troubleshooting information. For example, you can find out:

- The number of applications connected to a database, their status, and which SQL statements each application is executing, if any.
- Information that shows how well the database manager and database are configured, and helps you to tune them.
- When deadlocks occurred for a specified database, which applications were involved, and which locks were in contention.
- The list of locks held by an application or a database. If the application cannot proceed because it is waiting for a lock, there is additional information on the lock, including which application is holding it.

Because collecting some of this data introduces overhead on the operation of DB2, monitor switches are available to control which information is collected. To set monitor switches explicitly, use the UPDATE MONITOR SWITCHES command or the sqlmon() API.

You can access the data that the database manager maintains either by taking a snapshot or by using an event monitor.

Taking a snapshot

You can take a snapshot in one of the following three ways:

- Use the Control Center for a graphical interface
- Use the GET SNAPSHOT command from the command line.
- Write your own application, using the sqlmonss() API call.

Using the Control Center

The Control Center, available from the DB2 folder or with the db2cc command, provides a Performance Monitor tool that samples monitor data at regular intervals by taking snapshots. This graphical interface provides either graphs or textual views of the snapshot data, in both detail and summary form. You can also define performance variables using data elements returned by the database monitor.

The Control Center's Snapshot Monitor tool also lets you specify threshold values on performance variables to define exception conditions. When a threshold value is reached, one or more actions that you define occurs: notification through a window or audible alarm or execution of a script or program.

If you take a snapshot from the Control Center, while you are performing snapshot monitoring on either that object, or on any it its child objects you cannot perform an action that either alters, changes, or deletes a database object, such as an instance or database. If you are monitoring a partitioned database system, you cannot refresh the view of partitioned database objects. For example, you cannot monitor database A if you want to remove its instance. If, however, you are monitoring the instance only, you can alter database A.

To stop all monitoring for an instance (including any of its child objects), select Stop all monitoring from the pop-up menu for the instance. You should always stop monitoring from the instance, as this ensures that all locks that are held by the Performance Monitor are released.

SQL Explain tools

If you want to know how a query will be executed by DB2, you must analyze its access plan, which is the method for retrieving data from a specific table. The Explain Facility will provide information about how DB2 will access the data in order to resolve the SQL statements.

If you have identified a particular application as the possible source of a performance problem, you need to obtain the SQL statements that the application issues and analyze the access plan for the SQL statements.

The following list summarizes the different ways by which SQL statements can be obtained for analysis:

- Directly from the user or developer, who can extract it from their source code. There may be cases, however, when the SQL statements are generated dynamically from some sort of querying tool.
- From the Dynamic SQL Snapshot Monitor (also known as the global package cache).

- From the Statements Event Monitor. You can use the db2expln and dynexpln tools to understand the access plan chosen by the DB2 optimizer for a particular SQL statement. You can also use the integrated Explain Facility in the Control Center in conjunction with Visual Explain to understand the access plan chosen for a particular SQL statement. Both dynamic and static SQL statements can be Explained using the Explain Facility. One difference from the Explain tools is that with Visual Explain the Explain information is presented in a graphical format. Otherwise the level of detail provided in the two methods is equivalent.

For details on the use of the Explain tools, see the manual, *DB2 UDB V8 Administration Guide: Performance*, SC09-4821, in Appendix C, “SQL Explain tools”.

6

Monitoring and management

Monitoring provides mechanisms and information to self-managed systems and pro-active systems that should react in the best way possible to manage failures or reduce damage. Information collected from monitoring can help development staff to improve system capability. Also, since environments can change from initial design to real world implementation, and the requirements of resources change across the time, monitoring holds an important role in validation and analysis of the environment as a whole.

In this chapter we start with monitoring and go on to self-management concepts:

- Monitoring data collection
- Passive systems or information generation from monitoring's data
- Proactive and self-healing systems with DBA support

The first topic we address involves the common resources utilized by DB2 and the OS. The next two topics involve the monitoring tools provided in each software package (Windows and DB2) for monitoring data, how you can integrate them, and how to make them work in a proactive way.

We focus on providing an effective and manageable way to do message handling, and this will be our tool to improve DB2 responsiveness. We also demonstrate some techniques to do DB2 auditing.

6.1 General system monitoring considerations

In this section we discuss some concepts behind DB2 and Windows and the most important things to monitor. Since DB2 relies on OS and the hardware to do its functions, we derive an approach to monitor if DB2 and OS are working together properly.

6.1.1 Introduction

At principle, monitoring is collecting performance (how many in a specified period of time), utilization (how much or how many are being used in a point of time), behavior data (system condition or properties in a point of time), and event data (when something occurs). Usually performance data is built with utilization data. But both performance and utilization data have meanings by themselves and can be addressed separately or be combined to generate more complex information. All these data can be extracted or are produced by each component of the referred system and can be stored in places referred to as elements. Elements are the presentation of the system components.

Both DB2 and Windows are bundled with internal monitors. They can be customized to monitor just some specific elements or a wide group of elements. An element always stores the same specific information.

It is possible to classify elements in different types. An element's behavior varies accordingly by its meaning. Also, a new element of a different type can be created using other element's data. The following are the available element types in which monitor elements store data:

- **Counter:** Counts the number of times an activity occurs. Counter values increase during monitoring (for example: Piped Sorts accepted and Total Rejected Async I/O requests).
- **Gauge:** Indicates the current value for an item. Gauge values can go up and down depending on component activity (for example, number of users logged on the system or percentage of locklist held).
- **Water mark:** Indicates the highest (maximum) or lowest (minimum) value an element has reached since monitoring was started (for example, maximum number of agents registered or maximum anonymous users).
- **Information:** Provides reference-type details of your monitoring activities. This can include items such as partition names, aliases, path details, and server name.
- **Timestamp:** Indicates the date and time that an activity takes place by providing a value representing a point in time where an event occurred (for example, the timestamp of the last backup).

- **Time:** Returns the time elapsed or the time spent on an activity (for example, how much time it takes to de-fragment drive C, or how much time the statistics process takes to finish).

6.1.2 Things that you should consider when monitoring

Be careful when choosing what process to monitor, the desired information, and the sampling rate. Monitoring can be a high resource consumption task. When planning the monitoring tasks, the resources consumed by monitoring tasks should be considered and included in the overall performance and utilization requirement.

Consider the impact and requirements for the following measurements.

How many times data is collected (sampling rate)

Be sure that the sampling rate isn't too short, as you may spend a high amount of resources with repeating information; or too long, as you can't detect spiking conditions. The sampling rate have impact over the following items:

- **CPU consumption:** Querying for data usually uses API and functions which consumes processing power, especially when extracts data on-line (real time). On-line data extracting could be more CPU consumptive, since you are running additional application on the system.
- **I/O consumption:** Consider the information volume generated and the I/O needed to move the data around. Server internal buses capacity, the network bandwidth involved and the I/O required for data archive and retrieval should also be considered.
- **Storage consumption:** Historical information requires storage. The sampling rate influences directly the storage requirement. Verify with your company the requirements for how long the monitoring information should be kept.

Which data should be collected (volume rate)

Collecting the *right* amount of data is one of the key elements of system monitoring. Be careful of collecting too much unneeded information, or possibly collecting too little information, thus hindering problem determination. The volume of data collected has impact upon the following items:

- **CPU consumption:** The previous concern is valid here too. Querying for unnecessary extra data is something that you should avoid.
- **I/O consumption:** Avoid handling and transmitting unnecessary extra data.
- **Storage consumption:** Some data collected may not be applied immediately but will be beneficial later. Such types of data should be considered when choosing the data to be stored.

How long to keep the data

It is important to define a group of elements for time-line comparisons or to identify seasonal conditions. The primary impact on data retention is storage consumption. The longer the data is kept, the more storage is needed.

One approach to designing the monitoring process is to obtain only the basic data from the resource that you want to monitor to construct the additional information, based upon the basic data. For example, a maximum number of connections can be obtained by comparing the connections measurement in a period of time. Be aware that the derived data may not show the true picture. In the sample example — a maximum number of connections — a connection could be initiated and terminated between the connection snapshot interval.

If the computing resources (hardware) of your company are limited or the systems are mission-critical, you should only run the necessary items on the database server. We recommend handling the monitoring workload on a separate server.

6.1.3 Monitor types

Monitors can be classified as follows:

- **Data monitors:** These are responsible to get the various measurements and information from each module of the system. The data for each is stored on its own repository. For example, Perfmon is a data monitor.
- **Event monitors:** These only generate information when a pre-defined condition or an exception occurs, like Windows memory Page Faults/sec and DB2 Application Sort Overflow. Some event monitors can handle messages from more than one module. For example, DB2 log monitor and Windows EventLog service are event monitors.

Elements of both monitors can be created with a combination of the two types. For example, in DB2 Application sort overflow, the DB2 monitor detects when a sort overflows occurs, and automatically increments an internal counter.

DB2 and Windows have a list of pre-defined elements. You can also code your own monitor data generators.

6.1.4 Obtaining the data from the monitors or monitor interfaces

Usually, monitors only obtain the data from the system with which they are working. Users or monitoring systems access the monitoring data via a standardized interface. This standardized interface can be a plain-text file, a common memory area, a structured network protocol, and so on. The important thing is that the information can be obtained from any tool that can query for and read the data.

Monitoring interfaces available

In the following sections we describe the available interfaces that DB2 and Windows offers.

SQL Snapshot Monitoring

SQL Snapshot Monitoring is a new feature added on DB2 Version 8, which makes it possible to query monitoring data using a simple standard connection to the database.

APIs

Both DB2 and Windows offer Application Programming Interfaces (APIs) to their internal elements; for example, the DB2 CLI API and the Win32 API.

WMI / WBEM

There is an industry initiative that establishes management infrastructure standards and provides a way to combine information from various hardware and software management systems. This initiative is called Web-Based Enterprise Management (WBEM). WBEM is based on the Common Information Model (CIM) schema, which is an industry standard driven by the Desktop Management Task Force (DMTF).

Microsoft Windows Management Instrumentation (WMI) is an implementation of the WBEM initiative for supported Windows platforms. WMI is useful in a Windows enterprise network where it reduces the maintenance and cost of managing enterprise network components. WMI provides:

- A consistent model of Windows operation, configuration, and status.
- A COM API to allow access to management information.
- The ability to operate with other Windows management services.
- A flexible and extensible architecture allowing vendors a means of writing other WMI providers to support new devices, applications, and other enhancements.
- The WMI Query Language (WQL) to create detailed queries of the information.
- An API for management application developers to write Visual Basic or Windows Scripting Host (WSH) scripts.

The WMI architecture has two parts:

- **Management infrastructure:** This includes the CIM Object Manager (CIMOM) and a central storage area for management data, called the CIMOM object repository. CIMOM allows applications to have a uniform way to access management data.

- **WMI providers:** WMI providers are the intermediaries between CIMOM and managed objects. Using WMI APIs, WMI providers supply CIMOM with data from managed objects, handle requests on behalf of management applications, and generate event notifications. Windows Management Instrumentation (WMI) providers are standard COM or DCOM servers that function as mediators between managed objects and the CIM Object Manager (CIMOM). If the CIMOM receives a request from a management application for data that is not available from the CIMOM object repository, or for events, the CIMOM forwards the request to the WMI providers. WMI providers supply data, and event notifications, for managed objects that are specific to their particular domain.

See DTMF site http://www.dtmf.org for board companies that have solutions and collaborate with the standard.

In this chapter we concentrate on the monitoring resources that DB2 offers through WMI interface. In addition to monitoring interfaces, DB2 has also implemented management functions on WMI. You can manage DB2 using WMI capable software.

These are the various Windows providers on WMI:

- **Perfmon provider:** The Perfmon provider is responsible to provide various performance and utilization data stored in the elements to the WMI interface.
- **SNMP provider:** A Simple Network Management Protocol (SNMP) provider is responsible to interface SNMP protocol to translate into WMI structure. The SNMP (Simple Network Management Protocol) provider allows client applications to access SNMP information through WMI.
- **Windows NT event log provider:** A Windows NT event log provider is responsible to interface the events data and information stored on Windows NT logs, like application, security and system logs.
- **Win32 provider:** A Win32 provider is responsible to interface monitoring data from Windows system perspective. The Win32 provider defines the classes used to describe hardware or software available on Windows systems and the relationships between them.
- **WDM provider:** A WDM provider is responsible to interface the performance, WDM-enabled device drivers

DB2 has implemented the WMI interface for the following providers:

- **Perfmon provider:** There is a DB2 implemented Perfmon interface to give access to DB2 snapshot and event monitors.
- **Registry event provider:** A Registry event provider is responsible to interface the messages generated from DB2 when its registry is changed.

- **Registry provider:** A Registry provider is responsible to interface the data and configuration stored into DB2 Registry.
- **Windows NT event log provider:** The DB2 errors that are in the Event Logs can be accessed by WMI by using the built-in Windows NT Event Log provider.

6.1.5 Information generation

We have started with the concept of collecting data. Here we go one step further and consider information generation.

Information generation basically looks like monitoring. Information generation is generating or classifying information about a system based on the behavior of one or more monitored elements.

Usually information generation tends to be management or action oriented. Following are a few information types:

- **Behavior analysis:** When a system has less than 300 users, this is a normal condition. When it reaches 300 users, this is an attention condition. And when it reaches 1000 users or more, this changes to a critical condition.
- **Consumption analysis:** When a system has 20% of disk I/O being consumed, this is a normal condition. When 50% is being consumed, this is a warning condition. When consuming 80%, this is an alarming condition, and if it reach 99% or more, this is an unacceptable condition.
- **Time analysis:** When a user's query runs at a maximum of 10 seconds, this is a normal condition. When it is running for more than 10 seconds, this is a warning condition. When it is running for more than 1 minute, this is an alarming condition.

Both DB2 and Windows are capable of doing information generation, but not all DB2 or Windows functionality is covered by the system graphical tools. The tools usually can direct the information generated to text files. A script or program can be coded to handle special conditions based on the information in the text files.

Automated management with DBA support

Once information generation criteria are set, the monitors produce the monitoring information accordingly. By studying the monitoring information, the administrators can understand the support systems and react quickly when a condition is reached or an event occurs. Automation of the system management task is desired for many reasons:

- Sometimes, given the complexity of the environment involved, managing this task could be extremely difficult for a single DBA or system administrator.

- On a high availability system, response time is extremely important. If a condition could be predicted, a corrective action could be done proactively. Also, if an exception or an extreme condition is met suddenly, the system could react quickly to reduce the damage.
- Some companies don't have a full time DBA just to monitor the system.
- Actions to respond to certain events or conditions are repetitive.

A collection of actions could be programmed to execute automatically to prevent, avoid, or reduce the damage when the condition is met or the event occurs.

Some automated systems have evolved to the point that the system can take corrective actions on the system and analyze the results of its actions. On an automated system, more than one action could be taken in a high speed sequence. If these actions are not chosen correctly, the system could be left in a worse state. This is a risk of an automated system.

To avoid this risk, instead of letting the system execute the correction action, the system could suggest different actions or a group of actions to an administrator or an operator. Based on this information, the administrator could choose the most suitable action or take a different action.

Monitoring process considerations

The following procedures can be done to improve overall monitoring:

- **Centralize logging:** When dealing with monitoring, define a common place and a standard file structure for logging files.
- **Replaying an incident:** This could be an extremely valuable process to learn about the incident and test different approaches to solve the problem. To do this, store ambient information and SQL commands to replay an incident.
- **Know your environment and customize the monitoring task:** Some system events are seasonal, which could alter the system condition on a particular period of time. Be sure to predict these events and suspend or modify the parameters of monitoring tasks; for example:
  - **System maintenance:** System maintenance tasks happen regularly, such as backup, runstats, or reorg. When these events occur, it is possible that the system resource consumption might grow and signal an alarming condition. Also, there could be maintenance tasks that force the shutdown of the system. Monitoring tasks should be stopped or run on a different schedule during the system maintenance window.
  - **Seasonal conditions:** Like system maintenance, seasonal conditions occur frequently and are predictable, like monthly and yearly accounting runs, and can have a significant impact on resources. Monitoring tasks should be adjusted accordingly to prevent a false alarm.

6.2 Common resources to monitor

We start with the basic resources that are common to DB2 servers. DB2 software relies on the hardware it runs on, so each hardware resource represents a constraint. In the following sections we address these situations.

6.2.1 Memory

Memory is the first place where the programs store their information. Memory size is usually small compared to the needs of the entire system. Caching is a common OS technique to utilize memory and improve performance. Good caching is obtained when the hit ratio of information seek on memory is high and minimum disk I/O is reached. This is what you need to monitor on the system.

When monitoring memory, look for the following types of information:

- **Free physical memory space left:** If the system is reaching the maximum memory allocated for each component, it will start swapping memory pages onto virtual memory or swap space on disk. This is a bad condition and should be avoided.
- **Data page missing:** If the information queried was expected to be in the memory but is not found there, an I/O to the disk storage subsystem is required. This waste of time and processing exacts a high penalty and should be avoided.
- **Low caching hit ratio:** If the caching hit ratio is low, it signifies that the memory space for caching data isn't fully utilized and is wasted. This should be watched carefully.

Windows elements

Pay attention to the following Windows memory elements on your system:

- **Free physical memory space left:** If the OS reaches the maximum memory available on the hardware, the system itself could crash. This is a bad condition and should be avoided.
- **Virtual memory / thrashing:** When a system runs out of physical memory, it starts to use disk storage as memory. When the OS requires the data or executes a program, it moves the memory pages from the disk to the main memory. This is a high resource and processing power consuming task. Thrashing occurs when the system spends more time and processing power on doing virtual memory movement instead of processing the data itself.
- **Page faults:** Usually the system can handle soft page faults without problems. But hard page faults (when data pages are required from disk) can be a penalty that you should watch for.

DB2 elements

Pay attention to the following DB2 memory elements on your system:

- **Bufferpools:** When monitoring bufferpools, look for the hit ratio. A higher hit ratio means that DB2 is able to avoid disk access and improve the response time of the query.
- **Connection information storage:** For each application that connects to it, DB2 reserves a minimum amount of memory to keep status and executing queries. Watch if this space is running out of space frequently, or too much space was reserved for this area. Either situation will impact performance.
- **Sort area:** If a query requests information in a certain order (sorted), but there are no indexes to satisfy it, DB2 generates the result query in a temporary table inside the memory or the disk. DB2 chooses the destination of sorted data by analyzing the most recent statistical information and verifying the available sort memory. If the data generated is too big to fit in memory, it is placed in temporary storage on the disks. If sort memory is exhausted during the query, DB2 start to move the sorted data from memory to disk. This condition should be always avoided. Out-of-date statistics could cause DB2 extra work in moving the sorted data between memory and disk. To avoid this, be sure to have the most updated statistics.

6.2.2 Disk

When monitoring disk access, you should look for these types of information:

- **Growing consumption:** If the consumption is growing at a higher pace than expected, you could have space problems sooner than expected.
- **Repetitive read of the same information:** If the same information is being read all the time, and this information doesn't change, it means that you could having bad caching.
- **Random access:** Random access can impact deeply on access time. It is better if the I/O is done in contiguous blocks.
- **High access times:** Long access times could mean that the I/O sub-system has hardware problems, or other applications are requiring the same data (high concurrency).
- **Long queues:** Having some or a defined amount of queries waiting on a queue could be acceptable, instead of having unused resources. But having long queues could mean that the response times of your queries are being impacted or are taking more time to terminate than the previewed time.

- **Growing queues:** A good environment can sustain throughput and can provide information in the time expected. But growing queues may mean that the system is not capable of handling its current workload; or some extraneous query or operation was not predicted or did not pertain to the common workload, and is demanding more resources that it should be.
- **Data blocking:** Data is blocked on disk to speed-up search time of data access. To improve space utilization, the size of the blocks varies with the purpose. Data blocks should be the same size, even if the data inside is smaller than the block. You should look for the indicators of the specified block size and actual data size to detect if the system is wasting space.

Windows elements

Pay attention to the following Windows disk elements on your system:

- **Space consumption:** Many systems fail simply because they run out of space. Log space, backup file space, and system logging space can grow quickly. Be sure to monitor space and have proper pruning procedures to avoid disk space wasting.

DB2 elements

Pay attention to the following DB2 disk elements on your system:

- **Random access:** Random access can impact deeply on access time. It is better if the I/O is done in contiguous blocks. You can detect this with DB2 snapshot monitors, database counters, and the following elements: direct read requests, direct read times, and direct reads from database.
- **Data blocking:** Data blocking can impact heavily on storage consumption and on I/O operation speed. Storage consumption is affected when the data pages on the disk frequently lose space when the rows are added. Also, I/O operation is affected when a high number or I/O requests are done to perform small data movement, since I/O devices work better with sequential and blocked operations. You can detect the blocking status monitoring querying distribution information from the catalog tables after running runstats.

6.2.3 Network

Since the information needs to travel from the database servers to the clients, the network poses a potential bottleneck. When monitoring network performance, you should look for these types of information:

- **Packet loss / long response time:** A harsh communications environment is a threat to database data transfer. When the network loses a considerable volume of network packages, the data are required to be transferred again. Also, when a network is not optimally configured, the data takes more time to reach its destination.

- **Data blocking:** Usually, network programs do not send their data in a contiguous stream, because network data loss could invalidate long streams of data. To prevent this, data are sent through the network in blocks called packets. Typically, packet size varies with the purpose and size of data to be sent. You can use the elapsed time and the process consumption to detect whether the packet sizes are so big that they are wasting resources and introducing a higher probability of loss, or so small as to impose a major impact on processing and transmission.
- **Long queues:** Having some or a defined amount of queries waiting on a queue could be acceptable, instead of having unused resources. But having long queues could mean that the response times of your queries are being impacted or taking more time to terminate than the previewed time.
- **Growing queues:** A good environment can sustain throughput and can provide information in the time expected. But growing queues may mean that the system is not capable of handling its current workload; or some extraneous query or operation was not predicted or did not pertain to the common workload, and is demanding more resources that it should be.

Windows elements

Pay attention to the following Windows network elements on your system:

- **Packet loss / long response time:** Performance monitors have network elements that show packet loss, response time, and other network elements from each network card attached to the system and the protocol being used. This can be used to detect packet loss and long response times.
- **Long queues / growing queues:** Performance monitor have network elements that show long queues from each network card attached to the system.

DB2 elements

Pay attention to the following DB2 network elements on your system:

- **Transmission time:** You should look for the transmission time with snapshot monitoring to detect growing consumption and predict consumption bandwidth.

6.2.4 Security

Since a company's data and information are their most valuable assets, it is important to protect these assets against inadvertent leakage and malicious activities such as corruption and theft. DB2 relies on the operating system where it resides, so some of the security functionality and information are mixed.

When monitoring security, you should look for these types of information:

- **Security breaches tentative:** Some types of attacks could exploit specific conditions on DB2 running on Windows.
- **Changes in environment structure:** Changes in environment structure could be an indication of system invasion or a breakdown in the process.
- **User handling and authorization:** Misuse or carelessness could allow an unauthorized user to access sensitive data or change it. Industry data theft is always a concern in today's companies.
- **Uncommon workload and tasks:** Uncommon workload and tasks could be an indication of system invasion. They also could damage the system's functionality during the work day or even render it inoperable.

Windows elements

Pay attention to the following Windows security elements on your system:

- **Log-on tries:** DB2 recognizes users and groups from the underlying operation system. DB2 user log-on tries can be logged on Windows Event Monitor, under Security log.
- **Changes in environment structure:** Directory creation, DB2 directory quota definition, and changes in DB2 process priorities can be monitored and logged through Windows instrumentation.
- **User handling and authorization:** User and group creation, changes of user rights, and permissions to perform tasks and to handle objects, especially on DB2 objects and process, can be monitored through Windows instrumentation.
- **Unknown services and programs:** After the definition and implementation of production servers, they usually have only the essential programs and services running. Unknown services and programs could be "trojan horses" or other type of programs to break DB2 security. Monitoring the process list and creation can be done with Windows Task Manager or Windows Performance monitor.

DB2 elements

Pay attention to the following DB2 security elements on your system:

- **Unprogrammed routines:** Backup, restore, stop, and start routines can be verified through the db2audit facility, showing when they ran and the user who ran them.
- **DDL executions:** DDL executions that change environment structure and handle user and group permission can be monitored through the db2audit facility.

- **SQL packages and SQL execution:** The DB2 SQL statement monitor can be used to monitor and log SQL statements that could be searching sensitive data by unauthorized people, or even by a hacker.
- **Uncommon workload:** One of the most common methods of attack are known as denial-of-service (DOS). Be aware to check the DB2 SQL statement monitor when uncommon workloads are occurring. Also, SQL commands to dump table contents could being running.
- **db2audit events:** Verify commands that stopped or started the db2audit facility, and check audit records inside the db2audit.log. Also, the db2 event monitor's events and actions need to be monitored and logged in the db2audit.log file.

6.3 Windows system monitoring and tools

A well tuned and performing Windows environment impacts positively on the database overall. In the same way monitoring is done on DB2, you should monitor Windows environments.

You can find more information about Windows 2000 in the Windows 2000 Resource Kit. Also, a nice group of tools to manage and obtain information about Windows 2000 can be found on http://www.sysinternals.com.

In this topic, we explain more about each of the Windows tools and how it can be used with DB2 and Windows.

6.3.1 Task Manager

Task Manager is the most basic monitoring tool on Windows. It includes:

- Applications running
- Processes running
- Memory usage
  - Kernel memory counter and usage
  - Physical memory counter and usage
  - Commit charge
- General counters
  - Handles
  - Threads
  - Processes

To run task manager, right-click the taskbar on the desktop and choose **Task Manager**.

We have used this tool to monitor quickly the basic performance and utilization data. This is an easy way to diagnose CPU and network utilization. This tool does not have an option to save the monitored data.

Networking monitor and Terminal services have been added to Windows .NET; see Figure 6-1.

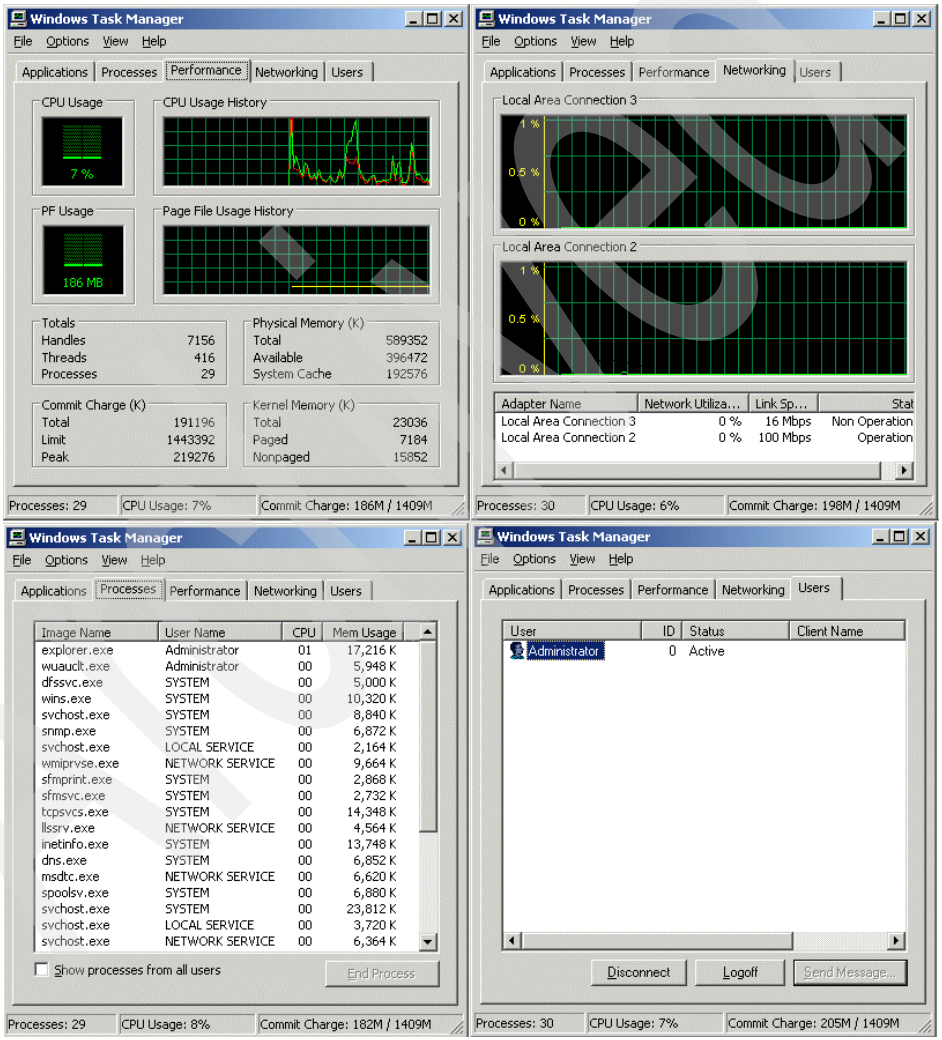

Figure 6-1 Windows .NET Task Manager

6.3.2 Performance Monitor and alert

DB2 is integrated with the underlying OS where it resides. All DB2 monitoring information is available through the Performance Monitor, which is the standard tool to do graphical analysis in the Windows environment. It contains a comprehensive group of counters for each Windows component, for example; networking counters, physical disk counters, processes counters, and so on. It also provides a way to save performance data and afterwards replay it to analyze the environment (Figure 6-2).

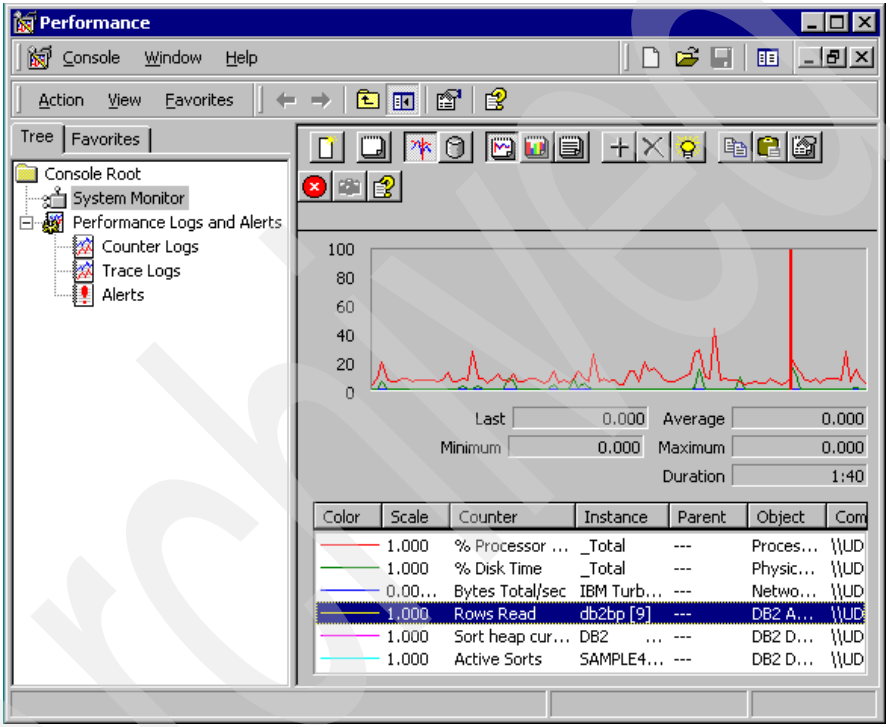

Figure 6-2 Performance Monitor in action

When DB2 is installed, it register its counters as a Perfmon provider. These counters are grouped as follows:

- DB2 Database Manager elements
- DB2 Database elements
- DB2 Applications

To start the Performance Monitor, go to **Start—>Programs—>Administrative Tools—>Performance**. To add DB2 counters, put the cursor in the middle of the chart panel and right-click. A menu will open; choose **Add Counters...** (Figure 6-3). A dialog box will open (Figure 6-4).

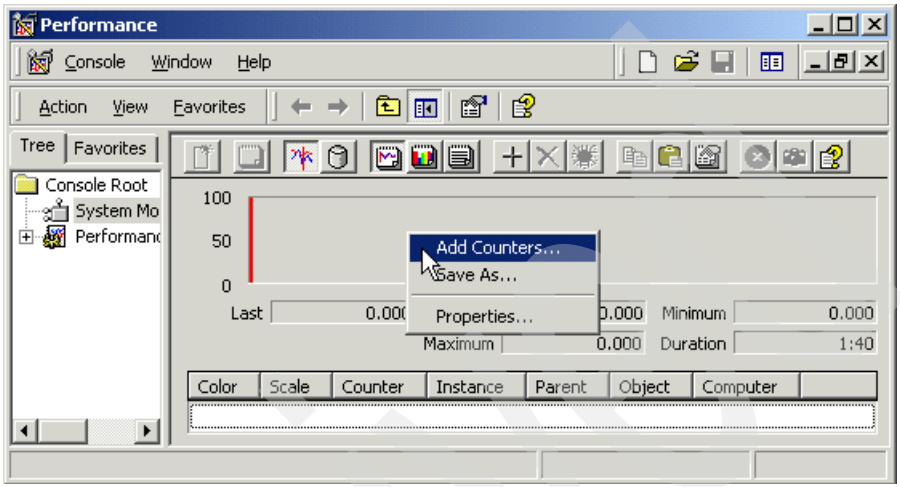

Figure 6-3 Adding counters

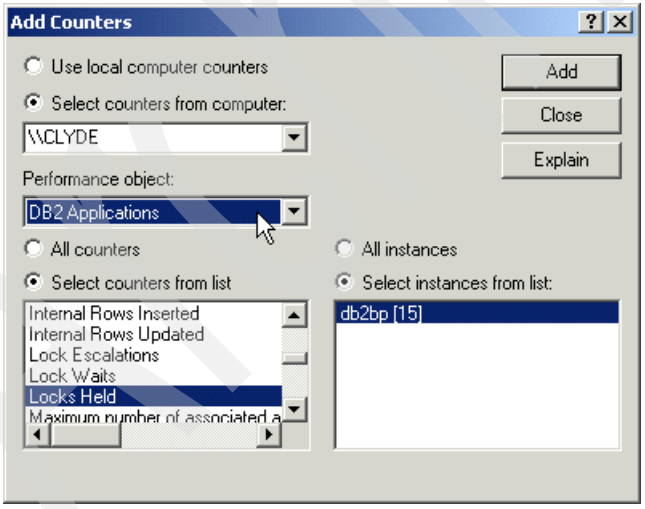

Figure 6-4 Choosing counters

In this dialog, the three groups are shown for the performance objects. When you select the desired group, their elements are shown on the list. It is possible to select all the counters, but the chart panel will become overwhelmed.

You can avoid this problem by selecting only the desired elements. When you select an element, all the objects that are currently *instantiated* on DB2 that have the selected element are shown on the right list of the panel. Select the desired object and click **Add**.

Repeat this procedure on all elements that you want to monitor, and finally, click **Close**. Now the chart panel will display a measure of the elements updated on the default time. To change the update time, right-click the chart panel and select **Properties**. There is a text box on the right side of the **Update automatically every:** check box. Change the value to the desired interval.

If the performance counters aren't shown, it could be that they weren't registered. To handle the Performance Monitor functions, there are three tools:

- **db2perfi**: Performance counters registration utility
- **db2perfr**: Performance Monitor registration tool
- **db2perfc**: Reset database performance values

To register DB2 performance counters, open a command line and run the command **db2perfi -i**. After that, you should be able to open the performance counters.

If you want to monitor DB2 performance counters on a server from a remote computer, or if you plan to start a Performance Monitor's logging task with DB2 counters, you could proceed as follows:

1. Run the command **db2perfr -r <userid> <password>** to register a user that db2perfmon will use to log on to the database to collect the data. This way allows any authorized user to start db2perfmon and collect data.
2. Change the user that starts the service **Performance Logs and Alerts** from the local system user to one that has privileges to connect to the database and query monitoring data.
   a. To do this, go to **Start—>Settings—>Control Panel**. Double-click **Administrative Tools** icon. Double-click **Services** icon. Right-click **Performance Logs and Alerts** service. Select **Properties**.
   b. A dialog will appear. Click the **Log On** tab and click the option box, **This Account**. Fill in the text box on the right side of the option box with the user name and fill in the text boxes below with the password and the password confirmation.

As mentioned before, the Performance Monitor can start background monitoring services to get data elements. With the DB2 db2perfmon provider, you can do background monitoring of the following types:

- **Counter logs:** Counter logs are monitoring tasks that could be programmed to run at a specific hour and for a defined amount of time. They collect the same monitoring data from the performance elements that the real time chart collects. But instead of displaying on the screen, the data is directed to text or binary files.
- **Alert logs:** Alert logs are monitoring tasks that could be programmed to run at a determined hour and for a defined amount of time. They collect the same monitoring data from the performance elements that the real time chart collects. But instead of displaying on the screen, the data is verified against a condition. If this condition is true, the alert log can do the following tasks:
  - Log an entry in the application event log
  - Send a network message to a user or a computer on the network
  - Start a performance data log or counter log
  - Run a program

Since the Alert Logs allow you to run a wide variety of tasks, it is possible to do some automated management with them in a similar way as you can with the DB2 Health Monitor.

6.3.3 Event viewer

The Windows environment has standard log mechanisms. System and applications can use these mechanisms to set a notification level. There are three levels of log entry:

Notification	Used when the system or one application just wants to send a message about successful operation or assorted information.
Warning	Used when the system or one application detects that a problem could happen or is about to happen, or when an expected behavior is not found or met, but the system could continue working without affecting the functioning at all.
Error	Used when the system or one application wants to send or write an error message about an unexpected termination or a failed operation. When a message of this type is generated, there is a potential chance of damage or data loss.

Windows has three event logs:

- **Application log:** The application log is the log where DB2 services and modules write their messages. All DB2 messages mentioned before are written inside the application log.
- **System log:** The system log applies to events that are related to the environment and basic structure and functionality of the system.
- **Security log:** Since DB2 relies on the underlying OS to do user authentication, all the tries that are done to log on are registered on the security log.

To control the DB2 event generation on Windows Event Logs, DB2 Version 8 adds a new parameter NOTIFYLEVEL to the database manager.

The event entries are recorded on these event logs on the following way: All common and administrative messages are registered on Windows events logs, and all messages that are for DB2 diagnosis are saved on the db2diag.log files. If DB2 db2diag.log file logs an error about an internal program detection, a log entry is also recorded on the Windows event log advising that an internal message has been generated.

Windows event logs don't have an option to run a command or a script when a new log is added. If you want to do so, you should use WMI functions that will loop on the Windows log to search a new log entry.

6.4 DB2 monitoring capability

On DB2 Version 8, many tools and functionalities have been added or modified to take advantage of features in the Windows environments.

DB2 Version 8 incorporates Self Managing And Resource Tuning (SMART) technology. To accomplish that functionality, DB2 is bundled with many tools and special commands that allow a broad analysis of the DB2 environment.

DB2 implements some types of proactive and self-healing techniques with the concept of monitoring by exception. Now on DB2, when some problems appear, DB2 recommends some options for what you can do, and offers tools to do what you choose, immediately.

6.4.1 Log files

Since DB2 Version 8 writes the administrative logs on the Windows Event Logs, the only file left on the disk is the diagnostic files *xxxdiag.log*, where *xxx* means the instance name that the file pertains. The files can be found on the root instance path inside the SQLLIB directory. The only exception to this is the diagnostic file of the administrative instance (db2dasdiag.log), where the file is inside the *dump* directory on the instance root path.

These are some examples of the path mentioned previously:

- DB2 instance log
  C:\Program Files\IBM\SQLLIB\DB2\db2diag.log
- DB2DAS00 instance log
  C:\Program Files\IBM\SQLLIB\DB2DAS00\dump\db2dasdiag.log

6.4.2 Health Center and Memory Visualizer

The Health Center and the Memory Visualizer are the newest monitoring tools that have joined the DB2 tools collection. Both of these tools have been designed with the SMART concept in mind (Figure 6-5 and Figure 6-6).

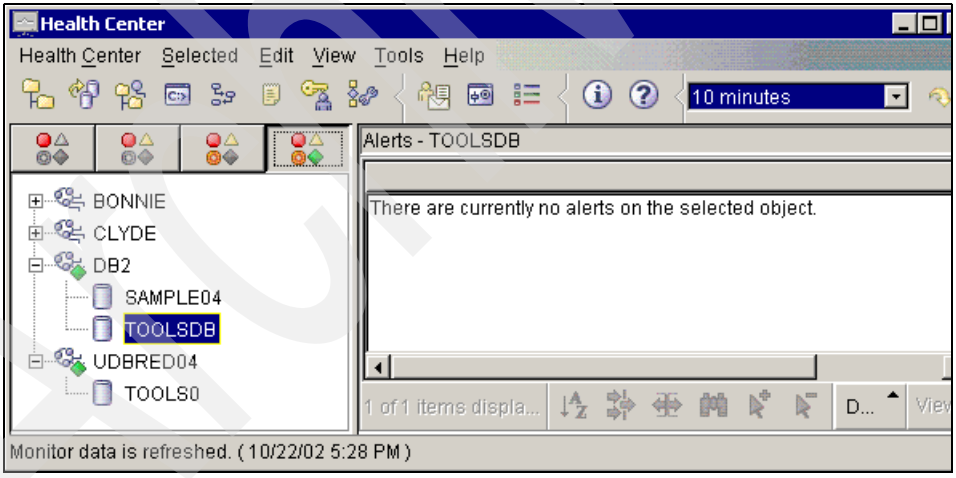

Figure 6-5 Health Center

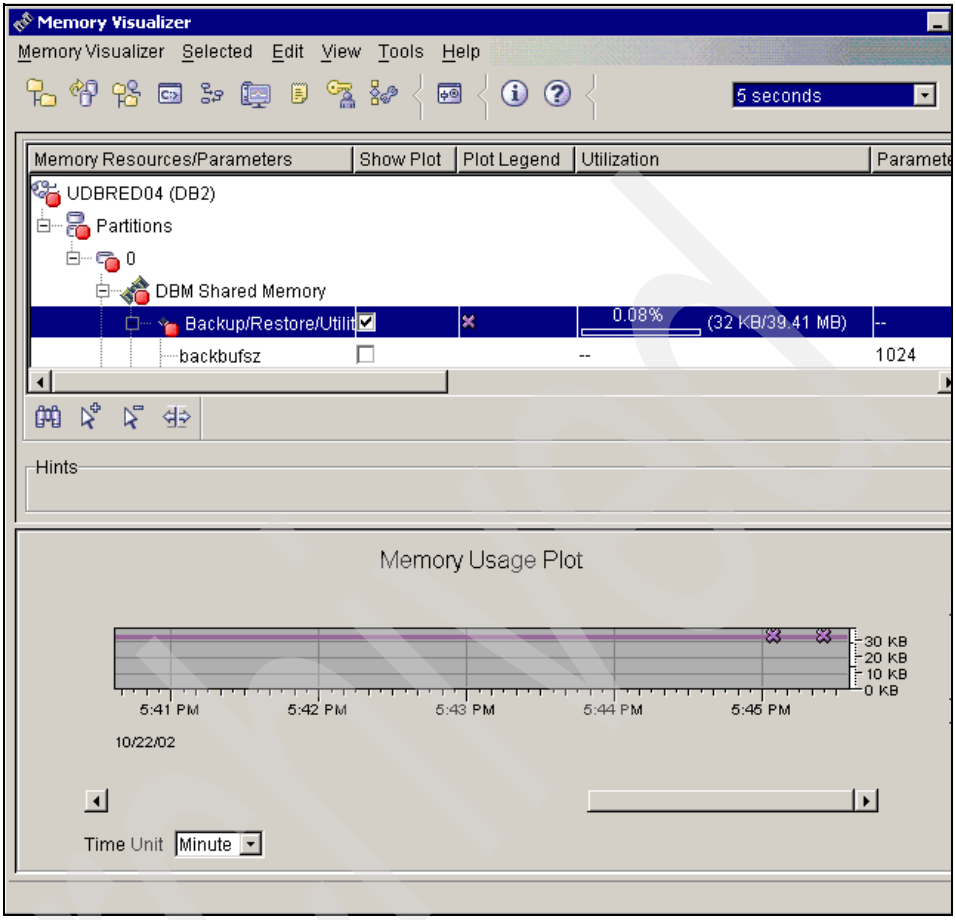

Figure 6-6 Memory Visualizer

Health Center

DB2 Version 8 has a new monitoring tool, the Health Center, which can be used to monitor the state and utilization of many parts of the database manager, the databases, and user agents.

In this section, we show a step-by-step utilization of DB2 Health Center. If you need more information on each category, please refer to Appendix A in the *System Monitor Guide and Reference Version 8*.

When you use DB2, a Health Monitor continuously monitors a set of health indicators. If the current value of a health indicator is outside the acceptable operating range defined by its warning and alarm thresholds, the Health Monitor generates a health alert. DB2 comes with a set of predefined threshold values, which you can later customize. For example, you can customize the alarm and warning thresholds for the amount of space used in a table space.

Depending on the configuration of the DB2 instance, the following actions can occur when the Health Monitor generates an alert:

- An entry is written in the administration notification log, which you can read from the Journal.
- The Health Center status beacon appears in the lower right corner of the DB2 GUI Tools window. You can set the status beacon options opening on any DB2 GUI tool by clicking the menu **Tools—>Tools Settings**, and then clicking the tab **Health Center Status Beacon** (Figure 6-7).

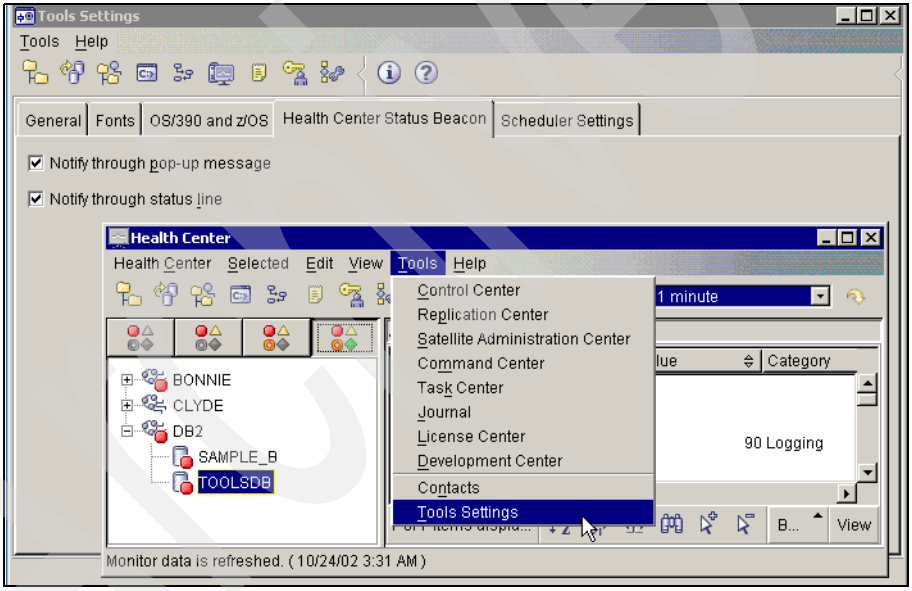

Figure 6-7 Status Beacon configuration options

- A script or task is executed.
- An E-mail or pager message is sent to the contacts that you specify for this instance.

These are some of the key tasks that you can perform with the Health Center:

- View the status of the database environment. Beside each object in the navigation tree, an icon indicates the most severe alert for the object (or for any objects contained by that object). For example, a green diamond icon beside an instance means that the instance and the databases contained in the instance do not have any alerts.
- View the alerts for an instance or a database. When you select an object in the navigation tree, alerts for that object are shown in the pane to the right.
- View detailed information about an alert, and recommended actions. When you double-click an alert, a notebook appears. The first page shows the details for the alert. The second page shows any recommended actions.
- Configure the Health Monitor settings for a specific object, and the default settings for an object type or for all objects within an instance.
- Select which contacts will be notified by an e-mail or pager message.
- Review the history of alerts for an instance. You can use the toggle buttons at the top of the navigation bar to filter the alerts according to their severity:
  - Displays only alarms (the most severe type of alert)
  - Displays alarms and warnings
  - Displays alarms, warnings, and attentions

Now, to demonstrate some of the functionality, we start a common Health Center instance and set some values. But before you begin, be sure that you have defined one instance and have created the scheduler or tools database server or created locally the TOOLSDB.

1. To start DB2 Health Center, go to **Start—>Programs—>IBM DB2 —>Monitoring Tools—>Health Center**. The Health Center will appear.
2. To see all the DB2 servers at your network, right-click the small button with the four icons highlighted or colored.
3. Right-click the server name that you want to monitor. A menu pop-up will be displayed (Figure 6-8):
   - The first option is up, to control the execution of Health Monitor. It changes its values depending on the running status of Health Center monitor. If it is running, a **Stop Health Monitor** option will appear. And if is stopped, a **Start Health Monitor** option appears. A small green diamond (beacon) is placed on the server icon where the Health Monitor is running.
   - The notification log of the events inside the Journal tool can be seen by clicking the option **Show notification log**.

– Positioning the mouse over the **Configure** option will open a second menu item.

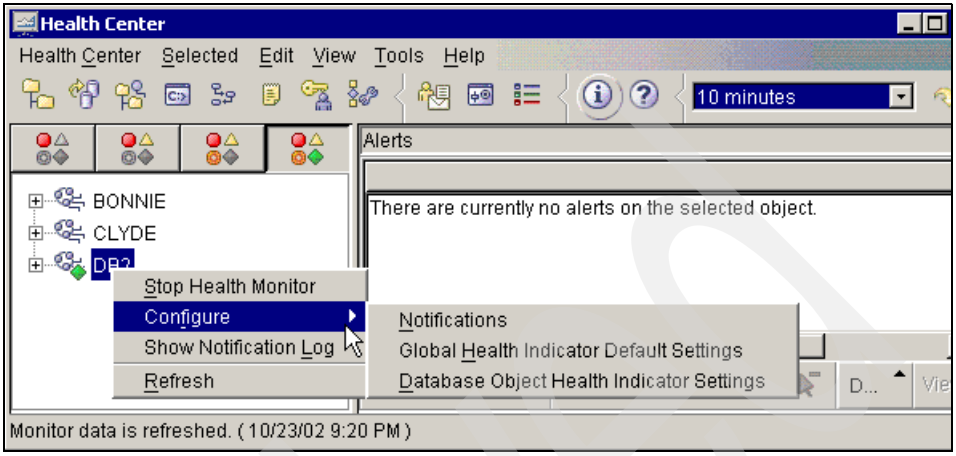

Figure 6-8 Health Center Server options

- Click the option **Notification** If you want to configure notifications via E-mail to an administrator or a group of administrators. A dialog will open showing the list of the contacts. You can set this or change it by clicking the **Change** button. In our case, there is a setting to send an E-mail to the operator (Figure 6-9).

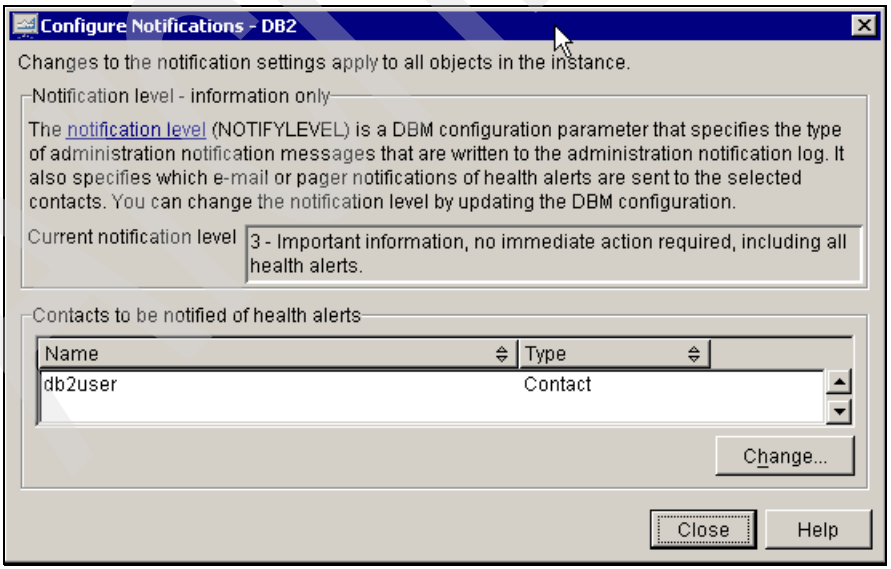

Figure 6-9 The administrator that will be contacted by E-mail

- On the **Global Health Indication Default Settings** option, you can set the database manager indicators threshold values.
- On the **Database Health Indicator Settings option**, you can set the database indicators threshold values for each database on the servers' instance.

4. We proceed by clicking the **Global Health Indication Default Settings** option. The dialog to configure global health indication default settings is shown (Figure 6-10).

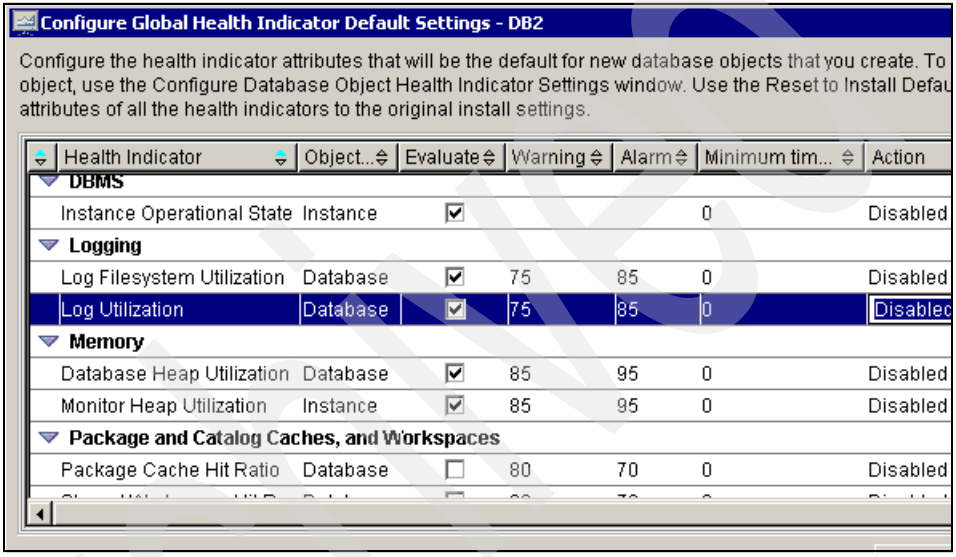

Figure 6-10 Configuring global health indicator settings

On this dialog, the columns that you should pay more attention to are:

- **Evaluate:** This option activates and deactivates the monitoring of the indicator itself.
- **Warning and Alarm:** These columns let you change the threshold value for each type of information that triggers notifications and action procedures.
- **Minimum time pre-alert:** Sometimes spike conditions occur, but it doesn't mean that the system is in a special condition. If you want to set a minimum time for the condition before starting notification, change the value inside this column. This value is in minutes.
- **Action:** In this column you can set up actions to do on warning or alert conditions. Clicking the cell corresponding to the action of the desired health indicator will bring up a button. By clicking this button, a dialog appears for you to set up actions.

– **Description:** In this column the name of the element or indicator on database monitor is shown.

5. Now we will change the warning value of **Log Utilization** from its original value to zero percent of utilization to generate a warning condition for demonstration purposes (Figure 6-11). Click the **OK** button, and if a dialog advises you that the updates are done, you just need to click the **Close** button.

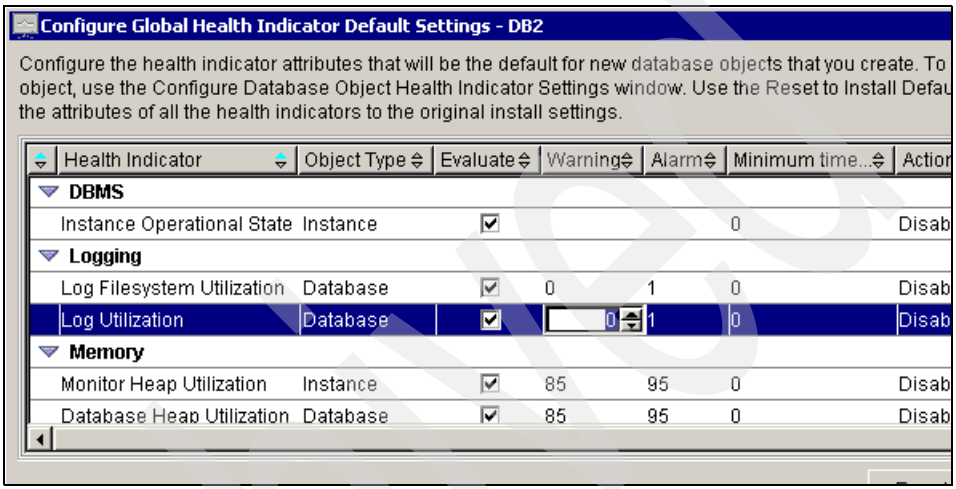

Figure 6-11 Changing the log utilization warning threshold to 0 per cent

6. Now, on the previous dialog (Figure 6-11), wait for the period of time of the refresh. If it is too long, you can change the refresh time on the combo box at top and right-most part of the dialog. After this period, the dialog should display a warning condition. Right-click the warning indicator on the alert list box. If the alert begins flooding the list, you can disable the evaluation of the item by clicking the **Disable Evaluation** item. We proceed with the process of analyzing the warning condition by clicking the **Show Details** option (Figure 6-12).

