Getting Started With

DB2 Express-C

Ideal for application developers and administrators

Updated to 9.7

Raul F. Chong

with

**Ian Hakes,
Rav Ahuja**

Foreword by
Dr. Arvind Krishna

A book for the Community by the Community

GETTING STARTED WITH

DB2 Express-C

A book for the community by the community

RAUL CHONG, IAN HAKES, RAV AHUJA

FOREWORD BY DR. ARVIND KRISHNA

THIRD EDITION

Contents

About this book

Notices and trademarks

Who should read this book?

This book is intended for anyone who works with or intends to work with databases, such as database administrators (DBAs), application developers, consultants, software architects, product managers, instructors, and students.

How is this book structured?

Part I, Overview and Setup, explains what DB2 Express-C edition is all about, introduces the DB2 family of products and features, assists with installation and creation of databases, and explores the tools available with DB2.

Part II, Learning DB2: Database Administration, is designed to familiarize you with the DB2 environment, architecture, remote connectivity, database objects, data movement (import/export/load), security, backup and recovery, concurrency and locking, and other common maintenance tasks.

Part III - Learning DB2: Application Development, introduces DB2 application development, including server-side and client-side development. It also discusses SQL/XML, XQuery, and pureXML®.

The Appendix contains useful information about troubleshooting.

Exercises are provided for most chapters; and any input files required for these labs are provided in the compressed file **`expressc_book_exercises_9.7.zip`** that accompanies this book.

The materials in this book are also used in courses offered as part of the ***DB2 on Campus Program***, and closely match the e-learning video presentations available at www.channelDB2.com/oncampus. You can read more about the DB2 on Campus program at the DB2 Express-C website: www.ibm.com/db2/express/students.html.

Note:

For more information about the DB2 on Campus program, watch the video at:
http://www.channeldb2.com/video/video/show?id=807741:Video:3902

Now in its third edition we have made several changes and additions. For those who have read the second edition of the book which covered DB2 9.5, we are making it easier for you to find changes to the book that correspond to new features or updates in version 9.7 of DB2. The changes can be easily identified with this icon:

A book for the community

This book was created by the DB2 Express-C team. The online version is released to the DB2 Express-C community at no-charge. As of the time of writing, this book has been downloaded more than 85,000 times and translated into 9 languages by volunteers around the world. A true community effort!. If you would like to provide feedback, contribute new material, improve existing material, or help translate this book to another language, please send an email of your planned contribution to db2x@ca.ibm.com with the subject "DB2 Express-C book changes."

The success of this book has been the inspiration to develop more than 25 new free ebooks about IBM products, and also about non-IBM technologies. The books are part of the *DB2 on Campus Book Series*, which was launched on January 2010.

For more information about this book or the DB2 on Campus Book Series visit the IBM® DB2 Express-C Web site at ibm.com/db2/express

Authors and Contributors

The following people have provided content and other significant contributions to this book.

Raul F. Chong – Lead Author

Raul is the DB2 on Campus Program Manager at the IBM Toronto Lab.

Ian Hakes – Co-author and Editor

Ian is a former DB2 Express-C Community Facilitator and now works as a usability expert at the IBM Toronto Lab.

Rav S. Ahuja – Co-author and Publishing

Rav is a senior DB2 Product Manager at the IBM Toronto Lab.

Acknowledgements

We greatly thank the following individuals for their assistance and developing materials referenced in this book:

- Ted Wasserman, Clara Liu, and Paul Yip from the IBM Toronto Lab who developed materials that served as the framework for this book.
- Don Chamberlin and Cindy Saracco for their IBM developerWorks articles on XQuery, and Matthias Nicola for his presentations on pureXML.
- Kevin Czap and Grant Hutchison for developing DB2 technical briefing materials.
- Katherine Boyachok and Natasha Tolub for the cover design of this book.
- Susan Visser for reviewing and providing assistance with publishing this book.

Foreword

Innovation is the cornerstone of technological progress. At IBM, innovation has been an integral part of our data server evolution. Having pioneered data management techniques in the 1960s and 1970s, IBM continues to deliver innovative information management technologies, as reflected in the thousands of data management patents authored by IBM technologists. As a result, some of the largest organizations in the world rely on IBM products, including DB2, to power their most demanding and mission-critical data management solutions.

However, DB2 is no longer just for large enterprises. With the release of DB2 Express-C, award-winning DB2 technology is now available to small and mid-size companies – and with no cost! Although there are other free or open-source data servers available, DB2 Express-C offers unique advantages over these alternatives.

There are many technological advances present in DB2 Express-C. These innovations provide new capabilities, reduce administrative burdens, improve performance, and reduce infrastructure cost.

DB2 Express-C hybrid technology is capable of managing both relational and XML data in their native formats. This makes DB2 ideal for powering the new breed of SOA and Web 2.0 applications where XML data flows in abundance. Unlike other "free" data servers, DB2 Express-C does not limit the amount of data you can store in a database or the number of databases you can create on a system. And of course, if you require support or assistance from IBM, help is just a click away.

This book serves as a guide to getting started with and using DB2 Express-C. It will assist you with understanding DB2 concepts and enable you to develop skills for DB2 administration and application development. The skills and knowledge you will gain are relevant to the other advanced editions of DB2 on Linux, UNIX, and Windows.

While DB2 Express-C is not an open-source product, at IBM we very much believe in supporting and fostering community initiatives. I am delighted that this book is being developed by DB2 Express-C community members and will be freely available to anyone in the community. I encourage you to enrich and update this book with your know-how and experiences, and also to assist with translating this book into other languages so others can benefit.

Arvind Krishna
General Manager
Information Management, IBM Software Group

PART I – OVERVIEW AND SETUP

1

Chapter 1 – What is DB2 Express-C?

DB2 Express-C data server software ("DB2 Express-C") is a member of the IBM DB2 family of powerful data server software for managing both relational and XML data. DB2 Express-C is a free, no-limits, and easy to use edition of DB2. The 'C' in DB2 Express-C stands for the Community. A community of DB2 Express-C users that bands together to assist each other, both online and offline. The DB2 Express-C community consists of all sorts of people and companies who design, develop, deploy, or utilize database solutions. Community members include:

- Application developers who require an open standards database software for building standalone, client-server, web-based, and enterprise applications

- ISVs, hardware vendors, infrastructure stack vendors, and other types of solution providers who want to bundle or embed a full-featured data server as part of their solutions

- Consultants, database administrators, and IT architects who need a robust data server for training, skills development, evaluation and prototyping

- Startups, small and medium-sized companies who need a reliable data server for their applications and operations

- Database hobbyists and cutting-edge technology enthusiasts who want an easy to use data server for building Web 2.0 and next generation applications

- Students, teachers, and other academic users who want a highly versatile data server for teaching, courseware, projects and research

DB2 Express-C shares the same core functionality and code-base as the other priced editions of DB2 on Linux, UNIX, and Windows. DB2 Express-C can be run on either 32-bit or 64-bit systems with Linux or Windows operating systems. It is also available on Solaris (x64) and as a beta on Mac OS X (x64). It can be run on systems with any amount of processors and memory and does not have any specialized storage or system setup requirements. DB2 Express-C also includes pureXML at no charge. pureXML is a technology unique to DB2 that stores and processes XML documents natively.

1.1 Free to develop, deploy, and distribute…no limits!

This sentence summarizes the key ideals behind DB2 Express-C:

- **Free to develop**: If you are an application developer and need a database for your application, you can use DB2 Express-C.

- **Free to deploy**: If you are working in a production environment, and need a data management system to store your vital records, you can use DB2 Express-C.

- **Free to distribute**: If you are developing an application or a tool that requires an embedded data server, you can include DB2 Express-C. Even though DB2 Express-C is embedded in your application, and distributed every time you sell your application, it is still free. You are required to register with IBM in order to re-distribute DB2 Express-C; however this registration is also free of charge.

- **No limits**: While other competitor database offerings set limits on database sizes, number of databases, and number of users, with DB2 Express-C there are NO data size limits. Your database can continue to grow without violating the licensing agreement. There are also no license imposed limits to the number of connections or users per server.

Note:

To learn more about DB2 Express-C and its role in the information on-demand world and Web 2.0, take a look at this video presentation:
http://www.channeldb2.com/video/video/show?id=807741:Video:3922

1.2 Downloading DB2 Express-C

All the DB2 Express-C images can be downloaded and used for free from ibm.com/db2/express. The following images are available:

- DB2 Express-C 9.7.2 for Windows
- DB2 Express-C 9.7.2 for Windows 64-bit
- DB2 Express-C 9.7.2 for Linux
- DB2 Express-C 9.7.2 for Linux 64-bit
- DB2 Express-C 9.7.2 for Linux on Power
- DB2 Express-C 9.7.2 for Solaris x86-64
- DB2 Express-C 9.5.2 beta for Mac OS X

Note:

DB2 Express-C on Windows is also offered in a lighter version that is 44% smaller than the regular one. It is available in English only and does not include GUI tools and the Text Search feature.

1.3 User assistance and technical support

If you have technical questions about DB2 Express-C, you can post your questions in the DB2 Express-C forum. This free forum is monitored by DB2 experts from IBM, though it is the community who provides most of the answers on a voluntary basis.

IBM also gives users the choice to purchase a low cost DB2 Express data server software ("DB2 Express") yearly subscription (also known as the Fixed Term License or FTL). This subscription comes with the backing of IBM for 24 x 7 technical support and software updates. In addition to support and software maintenance, with the yearly low cost subscription fee (approximately $1,990 per server per year in the United States – may vary in other countries) you also get to use additional features: HADR (clustering for High Availability and Disaster Recovery), SQL replication (for replicating data with other DB2 servers), and Backup Compression (for creating compressed backup copies of the database). Further information about the subscription option can be found at: ibm.com/db2/express/support.html

1.4 DB2 servers

All DB2 server editions contain the same core components; they are packaged in such a way that users can choose the functions they need at the right price. *Figure 1.1* illustrates the different DB2 product editions.
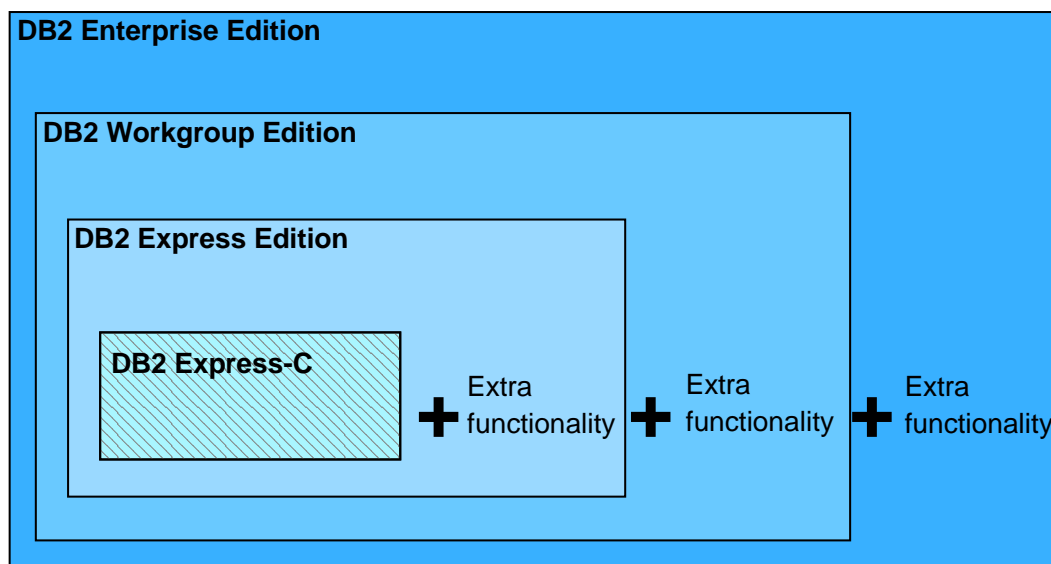
Figure 1.1 – DB2 Servers

As shown in *Figure 1.1*, DB2 Express-C is the same as DB2 Express without a few components. DB2 Express-C is free to the community. Technical assistance is available

through a free online forum, or you can receive official 24 x 7 IBM DB2 technical support if you purchase the yearly subscription (DB2 Express Fixed Term License).

Figure 1.1 also explains why it is so easy to upgrade from DB2 Express-C. If you wish to upgrade to any of the other DB2 servers in the future, all DB2 servers have the same core components. This also means that any application developed for one edition will work, without modification, in other editions. And any skills you learn in one edition will apply to other editions.

1.5 DB2 clients and drivers

A DB2 client includes the necessary functionality to connect to a DB2 server; however, a DB2 client does not always need to be installed. For example, a JDBC Type 4 application only requires a JDBC driver to be installed to connect to a DB2 server. DB2 Clients and drivers come in several different flavors:

- IBM Data Server Client: most complete, includes GUI Tools, drivers
- IBM Data Server Runtime Client: a lightweight client with basic functionality, and includes drivers
- DB2 Runtime Client Merge Modules for Windows: mainly used to embed a DB2 runtime client as part of a Windows application installation
- IBM Data Server Driver for JDBC and SQLJ: allows Java applications to connect to DB2 servers without having to install a full client
- IBM Data Server Driver for ODBC and CLI: allows ODBC and CLI applications to connect to a DB2 server without the large footprint of having to install a client
- IBM Data Server Driver Package: Includes a Windows-specific driver with support for .NET environments in addition to ODBC, CLI and open source. This driver was previously known as the IBM Data Server Driver for ODBC, CLI and .NET.

new in
V9.7

Figure 1.2 shows the different DB2 clients and drivers available.
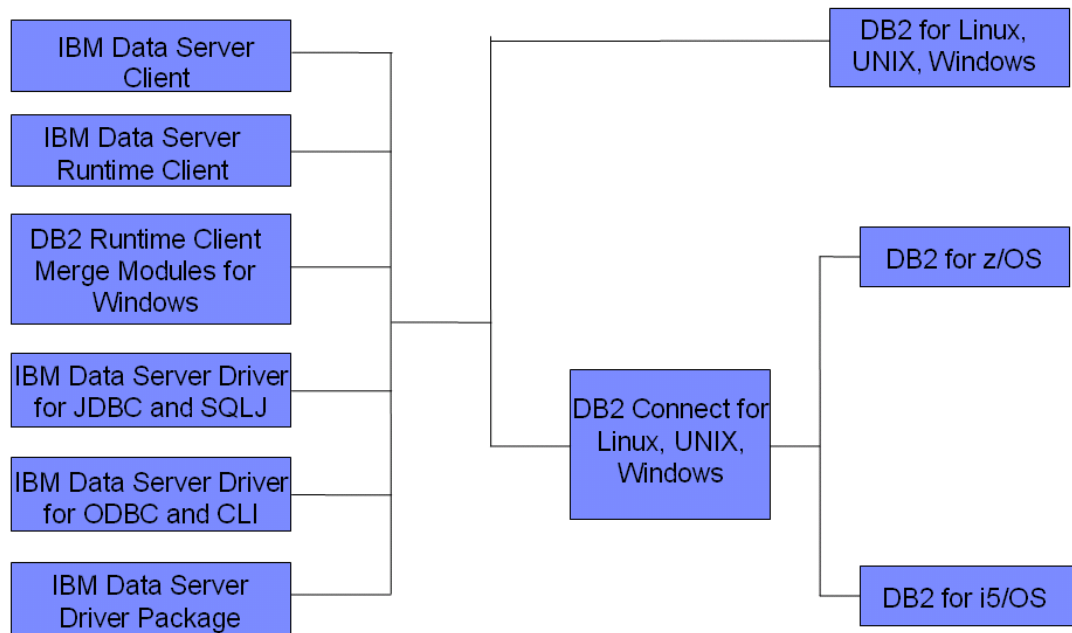
Figure 1.2 – DB2 clients and drivers

On the left side of *Figure 1.2*, all the DB2 clients and drivers are shown. Although all DB2 clients include the required drivers, starting with DB2 data server software ("DB2") v.9 we provide the individual drivers as well. DB2 clients and drivers are all free and available for download from the DB2 Express-C web site. The clients and drivers can be used to connect to a DB2 server on Linux, UNIX or Windows. To connect to a DB2 for z/OS® or DB2 for i5/OS® server, you need to go through a DB2 Connect™ server (shown in the middle of *Figure 1.2*). We will discuss the DB2 Connect software ("DB2 Connect") in *Chapter 2.*

Note:

Though this book focuses on the DB2 data server, the IBM Data Server clients can also connect to other data servers in the IBM family such as Informix. Hence the generic name "IBM Data Server client" as opposed to the more specific "DB2 client".

1.6 Application development freedom

DB2 offers an application development environment that is standards-based and is transparent across the DB2 family. SQL standardization across the DB2 product line provides a common set of application programming interfaces for database access.

In addition, each DB2 product provides SQL pre-compilers and application programming interfaces (APIs) which allow developers to embed static and dynamic SQL in portable application programs. DB2 even has a native .NET managed provider and integration with Microsoft® Visual Studio tools.

Languages and standards you can use with DB2 include:

- SQL, XQuery, XPath

- C/C++ (CLI, ODBC and embedded SQL)

- Java (JDBC and SQLJ)

- COBOL

- PHP

- Perl

- Python

- Ruby on Rails

- .NET languages

- OLE-DB

- ADO

- MS Office: Excel, Access, Word

- Web services

1.7 DB2 versions versus DB2 editions

If you are new to DB2, you may be a bit confused as to the distinction between a DB2 version, and a DB2 edition.

Every few years, IBM publicly releases a new DB2 version. A version includes new features and significant improvements to the product. Currently, DB2 Version 9 is officially supported by IBM. A version may also have a few releases which are updates that can include some new functionality but are usually not significant enough to warrant a new version. For example 9.5 and 9.7 are release levels for DB2 Version 9. Over the past few years, IBM has come out with a new release of DB2 every 1-2 years, however new versions are typically spaced over 3 or more years apart. The most current release is V9.7, which became generally available (GA) in June of 2009. Each release may also have several modification levels, which typically contain fixes or correspond to fix pack levels, and seldom deliver new functionality. At the time of writing the most current version, release, and modification (V,R,M) level of DB2 Express-C is 9.7.0 which corresponds to a code-level of 9.7 with Fix pack 0, which means it is at the GA level.

On the other hand, editions are select offerings or package groupings within each version. As discussed earlier, an edition is a packaging of different functions for a given price and license. DB2 Version 9.7 (also known as DB2 9.7) has several editions; for example, DB2 Express-C 9.7, DB2 Express 9.7, DB2 Workgroup 9.7, and DB2 Enterprise 9.7 (see *Figure 1.1*).

1.8 Moving up to another DB2 edition

As your database needs grow, you may need to upgrade to a DB2 edition that supports a larger hardware configuration. If this situation arises, it is easy to upgrade to another DB2 edition:

- If you are upgrading from DB2 Express-C to DB2 Express (Fixed Term License) on the same computer, all you need to do is apply the license with the db2licm command.

- If you are upgrading to another DB2 edition on the same computer system, uninstall DB2 Express-C, and then install the new DB2 edition. When you uninstall DB2 Express-C, your databases are not deleted (but a backup is always recommended).

- If you are upgrading DB2 and the new edition will be installed on a different, larger computer using the same operating system, then install the new DB2 edition on the larger computer, backup your databases on the smaller computer, move the backup images to the larger computer, and restore the backup images to databases on the larger computer. You also need to save the instance configuration settings (dbm cfg) from your smaller computer, and apply this configuration to the larger computer. The backup and restore commands are discussed in more details in *Chapter 11, Backup and Recovery*. The dbm cfg is discussed in more detail in *Chapter 5, The DB2 Environment*.

- In either case, your client application will not need modification.

1.9 Maintenance and updates for DB2 Express-C

DB2 Express-C installation images are refreshed periodically. These refreshes generally coincide with availability of new releases or versions or when there are significant number of bug fixes accumulated for the product. In the past, refreshes for DB2 Express-C have typically been made available once a year. However, note that DB2 Express-C, being a no-charge unwarranted offering, does not come with any official maintenance releases or regularly scheduled fix packs (that are released several times a year). Once a new refresh or a new release of DB2 Express-C is made available, the previous releases of DB2 Express-C are no longer maintained.

As discussed earlier, if you require access to security patches and regularly scheduled software updates or fix packs containing bug fixes, IBM offers the DB2 Express yearly subscription license (FTL). Once you purchase the subscription, your DB2 Express-C installation can be updated with the license key for FTL, which entitles you to DB2 technical support and access to updates and fix packs during the period the subscription license is valid. The subscription license also entitles you to free version upgrades, or if you prefer you have the flexibility to stay at a particular version or release, and just apply fix packs and security patches for as long as that release is supported and your yearly subscription is maintained.

1.10 Related free software and DB2 components

All the software that is available for download on the DB2 Express-C download page (www.ibm.com/db2/express/download.html) is free of charge. Besides the DB2 Express-C software, there are other useful software packages that can be downloaded and used for free:

- Visual Studio Add-ins

- DB2 Spatial Extender

There are also additional starter toolkits based on DB2 Express-C available for download from the IBM Alphaworks web site (www.alphaworks.ibm.com/datamgmt) that you may find useful:

- Starter Toolkit for DB2 on Rails (www.alphaworks.ibm.com/tech/db2onrails/)

- Web 2.0 Starter Toolkit for DB2 (www.alphaworks.ibm.com/tech/web2db2)

If you are looking for a lightweight application server that is free, IBM offers:

- WebSphere® Application Server – Community Edition (WAS CE)

Note:

Although spatial capabilities with DB2 have been available for close to 10 years, not many users know about it. You can take advantage of the DB2 Spatial Extender free of charge in all DB2 editions, including DB2 Express-C. The Spatial Extender allows you to work with spatial and geodetic data using SQL. For example, this capability can help you answer questions like, "What is the closest retail outlet for each customer who lives in Toronto and spent more than $3000 last year with us?" You can even use DB2 Spatial Extender for medical applications. For example, it could help answer the question, "What are the patterns of malignant cells in an MRI brain scan?"

For more information, refer to the About DB2 Spatial Extender topic in the IBM DB2 Database for Linux, UNIX, and Windows Information Center.

1.10.1 IBM Data Studio

IBM Data Studio is a tool based on Eclipse that allows you to manage your databases and help you develop XQuery, SQL scripts, user-defined functions, and stored procedures. An integrated debugger is included. In addition, IBM Data Studio allows you to work with physical data modeling (PDM) diagrams to understand entity relationships between tables. It can also help you to develop and publish data as a Web service using a drag and drop approach that requires no programming. IBM Data Studio replaces DB2 tools such as the Control Center and the Command Editor, which are now deprecated (they are included with DB2, but are no longer under development). IBM Data Studio is discussed in more detail in *Chapter 5, DB2 Tools*.

1.10.4 DB2 Text Search

DB2 Text Search is an optional integrated component of DB2. It is powered by the IBM OmniFind™ technology, and it allows you to perform powerful, fast and detailed full-text searches in text documents, including any XML documents stored natively in DB2. This component uses linguistic processing to find different forms of the search term within the text. For example, if you are looking for the word "study", DB2 Text Search also finds other forms of the word such as "studies" or "studied".

To install the DB2 Text Search component, choose a custom installation of DB2 Express-C, and select the DB2 Text Search feature within the Server support category.

Note:

Similar functionality is also available in a DB2 extender called the Net Search Extender (NSE). NSE is being deprecated in favor of DB2 Text Search,

1.10.5 WebSphere Application Server – Community Edition

IBM WebSphere Application Server - Community Edition (WASCE) is a lightweight Java EE 5 application server that is available free of charge. Built on Apache Geronimo technology, it harnesses the latest innovations from the open-source community to deliver an integrated, accessible, and flexible foundation for developing and deploying Java applications. Optional technical support for WASCE is available through an annual subscription.

1.11 Summary

The DB2 Express-C edition offers a best-of-breed product at no cost. It delivers the freedom to develop, deploy and distribute without any database size limitations, while still including the same core functionality and pureXML technology as the other editions of DB2. DB2 Express-C supports a wide array of clients, drivers and development languages, and it provides an easy upgrade path to other DB2 editions.

2

Chapter 2 – Related features and products

This chapter describes DB2 features included with the purchase of a DB2 Express yearly subscription license (Fixed term License or FTL). It also describes features included with other DB2 editions, in some cases, for an additional fee.

The differences between the free (unwarranted) DB2 Express-C and the Yearly Subscription option for DB2 Express are highlighted in the *Table 2.1* below.

Feature	Free (Unwarranted)	Paid Subscription* (FTL)
Core DB2 capabilities	Yes	Yes
Free admin tools	Yes	Yes
Free development tools	Yes	Yes
Autonomic capabilities	Yes	Yes
pureXML feature	Yes	Yes
No-charge community-based assistance***	Yes	Yes
Official IBM 24x7 support	No	Yes
Fixpacks	No	Yes
High Availability (HADR)	No	Yes
SQL Data replication	No	Yes
Backup Compression	No	Yes
Max. processor utilization	2 cores	4 cores (max 2 sockets)
Max. memory utilization	2GB	4GB
Update availability	Full refreshes at new releases, generally once a year	Security patches and Fixpacks several times a year
Access to install images for previous versions/releases	No, only the current release and beta images available	Yes, through IBM Passport Advantage
Price per server per year**	0	US $2,995

Table 2.1: Comparing the FREE DB2 Express-C with Paid Subscription (FTL)

* Features entitled with Subscription are available only while Subscriptions are valid.

** Subscription Price is for United States and subject to change without notice. Pricing in other countries may vary.

*** No-charge community-based assistance is available via the online forum.

Table 2.2 lists product features and whether they are included with the different editions of DB2 9.7. Features that you can purchase separately are listed by name for the corresponding DB2 edition and highlighted with a light grey background.

Function	DB2 Express Subscription (FTL)	DB2 Express Edition	DB2 Workgroup Server Edition (WSE)	DB2 Enterprise Server Edition (ESE)
Homogenous SQL Replication	Yes	Yes	Yes	Yes
Homogenous Federation	Yes	Yes	Yes	Yes
Net Search Extender, DB2 Text Search	Yes	Yes	Yes	Yes
Spatial Extender	Yes	Yes	Yes	Yes
Backup Compression	Yes	Yes	Yes	Yes
pureXML technology	Yes	Yes	Yes	Yes
High availability disaster recovery	Yes	High Availability Feature	Yes	Yes
Tivoli® System Automation	Yes		Yes	Yes
Advanced Copy Services	No		Yes	Yes
Online reorganization	No		Yes	Yes

MQT	No	No	No	Yes
MDC	No	No	No	Yes
Query parallelism	No	No	No	Yes
Connection concentrator	No	No	No	Yes
Table partitioning	No	No	No	Yes
Governor	No	No	No	Yes
Compression: row level, index, XML, temp table	No	No	No	Storage Optimization Feature
Label-based access control (LBAC)	No	No	No	Advanced Access Control Feature
Geodetic Extender	No	No	No	Geodetic Data Management Feature
Query Patroller	No	No	No	Performance Optimization Feature
DB2 workload management	No	No	No	
Performance Expert	No	No	No	
Homogenous Q Replication	No	No	No	Homogeneous Replication Feature for ESE
Database partitioning	No	No	No	No

Table 2.2: DB2 Version 9.7 editions: feature and function support

Features available with other DB2 editions are:

Chargeable DB2 Express Edition Features

- High Availability Feature

Features included at no-charge in DB2 Workgroup Edition:

- High Availability and Disaster Recovery (HADR), Tivoli System Automation, Online Re-org, Advanced Copy Services
- DB2 availability on additional UNIX platforms: AIX®, Solaris, and HP-UX

Features included at no-charge DB2 Enterprise Edition:

- Table (Range) Partitioning
- Materialized Query Tables (MQT)
- Multi-dimensional Clustering (MDC)
- Query Parallelism
- Connection Concentrator
- Governor

Chargeable DB2 Enterprise Edition Features

- Storage Optimization Feature (includes compression)
- Advanced Access Control (fine grained and advanced security)
- Performance Optimization (Workload Management, Performance Expert, Query Patroller)
- Geodetic Data Management (geographical location analysis)

Fee-based products related to DB2:

- DB2 Connect
- InfoSphere Warehouse Editions
- InfoSphere Balanced Warehouse
- WebSphere Federation Server
- WebSphere Replication Server

2.1 Features included with DB2 Express subscription (FTL)

This section outlines DB2 Fix packs, HADR and SQL replication.

2.1.1 Fix packs

A DB2 fix pack is a set of code fixes applied onto an installed DB2 product, in order to fix different issues reported after the product was released. With an installed subscription license, fix packs are free to download and install. They are typically available every four months or as warranted.

To download the latest fix pack, review the DB2 technical support site at
http://www.ibm.com/software/data/db2/support/db2_9/

2.1.2 High Availability Disaster Recovery (HADR)

High Availability Disaster Recovery (HADR) is a database reliability feature that provides a high-availability and disaster recovery solution for complete as well as partial site failures. An HADR environment generally consists of two data servers, the primary and the secondary (which can be in geographically apart locations). The primary server is where the source database is stored and accessed by client applications. As transactions are processed on the primary database, database log records are automatically shipped to the secondary server across the network. The secondary server has a cloned copy of the primary database, usually created by backing up the primary database and restoring it on the secondary system. When the primary database logs are received they are replayed and applied to the secondary database. Through continuous replay of the log records, the secondary database keeps an in-sync replica of the primary database that can take over if the primary database fails.

A full DB2-supported HADR solution gives you:

- Lightning fast failover capability, with complete transparency for customers and client applications

- Full transaction atomicity to prevent data loss

- The ability to upgrade systems or applications without visible service interruption

- Remote system failover, providing full recovery from local disaster striking the data center

- Easy management with DB2 graphical tools

- All of this with negligible impact on overall system performance

Note:

To view a demonstration of how HADR works, please visit:

http://www.ibm.com/software/data/db2/express/demo.html
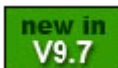
New with DB2 9.7 will be the ability to allow clients to read on the stand-by server. This 'read-on-standby' capability is expected to be available with DB2 9.7 Fixpack 1.

2.1.3 Data Replication

This feature allows for replication of data between a source server where data changes are captured, and a target server where data changes are applied. *Figure 2.1* provides an overview of how replication works.
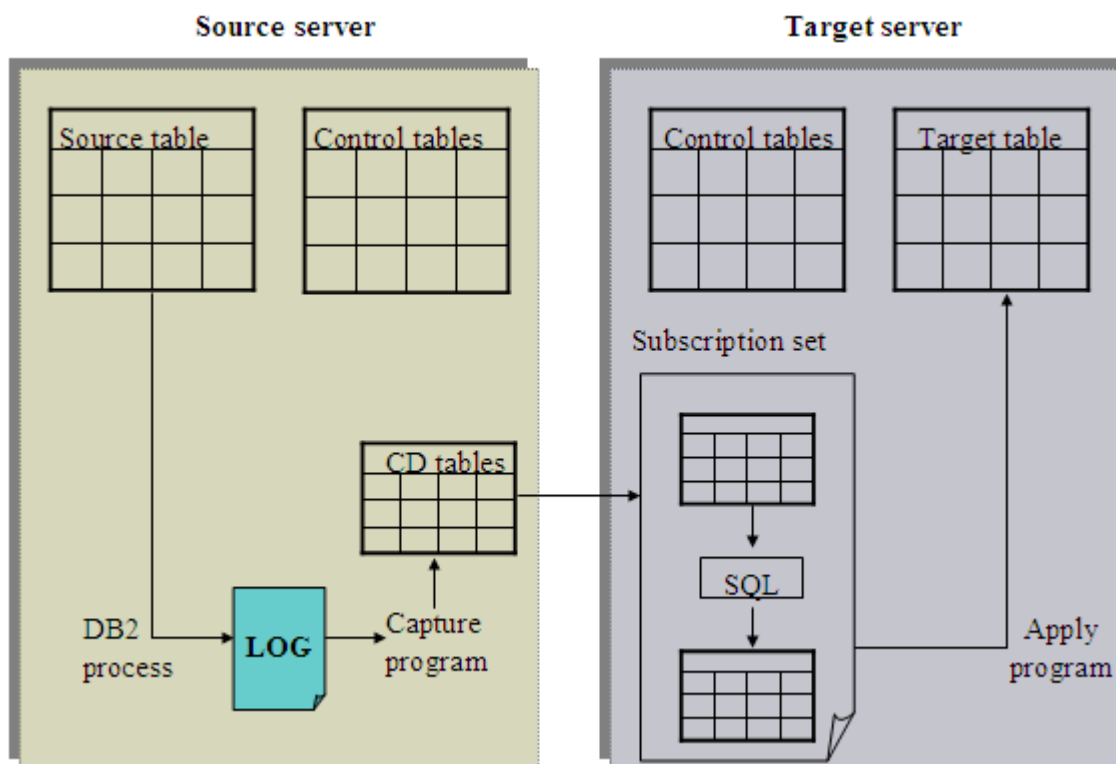
Figure 2.1 –SQL Replication

In *Figure 2.1* there are two servers, a source server and a target server. On the source server, a Capture program captures the changes made to the database. On the target server, an Apply program applies the changes to the database replica. Replication is useful for a variety of purposes that require replicated data, including capacity relief, feeding data warehouses and data marts, and auditing change history. Using the SQL replication feature you can replicate data between DB2 Express and other IBM data servers, including those on other Linux, UNIX, z/OS, and i5/OS systems.

2.2 Features not available with DB2 Express-C

This section describes some of the features available in other editions of DB2 but not in DB2 Express-C or the yearly subscription license for DB2 Express.

2.2.1 Database Partitioning

The database partitioning feature (DPF) provides distributed query processing across a cluster of database servers. It is only available with InfoSphere Warehouse Editions and allows data to be spread across multiple database partitions or nodes which can reside in several different servers. DPF is based on a shared-nothing architecture where each database partition has a subset of the overall data on its own independent disks.

Each computer, as it is added to the database cluster, brings additional data processing power with its own CPUs, memory, and disks, allowing for large tasks and complex queries to be broken down into smaller pieces and distributed across the various database nodes to be executed in parallel. This results in higher concurrency and faster response times than would be possible if the database resided on a single server. DPF is particularly useful in large data warehousing environments and business intelligence workloads involving anywhere from a few hundreds of gigabytes to several hundreds of terabytes of data.

2.2.2 Connection Concentrator

Connection concentrator is a feature that allows for support of a large number of concurrently connected users. Previously, every database connection required one database agent. The connection concentrator introduces the concept of a "logical agent", allowing one agent to handle several connections. Agents are discussed in more detail in *Chapter 6, DB2 Architecture.*

2.2.3 Geodetic Extender

DB2 Geodetic Extender is available as priced option for DB2 Enterprise Server Edition. This extender makes development for business intelligence and e-government applications that require geographical location analysis much easier. DB2 Geodetic Extender can construct a virtual globe at any scale. Most location information is collected using worldwide systems, such as global positioning satellites (GPS), and can be represented in latitude/longitude coordinates (geocode). Business data, such as addresses, can be converted to a geocode by DB2 Geodetic Extender and enterprise applications work better when they keep the data in this unprojected form, leaving map projections (earth to flat map) where they belong: in the presentation layer, to display and print maps.

2.2.4 Label-based Access Control (LBAC)

Label-based access control provides granular security at the row and column level. It uses a label that is associated with both user sessions and data rows or columns to grant access to data in your table. *Figure 2.2* illustrates how LBAC works.

	No LBAC	SEC=254	SEC=100	SEC=50	ID	SALARY
SELECT * FROM EMP WHERE SALARY >= 50000	■				255	60000
	■	■	■		100	50000
	■	■	■	■	50	70000
					50	45000
					60	30000
	■	■			250	56000
	■	■			102	82000
	■	■	■		100	54000
					75	33000
					253	46000
	■	■	■		90	83000
	■	■			200	78000

Figure 2.2 - An example of how LBAC works

In the figure, the table *EMP* has one column, *SALARY*, and an internal column *ID* containing the label for a given row. The other columns in the figure are used only for illustration purposes. If the query shown in the figure is executed, depending on the label the user has, he will be able to see different rows. The column with the title 'No LBAC' represents the rows that would be selected if LBAC was not implemented. As you can see, all the rows with a salary greater or equal to 50,000 are selected.

Now let's say the user issuing the query has a security label of 100. You can see the rows selected in this case on the third column counting from the left. In this case, DB2 will find the rows where salary is greater or equal to 50,000, and then it will review the security label for the row. For example, the first row has a salary of 60000 and a label ID of 255. Since this user has a label ID of 100 which is smaller than 255, he cannot see this row, and therefore the output from the query will not return it.

LBAC security needs to be implemented by a security administrator that has the SECADM authority.

2.2.5 Workload Manager (WLM)

The Workload Manager manages workloads across a database based on user and application priorities combined with resource availability and workload thresholds It allows you to regulate your database workload and queries so that important and high-priority queries can run promptly, and prevent 'rogue' queries from monopolizing your system resources, ensuring that your system runs efficiently. WLM has been further enhanced in DB2 9.7 and provides more powerful capabilities than Query Patroller and DB2 Governor tools available with previous versions of DB2.

2.2.6 Deep compression

DB2 supports several types of compression:

- NULL and Default Value Compression.

  This type of compression applies to columns whose values are normally NULL or the system default values such as 0, where no disk storage is consumed

- Multidimensional Clustering

  Multidimensional clustering (MDC) tables where the physical data pages are clustered in multiple dimensions. They use block indexes which in a sense is a way to compress indexes because they point to a block of records rather than to a single record.

- Database Backup Compression

  This applies to backup images. Indexes and LOB tablespaces are compressed.

- Data Row Compression

  Row compression works by replacing repeating strings within a row of data with a much smaller symbol. The mapping of this smaller symbol and the string is kept in a dictionary. Row compression can drastically improve performance on I/O bound workloads given that more rows can be brought back and forth from disk to memory (and viceversa) because the rows are smaller. You can also benefit from storage savings which normally account for one of the largest expenses in the IT budget of companies. For CPU-bound workloads there can be some extra overhead as compressed rows need to be uncompressed before processing. Note as well the the log data for from compressed records is also in compressed format.

When accessing XML and LOB columns generally DB2 will not use the bufferpool (memory), but perform direct I/Os to the disk. This is done because XML and LOBs are normally large in size; therefore bringing them to memory would cause pages that are needed to be moved from memory. With DB2 9.5 however, XML inlining for small XML documents (less than 32K) is allowed. This means that small XML documents can be stored with the base table rows, and not in a separate internal storage object known as XDA. The advantages of this approach are two-fold: First, XML documents can now be access through the bufferpool, and second, XML documents could also benefit from data row compression.
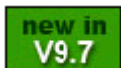
New with DB2 9.7 are further enhancements to compression:

- XDA internal objects (where XML is stored) can now also be compressed.
- Indexes and temporal tables (system and user) can be compressed
- LOBs can be inlined in a similar way to XML inlining.

2.2.7 SQL Compatibility

new in V9.7

While many vendors follow the SQL 92 and SQL/PSM standards, not all features of the standards are supported, and other features not included in the standards are. With DB2 9.7 SQL compatibility feature, DB2 can now support most PL/SQL syntax which is supported by other RDBMS vendors in addition to DB2's own SQL PL. *Figure 2.3* summarizes how this support works.
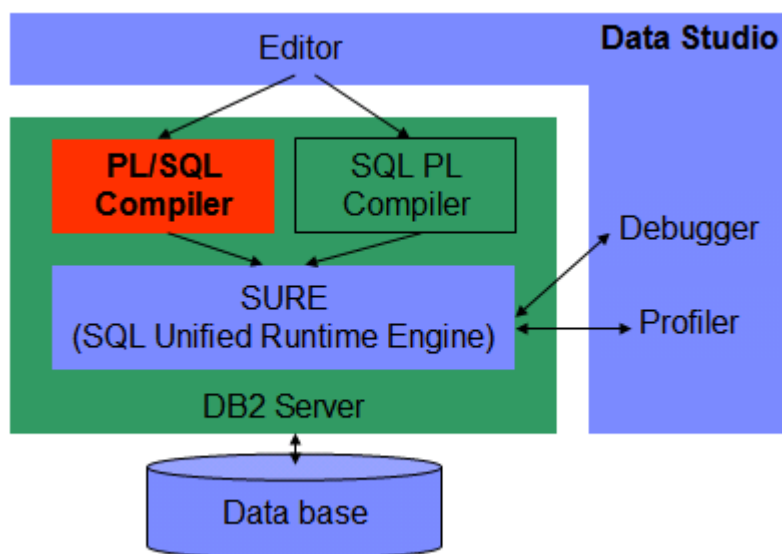
Figure 2.3 –PL/SQL support in DB2

From the figure you will note that a PL/SQL compiler has been developed and built into the DB2 engine.

SQL Compatibility feature also includes support for a tool called CLPPlus. CLPPlus is a command line tool that allows you to run SQL and other commands. It is similar to the existing DB2 Command Line Processor (CLP). *Figure 2.4* illustrates the CLPPlus tool.

Figure 2.4 –CLPPlus

Support for most PL/SQL data types has also been incorporated such as BINARY_INTEGER, RAW, and so on. Other Oracle data types such as VARCHAR2 are supported without the need for the SQL Compatibility feature, but you need to enable them using the DB2_COMPATIBILITY_VECTOR registry variable. There is more explanation about Oracle data types and this registry variable later in the book.

The SQL compatibility features outlined above are currently available with DB2 9.7 Workgroup and Enterprise editions. They are expected to be available in DB2 Express edition (including the yearly Subscription option or FTL) in the near future.

While PL/SQL support and CLPPlus features are not available in DB2 Express-C 9.7, other features included do simplify enablement of Oracle applications to DB2. These include new data types, new scalar functions, module support, and currently committed (CC) semantics for the cursor stability (CS) isolation level. These features are discussed later in the book.

2.3 Fee-based products that are related to DB2

This section provides a brief description of fee-based products and offerings that can be used with DB2.

2.3.1 DB2 Connect

DB2 Connect is fee-based software that allows a DB2 for Linux, UNIX or Windows client to connect to a DB2 for z/OS or DB2 for i5/OS server as shown in *Figure 2.5*. DB2 Connect is not required when the connection occurs in the opposite direction; when you connect from DB2 for z/OS or DB2 for i5/OS to a DB2 for Linux, UNIX or Windows server. DB2 Connect comes in two main editions depending on your connection needs: A DB2 Connect Personal Edition, and a DB2 Connect Enterprise Edition.
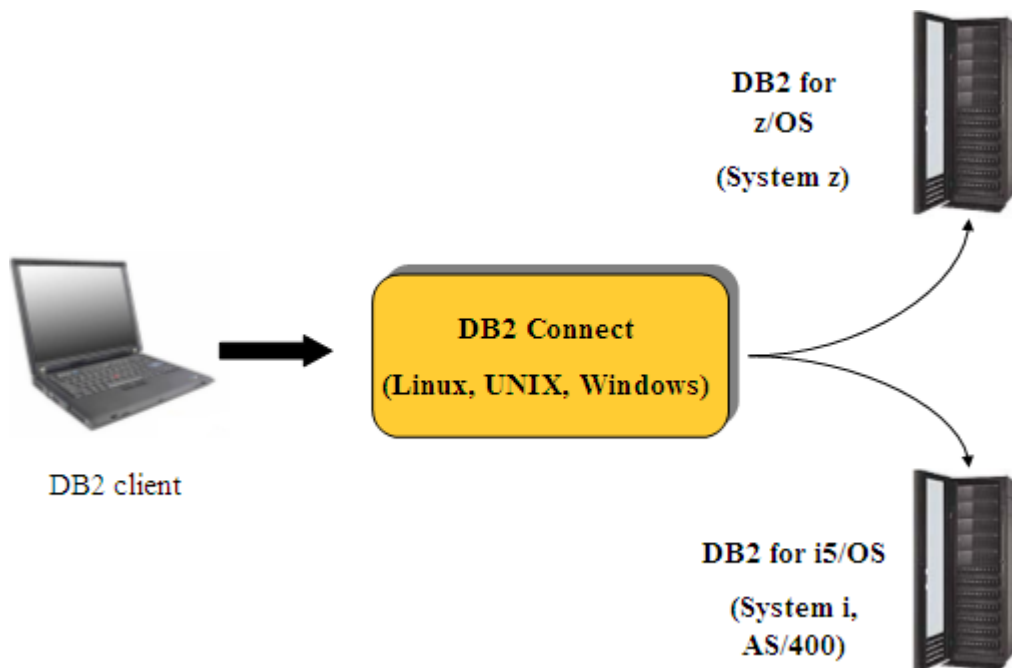
Figure 2.5 – DB2 Connect

2.3.2 InfoSphere Federation Server

Formerly known as WebSphere Information Integrator (for federation support), the WebSphere Federation Server allows for federation of databases, meaning that you can run database queries that can work with objects from different relational database systems. For example, if you buy WebSphere Federation Server, you can run the query shown in *Listing 2.1* below.

```
SELECT   *
FROM      Oracle.Table1  A
             DB2.Table2  B
      SQLServer.Table3  C
WHERE
          A.col1 < 100
       and B.col5 = 1000
       and C.col2 = 'Test'
```
Listing 2.1 - A federated query

Figure 2.6 provides an illustration where WebSphere Federation Server is used.
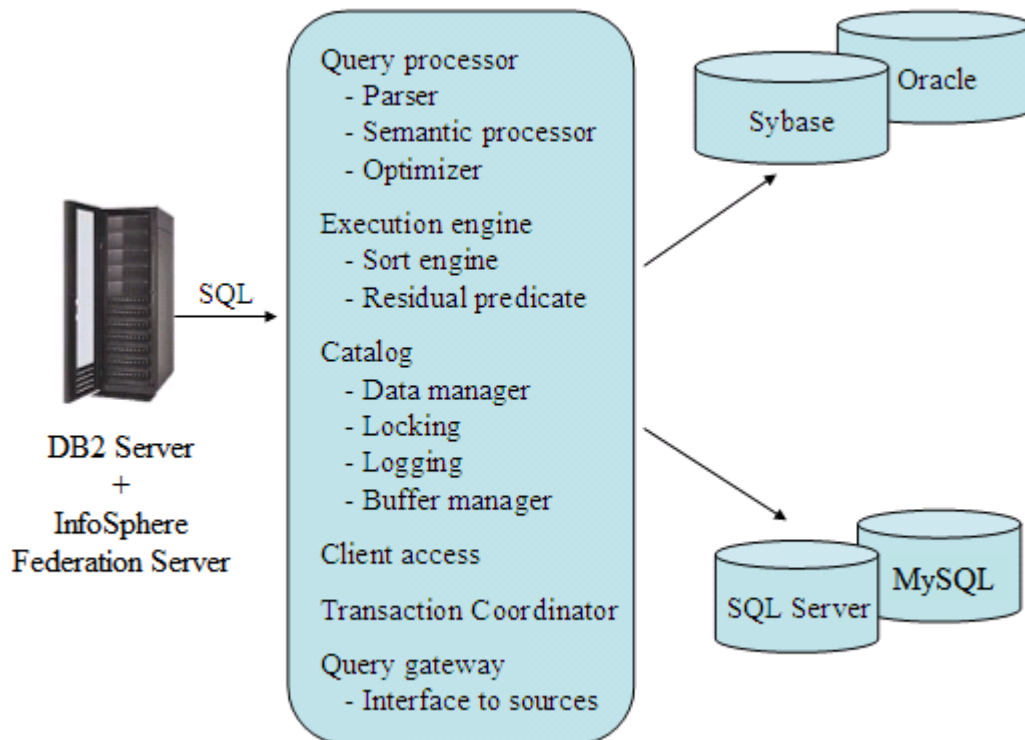
Figure 2.6 – InfoSphere Federation Server

For relational database management systems that are part of the IBM family, federation support is built into DB2 Express-C. This means that the WebSphere Federation Server product is not required when, for example, you want to run a query between two different DB2 databases, or between one DB2 database and an Informix database (Informix is part of the IBM family).