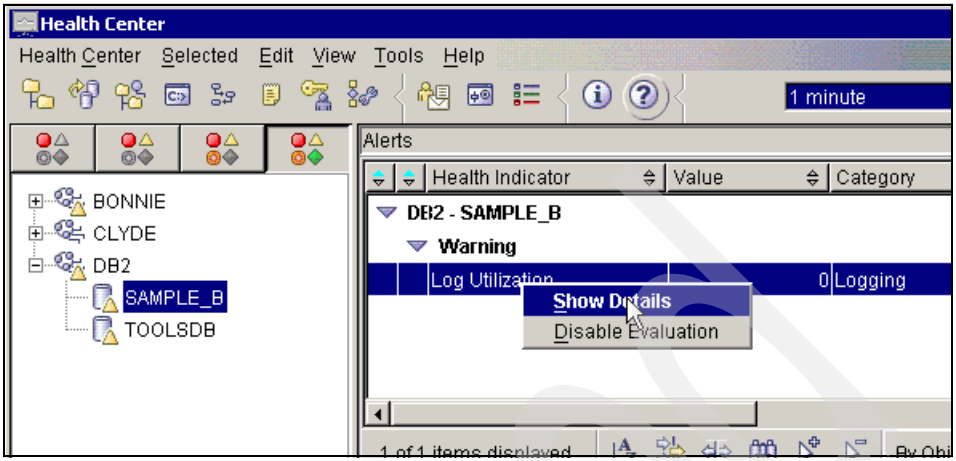

Figure 6-12 The warning condition

7. A pop-up menu will display the alert details (Figure 6-13). This information contains most of the collectable data about the warning event itself and an explanation. Click the tab **Recommendations**; DB2 will analyze this event using the SMART technology and enumerate some actions that you can do to solve the warning condition.

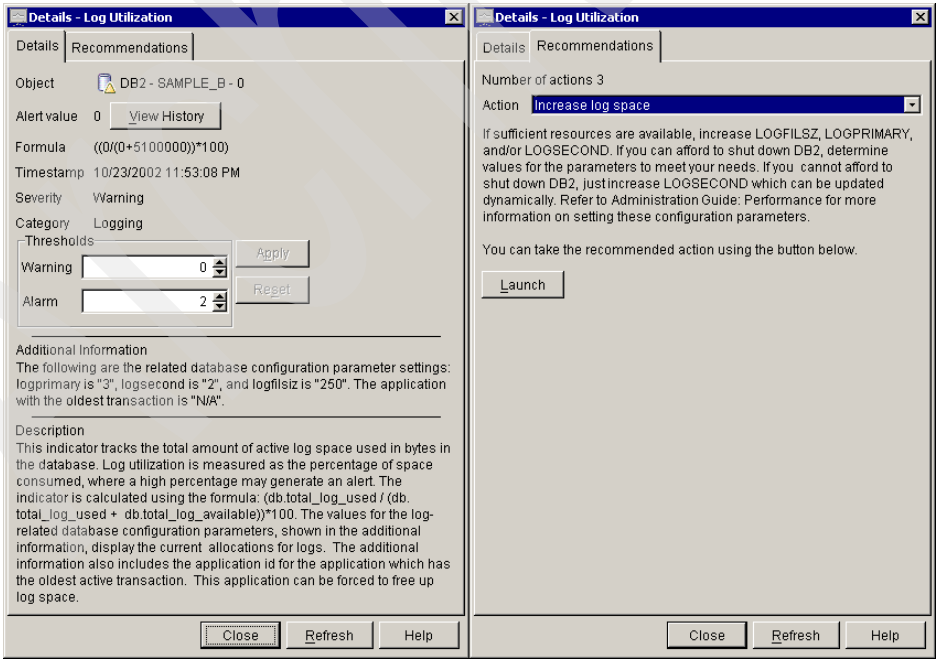

Figure 6-13 Warning information and SMART technology in action

8. Here is the E-mail sent to the user about the warning condition (Figure 6-14):

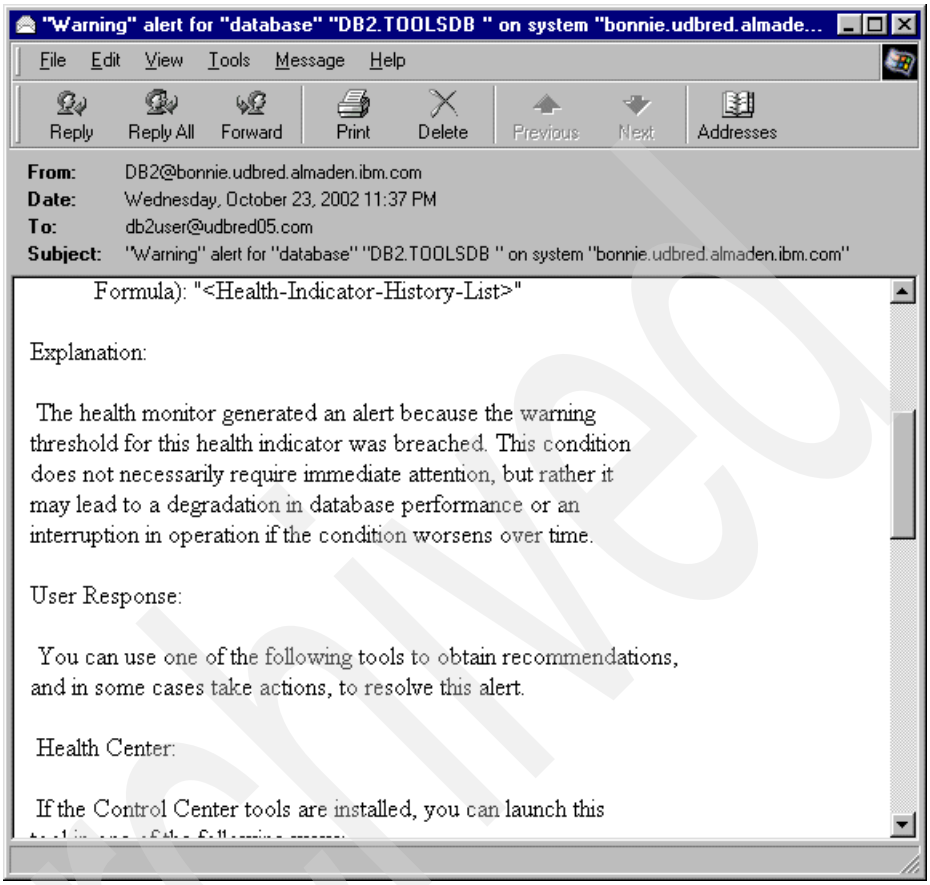

Figure 6-14 E-mail automatically sent about the warning condition

Memory Visualizer

You can use the Memory Visualizer to monitor the memory allocation of a DB2 Version 8 instance.

The top pane of the window shows memory components organized in a navigation tree. Historical values of alarm and warning thresholds are shown to the right of each component. The lower pane shows a memory usage plot.

The high-level memory components include:

- Database manager shared memory
- Database global memory
- Application global memory
- Agent/Application shared memory
- Agent private memory

Each high-level component is divided into lower-level components that determine how the memory is allocated and deallocated. For example, memory is allocated and deallocated when the database manager starts, when a database is activated, and when an application connects to a database.

To display a memory usage plot, select the **Show Plot** check box for the desired component or components. The usage plots for different components are shown with a different color and shape, for each change to a configuration parameter. The usage plot also shows the original value, the new value, and the time when the value changed.

These are some of the key tasks you can perform with the Memory Visualizer:

- View overall memory usage
- Specify which memory information to display and which information to hide for DB2 instance and its databases
- Update the configuration parameters for an individual memory component to prevent it from using too much or too little memory
- Save the memory allocation data
- Load memory allocation data from a file into a Memory Visualizer window

6.4.3 DB2 Event Monitor

In this section we describe the main functionality and the best resources found on DB2 event monitors. You can find a very extensive explanation of DB2 monitoring in the *IBM DB2 Universal Database System Monitor Guide and Reference Version 8*.

In DB2 Version 8, considerable effort was expended to improve event monitor functionality and usability. These are the most interesting capabilities added:

- **Creation of event monitor with SQL commands**:

  Now it is possible to create DB2 event monitors using standard SQL. For example:

  ```
  CREATE EVENT MONITOR CONN_MON FOR CONNECTIONS WRITE TO FILE C:\CONN_MON.EMD
  ```

  This command will create an event monitor called CONN_MON to monitor connections and save the data in a file called CONN_MON at the c:\ directory.

  The command line program to create a DB2 event monitor is also included in this version, for example:

  ```
  db2evmon [-db] <database-alias> [-evm] <event-monitor-name>
  ```

  or

  ```
  db2evmon -path <event-monitor-target>
  ```

- **Directing the output data to a table or pipe:**

  DB2 Version 8 also offers the possibility of directing the output data to tables and pipes. For example:

```
CREATE EVENT MONITOR CONN_MON FOR CONNECTIONS WRITE TO TABLE
```

  or

```
CREATE EVENT MONITOR CONN_MON FOR CONNECTIONS WRITE TO PIPE
'/UDBRED04/EVMON/PIPE'
```

6.4.4 DB2 Governor

DB2 Governor is a very effective tool that can monitor the behavior of applications that run against a database. It also can change certain behaviors, depending on the rules that you specify in the governor configuration file. The governor logs any actions that it takes.

You can find a very detailed explanation of DB2 governor and how to use it in *Chapter 9, Administration Guide: Performance DB2 Version 8.*

7

High availability

The concept of High Availability (HA) roughly equates to “available almost all of time, 24 hours a day, 7 days a week, 365 days a year”. We know that 100% availability is not a reality today, but rather a goal. Reality is closer to 99.99% uptime, and we strive for 99.999% uptime. Our goals are to design and build systems that are highly available by compensating for both planned and unplanned outages that can be caused by single points of failure. As software and hardware technology evolves in availability, we find ourselves compensating, more often than not, for unplanned outages.

In this chapter we describe the high availability (HA) features of DB2 Universal Database V8.1 for Windows 2000 operating systems. We discuss not only the high availability features that are built into DB2 UDB, but also the integration of DB2 UDB into the high availability features of the Windows 2000 operating system. We begin by describing database availability features that can be implemented with relative ease and progress, then go on to the more highly available configuration features that can provide for 99.999% uptime. We cover the following capabilities for high availability:

- Database features
- Monitoring instances
- Standby server
- Clustered server

7.1 Database features for high availability

Most people think of high availability in terms of redundant hardware and software provided by standby and clustered server implementations. However, there are many more, perhaps less exciting, features built into DB2 UDB that provide for high availability, which we simply take for granted. In this section we cover features that ship with DB2 Universal Database V8.1, right out of the box, and that significantly contribute to database high availability.

7.1.1 Buffer pool management

Most database professionals will probably tell you that buffer pools are the single most important database resource that can be tuned to improve performance. Given the importance of memory (specifically, buffer pool tuning) to database performance, it is no wonder that IBM has focused more attention to this area.

In previous versions of DB2 UDB, buffer pools could be created, changed, and dropped while the database was online. In fact, the database had to be online; however, these changes did not take effect until the database was recycled, which resulted in downtime.

What's new with Windows Server 2003: Microsoft has announced that both Windows Server 2003 Enterprise and Datacenter editions will provide support for Hot Pluggable Memory. This high availability feature, called Hot Add Memory, will allow memory to be automatically added to the operating system addressable memory space without re-booting. Support will require, at a minimum, hardware that supports hot pluggable memory.

Online buffer pools

DB2 UDB V8.1 introduces online buffer pool management. This new feature allows you not only to create, change, and drop database buffer pools without having to recycle the database; it allows you to put these changes into effect immediately, providing for high availability during these changes.

New options have been added to SQL statements for managing buffer pools. The default option is now IMMEDIATE, which indicates that changes to buffer pools will take effect without recycling DB2 databases. For backward compatibility, the DEFERRED option allows you to implement changes after the database has been recycled.

For example, let us say you are running on a small 4-way SMP system with 4 GB of memory on Windows 2000 Server. You start out with a single large 1-GB buffer pool, but later decide to create a second buffer pool for a select few table spaces. Now you can the reduce the size of the first buffer pool and create the second buffer pool on-the-fly, without recycling the database:

```
ALTER BUFFERPOOL MyBp1 SIZE 250000 IMMEDIATE
CREATE BUFFERPOOL MyBp2 SIZE 250000 IMMEDIATE
ALTER TABLESPACE MyTableSpace2 BUFFERPOOL MyBp2
```

7.1.2 Tablespace management

Although most database professionals will agree that memory management, specifically buffer pool tuning, is important for performance, most are unlikely to site any specific instances where failure to perform this important task has resulted in system downtime. Table space management, on the other hand, is not only vital to database performance, but if neglected could result in system downtime.

In previous versions of DB2 UDB, table space management consisted of two types: autopilot and manual. The first type, autopilot or System Managed Storage (SMS), has been part of DB2 on Intel platforms for over a decade. Database Managed Storage, the manual type, was first introduced in DB2 UDB v5, and resulted in faster I/O operation. In this version of DB2 UDB, DMS table spaces could only be extended by adding additional containers which would immediately invoke table space re-balancing. Later, in DB2 UDB v7.2, some relief was provided by allowing DMS table space containers to be extended in size.

DB2 UDB V8.1 introduces new management functionality for table spaces, defined as Database Managed Storage (DMS). Now you can add a DMS table space container without having to re-balance the containers, a task that can sometimes overwhelm systems. You can also shrink table spaces that might have been accidently over-allocated or have had a significant amount of data removed. All of these new options can be performed without impacting performance and without bringing the database or instance offline, providing for high availability.

7.1.3 Configuration parameters

Once common system maintenance task involves making configuration changes to DB2 instances and databases. In the past, very few of these changes could be performed without having to recycle either the database or the instance to get the new configuration parameter value to take effect. DB2 UDB V8.1 provides for high availability by allowing buffer pools to be managed while the database is still online, as well as allowing over 50 configuration parameters at the instance and database levels to be changed without having to bring systems offline.

Online configuration parameters

DB2 UDB V8.1 introduces online configuration parameters that provide for high availability, as changes to these specific configuration parameters take effect immediately without having to bring down DB2 instances or databases. This means that users will not have to be forced off of databases so that database configuration parameters can take effect. It also means that DB2 instances will not have to be stopped and started to have instance (dbm) configuration parameters take effect.

IBM has concentrated on specific instance (dbm) and database configuration parameters in the area of memory utilization, such as catalog and package cache, statement, sort, and utility heaps. In the event that you prefer the old school, IBM provides for backward compatibility, thus allowing you to defer these configuration changes until the next instance or database recycle. This is a key requirement for performance tuning during a series of benchmark runs:

```
DB2 GET DBM CFG SHOW DETAIL
DB2 GET DB CFG FOR <database_alias> SHOW DETAIL
```

Automatic configuration parameters

DB2 UDB V8.1 also introduces the beginning of self tuning configuration parameters. Only a few key parameters can be set to AUTOMATIC. DB2 UDB V8.1 currently supports automatic settings for instance_memory, database_memory and maxappls.

You can determine the current value of configuration parameters that support the automatic setting by using the SHOW DETAIL option on the DB2 following DB2 commands:

```
DB2 GET DBM CFG SHOW DETAIL
DB2 GET DB CFG FOR <database_alias> SHOW DETAIL
```

7.1.4 Loading data

As data sizes increase in database systems and user demands for that data approach 24x7x365, one of the challenges we face is loading large volumes of data without impacting the availability of that data for users and applications.

Prior to V8 of DB2 UDB, the only method to add data to a table and still have the table available for user and application access was to issue SQL INSERT statements and issue frequent COMMITs. This process can be very slow and places large demands on the logging facility for both disk space and CPU. We discuss the new online loading capabilities introduced in DB2 UDB V8.

Online loading

When loading data into a table in Version 8, the table space in which the table resides will no longer be locked. Users have full read and write access to all the tables in the table space, except for the table being loaded. For the table being loaded, the existing data in the table will be available for read access if the load is appending data to the table. To utilize the online load feature a new option ALLOW READ ACCESS has been added to the syntax of the LOAD command.

When using the ALLOW READ ACCESS option load will lock the target table in a share mode. The table state will be set to both LOAD IN PROGRESS and READ ACCESS. Readers may access the non-delta portion of the data while the table is being load. In other words, data that existed before the start of the load will be accessible by readers to the table, data that is being loaded is not available until the load is complete. This option is not valid if the indexes on the target table are marked as requiring a rebuild.

When there are constraints on the table, the table state will be set to CHECK PENDING as well as LOAD IN PROGRESS, and READ ACCESS. At the end of the load the table state LOAD IN PROGRESS state will be removed but the table states CHECK PENDING and READ ACCESS will remain. The SET INTEGRITY command must be used to take the table out of CHECK PENDING. While the table is in CHECK PENDING and READ ACCESS, the non-delta portion of the data is still accessible to readers, the new (delta) portion of the data will remain inaccessible until the SET INTEGRITY command has completed. A user may perform multiple loads on the same table without issuing a SET INTEGRITY command. Only the original (checked) data will remain visible, however, until the SET INTEGRITY command is issued.

For additional options available for use with the LOAD command see the manual DB2 UDB V8 Command Reference publication number SC09-4828.

These new load features significantly improve the availability of the data and help customers deal with the maintenance of large data volumes and shrinking maintenance windows.

7.1.5 Reorganizing data

After many changes to table data, logically sequential data may be on non-sequential physical data pages so that the database manager must perform additional read operations to access data. Additional read operations are also required if a significant number of rows have been deleted.

As tables are updated with deletes and inserts, index performance can also degrade in the following ways:

- Fragmentation of leaf pages may occur. When leaf pages are fragmented, I/O costs increase because more leaf pages must be read to fetch table pages.
- The physical index page order no longer matches the sequence of keys on those pages, which is referred to as a badly clustered index. When leaf pages are badly clustered, sequential prefetching is inefficient and results in more I/O waits.
- The index develops more than its maximally efficient number of levels. In this case, the index should be reorganized.

For both tables and indexes that have become fragmented, running the REORG utility can improve performance. Prior to DB2 UDB V8, the REORG utility execution resulted in the table and indexes being unavailable for user and application access. We discuss below how to run the REORG utility online for both tables and indexes that permit access while executing, providing you with higher availability.

For detailed information on the REORG utility see the manual DB2 UDB V8 Command Reference and Chapter 5, “Performance” on page 191 of this book.

Online index reorganization

With DB2 Version 8, you can read and update a table and its existing indexes during an index reorganization using the new REORG INDEXES command.

During online index reorganization, the entire index object (that is, all indexes on the table) is rebuilt. A *shadow copy* of the index object is made, leaving the original indexes and the table available for read and write access. Any concurrent transactions that update the table are logged. Once the logged table changes have been forward-fitted and the new index (the shadow copy) is ready, the new index is made available. While the new index is being made available all access to the table is prohibited.

The default behavior of the REORG INDEXES command is ALLOW NO ACCESS, which places an exclusive lock on the table during the reorganization process, but you can also specify ALLOW READ ACCESS or ALLOW WRITE ACCESS to permit other transactions to read from or update the table.

Online table reorganization

Online table reorganization allows applications to access the table during the reorganization. In addition, online table reorganization can be paused and resumed later by anyone with the appropriate authority by using the schema and table name.

Online table reorganization is allowed only on tables with type-2 indexes and without extended indexes.

To utilize online table reorganization, the REORG command syntax has added the following options:

- **INPLACE:** Reorganize the table while permitting user access.
- **ALLOW READ ACCESS:** Allow only read access to the table during reorganization.
- **ALLOW WRITE ACCESS:** Allow both read and write access to the table during reorganization. This is the default behavior.

7.1.6 Database recovery

Developing a database recovery strategy is one of the most critical step in business continuity and resumption. In this section we do not discuss how to go about developing a recovery strategy. Instead we focus on key DB2 UDB features that afford high availability as part of your recovery strategy. Specifically, we discuss online database backup, online database restore, and database transaction logs as they pertain to high availability.

Online backups

The primary ingredient in database recovery is of course a backup. Without a backup copy of the data in our database we are completely helpless in the event of a hardware, software, or human failure that corrupts or otherwise makes the database unusable. Having said that, it is not in our best interests to impose planned outages on our system in order to satisfy our requirement for backing up the database. Although not the only solution, but perhaps the most practical, is to implement online database backups.

DB2 UDB provides for high availability during the backup process by supporting online backups. Database backups can be invoked from the DB2 Control Center, the DB2 Command Line Tools, or the appropriate DB2 Administrative API.

Performing online database backups requires the implementation of linear logging. Linear logging can be enabled by updating the database configuration parameter logretain. Once enabled, the database will be placed in a backup bending state, preventing any connections to the database until a backup is taken or logretain is disabled. The logretain database configuration parameter can be enabled using the DB2 Control Center, DB2 Command Line Tools, or the appropriate DB2 Administrative API.

Online restore

DB2 UDB provides for high availability during the restore process as well by supporting online restore and roll forward processing. During an online restore the database remains online. Online restore can only be performed at the table space level, but table spaces can be individually restored from complete backup images.

A database or table space can be restored and subsequently rolled forward using the DB2 Control Center, the DB2 Command Line Tools, or the DB2 Administrative API.

Note: The database configuration file will not be recovered during a database restore, unless it is corrupt or missing, in which case it is recovered and a warning message is generated.

Transaction log blocking

Introduced in DB2 UDB v7.2 as a DB2 registry variable, transaction log blocking, is now implemented in DB2 UDB V8.1 as a database configuration parameter. This transaction log features provides for high availability by avoiding a database shutdown in the event that the primary transaction log path and secondary transaction log path (mirrorlogpath) in the case of log mirroring becomes full.

The blk_log_dsk_ful database configuration parameter can be set using the DB2 Control Center, the DB2 Command Line Tools, or the appropriate DB2 Administrative API. By default this parameter is disabled, probably for backward compatibility, but should be enabled as part of your standard configuration procedures, shortly there after you create a database.

```
db2 update db cfg for <database_alias> using blk_log_dsk_ful YES
```

Once enabled, if a log path full condition occurs read only transaction will be permitted against the database, however insert, update, and delete transactions will be unable to commit units of work until the condition is rectified. During the log full condition, the database logger will periodically attempt to create the new log file. Once created, the log path full condition is removed and pending UOW will be committed. If transaction log archiving is enabled, the log path full condition may be alleviated by the successful archiving of a log file, in which case the database logger will recycle the archived log by renaming it as the next log file.

Note: On Windows, log path full conditions are written to the Application log file that can be viewed using the Windows Event Viewer.

Transaction log archiving

Log archiving, the process of removing database transaction logs from the database server to a safer place, has been supported by DB2 UDB for close to a decade now. The concept is relatively simple: When the database manager determines that a log file is no longer needed, it calls a user exit to archive the log. Once the log has been successfully archived by the user exit program, the database manager simply re-uses the old log file by renaming it. The only minor detail that we left out is the user exit. Although IBM does provide some very good sample programs in \IBM\SQLLIB\SAMPLES\USEREXIT\, it should be noted that the “u” in user exit means “you” get to write it!

Note: Sample user exit programs are provided for all supported platforms. You can modify these programs to suit your particular requirements. The sample programs are well commented with information that will help you to use them most effectively.

Developing user exits

If you plan on implementing log archiving, as well as log shipping for that matter, you must first develop a user exit program. The user exit program will be responsible for archiving database transaction logs from the current active log path. It will also be responsible for retrieving transaction logs during roll forward recovery.

Although the sample programs provided by IBM are developed in C, you can write a user exit program in just about any programming language that can be called by the database manager and accept command line parameters. The user exit program must be called *db2uext2* and must also be located in a directory that is included in the systems *path* statement.

In order to enable the user exit feature of DB2 UDB you must first update the database configuration parameter called userexit. This parameter can be enabled using the DB2 UDB Control Center, through any of the Command Line Tools, or through a Administrative API. Once the userexit database configuration parameter is enabled (userexit=YES) linear logging will also be enabled for the database and a database backup must be performed. It is also a good idea to enable the logretain (logretain=YES) database manager configuration parameter as well. This enables linear logging which is required for many other high availability features such as online backups.

The user exit program is invoked by DB2 as required. Information about the database transaction log file to be archived is passed to the user exit program via command line parameters. After the user exit program has successfully archived the log file, by means of copying the log file somewhere, it indicates so by passing a zero (0) return code back to DB2. If the user exit program is unable to successfully archive the log file, it indicates so by returning a non-zero return code; in which case DB2 UDB will not recycle the log, but rather attempt to archive it at a later time.

The syntax for the user exit call is as follows:

```
db2uext2 -OS<os> -RL<db2rel> -RQ<request> -DB<dbname> -NN<nodenum>
-LP<logpath> -LN<logname> -AP<tsmpasswd> -SP<startpage> -LS<logsize>
```

Note: You can find a list of acceptable user exit return codes in the online documentation under **Concepts —> Administration —> Data recovery —> Recovery logs —> Log file management**.

Transaction log mirroring

Introduced in DB2 UDB v7.2, log mirroring provides for high availability in the event of a database transaction log failure. Database transaction log failures can occur due to hardware, software, and/or human failure. You can protect the database transaction logs, and therefore keep the database online, by implementing transaction log mirroring.

In DB2 UDB v7.2, log mirroring was implemented with a DB2 Registry variable called db2_newlogpath2. This registry variable did not support a given path, but rather is simply enabled (set to 1) or disabled (set to 0) log mirroring. Enabling the registry variable created a second set of mirrored logs in a directory located in the same path as the original logs but with the number two “2” appended to the log path.

This technique required the use of volume mounting to place the mirrored logs on a secondary physical device. The Windows 2000 mount volume utility should be used to mount a separate physical drive into the new log path subdirectory. This created a conflict because the Microsoft Cluster Service does not support clustering mounted volumes. DB2 UDB V8.1 supports a new database configuration parameter called mirrorlogpath that overcomes this limitation.

Note: The release notes for DB2 UDB v7.1 FixPak 3 indicate that… “Because Windows NT and OS/2 do not allow “mounting” a device under an arbitrary path name, it is not possible (on these platforms) to specify a secondary path on a separate device.”… note that this does not apply to Windows 2000.

MIRRORLOGPATH parameter

The new V8.1 MIRRORLOGPATH configuration parameter enables the database to write an identical second copy of log files to a physically separate disk without having to mount volumes. The database configuration parameter requires the database to be recycled (deactivate/activate) before it will be in effect. The following diagram, Figure 7-1, shows the Configure Database Logging Wizard being used to enable log mirroring.

You should consider placing the mirrored logs on a hard drive that is on a separate disk controller or host bus adapter to eliminate the hardware as a single point of failure.

In the event of a write failure to either the active or mirror log paths the database logger will mark the failing path bad and continue writing log records to the good path. After the current log is filled, the database logger will attempt to use the bad log path again. If a failure occurs while writing to the remaining good log path, database manager will shut down the database.

In the event of a read failure from the active log path, the database logger will attempt to use the secondary log path. If the secondary log path is available the logger will use this path until the next log file is required. The DB2 logger makes no attempt to synchronize the primary and secondary mirrored log files, but instead keeps track of bad log path information, so that the correct paths are used during crash recovery, log archiving/shipping, and roll forward recovery.

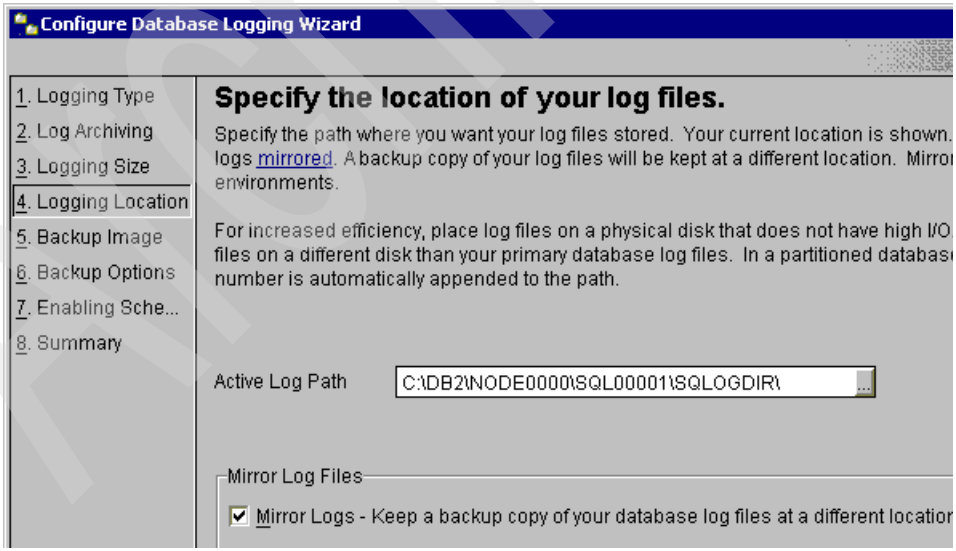

Figure 7-1 Configure Database Logging Wizard

7.1.7 Application processing

When considering providing high availability for a database, an area that is often overlooked is the impact that application processing can have on the availability of data. When applications perform massive inserts, updates, and deletes on a database, the database management system must lock the modified rows to provide data integrity, making them unavailable to other applications and users.

When many users access the same database, you must establish some rules for the reading, inserting, deleting, and updating of data records. The rules for data access are set by each application connected to a DB2 database and are established using locking mechanisms. Those rules are crucial to guarantee the integrity of the data, but they may decrease concurrency on database objects. On a system with much unnecessary locking, your application may take a very long time to process queries due to lock-waiting, even if the system is rich in hardware resources and well tuned. In this section we discuss how you can control concurrency appropriately and minimize lock-waits to improve your application's performance.

To minimize lock-waits, what you should consider first is eliminating unnecessary locks, by doing the following:

- Issue COMMIT statements at the right frequency.
- Specify the FOR FETCH ONLY clause in the SELECT statement.
- Perform INSERT, UPDATE, and DELETE at the end of a unit of work if possible.
- Choose the appropriate isolation level.
- Eliminate next key locks by using type-2 indexes.
- Release read locks using the WITH RELEASE option of the CLOSE CURSOR statement if acceptable.
- Avoid lock escalations impacting concurrency by tuning the LOCKLIST and MAXLOCKS database configuration parameters.

Each of these guidelines is further explored in the following sections.

Issue COMMIT statements

Executing COMMIT statements takes overhead due to disk I/O such as flushing logged data into disks; however, since all locks held in a unit of work are released at the end of the unit of work, putting COMMIT statements frequently in your application program improves concurrency. When your application is logically at a point of consistency; that is, when the data you have changed is consistent, put in a COMMIT statement.

Be aware that you should commit a transaction even though the application only reads rows. This is because shared locks are acquired in read-only applications (except for the uncommitted read isolation level, which will be discussed in the next section) and held until the application issues a COMMIT or closes the cursor using the WITH RELEASE option (it will be discussed later in this chapter).

Note: If an application opened with cursors declared using the WITH HOLD option, locks protecting the current cursor position of them will not be released when a COMMIT is performed.

Specify FOR FETCH ONLY clause

A query with FOR FETCH ONLY clause never holds exclusive locks on the rows, thus you can improve concurrency using this clause.

INSERT, UPDATE, and DELETE at end of UOW

When an application issues an INSERT, UPDATE, or DELETE statement, the application acquires exclusive locks on the affected rows and will keep them until the end of the unit of work. Therefore, perform INSERT, UPDATE, and DELETE at the end of a unit of work if possible. This provides the maximum concurrency.

Isolation levels

DB2 UDB provides different levels of protection to isolate the data from each of the database applications while it is being accessed.

These levels of protection are known as isolation levels, or locking strategies. Choosing an appropriate isolation level ensures data integrity and also avoids unnecessary locking. The isolation levels supported by DB2 are listed below, ordered in terms of concurrency, starting with the maximum:

- Uncommitted Read
- Cursor Stability
- Read Stability
- Repeatable Read

Uncommitted Read

The Uncommitted Read (UR) isolation level, also known as a “dirty” read, is the lowest level of isolation supported by DB2. It can be used to access uncommitted data changes of other applications. For example, an application using the Uncommitted Read isolation level will return all of the matching rows for the query, even if that data is in the process of being modified and may not be committed to the database. You need to be aware that two identical queries may get different results even if they are issued within a unit of work, since other concurrent applications can change or modify those rows that the first query retrieves.

Uncommitted Read transactions will hold very few locks. Thus they are not likely to wait for other transaction to release locks. If you are accessing read-only tables or it is acceptable for the application to retrieve uncommitted data updated by another application, use this isolation level, because it is most preferable in terms of performance.

Note: Dynamic SQL applications using this isolation level will still acquire locks on the system catalog tables.

Cursor Stability

The Cursor Stability (CS) isolation level locks any row on which the cursor is positioned during a unit of work. The lock on the row is held until the next row is fetched or the unit of work is terminated. If a row has been updated, the lock is held until the unit of work is terminated. A unit of work is terminated when either a COMMIT or ROLLBACK statement is executed.

An application using Cursor Stability cannot read uncommitted data. In addition, the application locks the row that has been currently fetched, and no other application can modify the contents of the current row. As the application locks only the row on which the cursor is positioned, two identical queries may still get different results even if they are issued within a unit of work.

When you want the maximum concurrency while seeing only committed data from concurrent applications, this isolation level should be chosen.

Read Stability

The Read Stability (RS) isolation level locks those rows that are part of a result set. If you have a table containing 10,000 rows and the query returns 10 rows, then only 10 rows are locked until the end of the unit of work.

An application using Read Stability cannot read uncommitted data. Instead of locking a single row, it locks all rows that are part of the result set. No other application can change or modify these rows. This means that if you issue a query twice within a unit of work, the second run can retrieve the same answer set as the first. However, you may get additional rows, as another concurrent application can insert rows that match to the query.

Note: Remember that selected rows are locked until the end of the unit of work. Therefore, do not forget to issue a COMMIT (or ROLLBACK) statement even if your application is read-only. A COMMIT (or ROLLBACK) statement will terminate the unit of work and release held locks.

Repeatable Read

The Repeatable Read (RR) isolation level is the highest isolation level available in DB2. It locks all rows that an application references within a unit of work, no matter how large the result set. In some cases, the optimizer decides during plan generation that it may get a table level lock instead of locking individual rows, since an application using Repeatable Read may acquire and hold a considerable number of locks.

An application using Repeatable Read cannot read uncommitted data of a concurrent application. As the name implies, this isolation level ensures the repeatable read to applications, meaning that a repeated query will get the same record set as long as it is executed in the same unit of work. Since an application using this isolation level holds more locks on rows of a table, or even locks the entire table, the application may decrease concurrency. You should use this isolation level only when changes to your result set within a unit of work are unacceptable.

Choosing an isolation level

When you choose the isolation level for your application, decide which concurrency problems are unacceptable for your application and then choose the isolation level which prevents these problems. Remember that the more protection you have, the less concurrency is available.

- Use the Uncommitted Read isolation level only if you use queries on read-only tables, or if you are using only SELECT statements and getting uncommitted data from concurrent applications is acceptable. This isolation level provides the maximum concurrency.
- Use the Cursor Stability isolation level when you want the maximum concurrency while seeing only committed data from concurrent applications.
- Use the Read Stability isolation level when your application operates in a concurrent environment. This means that qualified rows have to remain stable for the duration of the unit of work.
- Use the Repeatable Read isolation level if changes to your result set are unacceptable. This isolation level provides minimum concurrency.

Setting an isolation level

The isolation level is defined for embedded SQL statements during the binding of a package to a database using the ISOLATION option of the PREP or the BIND command. The following PREP and BIND examples specify the isolation level as the Uncommitted Read (UR).

```
PREP program1.sqc ISOLATION UR
BIND program1.bnd ISOLATION UR
```

If no isolation level is specified, the default level of Cursor Stability is used.

If you are using the command line processor, you may change the isolation level of the current session using the CHANGE ISOLATION command.

```
CHANGE ISOLATION TO rr
```

For DB2 Call Level Interface (DB2 CLI), you can use the SQLSetConnectAttr function with the SQL_ATTR_TXN_ISOLATION attribute at run time. This will set the transaction isolation level for the current connection referenced by the ConnectionHandle. The accepted values are:

- SQL_TXN_READ_UNCOMMITTED: Uncommitted Read
- SQL_TXN_READ_COMMITTED: Cursor Stability
- SQL_TXN_REPEATABLE_READ: Read Stability
- SQL_TXN_SERIALIZABLE: Repeatable Read

You can also set the isolation level using the TXNISOLATION keyword of the DB2 CLI configuration as follows:

```
UPDATE CLI CFG FOR SECTION sample USING TXNISOLATION 1
```

The following values can be specified for the TXNISOLATION keyword: 1, 2, 4, 8, or 32. Here are their meanings:

- 1 = Uncommitted Read
- 2 = Cursor Stability (default)
- 4 = Read Stability
- 8 = Repeatable Read

You can use the DB2 CLI configuration for JDBC applications as well. If you want to specify the isolation level within the JDBC application program, use the setTransactionIsolation method of java.sql.Connection. The accepted values are:

- TRANSACTION_READ_UNCOMMITTED: Uncommitted Read
- TRANSACTION_READ_COMMITTED: Cursor Stability
- TRANSACTION_REPEATABLE_READ: Read Stability
- TRANSACTION_SERIALIZABLE: Repeatable Read

Eliminate next key locks by using type-2 indexes

In DB2 UDB V8 a new type of index was introduced called a type-2 index. The primary advantages of type-2 indexes are:

- They improve concurrency because the use of next-key locking is reduced to a minimum. Most next-key locking is eliminated because a key is marked deleted instead of being physically removed from the index page.
- An index can be created on columns with a length greater than 255 bytes.
- A table must have only type-2 indexes before online table reorg and online table load can be used against the table.
- They are required for the new multidimensional clustering facility.

All new indexes are created as type-2 indexes, except when you add an index on a table that already has type-1 indexes. In this case the new index will also be a type-1 index because you cannot mix type-1 and type 2 indexes on a table.

All indexes created before Version 8 were type-1 indexes. To convert type-1 indexes to type-2 indexes, uses the REORG INDEXES command. To find out what type of index exists for a table, use the INSPECT command.

Close cursor with release

You should use the Read Stability isolation level when qualified rows have to remain stable for the duration of the unit of work. If changes to your result set are unacceptable, you should use the Repeatable Read isolation level. When using Read Stability or Repeatable Read, more locks are hold than using Cursor Stability or Uncommitted Read.

If you look at your application using Read Stability or Repeatable Read, you may find that all queries in the application do not need the protection which Read Stability or Repeatable Read provides, that means, there may be queries which can release locks before the end of the unit of work. For such queries, use a CLOSE CURSOR statement that includes the WITH RELEASE clause when closing the cursor.

```
CLOSE c1 WITH RELEASE
```

If the WITH RELEASE clause is specified, all read locks (if any) that have been held for the cursor will be released. If it is not specified, held locks will not be released until the unit of work ends.

The WITH RELEASE clause has no effect for cursors that are operating under the CS or UR isolation levels. When specified for cursors operating under RS or RR isolation levels, the WITH RELEASE clause ends some of the guarantees of those isolation levels, because all read locks will be released before the end of the unit of work.

An RS and an RR cursor may experience the nonrepeatable read phenomenon, which means that if you open a cursor, fetch rows, close the cursor with WITH RELEASE clause, reopen the cursor, and fetch rows again, then the second query can retrieve the different answer set as the first because other applications can update the rows that match to the query. An RR cursor may experience the phantom read phenomenon as well. After closing the cursor with the WITH RELEASE clause, the second query can retrieve the additional rows which were not returned by the first query, because other applications can insert rows that match to the query.

If a cursor that is originally RR or RS is reopened after being closed using the WITH RELEASE clause, then new read locks will be acquired.

Lock escalation

If your application acquires and holds locks on almost all of the rows in one table, it may be better to have one lock on the entire table. Each database allocates a memory area called a lock list, which contains all locks held by all applications concurrently connected to the database.

Each lock requires memory to represent it. If a number of row locks can be replaced with a single table lock, the locking storage area can be used by other applications.

When DB2 converts the row locks to a table lock on your behalf, this is called lock escalation. DB2 will perform lock escalation to avoid resource problems by too many resources being held for the individual locks.

Two database configuration parameters have a direct effect on lock escalation:

- **LOCKLIST:** This defines the amount of memory allocated for the locks.
- **MAXLOCKS:** This defines the percentage of the total lock list permitted to be allocated to a single application.

There are two different situations possible for lock escalation:

- **Case 1:** One application exceeds the percentage of the lock list as defined by the MAXLOCKS configuration parameter. The database manager will attempt to free memory by obtaining a table lock and releasing row locks for this application.
- **Case 2:** Many applications connected to the database fill in the lock list by acquiring a large number of locks. DB2 will attempt to free memory by obtaining a table lock and releasing row locks.

Also note that the isolation level used by the application has an effect on lock escalation:

- Cursor Stability will acquire row level locks initially. If required, table level locks can be obtained in such a case as updating many rows in a table. Usually, a very small number of locks are acquired by each cursor stability application, since they only have to guarantee the integrity of the data in the current row.
- Read Stability locks all rows in the original result set.
- Repeatable Read may or may not obtain row locks on all rows read to determine the result set. If it does not, then a table lock will be obtained instead.

If a lock escalation is performed, from row to table, the escalation process itself does not take much time; however, locking entire tables decreases concurrency, and overall database performance may decrease for subsequent accesses against the affected tables.

Once the lock list is full, performance can degrade, since lock escalation will generate more table locks and fewer row locks, thus reducing concurrency on shared objects in the database. Your application will receive an SQLCODE of -912 when the maximum number of lock requests has been reached for the database.

Configure LOCKLIST and MAXLOCKS parameter

To avoid decreasing concurrency due to lock escalations or errors due to a lock list full condition, you should set appropriate values for both the LOCKLIST and MAXLOCKS database configuration parameters. The default values of these parameters may not be big enough (LOCKLIST: 10 pages, MAXLOCKS: 10%) and cause excessive lock escalations.

See Section , “Locking” on page 262 in Chapter 5, “Performance” for details on configuring LOCKLIST and MAXLOCKS.

7.2 Monitoring instances for high availability

DB2 Universal Database has supported automated crash recovery for so long now that we simply take it for granted. Automated crash recovery, as you may remember, is DB2’s ability to automatically invoke crash recovery when a database is activated and crash recovery is required. Automated crash recovery or auto-restart is configured individually for each database by enabling the auto restart (autorestart=on) database configuration parameter. Prior to the availability of this configuration parameters, it was the Database Administrators task to manually restart a crashed database by invoking the DB2 restart database command.

Auto-restart is indeed a great high availability feature, but it only works as long as the database crash was not a result of a DB2 instance crash — in which case, there is no equivalent auto-restart parameter available at the instance level. This, of course, could be implemented and monitored by the DB2 Administration Server (DAS) instance, but unfortunately it is not. Instead, DB2 relies on the services provided by the operating system to automatically restart instances.

On UNIX based systems, the Fault Monitor Facility improves the availability of non-clustered DB2 environments through a sequence of processes that work together to ensure that DB2 is running. That is to say, the init daemon monitors the Fault Monitor Coordinator (FMC); the FMC; in turn; monitors the fault monitors; and the fault monitors, in turn, monitor DB2. On the Windows operating system, we simply call this capability Services Recovery.

7.2.1 Services Recovery

With Windows 2000, the Services Recovery facility is built into the operating system. It improves the availability of non-clustered services by providing recovery actions that will be performed in the event of a services failure. It can be configured to monitor and recover any component of DB2 that is implemented as a Windows service, including:

- DB2 Instance
- DB2 Governor
- DB2 JDBC Applet Service
- DB2 License Server
- DB2 Remote Command Server
- DB2 Security Server
- DB2 Warehouse Logger
- DB2 Warehouse Server
- DB2 Database Administration Server (DAS)

Additionally, Data Replication support in DB2 UDB V8.1 has been enhanced for the Windows operating system so that it is now completely managed as a service by the operating system. That is to say, the Replication Service, once installed, can be automatically or manually started, paused, resumed, restarted, and recovered as well.

The DB2 services created by the DB2 Setup Wizard can be configured using the Service Microsoft Management Console (MMC). The Services MMC can be readily accessed by selecting **Start —> Program —> Administrative Tools —> Services** (Figure 7-2).

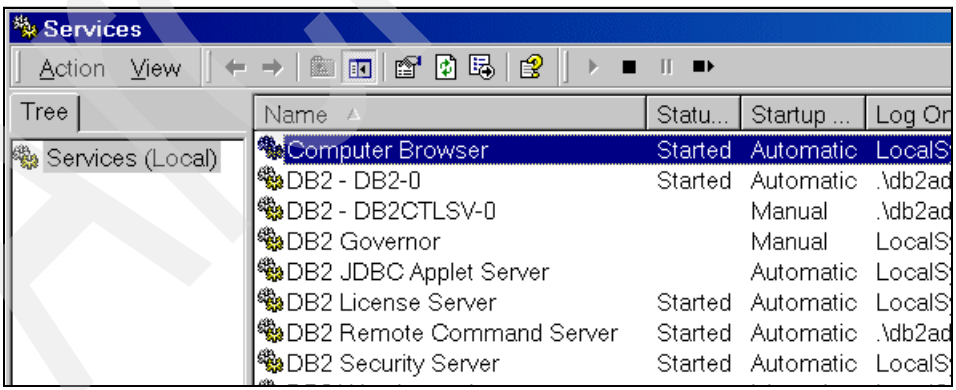

Figure 7-2 MMC Services

The properties of each individual service can be configured by highlighting the service from the Services dialog, right-clicking, and then selecting **Properties** from the pop-up menu.

The Services Recovery facility can be configured using the **Recovery** tab of the properties dialog (see Figure 7-3). It allows the user to configure recovery actions for service failures. This can include the following actions:

- Restart the service
- Run a file
- Restart the computer

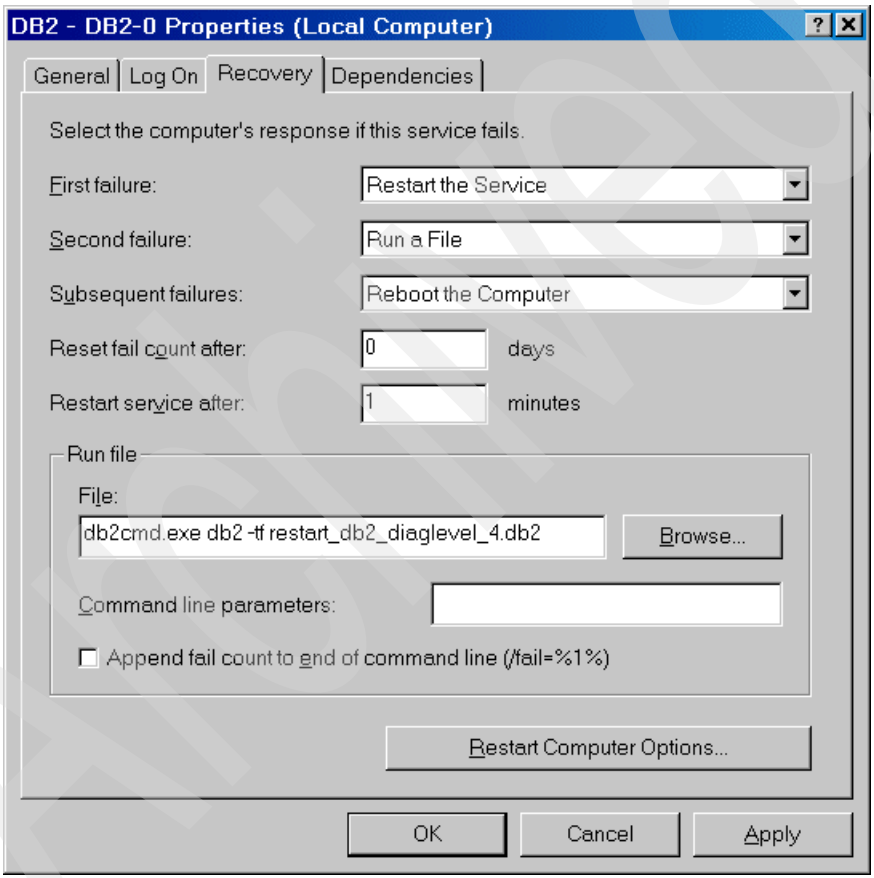

Figure 7-3 Services Recovery

In the next example (see Figure 7-4), we see that the DB2 service has been configured to automatically restart upon the first failure. Upon the second failure, a DB2 script file will be executed that will modify the DB2 diagnostic level to four (4) and restart the service. Finally, there is a last resort option to completely reboot the system. If this option is selected, you can optionally specify a time delay and a message to be broadcast.

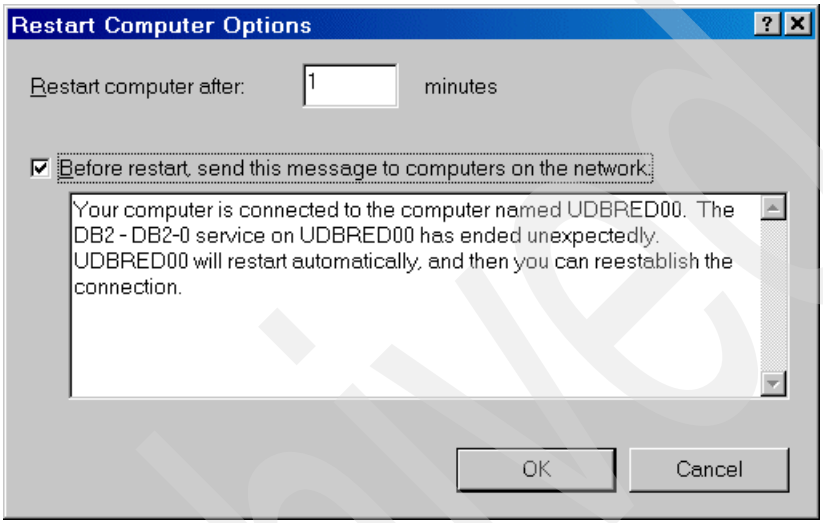

Figure 7-4 Services Recovery Restart Computer Options

7.3 Standby servers for high availability

Implementing standby servers provide for high availability by keeping a backup server in standby mode in case of a hardware or software failure to the primary database server. There are several techniques that can be used to implement a standby server.

If the primary concern is for an internal failure of the server hardware or software, the database can be placed on an externally attached storage device that can be physical detached from the primary server and re-attached to the backup server where the database can be restarted.

If the primary concern is for a complete system failure, the database can be copied from the primary server to the backup server. This can be done in one of two ways:

- The first and most common method is to take a backup image of the database from the primary server and restore it to the backup server.

- The second method is to use the feature built into the disk subsystem to mirror the database files and eventually split the mirror image and move the mirrored copy over to the standby server.

In either case, you want to keep both images in sync, that is, to have the transaction logs from the primary server applied to the standby server. To do this, you will need to leave the image of the database on the standby server in a roll forward pending state. You will then need to start shipping transaction log files from the primary database server to the backup database server and rolling the transaction logs onto the database.

7.3.1 Online Split Mirror Images

Introduced in DB2 UDB v7.2, split mirror images provides high availability support by combining features built into DB2 UDB with advances in high end storage subsystems.

The concept of mirrored images is nothing new. We have been using RAID technology for over a decade to protect data in the event of disk failures by mirror disks with RAID controllers. Today, with advances in disk system technology we are able to break mirrored disks and move images stored on these disks to other system. DB2 UDB supports this technique while keeping the database system online. This is the idea behind online split mirror images as depicted in Figure 7-5.

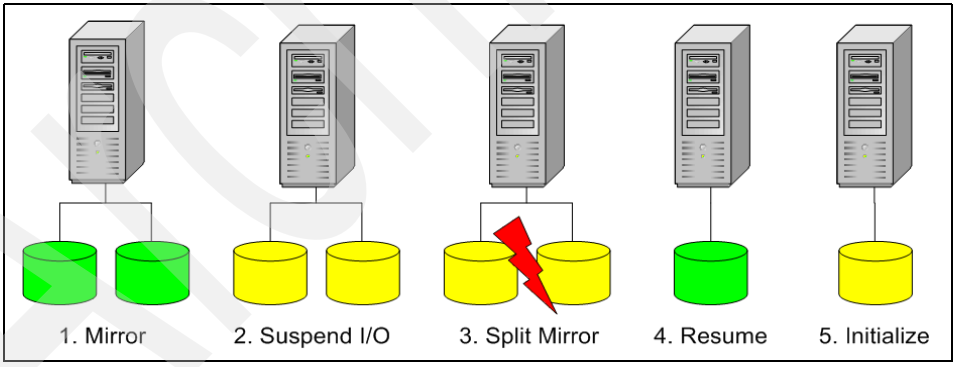

Figure 7-5 Online Split Mirror Image Overview

There are several applications for splitting mirrored images of databases and they all provide for high availability. First, you can remove the overhead associated with a database backup by splitting the mirror image and backing up the database on a separate system. Once the backup is complete, you simply re-mirror the disk.

Next, you can quickly create a idle standby server by splitting the mirror image and moving the image to the standby server. Finally, you can efficiently create a near real time copy of a database on another system for reporting or OLAP. All of these techniques are accomplished by DB2 UDB's support for online split mirror images that we discuss in the next section.

Suspending database I/O

DB2 UDB provides support for online split mirror images by temporarily suspending database I/O while the disk subsystem splits mirrored disks. After the mirrored disks images are split, DB2 UDB resumes I/O activity on the primary image of the database. Once the secondary image has been moved to another system running DB2 UDB the secondary database image is initialized. The secondary database image can be initialized in one of three ways, standby, backup, or mirror.

In order to prepare the database for breaking of the mirrored disks the database must suspend I/O to the disk subsystem. The database will remain online and continue processing transactions (select,insert,update,delete) during the suspended I/O state, however no writes to the database data (table space containers) or transaction log files will be performed. Database I/O is suspended by first connecting to the database and invoking the DB2 set write suspend command as follows:

```
db2 set write suspend for database
```

Resuming database I/O

After the mirror image has been broken by the disk subsystem, database I/O can be resumed. Database I/O is resumed by first connecting to the database and invoking the DB2 set write resume command as follows:

```
db2 set write resume for database
```

It is important to note that any applications connected to the database will remain online and continue processing transactions (select, insert, update, delete) during the suspended I/O state. However, no writes to the database data (table space containers) or transaction log files will be performed.

In the event of a database crash while the database is in the suspended I/O state, the database will need to be restarted with the resume write state so that crash recovery can be initiated and completed. This can be accomplished by invoking the DB2 restart database command with the write resume option as follows:

```
db2 restart database <database_alias> write resume
```

Initializing the database

Once the secondary database image has been moved to another system running DB2 UDB, the image must be initialized in order to reconcile discrepancies in the database transaction log files with data in the database files. This is done using the DB2 Productivity Tool Initialize Database (db2initdb.exe).

The syntax is as follows:

```
db2inidb database_alias as < SNAPSHOT | STANDBY | MIRROR >
[ RELOCATE USING config_file ]
```

Snapshot database image

The SNAPSHOT option allows the secondary database image to be initialized as a duplicate copy of the original database. The database will be initialized with a new set of transaction log file independent of the primary database image. Use this option when you want to create a secondary copy of the database to off load reporting or OLAP queries.

Standby database image

The STANDBY option allows the secondary database image to be initialized as a standby copy of the original database. The database will be initialized in roll forward pending state allowing additional database transaction log files from the primary database image to be applied to this image. The database remains in a continuous roll forward pending state until otherwise required at which point the roll forward state is completed and the database is brought online.

Note: You will need to implement log shipping from the primary database image to this secondary database image in order to continuously roll the database forward. This technique is discussed in the previous section.

Mirror database image

The MIRROR option allows the secondary database image to be initialized as a mirror image of the original database. The database will initialized in a backup pending state so that a backup image can be taken that if required can be used to restore the primary database image as if the backup image had been taken from the original database.

Relocating database image

The RELOCATE USING option is provides the functionality of the Relocate Database (db2relocatedb.exe) Productivity Tool to update internal database structures to the new location of database files. The information is provided using a plain text configuration file. More information about the Relocate Database configuration file can be found in the *Online DB2 Commands Reference*.

7.4 Clustered servers for high availability

In the world of e-business on the internet, *always on* means clustering. In general clustering refers to linking computers together with hardware and software (see Figure 7-6) so that they can work together to achieve a specific goal. One goal could be to balance the load of a system. Another goal could be to achieve scalability. Our goal, in this section of the book, is to provide high availability.

Clustering for high availability, sometimes called failover clustering, is much like the buddy system in scuba diving. The idea behind the buddy system is very simple. If your system fails (air supply) you will be down (unable to breath) for a short period of time (down time) until you are able to locate, notify, and gain access (fail over) to your buddy's system (air supply).

Clustering for high availability significantly reduces system down time caused by both unplanned and planned hardware and software failures. It is however, important to note, that even with failover clustering, you will experience down time. Just like the buddy system in scuba diving, you will be down (unable to breath) for a short period of time.

The next diagram (Figure 7-6) depicts a typical cluster configuration. The cluster name is Borrow, as in the infamous Borrow Gang. The members of the cluster are servers Bonnie and Clyde. Clients communicate over the public network to the cluster through the IP address assigned to the cluster's host name.

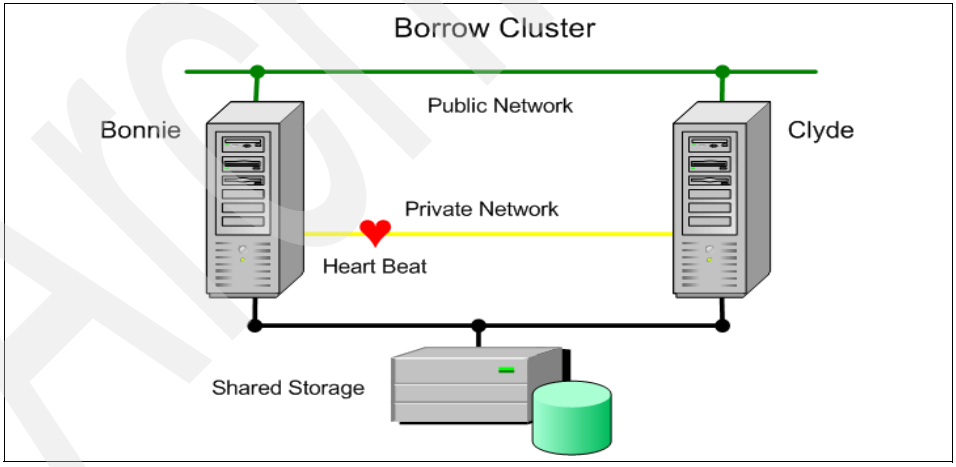

Figure 7-6 Failover Clustering Overview

The cluster's host name can only be assigned to one member of the cluster at any given time but can move to any member of the cluster. The shared storage is accessible to all members in the cluster as well, but can only be assigned to one member server at any given time. The member servers use a private network to check on the vitality of other members in the cluster. This communication is generally called the server's heart beat.

Although perhaps obvious, it is important to note that most hardware and software resources such as CPU, Memory, RAID Controllers, and Network Adapters do not fail over. The only hardware resources that fail over are the shared physical storage that is attached to both servers. It is also important to note that most software doesn't fail over either. Usually the software is installed on both servers and only the configuration information moves from one server to another by way of the shared storage.

7.4.1 Overview of Microsoft Cluster Service

Microsoft Cluster Service (MSCS) is one of many solutions for high availability on the Windows platform. Microsoft includes this optionally installed service in both Windows 2000 Advanced Server (AS) and Windows 2000 Datacenter Server (DC). The Cluster Service can be installed as part of the initial operating system load or installed at a later time. The base Windows 2000 server product does not support clustering.

Once installed and configured, high availability is achieved by providing an environment in which applications such as database management systems (DBMS) can move from one server to another in the event of a hardware or software failure.

The following DB2 UDB V8.1 products have support for MSCS:

- DB2 Universal Database Workgroup Server Edition
- DB2 Universal Database Enterprise Server Edition (ESE)
- DB2 Universal Database Connect Enterprise Edition (CEE)

Windows 2000 cluster support

Both Windows 2000 Advanced Server and Windows 2000 Datacenter Server support active/passive and active/active cluster configurations. The difference between the two is that a cluster configured as active/passive has at least one server in the cluster that remains idle. The passive server has no other responsibilities during normal day-to-day operations.

Windows 2000 Advanced Server supports a cluster of two nodes (servers). See Figure 7-7.

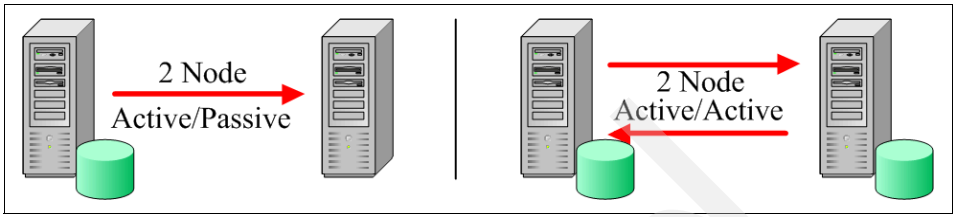

Figure 7-7 Windows 2000 Advanced Server — 2 Node Clusters

Windows 2000 Datacenter Server supports a cluster of up to four nodes (servers). See Figure 7-8.

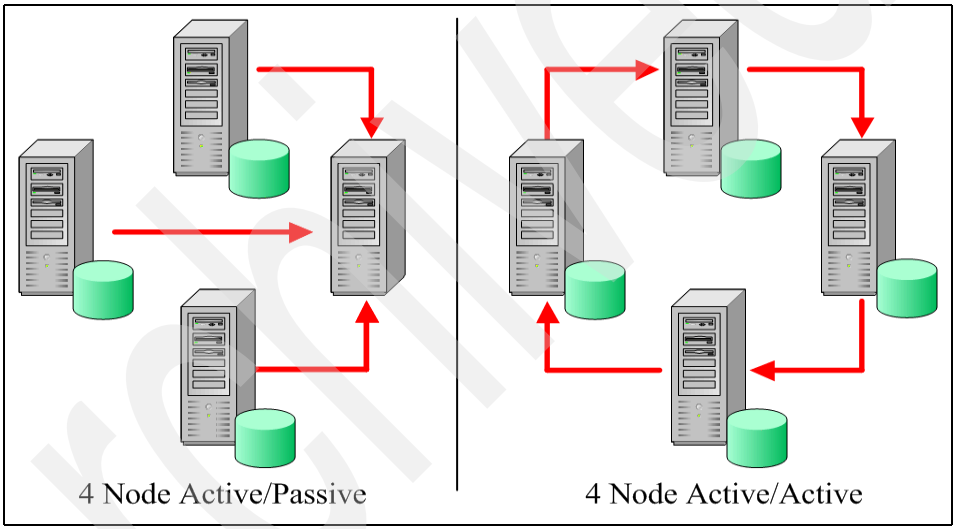

Figure 7-8 Windows 2000 Datacenter Server 4 Node Clusters

What's new with Windows Server 2003: Microsoft has announced that Microsoft Cluster Service will be included in both Windows Server 2003 Enterprise and Datacenter Editions. These high availability enhancements include support a single cluster with up to eight nodes, support for multiple clusters within a single Storage Area Network (SAN), and geographic cluster support (clusters with geographically dispersed servers).

The process of resources moving from one server to another due to a hardware or software failure is called failover. Failback, also called fallback, refers to the moving of applications or databases back to their primary or preferred server. This process can be configured to be performed immediately after the failing server is back online, or it can be deferred until a more appropriate time, as to not cause any additional outages.

High availability is also provided for outages normally associated with system maintenance. This technique, referred to as rolling upgrades, relies on clustering support for high availability when systems need to be brought down for hardware or software upgrades. Instead of bringing the system down for maintenance the clustered resources are moved to another server in the cluster temporarily while the maintenance is being performed.

The basic components consist of two servers that establish a cluster when MSCS software is installed and configured on both servers. Prior to installing the MSCS software these two servers must be able to communicate with each other over a network. Although not required, but highly recommended is a dedicated private network between the two servers that can be used to communicate each others *heart beat* without interference from traffic on the public network. Both servers must also have access to shared storage.

The MSCS software configuration process will create several default cluster resource types. These include a Network Name, an IP Address, and at least one Physical Disk that is referred to as the quorum drive and usually assigned the Q: drive letter. The network name represents the cluster's host name that is registered in DNS and assigned the IP Address. The primary purpose of the cluster's network name is to manage the cluster by a DNS name.

7.4.2 Before installing Microsoft Cluster Service

Prior to installing and configuring the Microsoft Cluster Service (MSCS) software, there are a number of pre-installation tasks that need to be addressed.

Hardware

The first item on the list is hardware. Although hardware compatibility should ideally be addressed during the planning stages, it doesn't hurt to check the hardware compatibility list prior to performing the install as you hardware configuration may have changed from the initial specifications at order time.

Microsoft maintains a Hardware Compatibility List (HCL) specifically for cluster implementations. The HCL includes both complete systems and system components that are certified for MSCS. You should verify that your system or components are on this list as part of your planning effort. For the purpose of this book, we assume all the hardware is certified, installed, and configured properly.

Shared storage

Once shared storage is physically connected to all servers in the cluster you should verify that each server can access the storage. Until Cluster Service is installed you should only boot one server at a time to test access to the storage. See Figure 7-9.

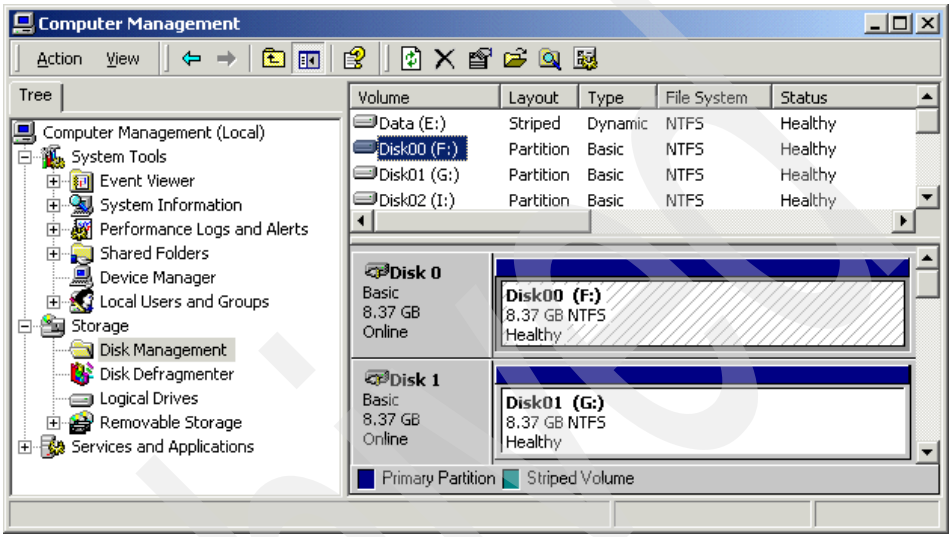

Figure 7-9 Computer Management — Disk Management

Verify that all of the shared disks that will be used in the cluster are defined as type basic, have a drive letter assigned, are formatted, and contain no mounted volumes as Cluster Services does not support dynamic disk volumes, physical disk names, mounted volumes, or raw devices. Some of these limitations can be overcome with the use of third party volume manager software.

Private network

The private network is used by MSCS as a means to communicate heartbeat information between all of the nodes in the cluster. Being the lifeline between nodes in the cluster, special attention should be given to the configuration of these network adapters.

Verify that you are not using Network Adapter Fault Tolerance or Load Balancing for the Private Network adapters. Microsoft Clustering Service does not support this type of configuration for private heartbeat communications and provides for fault tolerance by using one or more public networks as a backup to the private heartbeat network.

In the next few steps, we walk through some additional tasks that should be performed prior to installing Microsoft Cluster Services:

1. Verify the order in which your network connections are accessed by DNS and network services. Select **Start —> Setting —> Control Panel —> Network and Dial-up Connections —> Advanced —> Advanced Settings.** Verify that the public network connections are listed first. See Figure 7-10.

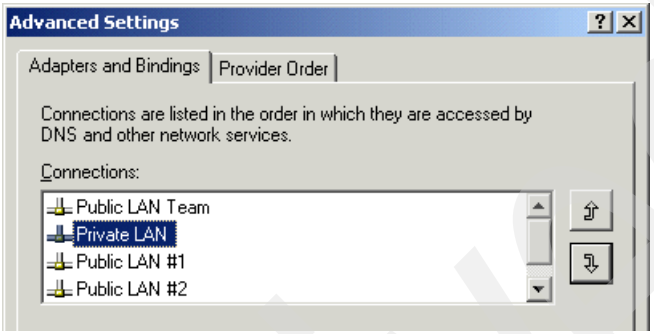

Figure 7-10 Network and Dial-up connections — Advanced Settings

2. Verify that no other client, service, or protocol network components are used by the private network connection except Internet Protocol (TCP/IP). Select **Start —> Setting —> Control Panel —> Network and Dial-up Connections —> Private LAN —> Properties**, and scroll through the list found on the Properties dialog. See Figure 7-11.

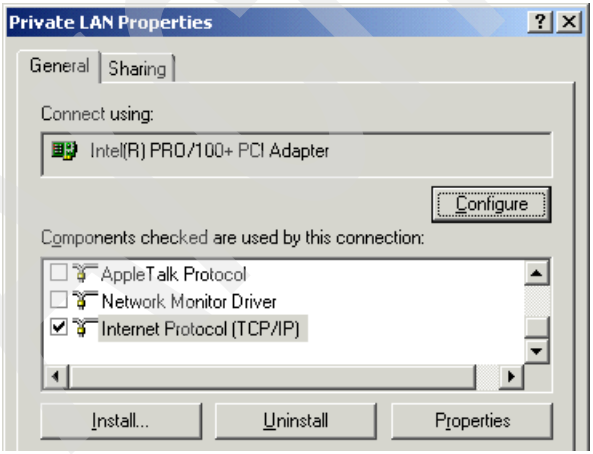

Figure 7-11 Private LAN Properties

3. Verify that your private network connection has a static TCP/IP address. Select **Start —> Setting —> Control Panel —> Network and Dial-up Connections —> Private LAN —> Properties —> Internet Protocol (TCP/IP) —> Properties**. See Figure 7-12.

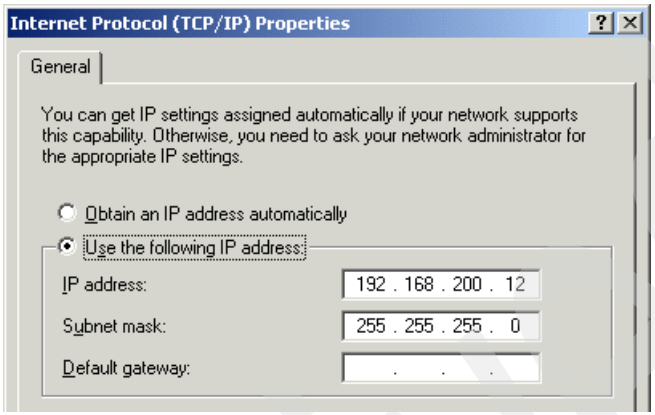

Figure 7-12 Internet Protocol (TCP/IP) Properties

4. Verify that NetBIOS over TCP/IP is disabled. Select **Start —> Setting —> Control Panel —> Network and Dial-up Connections —> Private LAN —> Properties —> Internet Protocol (TCP/IP) —> Properties —> Advanced —> WINS**. See Figure 7-13.

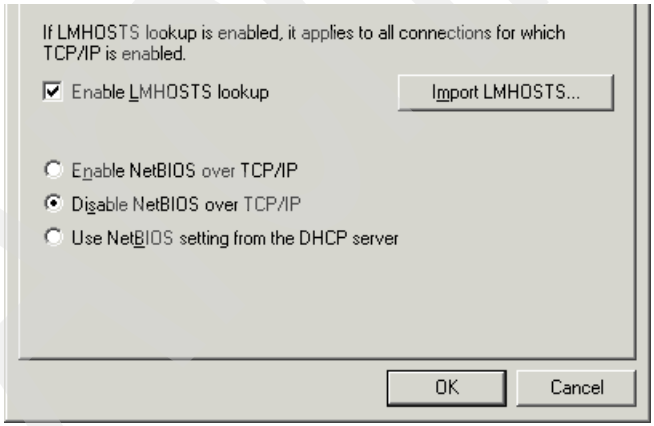

Figure 7-13 Advanced TCP/IP Settings

5. Verify that the Link Speed & Duplex on the private network adapter is set to 10Mbps/Half Duplex. Select **Start —> Setting —> Control Panel —> Network and Dial-up Connections —> Private LAN —> Properties —> Configure —> Advanced —> Link Speed & Duplex**. See Figure 7-14.

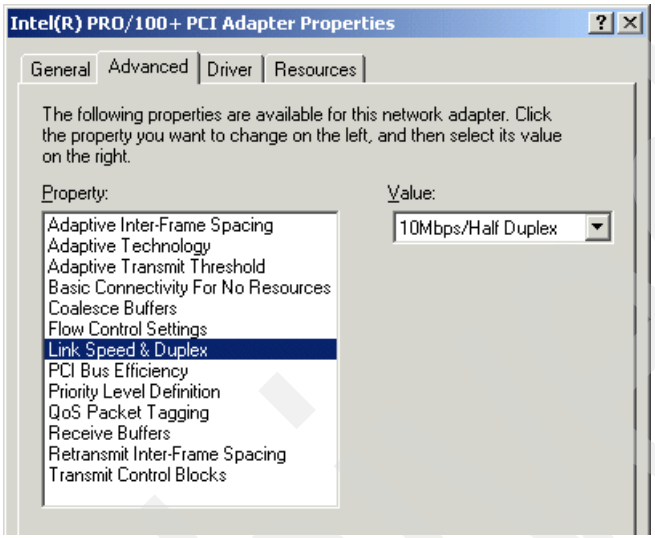

Figure 7-14 Adapter Properties — Link Speed & Duplex

7.4.3 Installing Microsoft Cluster Service

In general, the installation and configuration of MSCS is very similar to that of a partitioned database server environment. Basically, the first node in the MSCS cluster establishes the cluster, and each additional node joins the existing cluster as a member during the process of installing the MSCS software on each node.

If the Cluster Service was not installed as part of the initial operating system load you can install it by selecting **Control Panel —> Add/Remove Programs —> Add/Remove Windows Components —> Components** and the Cluster Service Configuration Wizard will start as part of this installation process.

If the Clustering Service was installed as part of the initial operating system load you can start the Cluster Service Configuration Wizard by selecting the **Control Panel —> Add/Remove Programs —> Add/Remove Windows Components —>> Configure**. See Figure 7-15.

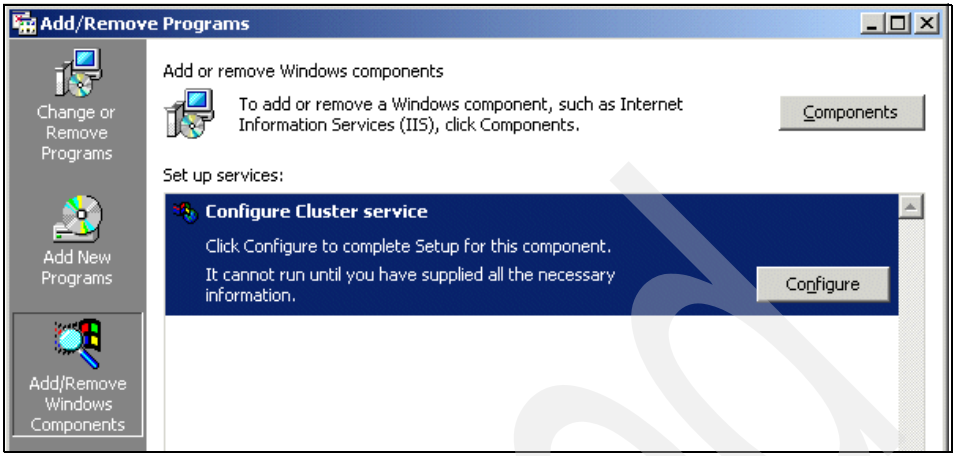

Figure 7-15 Add/Remove Programs

In the pages that follow, we walk through the installation and configuration of the Cluster Service. At this point only all servers are shut down except one.

1. Once the installation process is started the Cluster Service Configuration Wizard steps you through the complete install and configuration. See Figure 7-16.

Figure 7-16 Cluster Service Configuration Wizard

2. The first choice presented by the Cluster Service Configuration Wizard is to create a new cluster or join an existing cluster. Since this is the first node in the cluster, we create a new cluster. See Figure 7-17.

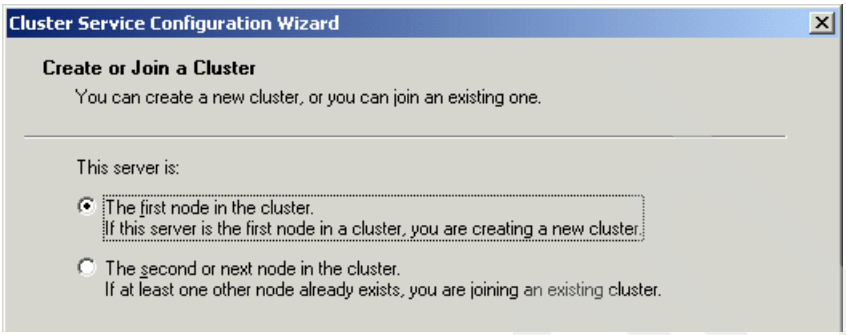

Figure 7-17 Create or Join Clusters

3. Since we are creating a new cluster, we must define the cluster name, in this case BORROW. This will be the name we use to manage the cluster either locally or remotely with the MSCS Cluster Administrator. Our cluster name will appear in the Cluster Administrator as a resource type of Network Name. See Figure 7-18.

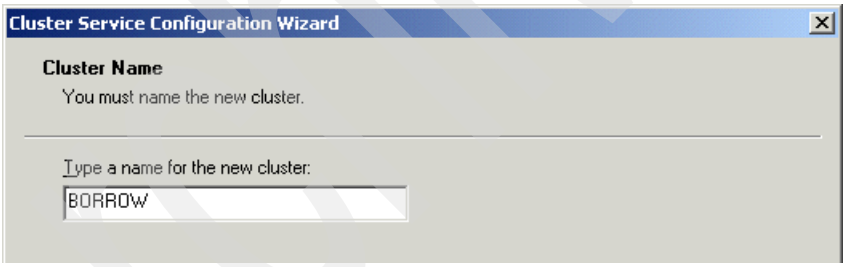

Figure 7-18 Cluster Name

4. The MSCS service requires a domain account that is a member of the local Administrators group. If you have not already done so, create a domain user account that will be used to run the Cluster Service and add this account to the local Administrators Group. If possible set the account password to never expire. Otherwise be aware that you will need to change the password within the Services MMC when the account password expires. See Figure 7-19.

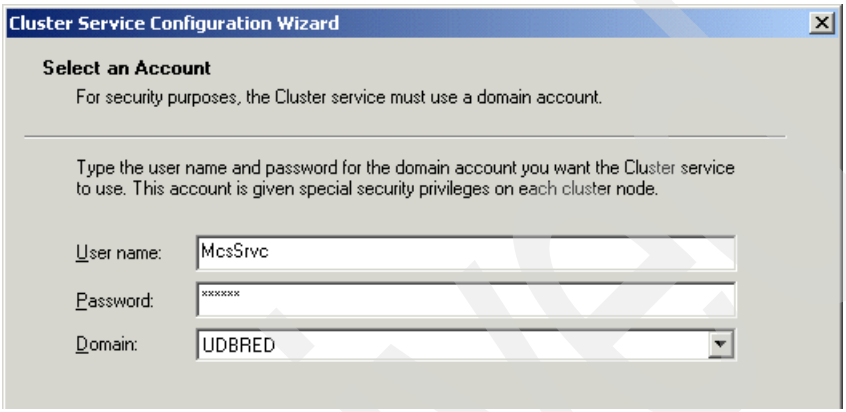

Figure 7-19 Select an Account

5. The Cluster Service Configuration Wizard presents a list of all disk on the shared storage that are supported by the Clustering Service. If your storage does not appear as expected go back to the pre-install tasks and verify the storage is configured correctly. See Figure 7-20.

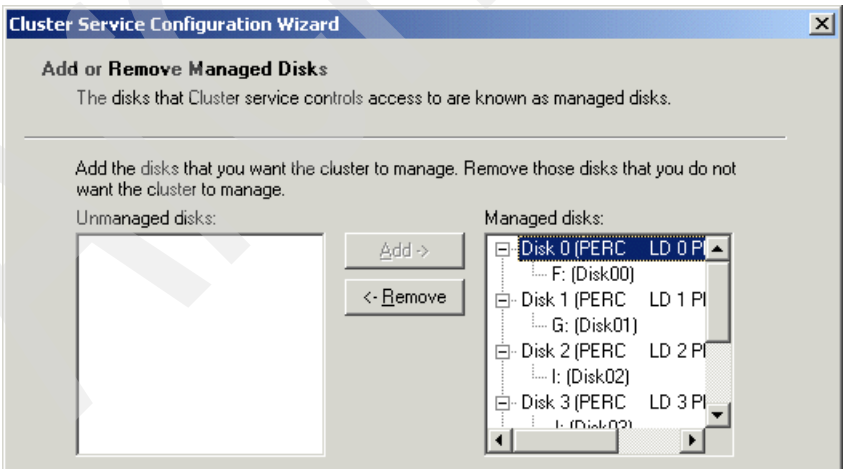

Figure 7-20 Add or Remove Managed Disks

6. Select the partition that will be used for the Quorum drive. It is a common practice, but my no means are requirement to use the drive letter Q: for the quorum drive. You should also dedicate this partition to the quorum. See Figure 7-21.

Figure 7-21 Cluster File Storage

7. Select the network connection that will be used for the private heartbeat communications. The order in which the network connections appear will depend on the order in which the network adapters are discovered during the installation of the operating system. Non-the-less when the network name for the private network appears, select Internal cluster communications only. See Figure 7-22.

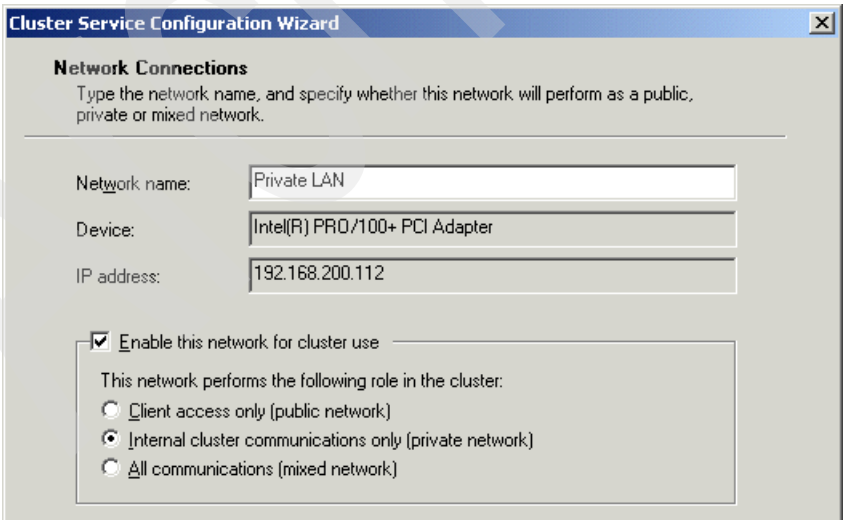

Figure 7-22 Network Connections

8. Once the first public network adapter appears, select All communications (mixed network). This will enable the public network connection to work as a backup to the private heartbeat network. See Figure 7-23.

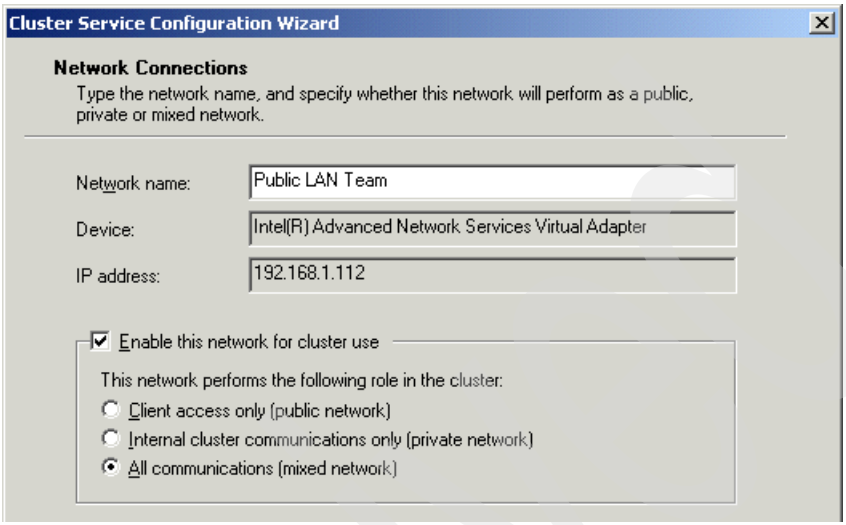

Figure 7-23 Network Connections

9. Verify that the private network connection has the priority for internal cluster communication. See Figure 7-24.

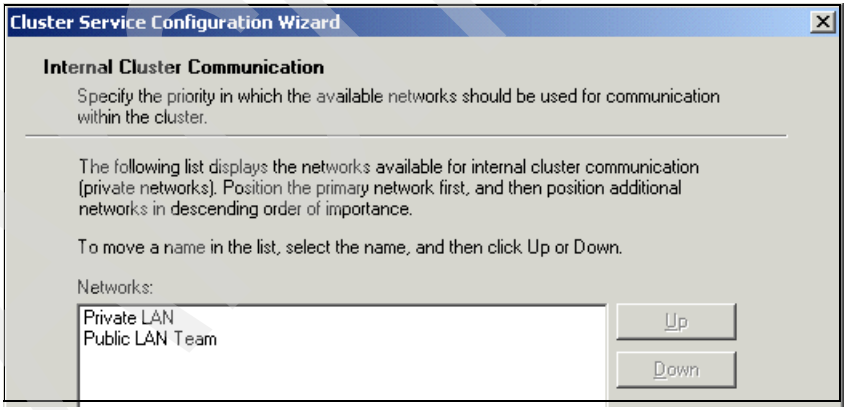

Figure 7-24 Internal Cluster Communication

10. Assign a TCP/IP address to the cluster. This address will be used to manage the cluster over the public network. As the cluster moves from one server to another this address along with the cluster name moves as well. See Figure 7-25.

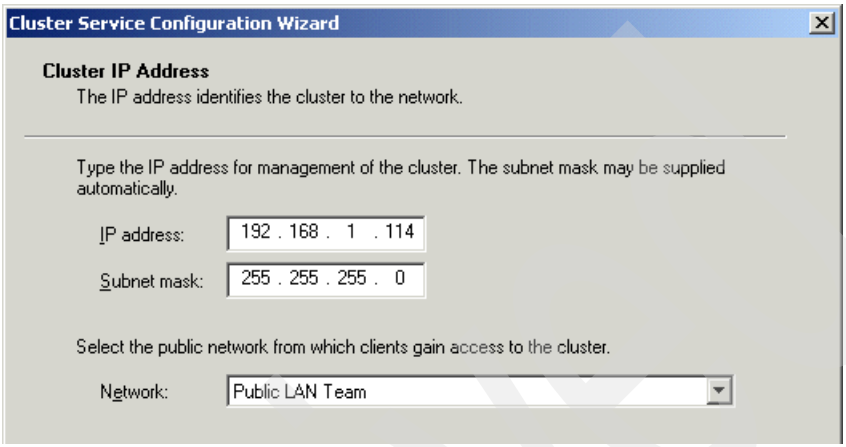

Figure 7-25 Cluster IP Address

11. If the Cluster Service Configuration Wizard is successful you will see this final confirmation that the cluster service has successfully started. At this point the shared storage is managed by the Cluster Service and the other servers can be booted. See Figure 7-26.

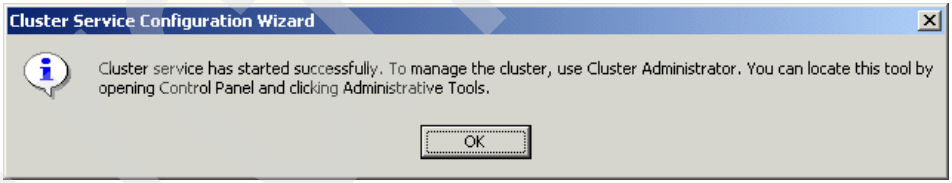

Figure 7-26 Cluster Service Configuration Wizard confirmation

12. Once the cluster has been created you can open the Cluster Administrator MMC and see the default Cluster Group containing the Cluster Name, Cluster IP Address, and Disk Q: quorum drive. See Figure 7-27.

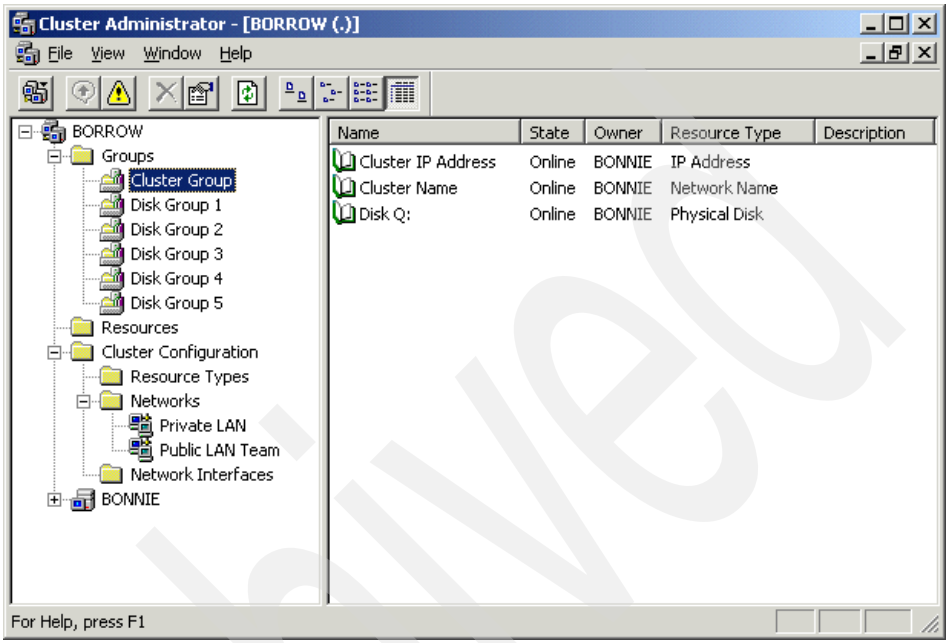

Figure 7-27 Cluster Administrator

13. Start the Cluster Service Configuration Wizard on the next server that will be added to the cluster. See Figure 7-28.

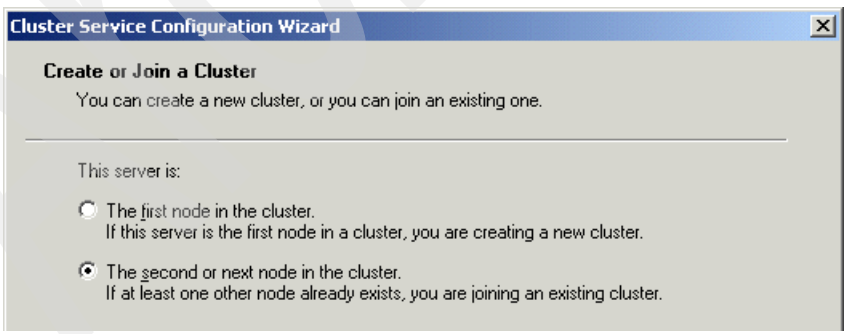

Figure 7-28 Create or Join a Cluster

14. Joining an existing cluster requires that we enter the cluster name. See Figure 7-29.

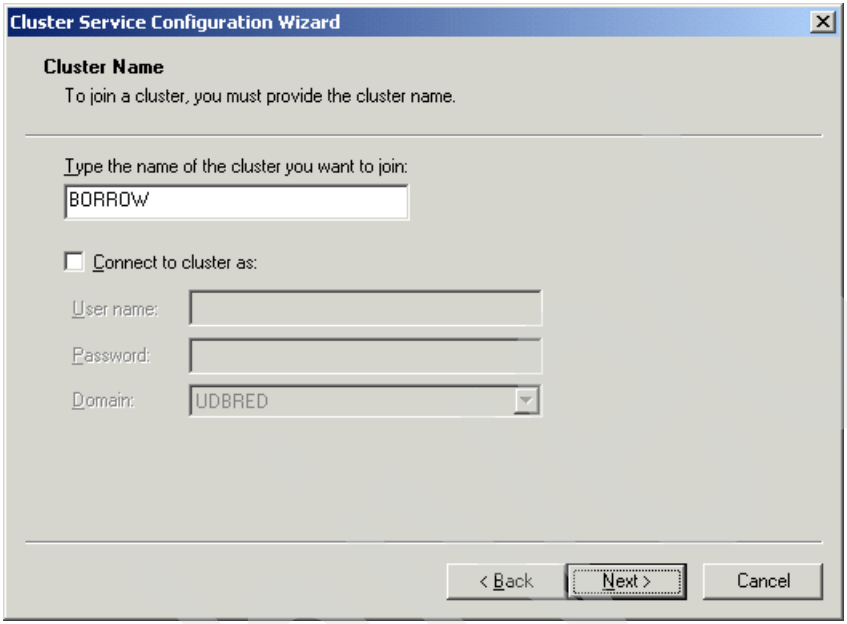

Figure 7-29 Cluster Name

15. Enter the password for the domain user account that was created to run the MSCS service and added to the local Administrators group. See Figure 7-30.

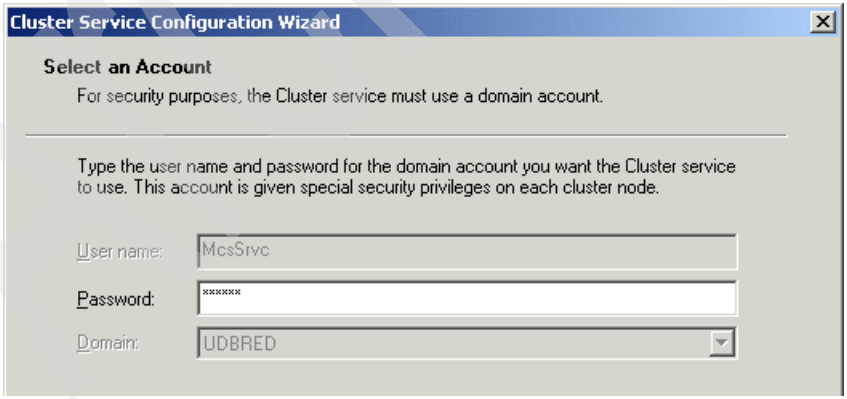

Figure 7-30 Select an Account

As each individual node is added to the cluster they will appear within the MSCS Cluster Administrator. In Figure 7-31 we can see that both Bonnie and Clyde are now members of the infamous Borrow Gang.

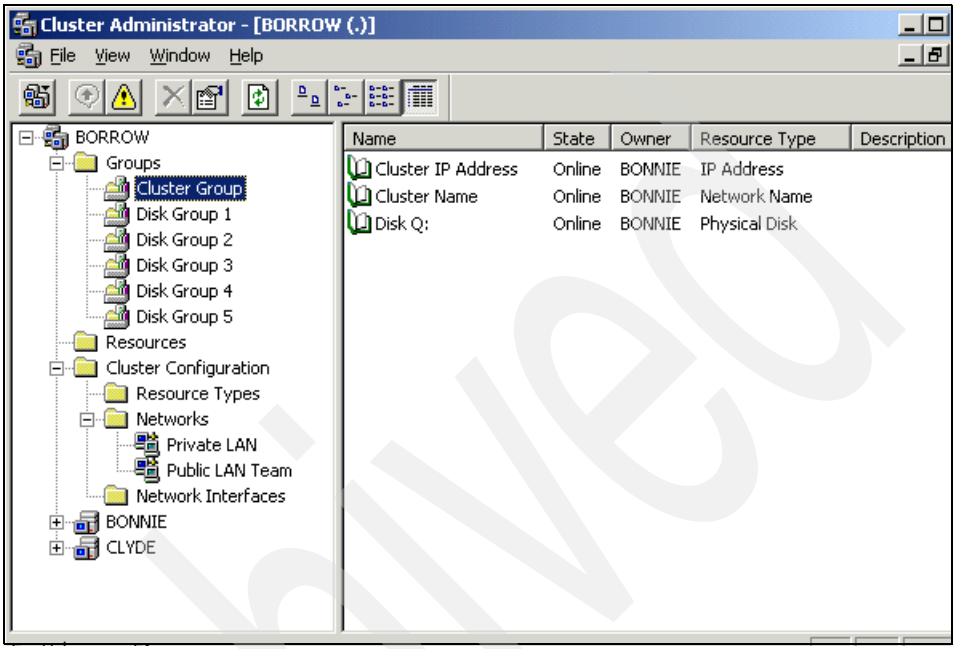

Figure 7-31 Cluster Administrator

7.4.4 After installing Microsoft Cluster Service

Once the Microsoft Cluster Service software has been installed on all nodes within the cluster we need to perform post install tasks to verify that everything is in working order. In order to prepare for these tasks we consolidate all of the resources into one group and rename the clusters quorum drive from Disk Q: to something a more meaningful name.

Consolidating the Physical Disk resources

Prior to enabling DB2 support for MSCS it is important to verify that the Cluster Service is working properly. In order to make our testing easier we move the individual Physical Disk resources into the Cluster Group and delete the Disk Group 1 through Disk Group 5. This can be done by simply selecting each physical disk resource and dragging it into the Cluster Group. The empty group can then be deleted.

Renaming the Quorum Drive resource

Next we have renamed the cluster's quorum drive from Disk Q: to Cluster Quorum. You can do this by either selecting the resource right clicking and selecting Rename or Properties from the pop-up menu. See Figure 7-32.

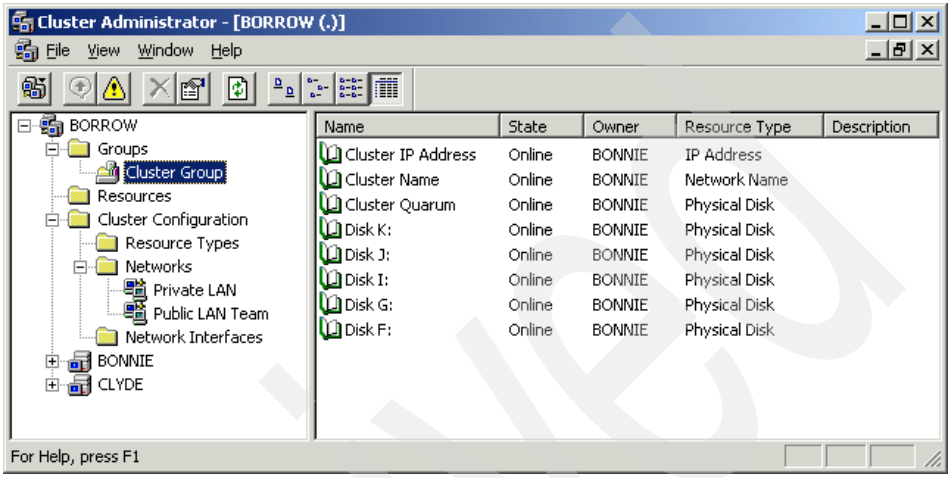

Figure 7-32 Cluster Administrator

Moving the Cluster Group

This can be accomplished by selecting the Cluster Group and moving to and from the primary and secondary owners. You should verify that you can access all of the resources within the group once the move is complete. See Figure 7-33.

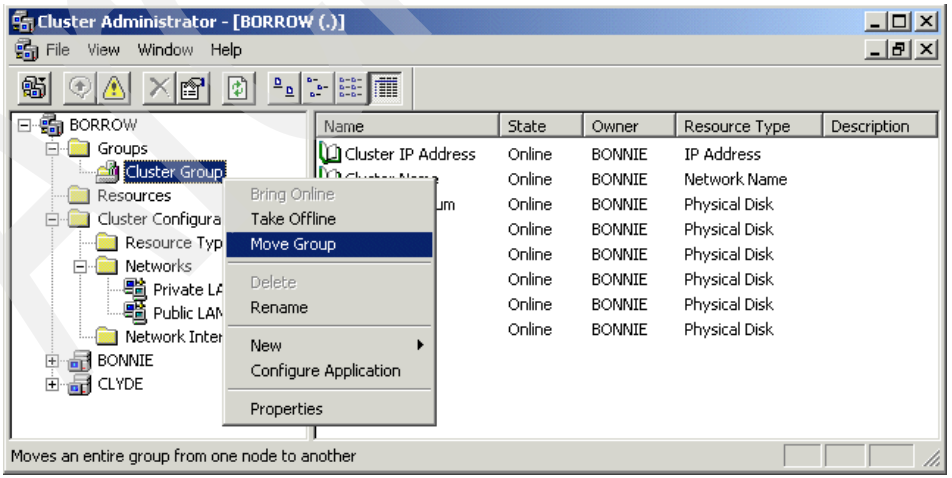

Figure 7-33 Cluster Administrator — Move Group

Initiating failure on the Cluster Group resources

In addition to moving the group, you can initiate failures on specific resources within the group. You should be aware that MSCS will first attempt to restart the resource based on the configured threshold before the initiated failure will cause the group to fail over to the secondary cluster node. See Figure 7-31.

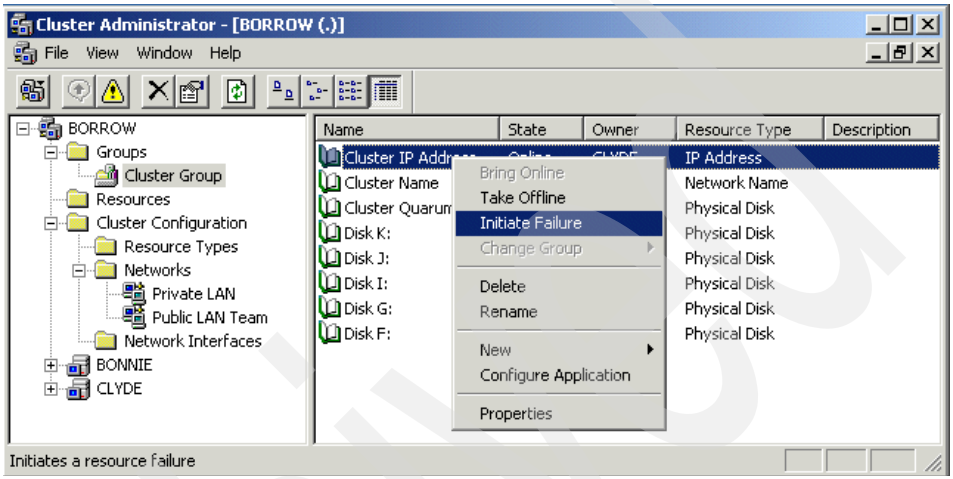

Figure 7-34 Cluster Administrator — Initiate Failure

Tip: The MSCS for Windows 2000 Advanced Server and Datacenter Server all events written to the Windows Event log on any node in a cluster are also replicated to the Windows Event log(s) of the other nodes in the cluster. You can test this by initiating a resource failure on one node and inspecting the System Log via the Event Viewer on all nodes in the cluster.

Testing the Cluster Group

The Cluster Service configuration process will create a default group called the Cluster Group. This group will have the cluster's Network Name, the clusters IP Address, and the Physical Disk used for the quorum drive. Table 7-1 is a list of the resources that appear in the Cluster Group after the Cluster Service has been installed.

Table 7-1 shows a list of tests that can be performed to verify that the Cluster Service is working properly.

Table 7-1 Resources in cluster group

Resource Name	Resource Type	Resource Value
Cluster Name	Network Name	BORROW
Cluster IP Address	IPAddress	192.168.1.114
Cluster Quorum	Physical Disk	Q: (DISK05)

Test 1: Logon to the first server in the cluster, verify that the Cluster Group is currently online at this server, open a Windows Command Prompt. Verify that you can ping the Cluster Group by IP address and name. Verify that you can access the Quorum drive (Disk Q:). Move the Cluster Group to another member in the cluster and repeat.

- Ping 192.168.1.114
- Ping BORROW
- DIR Q:

Test 2: Logon to a server that is not a member of the cluster, verify that the Cluster Group is currently online at the primary server, open a Windows Command Prompt. Verify that you can ping the Cluster Group by IP address and name. Verify that you can access the Quorum Drive. Do this while moving the Cluster Group from one member of the cluster to another.

- Ping 192.168.1.114 -t
- Ping BORROW -t
- NET USE Q: \\borrow\q$

Test 3: Logon to a client that will use the resources of this cluster, verify that the Cluster Group is currently online at the primary server, open a Windows Command Prompt. Verify that you can ping the Cluster Group by IP Address and Cluster Name. Verify that you can access the Quorum Drive. Do this while moving the Cluster Group from one member of the cluster to another.

- Ping 192.168.1.114 -t
- Ping BORROW -t
- NET USE Q: \\borrow\q$

7.4.5 Before enabling DB2 MSCS support

In this section we cover tasks that should be performed prior to enabling DB2 UDB high availability (HA) support with MSCS.

DB2 installation

DB2 should be installed on a local (non-clustered) drive on all servers that will participate in the cluster. Since the DB2 software is installed on local driver it can be installed before or after the servers are clustered.

DB2 instances

The DB2 instance is the primary DB2 resource that will be clustered. During the configuration of the instance to support MSCS, a new resource type of DB2 will be registered with MSCS. All DB2 instances will appear in MSCS Cluster Administrator as a resource of type DB2. All databases belonging to the clustered instance will be clustered as well, and therefore must completely reside on the shared storage.

The DB2 instance directory, usually \SQLLIB\DB2\, will be moved to the shared storage, the clustered instance will exist on both clustered servers, but only run on one at any given point in time. If during the clustering of the DB2 instance, another instance with the same name resides on a server within the cluster, it will be redefined as a clustered instance.

Tip: The DB2 MSCS (db2mscs.exe) system command used to cluster and/or de-cluster a DB2 instance does not update the Windows services file typically found in \winnt\system32\drivers\etc directory with the TPC/IP port numbers for the instance. If you are clustering an instance that already resides on both servers this is not an issue. If you are clustering a DB2 instance that only exists on one server you will have to manually update the services file.

DB2 services

Once a DB2 instance has been configured for high availability with MSCS it will be monitored by MSCS for availability. In order for MSCS to accomplish this, MSCS must be responsible for starting and stopping the DB2 service. You should configure the DB2 instance that will be clustered to start manually in the Windows Services dialog, see Figure 7-35.

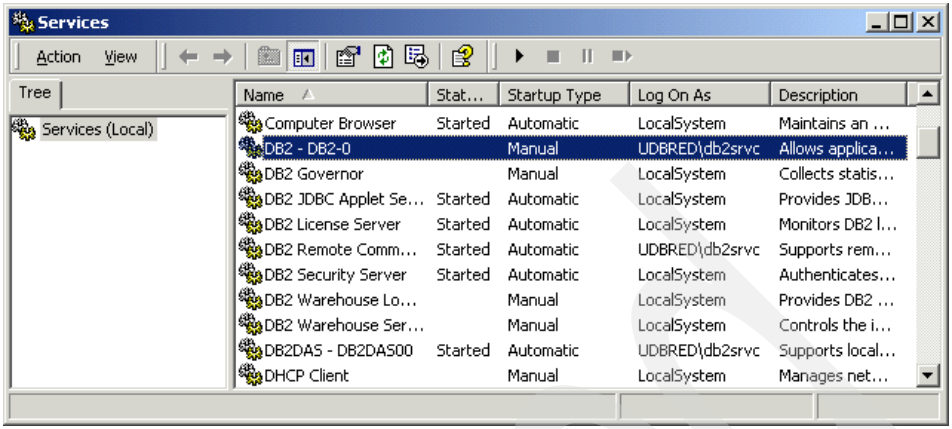

Figure 7-35 Windows Services

DB2 fallback

Under normal circumstances a DB2 instance cannot be stopped if there is at least one database in the instance with active connections. In previous versions of DB2 this was true even if the database was activated but had no existing connections. With this being the case, you will be unable to move or have DB2 fallback to the primary server if database connections are active. This would of course be desirable, in the case of an active/passive cluster where the presence of DB2 on the secondary server is not impacting any other applications.

However, if you want to be able to move or have DB2 fallback immediately to the primary cluster you will have to explicitly indicate so by setting a DB2 registry variable. This important DB2 registry variable can be used to change the default behavior of DB2 when it is clustered.

In a high availability environment you may want DB2 to fall back to the primary server as soon as the primary server is available. You may also want DB2 to move back and forth between servers for testing purposes only. To accomplish this, the DB2 registry variable DB2_FALLBACK must be set to YES, which is the default value in version 8. To set the DB2_FALLBACK value, use the following command:

```
db2set DB2_FALLBACK=YES
```

What's new in DB2: The DB2_FALLBACK registry variable can now be included in the DB2 MSCS configuration file so that it will be set as part of the clustering of the DB2 instance.

If the registry variable is undefined, then DB2 will not automatically fall back to the primary server, even if the MSCS group is configured to automatically fall back to the primary server, DB2 will refuse to fallback with the rest of the group. The group will eventually come back and join the DB2 instance.

DB2 databases

It should go without saying that if you currently have any databases in the instances that will be clustered you should back them up. If you have database in the instance that will be clustered you will need to move the database onto the shared storage.

If you have databases on both servers that reside in the same instance name, for example DB2, and your plan is to have an active/active configuration with a separate instance running on each machine, you should first create a new instance with a different instance name on one of the servers and move the database to this instance.

A database can be moved from one instance to another and from non-shared storage to shared storage by taking a backup followed by a redirected restore. You can also use a utility called DB2 Relocate Database (db2relocatedb.exe) to move the database without having to take a backup. Once you have manually move the DB2 database files, you can use this utility to update database information within the database as to the new location of database files. However, you should not use this approach until after you have a recoverable database backup image in a safe place.

7.4.6 Enabling DB2 MSCS support

In this section we walk through the steps to cluster the default DB2 instance that is created during the installation of DB2 UDB V8.1. One important concept to keep in mind relates to the changes in packaging from v7.x to v8.x.

In DB2 UDB v7.x both the Workgroup Edition (WE) and Enterprise Edition (EE) the install wizard created a non-partitioned database server DB2 instance. DB2 UDB v7.x Enterprise Extended Edition (EEE) provided the option of either a non-partitioned database server instance or a partitioned database server instance.

In DB2 UDB V8.1 Enterprise Server Edition (ESE) the install wizard does not provide an option for a non-partitioned database server instance. You now have the options to create an single-partitioned database server instance or the first partition in a partitioned database server instance. The long and the short of this is that the default instance for DB2 UDB V8.1 ESE is a partitioned database server instance. If you want to create a non-partitioned database server instance you need to use the DB2 Instance Create (db2icrt.exe) utility with the -s WSE option.

Using the DB2 MSCS system command

IBM has simplified the configuration tasks required to cluster enable a DB2 instance for high availability by implementing a single DB2 Productivity Tool called DB2MSCS (db2mscs.exe). In DB2 UDB V8.1 the DB2 Productivity Tools are now referred to simply as system commands.

What's New in DB2: In DB2 UDB V8.1 the DB2 Microsoft Clustering Utility (db2mscs.exe) has a new Uninstall feature that can be used to de-cluster a DB2 instance. The new syntax for this option is db2mscs.exe -u.

DB2 MSCS (db2mscs.exe) is a command line utility used to cluster enable a DB2 instance. This system command provides a *one-stop shopping* facility that uses a single configuration file to create and/or modify Cluster Service resources to establish cluster enabled DB2 instance:

```
db2mscs.exe [-f:config_file_name ] | [ -u: instance_name ]
```

In this statement:

-f: config_file_name: When followed by an optional name for your configuration file, by default the utility will look for a configuration file name of *db2mscs.cfg* in the current directory. The configuration file is a text file with specific configuration parameters and values.

-u: instance_name: When followed by an optional instance name, by default the utility will use the default instance value specified in the DB2INSTANCE environment variable. This new optional parameter that can be used to de-cluster a DB2 instance. During the actual clustering of the DB2 instance the db2mscs.exe utility saves a copy of your original configuration file in the DB2 instance directory. The binary file (db2mscs.bak) is used by the db2mscs.exe utility to de-cluster the instance.

Tip: The DB2 MSCS system command's uninstall (-u) option cannot be used if the DB2 Cluster has been modified outside this utility. For example, if you changed the DB2 Cluster's group name or you add another service such as the DB2DAS after clustering the instance. You will have to change it back to the original name and/or remove the DB2DAS before using the uninstall option. For this reason, you should keep a copy of your original DB2 MSCS configuration file.

Inside the DB2 MSCS configuration file

The DB2 MSCS configuration file is an ASCII text file that can be created with your favorite text editor. In this section we discuss the parameters required to cluster a non-partitioned database server. You can find two sample configuration files, db2mscs.ee and db2mscs.eee in \IBM\SQLLIB\CFG\.

Most of the parameter that end in NAME correspond to resource names that in most cases can be assigned any arbitrary value as long as it is unique within the cluster. An exception to this rule, is of course the CLUSTER_NAME, which corresponds to the name of the MSCS cluster and must already exists prior to invoking the db2mscs.exe utility. Prior to cluster enabling a DB2 instance it is a good idea to explore the resource types using the MSCS Cluster Administrator.

DB2_INSTANCE: The name of the DB2 instance to cluster enable. Only one DB2_INSTANCE parameter should be included in the configuration file. If not specified the default DB2 instance, determined by the value of the environment variable DB2INSTANCE, will be clustered.

DAS_INSTANCE: The name of the DB2 Administration Server instance. DAS_INSTANCE is a new keyword on Version 8. This parameter has a global scope and should be specified only once in the DB2MSCS.CFG file. This parameter cannot be used in conjunction with DB2_INSTANCE.

CLUSTER_NAME: The existing MSCS cluster name that the DB2 instance resources will be registered with, in our example the MSCS cluster name is BORROW. A MSCS cluster with this name must already exist and the server must already be a member of this cluster. All the resources following this parameter will be created in this cluster.

GROUP_NAME: The name of the group that will be created and assigned the DB2 instance resources, for example *DB2 Group*. Any name can be used as long as the name is unique within the cluster.

DB2_LOGON_USERNAME: The domain user account that will be used for the DB2 instance to run as a Windows service. Note that user accounts are not case sensitive. In DB2 UDB V8.1 this parameter is optional for Workgroup Server Edition (WSE) instances, but required for Enterprise Server Edition (ESE) instances.

DB2_LOGON_PASSWORD: The domain user account's password for the DB2 instance to run as a Windows service. Note that passwords are case sensitive. In DB2 UDB V8.1 this parameter is optional for Workgroup Server Edition (WSE) instances, but required for Enterprise Server Edition (ESE) instances.

IP_NAME: The name of the IP Address resource type that will be created and assigned a TCP/IP Address, Subnet, and Network connection, for example *DB2 IP Address*. Any name can be used as long as the name of the IP Address resource type is unique within the cluster.

IP_ADDRESS: The value of the TCP/IP address that the DB2 instance will use to support incoming client connections. This is the TCP/IP address value that corresponds to the IP_NAME resource type. This parameter is required if the IP_NAME parameter is specified.