2.3.3 InfoSphere Replication Server

Formerly known as WebSphere Information Integrator (for replication support), the InfoSphere Replication Server allows for SQL replication of database records when non-IBM data servers are involved. It also includes a feature known as Q-Replication for replicating data using message queues.

2.3.4 Optim Development Studio (ODS)

Formerly known as Data Studio Developer, ODS is an Eclipse-based tool that can be easily integrated with Data Studio, and share the same Eclipse. ODS can help you create development databases using copy and paste from existing Oracle or DB2 databases.

2.3.5 Optim Database Administrator (ODA)

Formerly known as Data Studio Administrator, ODA is an Eclipse-based tool that can be easily integrated with Data Studio and share the same Eclipse. ODA provides a change management capability and the ability to automate schema changes more easily.

2.4 DB2 Offerings on Amazon Elastic Compute Cloud

It is noteworthy to mention that IBM has entered into a partnership with Amazon Web Services (AWS) for running DB2 on Amazon's Elastic Compute Cloud (EC2). AWS delivers a set of integrated services that form a computing platform "in the cloud", and is available on a pay-as-you-go model. That is, AWS lets you 'rent' compute capacity (virtual servers and storage), and you only pay for the capacity that you utilize. For example, let's say you provision one EC2 virtual server for normal database operations, and during peak times or for seasonal needs you provision an extra database server for a few hours. In this example you would pay AWS only for the extra database server only for the few hours that you have it running.

IBM offers three different deployment options for DB2 on Amazon's cloud platform:

- DB2 Express-C AMIs for evaluation and development

- Pay-as-you-go Production-ready AMIs with DB2 Express and DB2 Workgroup

- Ability to create your own AMIs using DB2 licenses you own

For more information and how to get started with DB2 on Amazon EC2, please visit: ibm.com/db2/cloud

2.5 Summary

DB2 Express-C provides no-charge, easy to use, and robust foundation for developing database applications, deploying them into production, and even embedding and distributing the applications with third party solutions. It is ideal if you are comfortable with community-based assistance and do not have a need for latest fixes or advanced features. If however you require formal technical support from IBM, regular software updates (fixpacks), or additional resource utilization, and high availability clustering support, IBM offers a DB2 Express subscription license (FTL) for a low yearly fee. If you require more advanced features for mission-critical workloads and massive-scale database applications, IBM offers more scalable DB2 editions and related products. This allows you to start small with DB2 Express-C, and yet scale to new heights as your business demands.

3

Chapter 3 – DB2 installation

Installing DB2 is fairly straightforward, and in a typical installation, simply choosing the default options will have a DB2 server up and running in a short time.

First download the appropriate DB2 Express-C image for your platform from the DB2 Express-C Web site (www.ibm.com/db2/express).

3.1 Installation prerequisites

DB2 Express-C is available on Linux®, Sun Solaris (x64), and Microsoft Windows® 2003, XP, Vista and Windows 7. It is also available as a beta on Mac OS X. The processor architectures available are 32-bit, 64-bit and PowerPC (Linux). If you need to run DB2 on another platform (such as UNIX), you should purchase one of the different data server editions described earlier in this book. Operating system requirements for all DB2 editions are also described in this document: www.ibm.com/software/data/db2/udb/sysreqs.html

In terms of hardware resources, DB2 Express-C can be installed on systems with any number of CPU cores and memory, however, it will only utilize up to 2 cores and 2GB of memory for the free unwarranted license version, and up to 4 cores and 4 GB of memory for the paid subscription version of DB2 Express. The systems can be physical systems, or virtual systems created by partitioning or running virtual machine software. You can of course run on smaller systems if you prefer, for example a single processor system with 1GB of memory.

For the latest information on DB2 Express-C hardware requirements, review the DB2 Express-C web page at www.ibm.com/software/data/db2/express/about.html

3.2 Operating system installation authority

To install DB2 Express-C on Linux or Windows, you must have an operating system user with sufficient authority.

For **Linux**, you need to be root (the superuser) to install DB2 Express-C. You can also install DB2 Express-C as a non-root user; however, you will be limited in what you can do with the product. For example, under a non-root installation, you cannot create more instances than the default one created at installation time.

For **Windows**, the user account must belong to the Administrators group on the machine where you will perform the installation. Alternatively, on Windows 2008, Windows Vista, or higher, a non-administrator can perform an installation, but will be prompted for administrative credentials by the DB2 Setup wizard.

The installation user ID must belong to the Domain Administrators group on the domain if the installation requires a domain account to be created or verified.

You may also use the built-in Local System account to run the installation, though it is not recommended. A Local System account does not require a password, but it cannot access network resources.

The user account must also have the user right to "Access this computer from the network".

Note:

See a video presentation about DB2 Express-C installation at this link. Although this presentation demonstrates DB2 9.5, it is no different from a DB2 9.7 installation, other than the color of the installation panels:
http://www.channeldb2.com/video/video/show?id=807741:Video:4442

3.3 Installation wizard

Although there are several methods to install DB2 Express-C, the easiest method is to use the GUI-based DB2 Installation wizard. After downloading and uncompressing the DB2 Express-C image, you can launch the wizard as follows:

- Windows: Execute the `setup.exe` file located in the EXP/image/ directory
- Linux: Execute the `db2setup` command located in the exp/disk1/ directory

DB2 Express-C is straightforward to install by following the instructions in the DB2 installation wizard. In most cases, the default settings are sufficient, so all you need to do is accept the license, click the *Next* button until the "Finish" button is active, and then click the *Finish* button. After a few minutes, your installation is complete and DB2 will be up and running!

Figure 3.1 shows the DB2 Setup Launchpad. Click on *Install a Product* and then choose *Install New* to install a new copy of DB2 Express-C on your system. If you have previously installed DB2 Express-C or another DB2 edition, you may see a button named "*Work with Existing".* In DB2, you are allowed to install the product several times, and those installations can be at different version or release levels.
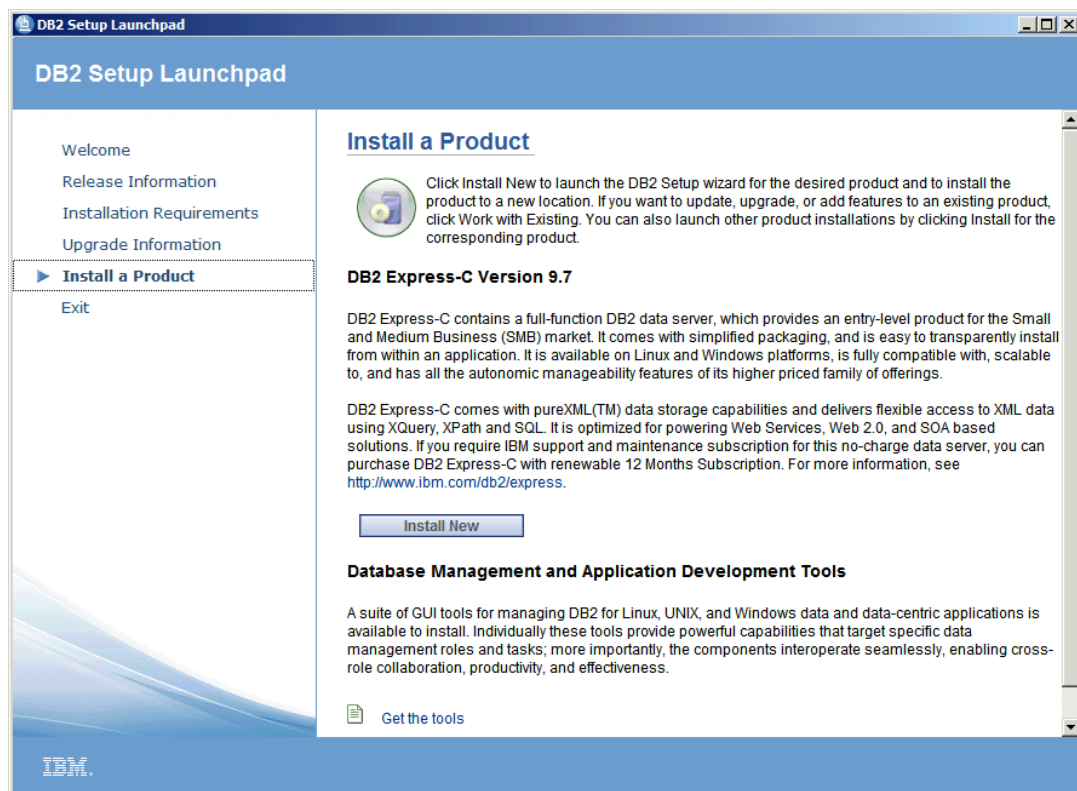
Figure 3.1 – The DB2 Setup Launchpad

After accepting the license, it is usually sufficient to choose the "*Typical*" installation (default) as shown in *Figure 3.2*. If you would like to include the DB2 Text Search component, choose "*Custom*".
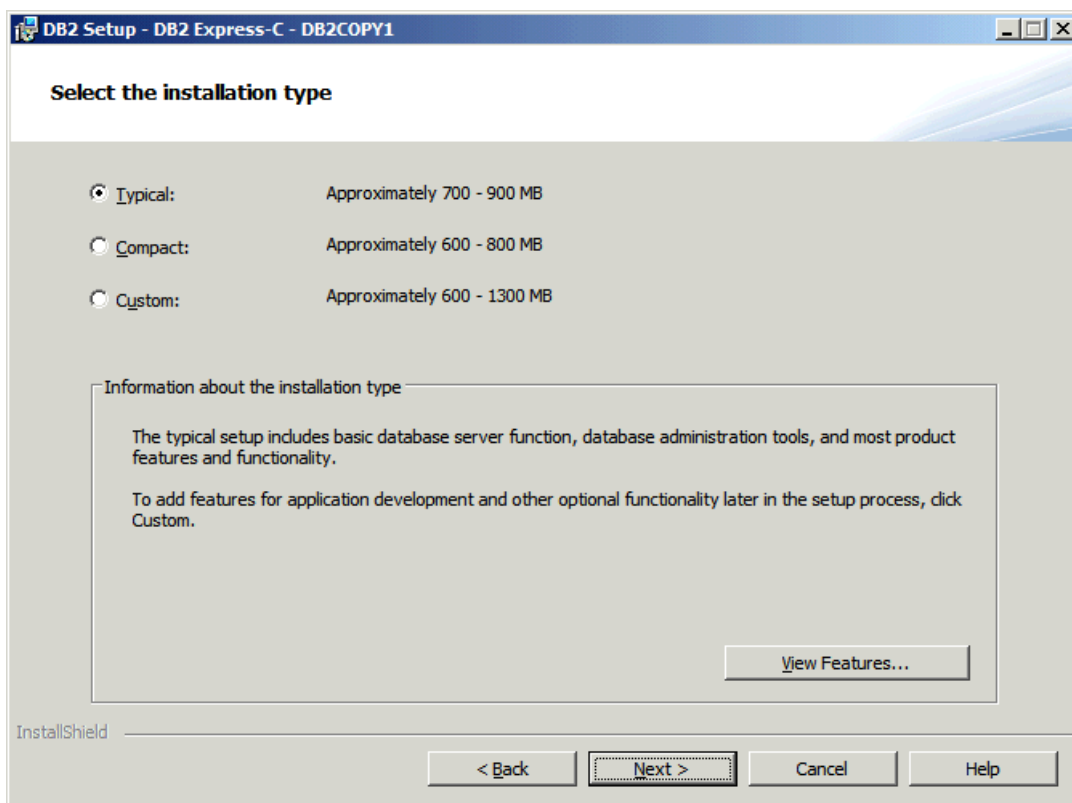
Figure 3.2 – Installation types

In the next step, as shown in *Figure 3.3*, you can choose to install the product, create a response file, or both. Response files are discussed in section *3.4, Silent Install.* Choosing the default ("*Install IBM DB2 Express Edition on this computer and save my settings in a response file*") is the default.
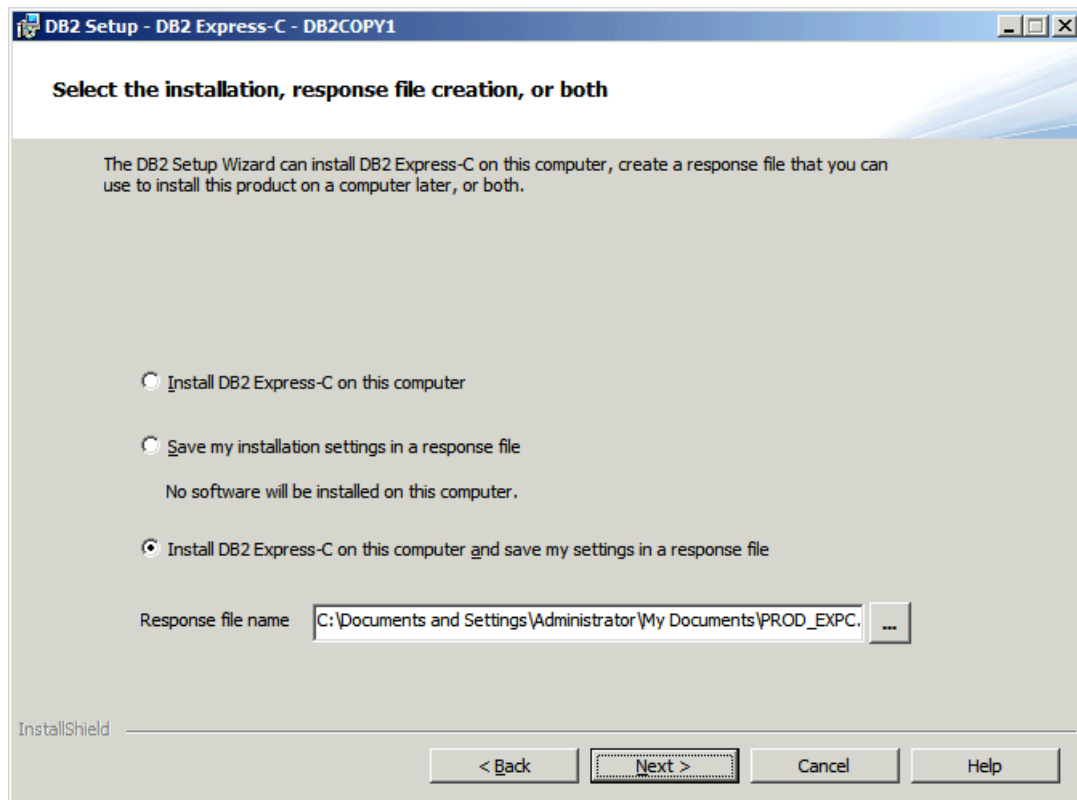
Figure 3.3 – Selecting the installation

Choose the default values for the next few screens. When you arrive at the panel shown in *Figure 3.4*, you will need to type in a user ID which will be used to setup and run the instance and other services.

If you use an existing user ID, this user must be part of the Local Administrator group in Windows.

If the user ID does not belong to an existing user, it will be created as a Local Administrator. You can leave the domain field blank if the user ID does not belong to a domain.

The default user ID name in Windows is called `db2admin`. In the case of Linux, the default user ID created is called `db2inst1`.

Figure 3.4 – Specifying user information for the DB2 Administration Server

Finally, in *Figure 3.5*, the installation wizard displays a summary of what will be installed, and the different configuration values provided in the previous steps. When you click the "*Finish*" button, the installation begins, and the program files will be laid down on your system.
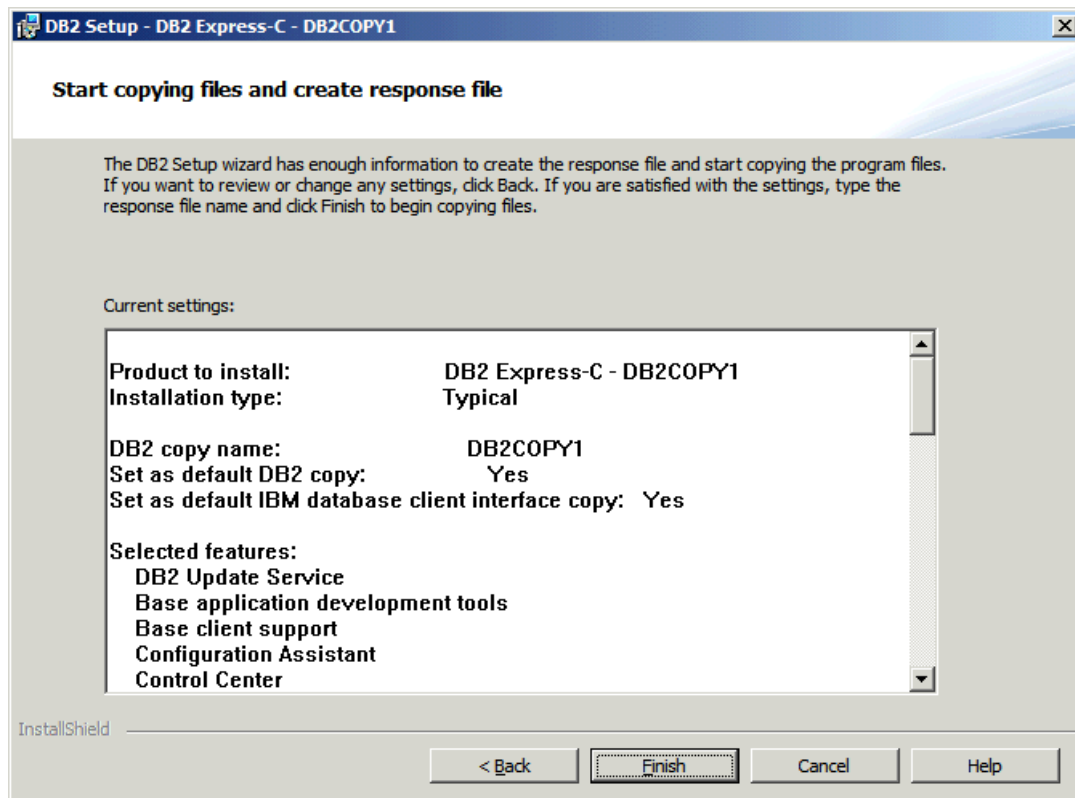
Figure 3.5 – Summary of what will be installed

When the installation completes, a window similar to the one in *Figure 3.6* appears, informing you of the results of the installation wizard process, as well as any further steps that are required to complete your installation.
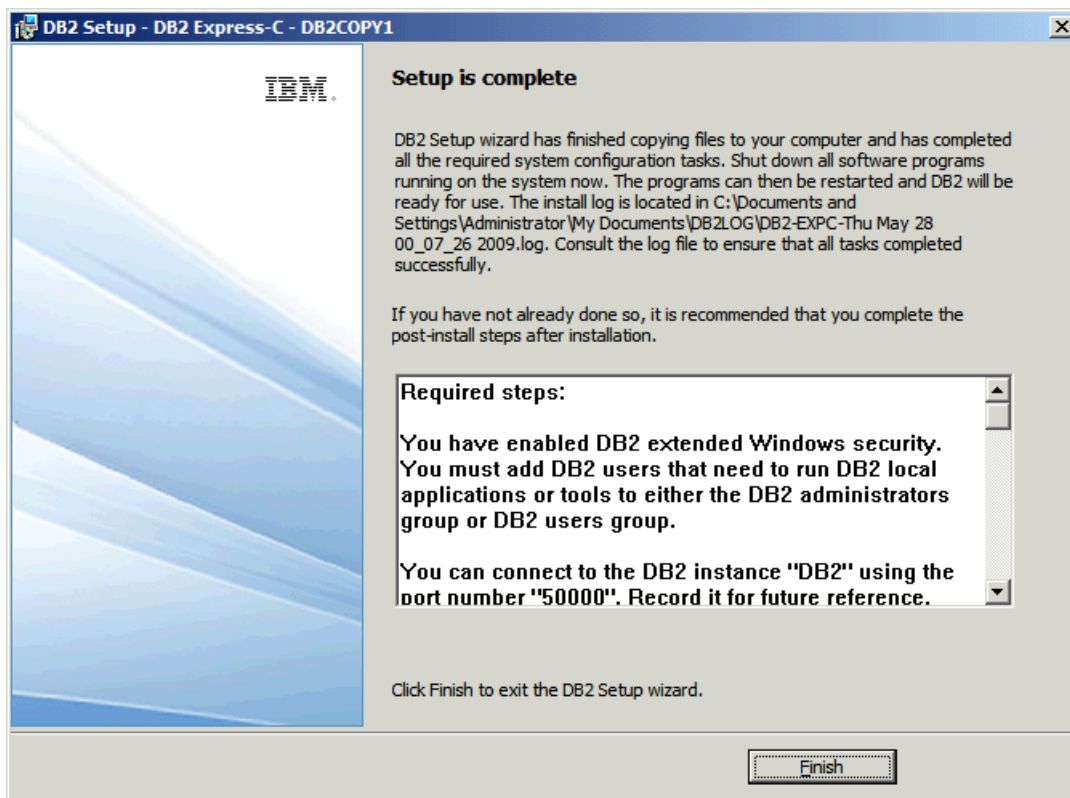
Figure 3.6 – The installation is complete

After you click *"Finish"* in the installation results window shown in *Figure 3.6*, the *DB2 First Steps* application is launched, as shown in *Figure 3.7*.

This small application outlines several different options to get you started with DB2, such as creating the default sample database (appropriately named **SAMPLE**) or to create a new database of your own. If you don't want explore DB2 through *First Steps* at this time, you can close the window, and invoke it at a later time.

To manually start *DB2 First Steps* on Windows, choose *Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> Set-up Tools -> First Steps* or run the **db2fs** command from the command prompt.

On Linux, execute the **db2fs** command from a terminal window.
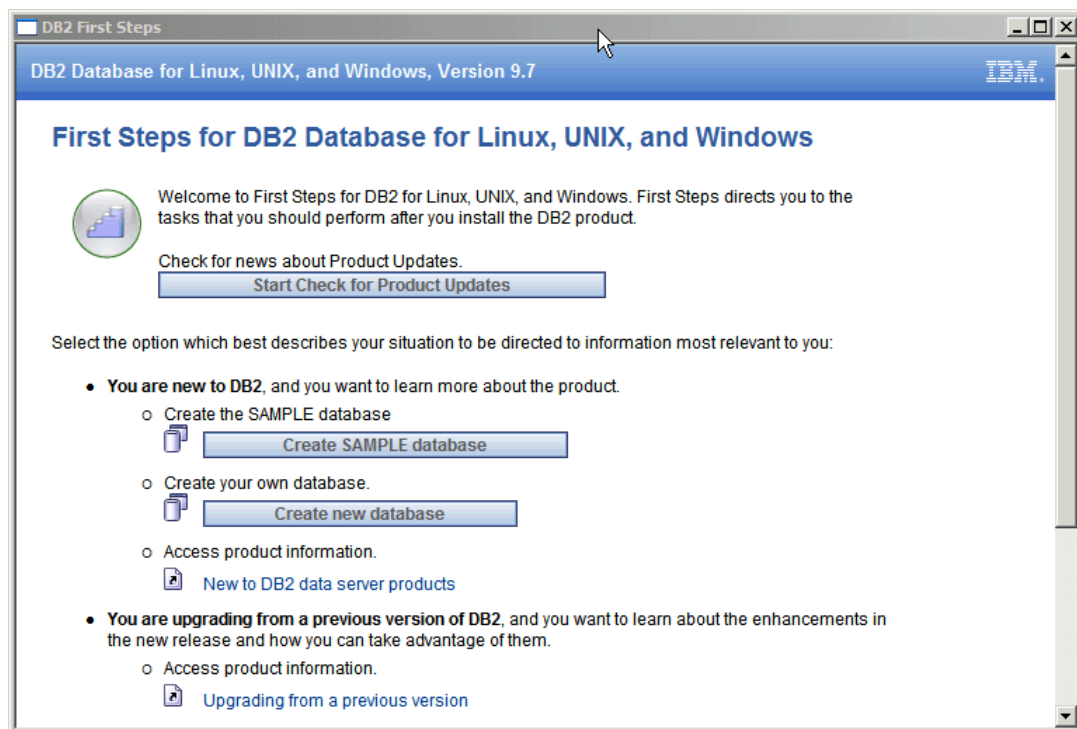
Figure 3.7 – First Steps

3.4 Validating your installation

After installing DB2, you can run three commands from the DB2 Command Window (on Windows) or from the terminal (on Linux) to verify that your installation is in good shape:

- **db2level**: This command displays information about the DB2 product installed, fix pack level, and other details.

- **db2licm -l**: This command lists all the licensing information specific to your installed DB2 products

- **db2val**: This is a new command available in DB2 9.7. It validates your installation by verifying the core functionality of your DB2 copy. It makes sure your instances are consistent, and that database creation and database connections work.

Figure 3.8 below provides an example of the output of these three commands.

Figure 3.8 – The db2level, db2licm, and db2val commands to validate your installation

In the figure, the **db2level** command output indicates you are running DB2 9.7 (DB2 v9.7.0.441) at Fix Pack 0, which means your DB2 code is at the base (GA) level with no fixes applied. The **db2licm -l** command indicates you have installed DB2 Express-C edition which has a permanent and unwarranted license that is allowed to use up to 2 cores, and up to 2GB of memory. The **db2val** command output is self explanatory.

Note:

If you would like to validate the consistency of a database at any time, use the INSPECT utility.

3.5 Silent Install

There may be situations where you need to install a DB2 client on multiple computers; or you need to embed a DB2 data server as part of your application, and would like to install it as part of your application installation process. In these situations, a silent install is the ideal way to install DB2.

DB2 enables silent installs through the use of response files which store installation information as simple text options. *Listing 3.1* shows a snippet of a sample response file.

```
PROD=UDB_EXPRESS_EDITION
LIC_AGREEMENT=ACCEPT
FILE=C:\Program Files\IBM\SQLLIB\
INSTALL_TYPE=TYPICAL
LANG=EN
INSTANCE=DB2
DB2.NAME=DB2
DEFAULT_INSTANCE=DB2
DB2.SVCENAME=db2c_DB2
DB2.DB2COMM=TCPIP
...
```
Listing 3.1 – A sample response file

There are a number of ways to generate a response file:

- Install DB2 Express-C once on a computer using the DB2 Installation wizard. One of the first wizard options (as shown in *Figure 3.3)* allows you to select the checkbox to save your install responses to a response file. At the end of the installation, the wizard generates a response file into a specified directory and filename. This is a text file, so you can manually edit it afterwards.

- Edit the sample response file provided with the DB2 Express-C image. This sample file (denoted with a .rsp file extension) is located in the `db2/platform/samples/` directory

- For Windows, you can also use the response file generator command:

  db2rspgn –d <output directory>

Finally, to silently install DB2 using a response file, issue the following command on Windows:

setup -u <response filename>

On Linux, issue the command:

db2setup -r <response filename>

3.6 Summary

This chapter covered the details of installing DB2 Express-C. This DB2 edition is available on Linux, Solaris and most varieties of Windows, and can run on 32-bit, 64-bit and Power PC architectures. After discussing the required user authority necessary to install DB2 on a system, we did a walkthrough of a straightforward installation using the DB2 Installation Wizard GUI. A discussion of post-install activities followed, including running DB2 First Steps and validating the install. Finally, we took a look at how to create and execute a silent install using DB2 response files.

3.7 Exercises

In this exercise, you will install DB2 Express-C and create the SAMPLE database

Objective

Before you can begin exploring all the features and tools that come with DB2 Express-C, you must first install it on your system. In this exercise, you will perform a basic installation of DB2 Express-C on Windows. The same installation wizard is available on Linux; therefore the steps are very similar on that platform.

Procedure

1. Obtain DB2 Express-C images: Download the appropriate DB2 Express-C image from the DB2 Express-C Web site (www.ibm.com/db2/express).   Unzip the files into any directory you wish.

2. Locate the files: Navigate to the directory (or drive) containing the unzipped DB2 product installation files.

3. Run the DB2 Setup Launchpad: Launch the DB2 Setup Launchpad by double-clicking on the `setup.exe` file. On Linux, run the `db2setup` command as root. From the Launchpad, click the *Install Product* option on the left pane of the window.

4. Run the DB2 setup wizard: The DB2 setup wizard checks that all system requirements are met and sees if there are any existing DB2 installations. Click on *Install New* to start the wizard, then click *Next.* .

5. Review the license agreement: Read and accept the license agreement (select the "*I Accept...*" radio button) and click the *Next* button to continue.

6. Choose the installation type: For this exercise, select the *Typical* option (this is the default). The Compact option performs a basic installation, while the Custom option allows you to customize the specific features you want to install. Click the *Next* button to continue.

7. Select the installation, response file creation, or both: Leave the default so that DB2 is installed, and also a response file is created. Click the *Next* button to continue.

8. Select the installation folder: This screen allows you to customize the drive and directory where the DB2 code is installed on your system. Ensure sufficient space exists for the installation. Use the default drive and directory settings for this example (shown below):

```
Drive:  C:

Directory:  C:\Program Files\IBM\SQLLIB
```

Click the *Next* button to continue.

9. Set the user information: Once DB2 Express-C is installed, certain DB2 processes are run as system services. These services require an operating system account in order to run. In the Windows environment, using the default **db2admin** user account is recommended. If the user account does not yet exist, DB2 creates it in the operating system for you. You can also specify to use an existing account, but that account must have local administrator authority. We recommend using the defaults suggested. Ensure you specify a password for the account. On Linux use the default **db2inst1** user ID for the instance owner, **db2fenc1** for the fenced user and **dasusr1** for the DB2 Administration server user. Click the *Next* button to continue.

10. Configure the DB2 instance: A DB2 instance can be thought of as a container for databases. An instance must exist before a database can be created inside it. During a Windows installation, an instance called **DB2** is automatically created. In a Linux environment, the default instance name is **db2inst1**. We will cover instances later in this book.

By default, the DB2 instance is configured to listen for TCP/IP connections on port 50000. Both the default protocol and the port can be changed by clicking the *Configure* button. We recommend using the default settings in this example. Click the *Next* button to continue.

11. Start the installation:. Review the installation summary options previously selected. Click the *Finish* button to begin copying the files to the installation location. DB2 will also perform some initial configuration processes.

12. First Steps. After the installation is complete, another launch utility, called First Steps, is displayed. First Steps can also be started later with the command **db2fs**.

13. The **SAMPLE** database is a database that you can use for test purposes. It can be created from First Steps by clicking the *Create SAMPLE database* button. Click on this button, and the window shown below appears. Choose the second option (*XML and SQL objects and data*). The SAMPLE database can also be created using the command **db2sampl -xml -sql**.
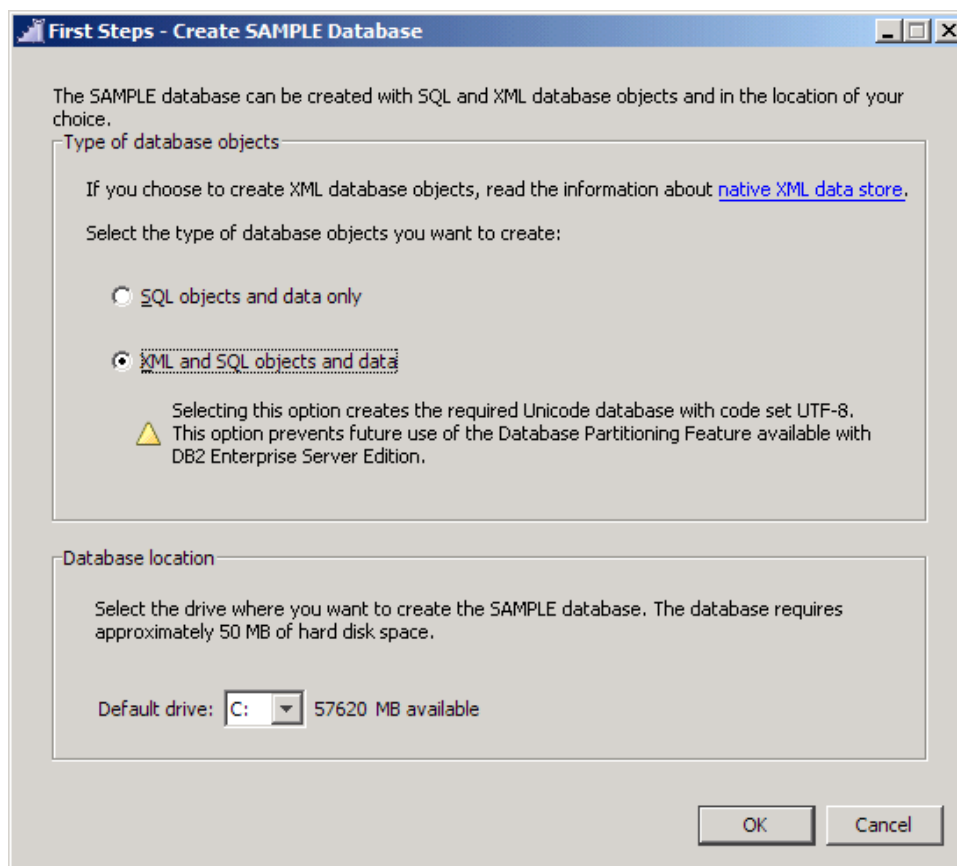
14. After a few minutes, you can verify the database was created. Open the DB2 Control Center tool choosing: *Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> General Administration Tools -> Control Center* .You can also start the Control Center with the command `db2cc`. The first time you start the Control Center, a pop-up window will ask you to choose which Control Center view you want to use. Leave the default (Advanced), and click *OK*. On the left panel, drill down the *All Databases* folder. If you cannot see the SAMPLE database in that folder, make sure you refresh your view by choosing *View -> Refresh*

15. Restart the computer. This is step is optional. Although this step is not mentioned in the official DB2 installation documentation, we recommend rebooting the system (if possible, at least on Windows) to ensure all processes start successfully and to clean up any memory resources that might not have been cleaned up correctly.

16. Validate your DB2 installation by running the commands: `db2level, db2licm,` and `db2val.` From the Windows Start Menu, open the DB2 Command Window as follows: *Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Window.* From the Command Window (Or the shell in Linux) type `db2level` and examine the output. Do the same for the command `db2licm`

-l. Next, issue the `db2val` command. If `db2val` finishes successfully, your installation is in good shape!. If there are errors, review the log file specified in the error message for more details. The output of the three commands should be similar to the ones shown in *Figure 3.8* earlier.

4

Chapter 4 – DB2 Environment

The DB2 environment includes different database objects and configuration files. *Figure 4.1* provides an overview of the different commands and tools to work with DB2, and it also highlights the DB2 environment on the right side. This is the area of focus for this chapter. The left side of the figure shows the different DB2 commands, and SQL, SQL/XML, and XQuery statements that can be issued to interact with a DB2 data server. The middle of the figure shows the names of the different tools used to interact with a DB2 data server. .
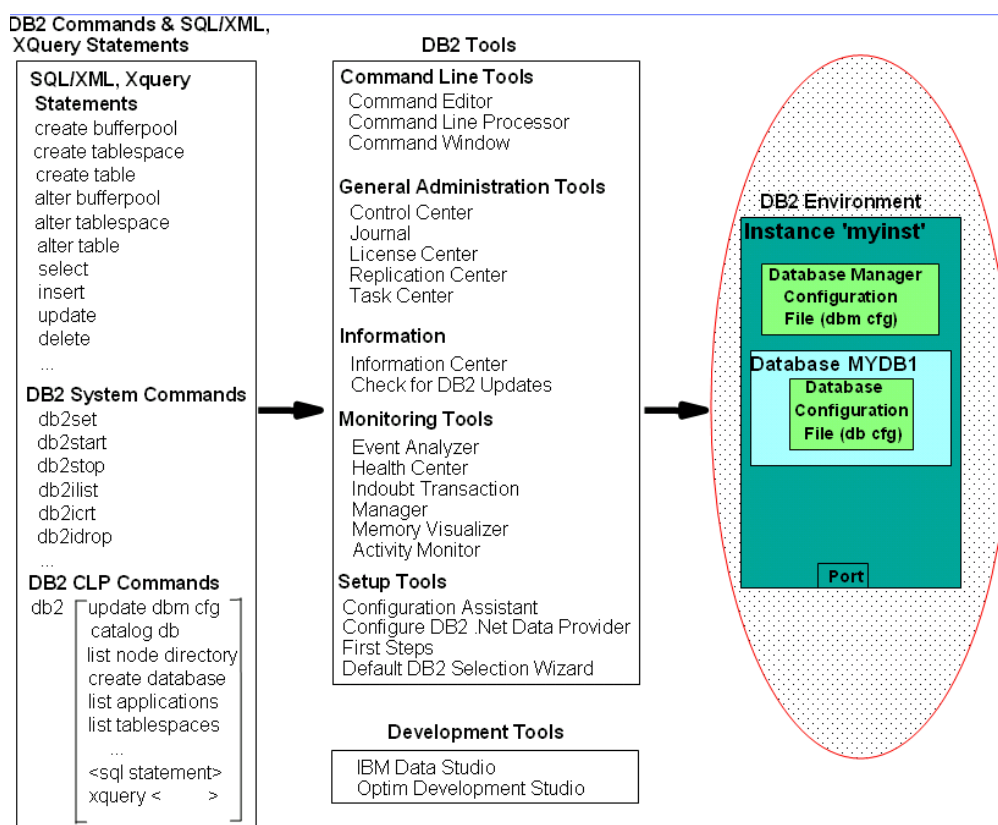
Figure 4.1 – The DB2 big picture: DB2 environment

Note:

See video presentations about the DB2 Environment at these links:
http://www.channeldb2.com/video/video/show?id=807741:Video:4029
http://www.channeldb2.com/video/video/show?id=807741:Video:4042

To describe the DB2 environment, let's describe each component element step by step. *Figure 4.2* shows a representation of a DB2 data server after installing DB2 Express-C 9.7.
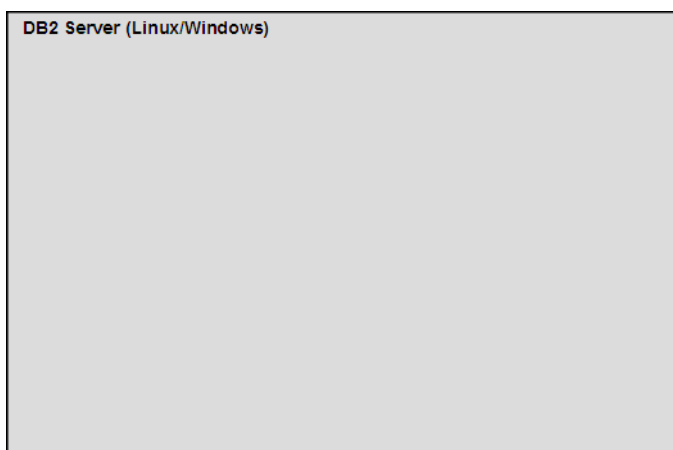
Figure 4.2 – Representation of a DB2 Server after installing DB2 Express-C 9.7

As part of the installation in Windows, a default instance called **DB2** (**db2inst1** on Linux) is created. This is represented by the green box on the left of *Figure 4.3*. An instance is simply an independent environment where applications can run, and databases can be created. You can create multiple instances on a data server, and use them for different purposes. For example, one instance can be used to hold databases for production use, another instance can be used for test environment databases, and another one for a development environment. All of these instances are independent; that is, performing operations on one instance will not affect the other instances.
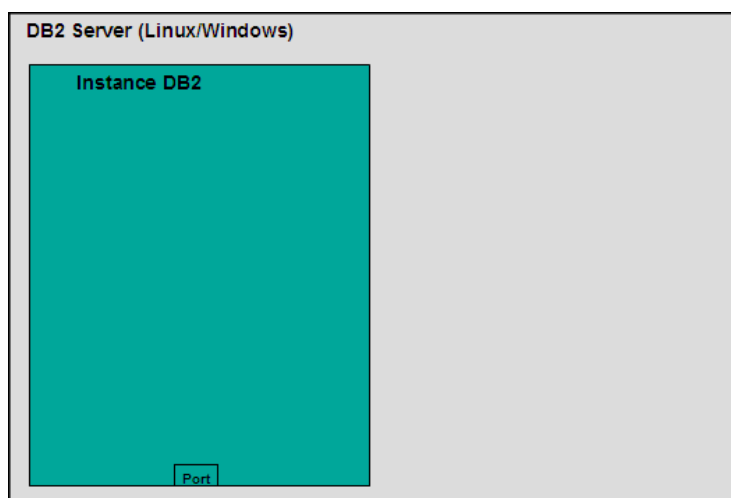
Figure 4.3 – The default DB2 instance created

To create a new DB2 instance, use the command **db2icrt <instance name>**, where **<instance name>** is replaced with any 8 character name. For example, to create the instance *myinst*, we use this command: **db2icrt myinst**.

Figure 4.4 shows a new instance called *myinst* as a separate green box on the right side.
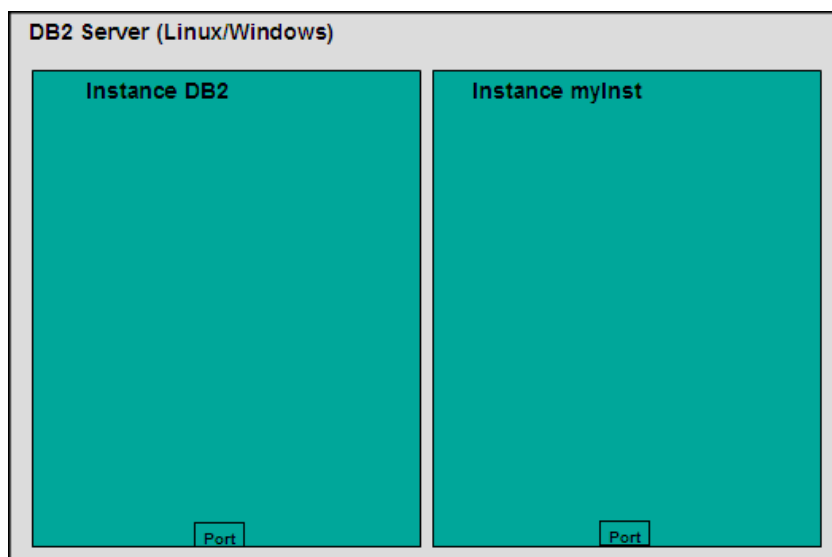
Figure 4.4 – A DB2 server with two instances

Note that each instance has a unique port number. This helps to distinguish between instances when you want to connect to a database in a given instance from a remote client using TCP/IP. If you use the DB2 Command Window, you can make any DB2 instance the active one by using this operating system command on Windows:

set db2instance=myinst

Note that there should not be any blank spaces before or after the equal (=) sign. In this example, if you now create a database from the Command Window, it would be created in the instance **myinst**.

To list the instances in your system, run the command:

db2ilist

On Linux, an instance must match a Linux operating system user; therefore, to switch between instances you can simply switch users. This user is known as the instance owner. You can logoff and logon with the instance owner user of the other instance, or use the **su** command.

Table 4.1 shows some useful instance level commands.

Command	Description
db2start	Starts the current instance
db2stop	Stops the current instance
db2icrt	Creates a new instance
db2idrop	Drops an instance
db2ilist	Lists the instances you have on your system
db2 get instance	Lists the current active instance

Table 4.1 – Useful DB2 commands at the instance level

Some of the above commands can instead be performed via the Control Center. For example, in the Control Center, if you expand the *Instances* folder and right-click the desired instance, you can choose *Start*, which is equivalent to issuing the **db2start** command from the DB2 Command Window, or *Stop*, which is equivalent to issuing a **db2stop** command as shown in *Figure 4.5*.
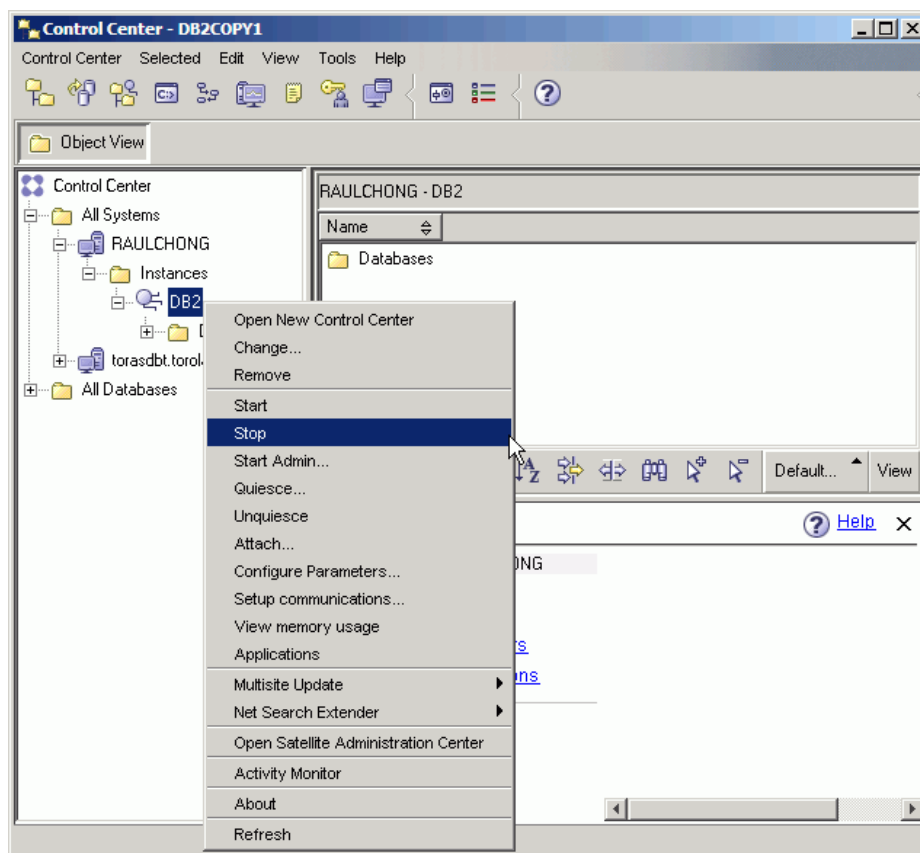
Figure 4.5 – Instance commands from the Control Center

To create a database in the active instance, issue this command from the DB2 Command Window:

```
db2 create database mydb1
```

To list all the databases created, run the command:

```
db2 list db directory
```

Within any one instance, you can create many databases. A database is a collection of objects such as tables, views, indexes, and so on. Databases are independent units, and therefore, do not share objects with other databases. *Figure 4.6* shows a representation of a database *MYDB1* created inside instance **DB2**.
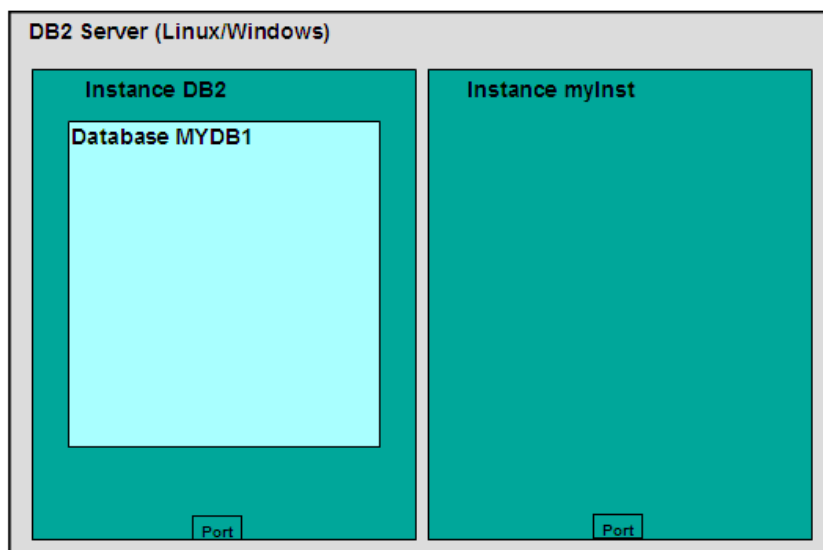
Figure 4.6 – Database MYDB1 created in instance DB2

Table 4.2 shows some commands you can use at the database level.