IP_SUBNET: The value of the TCP/IP subnet mask that the DB2 instance will use to support incoming client connections. This is the subnet mask value that corresponds to the IP_NAME resource type. This parameter is required if the IP_NAME parameter is specified.

IP_NETWORK: The value of the TCP/IP network that the DB2 instance will use to support incoming client connections. This is the network connection name value that corresponds to the IP_NAME resource type. The value of this parameter should be specified exactly as it appears in the Cluster Administrator MMC Snap-In under Cluster Configuration Networks. If not specified, the first MSCS network detected by the system is used.

NETNAME_NAME: The name of the Network Name resource type that will be created and assigned a host name, for example *DB2 Name*. Any name can be used as long as the name of the IP Address resource type is unique within the cluster.

NETNAME_VALUE: The value for the Network Name resource type that will be used to support incoming client connections. This parameter must be specified if the NETNAME_NAME parameter is specified.

NETNAME_DEPENDENCY: The name for the IP resource that the Network Name resource depends on. Each Network Name resource must have a dependency on an IP Address resource. This parameter is optional. If it is not specified, the Network Name resource has a dependency on the first IP resource in the group.

DISK_NAME: The name of the physical disk resource type that will be used by the DB2 instance to support database files. The physical disk resources must already exist within the cluster and should be specified exactly as it appears in the MSCS Cluster Administrator MMC Snap-In.

INSTPROF_DISK: The name of the physical disk resource type that will be used for the DB2 instance directory. This parameter is optional. If not specified the DB2 instance directory is moved from local disk to the first MSCS physical disk resource found in the cluster that belongs to the same GROUP_NAME.

INSTPROF_PATH: The actual value of the fully qualified path that will be used for the DB2 instance directory, for example F:\SQLLIB. This parameter is optional. If not specified the DB2 instance directory is moved from the local disk to either the root directory of the value specified for INSTPROF_DISK or the root directory of the first MSCS physical disk resource found in the cluster that belongs to the same GROUP_NAME. If specified, the value for the INSTPROF_DISK parameter is ignored.

DB2_FALLBACK: This new V8.1 configuration file parameter can be used to enable the DB2_FALLBACK registry variable for the DB2 instance being clustered. This registry variable is discussed in the previous section.

TARGET_DRVMAP_DISK: This is an optional parameter to specify the target MSCS disk for database drive mapping. This parameter will specify the disk the database will be created on by mapping it from the drive the create database command specifies. If this parameter is not specified, the database drive mapping must be manually registered using the DB2DRVMP utility.

SERVICE_DISPLAY_NAME: This new V8.1 configuration parameter is the name of the Generic Service resource type that will be created and added to the DB2 instance's group. This parameter can be used to automate the clustering of generic services, such as the DB2 Database Administration Server. You should use the Display Name that appears in the Properties dialog for the service that is listed in the Windows Services MMC.

Warning: Clustering generic services such as the DB2 Database Administration Server insures that these services are started on the secondary node as part of the move or failover processing. They should only be clustered in an active/passive environment where there are no other DB2 instances running on the passive node. Care should also be given to remove dependencies that might bring down the DB2 database instance and/or affect the entire DB2 resource group.

SERVICE_NAME: This new V8.1 configuration parameter is the actual service name of the Generic Service resource. You can obtain the value of the service name for DB2 services, such as the DB2 Database Administration Server service via the Windows Services MMC. This parameter must be specified if the SERVICE_DISPLAY_NAME parameter is specified.

SERVICE_STARTUP: This new V8.1 configuration parameter is the service startup parameter for the Generic Service resource type.

Tip: In a non-partitioned database environment with a single clustered DB2 instance the default *Cluster Group* group, Cluster IP Address resource, and Cluster Network Name resource can be used, however, it is recommended that a new Network Name resource type be created separate from the existing clusters IP Name. This allows the DB2 instance to fail over independent of the Cluster Group.

The DB2 MSCS configuration files shown in Example 7-1 and Example 7-2 are used to cluster the DB2 instance and the DB2 Administration Server. To cluster the DB2 Administration Server, you need to drop the Administration Server on all cluster nodes except the first node by performing the following command on each machine:

```
db2admin drop
```

On the first node, you also need to set the way the Administration Server is started manually.

Make sure you run the DB2 MSCS utility from the same server where the resources are currently online. The cluster is created by invoking:

```
db2mscs.exe -f:db2mscs_db2.cfg
db2mscs.exe -f:db2mscs_das.cfg
```

Example 7-1 DB2MSCS_DB2.CFG file

```
DB2_INSTANCE=DB2
CLUSTER_NAME=BORROW
GROUP_NAME=DB2 Group
DB2_NODE=0
DB2_LOGON_USERNAME=db2srvc
DB2_LOGON_PASSWORD=db2pass
IP_NAME=DB2 IP Address
IP_ADDRESS=192.168.1.115
IP_SUBNET=255.255.255.0
IP_NETWORK=Public LAN Team
NETNAME_NAME=DB2 Network Name
NETNAME_VALUE=CHAMPION
NETNAME_DEPENDENCY=DB2 IP Address
DISK_NAME=Disk F:
DISK_NAME=Disk G:
DISK_NAME=Disk I:
DISK_NAME=Disk J:
DISK_NAME=Disk K:
INSTPROF_DISK=F:
INSTPROF_PATH=F:\SQLLIB
#
```

Example 7-2 DB2MSCS_DAS.CFG file

```
#
# Cluster DB2 Administration Server service
#
DAS_INSTANCE=DB2DAS00
DB2_LOGON_USERNAME=db2srvc
DB2_LOGON_PASSWORD=db2pass
CLUSTER_NAME=BORROW
GROUP_NAME=DB2 Group
DISK_NAME=Disk F:
```

You can watch the progress of the DB2 MSCS utility using the MSCS Cluster Administrator. After creating the DB2 Group, it will creating the IP Address and Network Name followed by moving the physical disk resources into the group and finally creating the DB2 instance resource. If any part of the configuration fails all changes will be backed out by the DB2 MSCS utility.

The next screen, Figure 7-36, shows the MSCS Cluster Administrator after the DB2 instance has been clustered. In addition to all the resources specified in the configuration file, a File Share resource type has been added. This resource is a requirement of a partitioned database server instance.

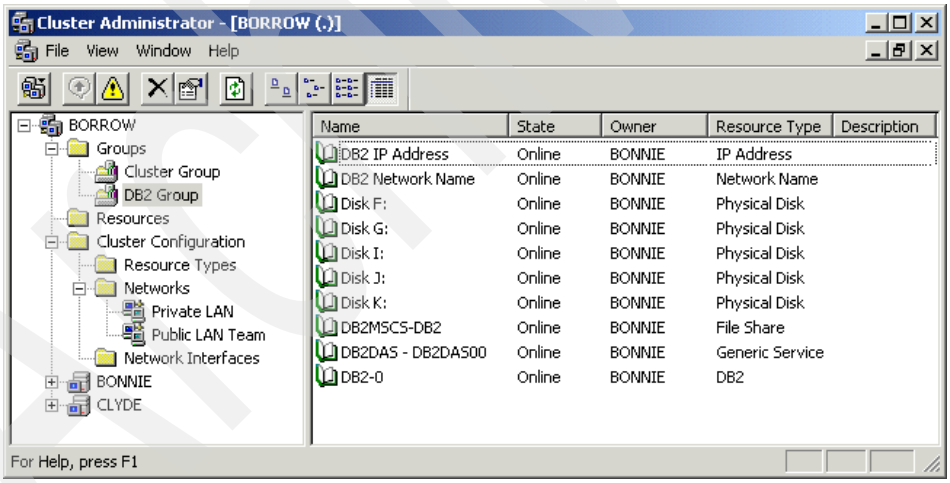

Figure 7-36 Cluster Administrator — DB2 Group

7.4.7 After enabling DB2 MSCS support

There are a few minor configuration tasks that should be performed after the DB2 Group has been successfully created. First the DB2 Group's preferred owners should be defined by opening the group's properties dialog. On the general tab modify the preferred owners to include all of the servers in the group, in order of preference. See Figure 7-37.

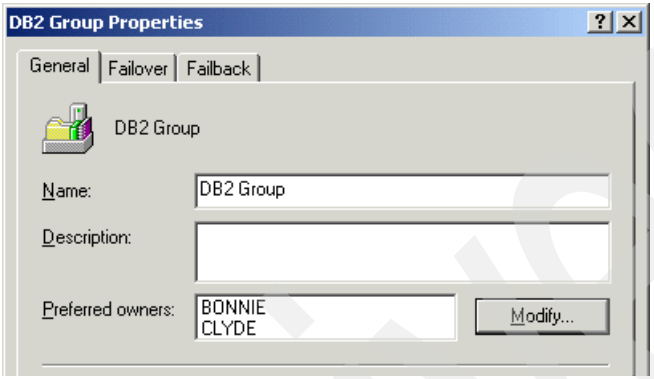

Figure 7-37 DB2 Group Properties

DB2 Administration Server considerations

If you elected to cluster the DB2 Administration Server as a Generic Service resources type you may want to consider changing the default properties for this resources. As by default the DB2 instance has a dependency on the DB2 DAS service that could bring the entire cluster down in the event of a DB2 DAS service failure. In the next couple of pages we walk through the process of doing this.

Modify DB2 dependencies

In order to remove dependencies from one resource on another resource the resources must be offline. You can do this by selecting **DB2 Group —> Take Offline.** Modify the DB2-0 resource dependencies. You can do this by selecting **DB2-0 —> Properties —> Dependencies —> Modify** and then removing the DB2DAS - DB2DAS00 resource from the Dependencies list on the right hand side of the Modify Dependencies dialog, see Figure 7-38.

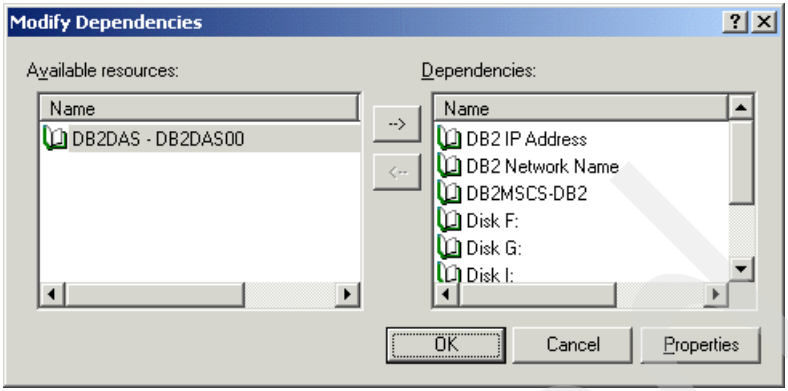

Figure 7-38 Modify Dependencies

Modify DAS restart options

In order to prevent a failure in the DB2 DAS service from affecting the entire DB2 Group you must modify the restart options for the DB2DAS - DB2DAS00 generic resource. You can do this by selecting **DB2DAS - DB2DAS00 —> Properties —> Advanced** then deselect the **Affect the group** option. Removing this option will prevent failures of this one service from affecting the entire DB2 Group. See Figure 7-39.

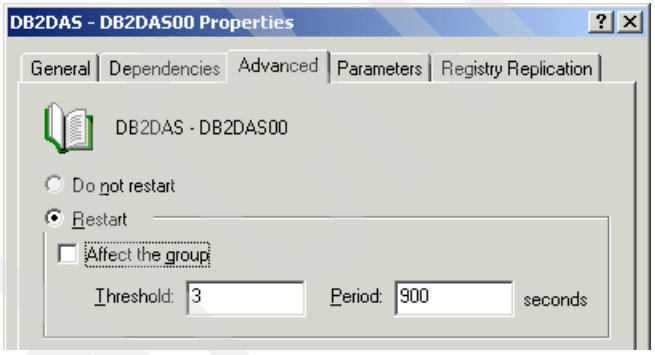

Figure 7-39 DB2DAS — DB2DAS00 Properties

Implementing post-failover scripts

The implementation of the DB2 resource type in Microsoft Cluster Service provides for the execution of post-failover processing. Post-failover processing can be performed either before the DB2 resource is brought online (pre-online) or after the DB2 resource is brought online (post-online) or both.

Implementing pre-online and post-online processing simply requires that you place a pre-online or post-online Windows Shell (wShell) script in the DB2 instance directory and the MSCS will automatically execute it. These scripts come in handy during testing of the clustered DB2 instance.

Pre-Online Scripts (db2cpre.bat)

Pre-Online processing occurs after failover immediately before the DB2 instance is brought online. This processing can be accomplished by creating a Windows Shell script that must be called *db2cpre.bat* and placed in the DB2 instance directory. The MSCS executes these scripts from the \winnt\system\ directory so you will need to fully qualify any stdout as in the Example 7-3 below.

In an Active/Active clustered environment, the pre-online processing can be used to adjust DB2 resources utilization that might have otherwise been optimized for a dedicated server.

Example 7-3 Pre-Online Script (db2cpre.bat)

```
NET SEND CHAMPION DB2 Instance arrived at %COMPUTERNAME% >>
F:\sqllib\db2\db2cpre.log
```

Post-Online Scripts (db2cpost.bat)

Post-online processing occurs after failover immediately after the DB2 instance is brought online. This processing can be accomplished by creating a Windows Shell script that must be called db2cpre.bat and placed in the DB2 instance directory. The MSCS executes these scripts from the \winnt\system\ directory so you will need to fully qualify any stdout as in the Example 7-4 below.

One very common task to perform after the DB2 instance is online (started) is to activate the one or more databases in the DB2 instance.

Example 7-4 Post-Online Script (db2cpost.bat)

```
NET SEND CHAMPION DB2 Instance online at %COMPUTERNAME% >>
F:\sqllib\db2\db2cpost.log
DB2CMD -c DB2 ACTIVATE DATABASE SAMPLE >> F:\sqllib\db2\db2cpost.log
```

In addition to pre-online and post-online scripts, DB2 UDB Version 8 provides the ability to execute two scripts, *db2apre.bat* and *db2apost.bat*, before and after each DB2 partition is brought offline. These scripts are also referred to as *pre-offline* and *post-offline* scripts. These batch files are optional. They do not exist by default and will only get executed if they exist. These batch files are launched by the MSCS Cluster Service and are run in the background. The script files must redirect standard output to record any output as a result of commands run from within the script file.

DB2 client application considerations

Although the Microsoft Cluster Service dramatically reduces downtime by providing hardware and software redundancy via a cluster of DB2 servers, it does not eliminate downtime all together. There will be a short amount of downtime that will occur during the failover processing that occurs when the DB2 instance is brought offline on the failing node and until it is brought online again.

Databases that belong to DB2 instances that are cluster enabled should be cataloged via the Network Name resource that was created during the clustering of the DB2 instance. In our example we created a MSCS group called the DB2 Group and the Network Name resource created in this group is called CHAMPION. See Example 7-5 below.

In order to connect to databases that belong to the DB2 instance in the DB2 Group, we must use CHAMPION as the DB2 node name. If we catalog the database using BONNIE or CLYDE we will not be able to connect to the database if the DB2 Group is not online at the node. If we use the BORROW network name we will not be able to connect the database if the DB2 Group moves to a node independent of the Cluster Group.

Example 7-5 DB2MSCS_DB2.CFG file

```
DB2_INSTANCE=DB2
CLUSTER_NAME=BORROW
...
NETNAME_NAME=DB2 Network Name
NETNAME_VALUE=CHAMPION
NETNAME_DEPENDENCY=DB2 IP Address
...
```

It should go without saying that database connections to a database that belongs to a clustered DB2 instance will be lost when the DB2 instance is brought offline. In-flight transactions that have not been committed will be rolled back by DB2 when the database is restarted at the secondary node.

Note: An excellent document regarding DB2 UDB clustering on MSCS *DB2 Implementing IBM DB2 Universal Database V8.1 Enterprise Server Edition with Microsoft Cluster Server* can be found on the Web site:

http://www-3.ibm.com/software/data/pubs/papers/#esemcs

7.5 Windows Datacenter Program for high availability

The Windows Datacenter Program is designed for businesses that require the most scalable, reliable, and available enterprise ready systems. This program provides customers with a complete solution that includes rigorously tested hardware, software, and support services with at minimum 99.9% availability.

Microsoft has teamed with industry-leading server manufacturers to develop the Windows Datacenter Program. This is a unique model, in that the operating system can only be purchased already loaded on third-party hardware. Only OEMs who have passed their hardware through rigorous testing are certified by Microsoft to license and support Datacenter Server.

Hardware

All OEM hardware must be submitted to the Windows Hardware Quality Labs (WHQL) and pass the Datacenter Server Program's Hardware Compatibility Test (HCT) before being placed on the Datacenter Server Program's Hardware Compatibility List (HCL). And then only the exact configuration submitted is HCL approved.

Microsoft requires a 99.9% guaranteed uptime of all Datacenter Server Program OEM vendors. This equates to no more than 8 hours or less of unplanned downtime in a twelve (12) month period. Some OEM vendors provide an additional optional 99.99% uptime guarantee, but usually require a clustered failover server solution. The Unisys ES7000 Server is currently the only server providing 32-way SMP support.

In addition to the OEM Vendors that provide server solutions for the Datacenter Server Program, Datacenter Infrastructure Vendors (DIV) provide infrastructure components such as Storage Area Networks.

Software

All OEM vendor software that runs on the Windows Datacenter Server platform must be certified. IBM Corporation is a Microsoft Gold Certified Partner and DB2 Universal Database was the first database product to be Certified for Windows 2000. It was certified for Windows 2000 Datacenter Server in June of 2001.

Services

In addition to providing hardware and software, OEM vendors that participate in the Datacenter Server program must provide Datacenter Server services, which include some of the following:

- Installation Configuration Services
- Support Services
- Joint Support Queue
- Change Control Management Service
- Reliability Measurement Services

The OEM vendor is required to install and configure Windows 2000 Datacenter Server. This configuration must pass an availability assessment test. They are required to provide 24x7 support services with onsite service guarantees.

A Joint Support Queue is staffed by both OEM and Microsoft personnel to ensure tight collaboration between the hardware and operating system vendors and has access to all OEM Datacenter HCL hardware configurations for problem reproduction and isolation. Customers have a choice between the OEM or Microsoft as first contact into the JSQ.

The OEM vendor must control and manage any changes to the Windows 2000 Datacenter Server operating system including Windows service packs, kernel level drivers, and hot fixes. The are also required to provide reliability measurements back to Microsoft in the form of Windows Event Logs, Dr. Watson, Blue Screen of Death, and Crash Dumps.

8

Application development

The requirements for application development have been accelerated in the past few years. Today, most applications are intended to integrate middle range servers, legacy systems, and newer technologies available in a company. Application developers also face the challenge of developing the applications with less resources and shorter time frames. These challenges exist in every company, from small shops to big enterprises. IT professionals must be ready to use the state-of-art technology as fast as they can.

DB2 is capable of dealing with many environments and platforms. DB2 is accessible from many standard interfaces. DB2 is also bundled with many tools to help the developers on various tasks, leaving more room to the developer to concentrate on development tasks.

In this chapter we describe the basic concepts and tools available to do application development with DB2. We cover these topics:

- The basic development tools available to the developer
- The common languages available on the Windows platform, their recommendations, and best practices.
- Application development tips

8.1 DB2 developer tools

To help in the development of DB2 applications, there are some very useful tools, which we describe in the following sections.

8.1.1 Development Center

Development center is a new addition to the DB2 Version 8 tools group. Development Center includes:

- Stored Procedure Builder
- UDF Builder

Stored Procedure Builder

In this section we show how to use Stored Procedure Builder to code and debug stored procedures and the requirements to run it. You can find more information on *Chapter 5 SQL Procedures, DB2 Application Development Guide: Building and Running Applications Version 8.*

Configuring the C++ compiler

DB2 SQL store procedures rely on an external C++ compiler to compile and link the C program generated by the builder.

SQL stored procedures are supported on the following Windows operating systems:

- Windows NT
- Windows 2000
- Windows XP
- Windows Server 2003

The following two complier environment variables on the server have to be set for developing SQL stored procedure:

- DB2_SQLROUTINE_COMPILER_PATH
- DB2_SQLROUTINE_COMPILE_COMMAND

If the environment variables for your compiler are set as SYSTEM variables, no configuration is needed. But we recommend using the procedures to set if you feel more comfortable on just dealing with the DB2_SQLROUTINE_COMPILE_COMMAND variable.

Restrictions

A problem exists with 64-bit Windows in setting the variable DB2_SQLROUTINE_COMPILER_PATH. This variable requires the complete

path to a file and does not allow arguments/switches. It would not work if a user specified the following:

```
db2set DB2_SQLROUTINE_COMPILER_PATH="C:\MsSdk64\SetEnv.bat /XP64 /RETAIL"
```

The work around is to create another batch file that calls Microsoft's setup batch file with the appropriate flags, for example:

```
db2set DB2_SQLROUTINE_COMPILER_PATH="C:\MsSdk64\SetEnvXP64.bat"
```

where the contents of C:\MsSdk64 \SetEnvXP64.bat would be:

```
call C:\MsSdk64\SetEnv.bat /XP64 /RETAIL
```

This problem does not exit on Windows 32-bit environments. The vcvars32.bat neither requires nor accepts any parameters.

Procedure

We assume that the C++ compiler has already installed on the C: drive. First step is to open a command prompt window and set the DB2_SQLROUTINE_COMPILER_PATH DB2 registry variable as follows:

For Microsoft Visual C++ Version 5.0:

```
db2set DB2_SQLROUTINE_COMPILER_PATH="c:\devstudio\vc\bin\vcvars32.bat"
```

For Microsoft Visual C++ Version 6.0:

```
db2set DB2_SQLROUTINE_COMPILER_PATH="C:\Program Files\Microsoft Visual
Studio\VC98\Bin\vcvars32.bat"
```

The compile command sample for Microsoft Visual C++ Versions 5.0 and 6.0 is as follows:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND="cl -Od -W2 /TC -D_X86_=1
-I%DB2PATH%\include SQLROUTINE_FILENAME.c /link -dll
-def:SQLROUTINE_FILENAME.def /out:SQLROUTINE_FILENAME.dll %DB2PATH%\lib
\db2api.lib"
```

Here the keyword SQLROUTINE_FILENAME should be replaced with the actual filename of the generated SQC, C, PDB, DEF, EXP, messages log, and shared library files.

Following are some tips for configuring the C++ compiler:

- Make sure the file vcvars32.bat is exist on the specified directory.
- If the directory name has space, double quote is required on the path. Without the double quote, db2set will interpret the rest part of the path as a parameter and probably will fail.

- If the SQLLIB path on server contains directory name with spaces, check how the path is specified in DB2PATH variable. It is recommended to replace the space in the directory name with tilde. For example: db2set DB2PATH=C:\Progra~1\IBM\SQLLIB. Another way to avoid problem is to change the path to the one without spaces during the server installation time, for example C:\SQLLIB.

Here is an example of the environment setup commands for the Microsoft Visual C++ Version 6.0 compiler (Example 8-1). The following commands can be executed by cutting and pasting them into a batch file and running the file in a DB2 command window. Be sure to make all necessary changes, including path settings, for your particular environment:

Example 8-1 Simple batch file to run that includes both variables configurations

```
@echo on
rem Setting the SQL PROCEDURE environment:
db2set DB2_SQLROUTINE_COMPILER_PATH="C:\Program Files\Microsoft Visual
Studio\VC98\Bin\vcvars32.bat"
db2set DB2_SQLROUTINE_COMPILE_COMMAND="cl -Od -W2 /TC -D_X86_=1
-I%DB2PATH%\include SQLROUTINE_FILENAME.c /link -dll
-def:SQLROUTINE_FILENAME.def /out:SQLROUTINE_FILENAME.dll %DB2PATH%\lib
\db2api.lib"
@echo off
```

Get and Put stored procedure routines

The GET and PUT stored procedure commands allow you to distribute compiled SQL stored procedures.

When you define an SQL stored procedure using the Stored Procedure Builder, it is in C program format. A C or C++ compiler is required to compiled and link the C program into executable code. This executable code can be distributed to other DB2 database server which has same operating system and DB2 version regardless if the machine has C or C++ compiler.

DB2 provides utilities to extract and install stored procedure under command line interface and programming interface. The command line interface consists of two CLP commands: GET ROUTINE and PUT ROUTINE. The programming interface consists of two built-in stored procedures: GET_ROUTINE_SAR and PUT_ROUTINE_SAR. For more information on the command line interface, refer to the Command Reference. For more information on the programming interface, refer to the SQL Reference.

To distribute a compiled SQL stored procedure from one database server to another database server, perform the following steps:

- Build the application, including defining the SQL stored procedures that are part of the application.
- After the procedures are tested, extract the compiled version of each procedure into a different file, either by issuing the GET ROUTINE command or by invoking the GET_ROUTINE_SAR stored procedure. Copy the files to your distribution media (if necessary).
- Install the compiled version of each procedure on each server, either by issuing the PUT ROUTINE command, or by invoking the PUT_ROUTINE_SAR stored procedure, using the files created in the previous step. Each database server must have the same operating system and DB2 level.

Note: The Application Development Client must be installed on the target machine in order for the PUT ROUTINE command or the PUT_ROUTINE_SAR stored procedure to work

Creating, debugging, and deploying a stored procedure

In this section we develop a simple stored procedure to demonstrate the stored procedure development process from development stage to production stage. We start with the default stored procedure built by DB2 Stored Procedure Builder and then modify it to suit our needs. You can find a step-by-step explanation of launching the Development Center wizard on Chapter 3, “Post-installation tasks” on page 95.

1. Open **Start Menu—>Programs—>IBM DB2—>Development Tools—>Development Center**. The Development Center launchpad wizard will appear. Click the upper right **X** button on the window to close the wizard.
2. Now you should have the **Development Center** dialog open. The first step is to create stored procedure project. Right-click the Projects folder icon and click **New**. See Figure 8-1.

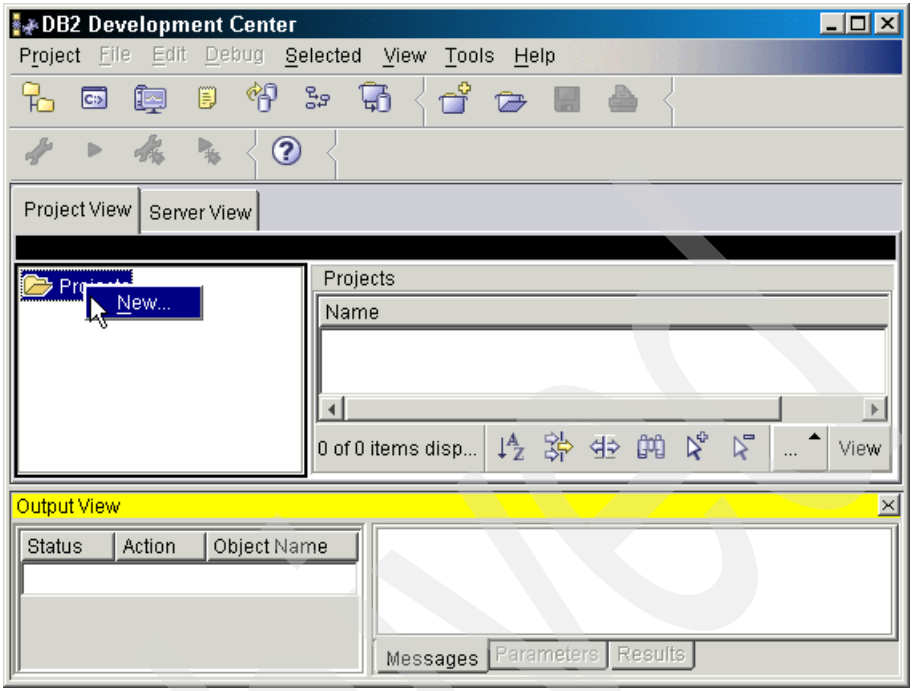

Figure 8-1 Creating a project

3. On the next dialog, fill in the project name and path text boxes and then click **OK**. In our case, we leave with the default name Project1 and default path.
4. Now we create a database connection. Right-click over the **Database Connection** folder and click **Add Connection** (Figure 8-2).

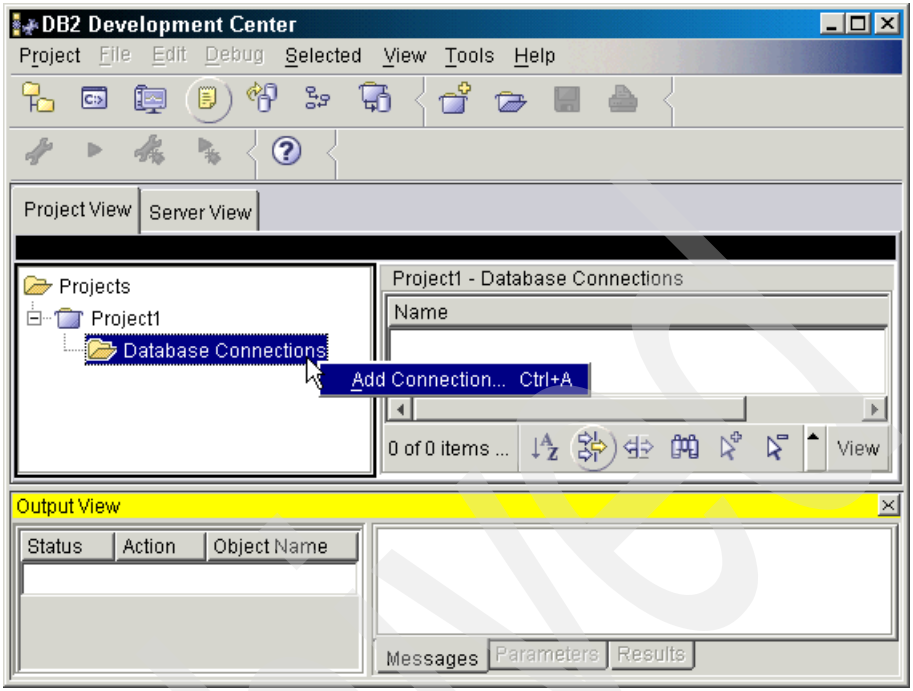

Figure 8-2 Adding a connection

5. On the next dialog, leave the **Online** option checked and click **2.Connection**. The connection wizard will ask you about the user information (Figure 8-3). Choose the desired database by clicking the **Alias** combo box or add a new one by clicking the **Add** button. Enter the user and the password on the spaces below. You can leverage your current user to connect to the database by clicking the **Use your current userid and password** check box. You can check the user information by clicking the **Test Connection** button. Click **Next** to proceed.

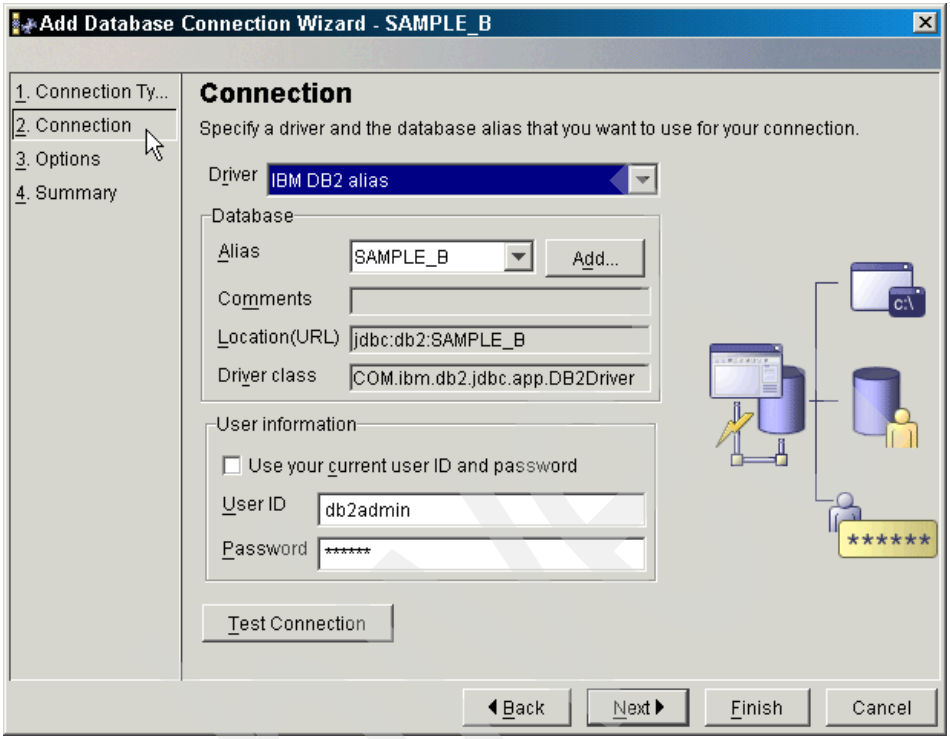

Figure 8-3 Filling in database connection information

6. If you want to change the default schema, over-write the **SQL schema or SQL ID** text box and then click **Finish** (Figure 8-4).

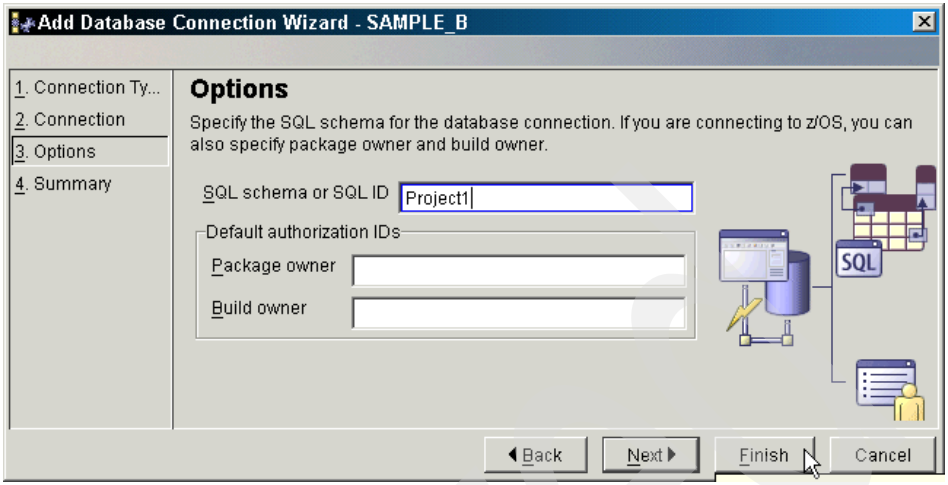

Figure 8-4 Defining the schema and finishing database connection wizard

7. You should be brought back to the main dialog of DB2 Development Center. Click the plus signal at the left side of database icon tree if it is not opened. To create a stored procedure, right-click the stored procedure icon at the left panel, select **New** menu and then select **Stored Procedure Using Wizard** menu (Figure 8-5).

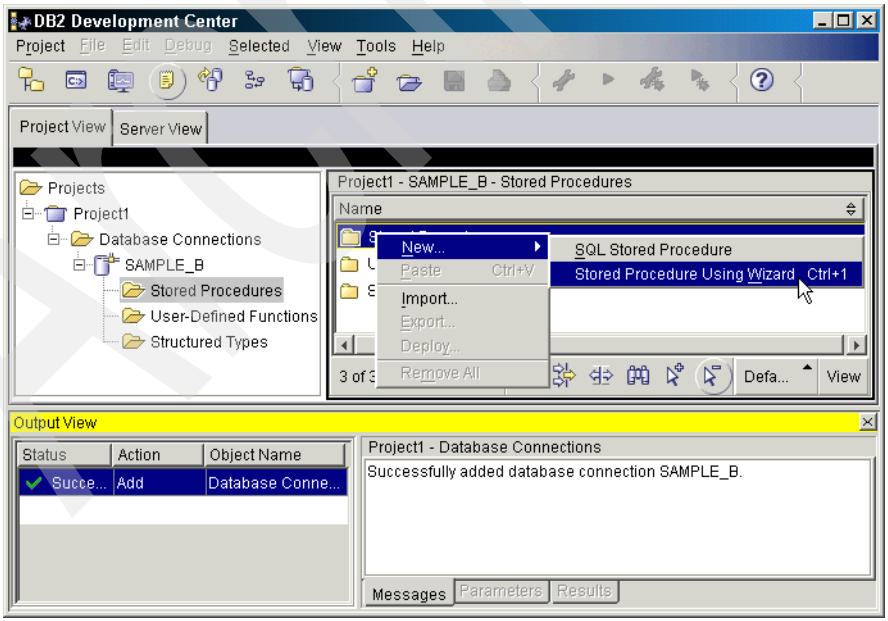

Figure 8-5 Adding a stored procedure

8. On the next dialog, select **SQL** to create a SQL stored procedure and then click **OK**. Now fill in the **Name** text box with the stored procedure name and then click **Next** (Figure 8-6).

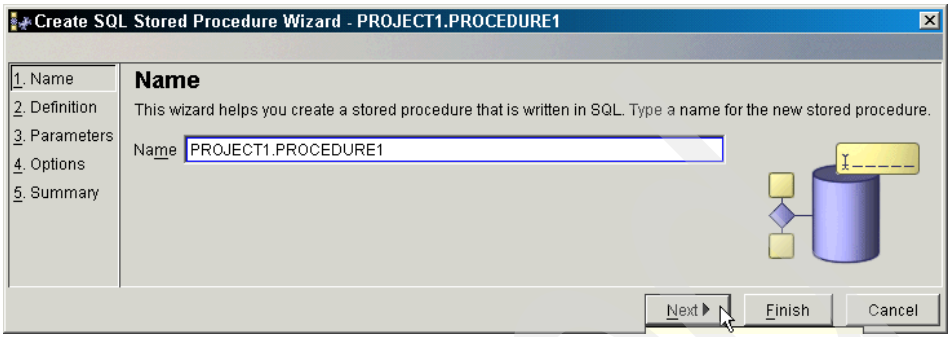

Figure 8-6 Filling in the name text box

9. On the next dialog you define the stored procedure structure setting. Change the information according to your needs and then click **Next** (Figure 8-7).

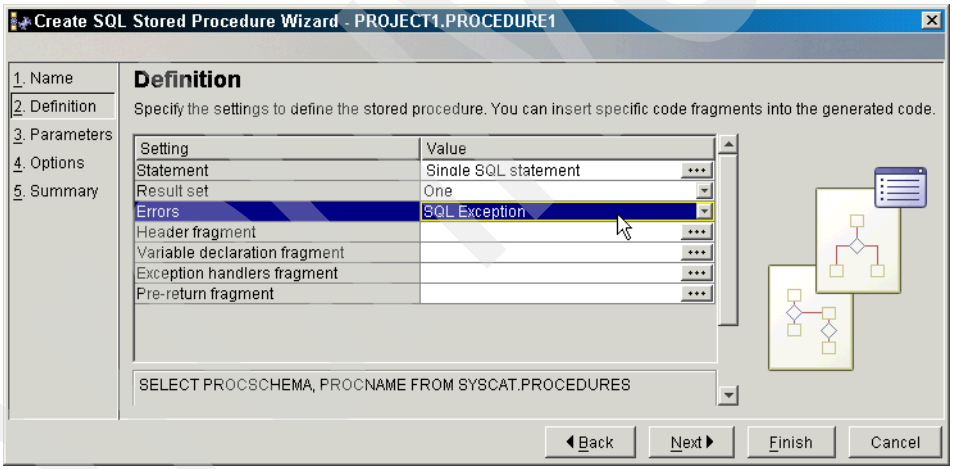

Figure 8-7 Changing the procedure definition

10. On the following dialog, use the buttons on right side to add or remove parameters. you can also change the parameters sequence by clicking the **Move Up** or **Move Down** buttons. Click **Next** to continue (Figure 8-8).

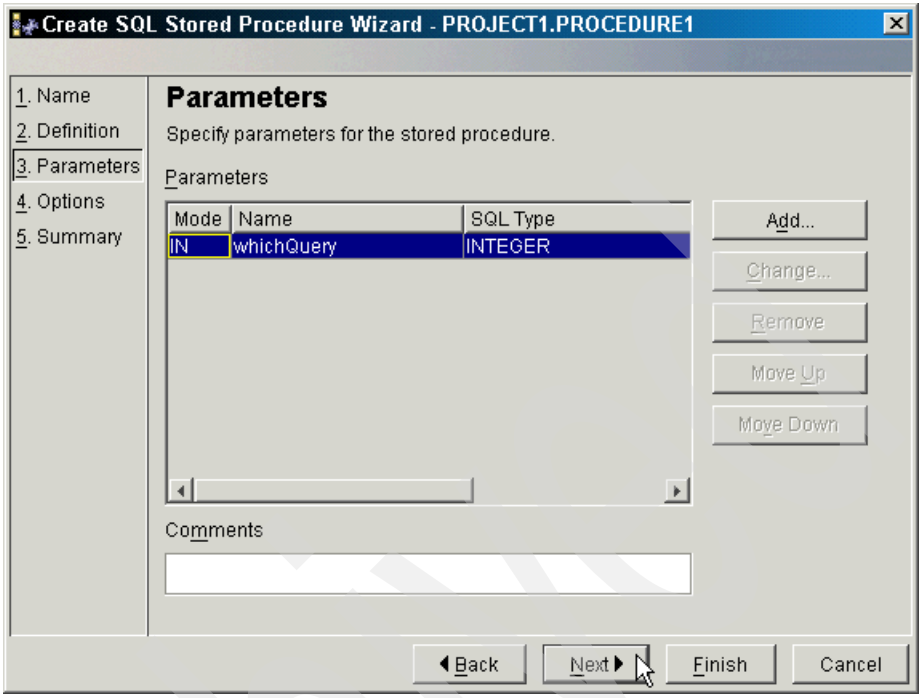

Figure 8-8 Adding or removing parameters

11. To add a comment on the stored procedure, fill in the **Specific Name** text box. We let the **Build** and the **Enable Debugging** check-box activated for tests purpose. If you want to review the choices made until now, proceed by clicking **Next**. Click **Finish** to create the stored procedure and build it (Figure 8-9).

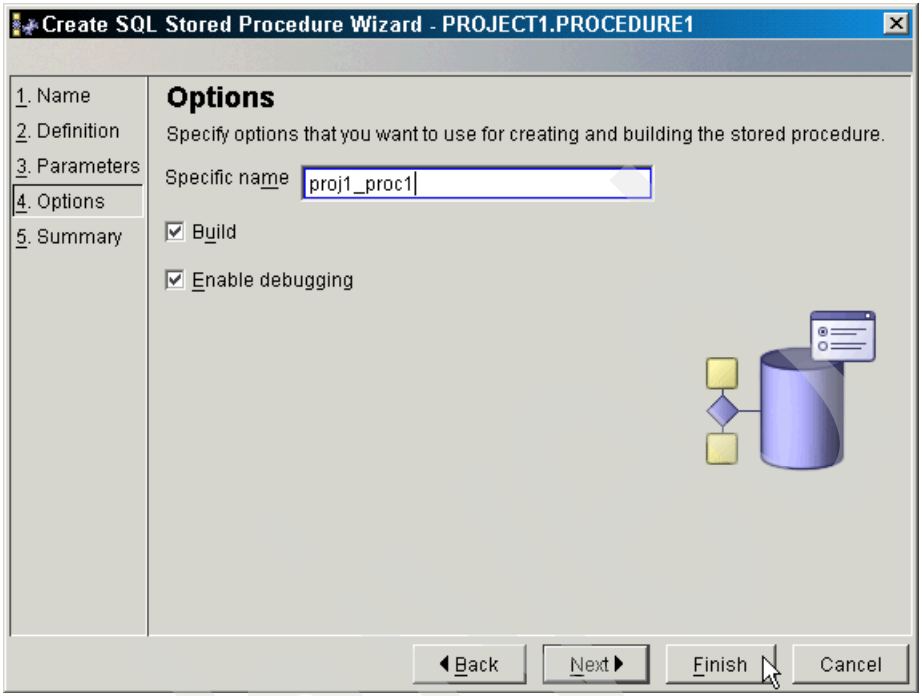

Figure 8-9 Stored procedure options and build

12. All the compile and build messages are shown on the bottom right hand corner under **Messages** tab**.** You can check if the stored procedure is built successfully.To run the stored procedure, right click the stored procedure entry on the stored procedures list, and then click the **Run** menu option (Figure 8-10).

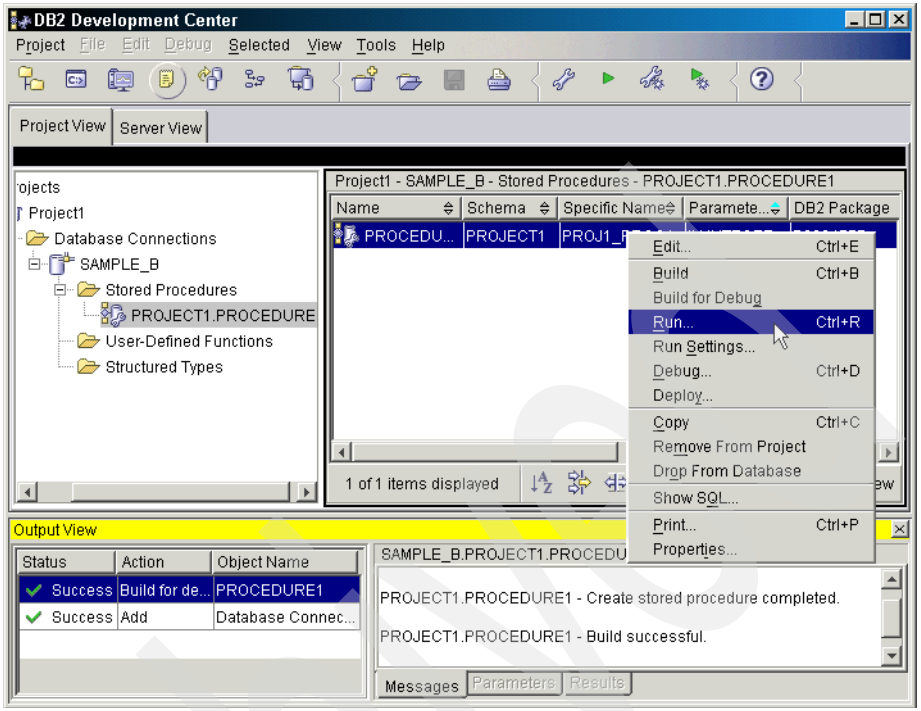

Figure 8-10 Run procedure after checking compiling and building messages

13. Since we have one input parameter defined, the stored procedure builder will open a dialog to enter the parameters. In our case, we enter the number 0. To proceed, click **OK** (Figure 8-11).

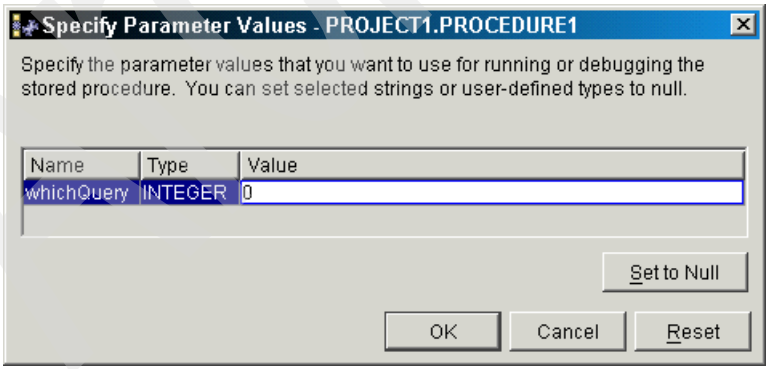

Figure 8-11 Specifying parameter values

14. Now on the list box at the bottom right corner of DB2 Development Center, you can see the result obtained from the stored procedure (Figure 8-12)

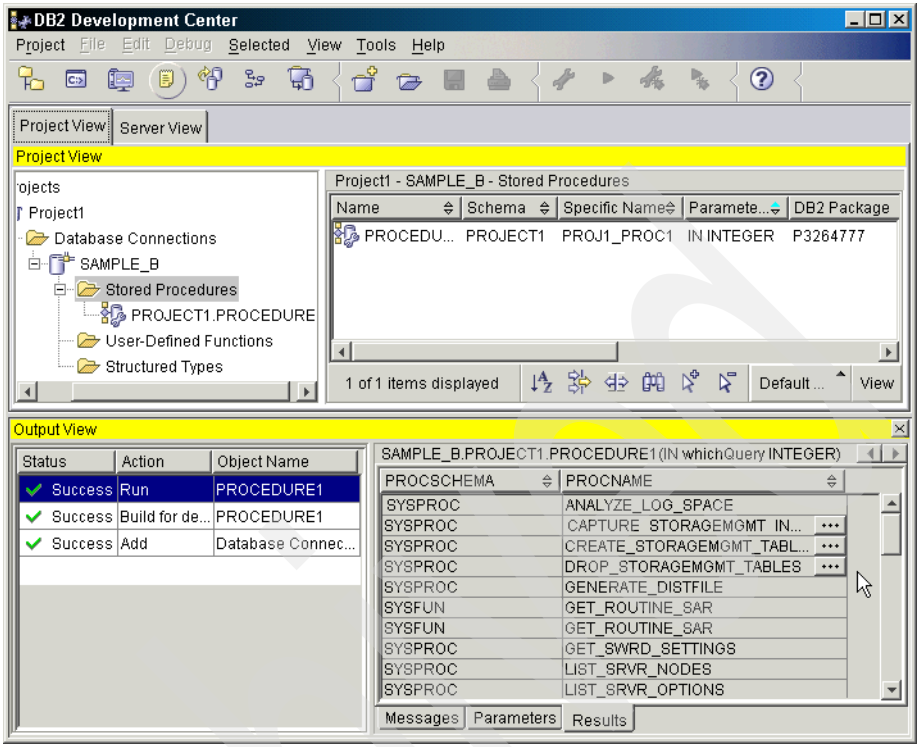

Figure 8-12 Displaying the result from the stored procedure.

15. Since the stored procedure is working, we now make some changes in the stored procedure code. To open the stored procedure code, right-click its name on the list at the right side of Development Center and then click **Edit** (Figure 8-13).

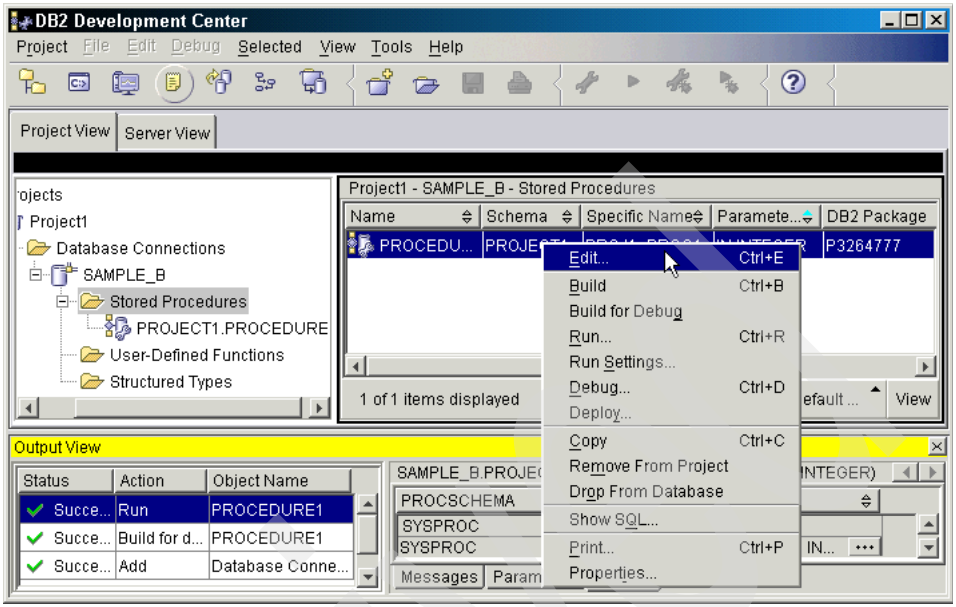

Figure 8-13 Opening the stored procedure code

16. On the next dialog (Figure 8-14) you can see the stored procedure code. Let's change the stored procedure to fit our needs. Here is the code that should be typed on the stored procedure body. The code has one error which we will fix at debug time (Example 8-2):

Example 8-2 Stored procedure body

```
P1: BEGIN
   -- Declare cursor
      DECLARE cursor1 CURSOR WITH RETURN FOR
         SELECT PROCSCHEMA, PROCNAME FROM SYSCAT.PROCEDURES;
      DECLARE cursor2 CURSOR WITH RETURN FOR
         SELECT PROCNAME;PROCSCHEMA FROM SYSCAT.PROCEDURES;

   -- Cursor left open for client application
   IF whichQuery = 1 THEN
      OPEN cursor1;
   ELSE
      OPEN cursor2;
   END IF;

END P1
```

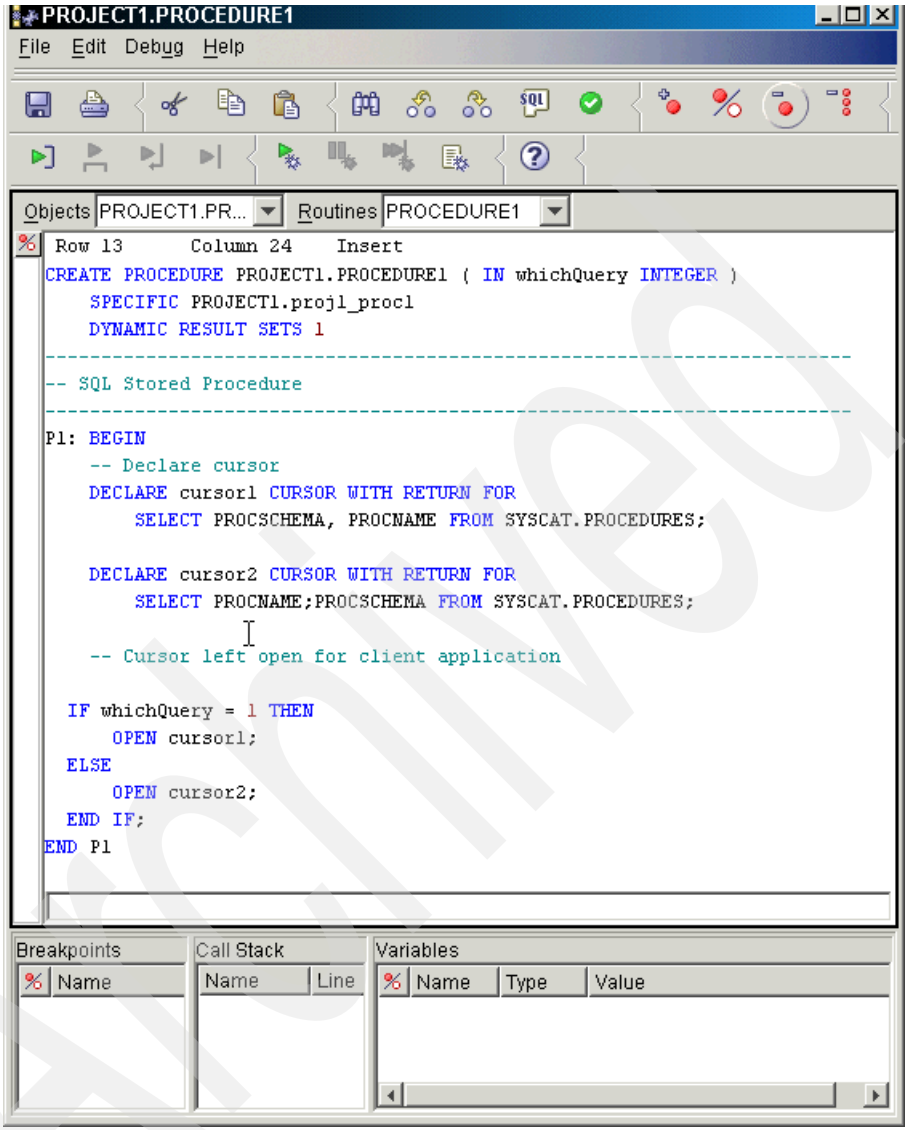

Figure 8-14 The stored procedure builder with the new code

Tip: You can add some basic code structures by clicking the **Edit** menu at the top of the dialog.

17. Now save your work on the stored procedure by clicking the menu **File—>Save Object** on the top left most area of the dialog. Close the stored procedure editor by clicking the button **X** at top right most area of the dialog. To build your procedure, right click again on the stored procedure name and then click **Build For Debug** menu option.

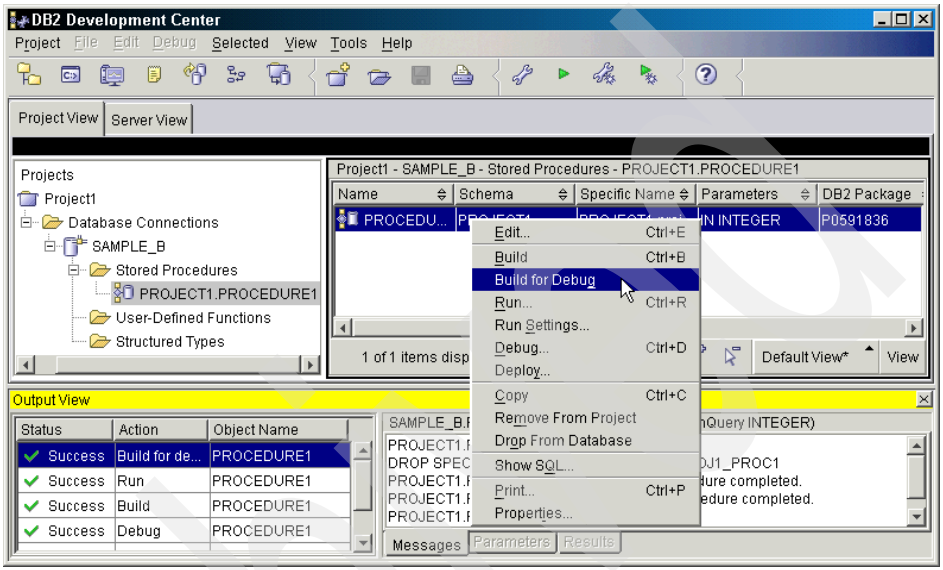

Figure 8-15 Build for debug

18. You can see the result of compilation and building on the list at right side in the **Output View**. There is a list of errors since our code failed on compilation. Reading the messages you will see that DB2 find a syntax error on the code at line 13 (Example 8-16).

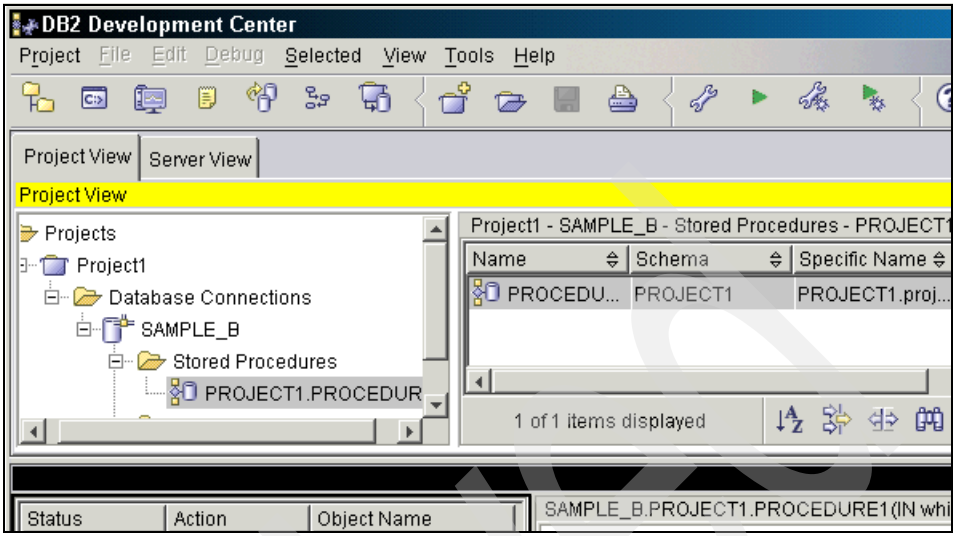

Figure 8-16 Build failed on syntax error detection

19. Now go up to the code, right click again on the stored procedure name and then click the **Edit** option. Correct the code as shown in the following (Example 8-3):

Example 8-3 Correcting the code on line 13

```
P1: BEGIN
   -- Declare cursor
      DECLARE cursor1 CURSOR WITH RETURN FOR
         SELECT PROCSCHEMA, PROCNAME FROM SYSCAT.PROCEDURES;
      DECLARE cursor2 CURSOR WITH RETURN FOR
         SELECT PROCNAME, PROCSCHEMA FROM SYSCAT.PROCEDURES;

   -- Cursor left open for client application
   IF whichQuery = 1 THEN
      OPEN cursor1;
   ELSE
      OPEN cursor2;
   END IF;

END P1
```

20. Save your work again by clicking the menu **File—>Save Object** on the top left most area of the dialog. Close the stored procedure editor by clicking the **X** button at top right most area of the dialog. To build your procedure again, right click the stored procedure name and then click **Build for Debug** menu option.

21. Now look at the **Output View** again and you should see a successful compilation. See Figure 8-17.

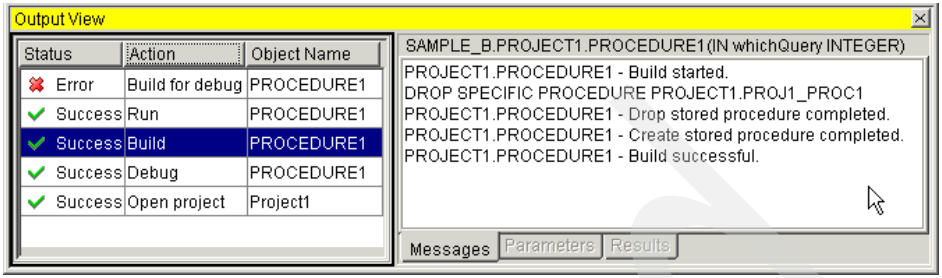

Figure 8-17 Successful compilation

22. Now we would like to demonstrate some debug functionality. Open the stored procedure code again and add a breakpoint on the code by clicking the left bar of the stored procedure editor (Figure 8-18)

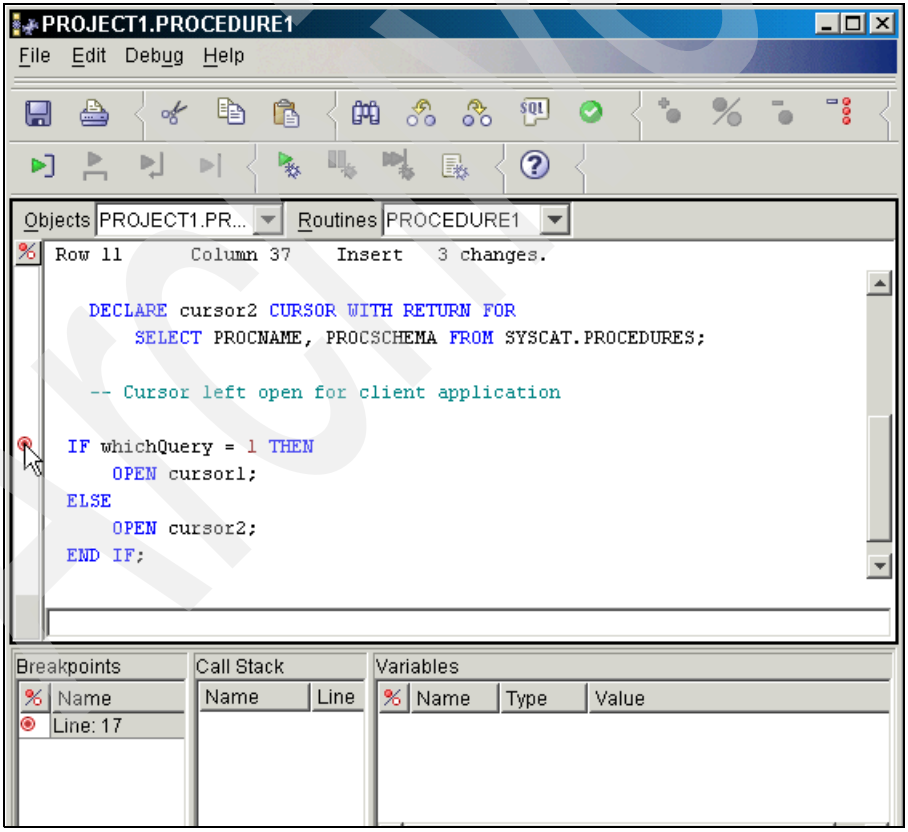

Figure 8-18 Adding a breakpoint

23. Now click the debug button at the top bar on the dialog. Enter the parameter value as 1 and then click **OK**. DB2 Stored Procedure builder now will start to run step-by-step. Click again on the button and you should have the following screen (Figure 8-19):

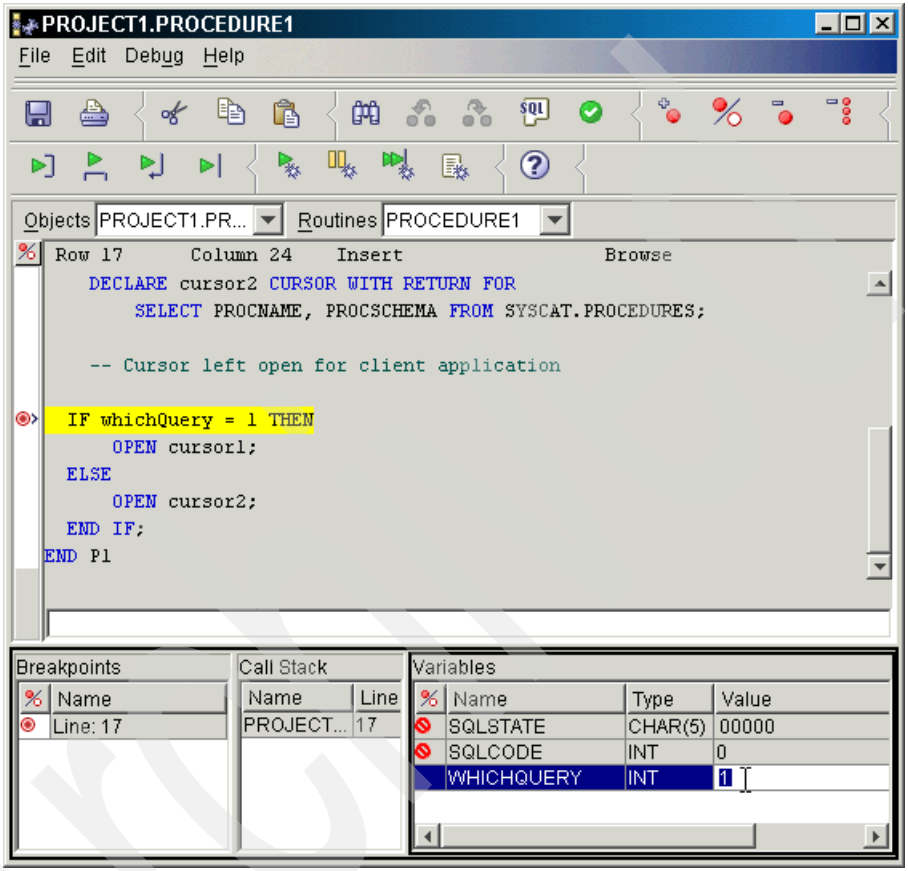

Figure 8-19 Stopping at the breakpoint

24. Now on the bottom part of Stored Procedure Builder, you can see a group of lists. The first list contains the breakpoints created. On the second list, you can see the sequence of stored procedure called since you can call a stored procedure inside a stored procedure and debug them together. On the third dialog, you can see the a list of all variables available on the stored procedure.

You can change the variable values before proceeding. In our case, place the cursor over the WHICHQUERY variable and then click. Now change the value to 0. Press the button  again to continue. Now back on the main dialog, you will see on the result list box the second query open instead the first one (Figure 8-20). We finish our stored procedure debugging. To complete the stored procedure building, right-click the project icon under the Projects folder and then click **Save Project** (Figure 8-21).

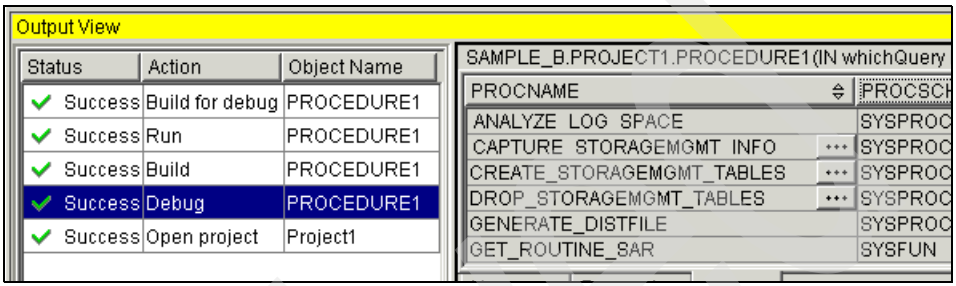

Figure 8-20 Displaying the second stored procedure instead of the first one

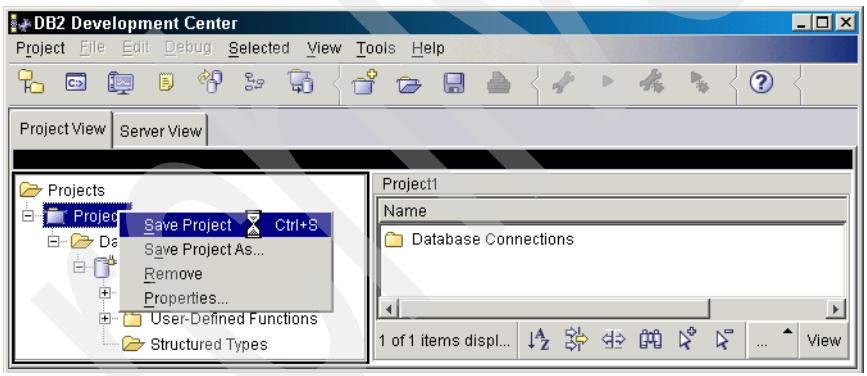

Figure 8-21 Saving your project

25. To deploy your stored procedure to another server, go to the main dialog on DB2 Development Center and right click the stored procedure icon and then click **Deploy.** A dialog will be opened (Figure 8-22) and you should choose the database alias where you plan to deploy your procedure.If you want to use another userid than the current one to connect to the database, uncheck **Use your current userid and password** check box and fill in the user and password desired. Proceed by clicking **Next**.

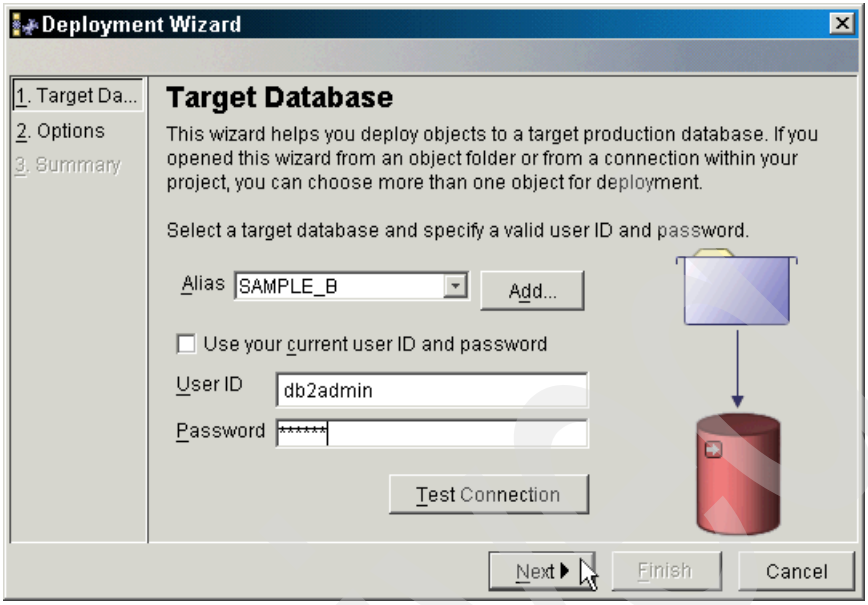

Figure 8-22 Target database login

26. On the next dialog (Figure 8-23), you can set deployment options. If your procedure is written in Java, you can deploy the sources along with the binary file by marking the **Deploy source to database** check box. In our case, we check **Deploy using files if available** check box to deploy only the binary file generated from SQL stored procedure. Click **Advanced** if you want to change compilation options or change the type of the stored procedure to be run as fenced or not fenced. If you want to review any step on the deployment, click **Next**. Otherwise click **Finish** to execute the deployment.

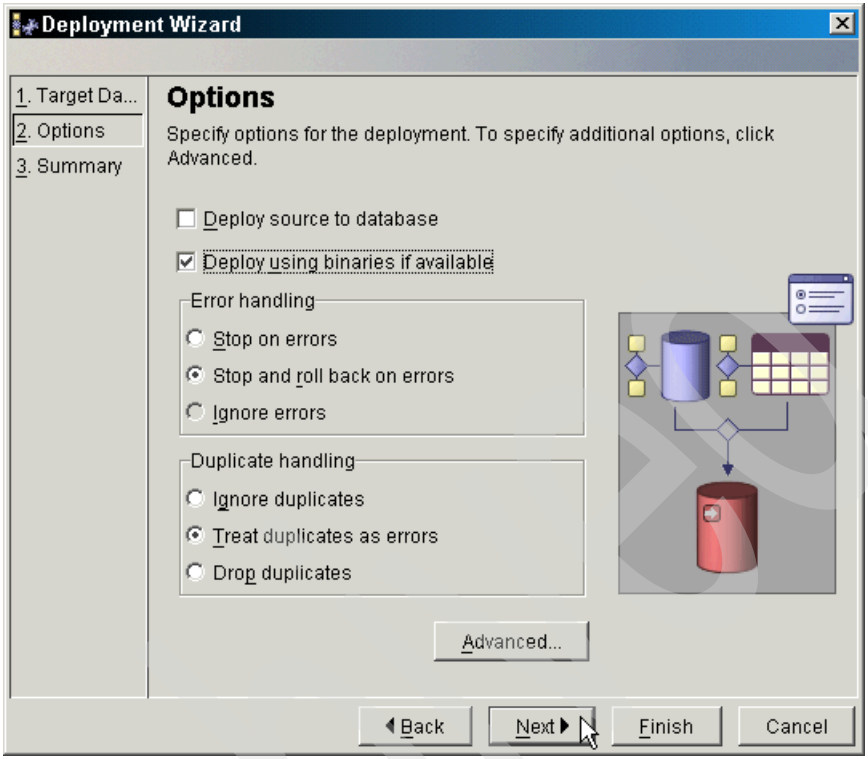

Figure 8-23 Set deployment options and finish the deployment

Creating and using a UDF

Another Development Center function is building a User Defined Function (UDF). You can find a good example of creating a UDF in Chapter 3, “Post-installation tasks” on page 95.

8.1.2 Project Deployment Tool

DB2 Development Center includes a GUI tool to deploy application objects from developing stage to production. Since this tool works with package distribution, the developer need send only one zipped file that contains all the object to be deployed to the DBA.

1. To deploy your stored procedure to another server, go to the main dialog on DB2 Development Center and right click the database icon on the project that you want to export. Now click the **Export** option (Figure 8-24).

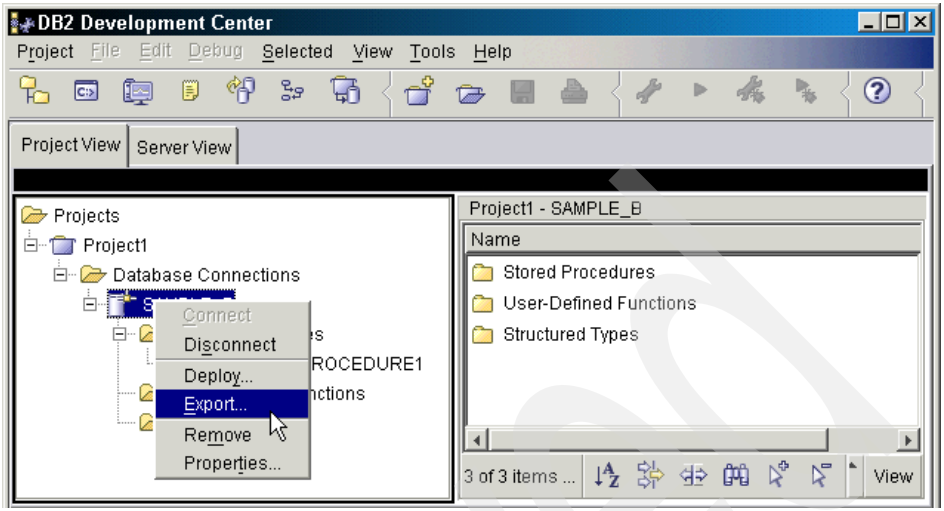

Figure 8-24 Export stored procedures from a database

2. On the next dialog, choose the stored procedures and user-defined functions that you want to export by drilling down on the database objects tree. To add a desired object or the entire database contents, select the object and then click the > button to add to the list of selected objects. Click **Next** to proceed.

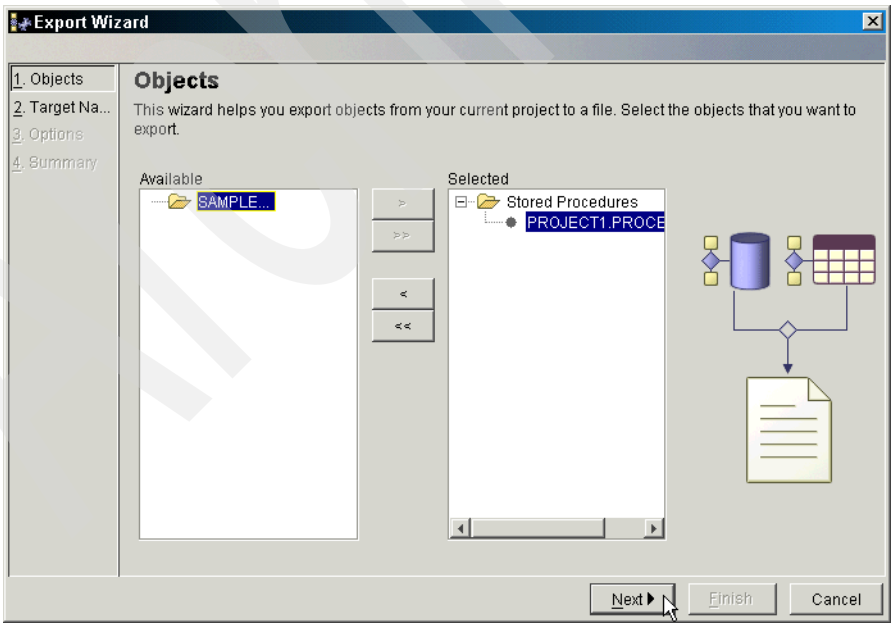

Figure 8-25 Adding the objects to be exported

3. On the next dialog (Figure 8-26), enter the file name and the path where the zipped file will be placed. Click **Next** to proceed.

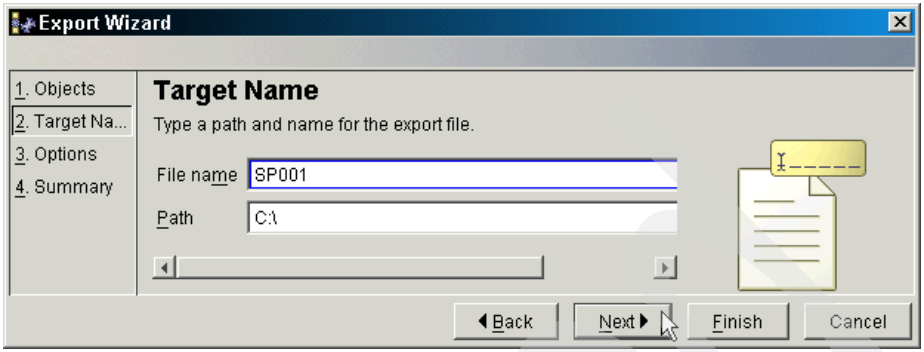

Figure 8-26 File name and path

4. On the next dialog, you can choose to deploy the final package using Deployment wizard or command line tool. We proceed by leaving **Export as a project** selected and the option to include source files checked. If you want to review your options, click **Next.** Otherwise, click **Finish** to generate the package**.**

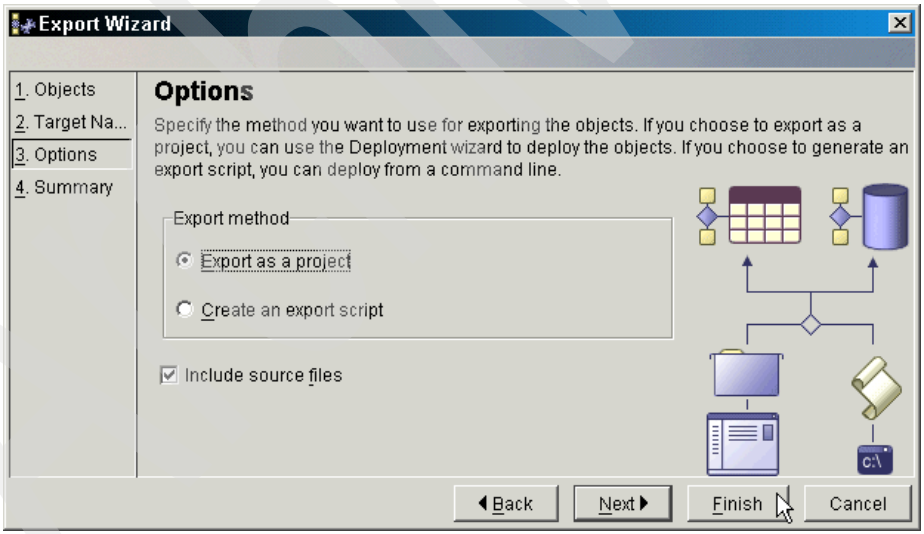

Figure 8-27 Selecting export options

5. Now, open the **Start Menu—>Programs—>IBM DB2—>Development Tools—>Project Deployment Tool**. On the initial dialog screen (Figure 8-28) enter the package directory and name. To search the path and file, click the button at the right side of the text box. Click **Next** to proceed.

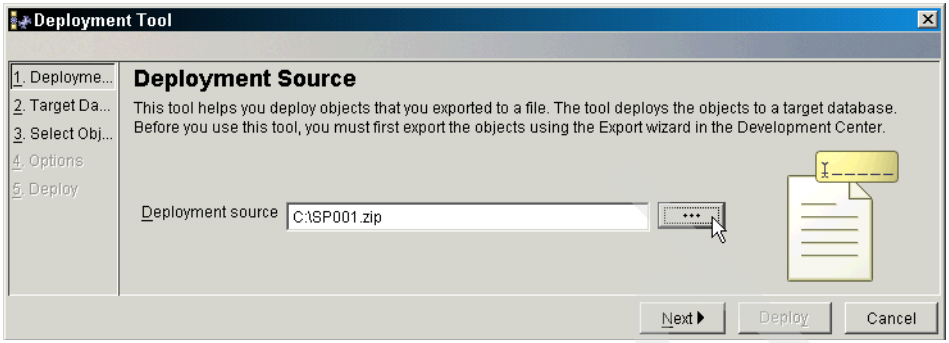

Figure 8-28 Select the deployment source file

6. On the next dialog (Figure 8-29), choose the stored procedures and user-defined functions that you want to export by drilling down on the database objects tree. To add a desired object or the entire database contents, select the object and then click the > button to add to the list of selected objects. Click **Next** to proceed.

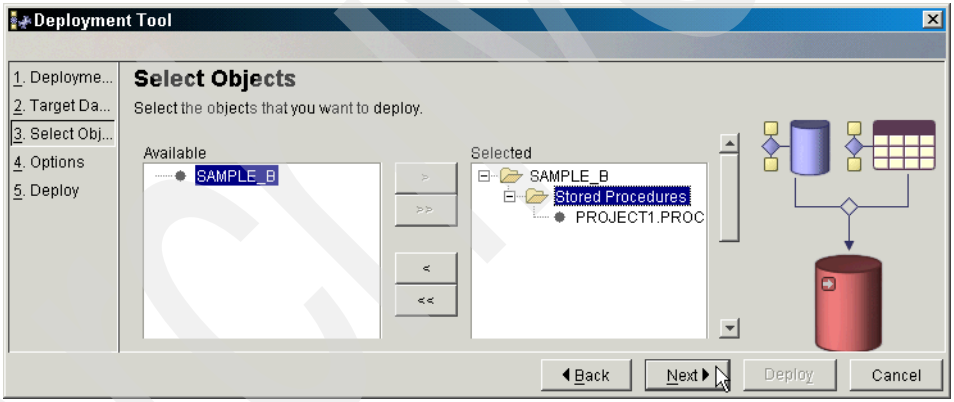

Figure 8-29 Selecting the objects to be deployed

7. Choose the database alias to deploy your procedure (Figure 8-30).If you want to use another user than the current one to connect to the database, uncheck **Use your current userid and password** check box and fill in the user and password desired. Proceed by clicking **Next**.

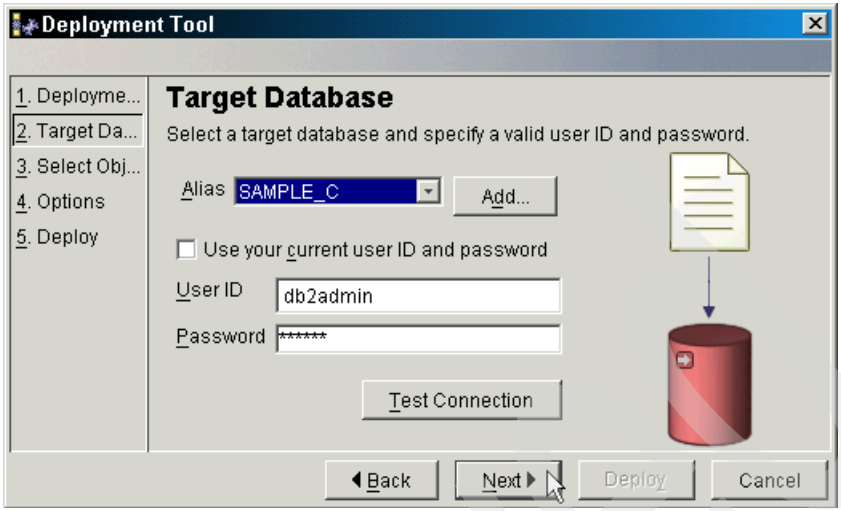

Figure 8-30 Target database login

8. On the next dialog (Figure 8-31), you can set deployment options. If your procedure is written in Java, you can deploy the sources along with the binary file by marking the **Deploy source to database** check box. In our case, we check **Deploy using files if available** check box to deploy only the binary file generated from SQL stored procedure. Click **Advanced** if you want to change compilation options or change the type of the stored procedure to be run as fenced or not fenced. Click **Next**.

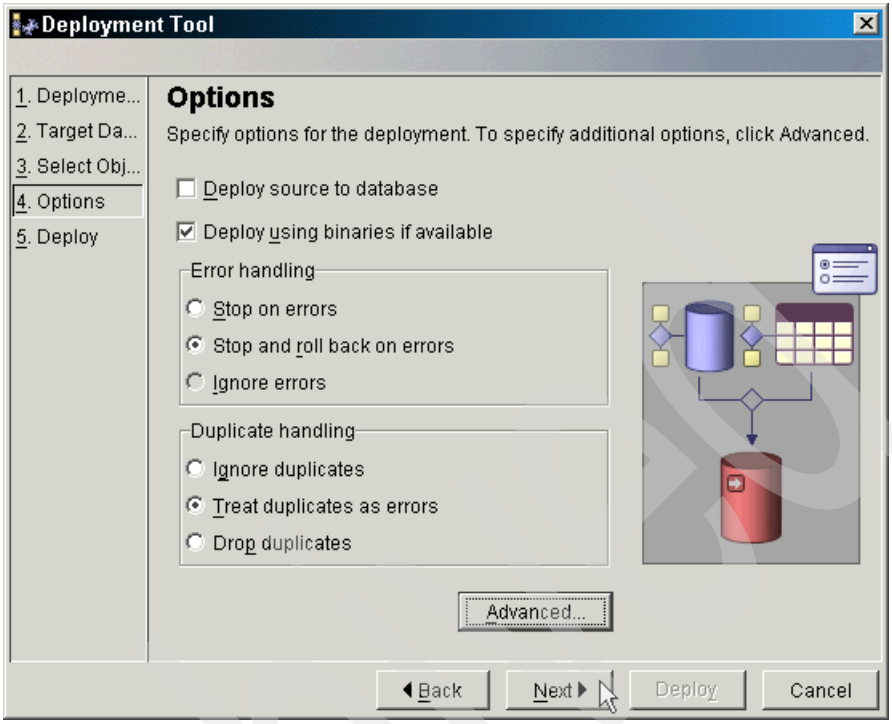

Figure 8-31 Setting deployment options

9. To finish, click **Deploy** and then verify the output of deployment (Figure 8-32).

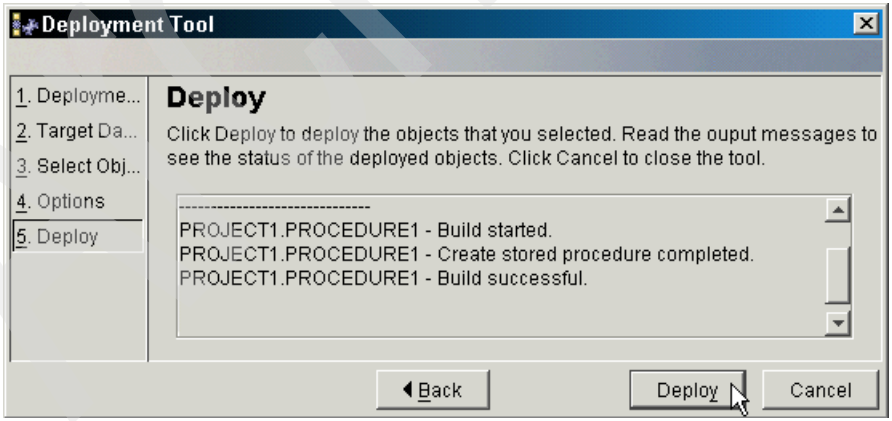

Figure 8-32 Procedure deployed

8.1.3 Command Center

The DB2 Command Center provides you with a powerful interactive tool for executing DB2 SQL statements and commands as well as the ability to build and test scripts containing those statements and commands.

You can start the Command Center in the following ways:

- Select Command Center from the Tools menu of another tool.
- Click the Command Center icon from the toolbar of another tool.
- From the tools menu of Microsoft Visual Studio, Microsoft Visual C++ or Microsoft Visual Basic if you have enabled the DB2 Add-ins.
- On Windows systems, click the **Start** button and select **Programs** —> **IBM DB2** —> **Command Line Tools** —> **Command Center**.

Use the Command Center to execute DB2 commands and SQL statements; to work with command scripts; and to view a graphical representation of the access plan for explained SQL statements.

On the Interactive page as shown in Figure 8-33, you can perform the following actions:

- Execute an SQL statement or DB2 Command Line Processor command.
- Run a command or statement, by clicking the gears icon (at the left of the toolbar).

The Database connection area shows the current database that you are connected to. You can bring up the drop-down list from this area and select the database and a connection will be made for you.

The Command history area tracks the most recent commands and statements issued interactively. The last command or statement issued is shown. If you bring up the drop-down list, it will show the most recent commands, you can select one of the commands to recall it to be executed again.

The Command area is a work area for entering a statement or command to be executed. For SQL statements you can enter them directly into the command area or invoke the SQL Assist to build the SQL for you. The SQL Assist will be discussed later in the section 8.1.4, “SQL Assist” on page 420.

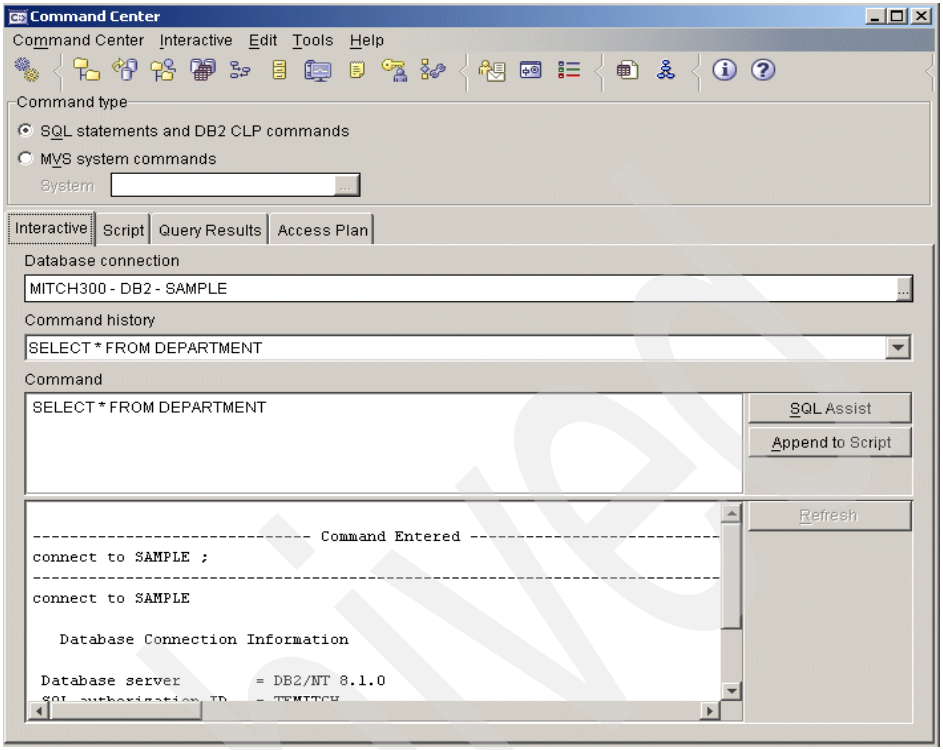

Figure 8-33 DB2 Command Center — Interactive window

On the Script page as shown in Figure 8-34, you can perform the following actions:

- Execute commands in sequence.
- Create and save a script. You can optionally store a saved script in the Task Center, where you can schedule the script to run at a specific time.
- Run an existing script.
- Schedule a task.

On the Query Results page as shown in Figure 8-35 on page 419, you can see the results of the queries. You can also save the results or edit the contents of the table.

On the Access Plan page as shown in Figure 8-36 on page 420, you can see the access plan for any explainable statement that you specified on the Interactive page or the Script page. DB2 generates the access plan when it compiles the SQL statement if you enable the option to do so automatically or if you select the

create access plan option from the interactive page menu bar. You can use this information to tune your queries for better performance.

Note: If you specify more than one statement on the Script page, an access plan is created only for the first statement.

We step through an example of using the Command Center below. We execute a simple SELECT statement against the sample DEPARTMENT table in the SAMPLE database.

In the interactive window as shown in Figure 8-33 on page 416 we have selected the SAMPLE database in the Database connection area, entered the SELECT statement in the Command area and executed the query with the results shown in the bottom portion of the screen. The query results window as shown in Figure 8-35 on page 419 will be presented after the query completes.

If you want to view the access plan for the query select **Interactive->Create Access Plan** from the menu bar. The Access plan window as shown in Figure 8-36 on page 420 will be presented if you select this option.

The script window in shows where we have built a script with the SQL to Connect to the SAMPLE database, query the DEPARTMENT table and disconnect from the database. The results of executing the script is shown in the bottom portion of the window.

Notice that each statement is terminated with a semicolon (;). This is required to delimit each statement. The delimiter character can be chosen by selecting **Tools->Tools Settings** from the menu bar. The default delimiter is the semicolon (;) that we are using. If you will be using SQL statements that might contain a semicolon within the statement like a CREATE PROCEDURE or CREATE TRIGGER you will need to use another delimiter to avoid errors. A good alternative would be to use the at sign (@) that is used by some of the DB2 provided sample Stored Procedures and Triggers.

You also have the option to save the script for execution at a later time by selecting **Script->Save Script** from the menu bar. After saving the script you can also schedule the script for execution with the DB2 Task Center by selecting **Script->Schedule** from the menu bar.

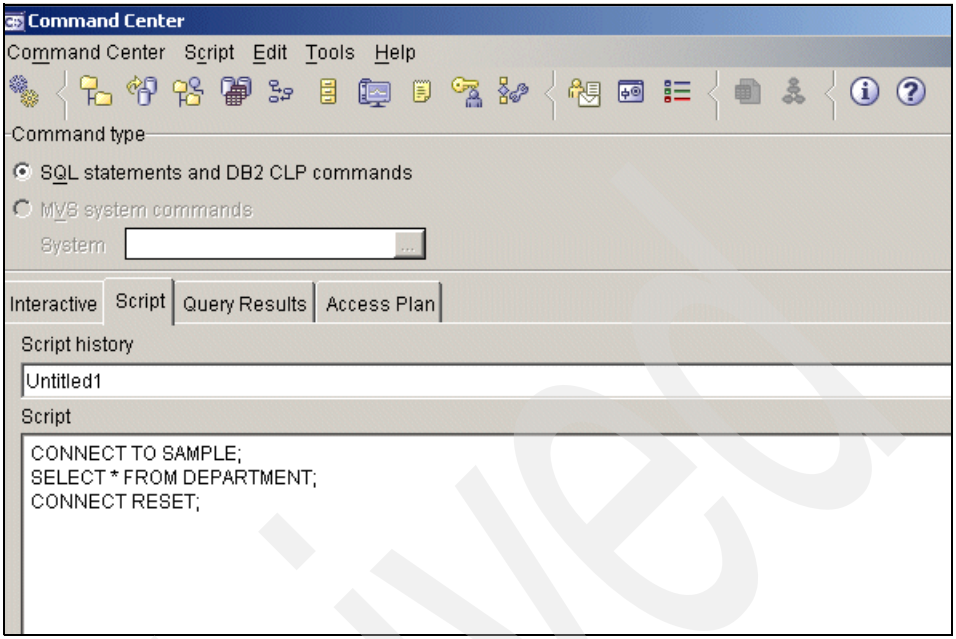

Figure 8-34 DB2 Command Center — Script window

The query results window is shown in Figure 8-35. The results of the query are presented in a tabular form. You can scroll forward and backward through the query results.

You can also modify the contents of the table by moving the cursor to row/column you want to change and making the change. To make your changes permanent select the **Commit Update** button.

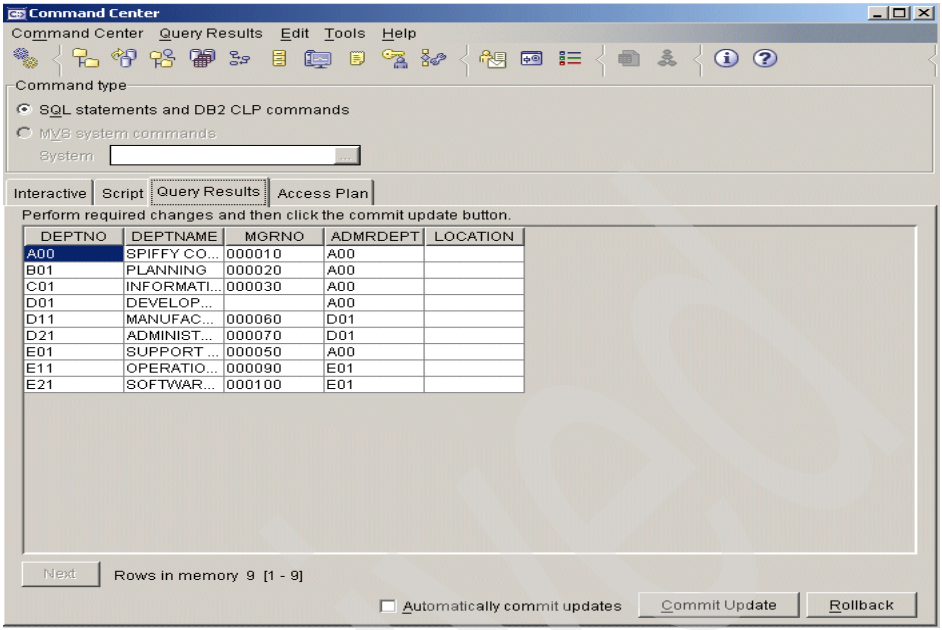

Figure 8-35 DB2 Command Center — Query results page

The Access Plan window shown in Figure 8-36 provides a graphical view of the access plan selected by the optimizer for our query. For our simple query the access plan is also very simple. For more information on access plans, see Section 8.1.5, “Visual Explain” on page 425

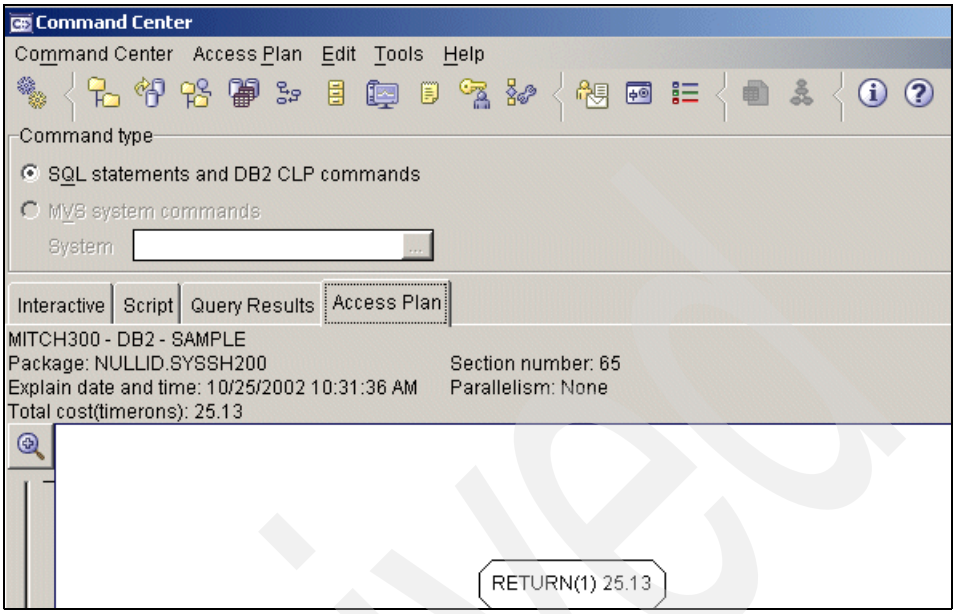

Figure 8-36 DB2 Command Center — Access Plan window

As mention above the Command Center permits you to invoke the SQL Assist tool by clicking the **SQL Assist** button on the Interactive page. To invoke the Visual Explain tool, execute an explainable statement on the Interactive page or the Script page. SQL Assist is discussed in the Section 8.1.4, “SQL Assist” on page 420. Visual Explain is discussed in the Section 8.1.5, “Visual Explain” on page 425 later in this chapter.

As you have seen, the Command Center is a very useful tool to improve productivity for application development and database administration tasks.

8.1.4 SQL Assist

SQL Assist is an easy-to-use GUI tool that will build SQL statements for you in step wise fashion. It supports the building of SQL SELECT, INSERT, UPDATE and DELETE statements.

To start the SQL Assist tool, click the **SQL Assist** button in another tool. The **SQL Assist** button exists at various locations in the Control Center, Command Center, Replication Center, Development Center, and Data Warehouse Center. If you have enabled the DB2 Add-ins for Microsoft Visual Studio, Microsoft Visual C++ or Microsoft Visual Basic, the SQL Assist can also be invoked from the tools menu bar in those development environments.

Each tool allows you to create SQL statements that are appropriate in the context of that tool. In some tools, you can create only SELECT statements. In other tools, you can also create INSERT, UPDATE, and DELETE statements.

The SQL Assist window as shown in Figure 8-37 contains the following components:

- The Outline area contains a high-level representation of the current SQL statement type. Select an element of the SQL statement.

  For SELECT statements, you can work with the following elements:

  - FROM: To select one or more tables to query. If more than one table is selected and there is a PRIMARY KEY/FOREIGN KEY relationship between the tables, a suggestion for joining the tables is presented.
  - SELECT: To select the columns for the query to return. All the columns from the tables selected above will be presented for you to choose from.
  - WHERE: To filter the rows to be returned by the query.
  - GROUP BY: To group rows by values in columns.
  - HAVING: To filter group by rows.
  - ORDER BY: To define sort criteria

  For INSERT statements the following elements can be worked with:

  - INSERT INTO: To specify the table to insert rows into.
  - VALUES: To specify the data to be inserted into the table.

  For UPDATE statements the following elements can be worked with:

  - UPDATE: To specify the target table of the update.
  - SET: To specify the values to update.
  - WHERE: To specify the row filtering for the update.

  For DELETE statements the following elements can be worked with:

  - DELETE FROM: To specify the table to delete rows from.
  - WHERE: To specify the row filtering for the delete.

- The Details area displays the appropriate panel for the element that you selected. You build your SQL statement by working with this panel. The changes that you make will be reflected in the SQL code area.
- The SQL code area displays the current SQL statement. Colors are used to highlight the syntax. You can edit the SQL statement or paste an existing SQL statement into this area. If you make any changes in this area, you will not be able to take any actions in the Details area until you validate the syntax of the modified SQL.

To validate the syntax, click the **Check** button. If the syntax is correct, the Details area will be enabled again, and its content will be updated to reflect your changes. To run the SQL statement, click the **Run** button.

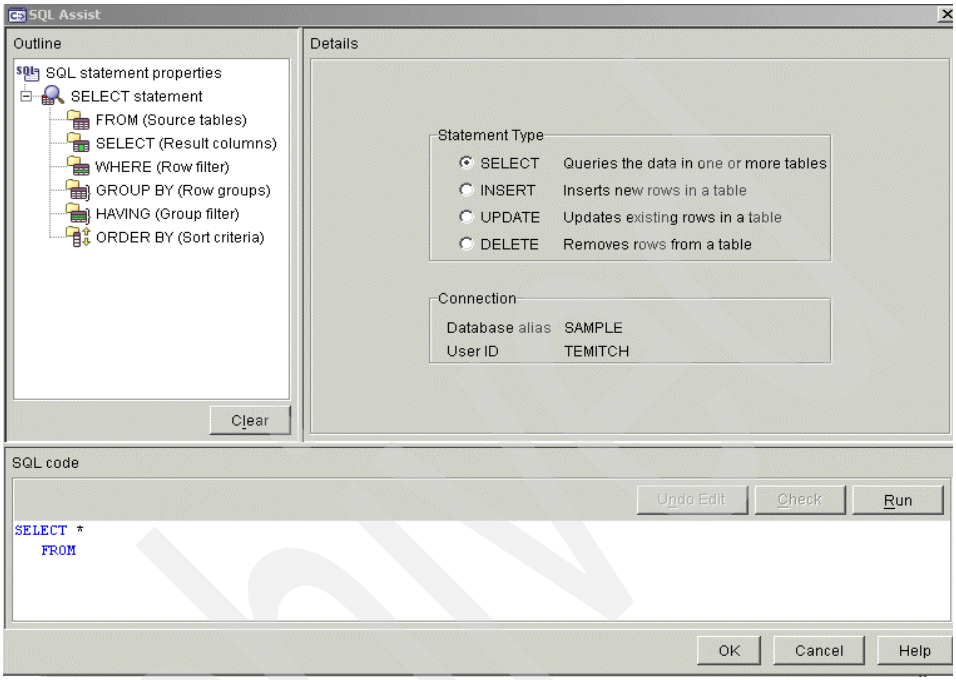

Figure 8-37 SQL Assist — Initial window

We now step you through a very simple example of building a SELECT statement on the sample DEPARTMENT table.

Figure 8-37 shows us the initial window that is presented when you start the SQL Assist. By default it will show the options for building a SELECT statement. If we highlight FROM in the outline area, the schemas available in the database will be presented, expanding a schema will show the tables available within that schema. Highlighting the table you want to query will place that table name in the query to be constructed in the SQL code area as you can see in Figure 8-38.

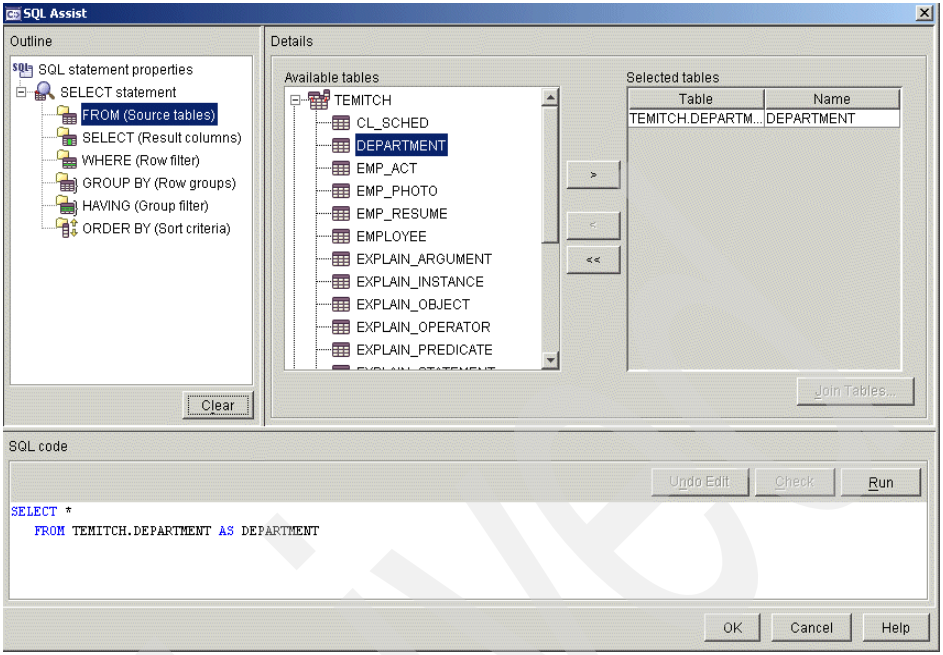

Figure 8-38 SQL Assist — Specify tables

To specify the columns you would like to use in our query, the SELECT in the outline area needs to be highlighted. All the columns available in the table or tables will be displayed in the Details area for you to select from. Highlight the columns you would like included in your query and select the > button to have them added to the query being built in the SQL code area. In our example, we select all columns in the table as shown in Figure 8-39.

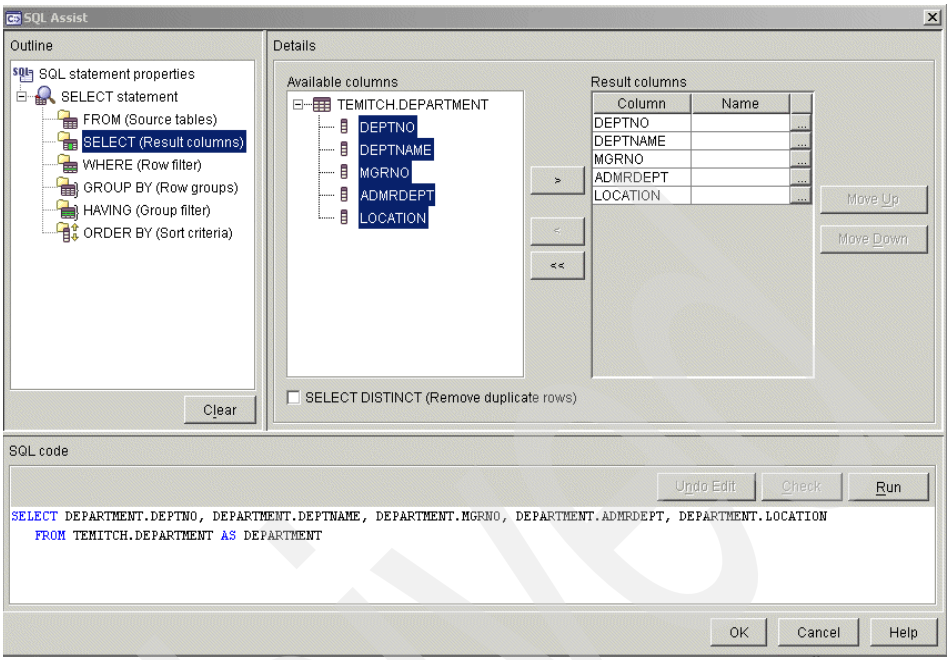

Figure 8-39 SQL Assist — Specify table columns

After selecting the columns, you have the options of selecting GROUP BY columns and a HAVING clause which we don't need for our example.

By highlighting the ORDER BY in the outline area, you can select the columns that you want the query result set to be sorted by. Highlight the columns you want the result set sorted by, then select the **>** button to have them added to the query being built in the SQL code area. In our example we selected the column DEPTNO as shown in Figure 8-40.

We can now test the constructed query by selecting the **Run** button or return to the tool we invoked SQL Assist from with the query text being passed back to the tool by selecting **OK**.

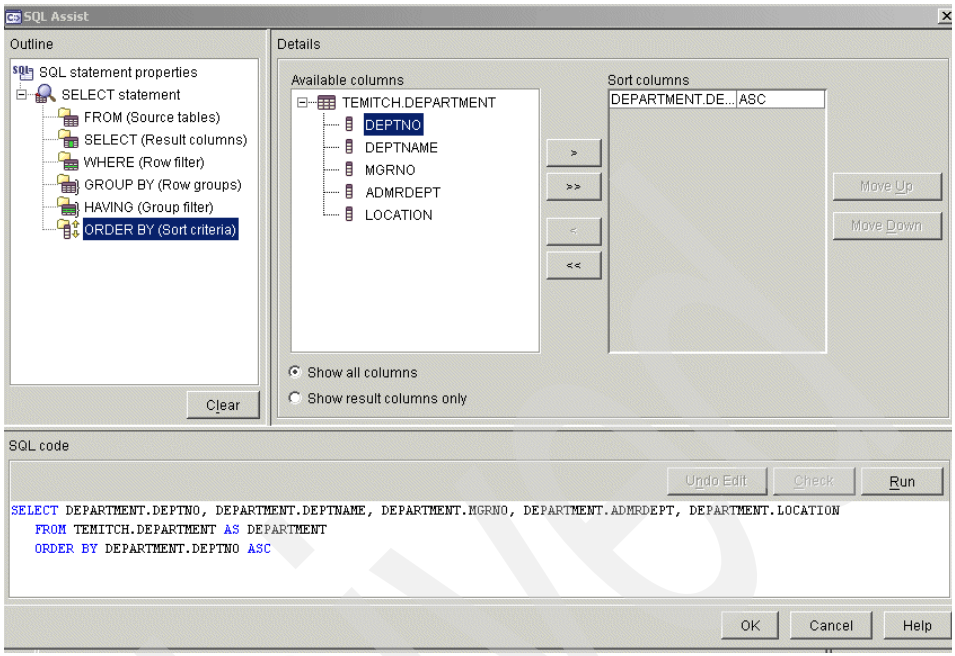

Figure 8-40 SQL Assist — Specify Order By Columns

As you have seen, the SQL Assist tool provides a very easy-to-use tool that can improve your productivity in the building and testing of SQL statements to include in your applications.

8.1.5 Visual Explain

Visual Explain translates the information generated by db2explain stored on explain tables to a visual graphic. It assists the user to understand the path generated by db2 optimizer to get the data.

Setting up the environment:
We need to create explain tables for DB2 Visual Explain to store data and give table access authority to the users who will generate explain data.

You can create the explain tables by using either command center or command-line tools. The command center can create them automatically by clicking **Command center —> Options** on the top left-most menu on the Command Center, clicking the **Access Plan** tab, and then checking the option **Automatically generate access plan**. But if you work on a secured environment, the common user may not have the authorities to create tables.

In these environments, you should create a common group of tables for the users and grant them access. In this case, you also need to choose a table space to create the tables. We continue with the default table space. After that, define the schema UDBRED for the explain tables:

1. Open command center by clicking **Start Menu—>Programs—>IBM DB2—>Command Line Tools—>Command Center**.
2. Change the current mode by clicking the **Immediate** tab.
3. Connect to the database by issuing **connect to sample_c** where **sample_c** is your database.
4. Create the schema to hold the tables by issuing **create schema UDBRED**.
5. Set the current connection to the schema using **set schema UDBRED**.
6. Import the file explain.ddl from **SQLLIB** path under **MISC** directory to create the explain tables. Example:
   `C:\Program Files\IBM\SQLLIB\MISC\explain.ddl`
   If the ddl file is on the server, DB2 import dialog allows you to browse the server directory to import that script.
7. Be sure that on the command tools the character terminator is active and set to semicolon (;). To verify, go to **Tools** menu at the top of Command Center and then select **Tools settings**. On **General** tab make sure that **Use statement terminator character** check box is checked and the text is set to semi-colon.
8. Run the script by clicking the  button at the top left most area of the dialog and verify the results on result list on the bottom of Command Center.
9. Grant the select, insert, delete, and update authorities to the desired users or groups. In our case we choose the **Public** group. Example 8-4 shows the grant script.

Example 8-4 Grant commands to allow the users work with the tables