Command/SQL statement	Description
`db2 create database`	Creates a new database
`db2 drop database`	Drops a database
`db2 connect to <database_name>`	Connects to a database
`db2 create table/create view/create index`	SQL statements to create table, views, and indexes respectively

Table 4.2 - Commands/SQL Statements at the database level

If we want to create another database with the same name (*MYDB1*) but in instance *myinst*, the following commands from the DB2 Command Window would be issued:

```
db2 list db directory
set db2instance=myinst
db2 create database mydb1
set db2instance=db2
```

Figure 4.7 shows the new database *MYDB1* created in instance *myinst*.
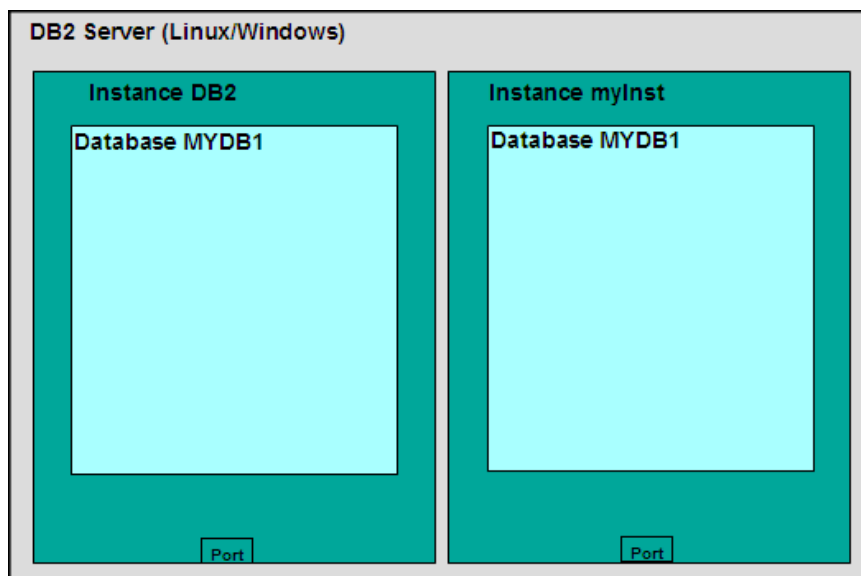
Figure 4.7 – Database MYDB1 created in instance myInst

When a database is created, there are several objects created by default: table spaces, tables, a buffer pool and log files. Creating these objects takes a bit of time, that's why the **create database** command requires a few minutes for processing. *Figure 4.8* shows three table spaces created by default on the left side of the figure. Table spaces will be discussed in more detail in *Chapter 6, DB2 Architecture*; but for now, think of table spaces as a logical layer between logical tables and physical resources, such as disks and memory.
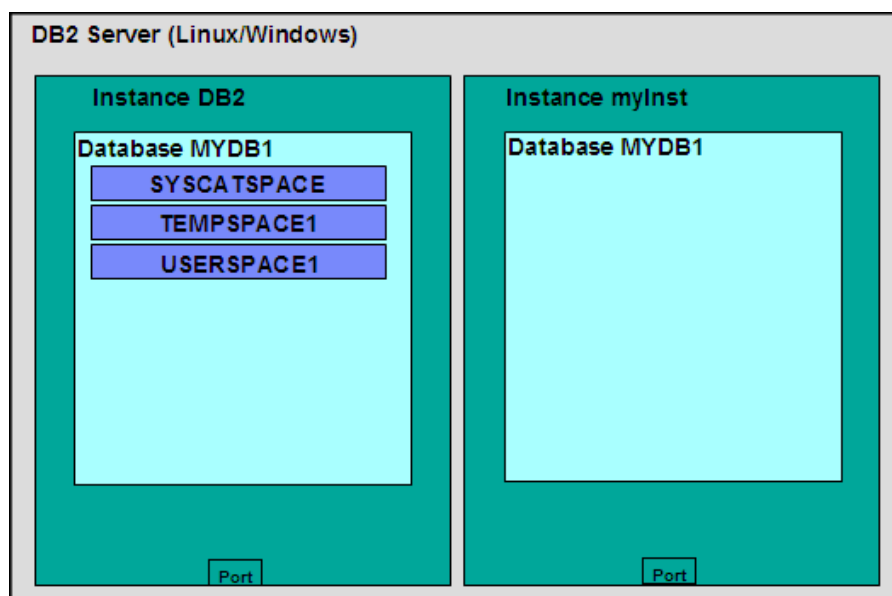
Figure 4.8 –Table spaces created by default when a database is created

Table space **SYSCATSPACE** contains the System Catalog tables. The System Catalog is called the data dictionary in other relational database management systems. It basically contains system information that should not be modified or deleted; otherwise the database will not work correctly. Table space **TEMPSPACE1** is used by DB2 when it needs additional space to perform some operations such as sorts. Table space **USERSPACE1** is normally used to store user database tables if there is no table space specified when creating a table.

You can also create your own table spaces using the **CREATE TABLESPACE** statement. *Figure 4.9* shows the table space *MYTBLS1* created inside database *MYDB1* on instance **DB2**. When you create a table space, you specify the disks to use and the memory (buffer pool) to use. Therefore, if you have a "hot" table, that is, a table that is used very often, you can allocate the fastest disks and the most memory by assigning a table space with these characteristics.

In *Figure 4.9*, we show two other objects created by default: A buffer pool called **IBMDEFAULTBP**, and the log files.

A buffer pool is basically a memory cache used by the database. You can create one or more buffer pools, but there should always be one buffer pool with a page size that matches the page size of existing table spaces. Pages and page size will be discussed in more detail in *Chapter 6, DB2 Architecture*.

The log files are used for recovery. When you work on a database, not only is information stored in the disks for the database, but while you are working on the database, log files store all the operations executed on the data. Think of logs as temporary files where an "autosave" operation is performed. Logs are discussed in more detail in *Chapter 11, Backup and Recovery*.
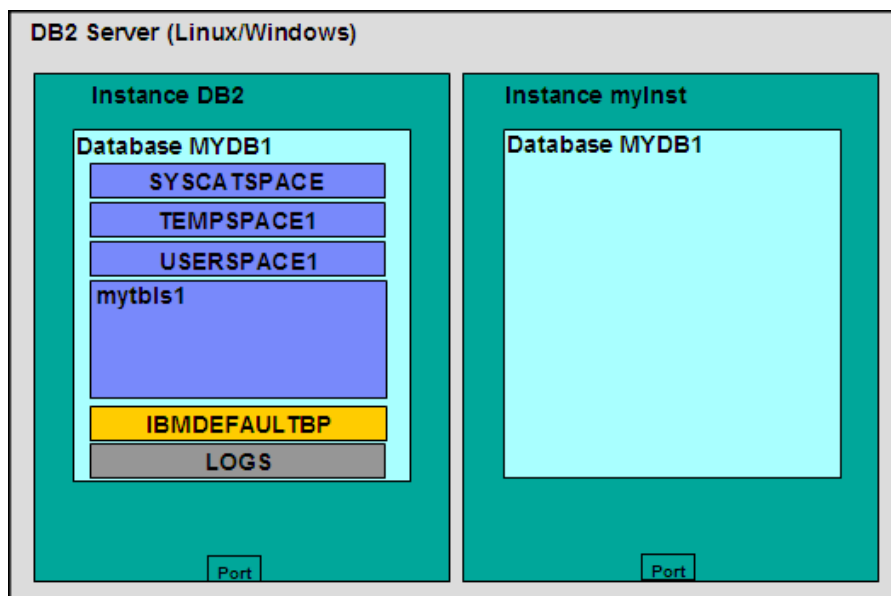
Figure 4.9 – Buffer pool and logs created by default

Earlier we discussed that instances are independent environments, and therefore, a database with the same name could be created in several instances. Just like instances, databases are also independent units; therefore, objects in one database have no relationship to objects in another database. *Figure 4.10* shows a table space *mytbls1* inside both the **MYDB1** database and the **SAMPLE** database, within instance **DB2**. This is valid because the databases are independent units. Note that *Figure 4.10* does not show the other default objects of database **SAMPLE** due to space constraints in the figure.
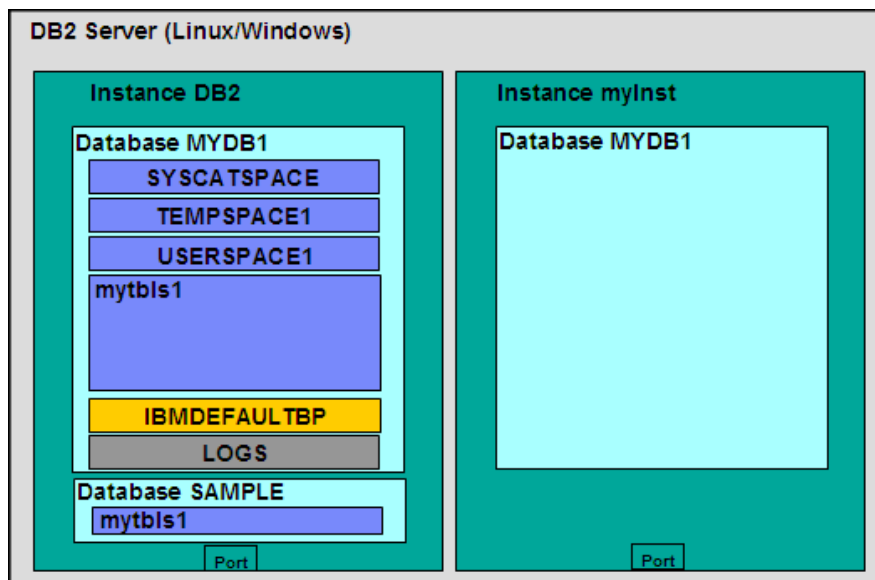
Figure 4.10 – Table spaces with the same name in different databases.

Once you have created a table space, you can create objects inside the table space such as tables, views and indexes. This is illustrated in *Figure 4.11*.
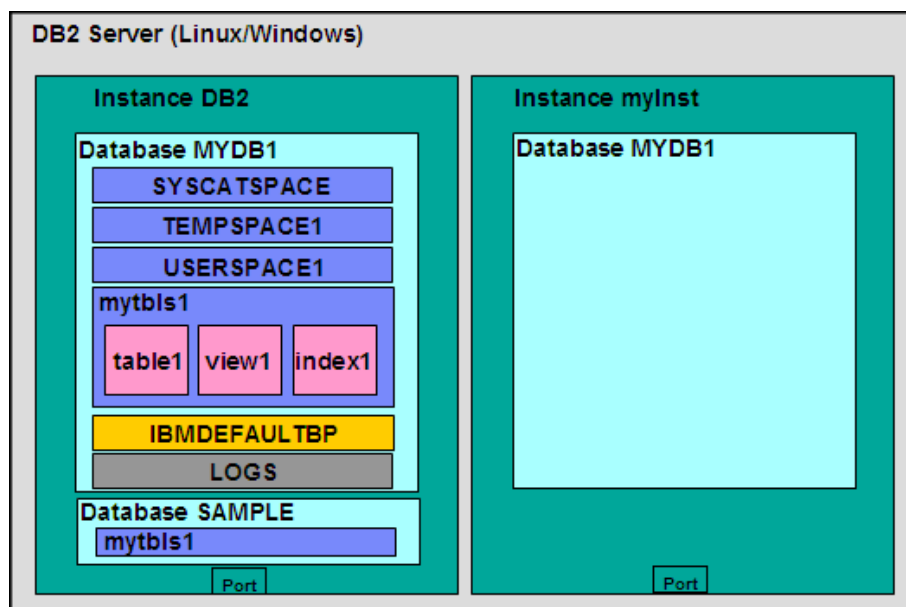
Figure 4.11 – Tables, views, indexes created inside the table space

4.1 DB2 configuration

DB2 parameters can be configured using the Configuration Advisor Tool. To access the configuration advisor through the Control Center, right click on a database and choose *Configuration Advisor*. Based on your answers to some questions about your system resources and workload, the configuration advisor will provide a list of DB2 parameters that should be changed with suggested values for each. If you would like more detail about DB2 configuration, keep reading; otherwise, use the Configuration Advisor and you are good to work with DB2!

A DB2 server can be configured at four different levels:

- Environment variables
- Database manager configuration file (dbm cfg)
- Database configuration file (db cfg)
- DB2 profile registry

This is also shown in *Figure 4.12*. In the figure, note where each of the boxes reside. For example, environment variables are set at the operating system level of the server, while database manager configuration file parameters are set at the instance level. Database

configuration parameters are set at the database level, and the DB2 profile registry is set either at the operating system or instance level.
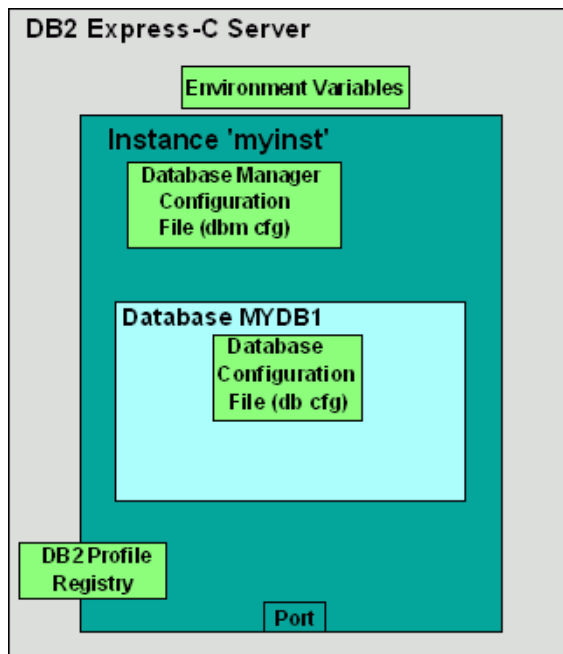
Figure 4.12 – DB2 Configuration

4.1.1 Environment variables

Environment variables are variables set at the operating system level. One key environment variable is **DB2INSTANCE**. This variable indicates the active instance you are working on, and for which your DB2 commands would apply. For example, to set the active instance to *myinst* in Windows, you can run this operating system command:

```
set db2instance=myinst
```

4.1.2 Database manager configuration file (dbm cfg)

The database manager configuration file (dbm cfg) includes parameters that affect the instance and all the databases contained within. The database manager configuration file can be viewed or modified using the command line, or through the DB2 Control Center.

To work with the dbm cfg from the Control Center, select the instance object from the instance folder of the control center, right-click to reveal the popup menu and select *Configure Parameters*. This is shown in *Figure 4.13*.
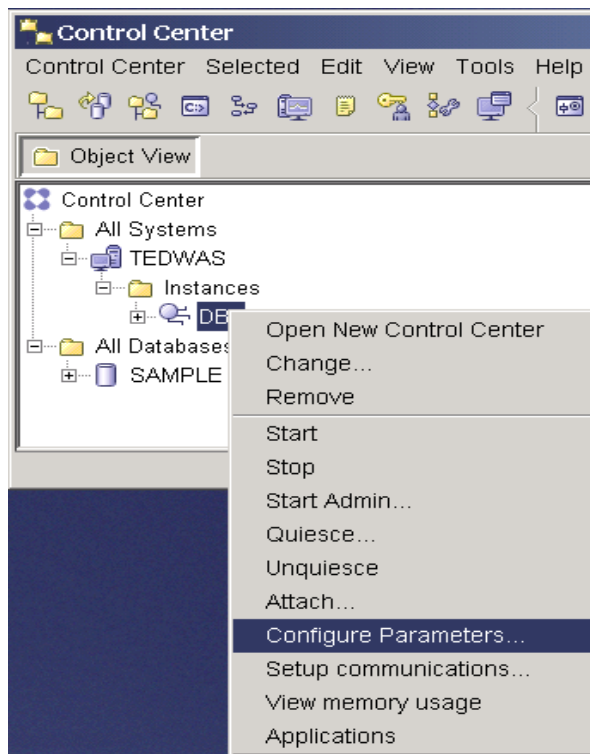
Figure 4.13 – Configuring the dbm cfg from the Control Center.

After choosing *Configure Parameters*, the screen shown in *Figure 4.14* will be displayed with the list of dbm cfg parameters.
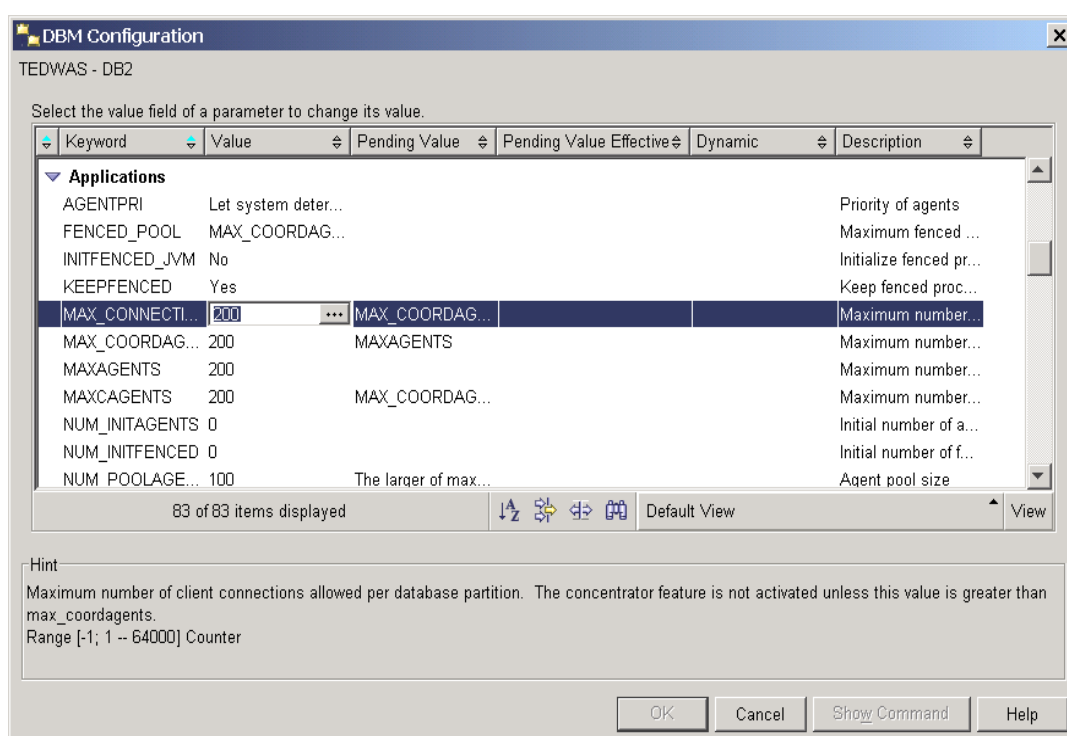
Figure 4.14 –The dbm cfg dialog

Many parameters are dynamic meaning that changes take effect immediately; however, changes to some parameters may require stopping and starting the instance. From the Command Line, this can be done using the **db2stop** and **db2start** commands.

Before an instance can be stopped, all applications must disconnect. If you wish to forcefully stop the instance, you can use the **db2stop force** command.

An instance can also be stopped through the Control Center by clicking on the instance object and selecting either *Stop* or *Start*.

Table 4.3 shows some useful commands to manage the dbm cfg from the Command Line.

Command	Description
`db2 get dbm cfg`	Retrieves information about the dbm cfg
`db2 update dbm cfg` `using <parameter_name> <value>`	Updates the value of a dbm cfg parameter

Table 4.3 - Commands to manipulate the dbm cfg

4.1.3 Database configuration file (db cfg)

The database configuration file (db cfg) includes parameters that affect a particular database. The database configuration file can be viewed or modified using the command line, or through the DB2 Control Center.

To work with the db cfg from the Control Center, select the database object from the database folder of the Control Center, right-click to reveal the popup menu and select *Configure Parameters*. This is shown in *Figure 4.15*.
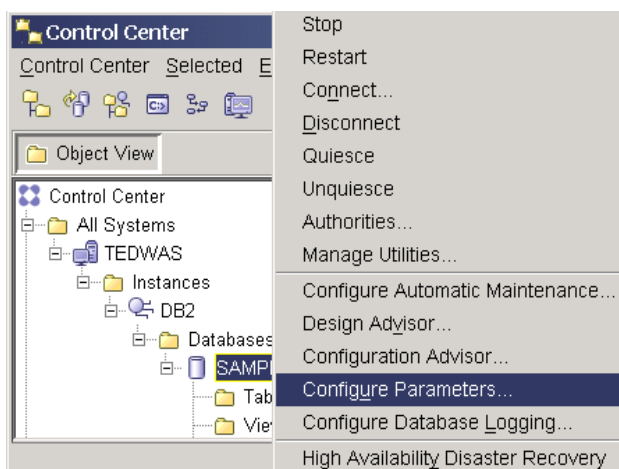
Figure 4.15 – Configuring the db cfg from the Control Center.

After choosing *Configure Parameters*, the screen shown in *Figure 4.16* will be displayed with the list of db cfg parameters.
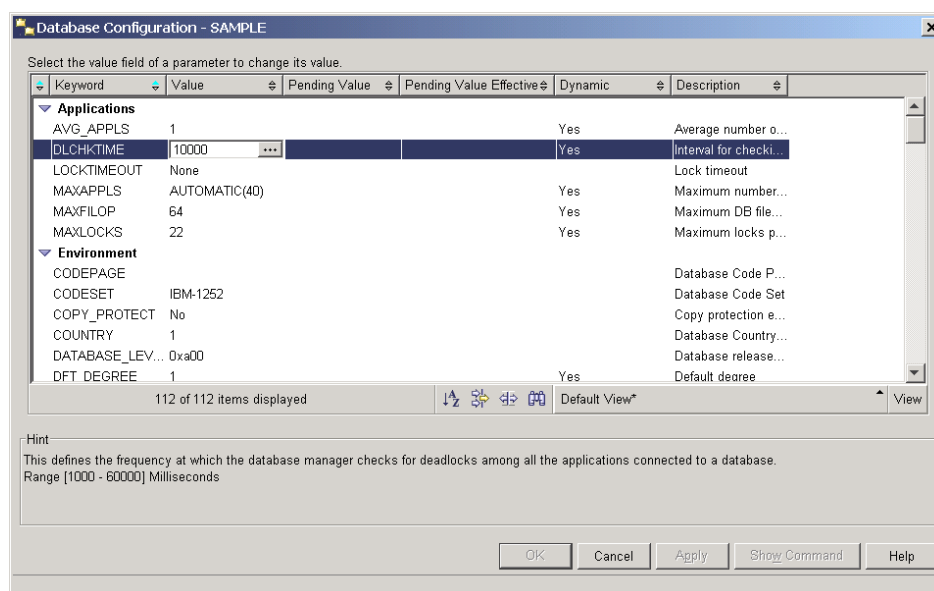
Figure 4.16 –The db cfg

Table 4.4 shows some useful commands to manage the db cfg from the Command Line.

Command	Description
`get db cfg for <database_name>`	Retrieves information about the db cfg for a given database
`update db cfg for <database_name>` `   using <parameter_name> <value>`	Updates the value of a db cfg parameter

Table 4.4 - Commands to manipulate the db cfg

4.1.4 DB2 profile registry

DB2 profile registry variables include parameters that may be platform specific and can be set globally (affecting all instances), or at the instance level (affecting one particular instance).

Table 4.5 shows some useful commands to manipulate the DB2 profile registry

Command	Description
`db2set –all`	Lists all the DB2 profile registry variables that are currently set
`db2set –lr`	Lists all the DB2 profile registry variables
`db2set <parameter>=<value>`	Assigns a parameter with a given value

Table 4.5 - Commands to manipulate the DB2 profile registry

Table 4.6 shows some of the most commonly used DB2 registry variables

Registry Variable	Description
DB2COMM	Specifies the communication managers that are started when the database manager is started.
DB2_EXTSECURITY	On Windows, prevents unauthorized access to DB2 by locking DB2 system files
DB2_COPY_NAME	Stores the name of the DB2 copy currently in use. To switch to a different DB2 copy installed, run the `installpath\bin\db2envar.bat` command. This variable cannot be used for this purpose.

Table 4.6 – Commonly used DB2 profile registry variables

For example, to allow for communication using TCPIP, set the DB2COMM registry variable to TCPIP as shown below:

```
db2set db2comm=tcpip
```

4.2 The DB2 Administration Server (deprecated)

The DB2 Administration Server (DAS) is a daemon process that runs at the DB2 server to allow remote clients to graphically administer the DB2 server. The DAS is needed only when using the DB2 graphical tools, either locally or remotely. There is only one DAS per physical computer as shown in *Figure 4.16*.
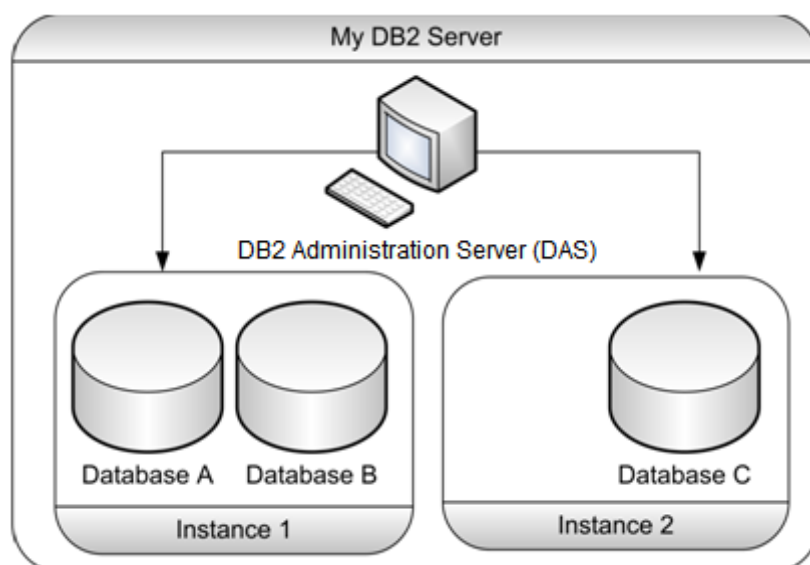
Figure 4.16 –The DB2 Administration Server (DAS)

4.3 Summary

In this chapter, we explored the DB2 environment, covering the concept and creation of instances and database, along with their common commands. From there, we looked at the other key aspects of instances, including table spaces: the three types of table space available, and the tables, views, and indexes that can be created within the table space; bufferpools; and logs.

Finally, we discussed the DB2 configuration structure, and how it can be changed in four different places, via: environment variables; the database manager configuration file; the database configuration file; and the DB2 profile registry.

4.4 Exercises

The exercises in this section will allow you to explore the concepts discussed in this chapter, and will introduce you to some of the DB2 tools.

Part 1: Create a new database using the Create Database wizard

In this part, you will create a database using the Create Database Wizard in the Control Center.

Procedure

1. From the Control Center Object View pane on the left side, right-click the *All Databases* folder and choose *Create Database* -> *With Automatic Maintenance*. This launches the Create Database Wizard.

2. Specify the database name and location in the *Name* page of the wizard. Use the following values:

   ```
   Database Name:          EXPRESS
   Default Drive (Windows): C:
   Default Path: (Linux):   /home/db2inst1
   Alias:                  This will default to EXPRESS if left blank
   Comment:                This is optional and can be left blank
   ```

   Click on the *Next* button to continue to the next page of the wizard.

   new in V9.7 Note: On Windows, by default you can only create a database on a drive, not on a path. If you would like to create a database on a path, set the DB2 registry variable DB2_CREATE_DB_ON_PATH

3. In the *Storage* page, don't make any changes, and click *Next*.

4. In the *Maintenance* page, leave the default choice (*Yes, I can specify an offline …*), and click *Next*.

5. Specify the offline maintenance time window in the *Timing* page of the wizard. Configure the window to start at 1AM every Monday through Thursday for a 6 hour duration. Click the *Next* button.

6. Configure notification on the *Mail Server* page of the wizard. DB2 can automatically page someone, or send an email if a problem or unhealthy condition is detected. If you want to configure this, indicate an available SMTP server for DB2 to use for sending email.   For this exercise we don't have an SMTP server, so leave this blank and click *Next*.

Review the options selected on the *Summary* page of the wizard. Click the *Finish* button to begin the database creation process. Database creation usually takes a few minutes, during which time a progress indicator is displayed.

Part 2: Working with instances, databases, and configuration

In this part, you will create a new instance, database, and change configuration parameters on a DB2 server on Windows. You can do it from either the Control Center or the DB2 Command Window. We provide the instructions using the DB2 Command Window.

Procedure

1. Open the DB2 Command Window by choosing *Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Window*. Alternatively, the short way to start it is to choose *Start -> Run*, and type **db2cmd**.

2. From the DB2 Command Window, create a new instance called *newinst*

    ```
    db2icrt newinst
    ```

3. Switch to the *newinst* instance and verify it is indeed your current instance. Then start it.

    ```
    set db2instance=newinst
    ```
    (Tip: No spaces before or after "=" sign!)
    ```
    db2 get instance
    ```
    (This verifies that *newinst* is the current instance)
    ```
    db2start
    ```

4. Create a database *newdb* with default values in instance *newinst.* This takes a few minutes as DB2 creates internal objects in the database and provides some initial configuration.

    ```
    db2 create database newdb
    ```

5. Connect and disconnect from the new database *newdb.* Then drop it.

    ```
    db2 connect to newdb
    db2 terminate
    db2 drop db newdb
    ```

6. Stop the current instance *newinst*

    ```
    db2stop
    ```

7. List all the instances in your server

    ```
    db2ilist
    ```

8. Switch to the DB2 instance and then verify you really switched

    ```
    set db2instance=db2
    db2 get instance
    ```

9. Drop the instance *newinst*

    ```
    db2idrop newinst
    ```

10. Find out the current value of the dbm cfg parameter FEDERATED. It should have a value of NO by default

    ```
    db2 get dbm cfg
    ```

    Tip: In Linux you could do: db2 get dbm cfg | grep FEDERATED

11. Change the value of the dbm cfg parameter FEDERATED to YES and verify the change occurred.

    ```
    db2 update dbm cfg using FEDERATED YES
    ```

    Since FEDERATED is not a dynamic parameter, changes are not effective until you stop and start the instance. However, to stop the instance, you have to be sure there are no connections. One way to ensure this is to issue these commands:

    ```
    db2 force applications all
    db2 terminate
    ```

    Restart the instance, and verify the new value for FEDERATED:
    ```
    db2stop
    db2start
    db2 get dbm cfg
    ```

12. Connect to the SAMPLE database with the userID and password you are using on the operating system

    ```
    db2 connect to sample user <userID> using <password>
    ```

13. Review how many applications are running in your current instance

    ```
    db2 list applications
    ```

14. Open another DB2 Command Window and connect again to the SAMPLE database without specifying a userID and password. Then review how many connections you have now.

```
db2 connect to sample
db2 list applications
```

15. Force off the connection from one of the DB2 command windows. This provides an example of how a DBA can forcefully terminate the work of a given user (who is probably hogging the system resources)

```
db2 force application (<application handle for db2bp.exe>)
```

The application handle is a number or 'handle' of the application you want to force. You obtain this number from the output of the **db2 list applications** command. The application db2bp.exe represents the DB2 Command Window.

16. Verify the connection of one of the DB2 command windows has been forced off. If you don't know which of the two DB2 Command Windows were forced off, issue this statement on both of them.

```
db2 select * from staff
```

The DB2 Command Window that was forced off should return an error message with code SQL1224N. The other DB2 Command Window should return you the output for the query.

17. Drop and recreate the DAS,and start it.

```
db2admin stop
db2admin drop
db2admin create
db2admin start
```

18. Take a look at the current value for the registry value DB2COMM

```
db2set -all
```

19. Unset the DB2COMM registry variable and verify this has been done

```
db2set db2comm=
db2stop
```
(Tip: You'll get an error on db2stop if you have connections. What should you do? Refer to a previous step to resolve this)

```
db2start
db2set -all
```

20. Set the DB2 Registry variable DB2COMM to **tcpip** and **npipe** in your instance and verify the new value

```
db2set db2comm=tcpip,npipe
db2stop
db2start
db2set -all
```

21. Check the current value of the LOGSECOND db cfg parameter, and then change it to a value of 5 and verify the new value

```
db2 connect to sample
db2 get db cfg
db2 update db cfg using LOGSECOND 5
db2 get db cfg
```

5

Chapter 5 – DB2 Tools

In this chapter, we describe some of the tools you can use with DB2. As of DB2 9.7, most of the tools described in this chapter are now deprecated, so they are still supported but will no longer be enhanced, and may not be included with the product in future releases. IBM Data Studio is the replacement for these tools.

The red circle in *Figure 5.1* shows the area of focus in this chapter.
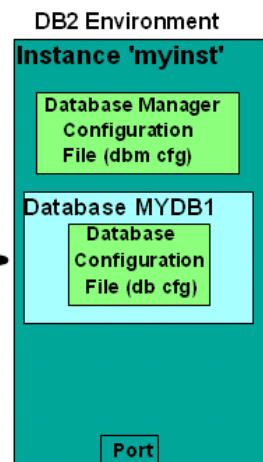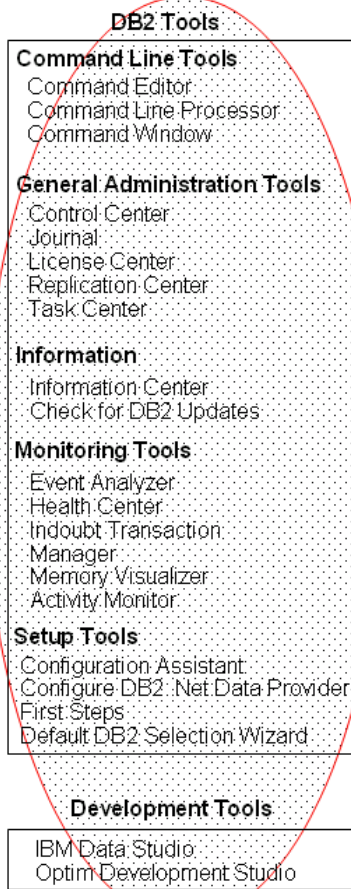
Figure 5.1 – The DB2 big picture: DB2 tools

Note:
See video presentations about the DB2 Tools and scripting at these links: http://www.channeldb2.com/video/video/show?id=807741:Video:4202 http://www.channeldb2.com/video/video/show?id=807741:Video:4182

Figure 5.2 lists all the DB2 Tools available from the IBM DB2 Start Menu shortcuts. Most of these tools are the same on Linux and Windows.
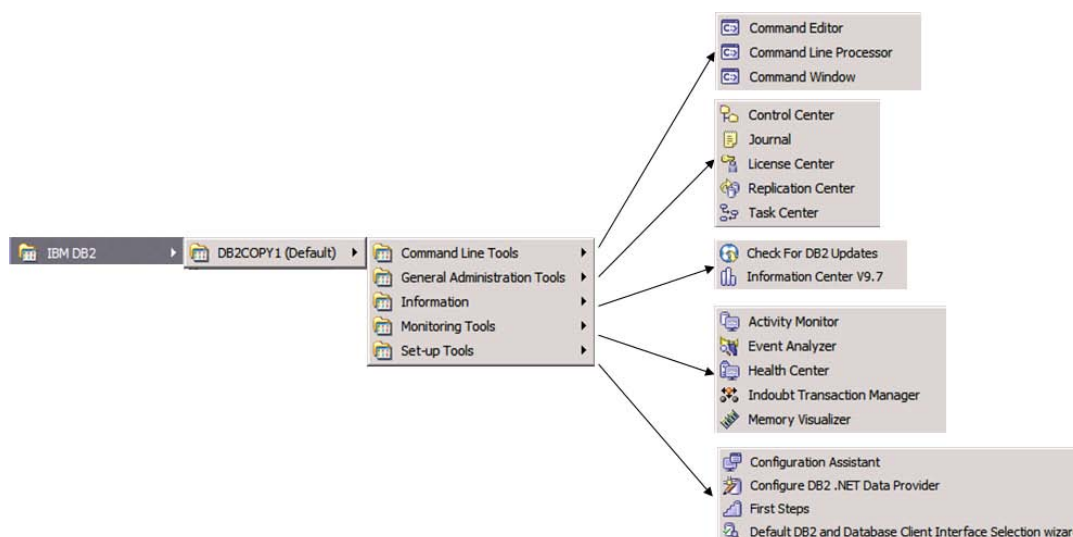
Figure 5.2 – DB2 tools from the IBM DB2 Start menu

Table 5.1 provides a list of shortcut commands that can be used to start some of the most popular tools in either Linux or Windows. It also lists which tools have been deprecated in DB2 9.7.

Tool name	**Command**	**Deprecated?**
Command Editor	db2ce	Yes
Command Line processor	db2	No
Command Window (Only on Windows platforms)	db2cmd	No
Control Center	db2cc	Yes

Task Center	db2tc	Yes
Health Center	db2hc	Yes
Configuration Assistant	db2ca	Yes
First Steps	db2fs	No

Table 5.1 – Shortcut commands to start some DB2 tools

5.1 IBM Data Studio

With DB2 9.7, IBM Data Studio is the primary tool to use for database administration, and database development with DB2. Data Studio is free. It can run on Linux and Windows, and is part of the IBM Integrated Data Management portfolio of products. The development of Data Studio follows a schedule which is not tied to the releases of DB2; however the products do coordinate their releases as often as possible. For example, DB2 9.7 and IBM Data Studio 2.2 were released the same day in June of 2009.

Figure 5.3 shows how IBM Data Studio looks like.
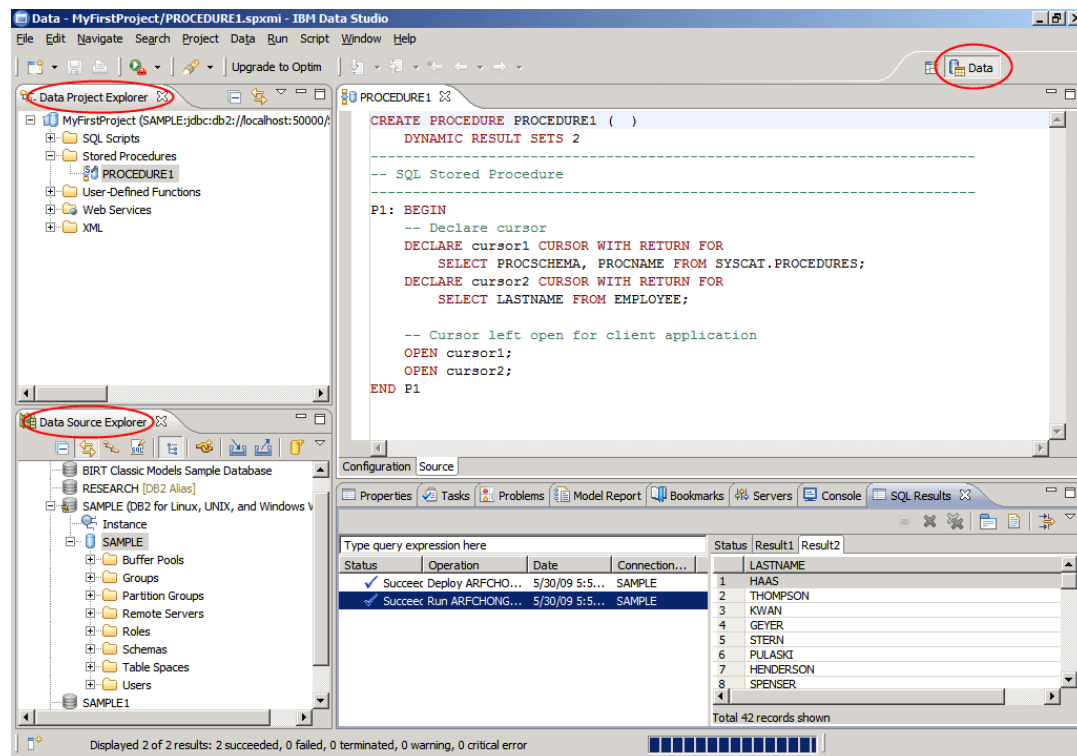
F**igure 5.3 - IBM Data Studio**

If you are familiar with Eclipse, you will note that Data Studio is Eclipse based. With Data Studio, typically you will work within the Data perspective window (highlighted in the figure

at the top right corner). You can also switch to the Java perspective, if you are developing a Java program. There are two views highlighted in the figure:

- Data Project Explorer (top left)
- Data Source Explorer (bottom left)

The Data Project Explorer view is used by database developers to work with SQL scripts, XQuery, stored procedures, UDFs, and Data Web services.

The Data Source Explorer view is used by database administrators to manage DB2 instances and databases. Using this view you can perform most of the functionality previously available in the Control Center.

In the figure, the view with title PROCEDURE1 is an editor for the procedure being highlighted in the Data Project Explorer. Depending on the task you are executing, editors or other windows will appear, allowing you to either code or perform more configurations.

IBM Data Studio comes in two flavors:

- Stand-alone package
- IDE package

The stand-alone package is a lot smaller than the IDE package but does not support Data Web services development, and cannot be extended to other Eclipse-based IBM products like InfoSphere Data Architect, for example. Other than that, the interface and capabilities are the same.

With IBM Data Studio, you can also work with other data servers, like Informix. Companies that work with several data servers and have a small DBA or database developer team now have the convenience of working and managing all of them from within one tool.

Note:

For more information about Data Studio, refer to the Getting Started with IBM Data Studio for DB2 free eBook which is part of this book series.

5.2 Control Center (deprecated)

Prior to DB2 9.7, the primary DB2 tool for database administration was the Control Center, as illustrated in *Figure 5.4.*
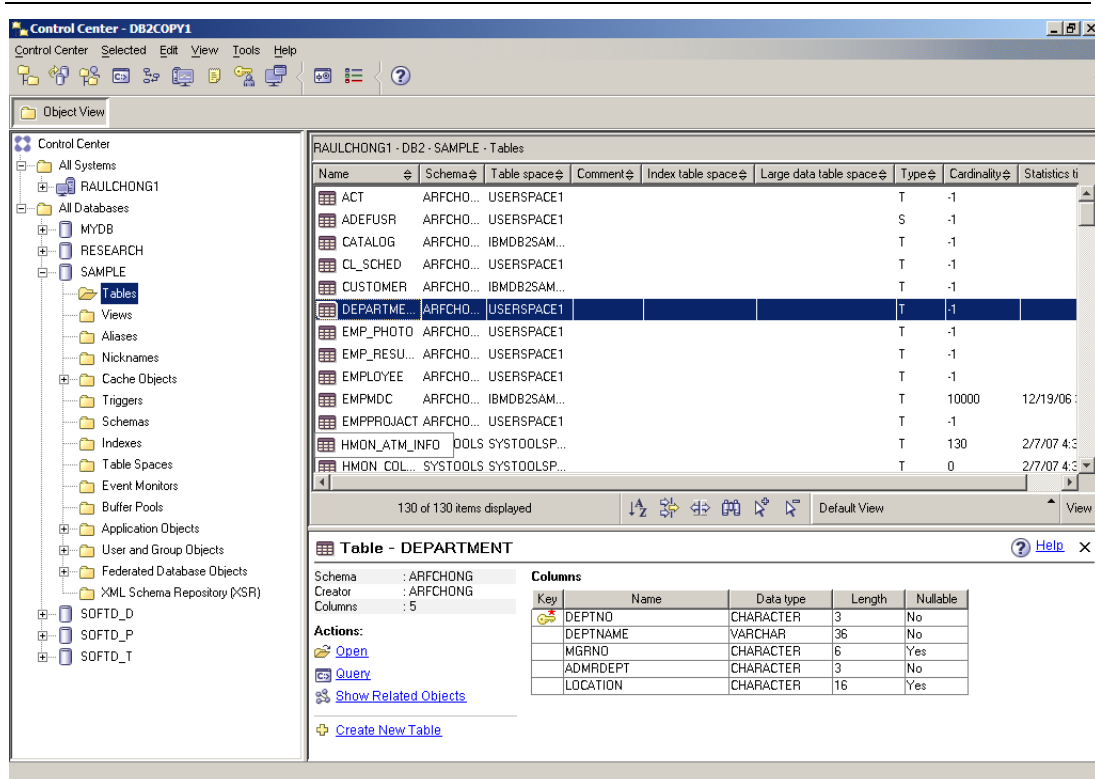
Figure 5.4 - The DB2 Control Center

The Control Center is a centralized administration tool that allows you to:

- View your systems, instances, databases and database objects;

- Create, modify and manage databases and database objects;

- Launch other DB2 graphical tools

The pane on the left side provides a visual hierarchy for the database objects on your system(s), providing a "folder" for Tables, Views, etc. When you double-click a folder (for example, the Tables folder, as shown in *Figure 5.4*), the pane on the top right will list all the related objects, in this case, all the tables associated with the **SAMPLE** database. If you select a given table in the top right pane, the bottom right pane provides more specific information about that table.

Right clicking on the different folders or objects in the Object tree brings up menus applicable to the given folder or object. For example, right-clicking on an instance and choosing *Configure parameters* would allow you to view and update the database manager configuration file. Similarly, if you right-click on a database and choose *Configure parameters*, you would be able to view and update the database configuration file. The DB2 environment and configuration parameters are discussed in more detail in *Chapter 5, The DB2 Environment*.

The first time you launch the Control Center, you are asked to choose what view you would like to use. The choice of view determines what types of options and database objects are exposed. *Figure 5.5* shows the Control Center View dialog box.
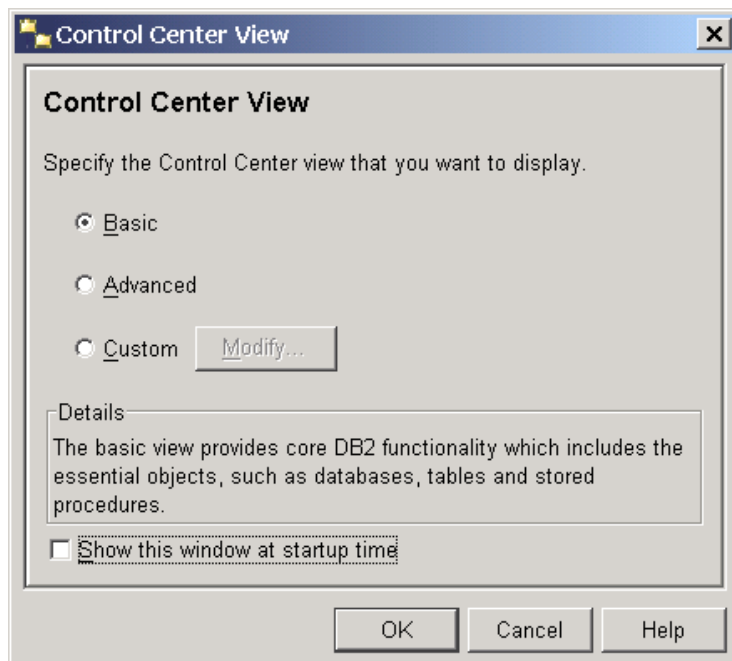
F**igure 5.5 - The DB2 Control Center View Dialog Box**

The basic view provides core DB2 functionality, The advanced view shows more options and features. The custom view allows you to customize the specific features, options, and objects you see.