```
GRANT SELECT,INSERT,UPDATE,DELETE ON TABLE UDBRED.ADVISE_INDEX TO PUBLIC;
GRANT SELECT,INSERT,UPDATE,DELETE ON TABLE UDBRED.ADVISE_WORKLOAD TO PUBLIC;
GRANT SELECT,INSERT,UPDATE,DELETE ON TABLE UDBRED.EXPLAIN_ARGUMENT TO PUBLIC;
GRANT SELECT,INSERT,UPDATE,DELETE ON TABLE UDBRED.EXPLAIN_INSTANCE TO PUBLIC;
GRANT SELECT,INSERT,UPDATE,DELETE ON TABLE UDBRED.EXPLAIN_OBJECT TO PUBLIC;
GRANT SELECT,INSERT,UPDATE,DELETE ON TABLE UDBRED.EXPLAIN_OPERATOR TO PUBLIC;
GRANT SELECT,INSERT,UPDATE,DELETE ON TABLE UDBRED.EXPLAIN_PREDICATE TO PUBLIC;
GRANT SELECT,INSERT,UPDATE,DELETE ON TABLE UDBRED.EXPLAIN_STATEMENT TO PUBLIC;
GRANT SELECT,INSERT,UPDATE,DELETE ON TABLE UDBRED.EXPLAIN_STREAM TO PUBLIC;
```

10. For each user that will use Visual Explain, create aliases with the user name as a schema for each explain table (EXPLAIN_*). Example 8-5 shows the create alias script.

Example 8-5 Create aliases for each user

```
CREATE ALIAS JONH.EXPLAIN_ARGUMENT FOR UDBRED.EXPLAIN_ARGUMENT;
CREATE ALIAS JONH.EXPLAIN_INSTANCE FOR UDBRED.EXPLAIN_INSTANCE;
CREATE ALIAS JONH.EXPLAIN_OBJECT FOR UDBRED.EXPLAIN_OBJECT;
CREATE ALIAS JONH.EXPLAIN_OPERATOR FOR UDBRED.EXPLAIN_OPERATOR;
CREATE ALIAS JONH.EXPLAIN_PREDICATE FOR UDBRED.EXPLAIN_PREDICATE;
CREATE ALIAS JONH.EXPLAIN_STATEMENT FOR UDBRED.EXPLAIN_STATEMENT;
CREATE ALIAS JONH.EXPLAIN_STREAM FOR UDBRED.EXPLAIN_STREAM;
```

Running the Explain tool

Follow these steps to use the Explain tool:

1. Open command center by clicking **Start Menu—>Programs—>IBM DB2—>Command Line Tools—>Command Center**.
2. Change the current mode by clicking the **Immediate** tab.
3. Connect to the database with your userid and password: **Connect to sample_c** where sample_c is the database where the explain tables resides.
4. Run the command to set to the explain tables schema. Example: **set schema UDBRED**.
5. Type the SQL statement you want to explain on the **Command** text box. Example: **select * from syscat.table**
6. Go to the menu on the top of the dialog and select **Interactive—> Create Access Plan**. The graphical representation of the access path will show up. (Figure 8-41).

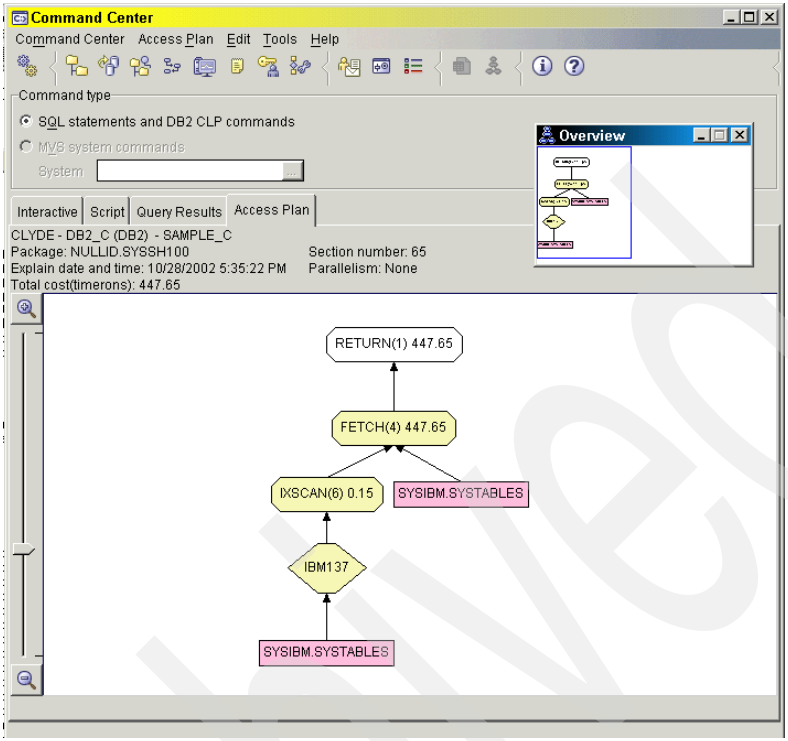

Figure 8-41 Visual Explain showing the access path

The draw showed by Visual Explain represents the access path that DB2 will use when retrieving the data. The access path flows from the bottom to the top. Please note that sometimes when an operation is required to be done more than one time, DB2 will reuse that operation, changing the usual draw from a tree perspective.

You can find an extensive tutorial on *Visual Explain Tutorial Version 8* manual.

You can also find a interesting article on DB2 Explain tools and Visual Explain on `http://gethelp.devx.com/techtips/db2_pro/10Min1101/gn1101-1.asp`

8.1.6 Command-line Explain tools

If you don't have Visual Explain installed on your machine, or you need to Explain static SQL, you can use these command line tools:

- db2expln
- dynexpln
- db2exfmt

db2expln
The db2expln tool describes the access plan selected for SQL statements. It can be used to obtain a quick explanation of the chosen access plan when explain data was not captured. For static SQL, db2expln examines the packages stored in the system catalog tables. For dynamic SQL, db2expln examines the sections in the SQL cache.

dynexpln
The dynexpln tool can also be used to describe the access plan selected for dynamic statements. It creates a static package for the statements and then uses the db2expln tool to describe them. However, because the dynamic SQL can be examined by db2expln, this utility is retained only for backward compatibility.

db2exfmt
The db2exfmt tool can be used to format the contents of the explain tables. This tool is located in the misc subdirectory of the instance sqllib directory. To use the tool, you need read access on the explain tables to be formatted.

You can find more explanation of these tools on *Appendix C, Administration Guide: Performance*.

8.2 Language support

In this section we supply some information on language support and libraries. DB2 supports a wide range of languages through standard libraries. The most common ones for the Windows platforms are:

- CLI
- ODBC
- ADO/OLEDB

Supported Cursor Modes for the IBM OLE DB Provider: The IBM OLE DB Provider for DB2 natively supports read-only and forward-only cursors, called Server Cursors. For updatable scrollable cursors, your application should use the OLE DB Cursor Service Component known as the Client Cursor. OLE DB native applications will have updatable and scrollable cursors available when the IDataInitialize or IDBPromptInitialize OLE DB core interface is used to connect to the database. This is because these interfaces automatically activate the OLE DB Cursor Service Component.

You can find more information on Visual Studio languages integration with DB2 on *Chapter 11, Application Development Guide: Building and Running Applications Version 8*.

Another book with good information is the *DB2 SQL, DB2 Cook Book*. This book can be found on Graeme Birchall's site at:
http://ourworld.compuserve.com/homepages/Graeme_Birchall/HTM_COOK.HTM.

8.2.1 Visual Basic

In this section we highlight the main functionality and tools provided with DB2 that are integrated with Visual Basic.

To activate the DB2 tools on the Visual Basic IDE, do the following steps:

1. Open Visual Basic IDE by clicking **Start Menu—>Programs—>Microsoft Visual Studio 6.0—>Microsoft Visual Basic 6.0**
2. Click the **Add-Ins—>Add-In Manager** menu at the top of Visual Basic IDE. The Add-In Manager pop-up dialog will be shown.
3. Verify whether, in the add-in list, the item **IBM DB2 Development Add-In for Visual Basic** has the load behavior of Startup/Loaded. If not, click it and then check the options **Loaded/Unloaded** and **Load on Startup**.

If you have installed DB2 before Visual Basic, or because of some other problem, the add-in tools are not shown, you can run the **db2vscmd** to register the tools. To use **db2vscmd**, follow these steps:

1. Open a DB2 command line by clicking **Start Menu—>Programs—>IBM DB2 —>Command Line Tools—>Command Window**.
2. Run the command **db2vscmd unregister VB** to free up any possible entry.
3. Run the command **db2vscmd register VB** to register the VB environment.

DB2 tools integrated with Visual Basic IDE

After checking the DB2 Development Add-In option, a DB2 Development Add-In icon is added to the add-in menu. By clicking the icon, the DB2 Development View is opened and the DB2 tools toolbar is added at the top of Visual Basic IDE (Figure 8-42).

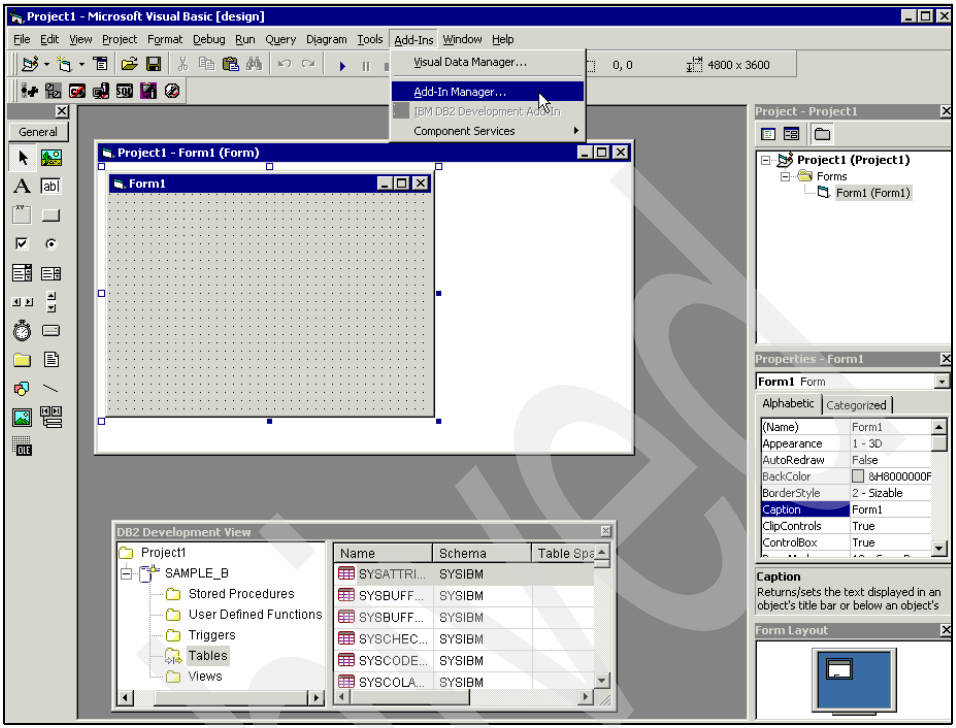

Figure 8-42 Visual Basic IDE and DB2 Development Add-in working.

The IBM DB2 Development Add-Ins toolbar contains the following tools:

- Development Center
- Control Center
- Command Center
- DB2 Client Config Assistants
- SQL Assist
- DB2 Deployment Tool
- DB2 Information Center

With DB2 Development View opened, you can perform the following functions:

- Add database connections by right-clicking the folder icon with the project name and then clicking the **Add Connection** menu option.
- Add ADO connection object or code to the project by right-clicking the database connection icon and then choosing **Add ADO Connection Code** or **Add ADO Connection Object**. To add any of them, you need to create a Data Environment in your project.
- Create and work with stored procedures by right-clicking the stored procedure icon under the database tree.

- Create and work with UDFs by right-clicking the user defined function icon under the database tree.

Libraries available to Visual Basic to do database connection

Visual Basic can connect to DB2 using the following common libraries:

- CLI
- ODBC
- ADO/OLEDB

DSN connection type and network connection type

DB2 drivers support two types of connection:

- **DSN type**: DSN type connections are those use the registered databases on DB2 registry configured manually, by LDAP or script. Following is an example of a ADO connection with DSN information (Example 8-6):

Example 8-6 DSN Type connection code generated by Add ADO Connection Code

```
'create and return ADO Connection Object.
Public Function SAMPLE_B_GetConnection( _
       Optional strUserName As String = "", _
       Optional strPassword As String = "") As ADODB.Connection
   On Error GoTo SAMPLE_B_ErrHandler

   Dim strConnectionString As String
   strConnectionString = "Provider=IBMDADB2; DSN=SAMPLE_B"
   If strUserName <> "" And strPassword <> "" Then
      strConnectionString = strConnectionString & "; User ID=" & _
      strUserName & "; Password=" & strPassword
   End If
   'Create new ADO connection object
   Dim adoConnection As New ADODB.Connection
   With adoConnection
      .CursorLocation = adUseClient
      .ConnectionString = strConnectionString
   End With

   'Open ADO Connection Object
   Call adoConnection.Open

   ‘Set schema to UDBRED
   adoConnection.Execute "SET SCHEMA UDBRED"

   'Return new ADO connection object
   Set SAMPLE_B_GetConnection = adoConnection
   Exit Function

SAMPLE_B_ErrHandler:
```

```
   MsgBox "Error Code: " & Err.Number & vbNewLine & _
          "Description: " & Err.Description & vbNewLine & _
          "Source: " & Err.Source, _
          vbOKOnly + vbCritical
   Err.Clear
   Set SAMPLE_B_GetConnection = Nothing
End Function
```

In this case, the code generated for the database connection type is DSN. You can see the DSN information is set as:

Provider=IBMDADB2; DSN=SAMPLE_B

- **Network type:** Network type connections are those the server name, the port number and the database name are informed. This type of connection does not require database configuration and server information on the client. Following is an example of a ADO connection with network information (Example 8-7).

Example 8-7 Same example using network type connection

```
'create and return ADO Connection Object.
Public Function SAMPLE_B_GetConnection( _
      Optional strUserName As String = "", _
      Optional strPassword As String = "") As ADODB.Connection
   On Error GoTo SAMPLE_B_ErrHandler

   Dim strConnectionString As String
   strConnectionString = "Provider=IBMDADB2.1; Data Source=SAMPLE_B;" & _
                         "Location=bonnie:50000"
   If strUserName <> "" And strPassword <> "" Then
      strConnectionString = strConnectionString & "; User ID=" & _
      strUserName & "; Password=" & strPassword
   End If
   'Create new ADO connection object
   Dim adoConnection As New ADODB.Connection
   With adoConnection
      .CursorLocation = adUseClient
      .ConnectionString = strConnectionString
   End With

   'Open ADO Connection Object
   Call adoConnection.Open

   'Set schema to UDBRED
   adoConnection.Execute "SET SCHEMA UDBRED"

   'Return new ADO connection object
   Set SAMPLE_B_GetConnection = adoConnection
   Exit Function
```

```
SAMPLE_B_ErrHandler:
   MsgBox "Error Code: " & Err.Number & vbNewLine & _
          "Description: " & Err.Description & vbNewLine & _
          "Source: " & Err.Source, _
          vbOKOnly + vbCritical
   Err.Clear
   Set SAMPLE_B_GetConnection = Nothing
End Function
```

Since both connections have the same final result, let us analyze the characteristics of each of them.

DSN connections

Following are some characteristics of DSN connections:

- They hide the server and database configuration.
- The connection configuration can be tuned on each machine.
- The connection information stored on the machine allows quick connection to the database.
- They require a maintenance staff with default authorization. You need to run log-on scripts to configure the connection automatically, or the application should run command-line scripts or call an API to configure the connection.
- They do not allow you to register the same database alias on environments that have databases with the same name, but on different servers.
- They allow the bind privilege on applications.

Network connections

Following are some characteristics of network connections:

- They do not require a connection configuration. The application has the freedom to be configured to connect to the database, or the application can easily be programmed to get the connection information from anywhere that the developer and DBA have planned.
- All connection information should be informed during the connection, such as BLOB configuration and work-around configuration.
- They allow you to use the same database alias on environments that have databases with the same name, but on different servers.
- They provide for a fast connection, since they already have all database information required to connect.
- They do not offer the capability of bind commands and this functionality, since there is no connection configuration on the client.

LDAP environments can alleviate maintenance configuration on DSN-based clients, since DB2 clients and servers can be configured to rely on LDAP server to provide server and database configuration from the domain as a whole. On LDAP environments, the additional connection information can be set on the client as the same way as DSN connections, or they can be retrieved from a external configuration provider where the developer specified in the applications.

Both connection types do not require userid and password information when the client is secure. Windows 95,98 and ME are considered insecure clients and DB2 does not allow connections from them without userid and password on the connection string.

Using parameters on ADO queries

Here is an example of Visual Basic code with parameters (Example 8-8):

Example 8-8 Sample Visual Basic code using parameters

```
   'Create Command object and open
   Dim adoRecordset As ADODB.Recordset
   Dim adoCommand As New ADODB.Command
   Dim adoParameter As ADODB.Parameter

   With adoCommand
      .ActiveConnection = adoConnection
      .CommandText = "SELECT CustID,CustName FROM Customer WHERE
UCASE(CustName) LIKE ? FOR READ ONLY"
   End With

   Set adoParameter = adoCommand.CreateParameter( _
      "CustName", adVarChar, adParamInput, 20, UCASE("JON%"))

   adoCommand.Parameters.Append adoParameter

   Set adoRecordset = adoCommand.Execute
```

Tips for Visual Basic programming with DB2

In this section we provide some tips for Visual Basic programming with DB2.

- Avoid opening and closing connections too frequently. Freeing up DB2 resources is a good practice when they are not being used for a long time or you don't need them anymore. But if you open and close a connection for each operation, your system response time will be impacted. We recommend that you create a common connection object and share it among your systems and check the connection before starting an operation. To free up connections that are not used for a long time, create a timer object on your system to disconnect them from the database.

Other approaches to solve the unused connection problem include using db2governor, or creating a script to query the last time the connection executed an SQL, and drop it if it has not being used for a long time. DB2 also offers connection pooling when you are dealing with loosely coupled systems that require many connections, such as Web pages.

- Use parameters on queries to reutilize SQL paths created by DB2.
- DB2 is case-sensitive in querying data string. To avoid problems when searching or storing data, use the function UCASE or LCASE to convert the parameters. Example: `SELECT CUSTID, CUSTNAME FROM UDBRED.CUSTOMER WHERE UCASE(NAME) = UCASE('john')`.
- Always create error trapping code to avoid your system being crashed by an unmanaged code.
- You may resort to external libraries compiled in C or C++ with static SQL to improve execution time. You can also use stored procedures since they are compiled and require less network traffic to do the job.
- Preferably, get all the data from the queries and then close the recordset when dealing with large tables or high concurrency. You can also work with unbound data controls.
- Preferably, code the begintrans and commit code. This assures that all operations are done as one logical unit, and you do not need to consider whether the server configuration was set to autocommit or not.
- Avoid using bound data controls. Bound data controls require a recordset opened for a long time. These are the main drawbacks:
  - **High resource consumption:** Since usually a data control retrieves a high amount of data that usually are not used at all. Preferably, query only the necessary data, store it in a secondary place to work, and then close the recordset.
  - **Low scalability:** Using bound data controls prevents you from using different and more specialized, suitable locking methods. Since bounded data controls usually retain a large number of rows, this can be an issue when some other operations need data from the row and require locks.
  - **Cursor limitations:** Please note that DB2 OLEDB drivers also implement only read-only and forward-only server-side cursors, thus making the data control working with client-side cursors slower.

Enabling MTS Support in DB2

MTS means Microsoft Transaction System. This is a transaction system product bundled with the Windows server product family. DB2 MTS support requires at least version 2.0 on Windows NT.

MTS has the following components:

- MTS runtime environment
- APIs that allow MTS applications to perform distributed transactions (two-phase commits)
- MTS Explorer, a graphical tool used to create, administer, and monitor MTS components
- Distributed Transaction Coordinator (DTC), a backend component which coordinates two-phase commit processing.

MTS provides Distributed Transactions (DTC service) and automatic management of processes and threads. Single user ActiveX components are automatically available to multiple users when run within the MTS runtime.

MTS has the following features:

- It is both network protocol and database independent.
- It supports both XA and OLE transactions.
- It is similar to CICS, but uses OLE TX protocol instead of XA.
- It provides an OLE TX TO XA Mapper so that it works with the Resource Manager that supports XA.
- It requires an ODBC driver to perform OLE TX to XA mapping on the application side. ODBC 3.0+ and ODBC 3.5 Driver Managers support MTS.

The application server logic uses ActiveX DLLs running inside MTS.

MTS components are language neutral and can work with following languages:

- Visual Basic
- Visual J++
- C/C++
- COBOL
- Delphi

Component communications are managed by MTS and DCOM. All systems must have DCOM installed and running.

DB2 provides MTS application sample programs. These samples are under the **SQLLIB\samples\MTS** directory. Copy the files to your working directory prior to modifying the sample programs.

Configuring DB2 server to run MTS packages:
Following are procedures for configuring DB2 to run MTS packages:

1. Set the DBM CFG parameter TP_MON_NAME to MTS by running
   `db2 update dbm cfg tp_mon_name='MTS'`
2. Have Service Pack 4 for Windows NT installed (includes fixes for MTS).

Building your own MTS application using Microsoft Visual Basic:
In this section, we provide the steps for building MTS application using Microsoft Visual Basic:

1. Check Unattended Execution in Project Properties. This allows Visual Basic to intercept any MsgBox calls and write them to Windows NT log.
2. Set a reference to Microsoft ActiveX Data Objects in **Project—>References**. This will enable database support.
3. When the DLL is built, set **binary compatibility** in **Project—>Properties—> Component** window for future changes.
4. After the DLL is created, use MTS explorer to create a new package and to import the DLL function into the new project.
5. For the sample application packages db2com needs to create. Import the following components into db2com from db2com.dll:
   - db2com.UpdateNumberColumn
   - db2com.UpdateRow
   - db2com.UpdateStringColumn
   - db2com.VerifyUpdate
6. Use the MTS Explorer to start MSDTC service if is not already started.

When creating an MTS client executable:

1. Add a reference to the server component previously created. Select Project | References and check the box containing the MTS package (created in step 4 above).
2. Set the client project to be the active project: highlight the project, right-click, and select **Set as startup**.

8.2.2 Visual C++

In this section we highlight the main functionality and tools provided with DB2 that integrate with Visual C++ and present some points of interest on DB2 with Visual C++.

To use DB2 add-in tools for Visual C++, follow these steps:

1. Open or create a new project.

2. On the menu at the top, select **Tools—>Customize**.
3. On the customize dialog, click the **Add-Ins and Macro Files** tab.
4. Check the option **IBM DB2 Development Add-In for Visual C++** to activate the ADO connection generator.
5. Check the option **IBM DB2 Tools Add-In** to activate additionalDB2 tools.

If you have installed DB2 before Visual C++ or, because of some other problem, it is not showing, you can run the **db2vscmd** to register the tools. To use **db2vscmd**, proceed this way:

1. Open a DB2 command line clicking **Start Menu—>Programs—>IBM DB2 —>Command Line Tools—>Command Window**.
2. Run the command **db2vscmd unregister VC** to free up any possible entry.
3. Run the command **db2vscmd register VC** to register the VC environment.

DB2 tools integrated with Visual C++ IDE

After checking and confirming the desired DB2 tools, tool bars will appear on the dialog top of Visual C++ (Figure 8-43).

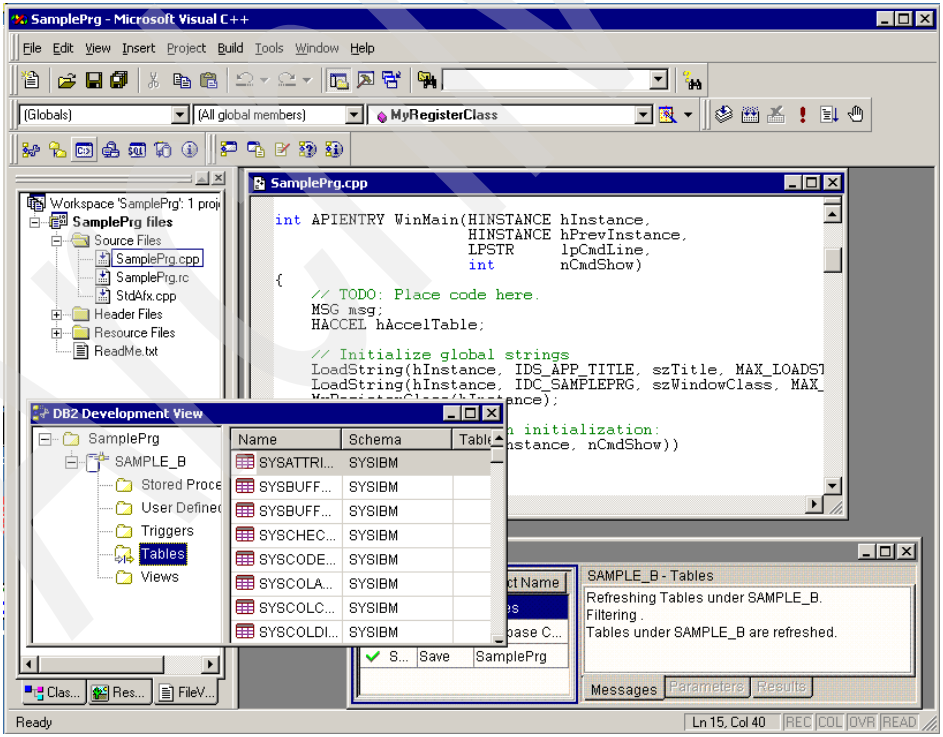

Figure 8-43 Visual C++ IDE and DB2 Development Add-in working

The IBM DB2 Tools toolbar contains the following tools:

- Development Center
- Control Center
- Command Center
- DB2 Client Config
- SQL Assist
- DB2 Deployment Tool
- DB2 Information Center

The IBM DB2 Development Add-Ins toolbar contains the following tools:

- Development View
- Output View
- Editor
- Development View Help

Libraries available to Visual C++ to do database connection

Like Visual Basic, Visual C++ can connect to DB2 using the most common libraries as follows:

- CLI
- ODBC
- ADO/OLEDB

Tips for Visual C++ programming with DB2

- Avoid opening and closing connections too frequently. Freeing up DB2 resources is a good practice when they seldom used for a long time or unneeded. But if you open and close a connection for each operation, your system response time will be impacted. We recommend that you create a common connection object and share it among your system and check the connection before starting an operation. To free up connections that are unused for a long time, create a timer object on your system to disconnect from the database. Other approaches to solve the unused connection problem are include using db2governor or creating a script to query the last time the connection executed an SQL and then drop it if it has not being used for a long time. DB2 also offers connection pooling when you are dealing with loosely coupled systems that require many connections, such as Web pages.
- Use parameters on queries to reutilize SQL paths created by DB2.
- Always create error trapping code to avoid your system crashed by an unmanaged code.
- You can also use stored procedures since they require less network traffic to do the job.
- Preferably, get all the data from the queries and then close the recordset when dealing with large tables or high concurrency.

8.2.3 Java

Java is one of the most popular Web application development languages. DB2 supports Java on various platforms,including Windows. In order to build Java applications on a Windows operating system with DB2 JDBC support, you need to install and configure the following components on your machine:

- One of the following:
  - IBM Developer Kit and Runtime Environment for Window, Java 2 Technology Edition, Version 1.3.1, 32-bit version. This product gets installed with DB2 if you use DB2 Setup.
  - Java Development Kit (JDK) 1.3.1 for Win32 from Sun Microsystems.
- DB2 Java Enablement, provided on DB2 UDB Version 8 for Windows clients and servers.

To run a DB2 Java routine, you need to update the DB2 database manager configuration on the server to include the path where the JDK is installed on that machine. You can do this by entering the following on the server command line:

```
db2 update dbm cfg using JDK_PATH c:\jdk13
```

Here, c:\jdk13 is the path where the JDK is installed.

DB2 provides a rich set of sample programs to demonstrate building and running JDBC program. These sample programs can be found in the following DB2 directories:

- sqllib\samples\java
- sqllib\samples\java\jdbc
- sqllib\samples\java\sqlj
- sqllib\samples\java\Webshpere
- sqllib\samples\java\plugin
- sqllib\samples\java\plugin\doc

8.2.4 COBOL

Another programming language DB2 supports for Windows systems is COBOL. You can develop a DB2 Version 8 COBOL application using this software:

- Micro Focus COBOL Version 4.0.20
- Micro Focus COBOL Net Express Version 3.1.0
- IBM VisualAge COBOL Version 2.0

For more detailed information on developing COBOL applications, refer to the DB2 Version 8 manual, *Application Development Guide: Building and Running Applications*, SG09-4825.

8.3 Migration Toolkit (MTK)

The IBM DB2 Migration Toolkit helps you migrate from Sybase, and Microsoft SQL Server (Versions 6 and 7) to DB2 UDB Version 8.1 databases on any supported DB2 UDB workstation platform. The toolkit can also aid in your migration to DB2 UDB Server for OS/390 and z/OS, and DB2 UDB for AS/400. It runs on AIX, Linux, Sun Solaris and Windows NT, Windows 2000, is in English only, and was updated in October 2002.

These are the possibilities with MTK:

- You can migrate simple databases with the included wizard.
- You can migrate complex databases with the full functioning GUI interface, which gives you more options to further refine the migration. For example, you can change the default choices that are made about which DB2 data type to map to the corresponding source database data type.
- The toolkit converts to and refines DB2 database scripts. This model makes the toolkit very portable, allowing you to import and convert on a machine remotely from where the source database and DB2 are installed.

The home page of MTK is:

http://www-3.ibm.com/software/data/db2/migration/mtk

There you will find a link to the tutorial. A very detailed article can also be found at:

http://www7b.software.ibm.com/dmdd/library/techarticle/0209jarzebowicz/0209jarzebowicz.html

8.4 Application development tips

When you are trying to design a new database system or analyze an existing database system, one of the most important considerations is your application design. Even though your database is well designed and tuned, inappropriate design of applications may cause performance problems. If your application has a design problem, fixing this often improves the application performance much more than tuning the configuration parameters of DB2 UDB.

For example, SQL is a high-level language with much flexibility. Different SELECT statements can be written to retrieve the same data; however, the performance can vary for the different forms of SELECT statements. This is because one statement may have a higher processing cost than another. In such a case, you should choose the SQL statement which has the lower processing cost, so that the application will have good performance.

In this section, we discuss application design considerations to obtain better performance. These include:

- Tips to write better SQL statements
- Minimizing data movement between applications and database
- Considerations for embedded SQL programs
- Considerations for Call Level Interface and ODBC

8.4.1 Tips to write better SQL statements

DB2 UDB provides the SQL compiler which creates the compiled form of SQL statements. When the SQL compiler compiles SQL statements, it rewrites them into a form that can be optimized more easily. This is known as query rewrite.

The SQL compiler then generates many alternative execution plans for satisfying the user's request. It estimates the execution cost of each alternative plan using the statistics for tables, indexes, columns, and functions, and chooses the plan with the smallest execution cost. This is known as query optimization.

It is important to note that the SQL compiler (including the query rewrite and optimization phases) must choose an access plan that will produce the result set for the query you have coded. Therefore, as noted in many of the following guidelines, you should code your query to obtain only the data that you need. This ensures that the SQL compiler can choose the best access plan for your needs.

Guidelines for using a SELECT statement are these:

- Specify only needed columns.
- Limit the number of rows.
- Specify the FOR UPDATE clause if applicable.
- Specify the OPTIMIZED FOR n ROWS clause.
- Specify the FETCH FIRST n ROWS ONLY clause if applicable.
- Specify the FOR FETCH ONLY clause if applicable.
- Avoid numeric data type conversion.

We further explore each of these guidelines.

Specify only needed columns in the select list

Specify only those columns that are needed in the select list. Although it may be simpler to specify all columns with an asterisk (*), needless processing and returning of unwanted columns can result.

Limit the number of rows by using predicates

Limit the number of rows selected by using predicates to restrict the answer set to only those rows that you require. There are four categories of predicates, each with its own processing cost. The category is determined by how and when that predicate is used in the evaluation process. These categories are listed below, ordered in terms of performance, starting with the most favorable:

- Range delimiting predicates
- IndexSARGable predicates
- Data SARGable predicates
- Residual predicates

Note: SARGable refers to something that can be used as a search argument.

Range delimiting predicates are those used to bracket an index scan. They provide start and/or stop key values for the index search. Index SARGable predicates are not used to bracket a search, but can be evaluated from the index because the columns involved in the predicate are part of the index key. For example, assume that an index has been defined on the NAME, DEPT, and YEARS columns of the STAFF table, and you are executing the following select statement:

```
SELECT name, job, salary
FROM staff
WHERE name = 'John' and dept = 10 years > 5
```

The first two predicates (name='John', dept=10) would be range delimiting predicates, while years > 5 would be an index SARGable predicate, as the start key value for the index search cannot be determined by this information only. The start key value may be 6, 10, or even higher. If the predicate for the years column is years => 5, it would be a range delimiting predicate, as the index search can start from the key value 5.

```
SELECT name, job, salary
FROM staff
WHERE name = 'John' dept = 10 years => 5
```

The database manager will make use of the index data in evaluating these predicates, rather than reading the base table. These range delimiting predicates and index SARGable predicates reduce the number of data pages accessed by reducing the set of rows that need to be read from the table. Index SARGable predicates do not affect the number of index pages that are accessed.

Data SARGable predicates are the predicates that cannot be evaluated by the Index Manager, but can be evaluated by Data Management Services (DMS). Typically, these predicates require the access of individual rows from a base table. If required, Data Management Services will retrieve the columns needed to

evaluate the predicate, as well as any others to satisfy the columns in the SELECT list that could not be obtained from the index.

For example, assume that a single index is defined on the projno column of the project table but not on the deptno column, and you are executing the following query:

```
SELECT projno, projname, repemp
FROM project
WHERE deptno='D11'
ORDER BY projno
```

The predicate deptno='D11' is considered data SARGable, because there are no indexes on the deptno column, and the base table must be accessed to evaluate the predicate.

Residual predicates, typically, are those that require I/O beyond the simple accessing of a base table. Examples of residual predicates include those using quantified sub-queries (sub-queries with ANY, ALL, SOME, or IN), or reading LONG VARCHAR or large object (LOB) data (they are stored separately from the table).

These predicates are evaluated by Relational Data Services (RDS). Residual predicates are the most expensive of the four categories of predicates.

As residual predicates and data SARGable predicates cost more than range delimiting predicates and index SARGable predicates, you should try to limit the number of rows qualified by range delimiting predicates and index SARGable predicates whenever possible.

Specify the FOR UPDATE clause

If you intend to update fetched data, you should specify FOR UPDATE clause in the SELECT statement of the cursor definition. By doing this, the database manager can initially choose appropriate locking levels, for instance, U (update) locks instead of S (shared) locks. Thus you can save the cost to perform lock conversions from S locks to U locks when the succeeding UPDATE statement is processed.

The other benefit to specifying the FOR UPDATE clause is that can decrease the possibility of deadlock, deadlock is a situation in which more than one application is waiting for another application to release a lock on data, and each of the waiting applications is holding data needed by other applications through locking. Let us suppose two applications are trying to fetch the same row and update it simultaneously in the following order:

1. Application1 fetches the row
2. Application2 fetches the row

3. Application1 updates the row
4. Application2 updates the row

In step 4, Application2 should wait for Application1 to complete the update and release the held lock, and then start its updating. However, if you do not specify the FOR UPDATE clause when declaring a cursor, Application1 acquires and holds an S (shared) lock on the row (step 1). That means the second application can also acquire and hold an S lock without lock-waiting (step 2). Then the first application tries to get a U (update) lock on the row to process an UPDATE statement, but must wait for the second application to release its holding S lock (step 3). Meanwhile the second application also tries to get a U lock and gets into the lock-waiting status due to the S lock held by the first application (step 4). This situation is a deadlock, and the transaction of the first or second application will be rolled back.

If you specify the FOR UPDATE clause in the DECLARE CURSOR statement, the U lock will be imposed when Application1 fetches the row, and Application2 will wait for Application1 to release the U lock. Thus, no deadlock will occur between the two applications.

Here is how to use the UPDATE OF clause in a SELECT statement:

```
EXEC SQL DECLARE c1 CURSOR FOR select * from employee
FOR UPDATE OF job;
EXEC SQL OPEN c1;
EXEC SQL FETCH c1 INTO...;
if ( strcmp (change,"YES") == 0)
EXEC SQL UPDATE employee SET job=:newjob
WHERE CURRENT OF c1;
EXEC SQL CLOSE c1;
```

Specify the OPTIMIZE FOR n ROWS clause

Specify the OPTIMIZE FOR n ROWS clause in the SELECT statement when the number of rows you want to retrieve is significantly less than the total number of rows that could be returned. Use of the OPTIMIZE FOR clause influences query optimization based on the assumption that n rows will be retrieved. This clause also determines the number of rows that are blocked in the communication buffer.

```
SELECT projno,projname,repemp
FROM project
WHERE deptno='D11'
OPTIMIZE FOR 10 ROWS
```

Row blocking is a technique that reduces database manager overhead by retrieving a block of rows in a single operation. These rows are stored in a cache, and each FETCH request in the application gets the next row from the cache. If

you specify OPTIMIZE FOR 10 ROWS, a block of rows is returned to the client every ten rows.

Note: The OPTIMIZE FOR n ROWS clause does not limit the number of rows that can be fetched or affect the result in any way other than performance. Using OPTIMIZE FOR n ROWS can improve the performance if no more than n rows are retrieved, but may degrade performance if more than n rows are retrieved.

Specify the FETCH FIRST n ROWS ONLY clause

Specify the FETCH FIRST n ROWS ONLY clause if you do not want the application to retrieve more than n rows, regardless of how many rows there might be in the result set when this clause is not specified. This clause cannot be specified with the FOR UPDATE clause. For example, with the following coding, you will not receive more than 5 rows:

```
SELECT projno,projname,repemp
FROM project
WHERE deptno='D11'
FETCH FIRST 5 ROWS ONLY
```

The FETCH FIRST n ROWS ONLY clause also determines the number of rows that are blocked in the communication buffer. If both the FETCH FIRST n ROWS ONLY and OPTIMIZE FOR n ROWS clause are specified, the lower of the two values is used to determine the communication buffer size.

Specify the FOR FETCH ONLY clause

If you have no intention of updating rows retrieved by a SELECT statement, specify the FOR FETCH ONLY clause in the SELECT statement. It can improve performance by allowing your query to take advantage of row blocking. It can also improve data concurrency since exclusive locks will never be held on the rows retrieved by a query with this clause specified.

Note: Instead of the FOR FETCH ONLY clause, you can also use the FOR READ ONLY clause. 'FOR READ ONLY' is a synonym for 'FOR FETCH ONLY'.

Avoid data type conversions

Data type conversions (particularly numeric data type conversions) should be avoided whenever possible. When two values are compared, it may be more efficient to use items that have the same data type. For example, suppose you are joining TableA and TableB using column A1 of TableA and column B1 of TableB, as in the following example.

```
SELECT * FROM TableA,TableB WHERE A1=B1
```

If columns A1 and B1 are the same data type, no data type conversion is required. But if they are not the same data type, a data type conversion occurs to compare values at run time and it might affect the performance. For example, if A1 is a decimal column and B1 is an integer column and each has a value ‘123’, data type conversion is needed, as TableA stores it as x‘123C’, whereas TableB stores it as x‘7B’.

Also, inaccuracies due to limited precision may result when data type conversions occur.

Other considerations for data types

DB2 UDB allows you to use various data types. You can use SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, and DOUBLE for numeric data; CHAR, VARCHAR, LONG VARCHAR, CLOB for character data; GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, and DBCLOB for the double byte character data, and so on. As the amount of database storage and the cost to process varies depending on the data type, you should choose the appropriate data type.

The following are guidelines when choosing a data type:

- Use character (CHAR) rather than varying-length character (VARCHAR) for short columns. The varying-length character data type can save database storage when the length of data values varies, but there is a cost to check the length of each data value.
- Use VARCHAR or VARGRAPHIC rather than LONG VARCHAR or LONG VARGRAPHIC. The maximum length for VARCHAR and LONG VARCHAR columns, VARGRAPHIC and LONG VARGRAPHIC are almost same (32,672 bytes for VARCHAR, 32,700 bytes for LONG VARCHAR, 16,336 characters for VARGRAPHIC, and 16,350 characters for LONG VARGRAPHIC), while LONG VARCHAR and LONG VARGRAPHIC columns have several restrictions. For example, data stored in LONG VARCHAR or LONG VARGRAPHIC columns is not buffered in the database buffer pool.
- Use integer (SMALLINT, INTEGER, BIGINT) rather than floating-point number (REAL or DOUBLE) or decimal (DECIMAL) if you do not need to have the fraction part. Processing cost for integers is much more inexpensive.
- Use date-time (DATE, TIME, TIMESTAMP) rather than character (CHAR). Date-time data types consume less database storage, and you can use some built-in functions for date-time data types such as YEAR and MONTH.
- Use numeric data types rather than character.

8.4.2 Minimizing data movement between applications and database

Network costs are often the performance gating factor for applications. A good first step in investigating this is to run the application, or individual queries from it, locally on the server and see how much faster it runs. That, and the use of network monitoring tools, can indicate if network tuning or a faster network is called for. Note that here “network” includes the local case, since even though local connections have much less overhead than a network protocol, the overhead is still significant.

If the network itself is in good shape, you should focus on reducing the number of calls that flow from the application to the database (even for local applications).

There are several ways to reduce network costs. Here we introduce two ways which involve having multiple actions take place through one call.

Compound SQL

Compound SQL is a technique to build one executable block from several SQL statements. When compound SQL is being executed, each SQL statement in the block is executed individually, but the number of requests transmitting between client and server can be reduced.

Here is an example executing two UPDATE statements and one INSERT statement using compound SQL:

```
EXEC SQL BEGIN COMPOUND ATOMIC STATIC
UPDATE tablea SET cola = cola * :var1;
UPDATE tableb SET colb = colb + :var2;
INSERT INTO tablec (colc,cold,cole) VALUES (:i,:j,0);
END COMPOUND;
```

When you execute compound SQL, you can choose two types of compound SQL, atomic and non-atomic. The type determines how the entire block is handled when one or more SQL statements in the block happen to end in error. The type of the example above is atomic.

Atomic: If one of the statements in the block ends in error, the entire block is considered to have ended in and error, and any changes made to the database within the block will be rolled back.

Non-atomic: When all statements have completed, the application receives a response. Even if one or more statements in the block end in error, the database manager will attempt to execute all statements in the block. If the unit of work containing the compand SQL is rolled back, then all changes made to the database within the block will be rolled back.

You can also use compound SQL to improve performance of the IMPORT command, which repeats data inserts. When you specify MODIFIED BY COMPOUND=x option (x is the number of inserts compounded into one block), the IMPORT command builds a block from multiple inserts. You will gain significant performance improvement from this option. See the DB2 UDB V8 Command Reference manual for more information on the IMPORT command.

Stored procedures

A stored procedure resides on a database server, where it executes, and accesses the database locally to return information to client applications. Using stored procedures allows a client application to pass control to a stored procedure on the database server. This allows the stored procedure to perform intermediate processing on the database server, without transmitting unnecessary data across the network. Only those records that are actually required are transmitted. This can result in reduced network traffic and better overall performance.

A stored procedure also saves the overhead of having a remote application pass multiple SQL statements to a database on a server. With a single statement, a client application can call the stored procedure, which then performs the database access work and returns the results to the client application. The more SQL statements that are grouped together for execution in the stored procedure, the larger the savings resulting from avoiding the overhead associated with network flows for each SQL statement when issued from the client.

To create a stored procedure, you must write the application in two separate procedures. The calling procedure is contained in a client application and executes on the client. The stored procedure executes at the location of the database on the database server.

See the section “Stored Procedure Builder” on page 388 for additional information on Stored Procedures.

8.4.3 Considerations for embedded SQL programs

Embedded SQL programs are those statements in which SQL statements are embedded. You can write embedded SQL programs in the C/C++, COBOL, FORTRAN, Java (SQLJ), and REXX programming languages, and enable them to perform any task supported by SQL, such as retrieving or storing data.

Static SQL

There are two types of embedded SQL statements: static and dynamic. Static SQL statements are ones where the SQL statement type and the database objects accessed by the statement, such as column names, are known prior to

running the application. The only unknowns are the data values the statement is searching for or modifying.

You must pre-compile and bind such applications to the database so that the database manager analyzes all of static SQL statements in a program, determines its access plan to the data, and store the ready-to-execute application package before executing the program. Because all logic to execute SQL statements is determined before executing the program, static SQL programs have the least run-time overhead of all the DB2 programming methods, and execute faster.

To prepare all SQL statements in a static embedded SQL program, all database objects being accessed must exist when binding the package. If you want to bind the package when one or more database objects are missing, specify the option SQLERROR CONTINUE in conjunction with VALIDATE RUN in the BIND or PREP command. Though you encounter errors for SQL statements which try to access missing database objects, the package will be bound. For the SQL statements which had errors during the bind, the rebind process will be performed at execution time. If you want to have the least run-time overhead, make sure that all database objects being accessed exist during the bind.

When and how the access plan is determined

The DB2 optimizer determines the best access plans for static SQL programs when the bind operation is performed. The determination is done based on the statistical information stored in the system catalog. Obsolete statistical information may lead the DB2 optimizer to select inefficient access plans and may cause performance problems. Therefore, it is very important to collect up-to-date statistical information using the RUNSTATS utility before binding packages.

Since static SQL statements are processed based on the access plans determined during the bind, there might be better access plans if you make numerous changes to the database after the bind. For example, assuming you have a very small table with an index and your application has static SQL statements retrieving data from the index keys, then the application tends to access the table using table scans rather than index scans because the size is too small to benefit from index scans; however, if the table considerably grows up, index scans are preferable. In such a case, you should consider executing RUNSTATS to refresh the table and index's statistical information stored in the system catalog, and execute the REBIND command to take new access plans for the static SQL statements.

There are various forms of RUNSTATS, but a good default to use is:

```
RUNSTATS ON TABLE xxx AND DETAILED INDEXES ALL
```

Adding the WITH DISTRIBUTION clause can be a very effective way to improve access plans where data is not uniformly distributed.

Access path to volatile tables

If you have volatile tables whose size can vary from empty to quite large at run time, relying on the statistics collected by RUNSTATS to generate an access path to a volatile table can be misleading. For example, if the statistics were collected when the volatile table was empty the optimizer tends to favor accessing the volatile table using a table scan rather than an index scan. If you know a table is volatile, you can let the DB2 optimizer to select index scans regardless of the existing statistics of these tables by executing the ALTER TABLE statement with the VOLATILE option.

```
ALTER TABLE tablename VOLATILE
```

When a table is known to be volatile to the optimizer, it will favor index scans rather than table scans. This means that access paths to a volatile table will not depend on the existing statistics on this table. These statistics will be ignored by the optimizer because they can be misleading, in the sense that they are static and do not reflect the current content of the table.

To deactivate the volatile option and let the DB2 optimizer choose access paths based on the existing statistics, execute the following statement:

```
ALTER TABLE tablename NOT VOLATILE
```

Dynamic SQL

Dynamic SQL statements are ones that your application builds and executes at run time. An interactive application that prompts the end user for key parts of an SQL statement, such as the names of the tables and columns to be searched, is a good example of dynamic SQL. The application builds the SQL statement while it is running, and then submits the statement for processing. Generally, dynamic SQL statements are well-suited for applications that run against a rapidly changing database where transactions need to be specified at run time.

When and how the access plan is determined

Dynamic embedded SQL requires the precompile, compile, and link phases of application development. However, the binding or selection of the most effective data access plan is performed at program execution time, as the SQL statements are dynamically prepared.

An embedded dynamic SQL programming module will have its data access method determined during the statement preparation and will utilize the database statistics available at query execution time. Choosing an access plan at program execution time has some advantages as well as a drawback.

Some of the advantages are:

- Current database statistics are used for each SQL statement.
- Database objects do not have to exist before run time.
- This method is more flexible than using static SQL statements.

One drawback is that dynamic SQL statements can take more time to execute, since queries are optimized at run time. To improve your dynamic SQL program's performance, the following are key:

- Execute RUNSTATS after making significant updates to tables or creating indexes.
- Minimize preparation time for dynamic SQL statements.

Keeping statistics information up-to-date helps the DB2 optimizer to choose the best access plan. You do not need to rebind packages for dynamic SQL programs after executing RUNSTATS since access plan is determined at run time.

In the following section, we discuss how to minimize preparation time for dynamic SQL statements.

Avoid repeated PREPARE statements

When an SQL statement is prepared, it is parsed, optimized, and made ready to be executed. The cost of preparing can be very significant, especially relative to the cost of executing very simple queries (it can take longer to prepare such queries than to execute them). To improve the performance of dynamic SQL, the global package cache was introduced in DB2 UDB Version 5.0. The generated access plan for an SQL statement is stored in the global package cache, and it can be reused by the same or other applications. Thus, if the same exact statement is prepared again, the cost will be minimal. However, if there is any difference in syntax between the old and new statements, the cached plan cannot be used.

For example, suppose the application issues a PREPARE statement for the statement:

```
SELECT * FROM EMPLOYEE WHERE empno = ‘000100'
```

Then it issues another PREPARE statement (the same statement, but with a different literal value):

```
SELECT * FROM EMPLOYEE WHERE empno = ‘000200'
```

The cached plan for the first statement cannot be reused for the second, and the latter's PREPARE time will be non-trivial. See the following example:

```
strcpy (st1,"SELECT * FROM EMPLOYEE WHERE empno=‘000100’");
strcpy (st2,"SELECT * FROM EMPLOYEE WHERE empno=‘000200’");
```

```
EXEC SQL PREPARE s1 FROM :st1;
EXEC SQL PREPARE s2 FROM :st2;
EXEC SQL DECLARE c1 CURSOR FOR s1;
EXEC SQL DECLARE c2 CURSOR FOR s2;
EXEC SQL OPEN c1;
EXEC SQL OPEN c2;
```

The solution is to replace the literal ‘000100' by a question mark (?), issue a PREPARE, declare the cursor for the statement, assign the literal when opening the cursor. By changing the program variable(s) appropriately before each OPEN statement, you can reuse the prepared statement. See the following example:

```
strcpy (st,"SELECT * FROM EMPLOYEE WHERE empno='?'");
EXEC SQL PREPARE s1 FROM :st;
EXEC SQL DECLARE c1 CURSOR FOR s1;
EXEC SQL DECLARE c2 CURSOR FOR s1;
strcpy (parmvar1,"000100");
strcpy (parmvar2,"000100");
EXEC SQL OPEN c1 using :parmvar1;
...
EXEC SQL OPEN c2 using :parmvar2;
```

Parameter markers can and should be used, not just for SELECT, but also for repeated executions of INSERT, UPDATE, or DELETE statements. For example, if your application is using EXECUTE IMMEDIATE to execute multiple statements that differ only in the literal values they contain, those EXECUTE IMMEDIATE statements should be replaced by PREPARE and EXECUTE statements using parameter markers. See the following example to read records from a file and insert them into a table:

```
for ( end of file ) {
...
//Read a record from the input file;
//Generate INSERT statement to store the record
//into a table and save the statement into
//the host variable stmt;
...
EXEC SQL EXECUTE IMMEDIATE :stmt
}
```

In this example, generated INSERT statements are prepared and executed for each record being inserted. If the input file has many rows, preparing all INSERT statements will be expensive. You should change this example as follows:

```
strcpy( stmt,"INSERT INTO tablea VALUES (?,?,?)");
EXEC SQL PREPARE st FROM :stmt;
for ( end of file ) {
...
//Read a record from the input file;
```