To re-invoke the Control Center View dialog, select *Tools -> Customize Control Center* as shown in *Figure 5.6*.
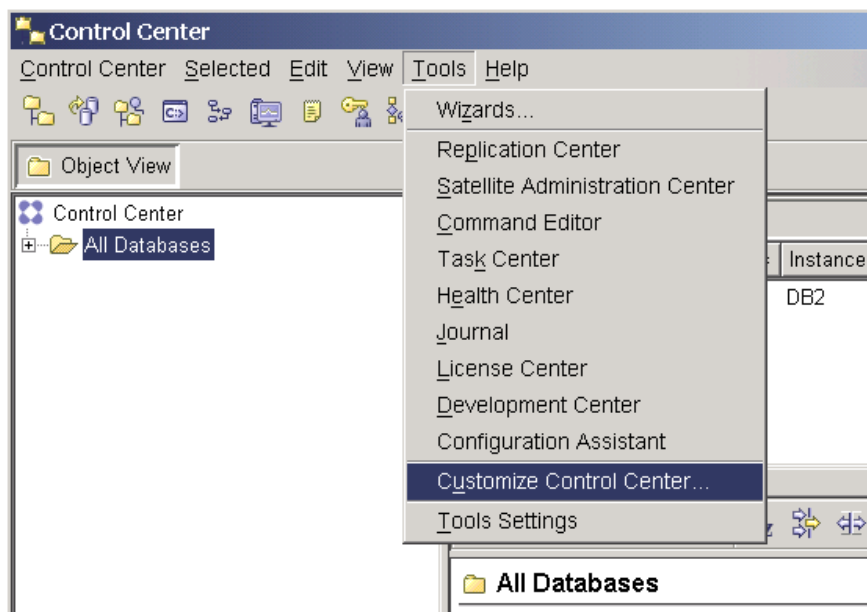
F**igure 5.6 – Customizing the Control Center**

5.2.1 Launching the Control Center

There are many ways to launch the Control Center:

- Navigating through the Windows *Start* menu

- By executing **db2cc** on a command prompt

- By clicking the Control Center icon ![icon] in the toolbar of any of the other DB2 GUI tools

- From the DB2 icon in the Windows system tray as shown in *Figure 5.7* (Right click on the DB2 green icon and select the DB2 Control Center menu option)
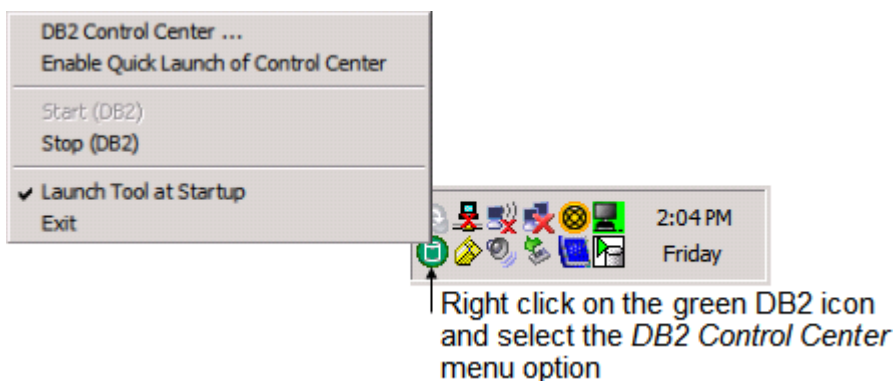
F**igure 5.7 – Launching the DB2 Control Center from the Windows system tray**

5.3 Command Editor (deprecated)

Using the DB2 Command Editor, you can execute DB2 commands, SQL and XQuery statements, analyze the execution plan of a statement, and view or update query result sets. *Figure 5.8* shows the Command Editor with a description of its elements.
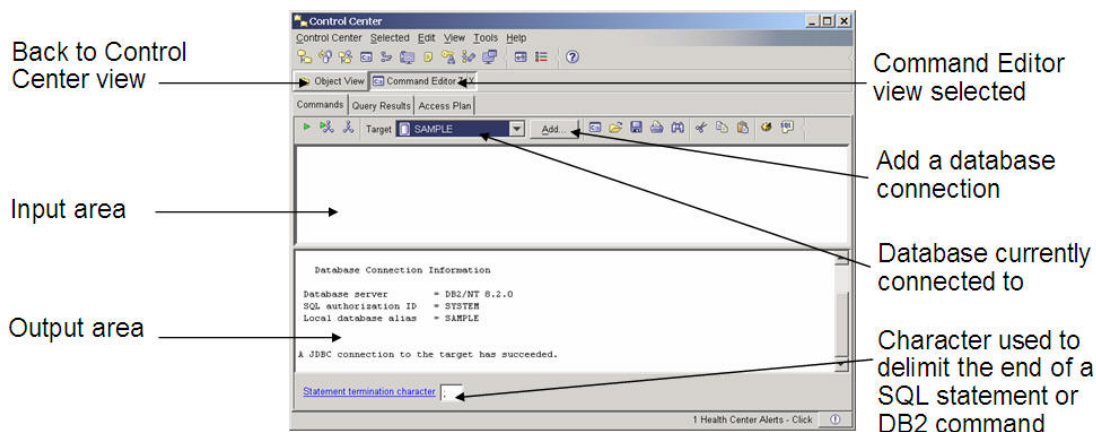
Figure 5.8 – DB2 Command Editor

In the input area, you can input multiple statements, so long as each statement ends with a termination character. If you press the execute button (first button on the left in *Figure 5.9*), the statements will be executed one after another. If you explicitly highlight a particular statement, only the highlighted statement will be executed. A database connection must exist in order to carry out any SQL statements, however, one of the statements can be a connect statement.
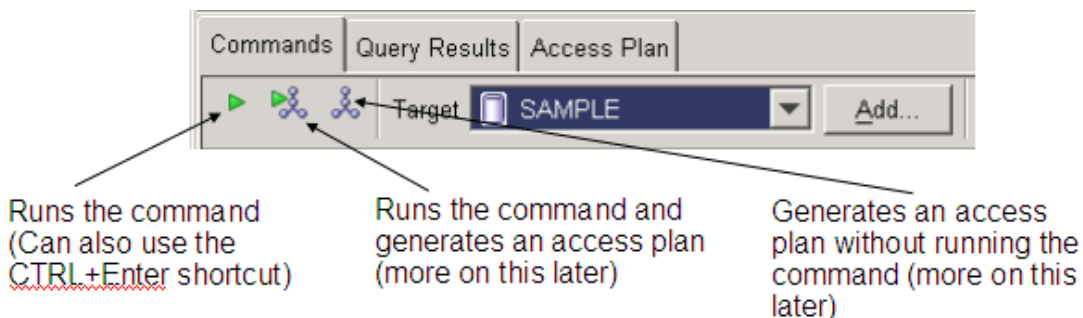
Figure 5.9 – The Command Editor – Commands tab

5.3.1 Launching the Command Editor

You can launch the Command Editor in several ways:

- From the Windows Start Menu: *Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Editor*
- From a command prompt, type **db2ce**
- From the Tools menu in the Control Center

- Embedded within the Control Center

    - Right click on the **SAMPLE** database icon in the Control Center's Object Tree pane and select the *Query* menu item

    - Any time a queryable object is selected (database, table, etc.), you can launch the Command Editor by clicking the *Query* link in the Control Center's Object Detail pane

- From the Control Center, click the Command Editor icon [icon] on the Control Center Toolbar as shown in *Figure 5.10*

Figure 5.10 – The Command Editor icon in Control Center

5.3.2 Adding a database connection

To add a connection to a database, click on the Add button (See *Figure 5.8*). A dialog as shown in *Figure 5.11* will appear.
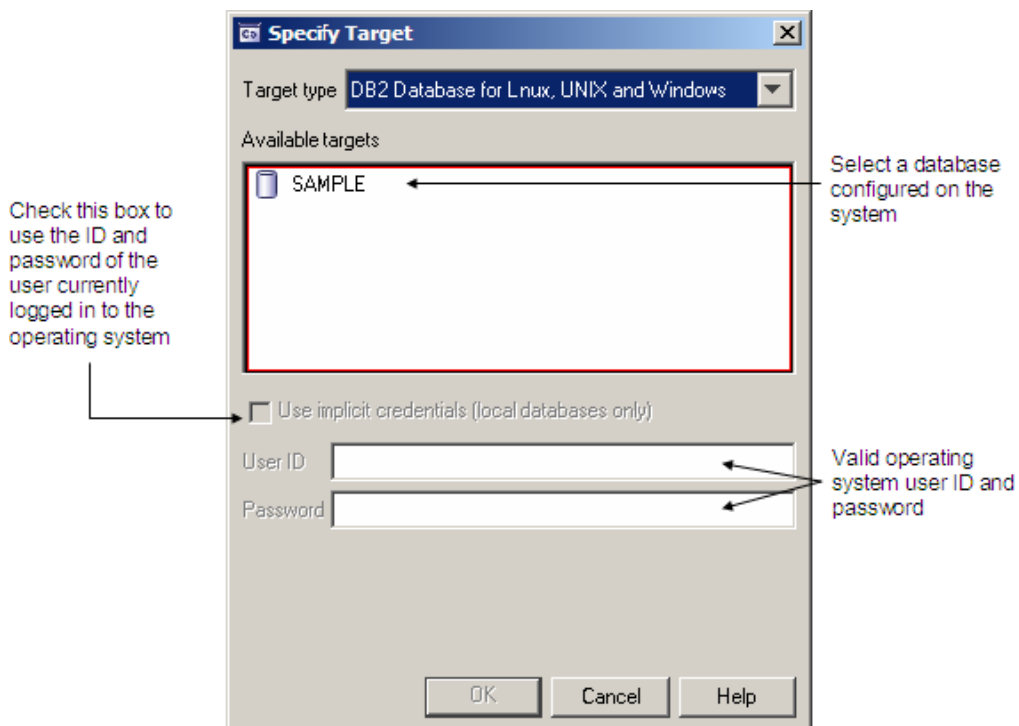
Figure 5.11 – Add a database connection

5.4 SQL Assist Wizard (deprecated)

If you are not familiar with the SQL language and would like to use an assistant or wizard to generate the SQL code, the SQL Assist Wizard is available from the Command Editor to help you. As shown in *Figure 5.12*, you invoke it from the Command Editor by clicking on the last icon with the SQL symbol. This icon will only appear after you connect to a database.
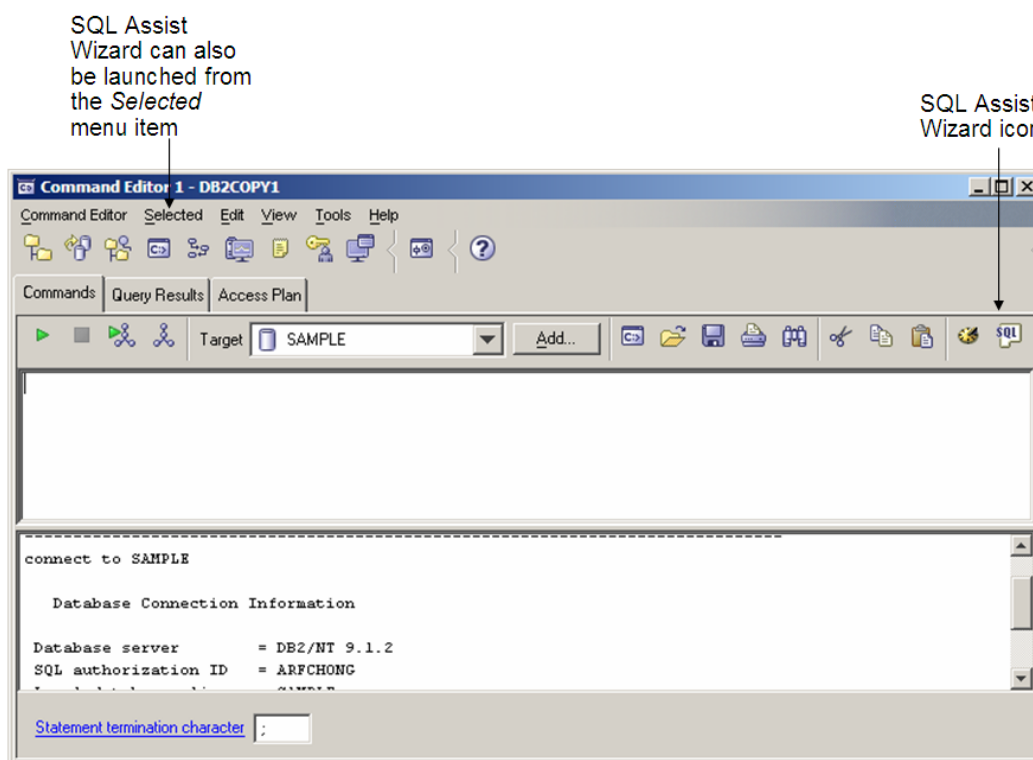
Figure 5.12 – Invoking the SQL Assist Wizard

Figure 5.13 shows the SQL Assist wizard. It is fairly straight forward to use. First indicate the type of SQL statement you need assistance with (SELECT, INSERT, UPDATE, DELETE). Depending on which statement you choose, different options will appear. At the bottom of the window you will see how the SQL statement is constructed as you select different choices in the wizard.
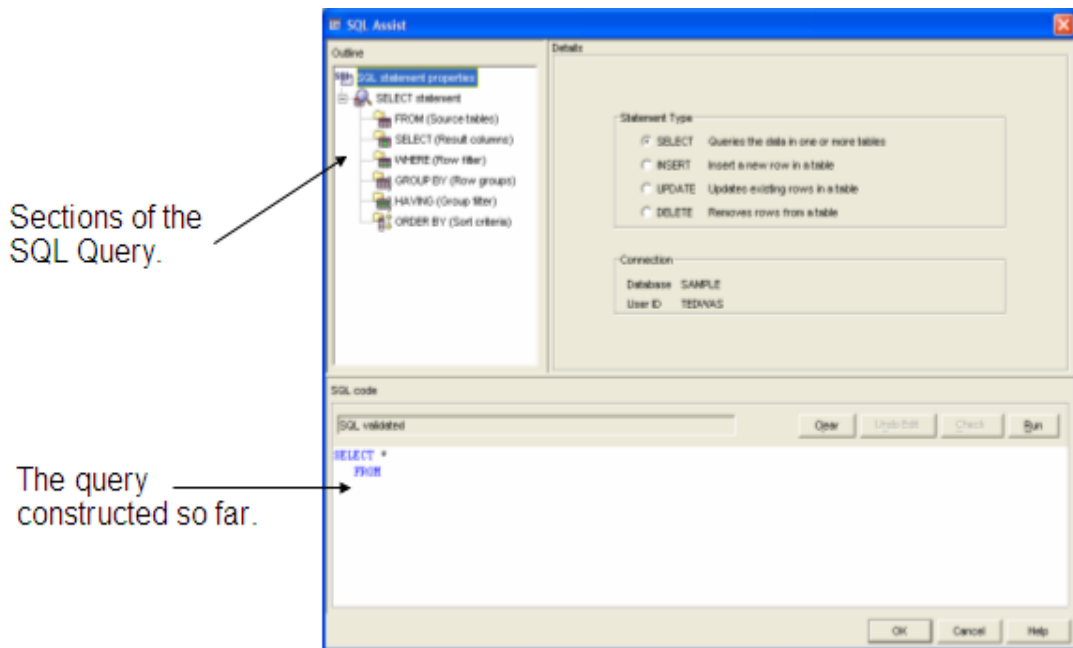
Figure 5.13 – The SQL Assist wizard

5.5 Show SQL Button (deprecated)

Most GUI tools and wizards in DB2 allow you to review the actual command or SQL statement that is created as a result of using the tool or wizard to perform an action. To see this, click on the *Show SQL* button in the tool you are working on, as shown in *Figure 5.14* and *Figure 5.15*

Figure 5.14 – The Show SQL button
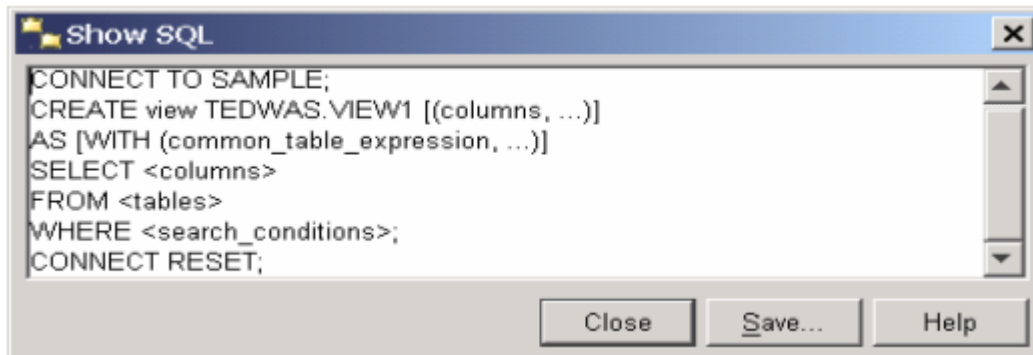
Figure 5.15 – The output of a Show SQL button

The ability to review the SQL statements and commands is very handy for learning SQL syntax, and for saving the commands or statements to a file for later use. You can also build scripts by reusing these generated commands and statements.

5.6 Task Center (deprecated)

The Task Center GUI tool allows you to create tasks: a set of operations such as running DB2 commands, operating system commands, or scripts. Subsequent actions can be performed if the task fails or succeeds. For example, if a task which involves backing up an important database at 3:00am in the morning is successful, an email can be sent to the DBA to provide this information. On the other hand, if the backup task fails, the Task Center can page the DBA. *Figure 5.16* shows the Task Center.
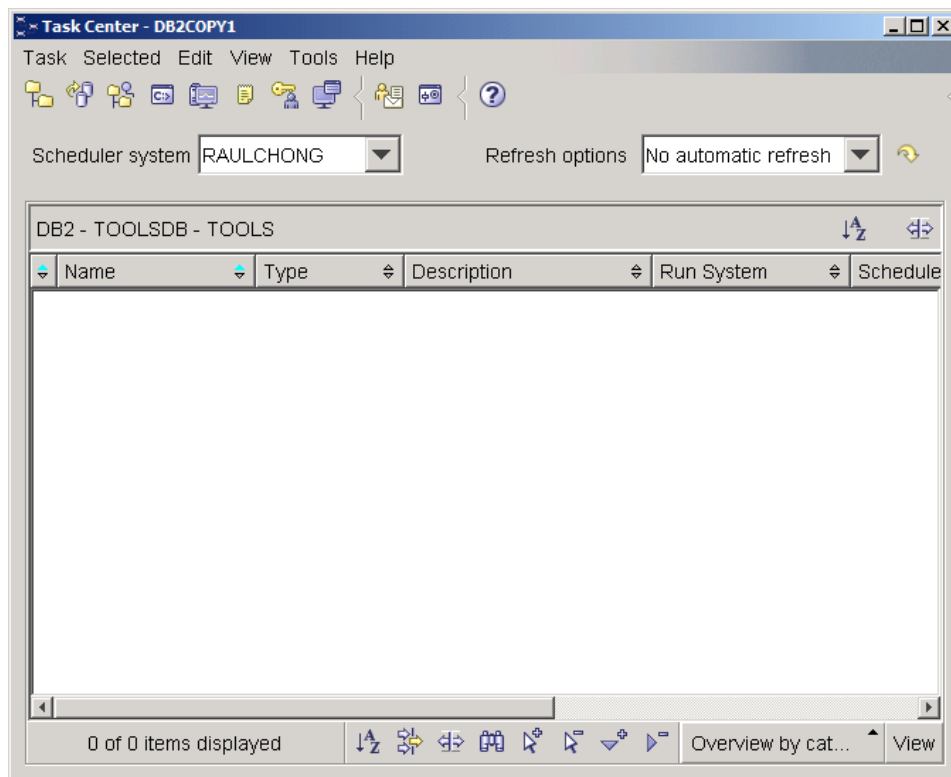
Figure 5.16 – The Task Center

5.6.1 The Tools Catalog database (deprecated)

All the details about your tasks and task scheduling are stored in a separate DB2 database called the Tools Catalog database. This database must exist ahead of time in order to schedule tasks. To create a Tools Catalog database you can use this command:

```
CREATE TOOLS CATALOG systools CREATE NEW DATABASE toolsdb
```

In the above example, *systools* is the schema name of all tables in the database, and the database name is *toolsdb*. We will talk more about schemas in *Chapter 8, Working with database objects.*

5.6.1.1 Launching the Task Center

You can launch the Task Center from the Control Center by clicking on *Tools -> Task Center*, as shown in *Figure 5.17*. Alternatively, you can start this tool from the Windows Start menu: *Start -> Programs -> IBM DB2 -> DB2COPY1(Default) -> General Administration Tools -> Task Center*
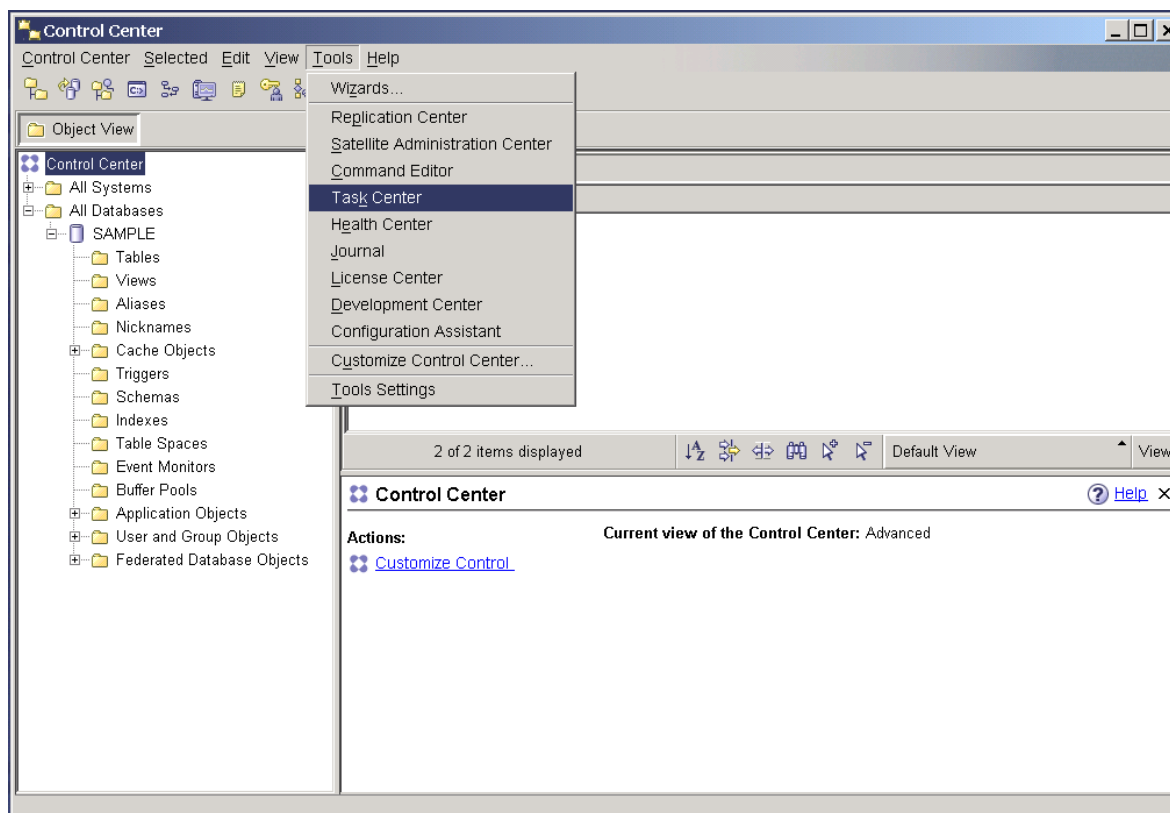
Figure 5.17 – Launching the Task Center

5.6.1.2 Scheduling with the Task Center

Any type of script, whether or not it was created through a DB2 GUI tool, can be scheduled using the Task Center. Tasks are run at their scheduled time from the system where you created the tools catalog database. We encourage you to explore the Task Center yourself. Creating a task is straightforward.

5.7 Journal (deprecated)

The DB2 Journal GUI tool provides a DBA with a journal of activities in online form. *Figure 5.18* shows the Journal and *Table 5.2* shows the information you can obtain from the Journal.
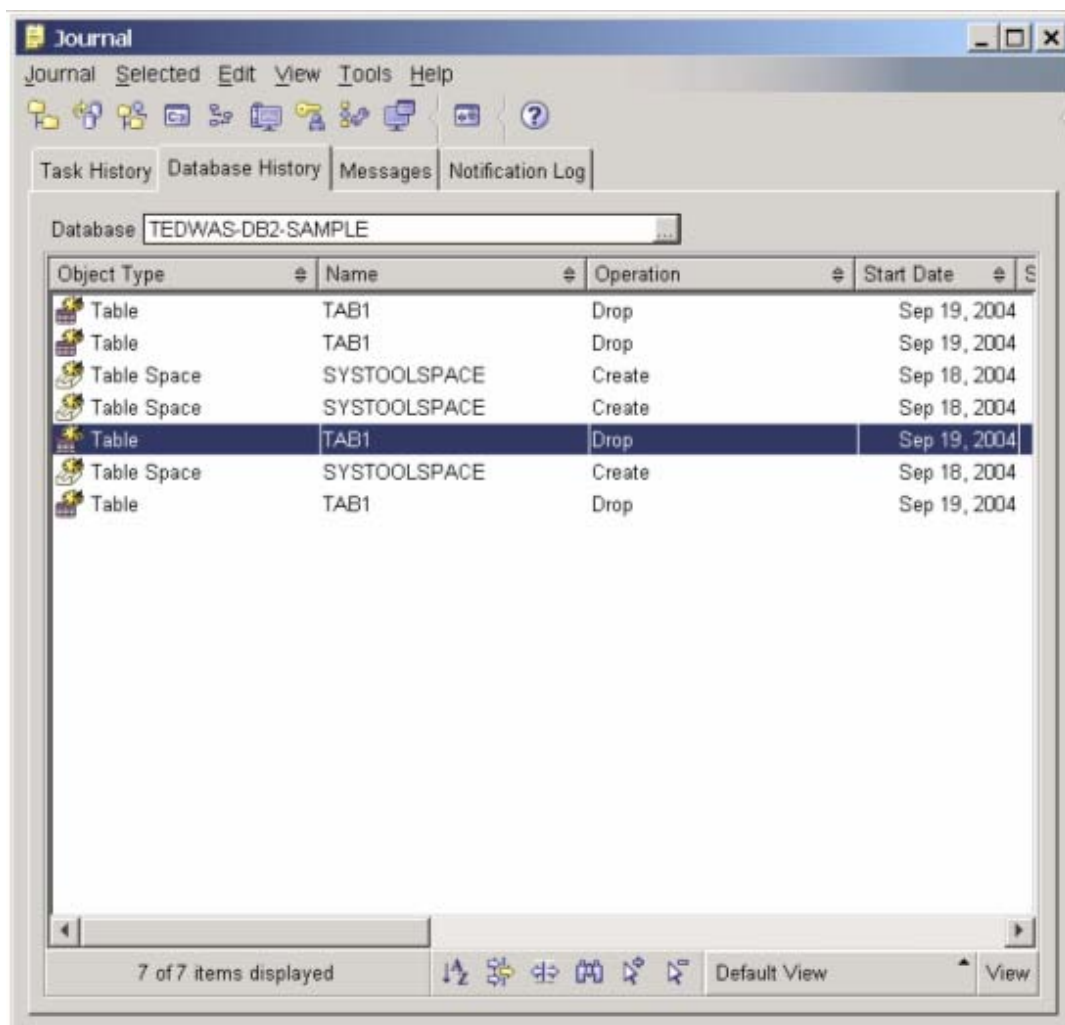
Figure 5.18 –The Journal

Type of Information	Description
Task History	All attempted scheduled tasks and their success status
Database History	A record of database activities performed (backup, restore, REORGs, etc.)
Message	History of messages returned by DB2 tools. This is useful if you want to recall and compare old error messages, or if you close a dialog box too quickly or by accident.
Notification Log	Contains system-level message. Critical errors are recorded

	here

Table 5.2 - Information provided in the Journal

5.7.1 Launching the Journal

You can launch the Journal from the Control Center by clicking on *Tools -> Journal*, as shown in *Figure 5.19*. Alternatively, you can start this tool from the Windows Start menu: *Start -> Programs -> IBM DB2 -> DB2COPY1(Default) -> General Administration Tools -> Journal.*
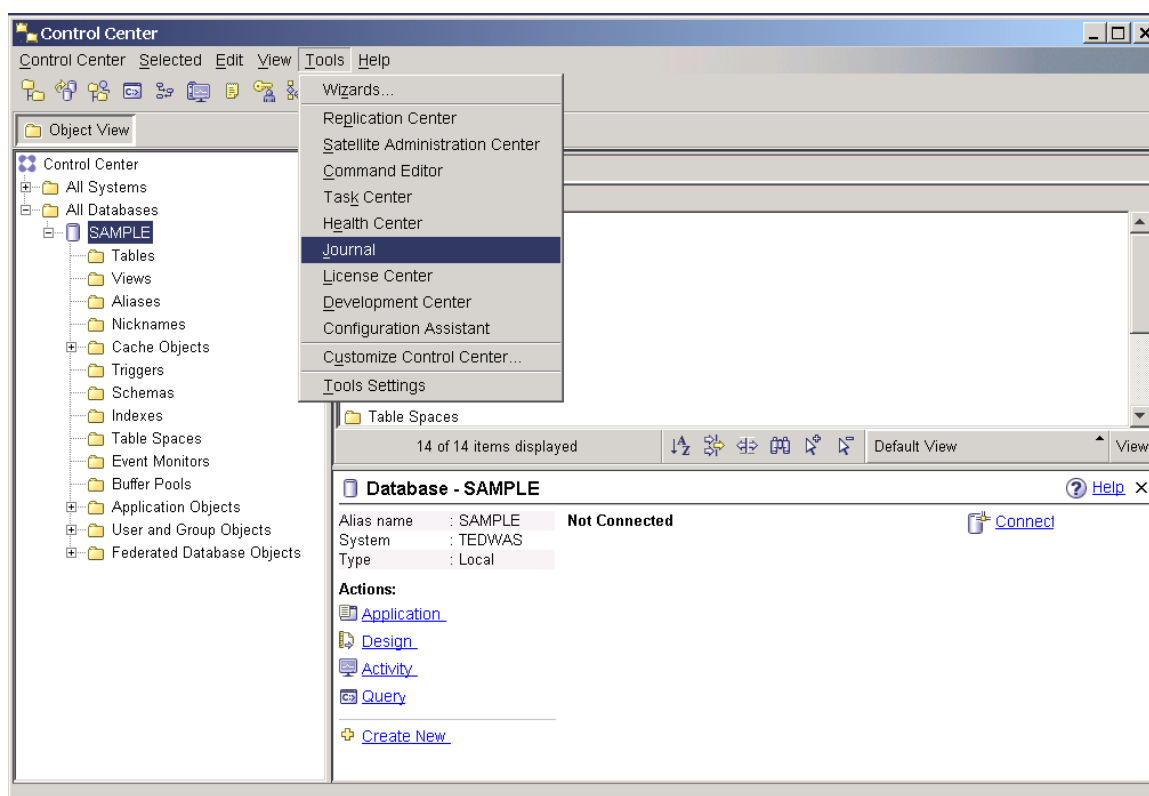
Figure 5.19 – Launching the Journal

5.8 Health Monitor (deprecated)

The Health Monitor is a default agent that runs within the DB2 Engine, monitoring all aspects of database health (memory, space management, automated activities previously defined, etc.). When some part of DB2 is operating outside of the set parameters, an exception is raised and brought to the attention of the DBA. There are three types of alert states:

- Attention: a non-normal state

- Warning: a non-critical state that does not require immediate attention but may indicate a non-optimal system

- Alarm: a critical condition requiring immediate action

The Health Monitor can be turned on or off using the database manager configuration parameter HEALTH_MON.

5.8.1 Health Center (deprecated)

The Health Center is a graphical tool for interacting with the Health Monitor. The Health Center breaks down health alerts on a system by instance, database, and table space levels. *Figure 5.20* shows the Health Center.
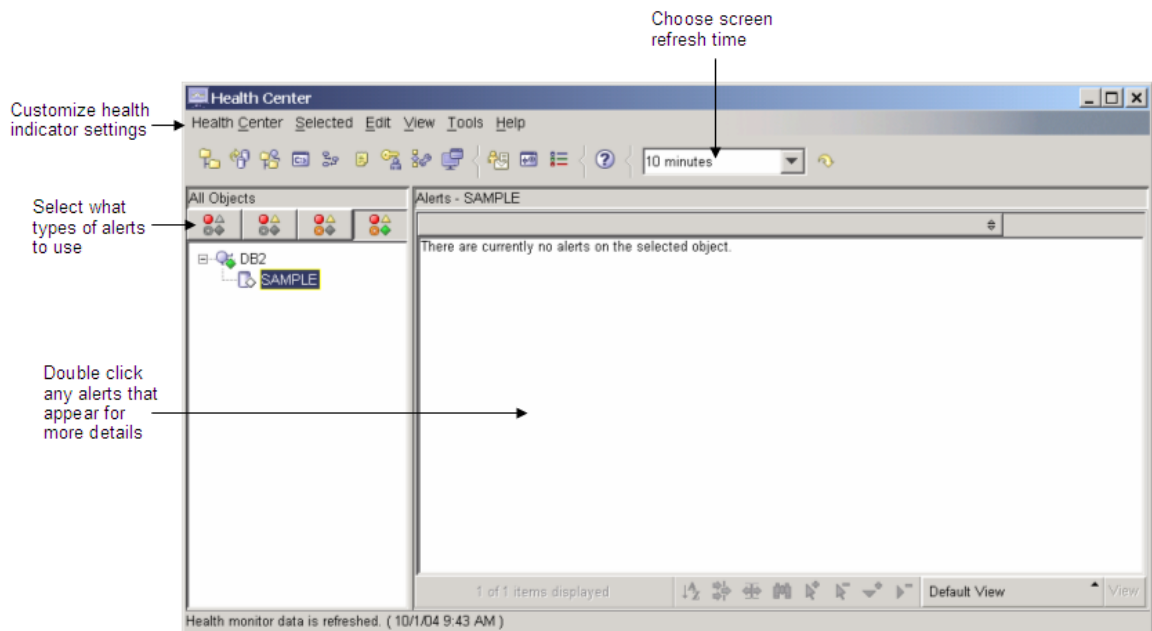
Figure 5.20 – The Health Center

5.8.1.1 Launching the Health Center

You can launch the Health Center from the Control Center by choosing *Tools -> Health Center*. This is shown in *Figure 5.21*. You can also start this tool from *Start -> Programs-> IBM DB2 -> DB2COPY1(Default) -> Monitoring Tools -> Health Center*
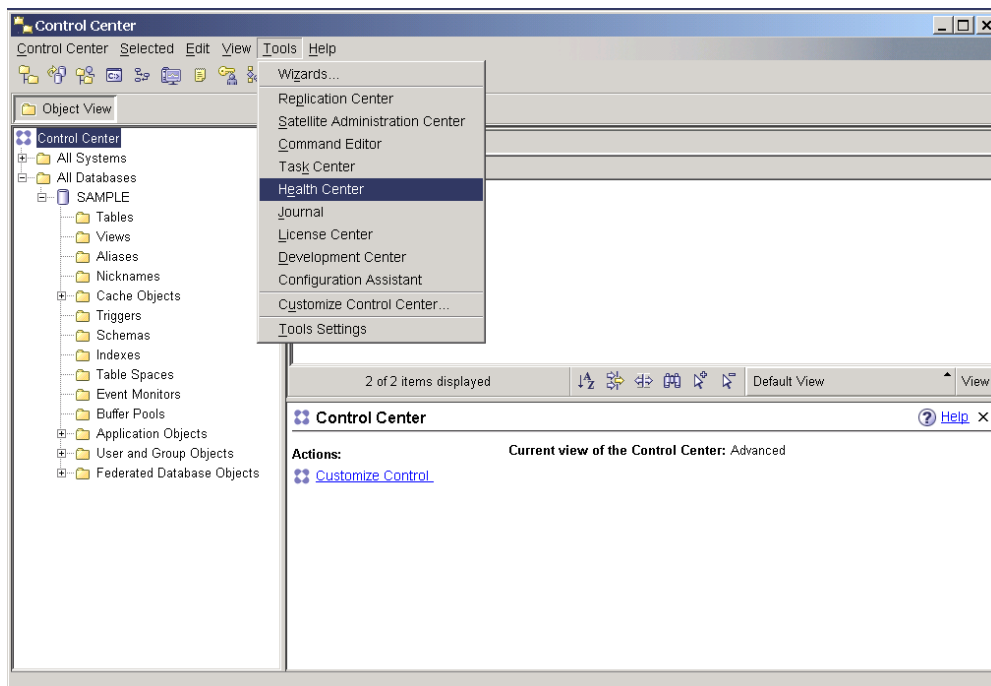
Figure 5.21 – Launching the Health Center

5.8.1.2 Configuring Health Alert Notification

Once your Health Center is started, you can configure the Alert notification by clicking on the *Health Center menu -> Configure -> Alert Notification* as shown in *Figure 5.22*. Alert notification allows you to input contact names with email addresses or pager numbers of the people who will be contacted if an alert is raised.
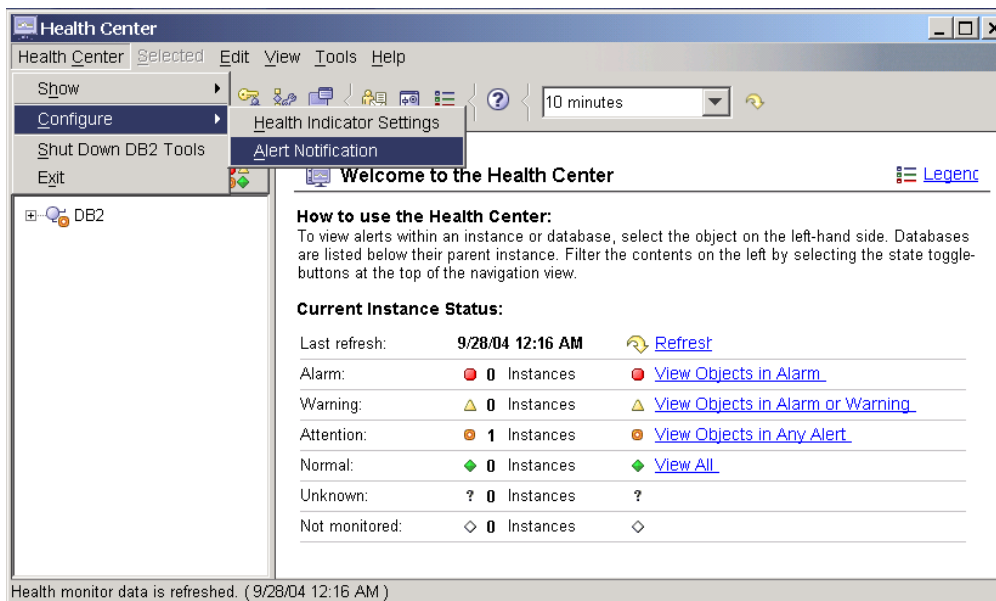
Figure 5.22 – Alert Notification

5.9 Self-tuning memory manager

The self-tuning memory manager (STMM) introduced in DB2 9 is one of several autonomic computing features that simplifies the task of memory configuration by automatically setting values for several memory configuration parameters. When enabled, the automatic tuner dynamically distributes available memory resources among several memory consumers for the database. The memory tuner responds to changes in workload characteristics, adjusting the values of memory configuration parameters and the sizes of buffer pools to optimize performance. To turn on STMM update the db cfg parameter SELF_TUNING_MEM to ON.

Other autonomic computing features such as automatic maintenance and automatic storage are discussed elsewhere in this book.

5.10 Scripting

It is always useful to be able to create script files that perform several DB2 commands or SQL statements repeatedly. For example, a DBA may want to run a given script every day to check the row count of important tables. There are two general forms of scripting:

- SQL scripts
- Operating system (shell) scripts.

5.10.1 SQL scripts

SQL scripts include query statements and database commands. These scripts are relatively simple to understand and are platform independent. However, variables or

parameters are not supported. For example, the commands shown in *Listing 5.1* below are saved in a file called `script1.db2`.

```
CONNECT TO EXPRESS;
CREATE TABLE user1.mytable
        (    col1 INTEGER NOT NULL,
             col2 VARCHAR(40),
             col3 DECIMAL(9,2));
SELECT * FROM user1.mytable FETCH FIRST 10 ROWS ONLY;
COMMIT;
```

Listing 5.1 - A sample SQL Script stored in file script1.db2

In the above script, all the statements are SQL statements and each statement is separated by a statement delimiter, in this case a semi-colon. The file name does not need to use the extension "db2". Any extension could be used.

5.9.1.1 Executing SQL Scripts

An SQL script can be executed from either the Command Editor or the DB2 Command Window on Windows, or through a Linux shell. To run the script in *Listing 5.1* from the DB2 Command Window or Linux shell, you can use the following command:

```
db2 -t -v -f script1.db2 -z script1.log
```

or the equivalent one:

```
db2 -tvf script1.db2 -z script1.log
```

In this command:

- **-t**   indicates that statements use the default statement termination character (a semi-colon)

- **-v**   indicates verbose mode; causing db2 to echo the command being executed

- **-f**   indicates that the filename specified after this flag is the script file.

- **-z**   indicates that the filename specified after this flag is used for appending screen output for later analysis (this is optional, but recommended)

When the **-t** flag is used and no line delimiter is specified, the semi-colon is assumed to be the delimiter of the statements. There may be situations where another delimiter is required. For example, a script containing SQL PL code needs to use a different statement termination character other than the default (semicolon), because semicolons are used within SQL PL object definitions to terminate procedural statements.

For example, in the script file `functions.db2` shown in *Listing 5.2* below, you see it contains the statement to create a function, and a semi-colon is needed at the end of the SELECT statement because it is part of the syntax required within the function. For the CREATE FUNCTION statement terminator we are using an exclamation mark (!). If we had

used a semi-colon for the statement terminator, there would have been a conflict in the
script with the one used for the SELECT statement, resulting in an error reported by DB2.

```
CREATE FUNCTION f1()
    SELECT …    ;
    …
END!
```
Listing 5.2 - Contents of the script file functions.db2

To inform DB2 that a different statement termination character is being used, use the **-d**
flag, followed by the terminator character desired as shown below:

```
db2 –td! –v –f functions.db2 –z functions.log
```

To find out more about the other flags that can be used from the Command Window or
Linux shell, use this command:

```
db2 list command options
```

5.10.2 Operating system (shell) scripts

Operating system scripts provide greater flexibility and power than SQL scripts, as they
give you the possibility to add additional programming logic. They are platform dependant,
but they support parameters and variables. *Listing 5.3* shows an example of a simple
Windows operating system (shell) script.

```
set DBPATH=C:
set DBNAME=PRODEXPR
set MEMORY=25
db2   CREATE   DATABASE   %DBNAME%   ON   %DBPATH%   AUTOCONFIGURE   USING
      MEM_PERCENT %MEMORY% APPLY DB AND DBM
db2 CONNECT TO %DBNAME% USER %1 USING %2
del schema.log triggers.log app_objects.log
db2 set schema user1
db2 –t –v –f schema.db2 –z schema.log
db2 –td@ -v –f triggers.db2 –z triggers.log
db2 –td@ -v –f functions.db2 –z functions.log
```
Listing 5.3 - Contents of the operating system script file create_database.bat

To execute this operating system script from the command line, you would issue the
following command on Windows:

```
create_database.bat db2admin ibmdb2
```

where ***db2admin*** is the userID and first parameter of the script, and ***ibmdb2*** is the
password and second parameter of the script.

On Windows using the "bat" extension tells the operating system that this is a batch
executable file. On Linux, you need to change the mode on the file to indicate the file is

executable using a command like **chmod +x**. Afterwards, you can run it in the same manner as listed above.

5.11 Windows Vista considerations

The User Access Control (UAC) feature in Windows Vista causes applications to start with standard rights, even if your user ID is a local administrator. This means that any DB2 tool or command you start in Vista will likely run, but will report problems related to access. To avoid this problem, use the shortcut called "Command window - Administrator" specifically created at installation time for Vista users. From this window, you can run other commands and launch other tools (using the commands shown in *Table 5.1* at the beginning of the chapter). Alternatively, from the Windows Vista Start menu or any DB2 shortcut, find the desired DB2 tool you want to launch, right-click on it, and choose the *Run as administrator* option.

If DB2 extended security is enabled, which is the default (see *Chapter 10, Database Security* for details), you must also ensure that your userID is a member of the DB2ADMNS group in order to launch graphical tools such as the Control Center.

5.12 Summary

In this chapter we looked at the wide array of tools available for administering, configuring, and managing your DB2 data server.

The arrival in DB2 9.7 of the IBM Data Studio as the main administration tool provides a new dimension to database administration and development work.

We also discussed a number of now deprecated GUI tools: the Control Center, the SQL Assist wizard, the Task Center and Journal, and the Health Agent and Monitor. However, the Command Line Processor and Command Window tools will continue to be a part of the application in versions after DB2 9.7. Also, the Self Tuning Memory Management tool remains a big part of the database optimization process.

A key component in the toolbox of any database administrator is the use of scripts to execute DB2 commands and functions. In this chapter, we took an in-depth look at both SQL and operating system (shell) scripts, in particular how they are composed, stored and executed.

Finally, we concluded with a notice about how to ensure that DB2 tools run smoothly on Windows Vista.

5.13 Exercises

The exercises in this section will let you practice working with scripts in DB2.

Part 1: Populating the EXPRESS database using scripts

In this part, you will populate the EXPRESS database (previously created) using the Command Editor and two supplied scripts.

Procedure

1. Populate the EXPRESS database with a few tables and some data. For your convenience, two scripts, `Lab_Chpt5.db2` and `Lab_Chpt5.dat` have been created to do this for you. The `Lab_Chpt5.db2` script contains the commands used to create the tables, and therefore <u>must be run first</u>. The `Lab_Chpt5.dat` script contains statements that insert data into the tables. Both scripts can be found in the `expressc_book_exercises_9.7.zip` file accompanying this book. To run these scripts, open Command Editor. Ensure that the new database you created is selected in the drop-down list in the toolbar. If the new database does not appear in the list, add a connection to it using the *Add* button.

2. Click the *Selected → Open* menu on the Command Editor and navigate to the folder where the scripts are stored. Select the `Lab_Chpt5.db2` file and click the *OK* button. The contents of the file should now be displayed in the Command Editor input area. Click the *Run* button to run the script. Verify that there were no errors encountered when running the script.

3. Repeat Step (2) for the `Lab_Chpt5.dat` file.

The new database you created is for a very simple Internet bookstore. The **BOOKS** table contains all the information about the books the store carries. The **CUSTOMERS** table contains information about each of the store's customers. Finally, the **SALES** table contains sales data. Whenever a customer purchases a book, a record is made in the **SALES** table. The diagram below shows the design and relationship between the tables.

BOOKS table

book_id	title	cost	image
(INTEGER)	(INTEGER)	DECIMAL(7,2)	BLOB (1MB)

CUSTOMERS table

cust_id	firstnme	lastname	address	email
(INTEGER)	VARCHAR	VARCHAR(100)	VARCHAR(300)	VARCHAR(100)

SALES table

sales_id	prod_id	cust_id	qty	price	purch_date
(INTEGER)	(INTEGER)	(INTEGER)	(INTEGER)	DECIMAL(7,2)	TIMESTAMP
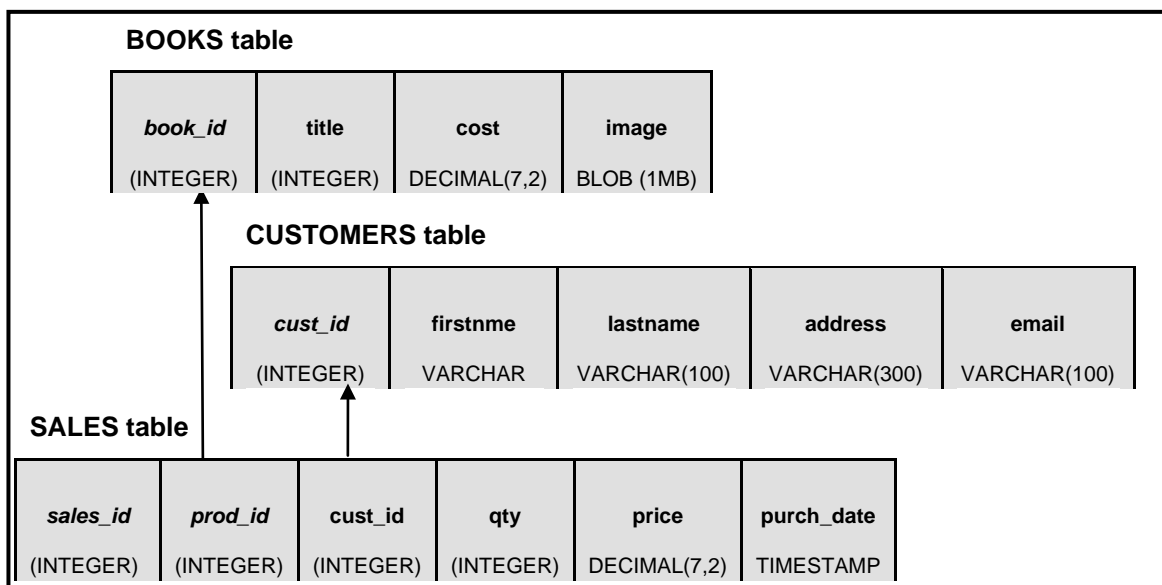
Part 2: Create an installation script for the EXPRESS Database
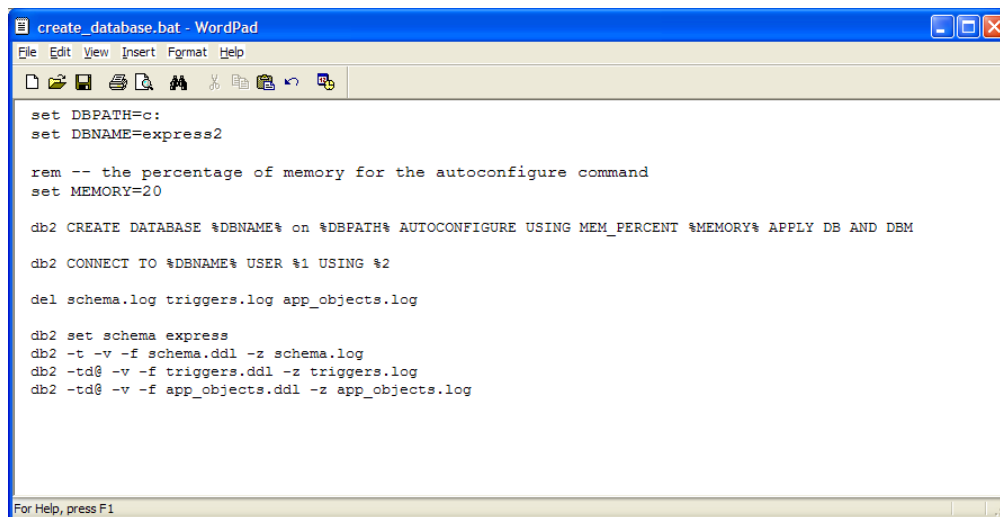
Scripts are a powerful mechanism for performing repetitive tasks such as database statistic collection, backups, and database deployment. Operating system scripts have the

advantage of supporting script parameters, making them more flexible. In this part, you will create an operating system script to redeploy the **EXPRESS** database as the **EXPRESS2** database. The script will call the previously generated SQL scripts for database objects. In order to save space, this exercise shows the scripts and commands specific to the Windows platform. If you prefer to work on Linux, ensure to make the appropriate changes to the instructions below.

Procedure

1. Open a text editor and input the information as shown below. Most people make typos when typing the lines below. We purposely do not provide this as a separate file so that you will make these errors and learn to fix them yourself!.

    Note as well that you may have to specify the correct paths for the `schema.ddl`, `triggers.ddl` and `app_objects.ddl` files which are also provided with the **expressc_book_exercises_9.7.zip** file accompanying this book.

2. Save the script file in a directory like `C:\express` and call it `create_database.bat`.

    Note: If you use Wordpad, then in the *Save As* dialog window, ensure you choose the *MS-DOS Format* option. If you save the file with a different format, Wordpad may introduce invisible characters which will cause problems in the execution of the script. Also, put quotes around the file name, as shown in the figure below, to ensure that Windows does not append a .TXT extension to it.
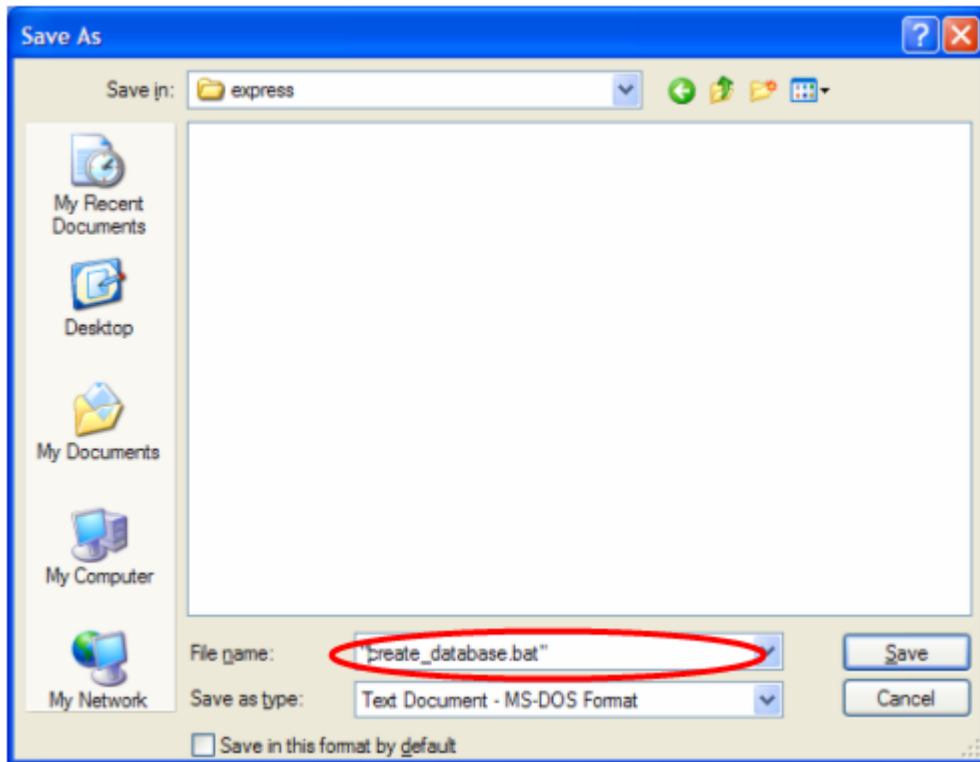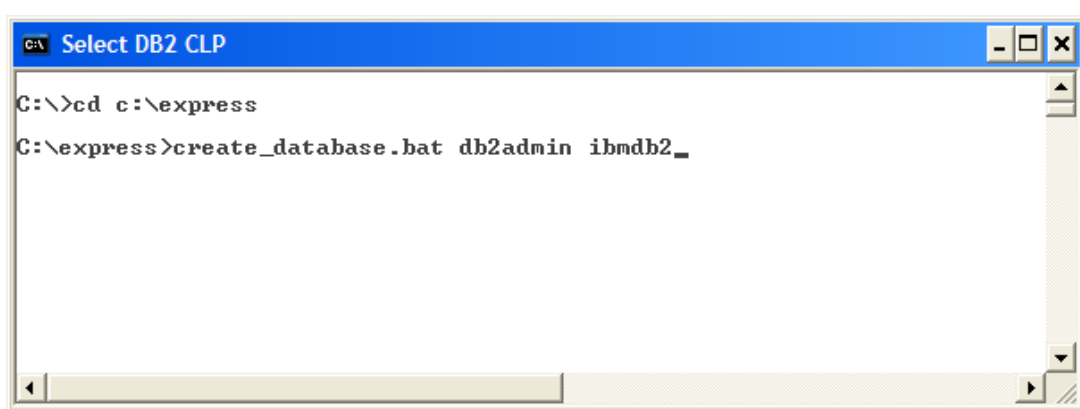
3. To run scripts that interact with DB2, you must have a DB2 command line environment. To open a DB2 Command Window, go to *Start -> Program Files -> IBM DB2 -> DB2COPY1 (default) -> Command Line Tools -> Command Window*. Alternatively, you can use *Start-> Run*, type **db2cmd**, and press *enter* as shown below.
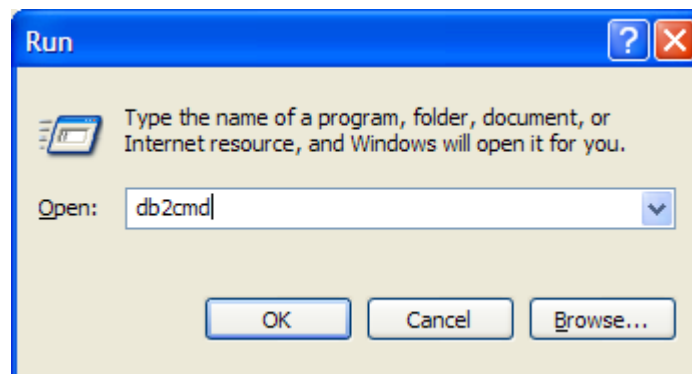
4. Then to run the script, enter these commands in the Command Window:

```
cd C:\express
create_database.bat db2admin ibmdb2
```

5. Take a moment to familiarize yourself with the script you just created. Do you understand what is happening on each line?

6. Try to answer the following questions:

   A. Where is the database connection established?

   B. What do the `%1` and `%2` mean?

   C. What does the following line of code do? Where is it used? For what?

      ```
      SET DBPATH=C:
      ```

   D. What does the following line of code do?

      ```
      del schema.log, triggers.log, app_objects.log
      ```

   E. What happens when the script is called without any parameters?

   F. Why don't the SQL scripts contain CONNECT TO statements? How do they connect to the database?

PART II – LEARNING DB2: DATABASE ADMINISTRATION

6

Chapter 6 – DB2 Architecture

In this chapter we briefly discuss the DB2 architecture. You will learn about:

- The DB2 process model

- The DB2 memory model

- The DB2 storage model

Note:

For more information about the DB2 architecture, watch this video:
http://www.channeldb2.com/video/video/show?id=807741:Video:4482

6.1 DB2 process model

Figure 6.1 depicts the DB2 Process Model. In this figure, rectangles represent processes while ellipses represent threads. The main DB2 process is called db2sysc. Under this process there are several threads, the main one is also called the db2sysc. This is the main thread that spawns other threads. When a remote application tries to connect to the server using an SQL CONNECT statement, the remote listeners for the communication protocol will receive this request and contact a DB2 coordinator agent (db2agent). A DB2 agent is like a little worker that performs operations on behalf of DB2. When the application is local, that is, running on the same server as DB2, the steps are very similar, only that a db2ipccm agent handles the request instead of the db2tcpcm thread. In some cases, such as when parallelism is enabled, a db2agent may spawn other agents which appear as db2agntp threads. Other agents shown in the figure such as db2pfchr, db2loggr, db2dlock may also be used for different purposes. Most common processes are described in *Table 6.1*, and most common threads are described in *Table 6.2*.
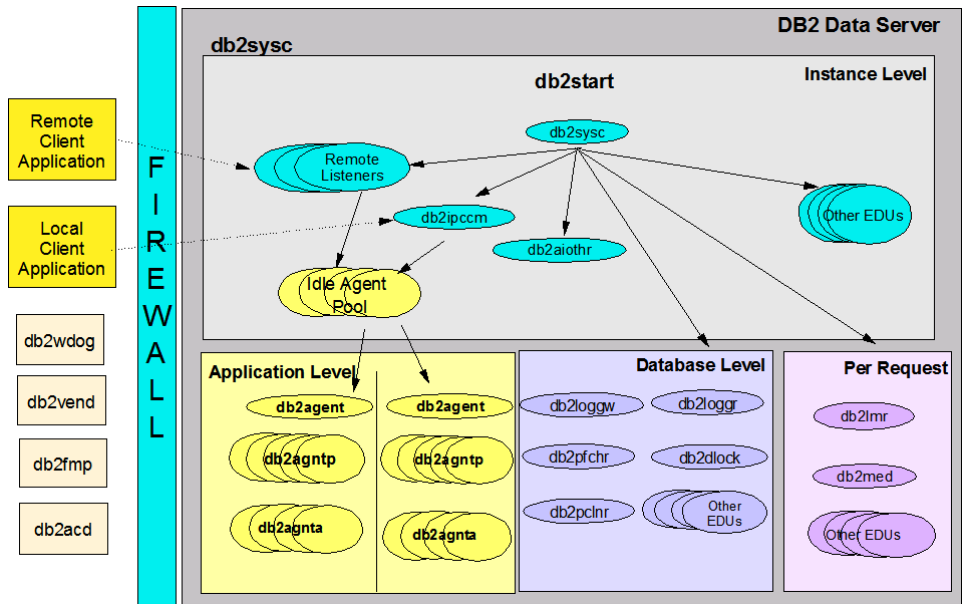
Figure 6.1 – The DB2 Process Model

Process Name	Description
db2sysc (Linux) db2syscs (Win)	The main DB2 system controller or engine. Starting in DB2 9.5, there is only one multi-threaded main engine process for the entire partition. All Engine Dispatchable Units (EDUs) are threads inside this process. Without this process, the database server cannot function.
db2acd	The autonomic computing daemon. It is used to perform client side automatic tasks, such as health monitor, automatic maintenance utilities, and the admin scheduler. This process was formerly called db2hmon.
db2wdog	The DB2 watchdog. The watchdog is the parent of the main engine process, db2sysc. It cleans up resources if the db2sysc process abnormally terminates.
db2vend	The fenced vendor process introduced in DB2 9.5 All 3rd party vendor code runs in this process outside of the engine. 3rd party vendor applications are non-IBM programs that can interact with DB2; for example, log archiving can be managed by a 3rd party vendor code by specifying a user exit routine parameter to point to this code.
db2fmp	Fenced processes that run user code on the server outside the firewall for both stored procedures and user defined functions. This process replaces both the db2udf and db2dari processes that were used in

	previous versions of DB2.

Table 6.1 – Common DB2 processes

Thread Name	Description
db2sysc	The system controller thread. This thread is responsible for the start-up and shut-down and the management of the running instance
db2tcpcm	TCP/IP communication listener
db2agent	Coordinator agent that performs database operations on behalf of applications (at least 1 per connection, depending if Connection Concentrator is enabled).
db2agntp	Active subagent spawned if INTRA_PARALLEL is set to YES. This thread performs database operations for the application. db2agent will coordinate the work between the different db2agntp subagents.
db2pfchr	DB2 asynchronous I/O data prefetcher (NUM_IOSERVERS)
db2pclnr	DB2 asynchronous I/O data writer (NUM_IOCLEANERS)

Table 6.2 – Common DB2 threads

6.2 DB2 memory model

The DB2 memory model consists of different areas in memory at the instance level, database level, and application and agent level as shown in *Figure 6.2*. We will not explain in detail each of the different areas in memory in this book, but just provide a brief overview.
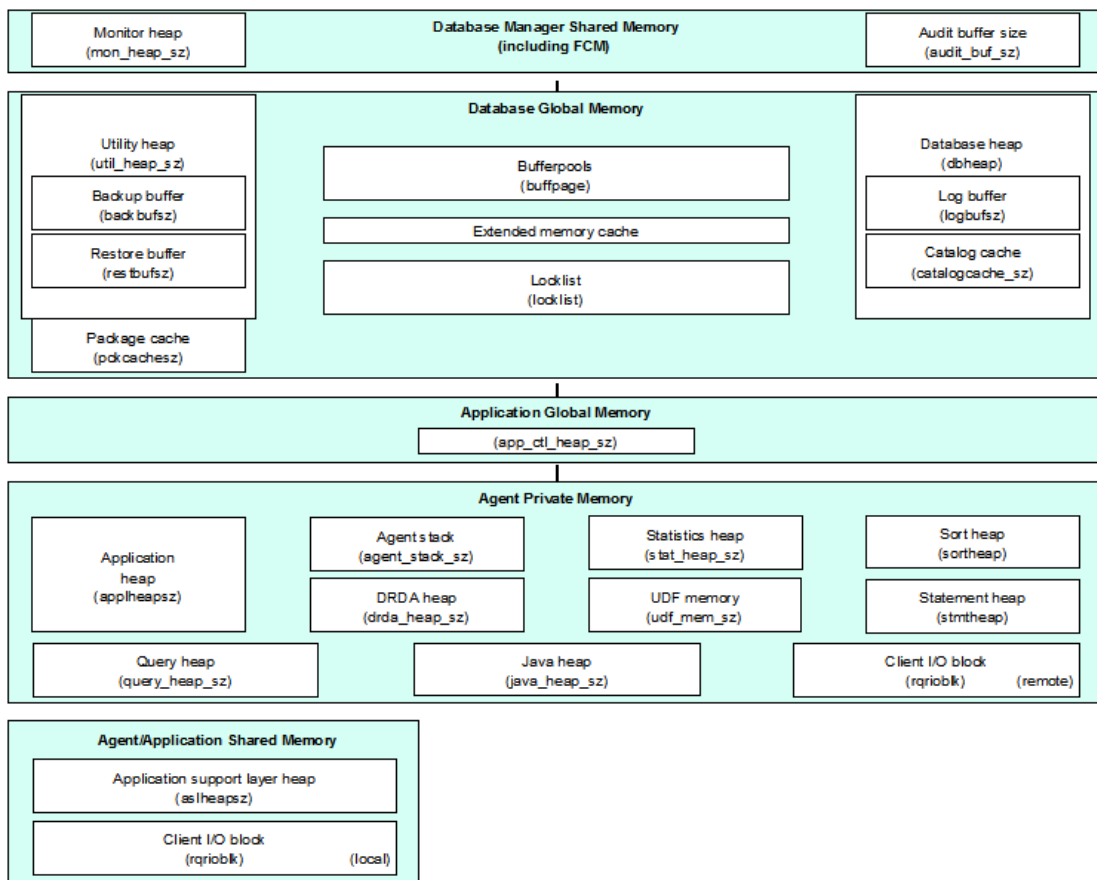
Figure 6.2 – The DB2 memory model

When an instance is started, the database manager shared memory is allocated. This normally does not take much space. When you first connect to a database, the Database Global Memory is allocated. In this block, the buffer pool is one of the most important parts, especially for improving query performance. The size of the buffer pools will determine the size of the entire Database Global Memory.

Agent private memory is the memory used by each DB2 agent. Without using the connection concentrator, each connection requires one agent. Typically an agent uses approximately 3 to 5 MB. With the connection concentrator, several connections can use one agent, therefore reducing the need for more physical memory.

6.3 DB2 storage model

In this section we will describe the following concepts:

- Pages and Extents
- Buffer pool
- Table space

6.3.1 Pages and Extents

A page is the minimum unit of storage in DB2. Allowed page sizes are: 4K, 8K, 16K and 32K. An extent is a grouping of pages. Working with one page at a time in DB2 would be costly in terms of performance; therefore, DB2 works with extents at a time instead. The page size and extent size are defined when working with buffer pools and table spaces as we will see in the next sections.

6.3.2 Buffer pools

A buffer pool is a real memory cache for table and index data. It improves performance by reducing direct sequential I/O and it promotes asynchronous reading (pre-fetching) and writing. That is to say, DB2 anticipates what pages will be needed and pre-fetches them from the disk to the buffer pool so they are ready to use.

Buffer pools are allocated in memory units of 4K, 8K, 16K, and 32K pages. There should be at least one buffer pool per database, and at least one matching buffer pool for a table space of a given page size.

6.3.2.1 Creating a Buffer Pool

To create a buffer pool you can use the **CREATE BUFFERPOOL** statement. Alternatively, using the Control Center you can right click on the Buffer Pool folder within a given database and choose *Create,* as shown in *Figure 6.3*
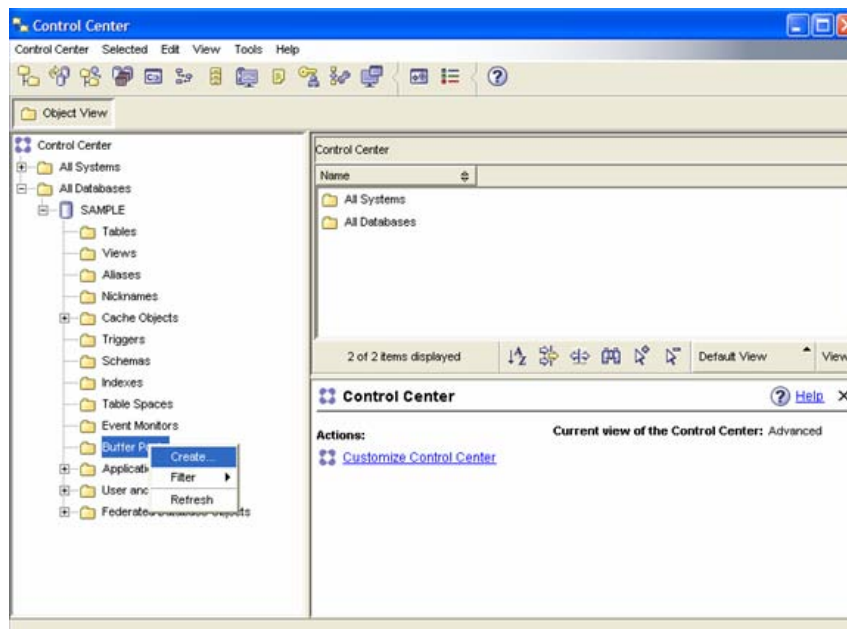
Figure 6.3 – Creating a buffer pool

After clicking on Create, the Create Buffer Pool Dialog will appear as shown in *Figure 6.4*
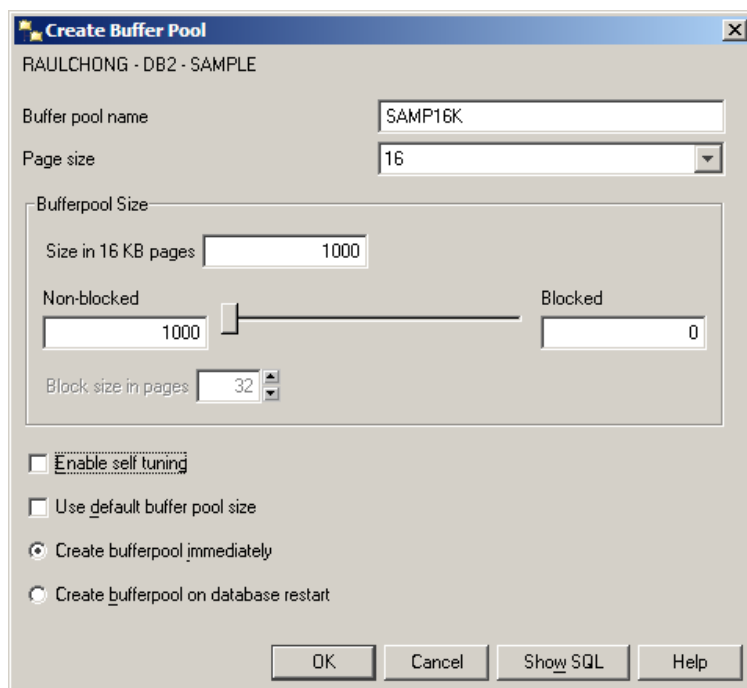
Figure 6.4 – Create a buffer pool dialog box

Most entries in *Figure 6.4* are self explanatory. The fields *Non-blocked* and *Blocked* refer to the number of pages that should exist as non-blocked and as blocked. Blocked-based buffer pools ensure that contiguous pages on disk are moved to the buffer pool also contiguously in a blocked area; this may improve performance. The number of pages must not be greater than 98 percent of the number of pages for the buffer pool. Specifying the value as 0 disables block I/O.

Once the buffer pool has been created, it is displayed in the Control Center, as shown in *Figure 6.5.*
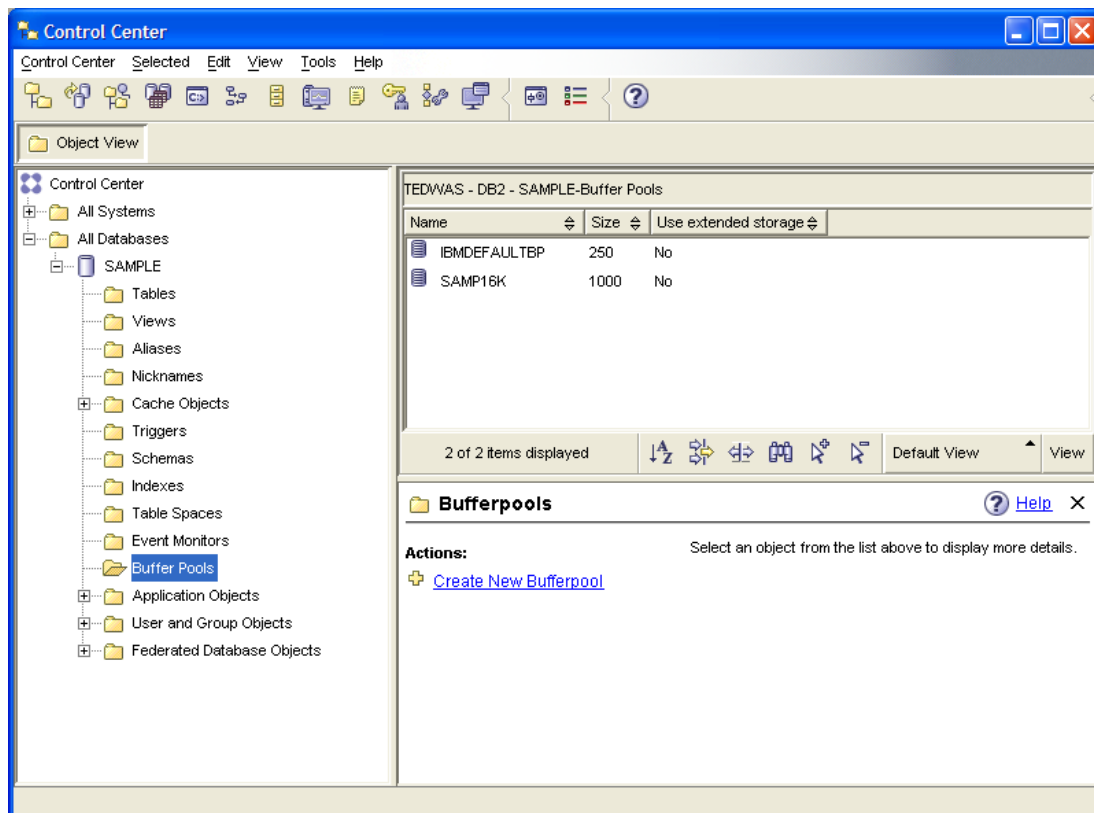
Figure 6.5 – The Control Center after the creation of buffer pool SAMP16K

6.3.3 Table spaces

Table spaces are a logical interface between logical tables and the system's physical memory (buffer pool) and containers (disks). Use the **CREATE TABLESPACE** statement to create a table space where you can specify:

- The page size for the table space (4KB, 8KB, 16KB, or 32KB). The page size must correspond to a buffer pool with the same page size.

- The buffer pool name associated to this table space.

- An extent size

- A pre-fetch size.

6.3.3.1 Table space types

There are three types of table spaces:

- Regular

These are for user tables. For example, the USERSPACE1 table space created by default is a regular table space.

- Large

   These are used to optionally separate LOB data into its own table space. It is also used for storing XML data for databases created with pureXML support using the XML data type for columns. Large table spaces are the default.

- Temporary

   There are two types of temporary table spaces:

   - System temporary

      These are used by DB2 for internal operations, such as sorts. For example, the TEMPSPACE1 table space, created by default when you create a database, is a system temporary table space. There must always be at least one system temporary table space.

   - User temporary

      These are used to create user-defined Declared Global Temporary Tables (DGTTs) and Create Global Temporary Tables (CGTTs) which are temporary in-memory tables. They are often confused with <u>system</u> temporary table spaces. Users must create a user temporary table space before DGTTs or CGTTs can be used.

6.3.3.2 Table space management

Table spaces can be classified based on how they are managed. This can be specified in the **CREATE TABLESPACE** statement:.

Managed by system

This type of table space is known as <u>S</u>ystem <u>M</u>anaged <u>S</u>torage (SMS). This means the operating system manages the storage. They are easy to manage, and the containers are file system directories. The space is not pre-allocated, but the files grow dynamically. Once you specify the containers, these are fixed at creation time and other containers cannot be added later, unless a redirected restore is used. When using SMS table spaces the table data, index and LOB data cannot be spread across different table spaces.

Managed by database

This type of table space is known as <u>D</u>atabase <u>M</u>anaged <u>S</u>torage (DMS). This means that DB2 manages the storage. Management of the space requires more manual intervention from a DBA. Containers can be pre-allocated files or raw devices. For raw devices, data is written directly without O/S caching.

Containers can be added, dropped or resized using the ALTER TABLESPACE statement. DMS table spaces are best for performance, and table data, index, and LOB data can be split into separate table spaces, which improves performance.

Managed by automatic storage

This type of table space is managed by automatic storage, and can benefit from the ease of use similar to SMS table spaces, but with the best performance and flexibility of DMS table spaces. Therefore, starting with DB2 9, this is the default type of table space. For these table spaces, a user first specifies a storage path and a logical group of storage devices which DB2 will use to manage the space. No explicit container definitions are provided. Containers are automatically created across the storage paths. Growth of existing containers and addition of new ones is completely managed by DB2. When a storage path is not specified in the `CREATE DATABASE` command, the database path is used as the storage path. The database path is where the main database definition resides. If the database path is not specified, it's obtained from the database manager configuration parameter DFTDBPATH. On Windows, this can only be a drive, not a path.

To allow for automatic storage, you first need to create a database with automatic storage enabled (this is the default behavior) and associate a set of storage paths with it. After creation, if needed, you can redefine the storage paths using a database RESTORE operation. Then, you can create table spaces to use automatic storage (again, this is the default behavior).

Managed by automatic storage is very similar to DMS table spaces, but operations have been automated so they are managed by DB2; this includes the assignment and allocation of containers and auto resizing.

Let's take a look at an example of a Managed by automatic storage table space. First create the database with automatic storage enabled, as in these examples:

Automatic storage is enabled by default:

```
CREATE DATABASE DB1
```

Automatic storage is explicitly specified:

```
CREATE DATABASE DB1 AUTOMATIC STORAGE YES
```

Automatic storage is enabled by default, but the storage paths are indicated. If the storage path is a directory, it must be created ahead of time:

Example on Windows:
```
CREATE DATABASE DB1 ON C:/, C:/storagepath1, D:/storagepath2
```

Note that the first item in the list is a drive, because it is representing a database path which can only be a drive, not a path on Windows. This item will also be used as one of the storage paths. Thus, the database path is C:, and the storage paths consist of C:, C:\storagepath1, and D:\storagepath2, where the last two directories must be created ahead of time.

Example on Linux:

```
CREATE DATABASE DB1 ON /data/path1, /data/path2
```

Automatic storage is disabled explicitly:

```
CREATE DATABASE DB1 AUTOMATIC STORAGE NO
```

Next, create the table space with automatic storage enabled as in these examples:

Automatic storage for table spaces is also enabled by default:

```
CREATE TEMPORARY TABLESPACE TEMPTS
```

Automatic storage is explicitly specified for the table space:

```
CREATE TABLESPACE TS2 MANAGED BY AUTOMATIC STORAGE
```

Automatic storage is implicitly specified, the initial size is allocated, along with how much it will increase, and the maximum size it can increase.

```
CREATE TABLESPACE TS1
    INITIALSIZE 500 K
    INCREASESIZE 100 K
    MAXSIZE 100 M
```

6.3.3.3 How data is stored in table spaces

By default, DB2 will write to disk extents at a time striped across containers. For example, if you have a 4K table space with an extent size of 8 using 3 raw containers on a DMS table space, this means that 32K of data (4K x 8 pages per extent = 32K) will be written to one disk before writing to the next. This is shown in *Figure 6.6.* Note that tables do not share extents.
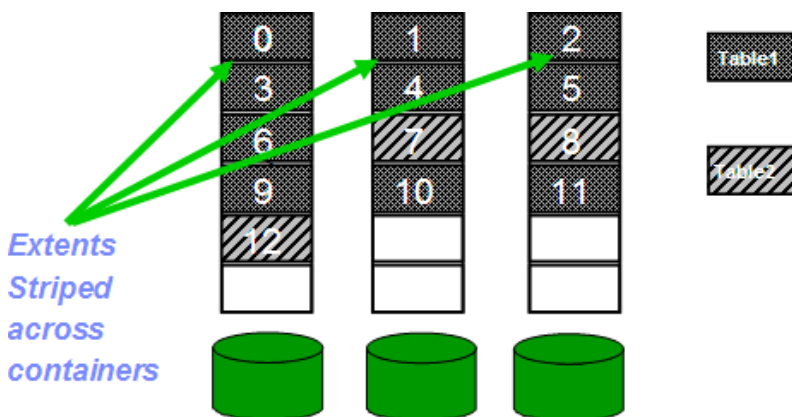
Figure 6.6 – Writing data in table spaces

6.3.3.4 Creating a Table Space using the Control Center

To create a table space from the Control Center, right click on the *Table Spaces* folder within a given database and choose *Create* as shown in *Figure 6.7*. The *Create table space* wizard will appear, as shown in *Figure 6.8*.
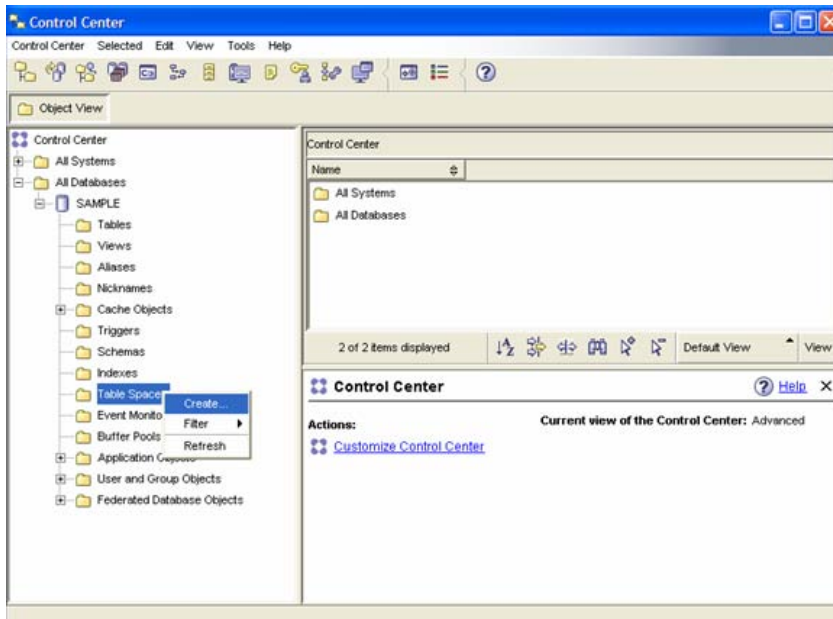
Figure 6.7 – Creating a Table Space from the Control Center
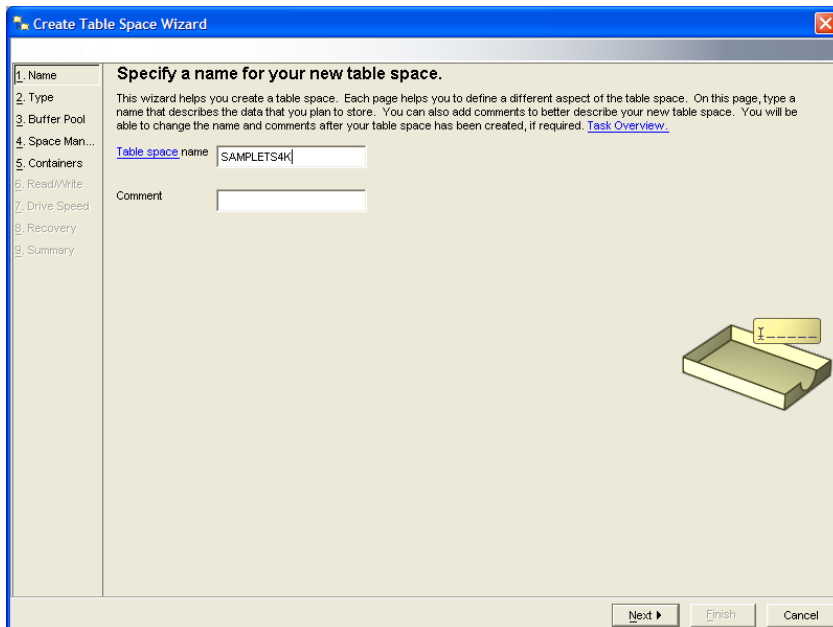
Figure 6.8 – Create Table Space Wizard

The wizard shown in *Figure 6.8* will guide you through the steps of creating a table space.

6.4 Summary

In this chapter we explored the three key aspects of DB2 architecture: the process model, the memory model, and the storage model. For the process model, we looked at the common processes and threads, including db2sysc, without which DB2 could not run.

The storage model was discussed in depth, covering the three most important aspects: pages and extents, the bufferpool (including creation details) and table spaces. Finally, we looked at the various types of table spaces, along with details on how they are managed (SMS, DMS, Automatic) and how to create a new table space using the Control Center.

6.5 Exercises

This exercise will help you understand the DB2 process model, memory model and storage model on Windows. You will review different processes and threads, monitor memory usage, and practice creating a database that uses automatic storage and storage paths on Windows. Ideally storage paths would be created across disks (drives), but since your computer may not have been configured with multiple disks, this exercise only uses your C:\ drive.

<u>Procedure</u>:

1. Let's take a look at some processes on Windows. First of all, open the DB2 Command Window (*Start -> run -> db2cmd*) and ensure your instance is stopped by issuing this command: **db2stop force**

2. Open the Windows Task Manager, choose the *Processes* tab, click on the *Image Name* column to sort by this column, and look for the db2sysc.exe process, as shown in the figure below.
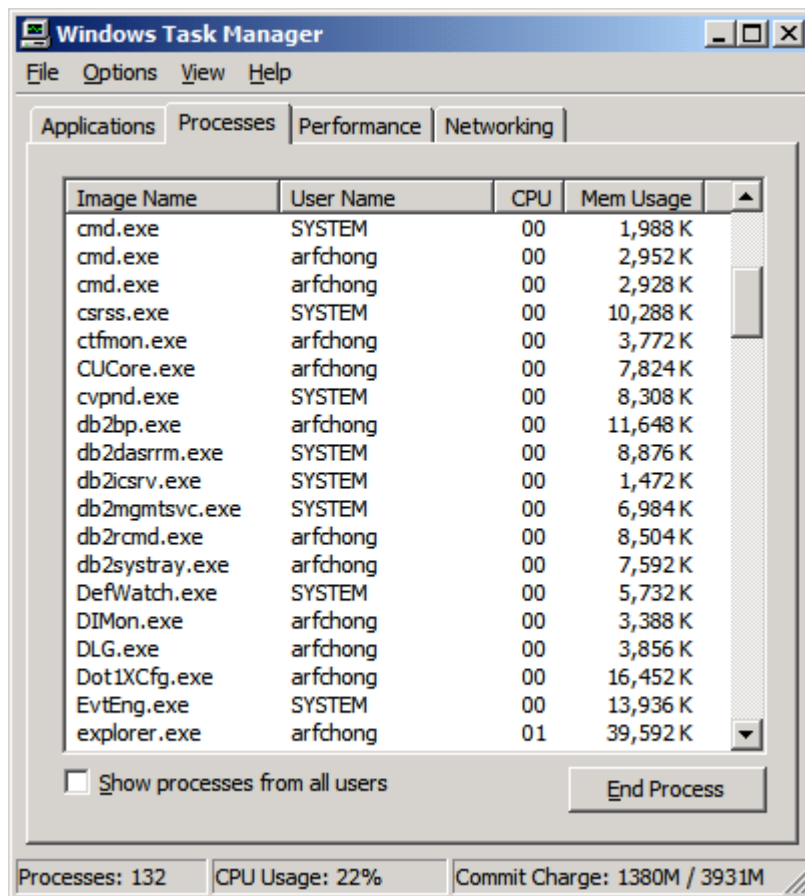
3. You should not find the process db2syscs.exe because we asked you in step one to stop the DB2 instance.

4. Start the DB2 instance with this command: **db2start**, and repeat the previous step. Now can you find the db2syscs.exe process?

5. Let's now look at CPU and memory consumption. Follow these steps:

    G. Ensure nothing is running in your system by closing all other applications

    H. Open a new DB2 Command Window and issue: **db2stop force**

    I. From the Task Manager, switch to the Performance Tab

    J. Keep the Task Manager and the DB2 Command Window open and side by side as shown below. Write down the amount of Physical Memory available and the CPU usage.
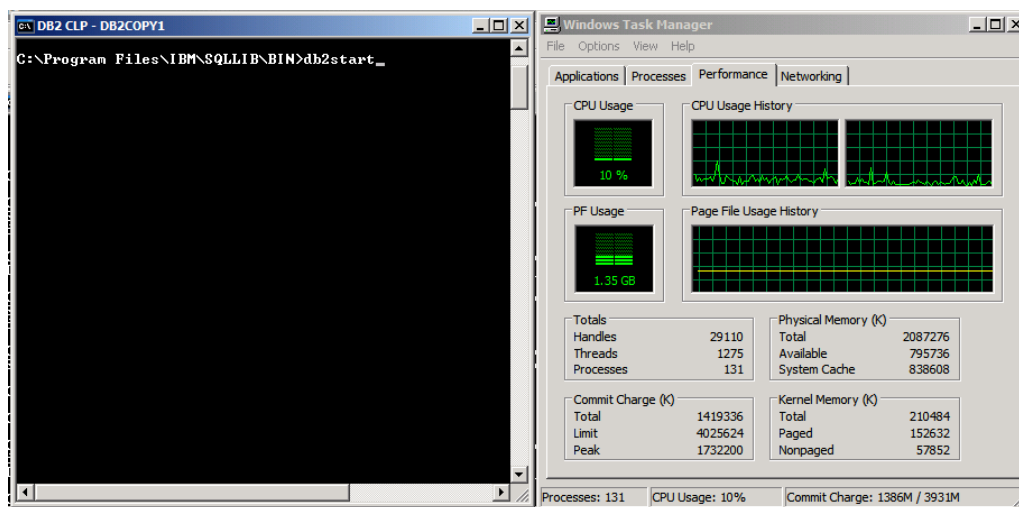
K. Issue **db2start** and at the same time monitor the CPU and memory usage as soon as you execute this command. You should see a brief spike in CPU usage, and that the memory available is reduced by about 50MB to 70MB. This is the amount of memory that the DB2 instance is consuming. If you issue a **db2stop force** again, the memory should return to its previous value.
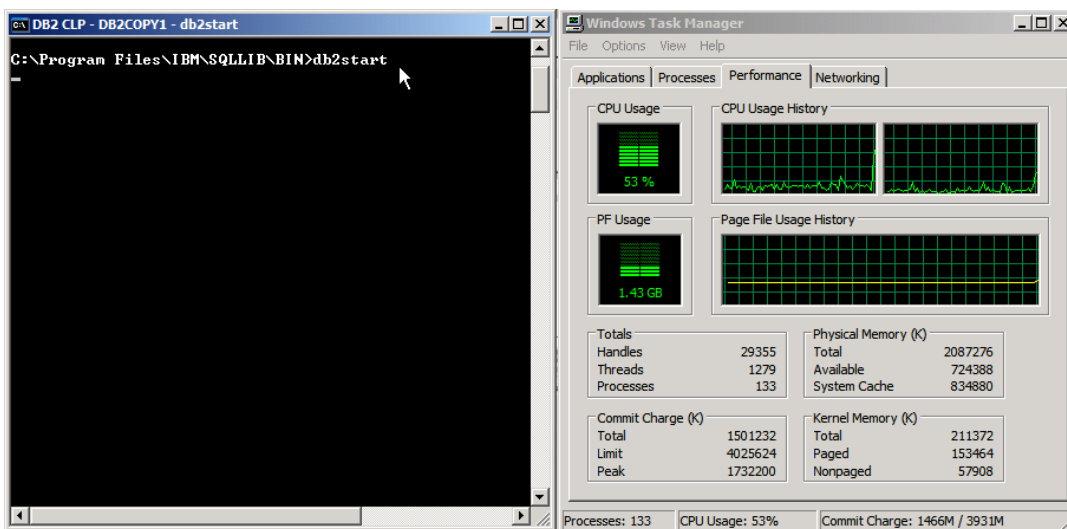
6. Repeat the previous step, but this time, monitor what happens after connecting to the SAMPLE database. Issue these commands from the DB2 Command Window:

```
db2start
db2 connect to sample
```

As soon as you connect to the SAMPLE database, you will see a reduction in the amount of physical memory available. This is because as soon as you connect to a database, the database global memory (bufferpool, catalog cache, etc) is allocated.

7. Repeat the previous step, but this time monitor what happens after creating a bufferpool of any size. Make sure you don't exceed the physical memory in your computer. If you do, DB2 will not allocate the bufferpool immediately, but will defer the creation until the database is deactivated. In addition, a small system bufferpool will be used instead, and DB2 will keep using it until there is enough memory. For example, to create a bufferpool of about 160MB issue this command while connected to the SAMPLE database:

```
db2 create bufferpool mybp immediate size 5000 pagesize 32k
```

8. Create a database *mydb1* that uses automatic storage, and where the database path is drive C:, and the storage paths are C:, C:\mystorage1, C:\mystorage2. Issue the following from the DB2 Command Window:

```
db2 create database mydb1 on C:\, C:\mystorage1, C:\mystorage2
```

If you issued the above command, you probably received an error because you must first create the directories C:\mystorage1, C:\mystorage2. Create them and try again!

7

Chapter 7 – DB2 Client Connectivity

This chapter covers the setup required to connect from a DB2 client to a DB2 server using TCP/IP. Note that a DB2 server comes with a client component, so a DB2 server can also behave as a client to connect to another DB2 server. There are several ways to set up DB2 client connectivity; however, in this chapter we discuss only the easiest method which is using the Configuration Assistant.

> **Note**:
>
> For more information about the DB2 Client Connectivity, watch this video: http://www.channeldb2.com/video/video/show?id=807741:Video:4222
>
> Starting with DB2 9.7, the Configuration Assistant has been deprecated; however, it can still be used and is included with the product.

7.1 DB2 Directories

DB2 directories are binary files that store information about which databases you can connect to from your machine. There are four directories:

- System database directory
- Local database directory
- Node directory
- DCS directory

Reviewing and updating the contents of all of these directories can be performed through the Configuration Assistant GUI tool.