```
//Assign read values into the host
//variables (var1,var2,var3) for the parameter markers;
EXEC SQL EXECUTE st USING :var1,:var2,:var3;
}
```

This example can complete the insert job faster, since only one INSERT statement is prepared and reused for all rows being inserted.

Tune optimization level

Sometimes another cause of long preparation times is the use of a query optimization class that is higher than necessary. That is, the DB2 Optimizer can spend more time finding the best access plan for a query than is justified by a reduction in execution time.

For example, if the database has large number of concurrent activity, numerous simple SQL statements to process, and a requirement to perform them in seconds, set the optimization class to a lower value such as 1 or 2 by the SET CURRENT QUERY OPTIMIZATION statement. If you do not set any optimization level in the CURRENT QUERY OPTIMIZATION special register, the DB2 optimizer will use the value set in the DFT_QUERYOPT database configuration parameter.

Most statements can be adequately optimized with a reasonable amount of resources by using optimization class 5, which is the default query optimization class. At a given optimization class, the query compilation time and resource consumption is primarily influenced by the complexity of the query, particularly the number of joins and subqueries. However, compilation time and resource usage are also affected by the amount of optimization performed.

Query optimization classes 1, 2, 3, 5, and 7 are all suitable for general-purpose use. Consider class 0 only if you require further reductions in query compilation time and you know that the SQL statements are extremely simple.

When you select an optimization class, consider the following general guidelines:

- Start by using the default query optimization class, class 5.
- To use a class other than the default, try class 1, 2 or 3 first. Classes 0, 1, and 2 use the Greedy join enumeration algorithm.
- Use optimization class 1 or 2 if you have many tables with many of the join predicates that are on the same column, and if compilation time is a concern.
- Use a low optimization class (0 or 1) for queries having very short run-times of less than one second. Such queries tend to have the following characteristics:
  - Access to a single or only a few tables
  - Fetch a single or only a few rows
  - Use fully qualified, unique indexes.

Online transaction processing (OLTP) transactions are good examples of this kind of SQL.

- Use a higher optimization class (3, 5, or 7) for longer running queries that take more than 30 seconds.
- Classes 3 and above use the Dynamic Programming join enumeration algorithm. This algorithm considers many more alternative plans, and might incur significantly more compilation time than classes 0, 1, and 2, especially as the number of tables increases.
- Use optimization class 9 only if you have specific extraordinary optimization requirements for a query.

Complex queries might require different amounts of optimization to select the best access plan. Consider using higher optimization classes for queries that have the following characteristics:

- Access to large tables
- A large number of predicates
- Many subqueries
- Many joins
- Many set operators, such as UNION and INTERSECT
- Many qualifying rows
- GROUP BY and HAVING operations
- Nested table expressions
- A large number of views.

Decision support queries or month-end reporting queries against fully normalized databases are good examples of complex queries for which at least the default query optimization class should be used.

Use higher query optimization classes for SQL that was produced by a query generator. Many query generators create inefficient SQL. Poorly written queries, including those produced by a query generator, require additional optimization to select a good access plan. Using query optimization class 2 and higher can improve such SQL queries.

Optimization classes

You can specify one of the following optimizer classes when you compile an SQL query:

- **0**: This class directs the optimizer to use minimal optimization to generate an access plan. This optimization class has the following characteristics:
  - Non-uniform distribution statistics are not considered by the optimizer.
  - Only basic query rewrite rules are applied.
  - Greedy join enumeration occurs.

- Only nested loop join and index scan access methods are enabled.
- List prefetch and index ANDing are not used in generated access methods.
- The star-join strategy is not considered.

This class should only be used in circumstances that require the lowest possible query compilation overhead. Query optimization class 0 is appropriate for an application that consists entirely of very simple dynamic SQL statements that access well-indexed tables.

- **1**: This optimization class has the following characteristics:
  - Non-uniform distribution statistics are not considered by the optimizer.
  - Only a subset of the query rewrite rules are applied.
  - Greedy join enumeration occurs.
  - List prefetch and index ANDing are not used in generated access methods although index ANDing is still used when working with the semijoins used in star joins.

  Optimization class 1 is similar to class 0 except that Merge Scan joins and table scans are also available.
- **2**: This class directs the optimizer to use a degree of optimization significantly higher than class 1, while keeping the compilation cost significantly lower than classes 3 and above for complex queries. This optimization class has the following characteristics:
  - All available statistics, including both frequency and quantile non-uniform distribution statistics, are used.
  - All query rewrite rules are applied, including routing queries to materialized query tables, except computationally intensive rules that are applicable only in very rare cases.
  - Greedy join enumeration is used.
  - A wide range of access methods are considered, including list prefetch and materialized query table routing.
  - The star-join strategy is considered, if applicable.

  Optimization class 2 is similar to class 5 except that it uses Greedy join enumeration instead of Dynamic Programming. This class has the most optimization of all classes that use the Greedy join enumeration algorithm, which considers fewer alternatives for complex queries, and therefore consumes less compilation time than classes 3 and above. Class 2 is recommended for very complex queries in a decision support or online analytic processing (OLAP) environment. In such environments, specific

queries are rarely repeated exactly, so that a query access plan is unlikely to remain in the cache until the next occurrence of the query.

- **3**: This class requests a moderate amount of optimization. This class comes closest to matching the query optimization characteristics of DB2 for MVS/ESA, OS/390, or z/OS. This optimization class has the following characteristics:
  - Non-uniform distribution statistics, which track frequently occurring values, are used if available.
  - Most query rewrite rules are applied, including subquery-to-join transformations.
  - Dynamic programming join enumeration, as follows:
    - Limited use of composite inner tables
    - Limited use of Cartesian products for star schemas involving look-up tables
  - A wide range of access methods are considered, including list prefetch, index ANDing, and star joins.

  This class is suitable for a broad range of applications. This class improves access plans for queries with four or more joins. However, the optimizer might fail to consider a better plan that might be chosen with the default optimization class.
- **5**: This class directs the optimizer to use a significant amount of optimization to generate an access plan. This optimization class has the following characteristics:
  - All available statistics are used, including both frequency and quantile distribution statistics.
  - All of the query rewrite rules are applied, including the routing of queries to materialized query tables, except for those computationally intensive rules which are applicable only in very rare cases.
  - Dynamic programming join enumeration, as follows:
    - Limited use of composite inner tables
    - Limited use of Cartesian products for star schemas involving look-up tables
  - A wide range of access methods are considered, including list prefetch, index ANDing, and materialized query table routing.

  When the optimizer detects that the additional resources and processing time are not warranted for complex dynamic SQL queries, optimization is reduced. The extent or size of the reduction depends on the machine size and the number of predicates.

When the query optimizer reduces the amount of query optimization, it continues to apply all the query rewrite rules that would normally be applied. However, it does use the Greedy join enumeration method and reduces the number of access plan combinations that are considered.

Query optimization class 5 is an excellent choice for a mixed environment consisting of both transactions and complex queries. This optimization class is designed to apply the most valuable query transformations and other query optimization techniques in an efficient manner.

- **7**: This class directs the optimizer to use a significant amount of optimization to generate an access plan. It is the same as query optimization class 5 except that it does not reduce the amount of query optimization for complex dynamic SQL queries.
- **9**: This class directs the optimizer to use all available optimization techniques. These include:
  - All available statistics
  - All query rewrite rules
  - All possibilities for join enumerations, including Cartesian products and unlimited composite inners
  - All access methods

  This class can greatly expand the number of possible access plans that are considered by the optimizer. You might use this class to find out whether more comprehensive optimization would generate a better access plan for very complex and very long-running queries that use large tables. Use Explain and performance measurements to verify that a better plan has actually been found.

8.4.4 Considerations for Call Level Interface and ODBC

The DB2 Call Level Interface (CLI) is a programming interface that your C and C++ applications can use to access DB2 databases. DB2 CLI is based on the Microsoft Open Database Connectivity Standard (ODBC) specification, and the X/Open and ISO Call Level Interface standards. Many ODBC applications can be used with DB2 without any modifications. Likewise, a CLI application is easily ported to other database servers.

DB2 CLI and ODBC provide a dynamic SQL application development environment. The SQL statements are issued through direct API calls. The DB2 optimizer prepares the SQL statements when the application runs. Therefore, the same advantages as dynamic embedded SQL programs are also true for DB2 CLI and ODBC programs. As we saw in the previous section, the advantages are:

- Current database statistics are used for each SQL statement.
- Database objects do not have to exist before run time.
- More flexible than static SQL statements.

Moreover, DB2 CLI and ODBC applications have the following advantages:

- Can store and retrieve sets of data.
- Can use scrollable and updatable cursors.
- Easy porting to other database platforms.

A drawback to using DB2 CLI and ODBC is that the dynamic preparation of SQL statements can result in slower query execution.

Improve performance of CLI/ODBC applications

Since the DB2 optimizer prepares the SQL statements in CLI/ODBC programs at run time like dynamic SQL programs, the following are considerations to improve performance:

- Execute RUNSTATS after making significant updates to tables or creating indexes
- Minimize preparation time for SQL statements

As we have already discussed, since the DB2 optimizer tries to find the best access plan for each SQL statement based on the current statistics information saved in the system catalog, refreshing statistics information using RUNSTATS will help the DB2 optimizer to determine the best access plan.

To minimize preparation time for SQL statements in CLI/ODBC programs, you should consider avoiding repeated PREPARE statement, and use the appropriate optimization level as discussed in the dynamic embedded SQL program section (see “Dynamic SQL” on page 452). In the following sections, we discuss how to avoid repeated PREPARE statement, and set optimization level in CLI/ODBC programs. We also introduce two methods to minimize preparation time for CLI/ODBC applications.

Avoid repeated PREPARE statements

When you need to execute multiple statements that differ only in the literal values they contain, you can use SQLExecDirect repeatedly for each statements; however, this approach is expensive since each statement is prepared one by one. To avoid preparing similar SQL statements repeatedly, you can use an SQLPrepare call instead of multiple SQLExecDirect. Your program should perform the following steps:

1. Call an SQLPrepare to prepare the SQL statement with parameter markers.
2. Issue an SQLBindParameter to bind a program variable to each parameter marker.

3. Issue an SQLExecute call to process the first SQL statement.
4. Repeat SQLBindParameter and SQLExecute as many times as required.

The ready-to-execute package prepared by SQLPrepare will be reused for each SQL statement.

Tune optimization level

As discussed in the dynamic embedded SQL section, if the database is in an environment such as Online Transaction Processing (OLTP), which typically has numerous simple SQL statements to process, set the optimization class to a lower value such as 1 or 2. To set the optimization level within the application, use SQLExecDirect to issue a SET CURRENT QUERY OPTIMIZATION statement. To set the same optimization level for all the CLI/ODBC applications on a client, use the UPDATE CLI CFG command from the client as in the following example:

```
UPDATE CLI CFG FOR SECTION database1 USING DB2OPTIMIZATION 2
```

This command sets the CLI/ODBC keyword DB2OPTIMIZATION=2 in the db2cli.ini file so that the DB2 optimizer will use the optimization level 2 to optimize SQL statements of all the CLI/ODBC applications accessing the database database1 from this client.

Use an optimized copy of catalog

Many applications written using ODBC or DB2 CLI make heavy use of the system catalog. Since the tables that make up the DB2 catalog contain many columns that are not required by the ODBC driver, ODBC/CLI applications can cause DB2 to retrieve much extraneous data when reading DB2 catalog data pages. Also, the ODBC driver often has to join results from multiple DB2 catalog tables to produce the output required by the ODBC driver's callable interfaces.

While this does not usually present a problem for databases with a small number of database objects (tables, views, synonyms and so on), it can lead to performance problems when using these applications with larger DB2 databases.

This performance degradation can be attributed to 2 main factors: the amount of information that has to be returned to the calling application and the length of time that locks are held on the catalog tables.

The db2ocat tool solves both problems by creating separate system catalog tables called the ODBC optimized catalog tables that has only the columns necessary to support ODBC/CLI operations.

The db2ocat tool is a 32-bit Windows program that can be used on Windows workstations running the DB2 Version 6.1 (or later) client. You can create ODBC optimized catalog tables in DB2 databases on any platform from this tool running on Windows workstations.

Using the db2ocat tool, you can identify a subset of tables and stored procedures that are needed for a particular application and create a ODBC optimize catalog that is used to access data about those tables. Figure 8-44 on page 462 shows the db2ocat tool GUI which is used to select tables that will be accessible through the ODBC optimized catalog:

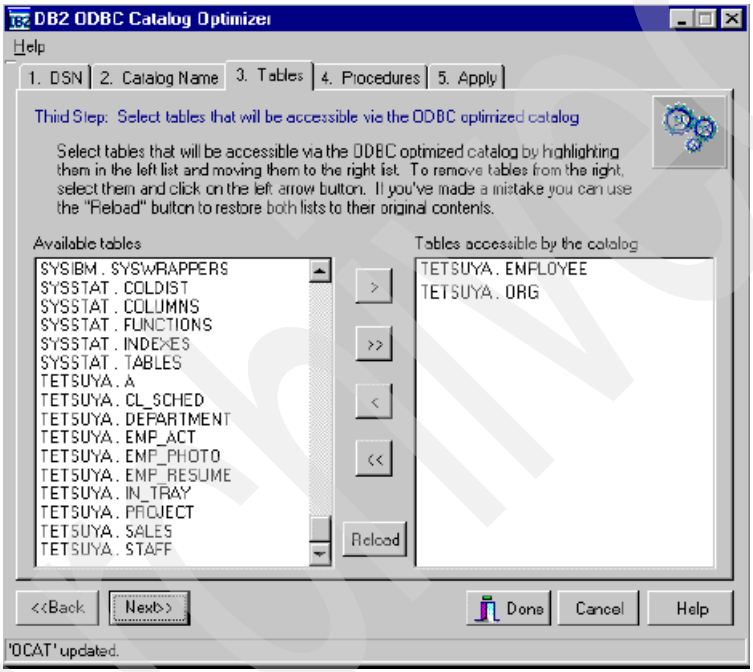

Figure 8-44 DB2OCAT Tool

An ODBC optimized catalog consists of ten tables with the specified schema name. If you specify OCAT as the schema name during the creation of the ODBC optimized catalog, the following tables will be created:

- OCAT.TABLES
- OCAT.COLUMNS
- OCAT.SPECIALCOLUMNS
- OCAT.TSTATISTICS
- OCAT.PRIMARYKEYS
- OCAT.FOREIGNKEYS
- OCAT.TABLEPRIVILEGES

- OCAT.COLUMNTABLES
- OCAT.PROCEDURES
- OCAT.PROCEDURESCOLUMNS

These tables contain only the rows representing database objects you selected and the columns required for ODBC/CLI operations. Moreover, the tables are pre-formatted for the maximum ODBC performance. By using the ODBC optimized catalog, the ODBC driver does not need to acquire locks on the real system catalog tables or perform join operations for results from multiple tables. Therefore, catalog query times and amount of data returned as a result of these queries are substantially reduced.

You can have multiple ODBC optimized catalogs for different clients. The ODBC optimized catalog is pointed to by the CLISCHEMA keyword. If the schema name of the ODBC optimized catalog is OCAT, then set CLISCHEMA=OCAT in db2cli.ini file on the client. You can directly edit the db2cli.ini file or execute the following command:

```
UPDATE CLI CFG FOR SECTION database1 USING CLISCHEMA OCAT
```

The contents in the ODBC optimized catalog is not replicated automatically from the system catalog. You must refresh the ODBC optimized catalog using the db2ocat tool when you perform something which changes the system catalog such as executing RUNSTATS or adding new columns (Figure 8-45).

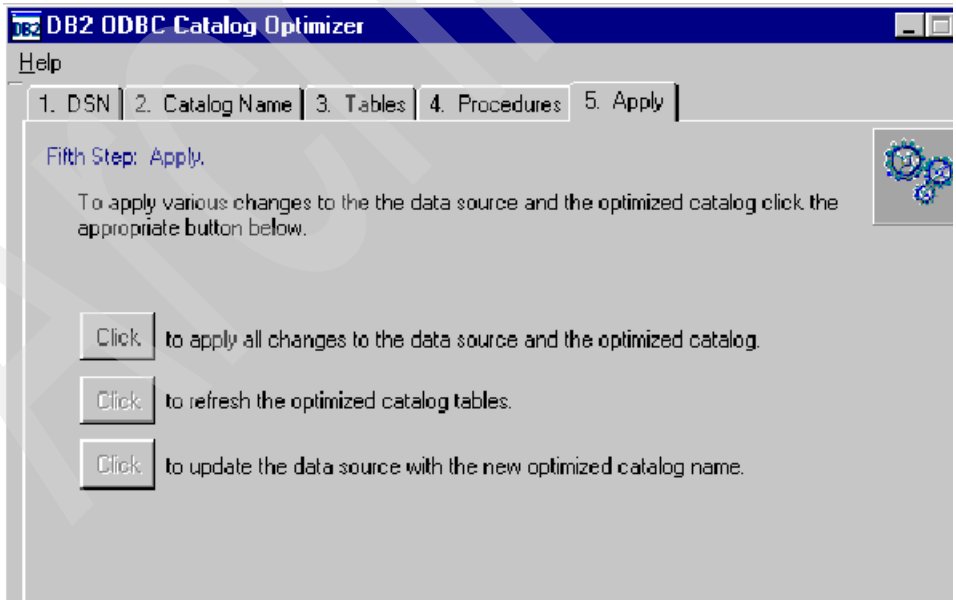

Figure 8-45 The db2ocat tool (refresh ODBC optimized catalog

The db2ocat tool is available at the following site:

ftp://ftp.software.ibm.com/ps/products/db2/tools/ in the file db2ocat.exe.

The readme file is available in the db2ocat.zip file at the same site.

Convert ODBC/CLI into static SQL

As ODBC/CLI applications are dynamic SQL applications, the most effective data access plan of each query is generated at program execution time. This process is expensive since the system catalog tables must be accessed for the resolution for the SQL statements and the statements are optimized. By using the db2ocat tool (see “Use an optimized copy of catalog” on page 461), the cost to access the system catalog can be reduced; however, ODBC/CLI applications and dynamic SQL applications can be still slower than static SQL applications whose SQL statements are ready-to-execute.

In this section, we introduce the method to convert ODBC/CLI applications into static SQL applications. The information of an executed ODBC/CLI application can be captured, and the executable form of statements are stored in the database as a package. Other ODBC/CLI applications can use it like static SQL applications without the preparation cost of the SQL statements.

ODBC/CLI applications run in the following three different modes:

- **Normal mode:** This is the default value and the traditional ODBC/CLI usage.
- **Capture mode:** This is the mode used by the database administrator who will run an ODBC/CLI application to capture its connection and statement attributes, SQL statements, and input as well as output SQLDAs. When a connection is terminated, the captured information is saved into an ASCII text file specified by STATICCAPFILE keyword in the db2cli.ini file. This file should be distributed to other clients as well as the application, and also the package should be created using the db2cap bind command, just as you would create a package using the bind command for a static SQL application.
- **Match mode:** This is the mode used by the end user to run ODBC/CLI applications that were pre-bound and distributed by the database administrator. When a connection is established, the captured information associated with the data source name will be retrieved from the local capture file specified by STATICCAPFILE keyword in the db2cli.ini file. If a matching SQL statement is found in the capture file, the corresponding static SQL statement will be executed. Otherwise, the SQL statement will still be executed as a dynamic SQL statement.

These modes are specified using the STATICMODE keyword of the db2cli.ini file as in the following example:

```
[SAMPLE]
STATICCAPFILE=C:\Program Files\IBM\SQLLIB\pkg1.txt
STATICPACKAGE=DB2INST1.ODBCPKG
STATICMODE=CAPTURE
DBALIAS=SAMPLE
```

This example specifies the capture mode. Captured information of the ODBC/CLI application accessed SAMPLE database is saved into the C:\Program Files\IBM\SQLLIB\pkg1.txt file. The STATICPACKAGE keyword is used to specify the package name to be later bound by the db2cap bind command.

You can directly edit the db2cli.ini file as shown above, use the Configuration Assistant, or use the UPDATE CLI CFG command as shown in the following example.

```
UPDATE CLI CFG FOR SECTION sample
USING STATICCAPFILE C:\Program Files\IBM\SQLLIB\pkg1.txt
STATICMODE CAPTURE
STATICPACKAGE DB2ADMIN.ODBCPKG
```

If necessary, you can edit the captured file to change the bind options such as QUALIFIER, OWNER, and FUNCPATH.

Then the **db2cap** bind command should be executed to create a package. The captured file and the database name must be specified as the following example:

```
db2cap bind C:\Program Files\IBM\SQLLIB\pkg1.txt -d sample
```

The created package name has a suffix number depending on its isolation level. The suffix for the package is one of the following:

- 0 = Uncommitted Read
- 1 = Cursor Stability
- 2 = Read Stability
- 3 = Repeatable Read

If the captured file has more than one statement and their isolation levels are different, multiple packages will be created with different suffixes.

You can have more than one captured file to create multiple packages in the same database by executing the db2cap bind command for each captured file. Be sure that the PACKAGE keyword of each captured file has a different value, since it is used to determine the package name.

Lastly, you should distribute both the captured file and the application to all client machines on which you intend to utilize the pre-bound package. On each client, the STATICMODE keyword of the db2cli.ini file should be set as MATCH, and the captured file should be specified using the STATICCAPFILE keyword.

```
[SAMPLE]
STATICCAPFILE=C:\Program Files\IBM\SQLLIB\pkg1.txt
STATICMODE=MATCH
DBALIAS=SAMPLE
```

9

Windows scripting

In this chapter we cover the various means available for creating scripts in Windows 2000. We describe in detail how each scripting language works, and we provide working examples, including specific DBA related scripts for managing DB2.

We cover the following topics:

- Description of Windows 2000 scripting
- Designing DBA scripts
- What DBA tasks are good candidates for scripts
- What scripting languages are available under Windows 2000
- What benefits are obtained by using a specific scripting language
- How to install support for the language
- Example for using a script
- Example of a DB2 script

9.1 Introduction

Traditional scripts on the Windows platform were written in DOS Batch files. Batch files can be used to launch the DB2 CLP (command line processor) and execute DB2 commands. Windows2000 scripts now can be written in most any language that a script interpreter support can be found for Windows 2000. Some of the most common languages are: WScript, VBScript, Jscript, Active Server Pages (ASP), PERL, JavaScript and REXX.

Scripts can also be written that make use of the DB2 Command line Interface. Scripts, regardless of the language they are written in, are interpreted languages. That means the script is run through a scripting engine which translated the script to lower level operating system operations. These operations can then connect to the database through compiled object that use native drivers to connect or through another abstract layer component object which leverages a connection through standard industry based specification libraries (ODBC).

Scripting is an old but still productive custom art of both the System Administrator and the Database Administrator (DBA). Scripting can be a very power tool. They can be used to automate tasks and simply tasks, with little or no manual intervention to execute. Today Database Administrators have four basic options when managing a database.

- They can use the DBA Tools that come usually come with the database
- The can use the traditional command line interface
- The can use some 3rd party SQL interface tool
- The can create their own scripts.

Database Administrators can leverage scripts to accomplish the simple to the difficult enterprise-wide level tasks under Windows 2000. They can schedule scripts to run at off-hours, so they won't affect the production database performance. The first three options listed above are fine for a single type of transaction. Use of creative scripts enable a DBA the ability to execute multiple transactions and operations, that can be scheduled to run any time of the day and provide a productive toolset to manage the database around the clock.

Windows offers several methods for writing scripts. Scripts can be written in any text editor such as Notepad, Wordpad, or with any ASCII based editor you like that runs on Windows 2000.

The scripts that have been written for this book illustrate how anyone with DBA-level permissions can use Database Objects through scripting.

9.1.1 Designing DBA scripts

Before you start writing scripts, or learning a language to write them in, a DBA needs to design what a script should do. Designing DBA scripts and the use of those scripts can provide a DBA with analysis information, which can influence a DBA in modifying existing configurations and planning for new ones

A DBA is faced with many challenges:

- Database design (both logical and physical)
- Space management (storage definition, physical layout, growth, and capacity planning)
- Creation and management of database objects
- Account management
- Security of database objects
- Performance and optimization
- Disaster and recovery
- Data integrity
- Data maintenance
- High availability
- Scalability
- Application development design considerations
- Integration with legacy systems
- Replication requirements
- General DB management and reporting

Manipulating the database through a SQL interface or through the command line interface is probably the easiest of the potential script interfaces for a DBA. DBA's of DB2 are familiar with the various DB2 commands and should have a fundamental understanding of SQL DDL and DML statements. There are other programmable means to connect and execute commands and SQL on DB2, within this chapter, we focus on the use of scripts for routine database administrative tasks. Scripts can be invoked by several means. We describe the various methods and provide illustrative examples. Later we cover other programming techniques beyond these scripts.

9.1.2 Choosing which DBA tasks to include in scripts

The first think to consider is your criteria for wanting to incorporate the task into a script.

- Do you plan to run the task unattended?
- D you want to run the task from a remote location?
- Do you want to run the task as many times as it takes to complete a job?
- Do you have more than one type of task to run?
- Do you want to run the task after hours?

If you answer “yes” to any of these questions, then scripting is a viable option to consider. Can any of the tasks you are considering be handled by the command line or through a SQL transaction? If so then you can easily accomplish your criteria through scripting.

The following listed DBA tasks which can be accomplished by scripts. We have included sample scripts for the listed tasks in Appendix A, “Advanced scripting” on page 497.

- **Design database:** This is for logical design and is a core function of a Database Administrator. A DBA is responsible for database designs and redesigns. The listed scripts are included in section 9.3.3, “Perl scripting” on page 478 or Appendix A, “Advanced scripting” Section , “Sample code” on page 503 which show how a DBA can create or destroy a database, create tables and review the data dictionary for redesign of the database structure (Schema).
  - List Data Dictionary (Tables and Columns Definitions)
  - Create New Database
  - Listing All DB2 Objects
  - Creation of Tables
  - Altering the Database
- **Space management:** This is for storage definition, physical layout, growth, and capacity planning. Space Management is an important aspect of a DBA job. Unfortunately it is usually ignored until it becomes a problem. It is important to know how to efficiently and effectively design the space required and the location or physical layout of where a database and it supporting objects will be created. This is important to minimize the risk associated with a crash, and the non functional requirements such as availability, scalability, reliability, performance, and capacity planning and maintenance management. The following lists scripts included in Appendix A, “Advanced scripting” Section , “Sample code” on page 503 provide you with some of the reporting detail you'll need.
  - List All Tables and Table Creation Date
  - List All Tables and Table Owners
  - List Table and TableSpace Sizes Definitions
- **Account management:** This is for access to database objects. A DBA is responsible for the management of who can access to the database and the level of privileges a user has permission to perform database operations against. The following scripts included in Appendix A, “Advanced scripting” on page 497 provide you an example of some of the tasks you may need to perform, as well as producing a useful status report.
  - Creating a new account
  - Removing a new account

  - Creating users from a file list
  - Listing all userids in db2 database

- **Performance and optimization**: This is always an ongoing task of a DBA. The DBA's first must gather statistical information about the objects operation in the database. Runstats is one of the basic operation that needs to be done. The sample script is included in 9.3.3, “Perl scripting” on page 478.
  - Runstats
- **Disaster recovery**: In the event of a disaster and the database get destroyed, it is important that the DBA has made backup copies of the database, and be able to restore the database back to the status that existed before it got destroyed. The following operations have been scripted in an example included in 9.3.3, “Perl scripting” on page 478.
  - Backup
  - Restore
- **General management and reporting:** There are many tasks a DBA is responsible for. These are some of the more routine task preformed practically every day illustrated in the following listed example scripts (see Appendix A, “Advanced scripting” on page 497. It include a Web-based interface form that enable the DBA to run reports interactive remotely
  - Starting the database
  - Stopping the database
  - Running SQL batch files
  - Running SQL from a command prompt
  - Querying DB2 application event log errors
  - Running any dynamic queries (reporting tool) through an interactive Web server form

9.2 Windows shell scripting (Wshell)

Windows shell scripting (Wshell) is the most basic form of scripting on Windows environments. Wshell scripting is based on the built-in command interpreter (cmd.exe) Wshell scripts allows you to use DB2 command-line tools in a more programmatic way.

You can find an interesting article about Windows shell scripting on:

http://www7b.boulder.ibm.com/dmdd/library/techarticle/0203yip/0203yip.html.

Checking and working with return code on DB2 Wshell scripts

One way to do return code checking on shell scripts is saving a checkpoint on each step done. You can also stop script execution when and error happens by

including the option **-s** on **db2** command call. Here is an example of creating and dropping a db2 table:

Example 9-1 Forwarding return code from sample.db2 script to runtask.bat

```
-- script SAMPLE.DB2 --
!ECHO 'CONNECTING' > SAMPLE.FLG@
CONNECT TO SAMPLE_B@

!ECHO 'CREATING' > SAMPLE.FLG@
CREATE TABLE SAMPLE (COL_1 VARCHAR(20))@

!ECHO 'DROPPING' > SAMPLE.FLG@
DROP TABLE SAMPLE@

!ECHO 'DISCONNECTING' > SAMPLE.FLG@
DISCONNECT SAMPLE_B@

!ECHO 'OK' > SAMPLE.FLG@
-- end script SAMPLE.DB2 --

-- script RUNTASK.BAT --
db2 -td@ -s -f SAMPLE.DB2 -z SAMPLE.LOG
FIND "OK" SAMPLE.FLG2 > NIL
IF %ERRORLEVEL% != 0 GOTO ERRHAND
ECHO SCRIPT OK
GOTO END
:ERRHAND
ECHO SCRIPT FAIL
:END
-- script RUNTASK.BAT --
```

Tools that you can use on Windows scripting

Here is a list of DB2 command line tools that you can use on scripting functions to control, execute, query information or compile on DB2.:

- db2batch — Benchmark tool
- db2bfd — Bind file description tool
- db2cap — CLI/ODBC Static Package Binding Tool
- db2ckrst — Check Incremental Restore Image Sequence
- db2cfexp — Connectivity configuration export tool
- db2cfimp — Connectivity configuration import tool
- db2cidmg — Remote database migration
- db2ckbkp — Check backup
- db2ckmig — Database pre-migration tool
- db2evtbl — Generate event monitor target table definitions
- db2cli — DB2 interactive CLI

- db2cmd — Open DB2 command window
- db2dclgn — Declaration generator
- db2exfmt — Explain table format tool
- db2expln — DB2 SQL explain tool
- db2flsn — Find log sequence number
- db2gncol — Update generated column values
- db2gov — DB2 governor
- db2govlg — DB2 governor log query
- db2icrt — Create instance
- db2idrop — Remove instance
- db2ilist — List instances
- db2imigr — Migrate instance
- db2inidb — Initialize a mirrored database
- db2inspf — Format inspect results
- db2iupdt — Update instances
- db2ldcfg — Configure LDAP environment
- db2licm — License management tool
- db2look — DB2 statistics and DDL extraction tool
- db2move — Database movement tool
- db2mscs — Configure Windows failover utility
- db2mtrk — Memory tracker
- db2nchg — Change database partition server configuration
- db2ncrt — Add database partition server to an instance
- db2ndrop — Drop database partition server from an instance
- db2perfc — Reset database performance values
- db2perfi — Performance counters registration utility
- db2perfr — Performance Monitor registration tool
- db2profc — DB2 SQLj profile customizer
- db2profp — DB2 SQLj profile printer
- db2rbind — Rebind all packages
- db2_recon_aid — Reconcile DATALINK file data
- db2relocatedb — Relocate database
- db2set — DB2 profile registry command
- db2setup — Install DB2
- db2sql92 — SQL92 compliant SQL statement processor
- db2start — Start DB2
- db2stop — Stop DB2
- db2support — Problem analysis and environment collection tool
- db2sync — Start DB2 synchronizer
- db2tbst — Get table space state
- db2trc — Trace
- db2uiddl — Prepare unique index conversion to V5 semantics
- db2undgp — Revoke execute privilege
- db2untag — Release container tag

Non-Wshell scripting options

If you want to develop shell scripts but think that default Wshell does not offer much control or options to develop a script, or you are planning to have cross-platform scripts, you can use cygwin. Cygwin is a UNIX environment developed by Red Hat, for Windows. It includes the bash and korn shell and other UNIX tools. You can find cygwin at http://www.cygwin.com.

9.2.1 DB2 CLP scripting

The db2 command starts the command line processor. The CLP is used to execute database utilities, SQL statements and online help. It offers a variety of command options, and can be started in:

- Interactive input mode, characterized by the `db2 => input prompt`
- Command mode, where each command must be prefixed by db2
- Batch mode, which uses the -f file input option.

On Windows 2000, **db2cmd** — Open DB2 Command Window opens the CLP-enabled DB2 window, and initializes the DB2 command line environment. Issuing this command is equivalent to clicking the DB2 Command Window icon.

QUIT stops the command line processor. TERMINATE also stops the command line processor, but removes the associated back-end process and frees any memory that is being used. TERMINATE is recommended if the database has been stopped, or if database configuration parameters have been changed. Existing connections should be reset before terminating the CLP.

The shell command (!), allows operating system commands to be executed from the interactive or the batch mode on the Windows operating system (i.e.!dir)

The DB2 UDB Command Reference contains the details information of the DB2 CLP commands and the command parameters.

Sample Script

The DB2 command can be executed under the DB2 Command Window directly or coded in a text file (see Example 9-2) then executed later.

Example 9-2 Create database

```
create database itsodb on c:/database
using codeset ITSO0889-1
territory US
with “ITSO San Jose”;
connect to itsodb;
list tablespaces show detail;
connect reset;
```

To execute the CLP script, under the command window, enter:

```
db2 -tvg CLP_script
```

9.3 Windows Script Host (WSH)

Microsoft's Windows Script Host is a language-independent scripting host for 32-bit windows operating systems. It provides the most powerful functionality of all the scripting methods discussed so far. Windows Scripting Host works seamless with all scriptable objects available to windows, allotting you to create complex, scripted applications. By providing extensive scripting capabilities combined with support for multiple scripting languages, WSH is quickly becoming the scripting method of choice.

Windows Script Host is controlled by two executable, CSCRIPT and WSCRIPT. CSCRIPT is the command-line host utility that is commonly used to run tasks in the background or in a command prompt. WSCRIPT is the graphical host utility commonly used to interact with the user. These two executables support many command-line parameters, as shown in Table 9-1.

Table 9-1 CSCRIPT and WSCRIPT command-line parameters

Parameter	Description
//B	Disables command prompt user input
//D	Enables Active Debugging
//E:engine	Uses the specific engine at script execution
//H:CSCRIPT	Sets CSCRIPT as the default execution host
//H:WSCRIPT	Sets WSCRIPT as the default execution host
//I	By default, enables command prompt user input
//JOB	Execute a WSC job
//LOGO	By default, displays logo at script execution
//NOLOGO	Suppresses logo at script execution
//U	For CSCRIPT only, specifies to use UNICODE for I/O operations

Parameter	Description
//S	Saves options on a per user basis
//T:seconds	Specifies the maximum time, in seconds, a script is allowed to run
//X	Executes the current script within the debugger
//?	Displays help content

9.3.1 VBScript scripting

VBScript is a scripting language based on the Visual Basic syntax. It does not offer all the functionality of Visual Basic but does provide to a set of basic functionality. VBSCRIPT is one of the Active Server Page languages, Jscript is the other. VBScript can be used in a few ways. It can be used form the command line, it can be used for client-side validation embedded within a Web page or it can be executed on a Web Server. VBScript can be used within the Windows Scripting Host. Also through Windows Script Host, you can use VBScript to call many VBA (Visual Basic for Applications) functions to automate Microsoft Office applications functions remotely. VBScript that is executed on a Web Server is commonly referred to as an .asp Active Server Pages (ASP) script.

Support for VBSCRIPT is part of the Windows 2000 install base. No additional software is necessary

VBScript sample script

There are many times when a DBA needs to reach each database users. The Windows Script Host does not include any methods to send messages to users or computers. Through Windows Script Host, you can call upon the NET.EXE utility or use automation to send messages. Following is an example VBscript sample which sends alerts to multiple users or computers:

```
On Error Resume Next
Set Shell = CreateObject("Wscript.Shell")
Dim Name(2)
Name(0) = "name1"
Name(1) = "name2"
MSG =  "message"
For X = 0 to UBound(Name)
SHELL.Run "Net Send " & Name(X) & " " & MSG, 0, False
Next
```

Here, Name is the array that holds the user or computer names to send messages to. The size of this array should be equal to the number of users or computers you want to send messages to. MSG is the message to send.

Following is a example of sending an E-mail Using Outlook Automation to send an E-mail using Outlook automation:

```
On Error Resume Next
RCP = "emailaddress"
SUB = "subject" "
MSG = "message"
Set Outlook = CreateObject("Outlook.Application")
Set MAPI = Outlook.GetNameSpace("MAPI")
Set NewMail = Outlook.Createltem(0)
NewMail.Subject = SUB
NewMail.Body = MSG
NewMail.Recipients.Add RCP
MAPI.Logon "profile", "password"
NewMail.Send
MAPI.Logoff
```

Here RCP stores the E-mail address to E-mail; SUB is the E-mail subject; MSG is the message to send; and profile and password are the logon credentials to send the E-mail.

9.3.2 JScript scripting

JScript is the Microsoft implementation of the ECMA 262 language specification (ECMAScript Edition 3). With only a few minor exceptions (to maintain backwards compatibility), JScript is a full implementation of the ECMA standard. This overview is intended to help you get started with JScript. JScript is an interpreted, object-based scripting language. Although it has fewer capabilities than full-fledged object-oriented languages like C++, JScript is more than sufficiently powerful for its intended purposes.JScript is not a cut-down version of another language (it is only distantly and indirectly related to Java, for example), nor is it a simplification of anything. It is, however, limited. You cannot write stand-alone applications in it, for example, and it has no built-in support for reading or writing files. Moreover, JScript scripts can run only in the presence of an interpreter or “host”, such as Active Server Pages (ASP), Internet Explorer, or Windows Script Host

Microsoft JScript functions perform actions; they can also return values. Sometimes these are the results of calculations or comparisons. Functions are also called “global methods”. Functions combine several operations under one name. This lets you streamline your code. You can write out a set of statements, name it, and then execute the entire set by calling it and passing to it any information it needs.

JScript offers a viable alternative to VBScript. Jscript is commonly used on the client-side for validation purposes on Web based data collection forms within Internet Explorer. Support for Jscript is part of the Windows 2000 install base and Internet Explorer. Example 9-3 is a sample Jscript.

Example 9-3 Sample JScript

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JScript">
function singOut()  {
var theMoment = new Date();
var theHour = theMoment.getHours();
var theMinute = theMoment.getMinutes();
var theDisplacement = (theMoment.getTimezoneOffset() / 60);
theHour -= theDisplacement;
if (theHour > 23)  {
theHour -= 24
}
document.write(theHour + " hours, " + theMinute + " minutes, Coordinated
Universal Time.");
window.setTimeout("singOut();", 60000);
}
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT>
singOut();
</SCRIPT>
</BODY>
</HTML>
```

9.3.3 Perl scripting

PERL is an old translated language. It runs on multiple operating systems. PERL is still the most prevalent of all Web based languages. PERL can also be run from the command line. There is practically every library you may need written for PERL. CPAN is a world wide support organization that provides archives of libraries for use at no cost http://www.cpan.org . PERL offers more possible integration means than other scripting languages. PERL scripts can integrate to DB2 through at least five different ways.

- Command line Interface
- DB2 Native Drivers
- ODBC Database Drivers
- DBI Library Interface

PERL for Windows 2000 can be downloaded from these sites:

http://www.activestate.com
http://www.perl.com

You can also download DB2 drivers for Perl from:

http://www-3.ibm.com/software/data/db2/per

Perl sample script

Create table is one of the first thing DBA will do after completing the physical database design. Following is a sample Perl script (Example 9-4) to create database:

Example 9-4 Create database script in Perl

```
# Windows 2000 Perl Script than creates a Database
# Frank Reddington Architect - National Architects COE

print "Enter the Database  you want to create \n";
$db = <STDIN>;
$dbcreate = `db2cmd -c db2 CREATE DATABASE $db`;
```

Creating individual tables can be time consuming. This script (Example 9-5) enables a DBA to create multiple tables from table definitions created in input files. Like all scripts they can be scheduled to run any time

Example 9-5 Create table script in Perl

```
# Windows 2000 Perl Script than creates tables from an input file
# Frank Reddington Architect - National Architects COE


print "Enter the DDL File  to read from (i.e. c:\\sqllib\\createtables.sql)\n";
$sqlinfile = <STDIN>;
chomp($sqlinfile);
$tabscreate =`db2cmd -c db2 -f $sqlinfile`;

#SAMPLE INPUT FILE
#connect to SAMPLE
#CREATE TABLE TAB1 (c1 int, c2 varchar(20), c3 varchar(20), c4 int)
#CREATE TABLE TAB2 (c1 int, c2 varchar(20), c3 varchar(20), c4 int)
#CREATE TABLE TAB3 (c1 int, c2 varchar(20), c3 varchar(20), c4 int)
```

The RUNSTATS utility collects statistical information for DB2 tables, table spaces, partitions, indexes and columns. It can place this information into DB2 Catalog tables or simply produce a report of the statistical information. The statistics in these tables are used for two primary reasons: to provide organizational

information for DBAs and to be used as input to the DB2 optimizer during the BIND process to determine optimal access paths for SQL queries The statistical information can also be queried using SQL. Example 9-6 is a sample Perl script performs Runstats function. By replacing the input file content with other DB2 command, this code can run other DB2 utility such as Reorg.

Example 9-6 Runstats script in Perl

```
# Running Statistics on Tables
# Frank Reddington National Architects Center of Excellence
# Example Use: runstats.pl c:\sqllib\stat_tabs.txt

$statfile = @ARGV[0];
chomp($statfile);
$runstats = "db2cmd -i -c db2 -f $statfile";
#print "$runstats \n";
$execute = `$runstats`;
print $execute;

Sample Input File   (i.e. stat_tabs.txt)

connect to sample
runstats on table admindb.sales
runstats on table admindb.org
runstats on table admindb.staff
```

Another daily activity a DBA will do is database backup. Following (Example 9-7) is sample Perl script to backup database. The restore command are also included but comment out. To perform the database restore, comment out the backup statement and uncomment the restore statement.

Example 9-7 Database backup and restore script in Perl

```
# Backup and Restore to disk utility
# Frank Reddington National Architects Center of Excellence

print "Would you like to perform a BACKUP or a RESTORE ?  ";
$operation = <STDIN>;
print "please enter the name of the database  ";
$dbname = <STDIN>;

print "please enter the path and directory (i.e. C:\\dbbackup) ";
$dirpath = <STDIN>;

print "please enter your authorized user name  ";
$username = <STDIN>;

print "please enter your authorized password  ";
$password = <STDIN>;
```

```
$operation =~tr/[a-z]/[A-Z]/;

chomp($operation);
chomp($dbname);
chomp($dirpath);
chomp($username);
chomp($password);

$backup = "db2cmd db2 $operation database $dbname USER $username USING
$password to $dirpath";
#$restore = "db2cmd db2 $operation database $dbname USER $username USING
$password from $dirpath";

#print "Backup command = $backup \n";
#print "Restore command = $restore \n";

if ($operation = "BACKUP") {
$execute = `$backup`;

                              }

if ($operation = "RESTORE") {
$execute = `$restore`;
                              }
```

9.3.4 Object REXX

Object REXX is IBM's object-oriented implementation of the REXX programming language. It has evolved from IBM's *Classic* REXX procedural programming language of the late 80s and early 90s. The REXX Product family runs in both workstation and host environments from Windows XP to OS/390. Object REXX is currently available on Linux (Intel and zSeries), OS/2, OS/390, UNIX (AIX), and Windows (ME/98/NT/2K). Object REXX support both the *Classic* and "*Object*" dialects on these platforms.

Object REXX includes support for:

- Classes, objects, and methods
- Messaging and polymorphism
- Inheritance and multiple inheritance

Object REXX is an attractive scripting language for automating database administration tasks for several reasons. First, the REXX language is easy to learn allowing database administrators to focus on getting the task at hand accomplished. Second, Object REXX is backward compatible with previous versions of REXX, protecting your investment in code written in the *Classic*

REXX dialects. Third, Object REXX is an interpretive language that is itself independent of any specific operating system. Fourth, DB2 provides several REXX Application Programming Interfaces (API) that can be called directly from the language.

Example 9-8 Classic REXX (HelloWorld.rex)

```
/* This is a sample REXX script. */
say 'Hello World!'
```

Object REXX for Windows is a Microsoft Windows-compatible REXX interpreter. It supports Windows Script Host, enabling Object REXX to be an ActiveX script engine and to be embedded in HTML and XML. Scripts written in Object REXX can be integrated into the Windows Scripting Host and re-used by other WSH supported languages such as JScript, VBScript, and Perl. Object REXX includes OLE (ActiveX) automation.

Example 9-9 Object REXX (HelloWorld.rex)

```
/* This is a sample REXX script. */
.output~lineout('Hello World')
```

Object REXX Editions

The latest version of IBM Object REXX for Windows is v2.1.1. Object REXX is available in two versions Interpreter and Development. The Developer Edition provides a means of compiling the REXX source into a tokenized REXX that can be invoked by a version of the REXX interpreter that can be freely distributed.

What's new: IBM Object REXX v2.1.1 provides support for running the REXX interpreter (rxapi.exe) as a Windows service. In previous version, the REXX interpreter (rxapi.exe) runs as a WIN32 background process. Now with v2.1.1 it can be registered as a Windows service and managed by the Services MMC.

Interpreter Edition

The Interpreter Edition provides the basic REXX language interpreter. Object REXX programs can be developed using any text editor. Each Object REXX program requires a licensed copy of the Object REXX interpreter be installed in order to runs Object REXX programs on the system. It includes a wide range of REXX utilities, APIs to other existing applications, and a powerful trace facility.

Development Edition

The Object REXX Development Edition includes all of the capabilities of the Interpreter Edition. It includes the Object REXX workbench which provides several tools to help you develop Object REXX scripts. The primary advantage of this edition is the Object REXX Tokenizer.

The Object REXX Tokenizer can be used to tokenize Object REXX programs in an internal binary format that is unreadable and faster as it has already passed through the REXX interpreters syntax checker. This edition also includes a tokenized runtime interpreter. This lightweight version of the Object REXX interpreter can be packaged and distributed freely with your REXX scripts.

Installing Object REXX

The Windows Installer simplifies the installation process of Object REXX. During the installation process a file type of REXXScript is created and associated with the Object REXX Workbench. Doing this allows you to invoke the Object REXX Workbench by simply typing the scripts name or double-clicking the file.

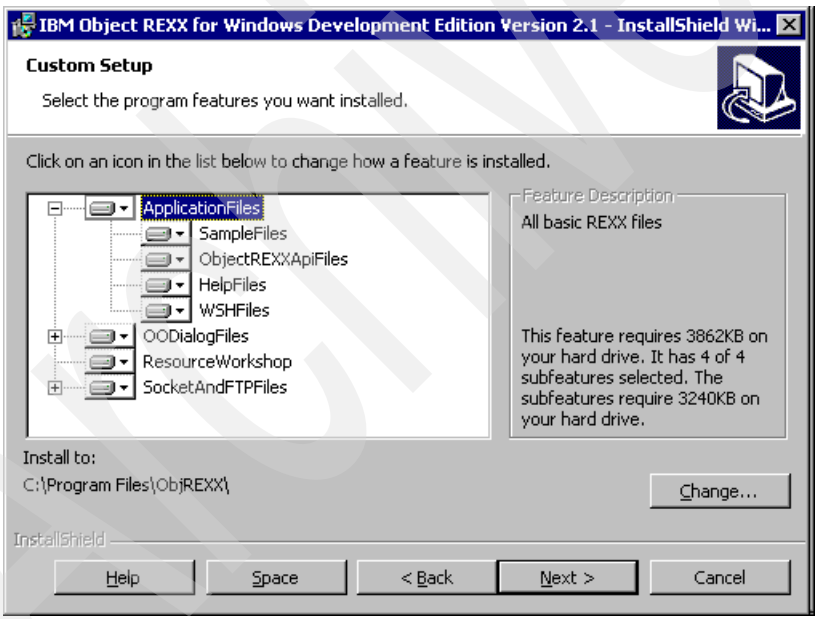

Figure 9-1 Object REXX for Windows, Custom Setup

Associating Object REXX Interpreter

The installation process creates a file type of REXXScript and associates the extension REX with the Object REXX Workbench. Doing this might be desirable in your development environment, but you may want to change this on your test, quality assurance, and production server so that by default the Object REXX interpreter is invoked.

In the example below the associate (assoc.exe) program shows that the .rex extension is associated with the file type REXXScript. We also see that the file type of REXXScript is opened by default by the Object REXX Workbench (orxwb.exe).

```
C:\>assoc .rex
.rex=REXXScript

C:\>assoc REXXScript
REXXScript=Object REXX - Script

C:\>ftype REXXScript
REXXScript="C:\Program Files\ObjREXX\orxwb.exe" /s "%1" %*
```

You may want to change the file type so that instead of running the Object REXX Workbench (orxwb.exe), the REXX interpreter (rexx.exe) is invoked. To do this simply use the file type (ftype.exe) utility as follows:

```
C:\>ftype REXXScript
REXXScript="C:\Program Files\ObjREXX\rexx.exe" "%1" %*
```

You may also want to add the Object REXX extension .rex to the Windows Path Extension environment variable. This will prevent you from having to invoke the REXX program with the .rex extension. For example, without the .rex extension as part of the Windows path extension environment variable, REXX programs must be typed as HelloWorld.rex on the command line. After adding the .rex file extension to the Windows path extension environment variable, the program can simply be executed by typing HelloWorld without the extension .rex. You can do this by selecting **My Computer —> Control Panel —> System —> Advanced —> Environment Variables** or you can also set this at the command prompt by typing:

```
c:\>SET PATHEXT=%PATHEXT%;.REX
```

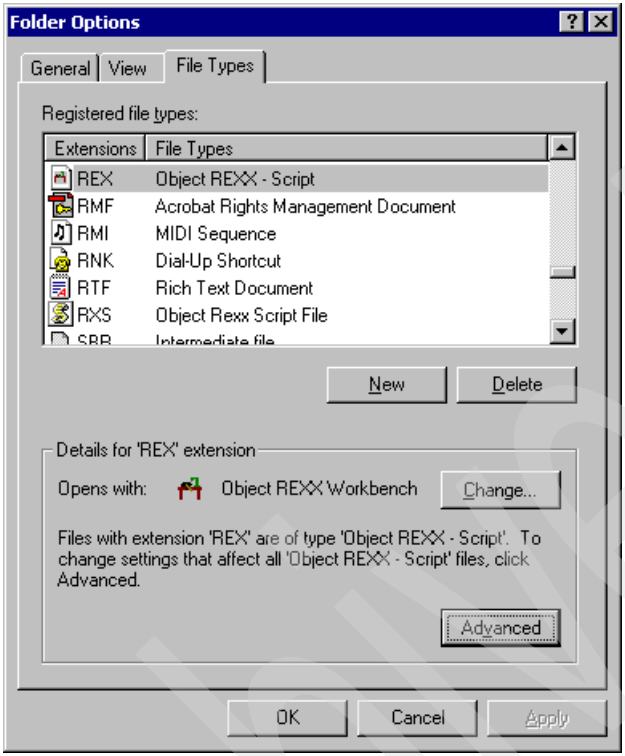

Figure 9-2 Object REXX File Type Association

Although this is not required to run the Object REXX program, it can be done to simply remove the requirement for having to type REXX before every REXX program. This is accomplished by using a couple of Windows utilities called file type (ftype.exe) and associate (assoc.exe).

Registering the Object REXX Interpreter service

The Object REXX interpreter (rxapi.exe) normally runs in as a WIN32 background process. Now with Object REXX v2.1.1 it can optionally be installed as a Windows service. This can be done by running the RXAPI.EXE program and passing one of three optional parameters. The syntax is as follows:

```
RXAPI.EXE [ /i /u /v ]
```

where:

- /i : Install and register RXAPI.EXE as service
- /u : Uninstall and deregister RXAPI.EXE as service
- /v: Version number and whether it is registered as service

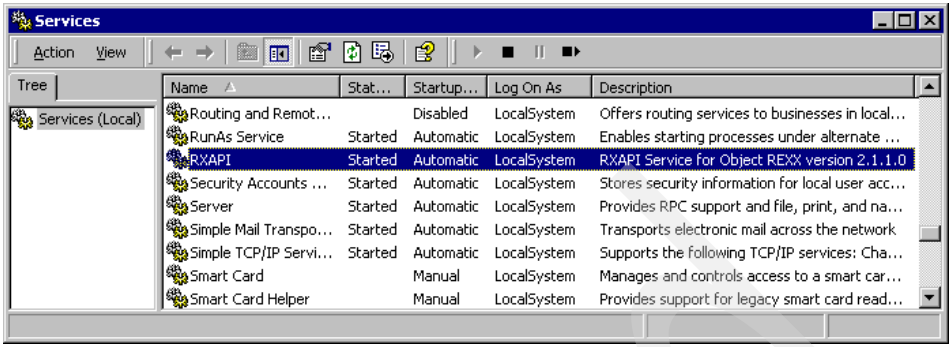

Figure 9-3 Object REXX Interpreter Service (RXAPI)

DB2 Scripting with Object REXX

IBM Object REXX turns out to be the only scripting language that is actually supported by DB2 Universal Database. This means that DB2 UDB provides a REXX Application Programming Interface (API) that can be called directly from the language. Support for the DB2 UDB REXX APIs has been stabilized an no further enhancements to the API are planned at this time. Having said that, REXX is still a very attractive alternative, primarily because even if support for the REXX API is removed from the product, it can still access the DB2 Command Line Interfaces. Additionally, a Object REXX interpreter is available on other platforms that run DB2 UDB such as Linux for Intel, Linux for zSeries, and AIX.So until IBM publishes a Java Native Interface or announces support for another scripting language, REXX makes a lot of since.

The DB2 UDB REXX APIs include:

- SQLEXEC: The SQLEXEC API provides an interface into the SQL language
- SQLDBS: The SQLDBS API provides an interface into the DB2 Administrative APIs
- SQLDB2: The SQLDB2 API provides an interface into the DB2 Commands

The first thing your REXX scripts should do is to make it a common practice to register the DB2 UDB REXX API should always be registered.

Example 9-10 Initialize DB2 REXX APIs

```
/*
// initialize SQLEXEC interface
*/
if RxFuncQuery('SQLEXEC') \= 0 then
   ok = RxFuncAdd('SQLEXEC','SQLEXEC','SQLEXEC')
/*
// initialize SQLDBS interface
*/
```

```
if RxFuncQuery('SQLDBS') \= 0 then
   ok = RxFuncAdd('SQLDBS','SQLDBS','SQLDBS')
/*
// initialize db2cmd interface
*/
if RxFuncQuery('SQLDB2') \= 0 then
   ok = RxFuncAdd('SQLDB2','DB2AR','SQLDB2')
```

Sample DB2 script using Object REXX

Scripting with Object REXX can be done in either the *Classic* or *Object* dialects. For most database administrators, the traditional REXX syntax gets the job done as database maintenance routines are procedural in nature. It is still possible within procedural programming techniques to re-use code written in Object REXX simply by creating common routines and placing these routines in dynamic link libraries. However, it is possible to create a database *class* with Object REXX and invoke a activate, deactivate, backup, or restore methods.

The following example shows an Object REXX routine that received the database name and target path as parameters. The routine calls the DB2 Command Interface (SQLDB2) API to invoke a database backup. The complete source code can be found in Appendix B, "Sample REXX programs" on page 519.

Example 9-11 Object REXX routine RxDb2BackupDatabase

```
/*--------------------------------------------------------------------------*/
::ROUTINE RxDb2BackupDatabase PUBLIC
/*--------------------------------------------------------------------------*/
use arg database, target
ok = CHAROUT(,.program.name'.RxDb2BackupDatabase('database')...')
call SQLDB2 'backup database 'database' to 'target
if RxSqlCodeCheck(SQLCA.SQLCODE,SQLMSG) \= 1 then
   return 0
return 1
```

Sample DB2 script using Object REXX and WSH

The Object REXX interpreter integrates into the Windows Scripting Host environment just as JScript and VBScript, however the Object REXX engine is not included as part of Microsoft's WSH distribution. In order to use Object REXX you must license the Object REXX interpreter.

REXX scripts that are written outside of a Windows script file (WSF) do not run under the control of the WSH and can be placed in a file with any file extension, although '"REX" is the most common.

Scripts written in Object REXX that are embedded within a Windows script file (WSF) must be wrapped within a begin script and end script tags so that the WSH can identify which scripting engine to submit the script to.

Example 9-12 Object REXX embedded within Windows Scripting Host

```
...
<script language="Object REXX">
...
     /*
     // initialize db2cmd interface
     */
     if RxFuncQuery('SQLDB2') \= 0 then
        ok = RxFuncAdd('SQLDB2','DB2AR','SQLDB2')
...
</script>
```

In the above example, Example 9-12, we see how the Object REXX code is wrapped within a begin <script language=”Object REXX”> tag and an end script </script> tag. Everything in between these tags is good old fashion REXX code.

In the next example we can see how the Windows Scripting Host support for XML makes it easy to generate command line parameter help. In this example the Object REXX script was invoked using the Command Script (cscript.exe) host.

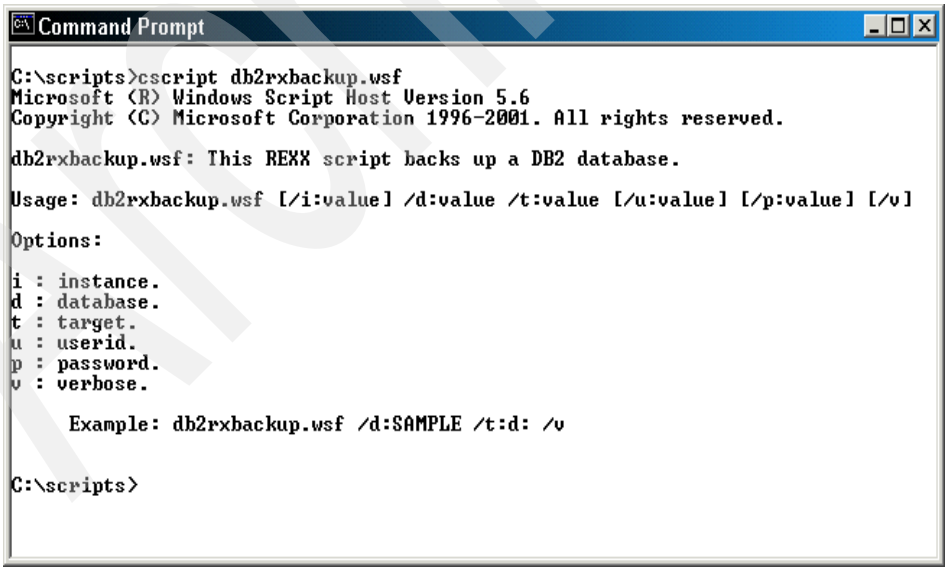

Figure 9-4 Object REXX with WSH (cscript .exe)

In the next example we can see the exact same Object REXX script invoked using the Windows Script (wscript.exe) host. The complete source code for this script can be found in Appendix B, “Sample REXX programs” on page 519.

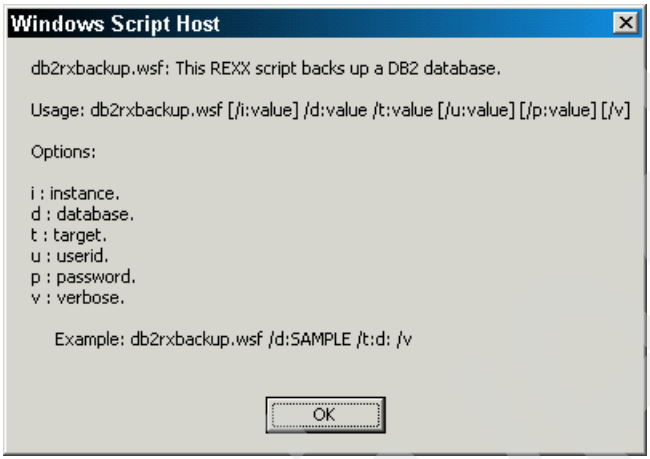

Figure 9-5 Object REXX with WSH (wscript.exe)

9.4 Scripting with DB2’s WMI providers

DB2 is integrated with Windows to obtain the most functionality from both environments. WMI offers an interface to do many specialized functions and allows the user to query variety of informations about the environment and the machine through an structured query language specialized for WMI (WQL).

In this section we explain more about WMI structure and show some functionality to get system information related to DB2.

9.4.1 Windows Management Instrumentation (WMI)

As enterprises grow larger they become more difficult to manage. Web-Based Enterprise Management (WBEM) is an initiative to provide an environment dependent solution to manage data and devices. WBEM was developed by the Desktop Management Task Force (DMTF), a collective organization consisting of IBM, Microsoft, Compaq, and other large corporations. Window's Management Instrumentation (WMI) is Microsoft's Windows implementation of the WBEM initiative.

WMI provides scripters and developers with a standardized method to monitor and rampage local and remote resources. It comes included with the Windows 2000 operating system. WMI provides a standard, scriptable interface to various

resources. The devices and applications controlled by WMI are known as managed objects. Managed objects can be anything from hardware, such as a hub or motherboard to software, such as the operating system or an application

The WMI Process

The executable that provides all the functionality of WMI is called WINMGMT.EXE. WINMGMT.EXE runs as a service under Windows 2000. When a script or application (known as a consumer) issues calls to the WMI name space, the executable awakes and passes these calls to the CIM Object Manager (CIMOM) The CIMOM is the entrance to the WMI infrastructure. It allows for the initial object creation and provides a uniform method to access managed objects. When CIMOM receives a request to control a managed object, it first checks the CIMOM object repository.