7.1.1 System database directory

This directory is like a table of contents of a book. It shows all the databases, whether they are local or remote, that you can connect to. For a local database, it will have a pointer to the *Local database directory*. For a remote database, it will have a pointer to an entry in the *Node directory*. To review the contents of this directory issue the command:

```
list db directory
```

7.1.2 Local database directory

This directory contains information about databases that you can connect to and that reside on your machine. To review the contents of this directory, issue the command:

```
list db directory on <drive/path>
```

7.1.3 Node directory

This directory includes information about how to connect to a given remote database. For example, if the TCP/IP protocol is used, a TCP/IP node entry would include the IP address of the server where the DB2 database you are trying to connect resides, and the port of the instance where this database resides. To review the contents of this directory, issue the command:

```
list node directory
```

7.1.4 DCS directory

This directory will only appear if you have installed the DB2 Connect software to connect to DB2 for z/OS (mainframe), or DB2 for i5/OS. To review the contents of this directory, issue the command:

```
list dcs directory
```

7.2 Configuration Assistant (deprecated)

Using the Configuration Assistant GUI tool, you can easily configure connectivity between a DB2 client and a DB2 server. To launch the Configuration Assistant on Windows, you can choose: *Start -> Programs -> IBM DB2 -> DB2COPY1 -> Set-up Tools -> Configuration Assistant*.

From the Command line, you can start the tool using the command **db2ca**. *Figure 7.1* shows the Configuration Assistant.
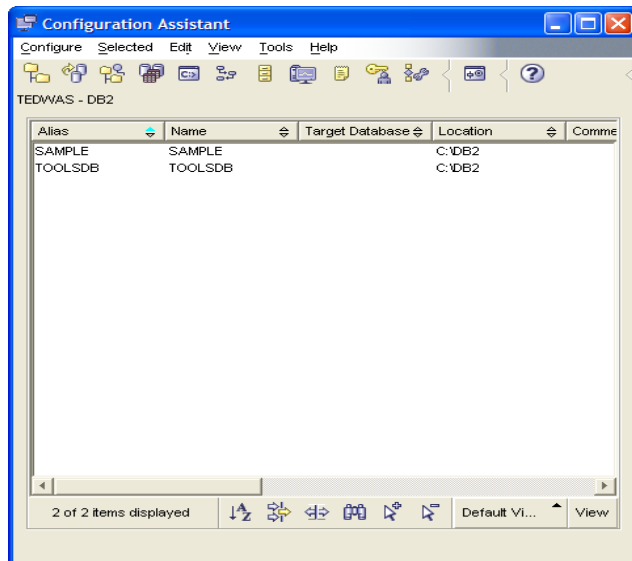
Figure 7.1 – The Configuration Assistant

7.2.1 Setup required at the server

There are two things that need to be set up at the server:

1) DB2COMM

This DB2 registry variable determines which communication protocol listeners should be monitoring requests from clients. Typically TCP/IP is the communication protocol most often used. Changing this parameter requires an instance re-start. To review and change the value of DB2COMM in the Configuration Assistant, choose *Configure -> DB2 Registry* as shown in *Figure 7.2* and *Figure 7.3*.
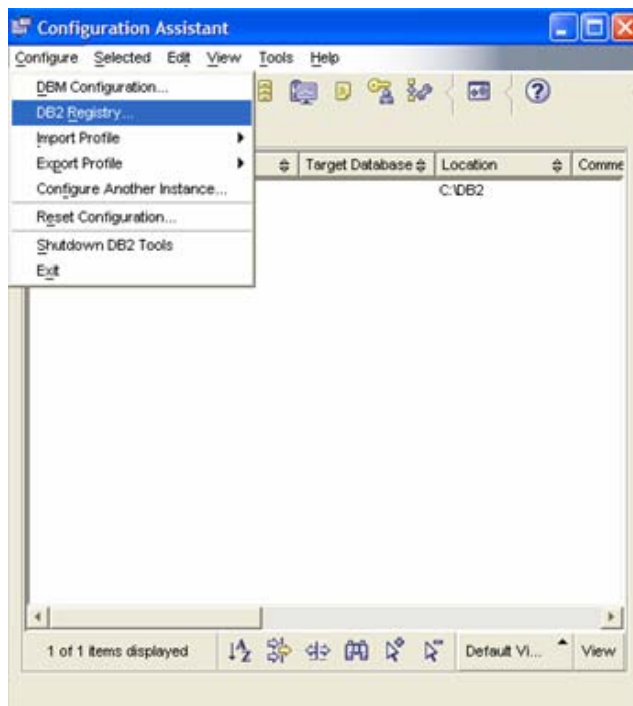
Figure 7.2 – Accessing the DB2 Registry
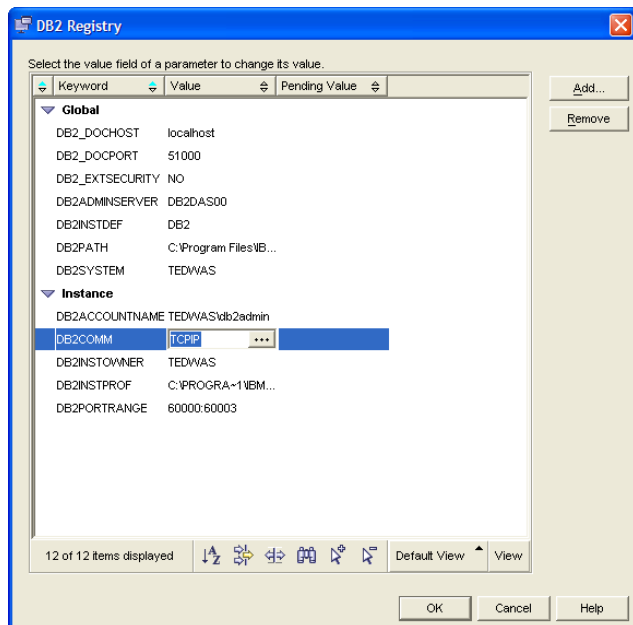
Figure 7.3 –Verifying the value of the DB2COMM DB2 Registry variable

2) SVCENAME

This database manager configuration parameter should be set to the service name (as defined in the TCP/IP services file) or to the port number to use when you want to access databases of this instance. From the Configuration Assistant, choose *Configure -> DBM configuration* as shown in *Figure 7.4*
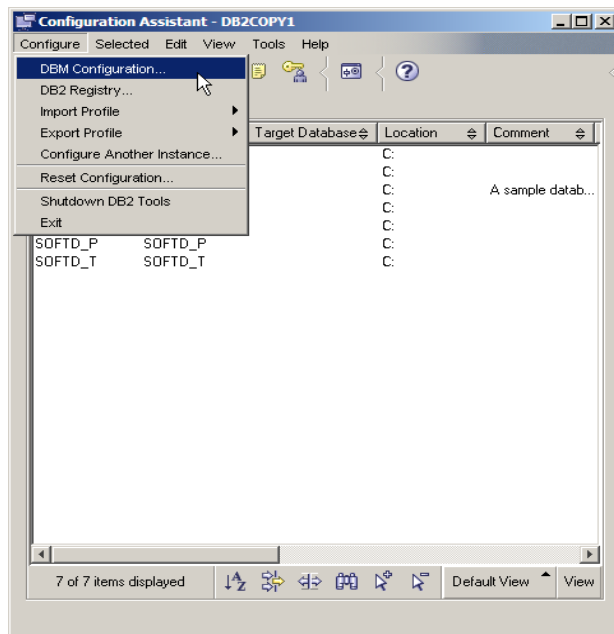
Figure 7.4 –Reviewing the dbm cfg from the Configuration Assistant

Once you are in the DBM Configuration window, find the Communications section, and look for SVCENAME. You can change the value to a string or even to a port number if needed. This is shown in *Figure 7.5.*
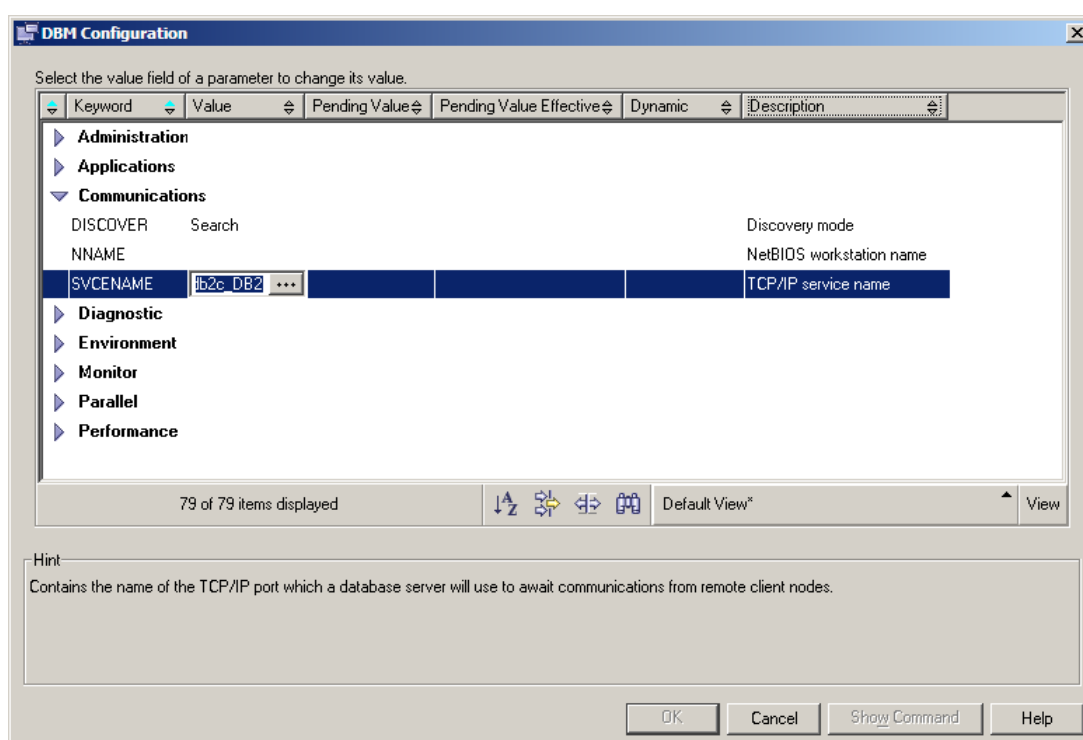
Figure 7.5 –Reviewing the SVCENAME dbm cfg parameter

7.2.2 Setup required at the client

At the client, you need to know this information beforehand:

- The name of the database you want to connect to

- The port number of the DB2 instance at the server where the database resides. You can also use a service name, as long as there is a matching entry in the TCP/IP services file

- The operating system user ID and password to connect to the database. This user ID must have been previously defined at the server

The above information can be input from the DB2 client using the Configuration Assistant. First, launch the *Add Database* Wizard by clicking on the *Selected -> Add Database Using Wizard* choice, as shown in *Figure 7.6*
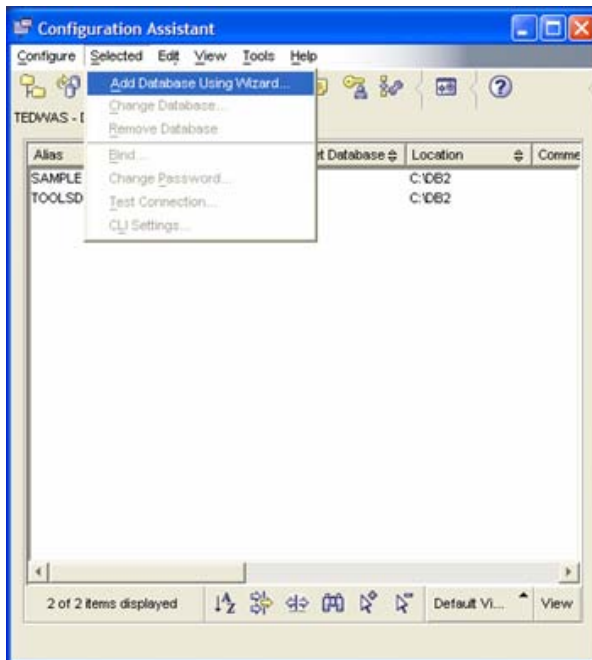
Figure 7.6 – Invoking the Add Database Wizard

You can also get to this wizard by right clicking on the white space in the Configuration Assistant and choosing *Add Database Using Wizard*. *Figure 7.7* shows the Add Database Wizard.
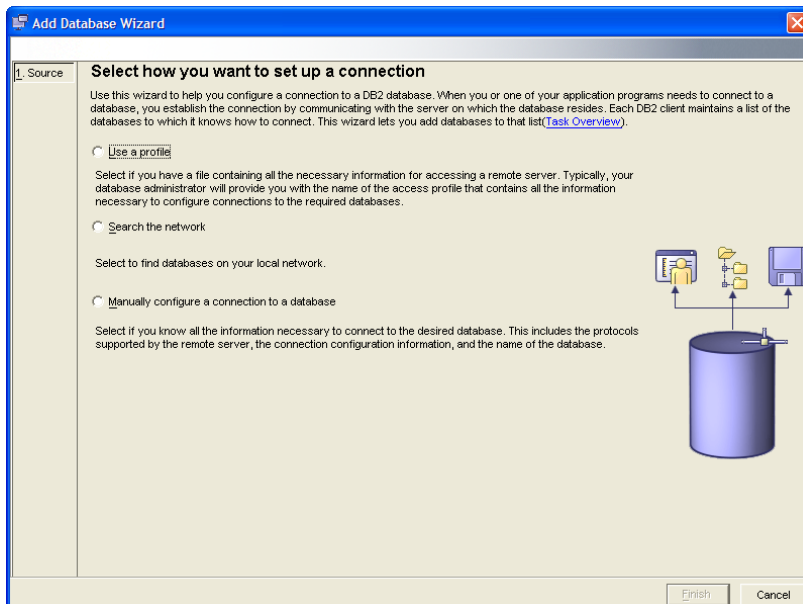
Figure 7.7 –Add Database Wizard

In the Add Database Wizard, there are three options:

1. Use a Profile

There may be situations when you need to configure many clients to connect to the same DB2 server. In these situations, it is convenient to perform all configurations from one client, and store these configurations into a "profile" file. With this file, you can load all the information directly to other clients. In *Figure 7.7*, if you choose *Use a Profile* you would be loading the information from an existing "profile". More details are provided later in this chapter describing how to create client and server profiles.

2. Search the network

This method, also known as *Discovery*, tells DB2 to search the network for a given server, instance, and database. For this method to work, the DAS must be running on each DB2 server where databases are to be discovered. With this method, there are two ways to perform the search:

- Search:

  Search the entire network. This is not recommended if your network is large and with many hubs, as it would take a long time to retrieve data from every system

- Known:

  Search the network for a known server at an address you provide.

  The two methods are illustrated in *Figure 7.8*
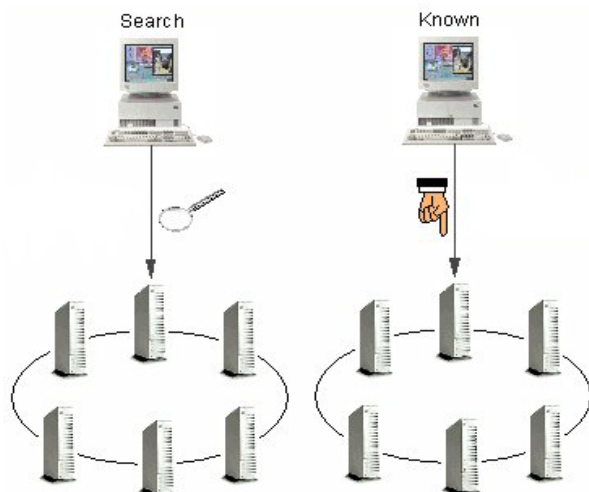
Figure 7.8 –The Search and Known search (or Discovery) methods

There may be circumstances when an administrator would not like clients to search the network for databases with confidential information. This can be prevented at the DAS, the instance or the database level. *Figure 7.9* provides details about this.
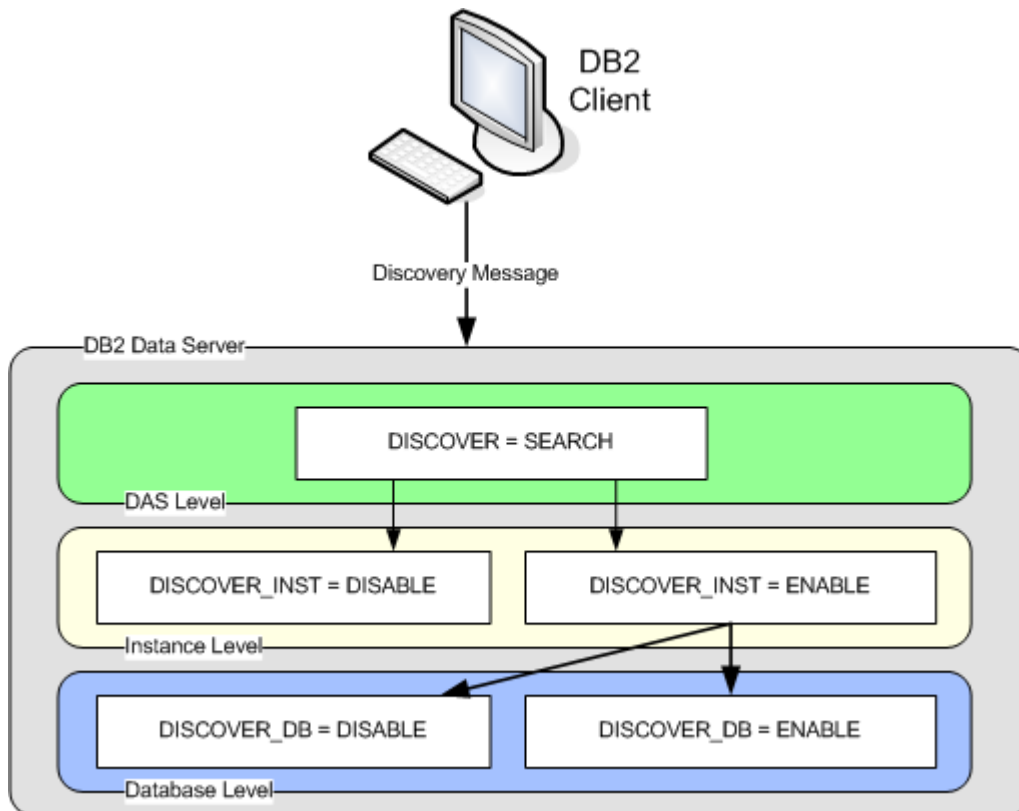
Figure 7.9 – Configuring parameters to allow for discovery

Figure 7.9 shows the different levels where you can enable or disable discovery. At the DAS level, you can give the DISCOVER parameter a value of SEARCH or KNOWN. At the instance level, the DISCOVER_INST database manager configuration parameter can be set to DISABLE or ENABLE. Finally, at the database level, the DISCOVER_DB parameter can also be set to ENABLE or DISABLE. Setting these parameters accordingly provides you granularity for database discovery.

3. Manually configure a connection to a database

Using this method, you manually add host name, port numbers and database information to the Configuration Assistant, which will then generate catalog commands to execute the connectivity configuration. The Configuration Assistant will not check that the information is correct. You will know it is incorrect if you cannot connect to a server. Also, ensure the user ID and password you provide to connect to the remote database is correct. By default the authentication takes place on the DB2 server you are trying to connect to, therefore, you must provide a user ID and password defined on that server.

7.2.3 Creating Client and Server Profiles

If you are configuring a large number of servers or clients, rather than set up each one individually, you can set up one, then export a profile from it, and then apply the profile to

the other clients/servers. This saves a lot of administration time when setting up the environment.

To create a customized profile from the Configuration Assistant, click on the *Configure* Menu, then select *Export Profile -> Customize*, as shown in *Figure 7.10*
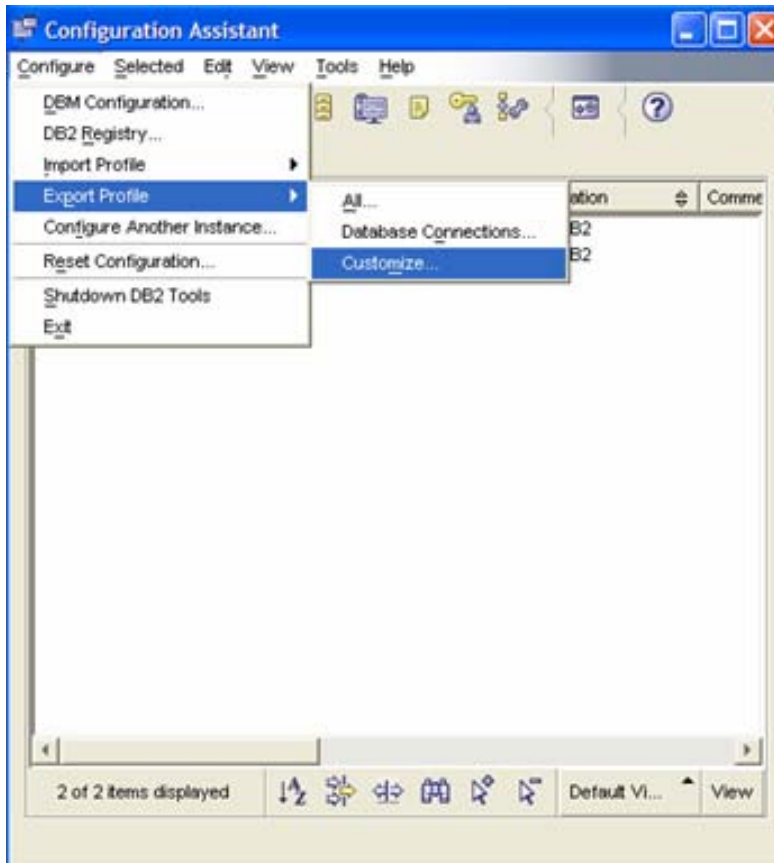
Figure 7.10 – Exporting a Profile

Figure 7.11 shows the fields that need to be completed to export a profile
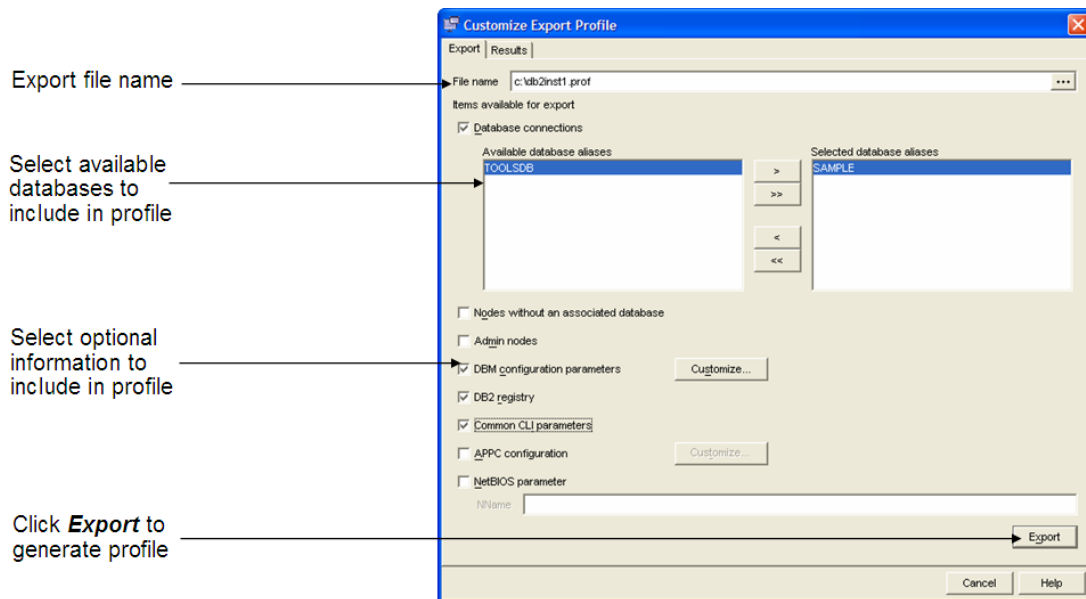
Figure 7.11 – Customize Export Profile dialog

Figure 7.12 show the results after clicking "Export" in the Customize Export Profile dialog.
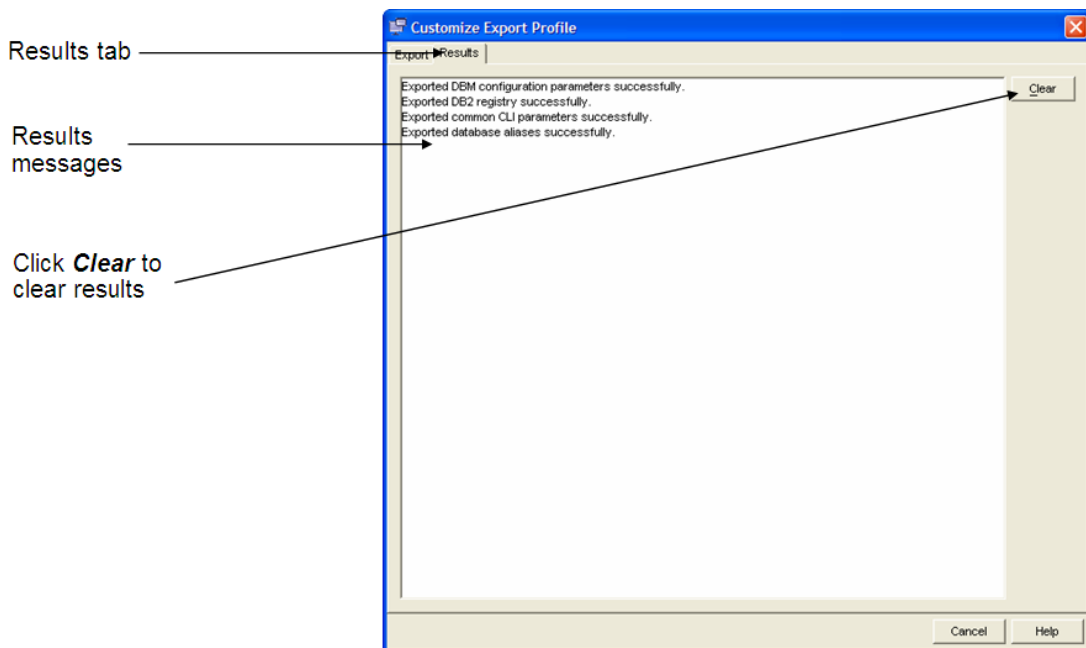
Figure 7.12 – Export Profile results

To import a customized profile from the Configuration Assistant, click on the *Configure* Menu, then select *Import Profile -> Customize*, as shown in *Figure 7.13*
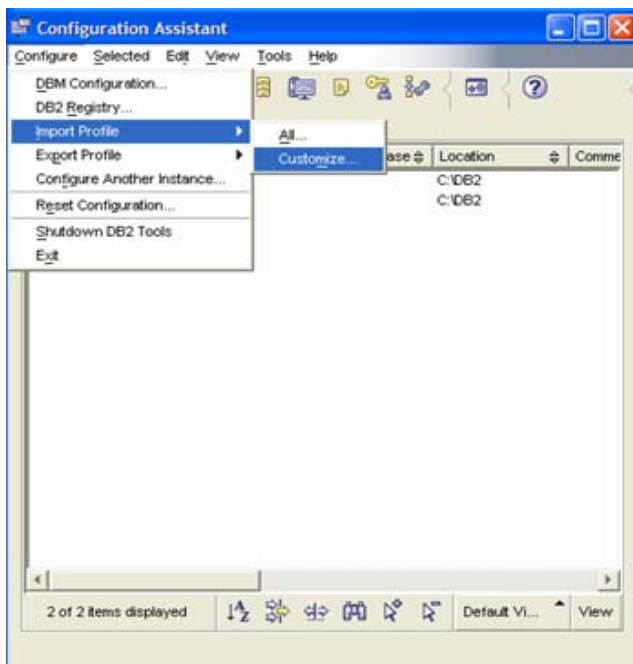
Figure 7.13 – Importing a profile

Figure 7.14 shows the fields that need to be completed to import a profile
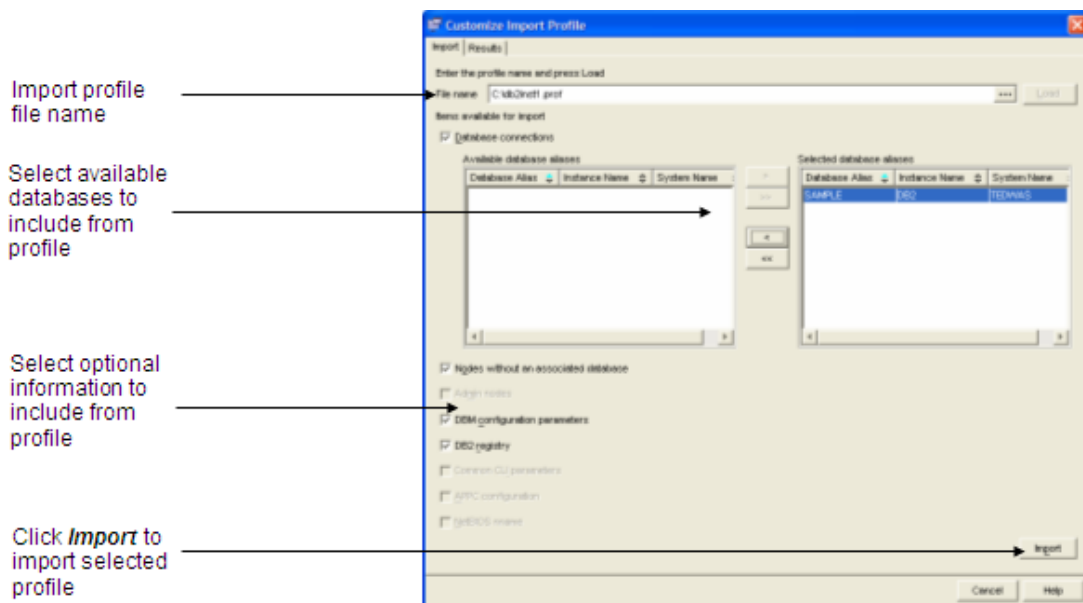
Figure 7.14 – Customize Import Profile

7.3 Summary

Connecting from a data client to a server is a core aspect of relational database management. In this chapter, we looked at client connectivity, starting with purpose and contents of the database and node directories associated with DB2.

Next, we discussed using the Configuration Assistant GUI to set up the client server connection, including what needs to be configured on both sides of the connection.

We also looked at how the Add Database Wizard can also be used to make a server connection through one of three methods: using a stored profile, searching the network (also known as Discovery), or manually by typing in the server information. Client and server profile creation was then covered in more detail.

7.4 Exercises

The Configuration Assistant can be used to quickly and easily configure remote database connections. In this exercise, you will catalog a database residing on a remote DB2 server (represented by your neighbor's workstation, using both Search and Discover modes. Once the database is cataloged, you will be able to access it as if it were on your local system. DB2 performs all the communication processes "under the covers".

This exercise assumes you are working within a network. If this is not the case, you can always use your own computer as both the client and server machines and follow the instructions for configuration below to connect to your own system.

<u>Procedure</u>

1. Ask your neighbor (or instructor) for the following information:

2. Remote Database Info:

(PR)	Protocol	__TCPIP___
(IP)	IP Address or hostname	_____
(PN)	Instance Port Number	_____
(DB)	Database Name	__SAMPLE__

Hints:

To obtain the hostname on Windows, type **hostname** from a command window

To obtain the IP address on Windows, type **ipconfig** from a command window

3. Open the Configuration Assistant. (Hint: it is accessible through the Start menu).

4. Open the *Selected* menu and select *Add Database Using Wizard*.

5. On the *Source* page of the wizard, select the *Manually Configure a Connection to a Database* option. Click the *Next* button to move to the next page of the wizard.

6. On the *Protocol* page of the wizard, select the TCP/IP option. Click the *Next* button to move to the next page of the wizard.

7. On the *TCP/IP* page of the wizard, enter the full hostname or IP address that you wrote down in step (1). Enter the Port number you wrote down in step (1). Click the *Next* button to move to the next page of the wizard.

8. Note: The option for *Service Name* can be used if you have an entry in the *local* Services file with a port number defined corresponding to the port the remote server instance is listening for. When you use this option, DB2 will look in the services file on the <u>local</u> machine, not on the server. You must add an entry to this file if you want to use this option.

9. On the *Database* page of the wizard, enter the name of the database defined on the remote server that you wrote down in step (1) in the *Database Name* field. Note how the *Database Alias* field is automatically filled out with the same value. The database alias is a name that local applications will use to connect to this database. Since you already have a local database called *SAMPLE* defined, DB2 will not let you catalog another database with the same name. You must therefore use a different alias name. For this example, change the database alias to *SAMPLE1*. You can enter an optional comment about this database if you want. Click the *Next* button to move to the next page of the wizard.

10. On the *Data Source* page of the wizard, you can optionally register this new database (data source) as an ODBC data source. This automatically registers it in the Windows ODBC Manager for you. For this example, un-check the *Register this database for ODBC* since you will not be using ODBC. Click the *Next* button to move to the next page of the wizard.

11. On the *Node Options* page of the wizard, specify the operating system of the server where the remote database is located. Since all workstations in this lab use Microsoft Windows, ensure the *Windows* item in the drop-down list is selected. The Instance name field should be set to *DB2*. If it is not, set its value to *DB2*. Click the *Next* button to move to the next page of the wizard.

12. This *System Options* page of the wizard gives you the opportunity to ensure the system and hostname are correct, and to verify the operating system setting. Click the *Next* button to move to the next page of the wizard.

13. The *Security Options* page of the wizard allows you to specify where you want user authentication to take place and what method you want to use. Select the option *Use authentication value in server's DBM Configuration*. This will use the method specified by the AUTHENTICATION parameter in the remote Instance's configuration file. Click the *Finish* button to catalog the remote database and close the wizard. A confirmation box should appear. Click the *Test Connection* button to ensure you can connect successfully to the database. Also, ensure the username and password you provide is a valid one *defined on the remote server* (since it is likely that the Server's AUTHENTICATION parameter is set to the value SERVER). If the test connection succeeds, then you have successfully cataloged the remote database. If it does not succeed, go back through the wizard and make sure all the

correct values are specified. (Click the *Change* button to go back through the wizard settings).

14. Open Control Center and try viewing the different tables in the newly cataloged remote database.

15. Go back to the Configuration Assistant and try to catalog a different database, this time using *Search the Network* option. Step through the wizard the same way you did for manually configuring the connection. Note that, on large networks, searched discovery could take a long time to return results.

8

Chapter 8 – Working with Database Objects

This chapter discusses database objects such as schemas, tables, views, indexes, sequences, and so on. Some advanced database application objects such as triggers, user defined functions (UDFs) and stored procedures are discussed in *Chapter 14, Introduction to DB2 Application Development.*

> **Note**:
>
> For more information about working with database objects, watch this video:
> http://www.channeldb2.com/video/video/show?id=807741:Video:4242

8.1 Schemas

Schemas are name spaces for a collection of database objects. They are primarily used to:

- Provide an indication of object ownership or relationship to an application
- Logically group related objects together

All DB2 database objects except public synonyms have a two-part fully qualified name; the schema is the first half of that name as shown below:

```
<schema_name>.<object_name>
```

A fully qualified object name must be unique. When you connect to a database and create or reference an object without specifying the schema, DB2 uses the user ID you connected to the database with for the schema name. For example, if you connect to the **SAMPLE** database with user *arfchong*, and create the table *artists* using the following **CREATE TABLE** statement:

```
CREATE TABLE artists …
```

the fully qualified name of the created table is *arfchong.artists*.

You can use the **set schema** statement to set the schema for a session. *Listing 8.1* provides an example.

```
connect to sample user arfchong using mypsw
select * from staff  ## This looks for arfchong.staff
```

```
set schema db2admin
select * from staff  ## This looks for db2admin.staff
```
Listing 8.1 - An example of using the set schema statement

A "contest system" can be used to illustrate the use of schemas. Say a company is running a contest where participants need to create their own tables and perform some SQL operations. All participants are given the same user ID to connect to the database, and the same script to create tables, where all objects are unqualified, that is, they do not have a schema name. After the contestants log on to the contest system, the system generates the schema name based on a timestamp after connection. This way, contestant A will work on a table with the same name as contestant B, but with different schemas, and therefore there would not be a conflict with their work.

8.2 Public synonyms (or aliases)

New with DB2 9.7 is the concept of public synonyms, also known as public aliases. Public synonyms allow you to reference objects without the need to specify a schema. *Listing 8.2* provides an example.

```
connect to sample user arfchong using mypsw
create public synonym raul for table arfchong.staff
select * from raul
select * from arfchong.raul  ## Error
connect to sample user db2admin using psw
select * from raul
```
Listing 8.2 - An example of a public synonym

In *Listing 8.2,* you first connect with user **arfchong** and create the public synonym **raul** which references the table **arfchong.staff**. The synonym itself doesn't use a schema. If you try using one, you will receive an error. Other users like **db2admin** in the example of *Listing 8.2* can also use synonym **raul** which is public.

In the example, if the keyword **public** is not used, the synonym created would be a private synonym. In *Listing 8.3*, let's examine the same example but using a private synonym.

```
connect to sample user arfchong using mypsw
create synonym raul for table arfchong.staff
select * from raul
select * from arfchong.raul  ## OK, it also works
connect to sample user db2admin using psw
select * from raul  ## Error, cannot find db2admin.raul
select * from arfchong.raul  ## OK, this works
```
Listing 8.3 - An example of a private synonym

In *Listing 8.3* you see that because the synonym is private, it cannot be referenced when connected as another user without specifying the schema.

8.3 Tables

A table is a collection of related data logically arranged in columns and rows. *Listing 8.4* below provides an example of how to create a table using the **CREATE TABLE** statement.

```
CREATE TABLE artists
 (artno           SMALLINT    not null,
  name            VARCHAR(50) with default 'abc',
  classification  CHAR(1)     not null,
  bio             CLOB(100K)  logged,
  picture         BLOB(2M)    not logged compact
 )
 IN mytbls1
```

Listing 8.4 - An example of a CREATE TABLE statement

In the following sections, we describe the main parts of this **CREATE TABLE** statement

8.3.1 Data Types

Figure 8.1, obtained from the DB2 Information Center, lists the data types supported in DB2
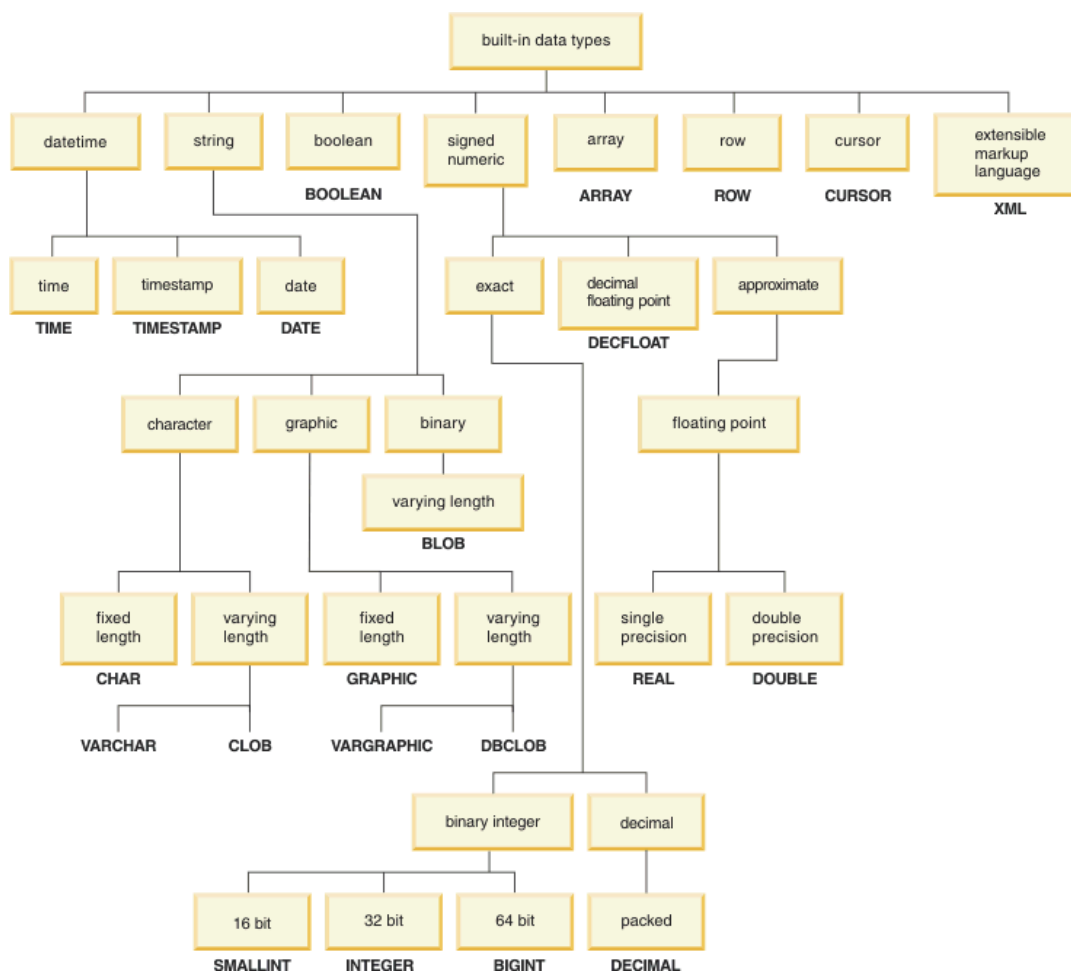
Figure 8.1 – DB2 built-in data types

The data types shown in *Figure 8.1* are described in detail in the DB2 documentation; most of them are common or very similar amongst relational database management systems so we will not describe them here. On the other hand, some data types like large objects (LOBs) may not be that intuitive for new users.

Large object data types are used to store large character strings, large binary strings or files as shown in *Figure 8.2*
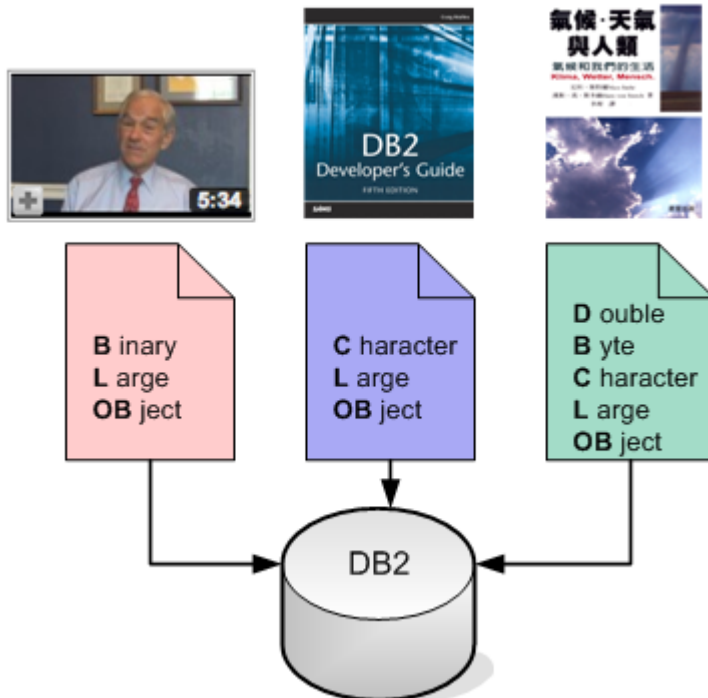
Figure 8.2 – LOBs data types

These large object binaries are usually abbreviated for clarity: a binary large object is a BLOB, a character large object is a CLOB, and a double byte character large object is also known as a DBCLOB.

Figure 8.1 also lists the new data types added with DB2 9.7:

- BOOLEAN

- ARRAY

- ROW

- CURSOR

These data types are part of the Oracle database server data types that are now supported in DB2. Oracle database server data types are discussed in a bit more detail later in this chapter.

8.3.1.1 User-defined types

DB2 allows you to define your own data types using user-defined types (UDTs). UDTs can be classified as:

- Distinct type

- Structured type

- Reference type

- Array type

- Row type

- Cursor type

Reference, array, row and cursor types are new with DB2 9.7 and are used with SQL PL routines. User-defined distinct data types are based on the built-in data types. These UDTs are useful when:

- There is a need to establish context for values

- There is a need to have DB2 enforce data typing using strong typing

The SQL statements in *Listing 8.5* illustrate an example of how and when to use distinct UDTs:

```
CREATE DISTINCT TYPE POUND AS INTEGER WITH COMPARISONS
CREATE DISTINCT TYPE KILOGRAM AS INTEGER WITH COMPARISONS
CREATE TABLE person
   (f_name    VARCHAR(30),
    weight_p  POUND NOT NULL,
    weight_k  KILOGRAM NOT NULL )
```

Listing 8.5 - An example of distinct data types

In this example, two distinct UDTs are created: POUND and KILOGRAM. Both are built based on the built-in data type INTEGER. The WITH COMPARISONS clauses defined as part of the syntax indicate that casting functions with the same name as the data types will also be created.

The table `person` uses the two new UDTs in columns `weight_p` and `weight_k`, respectively. If we now issue the following statement:

```
SELECT F_NAME FROM PERSON
  WHERE weight_p > weight_k
```

You will receive an error message because two columns with different data types are being compared. Even though weight_p and weight_k use the POUND and KILOGRAM data types respectively, both of which were created based on the INTEGER data type, by creating UDTs, you make this type of comparison impossible. This is exactly what you want, because in real life, a comparison between pounds and kilograms would not make sense.

In the next example, you would like to compare the column `weight_p` with an integer; however, these two data types are different, and therefore you would receive an error unless you use a casting function.

As you can see from the statement below, we use the casting function POUND() so that this comparison is possible. As indicated earlier, the POUND() casting function was created with the UDT when using the WITH COMPARISONS clause in the CREATE DISTINCT TYPE statement.

```
SELECT F_NAME FROM PERSON
  WHERE weight_p > POUND(30)
```

8.3.1.2 Oracle database server data types

The following data types used in Oracle database server are now supported with DB2 data server: NUMBER, VARCHAR2, TIMESTAMP(n), "DATE", BOOLEAN, INDEX BY, VARRAY, Row Type, Ref Cursor. For them to work you first need to enable the DB2_COMPATIBILITY_VECTOR registry variable as follows:

```
db2set DB2_COMPATIBILITY_VECTOR=FF
db2stop
db2start
```

Once this registry variable is enabled, new databases will be able to support these data types. Some of these types can only be used in the context where SQL PL is used.

Note:

If you are using an edition of DB2 that supports the SQL Compatibility feature (described in *Chapter 2*), you can also use these data types where PL/SQL is used. If that's the case, the value to set for the registry variable DB2_COMPATIBILITY_VECTOR should be 'FFF' instead of 'FF' as shown in the example above.
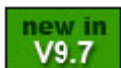
8.3.1.3 Implicit casting or weak typing

Many dynamic languages like Ruby on Rails or PHP allow for implicit casting. This was a problem in DB2 because of its strong typing requirements. With DB2 9.7, rules have been relaxed so that implicit casting or weak typing is allowed. What this means is that, for example, you can now assign or compare strings to numeric types as shown below:

```
create table t1 (col1 int)
select * from t1 where col1 = '42'
```

In the example, the string '42' can now be compared to the integer column *col1*.

In addition, with DB2 9.7 you are now allowed to specify untyped parameter markers and untyped NULLs in more situations. Previously, you had to explicitly cast them to a given data type. For example, the following statement now works:

```
select ?, NULL, myUDF(?, NULL) from t1
```

8.3.1.4 Null Values

A null value represents an unknown state. The CREATE TABLE statement can define a column using the NOT NULL clause to ensure that the column contains a known data

value. You can also specify a default value for the column if NOT NULL is declared. The next statement provides examples of this behaviour:

```
CREATE TABLE staff (
    ID       SMALLINT NOT NULL,
    NAME     VARCHAR(9),
    DEPT     SMALLINT NOT NULL with default 10,
    JOB      CHAR(5),
    YEARS    SMALLINT,
    SALARY   DECIMAL(7,2),
    COMM     DECIMAL(7,2) with default 15
    )
```

In this example, the *ID* and *DEPT* columns are defined as NOT NULL. The *DEPT* column also includes a default value of 10 in case no value is provided.

8.3.2 Identity Columns

An identity column is a numeric column which automatically generates a unique numeric value for each inserted row. There can only be one identity column per table.

There are two ways to generate values for an identity column, depending on how it was defined:

- **Generated always:** values are always generated by DB2. Applications are not allowed to provide an explicit value.

- **Generated by default:** values can be explicitly provided by an application or, if no value is given, then DB2 generates one. DB2 cannot guarantee uniqueness. This option is intended for data propagation, and for the unloading and reloading of a table

Let's take a look at the following example:

```
CREATE TABLE subscriber(
    subscriberID INTEGER GENERATED ALWAYS AS
        IDENTITY (START WITH 100 INCREMENT BY 100),
    firstname VARCHAR(50),
    lastname  VARCHAR(50) )
```

In the example, the column *subscriberID* is an INTEGER defined as an identity column that is always generated. The value generated will start from 100, and it will be incremented by 100.

8.3.3 Sequence objects

Though sequence objects are independent of tables, they are mentioned in this section because they work in a similar way as identity columns. The difference is that sequence

objects generate unique numbers across the database, while identity columns generate unique numbers within a table. The following statements provide an example:

```
CREATE TABLE t1 (salary int)

CREATE SEQUENCE myseq
   START WITH 10
   INCREMENT BY 1
   NO CYCLE

INSERT INTO t1 VALUES (nextval for myseq)

INSERT INTO t1 VALUES (nextval for myseq)

INSERT INTO t1 VALUES (nextval for myseq)

SELECT * FROM t1

SALARY
-----------
         10
         11
         12
  3 record(s) selected.

SELECT prevval for myseq FROM sysibm.sysdummy1

1
-----------
         12
  1 record(s) selected
```

Listing 8.6 - An example of sequences

PREVVAL provides the current value of the sequence, while NEXTVAL provides the next value. The above example also uses SYSIBM.SYSDUMMY1. This is a system catalog table that contains one column and one row. It can be used in situations where the query requires the output of a single value. System catalog tables are described in the next section.

8.3.4 System catalog tables

Each database has its own system catalog tables and views. These store *metadata* about the database objects. If these system tables become corrupted, your database would be rendered unusable. You can query these tables just like any normal database tables. Three schemas are used to identify the system catalog tables:

- SYSIBM: base tables, optimized for DB2 use
- SYSCAT: views based on the SYSIBM tables, optimized for ease of use

  - SYSSTAT: database statistics

The following are some examples of catalog views:

  - SYSCAT.TABLES

  - SYSCAT.INDEXES

  - SYSCAT.COLUMNS

  - SYSCAT.FUNCTIONS

  - SYSCAT.PROCEDURES

8.3.5 Declared global temporary tables (DGTTs)

Declared temporary tables are tables created in memory and are used by an application and then dropped automatically when the application terminates. These tables can only be accessed by the application that created them, and they do not exist in any DB2 catalog tables. Accessing these tables provides very efficient performance because there is no catalog contention, no locking of rows, no default logging (logging is optional), and no authority checking. There is also index support for these temporary tables, that is, any standard index can be created on a temporary table. You can also run RUNSTATS against these tables.

Declared temporary tables reside inside a user temporary table space, which must be defined prior to creating any declared temporary tables. The statements in *Listing 8.7* provide an example on how to create three declared temporary tables:

```
CREATE USER TEMPORARY TABLESPACE apptemps
     MANAGED BY SYSTEM USING ('apptemps');


DECLARE GLOBAL TEMPORARY TABLE temployees
     LIKE employee NOT LOGGED;


DECLARE GLOBAL TEMPORARY TABLE tempdept
      (deptid CHAR(6), deptname CHAR(20))
       ON COMMIT DELETE ROWS NOT LOGGED;


DECLARE GLOBAL TEMPORARY TABLE tempprojects
     AS ( fullselect ) DEFINITION ONLY
     ON COMMIT PRESERVE ROWS NOT LOGGED
     WITH REPLACE IN TABLESPACE apptemps;
```
Listing 8.7 - Working with DGTTs

When a declared temporary table is created, its schema is SESSION; this schema must be specified when referencing the DGTT. The user ID used to create a temporary table will have all privileges on the table. Each application that creates a temporary table will have its own independent copy as shown in *Figure 8.3*.

Figure 8.3 – Scope of declared global temporary tables

Listing 8.8 illustrates the scope limitations of declared global temporary tables. It assumes a user temporary table spaces has already been created.

From DB2 Command Window #1:

```
db2 connect to sample
db2 declare global temporary table mydgtt (col1 int, col2 varchar(10)) on
commit preserve rows
db2  insert  into  session.mydgtt  values  (1,'hello1'),(2,'hello2'),
(3,'hello3')
db2 select * from session.mydgtt


COL1        COL2
----------- ----------
          1 hello1
          2 hello2
          3 hello3
3 record(s) selected.


From DB2 Command Window #2:
db2 connect to sample
db2 select * from session.mydgtt
SQL0204N  "SESSION.MYDGTT" is an undefined name.   SQLSTATE=42704
```

Listing 8.8 - Working with DGTTs - scope

As you can see from *Listing 8.8* when you try to use ***SESSION.MYDGTT*** in the second session (DB2 Command Window #2), you receive an error because the DGTT was not defined in that session. Note that the DGTT definition uses the ON COMMIT PRESERVE

ROWS clause because working in the DB2 Command Window will commit by default on each statement entered.
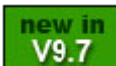
8.3.6 Create Global Temporary Tables (CGTTs)

While DGTTs allow you to create a temporary table, the table definition cannot be shared across different connections or sessions. Every time a session is established, the DECLARE GLOBAL TEMPORARY TABLE statement has to be executed. With Create Global Temporary Tables (CGTTs) on the other hand, the temporary table definition only needs to be created once as it is permanently stored in the DB2 catalog. This means that other connections can simply use the table without having to create it again. Though the table structure can be used immediately, the data for each connection is independent from each other, and will disappear after the connection closes. For an example, let's take a look at *Listing 8.9.* It assumes a user temporary table space has already been created.

```
From DB2 Command Window #1:
db2 connect to sample
db2 create global temporary table mycgtt (col1 int, col2 varchar(10)) on
commit preserve rows
db2 insert into mycgtt values (1,'hello1'),(2,'hello2'), (3,'hello3')
db2 select * from mycgtt


COL1        COL2
----------- ----------
          1 hello1
          2 hello2
          3 hello3
3 record(s) selected.


From DB2 Command Window #2:
db2 connect to sample
db2 select * from mycgtt


COL1        COL2
----------- ----------
  0 record(s) selected.
```

Listing 8.9 - Working with CGTTs - scope

In *Listing 8.9*, we saw that within DB2 Command Window #2 (another session or connection), there is no need to create the CGTT again, you can simply reference it; however, no rows are returned because the data was particular to the first session.

8.4 Views

A view is a representation of the data in tables. The data for the view is not stored separately, but is obtained when the view is invoked. Nested views, that is, a view created

based on other views, are supported. All view information is kept in the following DB2 catalog views: SYSCAT.VIEWS, SYSCAT.VIEWDEP, and SYSCAT.TABLES. *Listing 8.10* provides an example of how to create and use a view.

```
CONNECT TO MYDB1;

CREATE VIEW MYVIEW1
  AS SELECT ARTNO, NAME, CLASSIFICATION
  FROM ARTISTS;

SELECT * FROM MYVIEW1;

Output:

ARTNO      NAME             CLASSIFICATION
------     -------------    --------------
  10       HUMAN            A
  20       MY PLANT         C
  30       THE STORE        E
  ...
```
Listing 8.10 - Working with views

8.5 Indexes

An index is an ordered set of keys each of which points to a row in a table. An index allows for uniqueness, and it also improves performance. Some of the characteristics that you can define on indexes:

- The index order can be ascending or descending

- The index keys can be unique or non-unique

- Several columns can be used for the index (this is called a compound index)

- If the index and the physical data are clustered in similar index sequence, they are a cluster index

For example:

```
CREATE UNIQUE INDEX artno_ix ON artists (artno)
```

8.5.1 Design Advisor

The Design Advisor is an excellent tool to advise you on the optimal design of your database for a given SQL workload. The design advisor can help you with the design of your indexes, Materialized Query Tables (MQTs), Multi-dimension clustering (MDC), and the database partitioning feature. The Design Advisor is invoked from the Control Center; right-click on a database and select *Design Advisor* as shown in *Figure 8.4*.
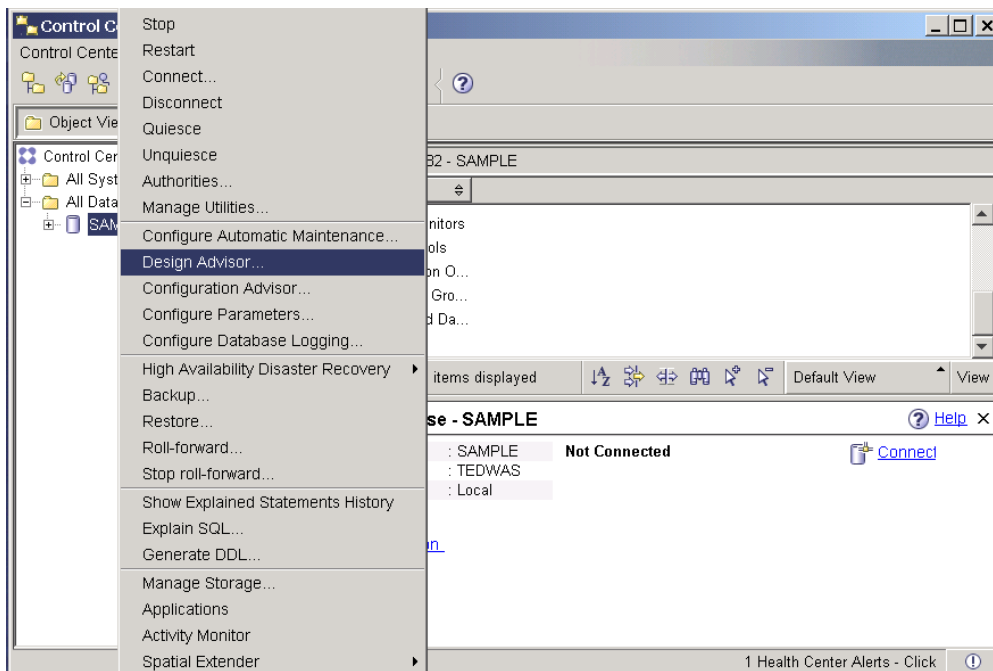
Figure 8.4 – Invoking the Design Advisor from the Control Center

Figure 8.5 shows the Design Advisor. Follow the steps in this wizard to obtain the design recommendations from DB2.
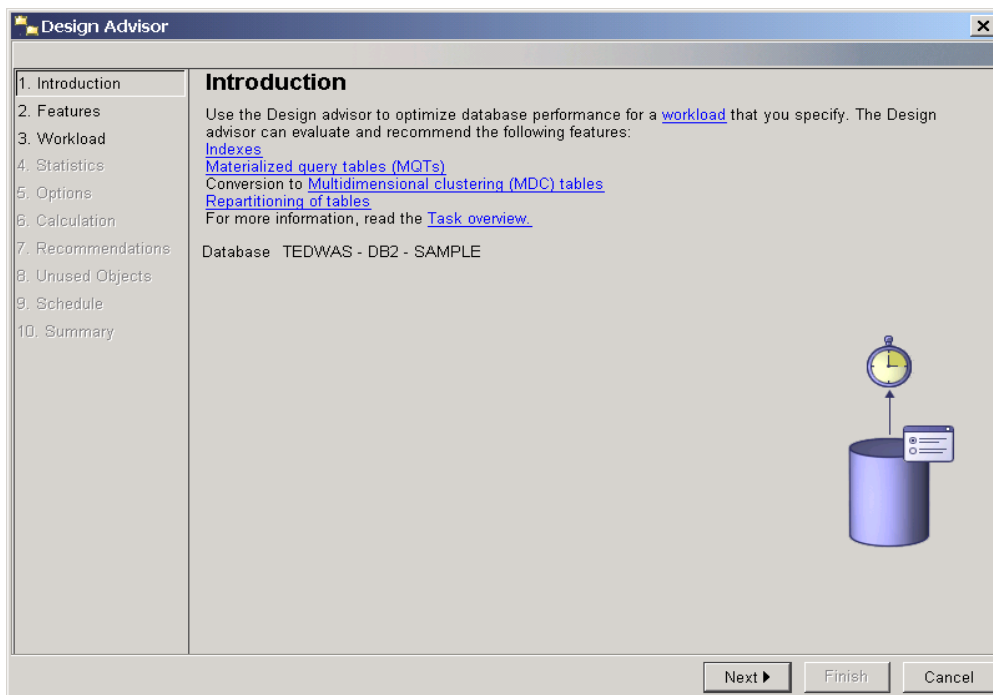
Figure 8.5 –The Design Advisor

8.6 Referential integrity

Referential integrity allows your database to manage relationships between tables. You can establish parent-child type of relationships between tables as shown in *Figure 8.6*. In the figure, there are two tables, DEPARTMENT and EMPLOYEE, related by the department number. The WORKDEPT column in the EMPLOYEE table can only contain department numbers that already exist in the DEPARTMENT table. This is because in this example, the DEPARTMENT table is the parent table, and the EMPLOYEE table is the child, or dependent, table. The figure also shows the necessary CREATE TABLE statement for the EMPLOYEE table needed to establish the relationship.
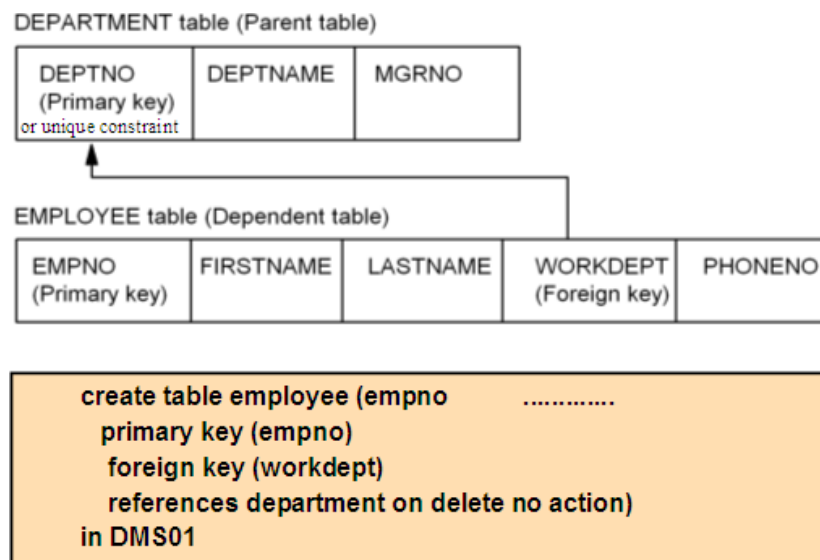
Figure 8.6 –An example of referential integrity between tables

In referential integrity, the following concepts listed in *Table 8.1* are often used.

Concept	Description
Parent table	A controlling data table in which the parent key exists
Dependent table	A table dependent on the data in the parent table. It also contains a foreign key. For a row to exist in a dependent table, a matching row must already exist within a parent table.
Primary Key	Defines the parent key of the parent table. It cannot contain NULL values and values must be unique. A primary key consists of one or more columns within a table.
Foreign Key	References the primary key of a parent table

Table 8.1 - Referential Integrity key concepts

Data in tables can be related to data in one or more tables with referential integrity. Constraints can also be imposed on data values so that they conform to a certain property or business rule. For example, if a table column stores the sex of a person, the constraint can enforce that the only values allowed are "M" for male, and "F" for female.

8.7 Schema Evolution

When business needs change, the supporting information technology (IT) infrastructure and systems must also change. What this means in the database world is that new tables have to be created, existing tables have to be dropped or modified, trigger logic has to change, and so on. Though these changes seem simple to make, in reality, they can be difficult and complex. Long maintenance windows are required, and intricate, risky procedures must be performed. One of the reasons these changes were difficult to make previously was that DB2 required all objects to be consistent at all times, and therefore, actions affecting objects which depend on the object being changed were either not allowed, or would cause the dependent objects to be dropped. *Figure 8.7* provides a sample scenario.
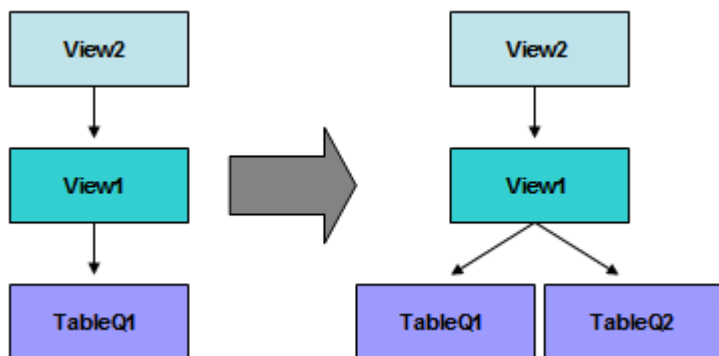
Figure 8.7 –Schema evolution sample scenario

As an example, in *Figure 8.7,* the definition of `View1` needs to change so that it's not just based on `TableQ1` with a company's first quarter financial information, but on `TableQ1` and `TableQ2` with first and second quarter information, respectively. Normally you would have to drop `View1` and recreate it with the new definition; however, `View2` is dependant on `View1`. Prior to DB2 9.7, DB2 would not allow you to drop `View1` because of the dependant `View2.` You would have to drop `View2` first, so that you could drop `View1` and then reconstruct both views. With DB2 9.7, rules have been relaxed. Now changes that affect dependant objects are allowed. The dependant object (`View2` in the example), must be revalidated before it is used, but this is done automatically for you. This is known as *automatic revalidation*. The db cfg parameter AUTO_REVAL is used to turn on or off automatic revalidation and to determine when revalidation will occur. For example, if you set this parameter to DEFERRED_FORCE, revalidation will be deferred until the invalid object or dependant objects are accessed, but CREATE statements will be allowed (with a warning) on dependant object that don't yet exist.

Other changes affecting the dependency model include the implementation of features like CREATE OR REPLACE syntax for views, functions, procedures, triggers, aliases, and so on. For example:

```
create or replace procedure p1 begin ... end
```

With this syntax, if an object (for example, procedure P1) didn't exist, it would be created. If it existed before, it would be replaced. This second behavior is what is important for object dependency. When P1 is replaced, objects dependant on P1 are automatically revalidated. A similar situation happens for new features like RENAME COLUMN, and for data type changes with ALTER COLUMN, which has also been enhanced to support more data type changes.

A related concept known as **soft invalidation** allows users to drop an object even as other running transactions are using it. Any new transactions will be denied access to the dropped object.

8.8 Summary

In this chapter we took a look at the database objects in DB2: what they are, how they are created and how to use them. We introduced database schemas and compared them with the new public synonyms, as well as how public synonyms compare to private synonyms.

Next, we discussed tables and their elements in depth: data types (both built-in and user-defined), identity columns, sequence objects, and global temporary tables. This was followed by looking at views, indexes and using the Design Advisor GUI to improve the accessibility and retrievabliity of the data inside a table.

Finally, we examined referential integrity to define relationships between tables, and the new concept of schema evolution which allows for data objects to be changed without unnecessary complication.

8.9 Exercises

So far, you have been using the existing tables in the **SAMPLE** database to illustrate concepts. Eventually, you will need to create your own tables in a database. In this exercise, you will use the *Create Table Wizard* to create two new tables in the **SAMPLE** database.

Procedure

1. Launch the *Create Table Wizard* as previously shown in the chapter. (*Control Center -> All Databases -> SAMPLE -> (right-click) Tables object -> Create …*)

2. Define the table name, column definitions, and any constraints. The table will be used to store information about the office supplies used by a project in the **SAMPLE** database. Each time supplies are purchased, a row will be added to this table. The table will have six columns:

- product_id: unique identifier of the item being purchased

- description: description of the item

- quantity: the quantity purchased

- cost: the cost of the item

- image: a picture of the item (if available)

- project_num: the project this product has been purchased for

3. In the first page of the wizard, for the schema name, enter the user ID you are currently logged on as, and use the following table name: **SUPPLIES**. You can also optionally enter a comment. Click the *Next* button to continue to the next page of the wizard.

4. From this page, you can add columns to the table. Click the *ADD* button to add columns
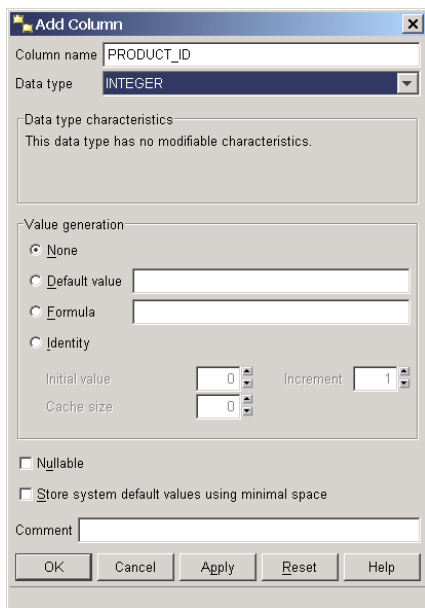
5. Enter the column name **product_id** and select the data type: **INTEGER**. Uncheck *Nullable*, and click the *Apply* button to define the column.

6. Repeat this step for the remaining columns of the table using the options shown in the table below. Once all columns have been added (Applied), click the *OK* button and the list of the columns you just created should be summarized. Click the *Next* button to continue to the next page of the wizard.

Column Name	Attributes
product_id (completed)	INTEGER, NOT NULL
description	VARCHAR, length 40, NOT NULL

quantity	INTEGER, NOT NULL
cost	DECIMAL, Precision 7, Scale 2, NOT NULL
image	BLOB, 1MB, NULLABLE, NOT LOGGED
project_num	CHAR, length 6, NOT NULL

Note: The NOT LOGGED option can be specified when declaring LOB columns. It is mandatory for columns greater than 1GB in size. It is also generally recommended for LOBs larger than 10MB, as changes to large columns can quickly fill the log file. Even if NOT LOGGED is used, changes to LOB files during a transaction can still be successfully rolled back. Also notice that the image column is the only one defined as a NULLABLE column. Why do you think that the column was defined like this?

7. At this point, all the mandatory information for creating a table has been provided. By skipping the other panels in the wizard, you are choosing the default values for those options. You can always add keys and constraints after a table has been created.

8. Add a constraint to the table to restrict values on the *quantity* column. On the *Constraint* page of the wizard, click the *ADD* button. In the *Check Name* field, enter: **valid_quantities**. In the *Check Condition* field, enter: **quantity > 0**

   Click the *OK* button. You should see a summary of the constraint you just added in the *Constraint* page of the wizard. Click the *Next* button to continue to the next page of the wizard.

9. You can continue going through the wizard, changing the other parameters of the table. Alternatively, you can skip to the *Summary* page, or simply click the *Finish* button to create the table.

10. From Control Center, click on the *Tables* folder under the **SAMPLE** database in the Object Tree pane. The table you just created should now appear in the list. It might be necessary to refresh the Control Center view in order to see the changes.

11. Let's now test implicit casting using the table **STAFF** in the **SAMPLE** database. Try the following:

    ```
    C:\>db2 describe table staff
    ```
    Note that the ID column is defined as a SMALLINT

    ```
    C:\>db2 select * from staff where id = '350'
    ```
    --> Note '350' is a string
    ```
    C:\>db2 select * from staff where id =  350
    ```
    --> Note 350 is a number

    In both cases, the output should be:

    ```
    ID     NAME      DEPT   JOB   YEARS  SALARY    COMM
    ```

```
------ --------- ------ ----- ------ --------- ---------
   350 Gafney       84 Clerk     5  43030.50    188.00
```

In the first SELECT statement using '350' as a string, DB2 is performing the implicit casting to a number (SMALLINT).

9

Chapter 9 – Data Movement Utilities

The tools or commands described in this section are used to move data within the same database or across databases in the same or different platforms. *Figure 9.1* provides an overview of the data movement utilities.
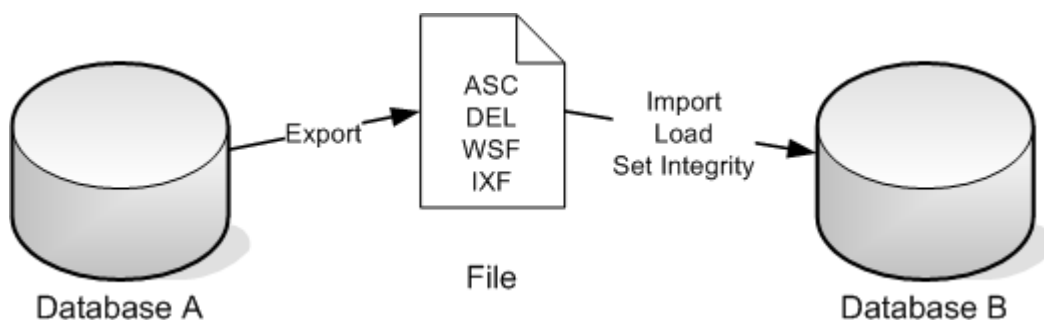
Figure 9.1 – Data movement utilities

In *Figure 9.1* there are two databases, database A, and B. Using the EXPORT utility, one can export the data from a table into a file. The file can have any of these formats:

- ASC = ASCII
- DEL = Delimited ASCII
- WSF = Worksheet format
- IXF = Integrated Exchange Format

ASC and DEL files are text files that can be opened and reviewed in any text editor. WSF is a format that can be used to move data to spreadsheets such as Excel or Lotus® 1-2-3. IXF is a format that not only includes the data but also the Data Definition Language (DDL) of the table in question. The IXF format is convenient because when the table needs to be reconstructed, it can be done directly from a file with an IXF formatted export; this is not possible if you use the other formats.

Once the data has been exported to a file, the IMPORT utility can be used to import the data from the file into another table. The table must exist beforehand for the ASC, DEL and WSF format, but it does not need to exist for the IXF format. Another method to load the data into a table is to use the LOAD utility. The LOAD utility is faster as it goes directly to

the database pages without interacting with the DB2 engine; however, this method does not make a check for constraints, and triggers are not fired. To guarantee consistency of the data loaded using LOAD, the SET INTEGRITY command is often used afterwards.

The next sections describe the EXPORT, IMPORT, and LOAD utilities in more detail.

> **Note**:
>
> For more information about working with data movement utilities, watch this video: http://www.channeldb2.com/video/video/show?id=807741:Video:4262

9.1 EXPORT utility

The EXPORT utility is used to extract data from a table into a file as discussed earlier. Behind the scenes, an SQL SELECT operation is what is really being performed. The following example exports to the file `employee.ixf` of IXF format 10 rows from the table *employee*.

```
EXPORT TO employee.ixf OF IXF
   SELECT * FROM employee
   FETCH FIRST 10 ROWS ONLY
```

We encourage you to try the above example. The employee table is part of the **SAMPLE** database, so you first need to connect to this database created in a previous chapter.

If you prefer to work with GUI tools, the EXPORT utility can also be invoked from the Control Center as shown in *Figure 9.2*.
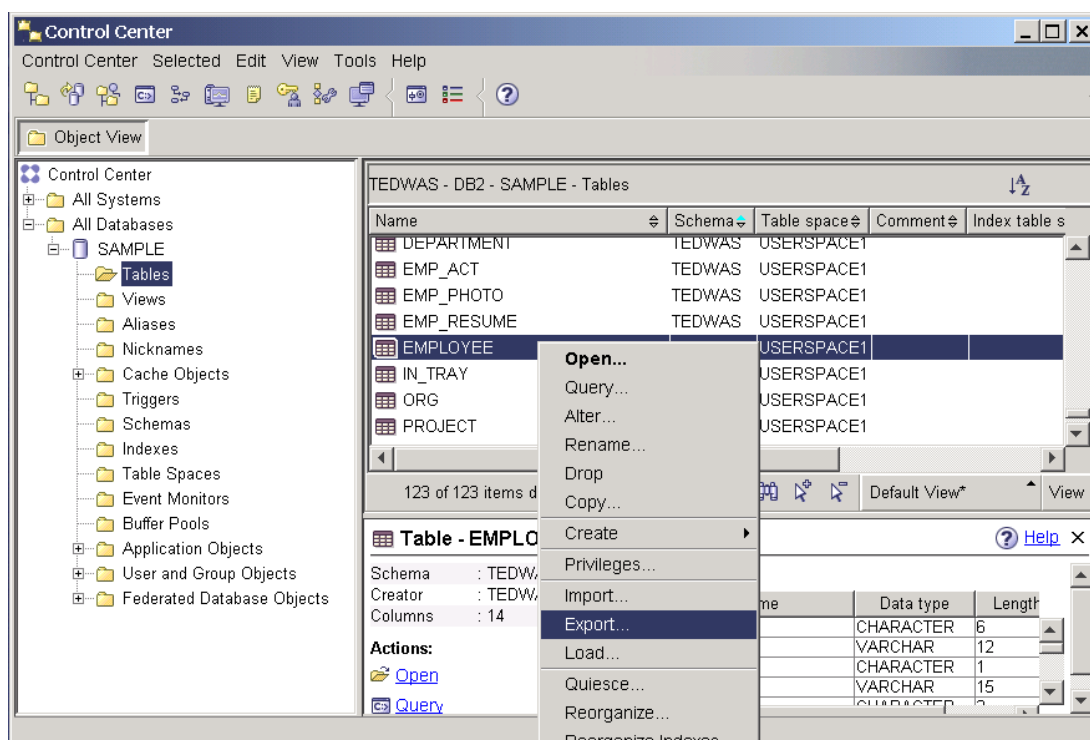
Figure 9.2 – Launching the EXPORT table dialog

As shown in the figure, you first select the employee table by clicking it once, and then right click on the table to obtain a pop-up menu from where you can choose the Export option. After choosing this option, a wizard will come up. Simply follow the steps the wizard provides to complete the operation.

9.2 IMPORT utility

The IMPORT utility is used to load data from a file into a table as discussed earlier. Behind the scenes, an SQL INSERT operation is really being executed. As an INSERT operation is being executed, any triggers are activated, all constraints are enforced immediately, and the database bufferpool is used. The following example loads all the data from the IXF formatted file `employee.ixf` into the table *employee_copy.* We encourage you to try the example, but you need to have run the EXPORT utility in the previous section.

```
IMPORT FROM employee.ixf OF IXF
   REPLACE_CREATE
   INTO employee_copy
```

The REPLACE_CREATE option is one of many options available with the IMPORT utility. This option replaces the contents of the **employee_copy** table if it previously existed before the IMPORT utility was executed, or it will create the table and load the data if the table didn't already exist.

If you prefer to work from the Control Center, you can launch the IMPORT utility by selecting any table, right-clicking on it, and choosing the Import option as shown in *Figure 9.3*
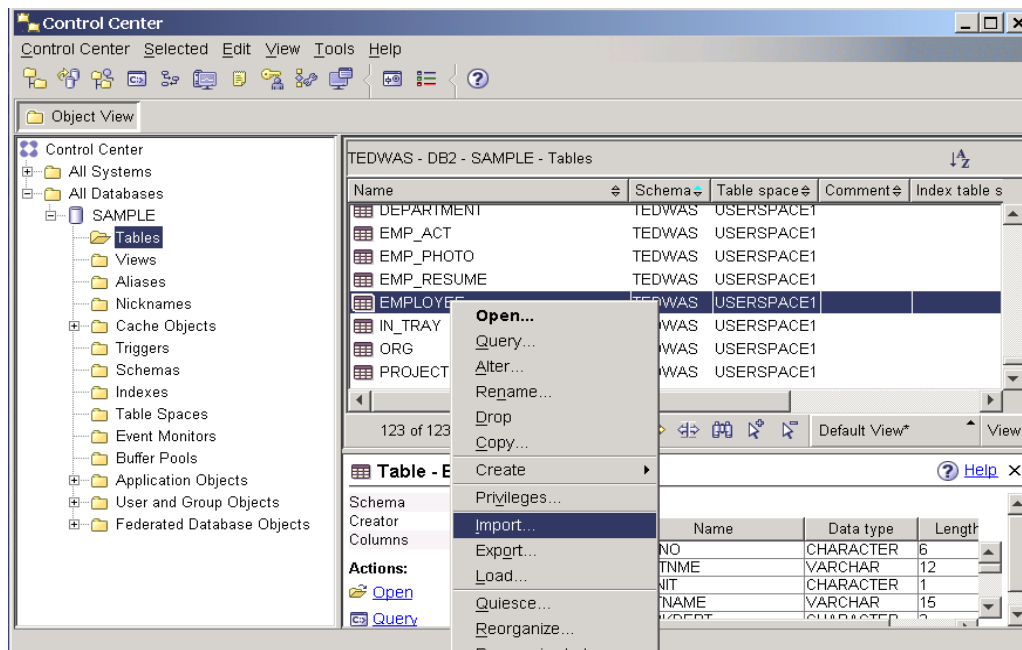
Figure 9.3 – Launching the IMPORT dialog

9.3 LOAD utility

The LOAD utility is a faster way to load data from a file into a table. As discussed before, the LOAD utility does not go through the DB2 engine, therefore triggers are not activated, the bufferpool is not used, and constraints can be enforced but only as a separate step. On the other hand, a LOAD operation is faster than IMPORT as it is a low level data loader directly accessing the data pages on disk. It works in three phases: LOAD, BUILD, and DELETE.

The following example loads all the data from the IXF formatted file employee.ixf into the table *employee_copy*. The REPLACE option is one of the many options available with LOAD. In this case it is used to REPLACE all of the contents of the *employee_copy* table.

```
LOAD FROM employee.ixf OF IXF
   REPLACE INTO employee_copy
```

After executing the above command, the table space where your table resides may have been placed in CHECK PENDING state. This means you need to run the SET INTEGRITY command to check the consistency of your data. The following example shows you how:

```
SET INTEGRITY FOR employee_copy
```

```
ALL IMMEDIATE UNCHECKED
```

If you prefer to work from the Control Center, you can launch the LOAD and the SET INTEGRITY utilities as shown in *Figure 9.4* and *9.5* respectively.
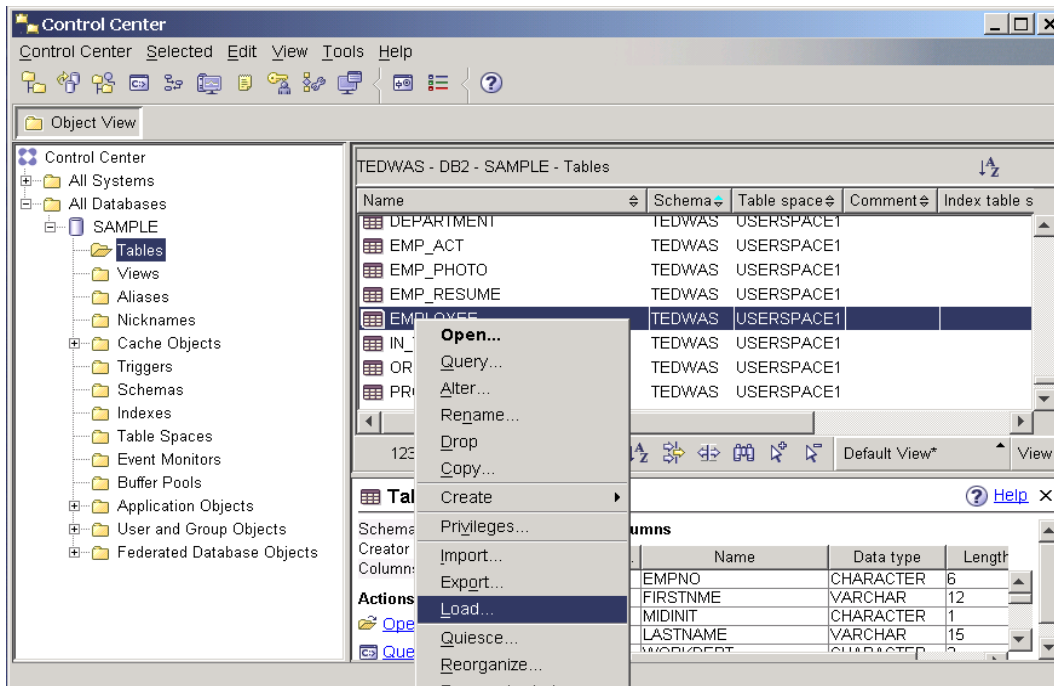
Figure 9.4 – Launching the LOAD utility
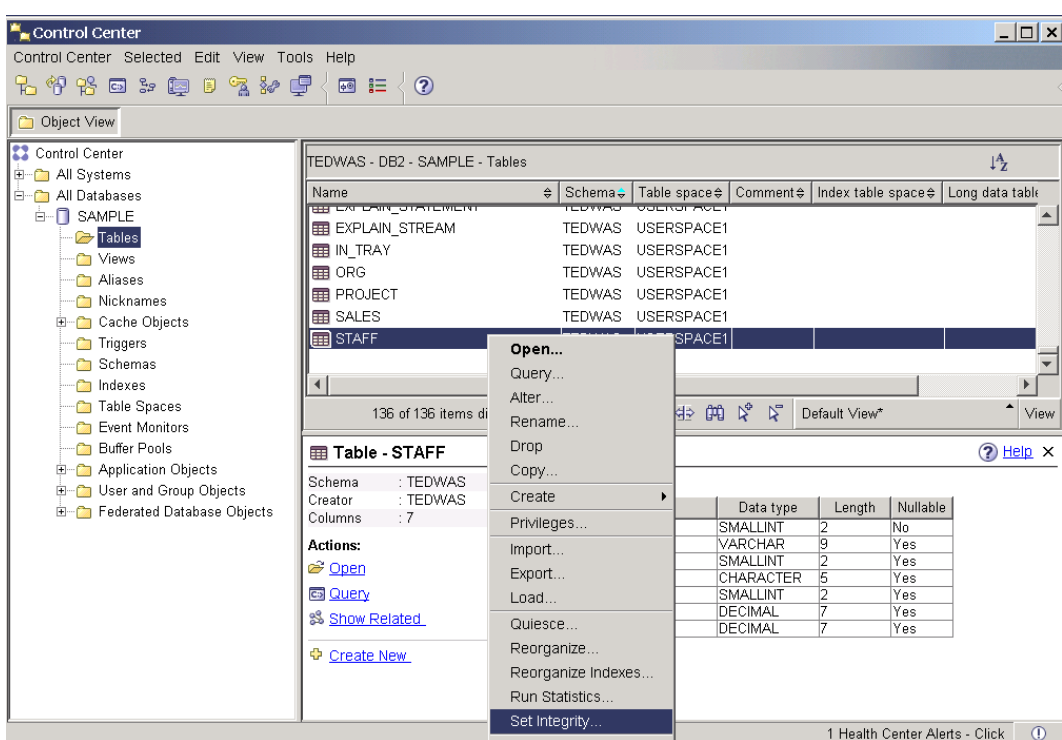
Figure 9.5 – Launching the SET INTEGRITY wizard

9.4 The db2move utility

The EXPORT, IMPORT, and LOAD utilities work on one table at a time. Though you could write a script to generate the above commands for each table in a database, another utility called **db2move** can do this for you. The **db2move** utility can only work with IXF files, and the file names will automatically be generated by **db2move**. The examples below show how to run **db2move** with the export, and import options respectively using the **SAMPLE** database.

```
db2move sample export
db2move sample import
```

The Control Center does not have an option for **db2move**.

9.5 The db2look utility

While EXPORT, IMPORT, LOAD and **db2move** utilities allow you to move data from one table to another, either within one database or across several databases, the **db2look** utility can be used to extract the DDL statements, database statistics and table space characteristics for a database and store them in a script file that can later be run on another system. For example, if you want to clone a database from a DB2 server running on Linux to a DB2 server running on Windows; you would first run the **db2look** utility on the DB2

Linux server to obtain the structure of the database and store this structure on a script file. You would then copy this script file to the DB2 Windows server, and execute the script to start building the cloned database. At this point, the structure of the database has been cloned. The next step would be to run the **db2move** utility with the export option in the DB2 Linux server, and then copy all the generated files to the DB2 Windows server, then execute the **db2move** with either of the import or load options. Once this is done, your database is fully cloned from one server to another on a different platform.

The above scenario may be needed when working with databases on different platforms such as Linux and Windows. If both servers are running on the same platform, you would likely use the backup and restore commands, which make this process easier and more straight-forward. The **backup** and **restore** commands are discussed in more detail in a later chapter of this book.

The following example extracts the table space and bufferpool layouts, along with the DDL statements from the **SAMPLE** database, and stores them into the file sample.ddl. We encourage you to run the command below and review the output text file sample.ddl.
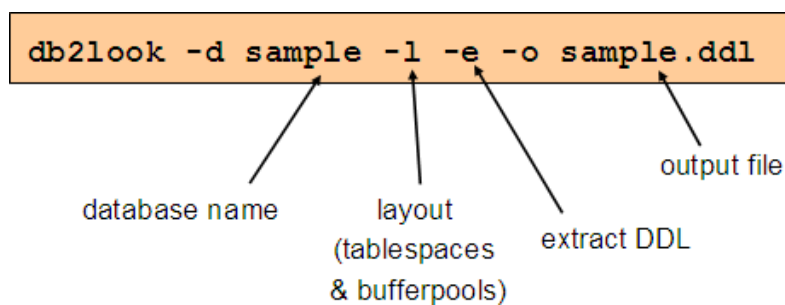
The **db2look** command has too many options to describe in this book; however you can use the −h flag to obtain a brief description of the available options:

```
db2look -h
```

The **db2look** utility can also be invoked from the Control Center as shown in *Figure 9.6*
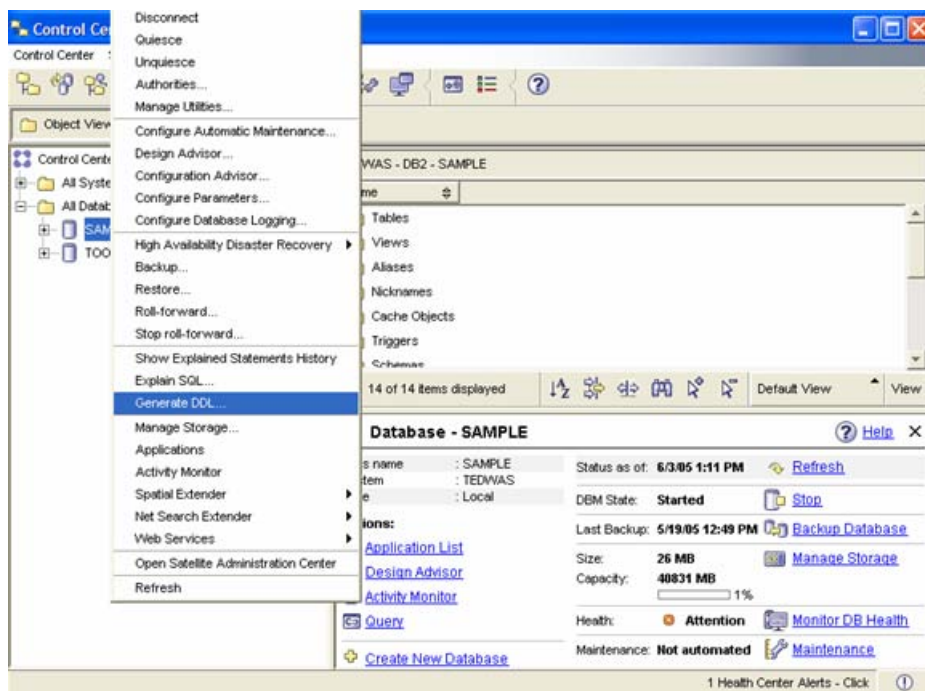
Figure 9.6 - Extracting DDL from the Control Center

In *Figure 9.6*, select the database from which you want to obtain the DDL, right click on it, and choose *Generate DDL*. The Generate DDL window appears, showing several extraction options, as shown in *Figure 9.7*.
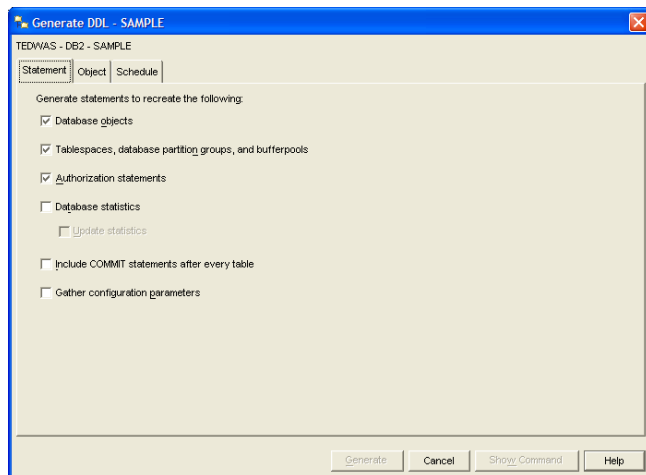
Figure 9.7 - Extracting DDL from the Control Center

9.6 Summary

In this chapter, we discussed the various export and import functions of DB2. Beginning with a look at the various export formats (ASC, DEL, WSF, and IXF), we moved onto an in-depth examination of the EXPORT utility. The import utilities IMPORT and LOAD were then discussed, along with the need for the SET INTEGRITY statement when using LOAD.

The db2move command provides you with a means to simplify the exporting and importing transfer process. A more complex command, db2look, allows you to extract and store all the database elements needed to independently recreate the entire database if desired.

9.7 Exercises

When you clone a database, your goal should be to recreate the database in a manner that is as straightforward and repeatable as possible. This is usually done using SQL scripts, which can be immediately executed after DB2 has been installed. In this exercise, you will extract the object definitions from the **EXPRESS** database (created in an earlier exercise) using the Control Center.

Procedure

1. Open the Control Center.

2. Right-click on the **EXPRESS** database in the object tree and select the *Generate DDL* menu item. This launches the *Generate DDL* dialog window.