The CIMOM object repository is a storage area for the Common Information Model (CIM). The CIM contains the WMI object models and a description of all the available managed objects, called the management schema. This repository is full of all the different access methods and properties of manageable objects, known as static management data. If the information requested carrot be found in the repository, the repository passes the request down to the object provider.

A provider is the interface between the device to be managed and the CIMOM. The provider collects the information from a device and makes it available to the CIMOM. This information is known as dynamic management data. Developers create providers when the CIM does not contain methods to access a managed resource. Several providers come packaged with WMI:

- Active Directory provider
- Event Log provider
- Performance Counter provider
- Registry provider
- SNMP provider
- View provider
- WDM provider
- Win32 provider
- Windows Installer provider

Once the provider has completed processing the request, it sends all results back to the originating script or application.

Scripting WMI

The process of connecting to the WMI object model is similar to connecting to the WSH object model. To gain access to an object, you use the GetObject function and set it to a variable. This is called instantiating an object, as in the following example:

```
Set variable = GetObject("winmgmts:\\computer\rootlnamespace").ExecQuery ( WQL)
WMI Query Language (WQL).
```

This language, similar to SQL (Structured Query Language), allows you to query WMI information. The basic syntax for a WQL statement is as follows:

```
.ExecQuery("select propmeth from class")
```

ExecQuery runs the WQL statement, which is stored in quotes and surrounded by parenthesis. Propmeth specifies the property or method to retrieve from the specified class. Classes are organized containers for properties and methods of a manageable device. For example, the Win32_TapeDrive class contains all the properties and methods to manage tape drives. In addition to the ExecQuery, you can use the ExecNotificationQuery to perform WQL queries. The ExecNotificationQuery method is used to detect when instances of a class are modified. This means that you can poll for events. Combined with WQL you can use this method to monitor the event log, CPU, memory, and more based on a specific interval.

Microsoft creates software development kits (SDKs) to assist third-party application developers in creating Windows applications. The WMI SDK includes the core WMI installation, documentation, utilities, and examples. You can obtain the WMI SDK for free from http://msdn.microsoft.com

VBScript WMI sample script

Following is an sample VBScipt and Windows Host Shell using WMI. This script query DB2 errors in application event log.

Example 9-13 VBScript using WMI

```
'Example Use: DB2_Application_Log_Information
'Returns a message box for error events of DB2 found in application event log
'Frank Reddington National Architect Center of Excellence

Option Explicit
Dim refWMI
Dim strQuery
Dim colEvents
Dim refEvent
Dim FoundFlag

Set refWMI = GetObject("winMgmts:")
strQuery = "SELECT * FROM Win32_NTLogEvent WHERE Logfile='Application' and
Type='error'"
Set colEvents = refWMI.ExecQuery(strQuery)

FoundFlag = 0
```

```
For Each refEvent in colEvents
if refEvent.SourceName = "DB2" Then
WScript.Echo refEvent.SourceName & " - " & refEvent.Message & " Application Log
Record # " & refEvent.RecordNumber
FoundFlag = 1
end if
Next
If FoundFlag = 0 Then
WScript.Echo "No DB2 Error found in Event Log"
End If

Set colEvents = nothing
Set refWMI = nothing
WScript.Quit
```

9.5 Scripting with ADSI

Active Directory Services Interfaces (ADSI), previously OLE Directory Services, is Microsoft's of a directory instance that organizes an enterprise into a tree-like structure. A directory service provides a standard consistent method to manage and locate network resources.

ADSI comes packaged with Windows 2000 Server and is available as a free, separate download from Microsoft for Windows/NT.

The DBA who also maybe acting as a the System Administrator may be responsible to access the operating system for tasks such as creating a user or running a system level operations in addition to a Database function (i.e. backup).

The following (Example 9-14) shows an example of creating a user account

Select **Start—>Run** and enter "cscript newuser.vbs"

Example 9-14 Creating a user account (newuser.vbs)

```
On Error Resume Next Set DomObj = GetObject("WinNT://Domain")
Set User - DomObj.Create("User", "Name")
User.SetPassword("pswd")
User.FullName = "fullname"
User.HomeDirectory = "homedir"
User.Profile - "profiledir"
User.LoginScript = "script"
User.Description - "describe"
User.SetInfo
```

Here, domain is the name of the domain; name is the name of the user account to create; pswd is the password to assign to the new account; fullname is the users full name; homedir is the path of the user's home directory; profiledir is the path of the user's profile; script is the name of the logon script; and describe is the user description.

To add a user account to a group using ADSI, proceed as Example 9-15). Here, scriJJUnle is the full path and file name of a script file that contains the following:

Example 9-15 Adding a user account to a group

```
On Error Resume Next
Set Group ~ GetObject("WinNT://Gdomajn/groupname.group")
Group.Add "WinNT://UDomain/useraccount.User"
```

9.6 Scheduling and managing scripts

It is important for a DBA to be aware of all enterprise database activity. Deciding when to run your scripts is essential in order to minimize the impact they may have on the performance and availability of the database. As you know that you can't please everyone every time. Some interruption may affect another use of the database. And there are ways to mitigate the risks. Selecting On-line backup is one way, when considering availability. When it comes to performance, there are many factors. One of the most common factor no matter how well crafted and optimized your script is deciding what time is best to run the script with the least amount of user performance impacted.

Windows scheduling

Windows systems offers two tools to run scripts: Scheduled Tasks and the AT command.

Windows 2000 has a job scheduling feature called Scheduled Tasks (Figure 9-6). You can reach it clicking **Start Menu—>Programs—>Accessories—>System Tools—>Scheduled Tasks**, or opening Control Panel and then double clicking **Scheduled Tasks**.

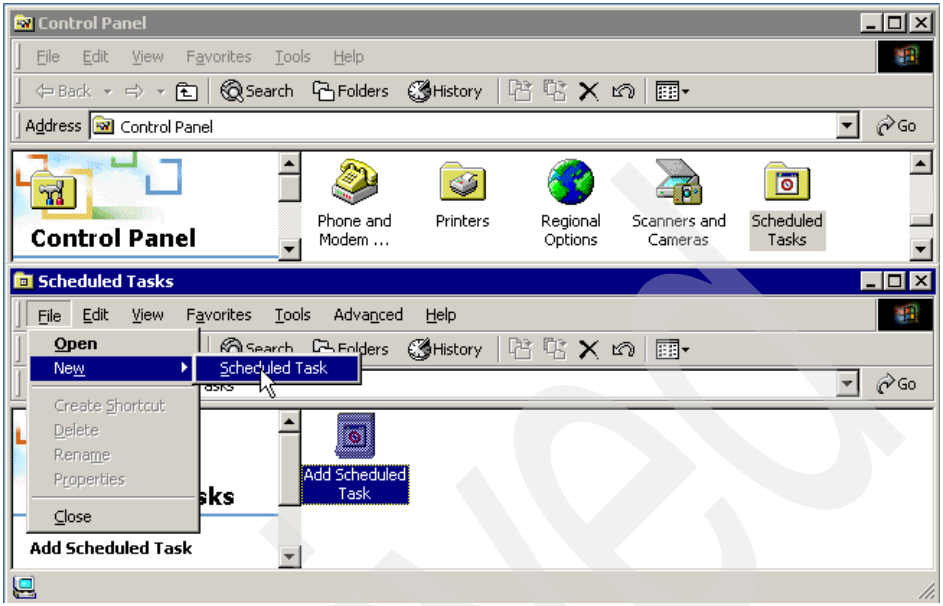

Figure 9-6 Creating a new scheduled task

The other Windows tool to schedule tasks is the **AT** command. **AT** command schedules commands and programs to run on a computer at a specified time and date. The Schedule service must be running to use the AT command.

```
AT [\\computername] [ [id] [/DELETE] | /DELETE [/YES]]
AT [\\computername] time [/INTERACTIVE] [ /EVERY:date[,...] | /NEXT:date[,...]]
   "command"
   \\computername     Specifies a remote computer. Commands are scheduled on
   the local computer if this parameter is omitted.
   id                Is an identification number assigned to a scheduled
   command.
   /delete           Cancels a scheduled command. If id is omitted, all the
   scheduled commands on the computer are canceled.
   /yes              Used with cancel all jobs command when no further
   confirmation is desired.
   time              Specifies the time when command is to run.
   /interactive      Allows the job to interact with the desktop of the user
   who is logged on at the time the job runs.
   /every:date[,...] Runs the command on each specified day(s) of the week or
   month. If date is omitted, the current day of the month is assumed.
   /next:date[,...]  Runs the specified command on the next occurrence of the
   day (for example, next Thursday).  If date is omitted, the current day of
   the month is assumed.
   "command"         Is the Windows NT command, or batch program to be run.
```

DB2 scheduling

DB2 provides a convenient GUI tool, Task Center, for scheduling and managing a job or a sequence of jobs. You can start Task Center by clicking the **Tools** menu on any DB2 GUI tool or by **Start Menu—>Programs—>IBM DB2—>General Administration Tools—>Task Center** (Figure 9-7).

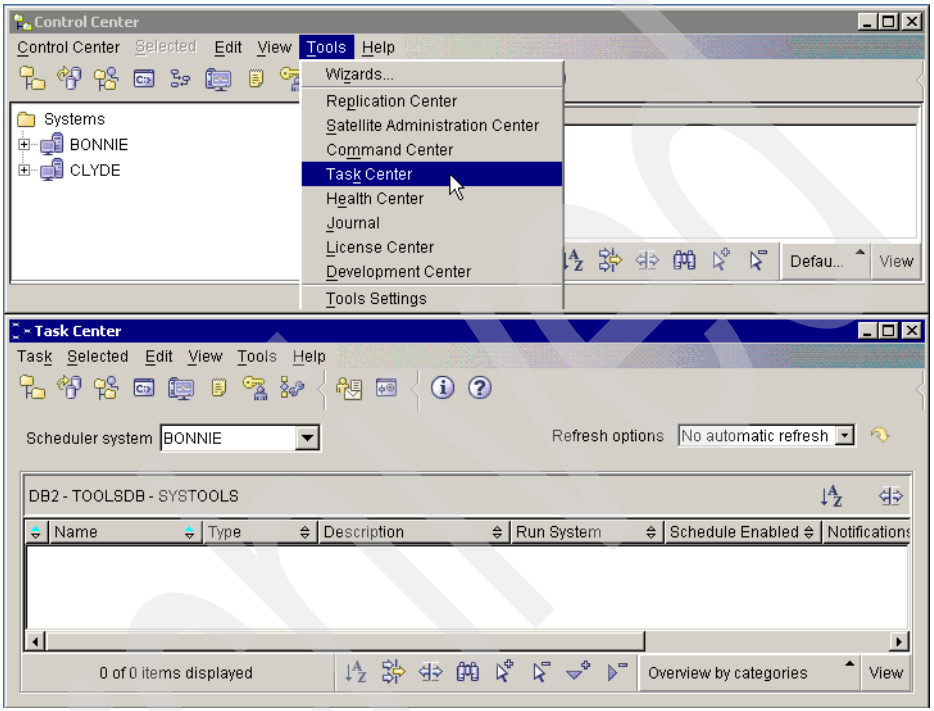

Figure 9-7 Running Task Center

To run Task Manager, you need to create a local tools catalog on a database or create a new database to hold the catalog tables on the server where you want to define scripts to be run or create a dedicated server (recommended) to manage and run scripts.

The command CREATE TOOLS CATALOG allows you to create the tools catalog. The syntax of CREATE TOOLS CATALOG is:

```
CREATE TOOLS CATALOG catalog-name {CREATE NEW DATABASE database-name |USE
EXISTING [TABLESPACE tblspace-name IN] DATABASE database-name}[FORCE] [KEEP
INACTIVE]
```

Example:

```
CREATE TOOLS CATALOG TOOLSCT CREATE NEW DATABASE TOOLSDB
```

The catalog information is written on the administrative instance. You can get it by running the command **GET ADMIN CFG**. The tools catalog configurations can be identified by the initial TOOLSCAT name. Example:

Example 9-16 Sample output of get admin cfg including Tools Catalog Information

```
Admin Server Configuration

Authentication Type DAS                 (AUTHENTICATION) = SERVER_ENCRYPT

DAS Administration Authority Group Name  (DASADM_GROUP) =

DAS Discovery Mode                            (DISCOVER) = SEARCH
Name of the DB2 Server System                (DB2SYSTEM) = BONNIE

Java Development Kit Installation Path DAS   (JDK_PATH) = C:\Program Files\IBM\
SQLLIB\\java\jdk\

DAS Code Page                              (DAS_CODEPAGE) = 0
DAS Territory                             (DAS_TERRITORY) = 0

Location of Contact List                   (CONTACT_HOST) =
Execute Expired Tasks                     (EXEC_EXP_TASK) = NO
Scheduler Mode                             (SCHED_ENABLE) = ON
SMTP Server                                  (SMTP_SERVER) = udbred05
Tools Catalog Database                       (TOOLSCAT_DB) = TOOLSDB
Tools Catalog Database Instance            (TOOLSCAT_INST) = DB2
Tools Catalog Database Schema             (TOOLSCAT_SCHEMA) = SYSTOOLS
```

A

Advanced scripting

In this appendix we provide sample coding for DBAs or system administrators to code scripts in the COM environment. We cover these areas:

- COM, ADO, and ASP concepts
- Sample scripts for database management
- Sample scripts for user account management

Leveraging COM

Regardless of the scripting language used, one of the powerful features of writing scripts under Windows 2000, is the ability to leverage existing components (COM objects). Administrators and developers can use any of the supporting languages and use them to leverage any Component Object Model object that will run under the Windows 2000 Operating System. That means that literally any functions from any of these objects classes may be publicly available to be used in these scripts without having to write an executable program.

Active Data Object (ADO)

This example below illustrates how scripts can use an standard Windows 2000 installed COM object, i.e.Active Data Object (ADO), which is Microsoft's Universal data access object (before .NET). This object gives the application access to the underlying data services layers that provide transaction access to compliant data sources such as commercial databases like DB2, excel spreadsheets and other sources. These connections can be made through dynamic OLE DB provider connections strings (if the data source version offers an OLE DB Provider, or use the standard OLE DB Provider ODBC connection

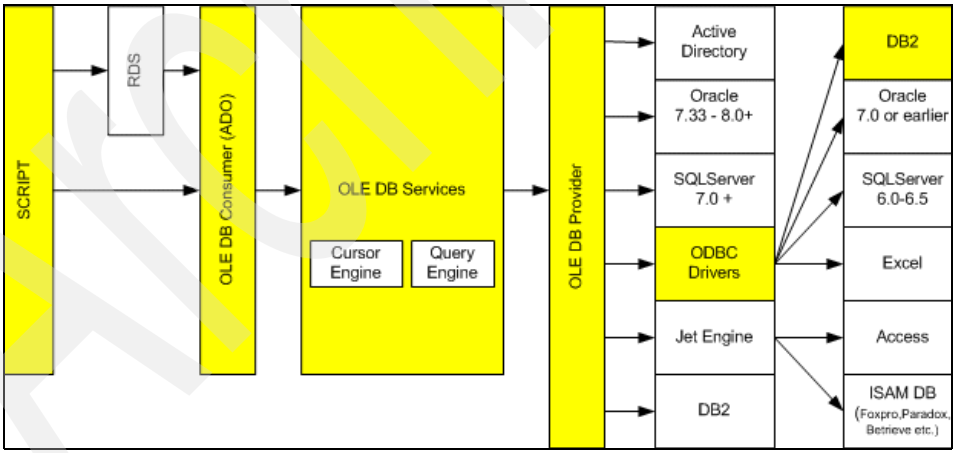

Figure 9-8 Scripts in COM

It is not required that a DBA know or understand this level of underlying operating system detail in order to write scripts that use this model. For the purpose of simplicity, an ODBC connection was used in several scripts in this chapter. In creating an ODBC connection the following procedure was used:

1. Start Menu -> Settings->Control Panel
2. Administrative Tools -> Data Sources (ODBC)
3. Add a System Data Source Name (DSN), hilight the database, see Figure 9-9

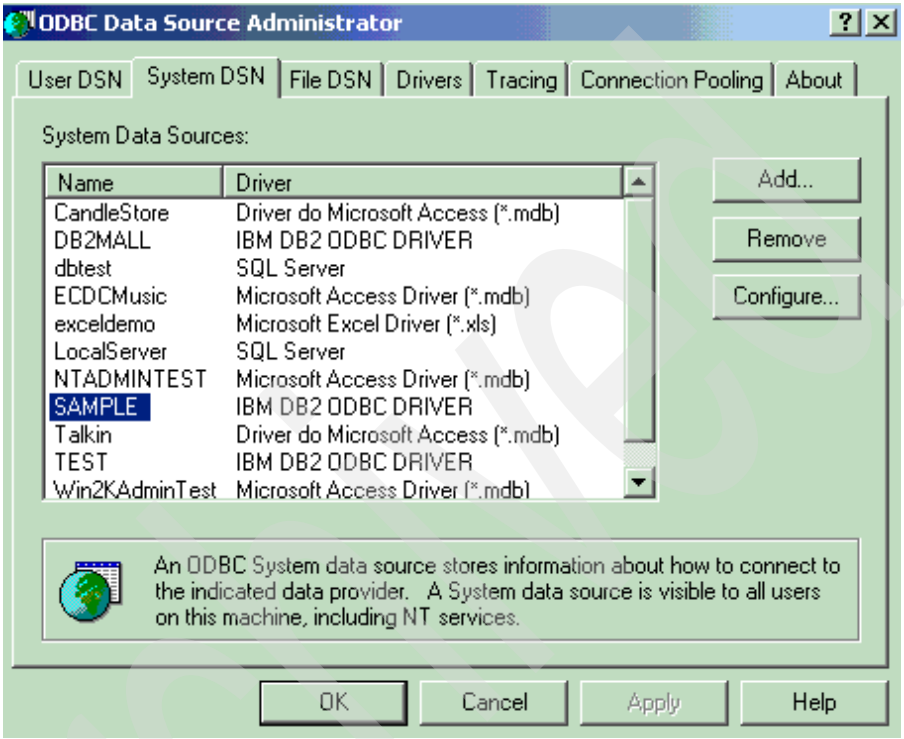

Figure 9-9 ODBC Data source Administrator

4. Select a driver (Figure 9-10).
5. Type in a Data Source Name (DSN).
6. Type in the Other Option information (suggested).

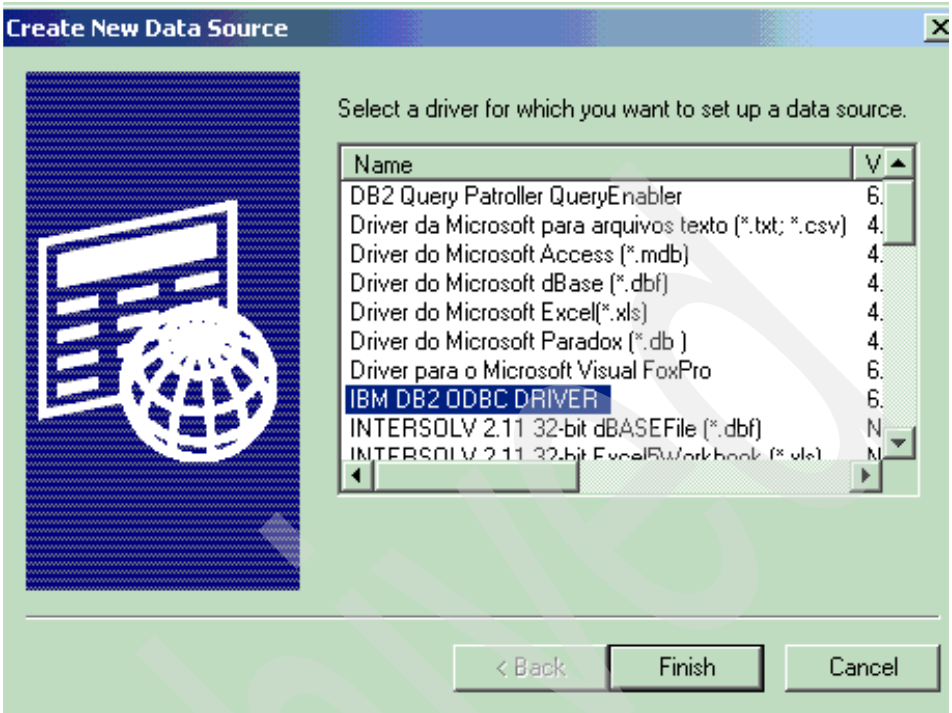

Figure 9-10 DB2 ODBC driver

7. Click the **Add** Button — and follow the DB2 Utility to find the actual database file and make a connection (Figure 9-11).

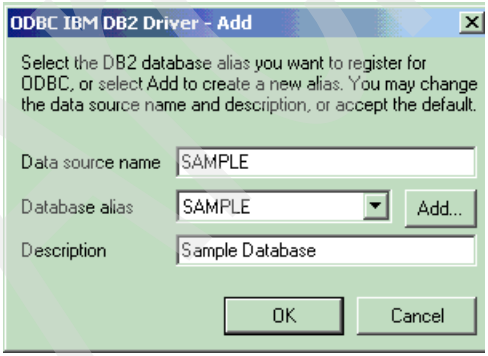

Figure 9-11 Add ODBC driver

8. Once you have completed the DB2 connection utility, you have now created a logical reference (DSN) that you can use in your scripts to connect to the DB2 database.

Active Server Pages (ASP)

Active Server Pages (ASP) is a server-side scripting environment that you can use to create interactive Web pages and build powerful Web applications. When the server receives a request for an ASP file, it processes server-side scripts contained in the file to build the Web page that is sent to the browser. In addition to server-side scripts, ASP files can contain HTML (including related client-side scripts) as well as calls to COM components that perform a variety of tasks, such as connecting to a database or processing business logic. Active Server Pages (ASP) makes it easy to generate dynamic content for the Web and to build powerful Web applications.

Since ASP is designed to be language-neutral, if you are skilled at a scripting language such as VBScript, JScript, or PERL, you already know how to use Active Server Pages. What more, in your ASP pages you can use any scripting language for which you have installed a COM compliant scripting engine. ASP comes with VBScript and JScript scripting engines, but you can also install scripting engines for PERL, REXX, and Python, which are available through third-party vendors.

Active Server Pages (ASP) has been enhanced with features that make it easier to use for scripters and Web application developers

- **New Flow Control Capabilities** The ASP Server object now has two new methods that you can use to control program flow: Server.Transfer and Server.Execute. Rather than redirecting requests, which requires an expensive round-trip to the client, you can use these methods to transfer requests directly to an .asp file, without leaving the server.
- **Error Handling** ASP now has a new error-handling capability, so that you can trap errors in a custom error message .asp file. You can use the new Server.GetLastError method to display useful information, such as an error description or the line number where the error occurred.
- **Performance-Enhanced Objects** ASP now provides performance-enhanced versions of its popular installable components. These objects will scale reliably in a wide range of Web publishing environments
- **XML Integration** Extensible Markup Language (XML) enables you to semantically describe the complex structure of data or documents, which can then be shared across a variety of applications, clients, and servers. Using the Microsoft® XML Parser, included with Microsoft® Internet Explorer version 4.0, or later, you can create server-side applications enabling your Web server to exchange XML-formatted data with Internet Explorer version 4.0 or later, or with other servers having XML parsing capabilities.
- **Windows Script Components** ASP supports Microsoft's powerful new scripting technology, Windows Script Components. Now you can turn your

business logic script procedures into reusable Component Object Model (COM) components that you can use in your Web applications, as well as in other COM-compliant programs

- **A New Way to Determine Browser Capabilities** ASP has a new feature for determining the exact capabilities of a browser. When a browser sends a cookie describing its capabilities (such a cookie can be installed by using a simple client-side script) you can create an instance of the Browser Capabilities component that retrieves the browser's properties as returned by the cookie. You can use this feature to discover a browser's capabilities and adjust your application accordingly
- **ASP Self-Tuning** ASP now detects when executing requests are blocked by external resources. It automatically provides more threads to simultaneously execute additional requests and to continue normal processing. If the CPU becomes overburdened, ASP curtails the number of threads. This reduces the constant switching that occurs when too many non-blocking requests are executing simultaneously.
- **Server-Side Include with SRC Attribute** You can now use the HTML <SCRIPT></SCRIPT> tag's SRC attribute to do server-side includes. When you use the SRC attribute to specify a virtual or relative path, and use the RUNAT=SERVER attribute to denote server-side execution, you can achieve the same functionality as the #Include directive.
- **Encoded ASP Scripts** Traditionally, Web developers have been unable to prevent others from viewing the logic behind their scripts. ASP now supports a new script-encoding utility provided with Microsoft® Visual Basic Scripting Edition (VBScript) and Microsoft® JScript 5.0. Web developers can apply an encoding scheme to both client and server-side scripts that makes the programmatic logic appear as unreadable ASCII characters. Encoded scripts are decoded at run time by the script engine, so there's no need for a separate utility. Although this feature is not intended as a secure, encrypted solution, it can prevent most casual users from browsing or copying scripts. Web site.
- **International ASP Development** Two new properties were added to the Response object: Response.CodePage and Response.LCID. These properties provide page level control over codepage and locale settings for dynamic strings, without having to enable sessions. Session.CodePage and Session.LCID still provide session level control over codepage and locale settings for dynamic strings; however, they no longer can be inadvertently overwritten by @CodePage and @LCID. @CodePage and @LCID still affect static strings in Web pages. By default, all of these values are implicitly set by the default system ANSI codepage (CP_ACP) and the default system locale (LOCALE_SYSTEM_DEFAULT). However, if the metabase properties AspCodePage or AspLCID are set for any Web site or virtual directory, these become the default values. The metabase properties allow for global.asa files

that need different codepages than the system default. Improved UTF-8 support includes support for surrogate characters and support for localized characters in the intrinsic objects, like form data and server variables.

Sample code

In this section we provide the sample ASP code using ADO via ODBC.

List data dictionary (tables and columns definitions)

This script illustrates how a DBA can extract table structures definitions out of an existing database schema. Most commercial enterprise level database have catalog or system data dictionary tables that store all database object definitions. This particular script is reporting the table structure definitions. This provides a convenient reference for the DBA. This script can also be easily modified by a SQL savvy DBA to re-create a schema definition that can be used to recreate a similar database schema or be imported into modeling tools like Erwin and others alike.

(ASP using ADO via ODBC)

```
<!-- DB2 System Catalog Tables Data Dictionary Structures
   Frank Reddington National Architect Center of Excellence -->


<html>
<body>

<%
Dim i
Dim fldSales
Set cn = Server.CreateObject("ADODB.Connection")
Set rs = Server.CreateObject("ADODB.Recordset")

cn.Open "dsn=SAMPLE"
set rs = cn.Execute("SELECT TABSCHEMA,TABNAME,COLNAME,TYPENAME,LENGTH FROM
SYSCAT.COLUMNS ORDER BY TABNAME,COLNAME")
%>
<center>
<table border = 1>
</tr>
<%
for each fldSales in rs.Fields
response.write  "<td align=center><font size =2><b>" & fldSales.Name &
"</b></font></td>"
```

```
Next
response.write "</tr> <tr>"
while rs.EOF <> True

for each fldSales in rs.Fields
response.write  "<td align=center><font size = 2>" & fldSales.Value &
"</font></td>"
Next
response.write "</tr> <tr>"
rs.MoveNext
Wend

%>
</table>
</center>
</body>
</html>
```

Listing all DB2 objects

In order to maintain a database, a DBA must know what that component make up that database. It is therefore important that a DBA must keep an inventory of all object created in a database. This script will provide an example listing.

(ASP using ADO via ODBC)

```
<!-Pass a db object (i.e.
TABLESPACES,TABLES,SCHEMATA,COLUMNS,PACKAGES,TRIGGERS,PROCEDURES)
as an argument. Example all_db_objects.asp?dbobject=TABLES -->

<html>
<body>
<%
Dim i
Dim fldSales
Dim syscatsql
syscat = "select * from SYSCAT. " & Request.QueryString("dbobject")

Set cn = Server.CreateObject("ADODB.Connection")
Set rs = Server.CreateObject("ADODB.Recordset")

cn.Open "dsn=SAMPLE"
set rs = cn.Execute(sycatsql)
%>

<center>
<table border = 1>
</tr>
<%
```

```
for each fldSales in rs.Fields
response.write  "<td align=center>" & fldSales.Name & "</td>"
Next
response.write "</tr> <tr>"
while rs.EOF <> True

for each fldSales in rs.Fields
response.write  "<td align=center>" & fldSales.Value & "</td>"
Next
response.write "</tr> <tr>"
rs.MoveNext
Wend

%>
</table>
</center>
</body>
</html>
```

List all tables, owners, schema and table creation date

ASP using ADO via ODBC.

```
<html>
<body>

<%
Dim i
Dim fldSales
Set cn = Server.CreateObject("ADODB.Connection")
Set rs = Server.CreateObject("ADODB.Recordset")

cn.Open "dsn=SAMPLE"
set rs = cn.Execute("select create_time, definer,tabname,tabschema from
syscat.tables where type = 'T' ")
%>

<center>
<table border = 1>
</tr>
<%
for each fldSales in rs.Fields
response.write  "<td align=center>" & fldSales.Name & "</td>"
Next
response.write "</tr> <tr>"
while rs.EOF <> True
```

```
for each fldSales in rs.Fields
response.write  "<td align=center>" & fldSales.Value & "</td>"
Next
response.write "</tr> <tr>"
rs.MoveNext
Wend

%>
</table>
</center>
</body>
</html>
```

List all tables and table owners

(ASP using ADO via ODBC)

```
<html>
<body>

<%
Dim i
Dim fldSales
Set cn = Server.CreateObject("ADODB.Connection")
Set rs = Server.CreateObject("ADODB.Recordset")

cn.Open "dsn=SAMPLE"
set rs = cn.Execute("select definer,tabname from syscat.tables where type =
'T'")
%>

<center>
<table border = 1>
</tr>
<%
for each fldSales in rs.Fields
response.write  "<td align=center>" & fldSales.Name & "</td>"
Next
response.write "</tr> <tr>"
while rs.EOF <> True

for each fldSales in rs.Fields
response.write  "<td align=center>" & fldSales.Value & "</td>"
Next
response.write "</tr> <tr>"
rs.MoveNext
Wend
```

```
%>
</table>
</center>
</body>
</html>
```

List table and tablespace size definitions

(ASP using ADO via ODBC)

```
<!-- ASP Program, that List Table Tablespace Size Defintion
     Frank Reddington Architect - National Architect COE -->

<html>
<body>

<%
Dim i
Dim fldSales
Set cn = Server.CreateObject("ADODB.Connection")
Set rs = Server.CreateObject("ADODB.Recordset")

cn.Open "dsn=SAMPLE"
set rs = cn.Execute("select a.tabname
Table,a.tbspace,b.pagesize,a.pctfree,b.extentsize from syscat.tables a,
syscat.tablespaces b where a.tbspace = b.tbspace")
%>

<center>
<table border = 1>
</tr>
<%
for each fldSales in rs.Fields
response.write  "<td align=center>" & " <font size = -2> " & fldSales.Name &
"</font></td>"
Next
response.write "</tr> <tr>"
while rs.EOF <> True

for each fldSales in rs.Fields
response.write  "<td align=center>" & " <font size = -2> " & fldSales.Value &
"</font></td>"
Next
response.write "</tr> <tr>"
rs.MoveNext
Wend

%>
```

```
</table>
</center>
</body>
</html>
```

The sample output is shown in Figure 9-12.

TABLE	TBSPACE	PAGESIZE	PCTFREE	EXTENTSIZE
ORG	USERSPACE1	4096	-1	32
STAFF	USERSPACE1	4096	-1	32
DEPARTMENT	USERSPACE1	4096	-1	32
EMPLOYEE	USERSPACE1	4096	-1	32
EMP_ACT	USERSPACE1	4096	-1	32
PROJECT	USERSPACE1	4096	-1	32
EMP_PHOTO	USERSPACE1	4096	-1	32
EMP_RESUME	USERSPACE1	4096	-1	32
SALES	USERSPACE1	4096	-1	32
CL_SCHED	USERSPACE1	4096	-1	32
IN_TRAY	USERSPACE1	4096	-1	32

Figure 9-12 Sample output

List all userids in DB2 database

(ASP using ADO via ODBC)

```
<html>
<body>

<%
Dim i
Dim fldSales
Set cn = Server.CreateObject("ADODB.Connection")
Set rs = Server.CreateObject("ADODB.Recordset")

cn.Open "dsn=SAMPLE"
set rs = cn.Execute("SELECT DISTINCT GRANTEE FROM SYSCAT.DBAUTH WHERE
GRANTEETYPE = 'U'")
%>

<center>
```

```
<table border = 1>
</tr>
<%
for each fldSales in rs.Fields
response.write  "<td align=center>" & fldSales.Name & "</td>"
Next
response.write "</tr> <tr>"
while rs.EOF <> True

for each fldSales in rs.Fields
response.write  "<td align=center>" & fldSales.Value & "</td>"
Next
response.write "</tr> <tr>"
rs.MoveNext
Wend

%>
</table>
</center>
</body>
</html>
```

Start the DB2 database service on a server

(ASP or VBScript (RUNAT=SERVER))

```
<%
' Start up a Service on this Computer
' Frank Reddington - National Architect Center of Excellence
' Example:servicestart.asp?ComputerName="independence"&TargetService="DB2"

Set Computer = GetObject("WinNT://" & Request.QueryString("ComputerName") & ","
& "computer")
Set Service = Computer.GetObject("service",
Request.QueryString("TargetService"))

'Check Service before Starting
If Service.Status = 1 Then
Service.Start
response.write Service.DisplayName & " started"
Else
    If Service.Status = 4 Then
    response.write Service.DisplayName & " already started"
    Else
    response.write Service.DisplayName & " cant start"
    End If
End If

' Display what the Current Configuration Settings Are
```

```
If Service.StartType = 2 Then
response.write "<br> The Service is configured for an Automatic Start"
Else
   If Service.StartType = 3 Then
   response.write "<br> The Service is configured to a Manual Start"
   Else
      If Service.StartType = 4 Then
      response.write "<br> The Service is Disabled"
      End If
   End If
End If

%>
```

Stop DB2 database service on a server

(ASP or VBScript (RUNAT=SERVER))

```
<%
' Stop up a Service on this Computer
' Frank Reddington - National Architect Center of Excellence
' Example: servicestop.asp?ComputerName="independence"&TargetService="DB2"


Set Computer = GetObject("WinNT://" & Request.QueryString("ComputerName") & ","
& "computer")
Set Service = Computer.GetObject("service",
Request.QueryString("TargetService"))

If Service.Status = 4 Then
Service.Stop
response.write Service.DisplayName & " Service Stopped"
Else
    If Service.Status = 1 Then
    response.write Service.DisplayName & " service is already stopped"
    Else
    response.write Service.DisplayName & " service can't be stopped"

    End If
End If

%>


Running Any Dynamic Queries (Reporting Tool) Through an Interactive Web Server
Form

(HTML)
```

Reporting tool through an interactive Web server form

Below is an HTML Form that will run interactive queries (SQL select statements) and format results in a table.

```
<html>
<body>
<table>
<form name = "dynamicsqlform" method=POST action =" dynamicsql.asp">
<tr>
<td>Enter a Query</td>
<td> <textarea  rows = 5 cols = 40 name = "dynamicsql"></textarea></td>
</tr>
<tr>
<td colspan =2 align=center><input type = "submit" value = "Execute
Query"></td>
</tr>
</form>
</table>
</body>
</html>
```

DB2 dynamic query from Web based form in ASP

```
<!-- DB2 DynamicQuery (dynamicsql.asp)  from Web Based Form Reporting Tool
   Frank Reddington National Architect Center of Excellence -->

<html>
<body>

<%
Dim i
Dim fldSales
Set cn = Server.CreateObject("ADODB.Connection")
Set rs = Server.CreateObject("ADODB.Recordset")

cn.Open "dsn=SAMPLE"
set rs = cn.Execute(Request.Form("dynamicsql"))
%>

<center>
<table border = 1>
</tr>
<%
for each fldSales in rs.Fields
response.write  "<td align=center>" & fldSales.Name & "</td>"
Next
response.write "</tr> <tr>"
```

```
while rs.EOF <> True

for each fldSales in rs.Fields
response.write  "<td align=center>" & fldSales.Value & "</td>"
Next
response.write "</tr> <tr>"
rs.MoveNext
Wend

%>
</table>
</center>
</body>
</html>
```

Running SQL from a batch file

Perl using ADO via ODBC.

```
# Windows 2000 Perl Script than takes a SQL Transactions in from a batch file
# Frank Reddington Architect - National Architects COE

use OLE;
$conn = CreateObject OLE "ADODB.Connection" ||
        die "CreateObject: $!";    # create ADO auto object
$conn->Open('DSN=TEST');   # connect to data source

print "Enter the sql to run (i.e. c:\\sqllib\\sqlinput.sql)\n";
$sqlinfile = <STDIN>;

#print $sqlinfile;

open (SQLIN,"$sqlinfile");
while ($sql=<SQLIN>) {

chomp($sql);
print "$sql \n";
$rs = $conn->Execute($sql);        # execute sql

if ($sql =~/^select/) {

$FieldsCount =  $rs->Fields->Count -1;

for ($i=0;$i<=$FieldsCount;$i++) {
    print $rs->Fields($i)->Name;
    print "\t";
                                    }
    print "\n\n";
```

```
while(!$rs->EOF()) {

    for ($i=0;$i<=$FieldsCount;$i++) {
    print $rs->Fields($i)->Value;
    print "\t";
                                        }
    print "\n";
    $rs->MoveNext();
}
                           }

      }
$rs->Close();                       # shut down the recordset
$conn->Close();                     # close the data source
```

Running any SQL from a command prompt

PERL using ADO via ODBC

```
# Windows 2000 Perl Script than takes a SQL Transaction in from the command
prompt
# Frank Reddington Architect - National Architects COE

use OLE;
$conn = CreateObject OLE "ADODB.Connection" ||
        die "CreateObject: $!";    # create ADO auto object
$conn->Open('DSN=TEST');   # connect to data source

print "Enter the sql statement you want to run \n\n";
$sql = <STDIN>;
$rs = $conn->Execute($sql);        # execute sql
if ($sql =~/^select/) {

$FieldsCount =  $rs->Fields->Count -1;

for ($i=0;$i<=$FieldsCount;$i++) {
    print $rs->Fields($i)->Name;
    print "\t";
                                        }
    print "\n\n";

while(!$rs->EOF()) {

    for ($i=0;$i<=$FieldsCount;$i++) {
    print $rs->Fields($i)->Value;
    print "\t";
                                        }
    print "\n";
    $rs->MoveNext();
```

```
}
                        }

#$rs->Close();                      # shut down the recordset
#$conn->Close();                    # close the data source
```

Running a report (static query)

PERL using ADO via ODBC.

```
use OLE;

$conn = CreateObject OLE "ADODB.Connection" ||
        die "CreateObject: $!";    # create ADO auto object
$conn->Open('DSN=SAMPLE');   # connect to data source
$sql = "SELECT * FROM sales";
$rs = $conn->Execute($sql);        # grab all records in table
while(!$rs->EOF()) {

    $SALES_DATE = $rs->Fields(0)->Value;
    $SALES_PERSON = $rs->Fields('SALES_PERSON')->Value;
    $REGION = $rs->Fields('REGION')->Value;
    write;                         # print them out
    $rs->MoveNext();
}

$rs->Close();                      # shut down the recordset
$conn->Close();                    # close the data source
# some formats for a quick printout
format STDOUT =
@<<<<<<<<<<      @<<<<<<<<<<<<<<<<<<<   @<<<<<<<<<<<<<<<<<<<<<<<<<
$SALES_DATE,     $SALES_PERSON,            $REGION
.
format STDOUT_TOP =
Page @<<
$%
Date        Sales Person           Sales Region
=====       ====================   ========================
```

User account managment

- **Creating a new account**

(PERL using ADO via ODBC)

```
# Windows 2000 Perl Script than create account access to the
# database in from the command prompt
# Frank Reddington Architect - National Architects COE


use OLE;
$conn = CreateObject OLE "ADODB.Connection" ||
        die "CreateObject: $!";    # create ADO auto object
$conn->Open('DSN=SAMPLE');   # connect to data source


print "Enter the name of database account you want to create\n";
$account = <STDIN>;
chomp($account);

print "Does $account represent an User or a Group ? \n";
$account_type = <STDIN>;

chomp($account_type);

$account_type =~tr/[a-z]/[A-Z]/;
#print "$account, $account_type \n";

$sql = "GRANT CONNECT ON DATABASE TO $account_type $account";
$rs = $conn->Execute($sql);        # execute sql
```

- **Creating accounts from a file list**

```
(PERL using ADO via ODBC)

# Windows 2000 Perl Script than create account access
# to the database from a ASCII Delimited file
# Frank Reddington Architect - National Architects COE
# Example contents of Input file JSMITH,USER  and/or Team15,GROUP

use OLE;
$conn = CreateObject OLE "ADODB.Connection" ||
       die "CreateObject: $!";    # create ADO auto object
$conn->Open('DSN=SAMPLE');   # connect to data source



print "Enter the file to get create database accounts \n";
print "(i.e. c:\\sqllib\\useraccounts.txt)\n";
$sqlinfile = <STDIN>;

#print $sqlinfile;
```

```
open (SQLIN,"$sqlinfile");
while ($line=<SQLIN>) {

chomp($line);
($account,$account_type) = split(/,/,$line);

$account_type =~tr/[a-z]/[A-Z]/;
#print "$account, $account_type \n";

$sql = "GRANT CONNECT ON DATABASE TO $account_type $account";

$rs = $conn->Execute($sql);        # execute sql


      }
```

Example Input file (i.e. useraccount.txt)

```
JSmith,user
MJones,user
Team12,group
Team7,group
Tean15,group
```

- **Removing an account**

(PERL using ADO via ODBC)

```
# Windows 2000 Perl Script than removes account access from the database
# from the command prompt
# Frank Reddington Architect - National Architects COE


use OLE;
$conn = CreateObject OLE "ADODB.Connection" ||
       die "CreateObject: $!";    # create ADO auto object
$conn->Open('DSN=SAMPLE');   # connect to data source


print "Enter the name of database connection account you want to delete\n";
$account = <STDIN>;
chomp($account);

print "Does $account represent an User or a Group ? \n";
$account_type = <STDIN>;

chomp($account_type);

$account_type =~tr/[a-z]/[A-Z]/;
```

```
print "$account, $account_type \n";

$sql = "REVOKE CONNECT ON DATABASE FROM $account_type $account";

$rs = $conn->Execute($sql);        # execute sql
```

- **Remove Accounts from a File Input**

```
(PERL using ADO via ODBC)

# Windows 2000 Perl Script that removes account access to the database
# from an ASCII Delimited file
# Frank Reddington Architect - National Architects COE
# Example contents of Input file JSMITH,USER  and/or Team15,GROUP

use OLE;
$conn = CreateObject OLE "ADODB.Connection" ||
        die "CreateObject: $!";    # create ADO auto object
$conn->Open('DSN=SAMPLE');   # connect to data source


print "Enter the file to get to remove database accounts \n";
print "(i.e. c:\\sqllib\\useraccounts.txt)\n";
$sqlinfile = <STDIN>;

#print $sqlinfile;

open (SQLIN,"$sqlinfile");
while ($line=<SQLIN>) {

chomp($line);
($account,$account_type) = split(/,/,$line);

$account_type =~tr/[a-z]/[A-Z]/;
#print "$account, $account_type \n";

$sql = "REVOKE CONNECT ON DATABASE FROM $account_type $account";

$rs = $conn->Execute($sql);        # execute sql

     }
```

B

Sample REXX programs

This appendix provides two Object REXX sample programs which perform DB2 database backup:

- dbrxbackup.rex: database backup REXX program source code
- dbrxbackup.wsp: Object REXX program within Windows script file.

dbrxbackup.rexx

This Object REXX program receives database information as input parameters and backup the specified database.

```
/*====================================================================*/
/* Program: DB2RXBACKUP.REX                                           */
/* Purpose: Backup database.                                          */
/*====================================================================*/
instance   = ''                          /* instance              */
database   = ''        /* database*/
target     = ''             /* target                */
userid     = ''             /* userid                     */
password   = ''             /* password                   */
/*====================================================================*/
parse arg parameters

parse value parameters with '/i:' instance  ' /'; instance = STRIP(instance)
parse value parameters with '/d:' database  ' /'; database = STRIP(database)
parse value parameters with '/t:' target    ' /'; target   = STRIP(target)
parse value parameters with '/u:' userid    ' /'; userid   = STRIP(userid)
parse value parameters with '/p:' password  ' /'; password = STRIP(password)
if WORDPOS('/v',parameters) \= 0 then verbose=1

if database = '' | target = '' then do
   ok = LINEOUT(,'')
   ok = LINEOUT(,'db2rxbackup.wsf: This REXX script backs up a DB2 database.')
   ok = LINEOUT(,'')
   ok = LINEOUT(,'Usage: db2rxbackup.wsf [/i:value] /d:value /t:value
[/u:value] [/p:value] [/v]')
   ok = LINEOUT(,'')
   ok = LINEOUT(,'Options: ')
   ok = LINEOUT(,'')
   ok = LINEOUT(,'/i : instance.   ')
   ok = LINEOUT(,'/d : database.   ')
   ok = LINEOUT(,'/t : target.     ')
   ok = LINEOUT(,'/u : userid.     ')
   ok = LINEOUT(,'/p : password.   ')
   ok = LINEOUT(,'/v : verbose.    ')
   ok = LINEOUT(,'')
   ok = LINEOUT(,'Example: db2rxbackup.wsf /d:SAMPLE /t:d: /v')
   ok = LINEOUT(,'')
   return 0
   end /* ifdo */
else do
     /*
     //  initialize global variables
     */
```

```
      .environment['PROGRAM.NAME'] = 'db2rxbackup.rex'
      /*
      // initialize db2cmd interface
      */
      if RxFuncQuery('SQLDB2') \= 0 then
         ok = RxFuncAdd('SQLDB2','DB2AR','SQLDB2')
      /*
      // assign arguments to variables
      */
      if verbose then do
         ok = LINEOUT(,'Defined Variables:')
         ok = LINEOUT(,'var instance='instance)
         ok = LINEOUT(,'var database='database)
         ok = LINEOUT(,'var target='target)
         ok = LINEOUT(,'var userid='userid)
         ok = LINEOUT(,'var password='password)
         ok = LINEOUT(,'var verbose='verbose)
         end /* ifdo */
      /*
      // Attach to instance
      */
      if instance \= '' then
         ok = RxDb2Attach(instance)
      /*
      // Backup database
      */
      ok = RxDb2BackupDatabase(database, target)
      /*
      // Detach to instance
      */
      if instance \= '' then
         ok = RxDb2Detach(instance)

end /* elsedo */

exit 1

/*--------------------------------------------------------------------------*/
::ROUTINE RxDb2Attach PUBLIC
/*--------------------------------------------------------------------------*/
use arg instance, userid, password

ok = CHAROUT(,.program.name'.RxDb2Attach('instance')...')

if password \= 'PASSWORD' & password \= ''  then
   db2command = 'ATTACH TO 'instance' USER 'userid' USING 'password
else
if userid \= 'USERID' & userid \= '' then
   db2command  'ATTACH TO 'instance' USER 'userid
```

```
else
   db2command = 'ATTACH TO 'instance

call SQLDB2 db2command

if RxSqlCodeCheck(SQLCA.SQLCODE,SQLMSG) \= 1 then
   return 0

return 1

/*--------------------------------------------------------------------*/
::ROUTINE RxDb2Detach PUBLIC
/*--------------------------------------------------------------------*/
ok = CHAROUT(,.program.name'.RxDb2Detach()...')

call SQLDB2 'DETACH'
if RxSqlCodeCheck(SQLCA.SQLCODE,SQLMSG) \= 1 then
   return 0

return 1

/*--------------------------------------------------------------------*/
::ROUTINE RxDb2BackupDatabase PUBLIC
/*--------------------------------------------------------------------*/
use arg database, target

ok = CHAROUT(,.program.name'.RxDb2BackupDatabase('database')...')

call SQLDB2 'backup database 'database' to 'target
if RxSqlCodeCheck(SQLCA.SQLCODE,SQLMSG) \= 1 then
   return 0

return 1

/*--------------------------------------------------------------------*/
::ROUTINE RxSqlCodeCheck PUBLIC
/*--------------------------------------------------------------------*/
use arg resultcode, resultmssg

if resultcode = 0 then do
   ok = LINEOUT(,'done!')
   end
else
if resultcode > 0 then do
   ok = LINEOUT(,'oops, rc='resultcode)
   ok = LINEOUT(,resultmssg)
   end /* ifdo */
else
if resultcode < 0 then do
```

```
   ok = LINEOUT(,'failed, rc='resultcode)
   if resultmssg \= '' then
      ok = LINEOUT(,resultmssg)
   return 0
   end /* elsedo */

return 1
```

dbrxbackup.wsf

This is an Object REXX program embedded within Windows script file. This program perform offline DB2 database backup.

```
<job>
    <runtime>
       <description>db2rxbackup.wsf: This REXX script backs up a DB2 database.
       </description>
        <named
            name="i"
            helpstring="instance."
            type="string"
            required="false"
        />
        <named
            name="d"
            helpstring="database."
            type="string"
            required="true"
        />
   <named
            name="t"
            helpstring="target."
            type="string"
            required="true"
   />
   <named
            name="u"
            helpstring="userid."
            type="string"
            required="false"
        />
        <named
            name="p"
            helpstring="password."
            type="string"
            required="false"
```

```
   />
        <named
            name="v"
            helpstring="verbose."
            type="simple"
            required="false"
        />
       <example>
     Example: db2rxbackup.wsf /d:SAMPLE /t:d: /v
       </example>
    </runtime>

<script language="Object REXX">

if \WScript~Arguments~Named~Exists("d") | \WScript~Arguments~Named~Exists("t")
then do
    WScript~Arguments~ShowUsage();
    WScript~Quit(0);
   end /* ifdo */
else do
     /*
     //  initialize global variables
     */
     .environment['PROGRAM.NAME'] = 'db2rxbackup.wsf'
     /*
     // initialize db2cmd interface
     */
     if RxFuncQuery('SQLDB2') \= 0 then
        ok = RxFuncAdd('SQLDB2','DB2AR','SQLDB2')
     /*
     // assign arguments to variables
     */
     if WScript~Arguments~Named~Exists("i") then
        instance = WScript~Arguments~Named("i")
     if WScript~Arguments~Named~Exists("d") then
        database = WScript~Arguments~Named("d")
     if WScript~Arguments~Named~Exists("t") then
        target = WScript~Arguments~Named("t")
     if WScript~Arguments~Named~Exists("u") then
        userid = WScript~Arguments~Named("u")
     if WScript~Arguments~Named~Exists("p") then
        password = WScript~Arguments~Named("p")
     verbose = WScript~Arguments~Named~Exists("v");
     if verbose then do
        WScript~Echo('Defined Variables:');
        WScript~Echo('var instance='instance);
        WScript~Echo('var database='database);
        WScript~Echo('var target='target);
        WScript~Echo('var userid='userid);
```

```
        WScript~Echo('var password='password);
        WScript~Echo('var verbose='verbose);
        end /* ifdo */

     if instance \= 'INSTANCE' & instance \= '' then
        ok = RxDb2Attach(instance,userid,password);

     ok = RxDb2BackupDatabase(database,target);

     if instance \= 'INSTANCE' & instance \= '' then
        ok = RxDb2Detach();

   end /* elsedo */

WScript~Quit(1);

/*--------------------------------------------------------------------------*/
::ROUTINE RxDb2Attach PUBLIC
/*--------------------------------------------------------------------------*/
use arg instance, userid, password

ok = WScript~StdOut~Write(.program.name'.RxDb2Attach('instance')...')

if password \= 'PASSWORD' & password \= ''  then
   db2command = 'ATTACH TO 'instance' USER 'userid' USING 'password
else
if userid \= 'USERID' & userid \= '' then
   db2command  'ATTACH TO 'instance' USER 'userid
else
   db2command = 'ATTACH TO 'instance

call SQLDB2 db2command

if RxSqlCodeCheck(SQLCA.SQLCODE,SQLMSG) \= 1 then
   return 0

return 1

/*--------------------------------------------------------------------------*/
::ROUTINE RxDb2Detach PUBLIC
/*--------------------------------------------------------------------------*/
ok = WScript~StdOut~Write(.program.name'.RxDb2Detach()...')

call SQLDB2 'DETACH'
if RxSqlCodeCheck(SQLCA.SQLCODE,SQLMSG) \= 1 then
   return 0

return 1
```

```
/*--------------------------------------------------------------------------*/
::ROUTINE RxDb2BackupDatabase PUBLIC
/*--------------------------------------------------------------------------*/
use arg database, target

ok = WScript~StdOut~Write(.program.name'.RxDb2BackupDatabase('database')...')

call SQLDB2 'backup database 'database' to 'target
if RxSqlCodeCheck(SQLCA.SQLCODE,SQLMSG) \= 1 then
   return 0

return 1

/*--------------------------------------------------------------------------*/
::ROUTINE RxSqlCodeCheck PUBLIC
/*--------------------------------------------------------------------------*/
use arg resultcode, resultmssg

.environment['PROGRAM.RESULTCODE'] = resultcode
.environment['PROGRAM.RESULTMSSG'] = resultmssg

if resultcode = 0 then do
   ok = WScript~StdOut~WriteLine('done!')
   end
else
if resultcode > 0 then do
   ok = WScript~StdOut~WriteLine('oops, rc='resultcode)
   ok = WScript~StdOut~WriteLine(resultmssg)
   end /* ifdo */
else
if resultcode < 0 then do
   ok = WScript~StdOut~WriteLine('failed, rc='resultcode)
   if resultmssg \= '' then
      ok = WScript~StdOut~WriteLine(resultmssg)
   return 0
   end /* elsedo */

return 1

</script>

</job>
```

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 528.

- *DB2 UDB V7.1 Performance Tuning Guide,* SG24-6012

Other resources

These publications are also relevant as further information sources:

- *IBM DB2 Universal Database Command Reference*, SC09-4828
- *IBM DB2 Universal Database What's New*, SC09-4848
- *IBM DB2 Universal Database Administration Guide: Planning*, SC09-4822
- *IBM DB2 Universal Database Administration Guide: Implementation*, SC09-4820
- *IBM DB2 Universal Database Administration Guide: Performance*, SC09-4821
- *IBM DB2 Universal Database Data Movement Utilities Guide and Reference*, SC09-4830
- *IBM DB2 Universal Database Data Recovery and High Availability Guide and Reference*, SC09-4831
- *IBM DB2 Universal Database Federated Systems Guide*, GC27-1224
- *IBM DB2 Universal Database Guide to GUI Tools for Administration and Development*, SC09-4851
- *IBM DB2 Universal Database SQL Reference, Volume 1*, SC09-4844
- *IBM DB2 Universal Database SQL Reference, Volume 2*, SC09-4845
- *IBM DB2 Universal Database System Monitor Guide and Reference*, SC09-4847
- *IBM DB2 Universal Database Application Development Guide: Building and Running Applications*, SC09-4825

- *IBM DB2 Universal Database Application Development Guide: Programming Client Applications*, SC09-4826
- *IBM DB2 Universal Database Application Development Guide: Programming Server Applications*, SC09-4827
- *IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1*, SC09-4849
- *IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2*, SC09-4850
- *IBM DB2 Universal Database Data Warehouse Center Application Integration Guide*, SC27-1124
- *IBM DB2 XML Extender Administration and Programming*, SC27-1234
- *IBM DB2 Universal Database Quick Beginnings for DB2 Clients*, GC09-4832
- *IBM DB2 Universal Database Quick Beginnings for DB2 Servers*, GC09-4836
- *IBM DB2 Universal Database Installation and Configuration Supplement*, GC09-4837

Referenced Web sites

These Web sites are also relevant as further information sources:

- DB2 ODBC Catalog Optimizer location:

  `ftp://ftp.software.ibm.com/ps/products/db2/tools/`

- IBM DB2 Universal Database

  `http://www.ibm.com/software/data/db2/udb`

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

`ibm.com/redbooks`

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

C

D

E

F

M

N

O

S

T

U

V

W

X

Y

DB2 UDB Exploitation of the Windows Environment

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

DB2 UDB Exploitation of the Windows Environment

Detailed examples for leveraging DB2 UDB V8.1 in Windows 2000 environment

Installing, security, monitoring, performance, high availability

Application development guidelines

Although roughly ninety percent of DB2 Universal Database (UDB) on distributed platforms is common code, IBM makes great efforts to exploit the features of each operating system to which it ports; and on Windows, this is no exception. Clearly, the first database product certified for Windows 2000 makes use of more features in the Windows operating system than any other database on the market today and delivers world class record setting price performance.

This IBM Redbook is an informative guide that describes how to effectively implement DB2 UDB V8.1 with Microsoft Windows 2000 operating systems. It is intended for anyone who needs both an introduction and detailed information on installing, configuring, and managing DB2 UDB on Windows.

We begin by exploring the DB2 UDB product family. We cover installation procedures for interactive, unattended, and enterprise deployment. We describe DB2 UDB integration with the Windows domain security model. We discuss system and database performance factors and how to optimize and monitor them as well. We walk you through high availability features including standby and clustered servers. We discuss application development from a DB2 UDB and Windows perspective, providing general tips on building applications and cover integration of the Microsoft development tools.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6893-00 ISBN 0738427985