3. In the *Generate DDL* window, specify options for the generated DDL, as shown below. If you created additional objects in your environment, such as table spaces, buffer pools, etc., you would select them here. Since you have not created these types of objects, uncheck the box. Database statistics have not been included because the production environment will likely contain a different set of statistics than the development environment. Similarly, configuration parameters will likely be different as well. In your own environment, if everything is configured exactly the way it will be deployed, you may choose to include those additional options.
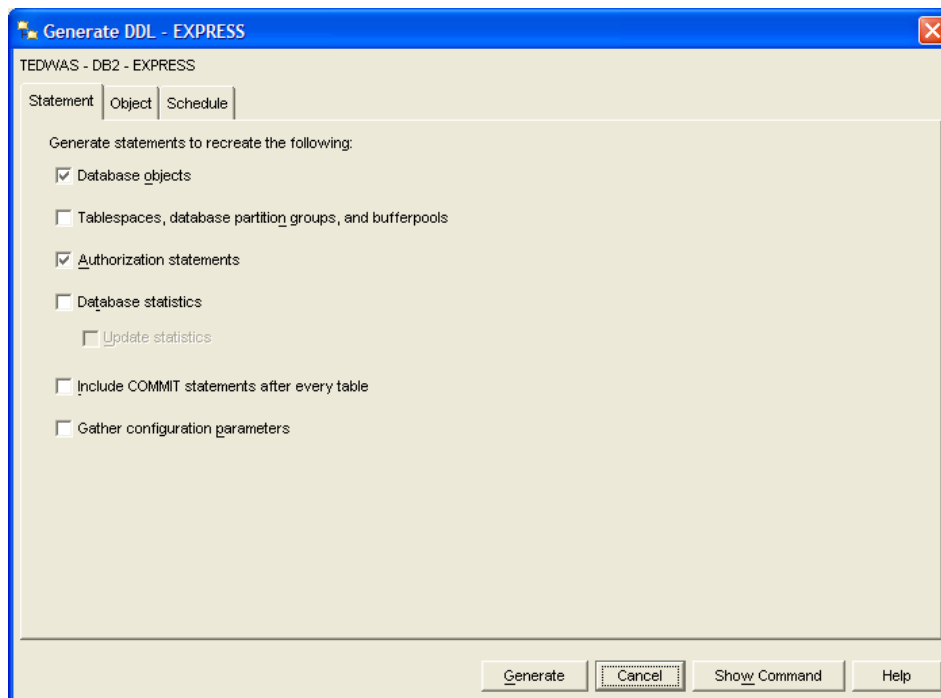
4. Move to the *Object* tab. You are able to specifically choose which objects you want to generate DDL. In this case, select the user and schema you have been using to create all your objects in and generate the DDL for all objects in that schema. Click the *Generate* button to start DDL generation.
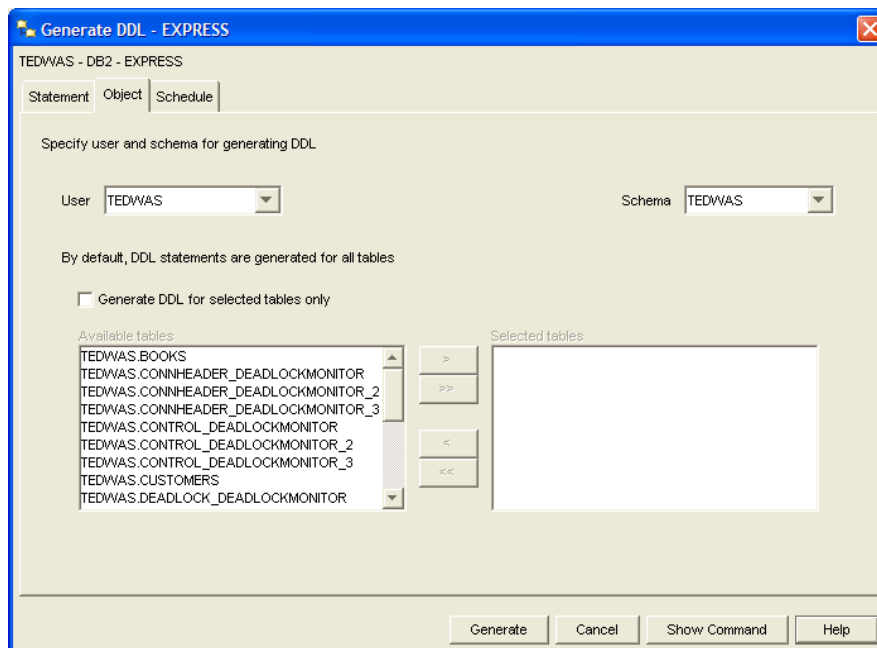
5. Review the resulting DDL. The result of the previous step is a single script with all the SQL statements for the chosen objects. You will now organize this script into logical groupings.

6. Create a directory called `C:\express` in the file system and save the generated DDL file in this new directory to a file called `schema.ddl`. (Click the *Save* button)
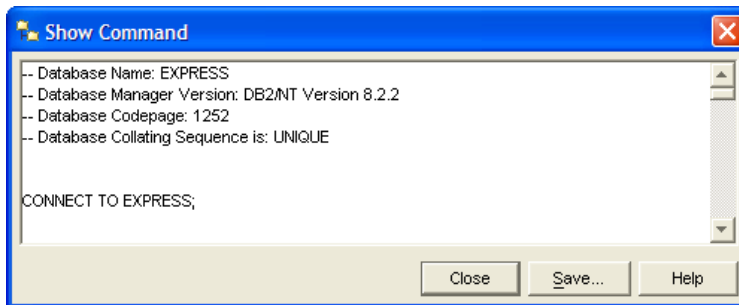
7. Open the newly saved file in Command Editor. (Hint: From Command Editor, choose *File -> Open*)

8. Although we only really wanted the DDL for tables, you will notice that DDL for other database objects is included as well. Move all the **CREATE TRIGGER** statements into a separate new file called `triggers.ddl`. Even though we only created one trigger, it is generally a best practice to separate objects by types.

9. For now, we also recommend removing all:

   ▪ CONNECT TO database statements

   ▪ DISCONNECT statements

   You should have two scripts at this point:

   `C:\express\schema.ddl` containing the DDL for tables, views, indexes, and constraints.

   `C:\express\triggers.ddl` containing the DDL for triggers

10. Cleanse the script for deployment:

    ▪ Remove unnecessary comments (e.g. -- CONNECT TO…)

    ▪ Separate the functions and procedures into their own files (useful when there are a lot of functions and procedures). You might also want to group them by function or application (e.g. `billing.ddl`, `math.ddl`, `stringfunc.ddl`, etc.)

11. You may have noticed that a special character is being used to delimit the end of the triggers, functions and procedures (@). This is necessary in order to delimit the end of the **CREATE <object>** statement as opposed to the end of a procedural statement within the object.

10

Chapter 10 – Database Security

This chapter discusses how security is handled in DB2. *Figure 10.1* provides a basic overview.
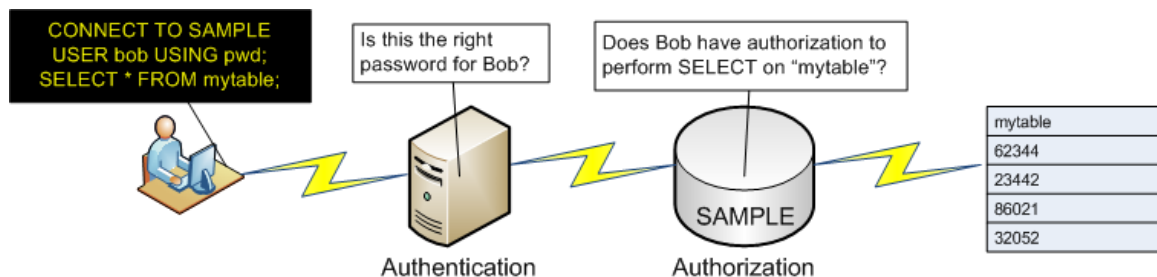
Figure 10.1 – DB2 security overview

As shown in *Figure 10.1*, DB2 security consists of two parts:

- **Authentication**

  It is the process by which the user identity is validated. Authentication is performed by a security facility outside of DB2 through a security plug-in. The default security plug-in relies on the operating system security, but you can also use plug-ins for Kerberos, LDAP, or you can build your own custom-built authentication plug-in. When using the default OS-based authentication plug-in, the userid and password are sent to the database server (e.g. as part of a connect statement). The database server then invokes the OS authentication to validate the userid and password.

- **Authorization**

  At this stage, DB2 checks if the authenticated user may perform the requested operation. The authorization information is stored in a DB2 catalog and a DBM configuration file.

For example, in *Figure 10.1*, user **bob** connects to the **SAMPLE** database with this statement:

```
CONNECT TO sample USER bob USING pwd
```

Both *bob* and *pwd* are passed to the operating system or external authentication facility to perform the authentication approval, verifying that a user named *bob* is already defined, and that the password provided matches that user. If this part is successful, the operating system will return security control to DB2. Next, when user *bob* executes a statement such as:

```
SELECT * FROM mytable
```

DB2 takes over security control to perform the authorization check and confirm that user *bob* has SELECT privilege on table *mytable*. If the authorization check fails, DB2 will return an error message, otherwise the statement will be executed against *mytable*.

Note:

For more information about working with DB2 security, watch this video:
http://www.channeldb2.com/video/video/show?id=807741:Video:4267

10.1 Authentication

Although the actual authentication is performed by the operating system through the default security plug-in (or another external security facility), DB2 does decide at which level this authentication occurs.

The database configuration parameter AUTHENTICATION, set at the DB2 server, has a range of possible values. For example, when the parameter is set to SERVER (the default), the authentication is performed by the operating system or external security facility on the server. However, if AUTHENTICATION is set to CLIENT, the authentication is performed by the operating system or external security facility at the client. This is shown in *Figure 10.2*.
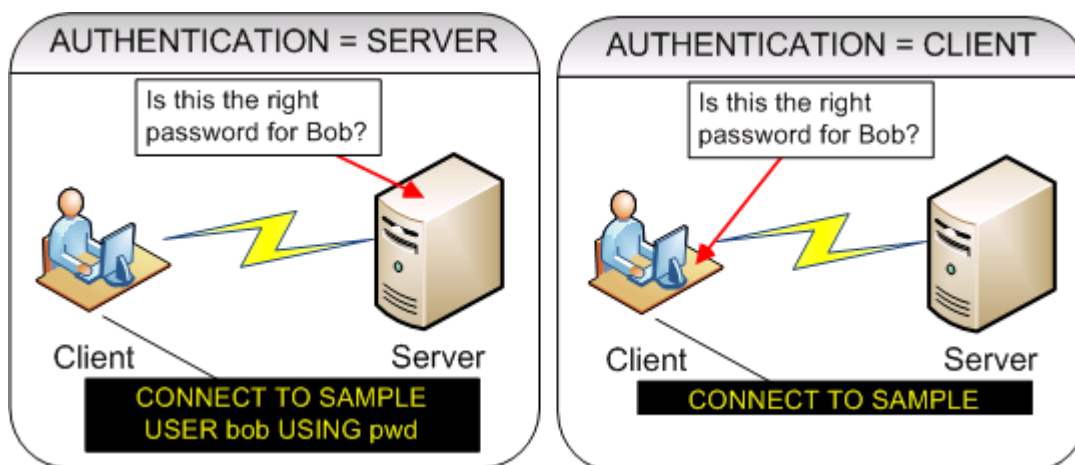
Figure 10.2 – Where authentication takes place

The AUTHENTICATION parameter can be set to any of the values listed in *Table 10.1*

Command	Description
SERVER (default)	Authentication takes place at the server
CLIENT	Authentication takes place on the client
SERVER_ENCRYPT	Like SERVER except user IDs and passwords are encrypted
KERBEROS	Authentication takes place using a Kerberos security mechanism
SQL_AUTHENTICATION_DATAENC	Server authentication plus connections must use data encryption
SQL_AUTHENTICATION_DATAENC_CMP	Like above, except data encryption only used when available
GSSPLUGIN	Authentication uses an external GSS API-based plug-in security mechanism

Table 10.1 – Valid AUTHENTICATION parameter values

10.2 Authorization

Authorization consists of the privileges, authorities, roles, and label-based access control (LBAC) credentials that are stored in DB2 system tables and are managed by DB2.

A privilege allows a user to execute a single type of operation against the database, such as CREATE, UPDATE, DELETE, INSERT, etc.

A role allows you to group together different privileges that you can grant to a user, group, or other roles.

An authority is a predefined role consisting of several privileges.

Label-based Access Control (LBAC) credentials include policies and labels supporting granular access to specific rows and columns by given users. LBAC is not included with DB2 Express-C, but you can read more about it in *Chapter 2*.

10.2.1 Privileges

Figure 10.3 shows some of the different privileges in DB2.
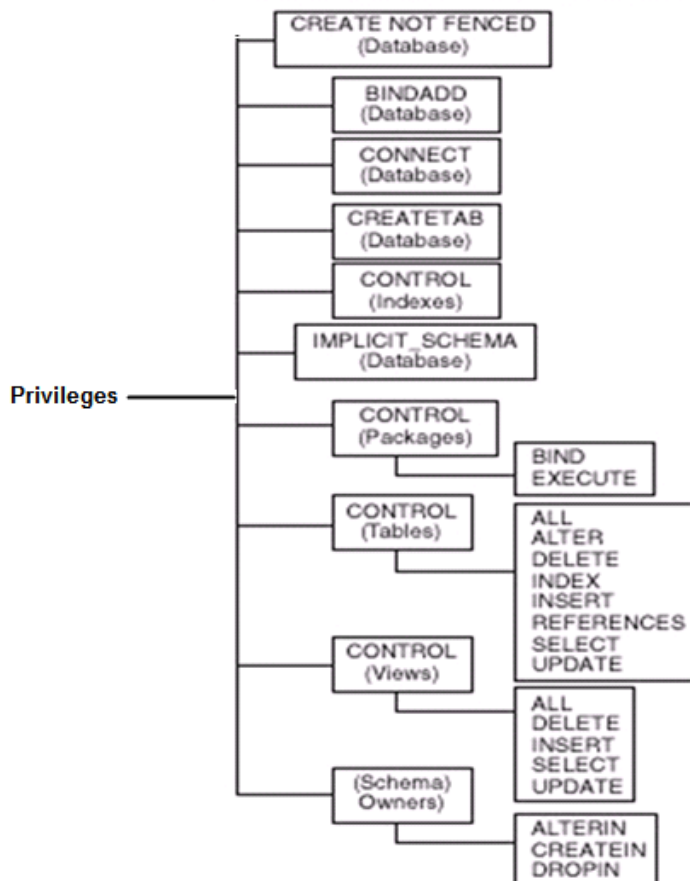
Figure 10.3 – Listing of some DB2 privileges

A user or group receiving CONTROL privileges implies that they can also grant the privilege to some other user or group. Refer to the DB2 Information Center for details about the other different privileges.

10.2.2 Authorities

Authorities are classified in two groups:

- Instance-level authorities: These authorities can operate at the instance level. For example, SYSADM.

- Database-level authorities: These authorities can only operate at the database level. For example, DBADM.

10.2.2.1 Instance-level authorities

Table 10.2 lists the instance-level authorities

Authority	Description
SYSADM	Manages the instance as a whole
SYSCTRL	Administers a database manager instance
SYSMAINT	Maintains databases within an instance
SYSMON	Monitors the instance and its databases

Table 10.2 - Instance-level authorities

To grant SYSADM, SYSCTRL, SYSMAINT, or SYSMON authority to a group, the DBM CFG parameters SYSADM_GROUP, SYSCTRL_GROUP, SYSMAINT_GROUP, and SYSMON_GROUP, respectively, can be assigned to an operating system group.

For example, to give SYSADM authority to the operating system group *myadmns*, you can issue this command:

```
update dbm cfg using SYSADM_GROUP myadmns
```

Each DB2 instance has its own authority group definitions. On Windows, these parameters are empty by default, which means the Local Administrators group will have SYSADM authority. In DB2 9.7, the DB2ADMNS group (if extended security is enabled) and the LocalSystem Account will also have SYSADM authority. The authorization ID for the LocalSystem account is SYSTEM. On Linux, the instance owner group is the default SYSADM group.

Figure 10.4, extracted from the DB2 Information Center, shows the different instance-level authorities and the different functions they can perform. As the figure illustrates, a SYSADM authority includes all the SYSCTRL functions and more. A SYSCTRL authority includes all the SYSMAINT functions and more, and so on.

SYSADM
- Update and restore a database manager configuration parameters (DBM CFG) including specifying groups that have SYSADM, SYSCTRL, SYSMAINT AND SYSMON
- Grant and revoke table space privileges
- Upgrade and restore a database

SYSCTRL
- Update a database, node, or distributed connection services (DCS) directory
- Restore to a new or existing database
- Force users off the system
- Create or drop a database (NOTE: automatically gets DBADM authority)
- Create, drop, or alter a table space
- Restore to a new or existing database
- Use any table space

SYSMAINT
- Back up a database or table space
- Restore to an existing database
- Roll forward recovery
- Start or stop an instance
- Restore or quiesce a table space, and query it's state
- Run tracing
- Database system monitor snapshots
- Reorganize tables
- Use RUNSTATS and update log history files

SYSMON
- GET DATABASE MANAGER MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST commands: ACTIVE DATABASES, APPLICATIONS, DATABASE PARTITION GROUPS, DCS APPLICATIONS, PACKAGES, TABLES, TABLESPACE CONTAINERS, TABLESPACES, UTILITIES--
- RESET MONITOR
- UPDATE MONITOR SWITCHES
- APIs: db2GetSnapshot and db2GetSnapshotSize, db2MonitorSwitches, db2mtrk, db2ResetMonitor
- All snapshot table functions, without running SNAP_WRITE_FILE
- Can connect to a database

Figure 10.4 - Instance-level authorities and their functions

10.2.2.2 Database-level authorities

Table 10.3 lists the database-level authorities.
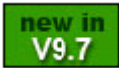
Authority	Description
SECADM	Manages security within a database
DBADM	Administers a database
new in V9.7 ACCESSCTRL	Grants and revokes authorities and privileges (other than SECADM, DBADM, ACCESSCTRL, and DATAACCESS authority. Note that SECADM authority is required to grant and revoke these authorities)
new in V9.7 DATAACCESS	Provides the ability to access data in a database.
SQLADM	Monitors and tunes SQL queries
WLMADM	Manages workloads
EXPLAIN	Users who need to explain query plans (EXPLAIN authority does not give access to the data itself)

Table 10.3 - Database-level authorities

In order to grant a database-level authority, use the GRANT statement. For example, to grant DBADM on the **SAMPLE** database to user *bob*, use:

```
connect to sample
grant DBADM on database to user bob
```

In the above example, you first need to connect to the database, in this case the **SAMPLE** database, and then you can grant DBADM to a user. To grant DBADM authority and any other database-level authorities, you need to be SECADM.

Note that a DBADM cannot create table spaces, even though they are objects inside a database, because a table space deals with containers (disks) and buffer pools (memory) which are physical resources of the system. This can be done by a SYSADM.

Figure 10.5, taken from the DB2 Information Center, shows the different database-level authorities and the functions they can perform.
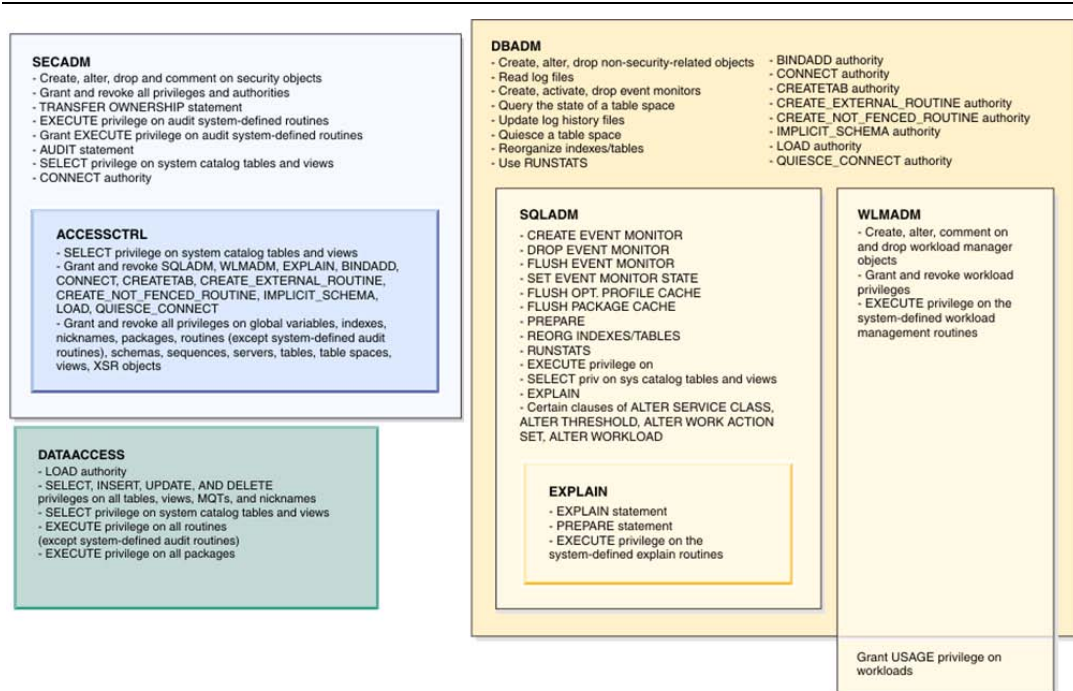
Figure 10.5 - Database-level authorities and their functions

Note:

In DB2 9.7, to provide for better data privacy and governance compliance, the authorization model has been updated to clearly separate the duties of the system administrator, the database administrator, and the security administrator.

In general, the functional scope of several authorities has been reduced compared to previous DB2 versions. For example, a SYSADM no longer has the rights to access data from any database. A DBADM no longer has the rights to access data for the database he administers. On the other hand, SECADM has gained more functionality such as the ability to grant and revoke authorities and privileges to users, roles, and groups.

New authorities have also been created to allow for more granularity and control of your system security. This also minimizes the risk of data exposure by not granting users more than what they need to do their job.

The DB2 9.7.2 refresh also includes auditing improvements that allow for the replay of past database activities. For example, if you need to analyze how a given request that happened a few years back affected some tables, you can now use the database audit information to obtain what you need for that analysis.

10.2.2.3 Enabling SYSADM and DBADM to work the same as versions of DB2 prior to 9.7

If you would like a SYSADM to behave the same as it did in versions of DB2 prior to DB2 9.7, there are two cases to consider:

- If the SYSADM is the creator of the database, then it automatically receives DATAACCESS, ACCESSCTRL, SECADM and DBADM authority for that database. This gives the SYSADM the same abilities as in versions of DB2 prior to 9.7.

- If the SYSADM is not the creator of the database, then to obtain the same capabilities as in previous versions of DB2 (except SECADM); a SECADM must GRANT DBADM with DATAACCESS and ACCESSCTRL (which is the default) to the SYSADM on the given database.

A few cases to consider for a SECADM:

- The default SECADM is the creator of the database.

- If a user with SECADM authority grants SECADM to a user with SYSADM authority, then the SYSADM can grant SECADM to other users.

- If a user with SECADM authority grants DBADM to a user, the DBADM also receives DATAACCESS and ACCESSCTRL by default.

If you are migrating a DB2 9.5 database, the capabilities of SYSADM and DBADM will not change because DB2 automatically grants DBADM, DATAACCESS and ACCESSCTRL to the SYSADM group upon migration. DB2 also automatically grants DATAACCESS and ACCESSCTRL to every authorization ID that holds DBADM upon migration. In addition, DB2 automatically grants SECADM to the user ID doing the migration if there is no authorization ID of type USER that holds SECADM in the database. SYSADM loses its implicit ability to grant or revoke DBADM and SECADM which can now only be performed by SECADM.

10.2.3 Roles

Roles allow a security administrator to assign privileges or authorities to several users or groups. Roles are very similar to groups, but they are defined within DB2, and therefore, provide some advantages. For example, the privileges and authorities granted to roles are always used when you create objects like views or triggers, which is not the case for groups. On the other hand, you cannot assign instance-level authorities such as SYSADM to a role, only privileges and database-level authorities; while for a group, all privileges and authorities can be assigned.

Working with roles requires following several steps:

1. A security administrator (SECADM) must first create a role using a command like

   ```
   CREATE ROLE TESTER
   ```

2. Next, a DBADM must grant privileges or authorities to the role. For example, to grant SELECT privilege on tables **STAFF** and **DEPT** in the **SAMPLE** database to role *TESTER*, issue:

```
GRANT SELECT ON TABLE STAFF TO ROLE TESTER
GRANT SELECT ON TABLE DEPT TO ROLE TESTER
```

3. Next, the security administrator grants the role ***TESTER*** to users RAUL and JIN:

```
GRANT ROLE TESTER TO USER RAUL, USER JIN
```

4. Next, if JIN were to leave the TEST department, the security administrator revokes the role ***TESTER*** from user JIN:

```
REVOKE ROLE TESTER FROM USER JIN
```

10.3 Group privilege considerations

If you decide to use groups instead of roles, take into consideration the following:

- When a group is granted privileges, members of the group are granted implicit privileges inherited through group memberships.

- When a user is removed from a group, they lose the implicit group privileges, but still retain any previous privileges that were explicitly granted. Privileges that were explicitly given to a user must be explicitly revoked from the user.

10.4 The PUBLIC group

DB2 defines an internal group called PUBLIC. Any user identified by the operating system or network authentication service is implicitly a member of the PUBLIC group. When a database is created, certain privileges are granted to PUBLIC automatically:

- CONNECT,
- CREATETAB,
- IMPLICIT SCHEMA,
- BINDADD

For added security, we recommend revoking all privileges from the PUBLIC group as shown below:

```
REVOKE CONNECT ON DATABASE FROM PUBLIC
REVOKE CREATETAB ON DATABASE FROM PUBLIC
REVOKE IMPLICIT_SCHEMA ON DATABASE FROM PUBLIC
REVOKE BINDADD ON DATABASE FROM PUBLIC
```

10.5 The GRANT and REVOKE statements

The GRANT and REVOKE statements are part of the SQL standard, and are used to give or remove privileges to a user, group or role. The user issuing this command must have at least ACCESSCTRL authority. Below are some examples of these statements:

To grant the SELECT privilege on table T1 to the user USER1:

```
GRANT SELECT ON TABLE T1 TO USER user1
```

To grant all privileges on table T1 to the group GROUP1:

```
GRANT ALL ON TABLE T1 TO GROUP group1
```

To revoke all privileges on table T1 from group GROUP1:

```
REVOKE ALL ON TABLE T1 FROM GROUP group1
```

To grant EXECUTE privilege on procedure p1 to user USER1:

```
GRANT EXECUTE ON PROCEDURE p1 TO USER user1
```

To revoke EXECUTE privilege on procedure p1 from user USER1:

```
REVOKE EXECUTE ON PROCEDURE p1 FROM USER user1
```

10.6 Authorization and privilege checking

The easiest way to check for authorization and privileges is through the Control Center. *Figure 10.6* illustrates how to launch the Table Privileges dialog for the ***EMPLOYEE*** table from the Control Center.
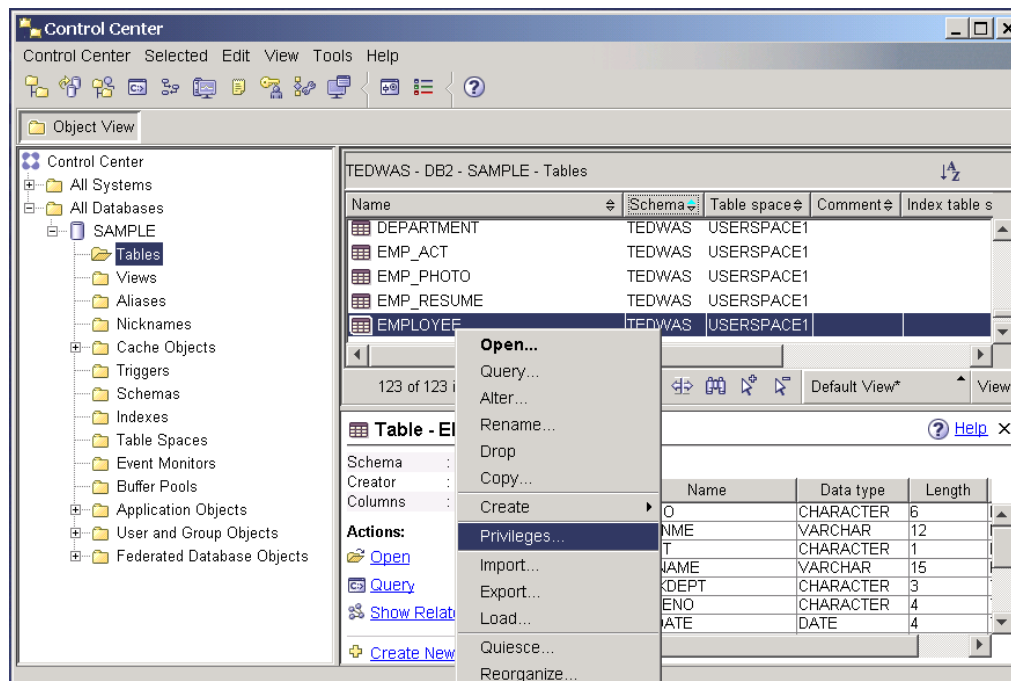
Figure 10.6 - Launching the Table Privileges dialog

As shown by *Figure 10.6*, you select the desired table, right-click on it, and choose *Privileges*. Once selected, the *Table Privileges* dialog box appears as shown in *Figure 10.7*. This figure also explains the different fields and elements of the dialog box.
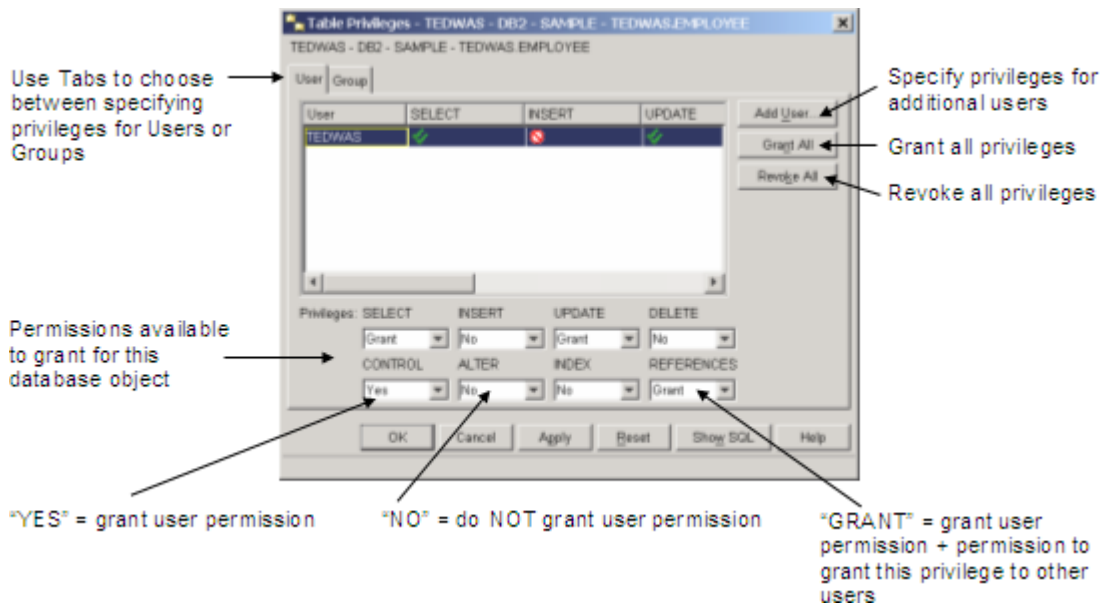
Figure 10.7 – The Table Privileges Dialog box

Alternatively, you can query the DB2 SYSCAT catalog views which contain the authorization information. For example, if you would like to know if user *DB2ADMIN* has SELECT privilege on table T2, and would like to know who granted this privilege, you could run a query like this:

```
SELECT grantor, grantee, selectauth
  FROM syscat.tabauth
 WHERE tabname = 'T2'


GRANTOR             GRANTEE             SELECTAUTH
-------------------------------------------------
ARFCHONG            DB2ADMIN                 Y
```

In the above example, user *ARFCHONG* granted SELECT privilege to user *DB2ADMIN*.

10.7 Extended Security on Windows

To prevent access through the Windows operating system to DB2 files and directories (such as the ones where DB2 stores instance information), DB2 enables by default extended security at installation time. Extended security creates two groups:

- DB2ADMNS: This group and local administrators will have complete access to all DB2 objects through the operating system.

- DB2USERS: This group will have read and execute access to all DB2 objects through the operating system.

new in V9.7

With DB2 9.7 the members of the DB2ADMNS group will automatically have SYSADM authority in DB2 if extended security is enabled and the database configuration parameter SYSADM_GROUP is not set.

10.8 Summary

This chapter covered the security aspects of DB2, beginning with a comprehensive discussion of the differences between and importance of authentication and authorization. From there, we looked at the various authority levels that provide security for the instance and the database.

Next, we covered the new concept of roles and how they can be used to your advantage with regard to security and the limitations of setting security through groups. In particular, the PUBLIC group was discussed, along with suggestions on how to secure it so that general users are blocked from the data server.

In addition, the GRANT and REVOKE statements were examined, and finally, we looked at how to use the Control Center and system catalog tables to check authorization and privilege levels.

10.9 Exercises

So far, you have been using the instance administrator account (SYSADM) to issue all the database commands. This account has wide access to all the utilities, data, and database objects. Therefore, it is very important to safeguard this account in order to avoid accidental or deliberate data loss. In most cases, you will want to create different user accounts or groups with a limited set of permissions. In this exercise, you will create a new user account, then assign it specific privileges.

Part 1 - Working with privileges

In this part of the exercise you will practice how to grant and revoke privileges to users using the Control Center.

Procedure

5. Open the Windows Computer Management console by right-clicking on the *My Computer* icon on the desktop, and selecting the *Manage* menu item.

6. Expand the *System Tools* selection in the tree on the left pane of the window and then expand the *Local Users and Groups* folder. Right-click on the *User* folder and select the *New User* item.

7. In the *New User* dialog window, enter the following information: in the *User name* field, enter **customer** and in the *Full name* field, enter **Customer1**. In the

Description field, enter **A typical bookstore customer**. In the *Password* and *Confirm password* fields, enter **ibmdb2ibm**. Remove the checkmark from the *User must change password on next logon* option, and click the *Create* button to create the new user, and then the *Close* button to dismiss the dialog window.

8. Open the Control Center, and choose the advanced view. To switch to the advanced view, select *Tools ->Customize Control Center* menu. Then select the *Advanced* option and click the *OK* button.

9. Expand the Control Center object tree in the left object tree pane to *All Databases -> EXPRESS -> Tables.*

10. Grant the required privileges to the newly created user. From the list of tables in the **EXPRESS** database, right click the **CUSTOMERS** table, and select the *Privileges* item to view the *Table Privileges* dialog window.

11. Click the *Add User* button and select the **customer** user just created. Click the *OK* button to close the *Add User* dialog box.

12. You will notice that the **customer** user has been added to the user list, but has no privileges assigned. To grant SELECT, INSERT, UPDATE, and DELETE privileges to the user, change each drop down box to *Yes*. An Internet customer should be able to view/add/update/delete their account data. We do not give the user the other permissions because they do not require them. Click the *OK* button to close the Table Privileges dialog window and accept the changes you made.
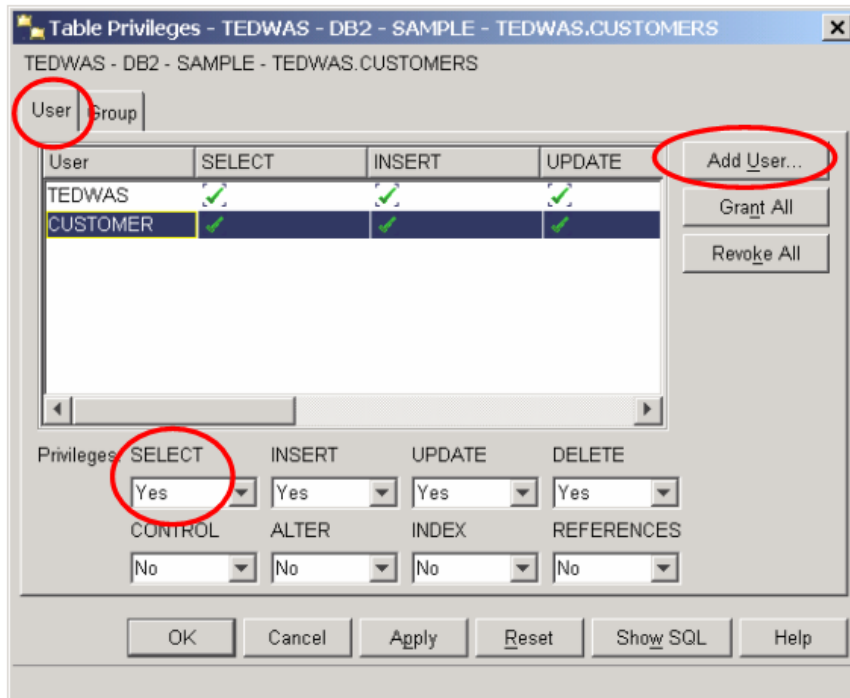
13. Repeat Steps 7-9 for the **BOOKS** and **SALES** tables. For the **BOOKS** table, only grant the SELECT privilege because the customer should not be able to modify any of the store's inventory data. For the **SALES** table, only grant the SELECT and INSERT privileges. The customer should NOT have the DELETE or UPDATE privilege because only store employees should have access to modify sales transactions.

14. Connect to the database using the **customer** user ID created above using the DB2 Command Window as follows:

```
db2 connect to express user customer using ibmdb2ibm
```

Try to SELECT data from the **customers** table. What happens? Try to DELETE or UPDATE data in the **SALES** table. What happens?

Part 2 - Working with SYSADM, DBADM and SECADM authorities

In this part of the exercise, you will practice how to assign SYSADM and DBADM authorities, and understand how these authorities work.

Procedure

1. Follow the same steps as in part 1 to create one new user: **mysysadm**

2. Create the **mysysadmgrp** Windows group. Follow the same steps used to create a user, but instead of right-clicking on the *Users* folder, right-click on the *Groups*

folder and choose *New Group*. For the Group name field, enter *mysysadmgrp*. In the Members section, click on *Add* to add a new member, and enter *mysysadm*. Click on the *Check Names* button to confirm you entered the member correctly. If you did, click on *Create*, then on *Close*.

3. Thus far you have created one user *mysysadm* and one group *mysysadmgrp* to which user *mysysadm* belongs. This has all been done on the Windows operating system. We now need to inform to DB2 that we want the *mysysadmgrp* group to be the SYSADM group using this command from the DB2 Command Window:

   ```
   db2 update dbm cfg using SYSADM_GROUP mysysadmgrp
   ```

   Since the SYSADM_GROUP parameter is not dynamic, you need to stop and start the instance. The force option in **db2stop** will guarantee all connections are removed prior to the **db2stop**.

   ```
   db2stop force
   db2start
   ```

4. Connect to the SAMPLE database using the *mysysadm* user from the DB2 Command Window, and issue a SELECT * statement on table STAFF. Note that you need to use the correct schema that was used when you created this table. In the example below we use *arfchong* as the schema.

   ```
   db2 connect to sample user mysysadm using ibmdb2ibm
   db2 select * from arfchong.staff
   ```

   You should receive an error message like this. Why? Are you not SYSADM?
   ```
   SQL0551N  "MYSYSADM" does not have the required authorization or
   privilege to perform operation "SELECT" on object "ARFCHONG.STAFF".
   SQLSTATE=42501
   ```

   Starting with DB2 9.7, SYSADM does not get DBADM authority by default, that's why you received the error.

5. Using the Windows user that was used to create the **SAMPLE** database, connect to the database. In the example, **ARFCHONG** is the user. Next grant DBADM without DATAACCESS to *mysysadm* user, and try the SELECT on STAFF again as *mysysadm*. Did it work? Why?

   ```
   db2 connect to sample user arfchong using ibmdb2ibm
   db2 grant dbadm without dataaccess on database to user mysysadm
   db2 connect to sample user mysysadm using ibmdb2ibm
   db2 select * from arfchong.staff
   ```

   As you saw, you still get the same error message even after *mysysadm* was granted DBADM. This behavior is expected because we included the clause WITHOUT DATAACCESS which means that the DATAACCESS authority was not

included, therefore *mysysadm* still does not have the authority to access the data. This shows you an example of how you can restrict access to data to the DBADM.

6. Let's now grant DATAACCESS to *mysysadm* and try again the SELECT.

```
db2 connect to sample user arfchong using ibmdb2ibm
db2 grant DATAACCESS on database to user mysysadm
db2 connect to sample user mysysadm using ibmdb2ibm
db2 select * from arfchong.staff
```

Now your SELECT should work!. This exercise shows you the new behavior for SYSADM and DBADM starting with DB2 9.7. The main idea you should take from this exercise is that now there is a separation between data access, and what a SYSADM and DBADM can do.

7. Reset the value of SYSADM_GROUP to NULL so that the Local Administrator group and LocalSystem account again become SYSADM:

```
db2 update dbm cfg using sysadm_group NULL
db2stop force
db2start
```

11

Chapter 11 – Backup and Recovery

In this chapter, we discuss DB2 database logging, how to make a full or partial copy of your database using the BACKUP utility, and how to recover your data using the RESTORE utility.

> **Note**:
>
> For more information about logging, backup, and recovery, watch this video:
> http://www.channeldb2.com/video/video/show?id=807741:Video:4282

11.1 Database Logging

If you were working with a text editor, every time you want to ensure your document is saved, you click the *save* button. In the database world, a COMMIT statement does just that. Every time a COMMIT statement is executed, you guarantee that whatever changes were made to the data, they will be saved somewhere.

In a similar way, when you work with a text document, sometimes you will see at the bottom right corner a brief message saying "auto-saving". In the database world, this happens as well, because any operation you perform against the data, such as an UPDATE, INSERT or DELETE, will be saved somewhere as you perform it.

That "somewhere" in the preceding paragraphs refers to the database logs. The database logs are stored on disk and are used to record actions of transactions. If there is a system or database crash, logs are used to playback and redo committed transactions during a recovery.

Figure 11.1 provides a graphical overview of what happens when you are working with a database in terms of logging.
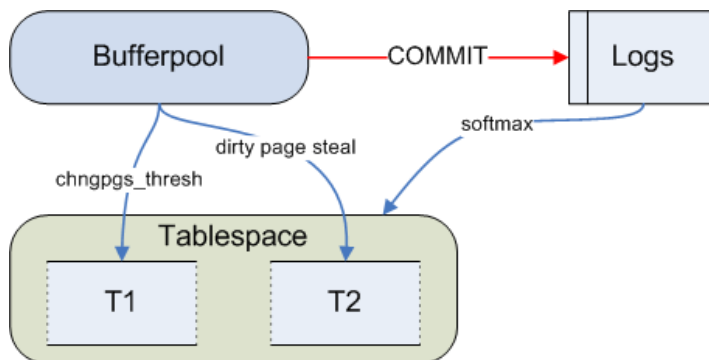
Figure 11.1 – Database logging

In *Figure 11.1*, we see a table space and logs. Both of them reside on disks, although we recommend that they are not kept on the same disk. When an UPDATE operation takes place for example, the pages for the row(s) in question will be brought to the buffer pool (memory). The update changes are performed in the buffer pool, and the old and new values will be stored in the log files, sometimes immediately, and sometimes when a log buffer is full. If a COMMIT is issued after the UPDATE, the old and new value will be stored in the log files immediately. This process is repeated for many other SQL operations that are performed on the database. Only when certain conditions are met, such as reaching the change page threshold specified in the CHNGPGS_THRES parameter, are the pages in the buffer pool "externalized" and written to the table space disk.   The CHNGPGS_THRES parameter indicates the percentage of the buffer pool with "dirty" pages, that is, pages containing changes.

From a performance point of view, it does not make sense to perform two writes for each COMMIT operation: One to write to the logs, and another one to write to the table space disk; that's why "externalization" of the data to the table space disk only occurs when parameters such as the CHNGPGS_THRES threshold are reached.

11.2 Types of logs

There are two types of logs:

- **Primary logs**

  These are pre-allocated and the number of primary logs available is determined by the LOGPRIMARY database configuration parameter.

- **Secondary logs**

  These are dynamically allocated as needed by DB2. The maximum number of secondary logs is set by the database configuration parameter LOGSECOND. Dynamically allocating a log is costly; therefore, for day to day operations, stay within your primary log allocation. Secondary log files are deleted when all the connections to a database are terminated.

Infinite logging is possible if you set LOGSECOND to a value of -1; however, this is not recommended as you may run out of file system space.

11.3 Types of logging

There are two types of logging: circular logging (default) and archive logging.

11.3.1 Circular logging

Circular logging is the default, and is enabled when both of the LOGARCHMETH1 and LOGARCHMETH2 database configuration parameters are set to OFF. These parameters indicate the method used to archive the logs, but if you turn them off, that means you do not want to archive the logs, which is how circular logging works. *Figure 11.2* outlines an example illustrating circular logging.
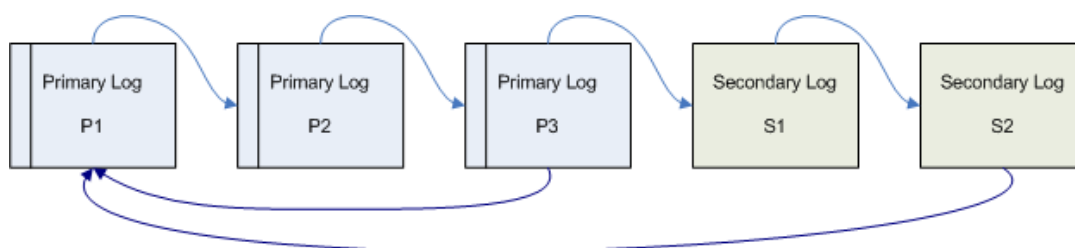
Figure 11.2 – Working with primary and secondary logs

In *Figure 11.2* there are 3 primary logs, therefore we can assume that the value of the LOGPRIMARY parameter is 3. For simplicity, there is only one transaction being performed in this example. As the transaction is performed, the log file P1 starts filling up, and then P2. If a commit occurs and the information is later externalized to the table space disk, then P1 and P2 can be overwritten, because the information is no longer needed for crash recovery (which will be discussed in more detail later in this chapter). If, on the other hand, the transaction is so long that it uses P1, P2, P3, and still needs more log space because the transaction has not been committed nor externalized, then a secondary log (S1 in the figure) is dynamically allocated. If the transaction continues, more secondary logs are allocated until the maximum LOGSECOND logs are allocated. If still more logs are needed, an error message indicating a log full condition is reached will be returned to the user, and the transaction will be rolled back. Alternatively, you can configure DB2 using the BLK_LOG_DSK_FUL configuration parameter to continue writing to the logs every 5 minutes while letting some transactions hang. This gives the DBA some time to find new space, so that the transaction can continue.

Circular logging allows you to recover from crash recovery, but if you want to recover your data to a given point in time, the closest available time would when you took your last offline backup.

11.3.2 Archive logging

In archive logging, also known as log retain logging, the log files are not overwritten, but are kept, either online or offline. Online archive logs remain with the active logs which are still needed for crash recovery. Offline archive logs are moved to another media such as tape, and this can be done with USEREXIT routines, Tivoli Storage Manager, or other third party archival products.

To enable archive logging set the database configuration parameters LOGARCHMETH1 or LOGARCHMETH2 (or both) to a value other than OFF. Another way to enable it is to set the LOGRETAIN configuration parameter to RECOVERY. This will automatically cause LOGARCHMETH1 to be set to LOGRETAIN. However, the LOGRETAIN parameter is deprecated and has been left mainly for compatibility with older versions of DB2.

Archive logging is normally used in production systems; because the logs are kept, this allows for database recovery back to point in time as early as the oldest log file. With archive logging, a DBA can recover from errors caused by humans. For example, if a user of a system inadvertently starts performing an incorrect transaction that lasts for days, then when the problem is detected, the DBA can restore the system back to the time before the problem was introduced. However, there may be some manual manipulation required for the transaction to rerun correctly.

Archive logging is required for roll forward recovery and on-line backup. *Figure 11.3* depicts the archive logging process.
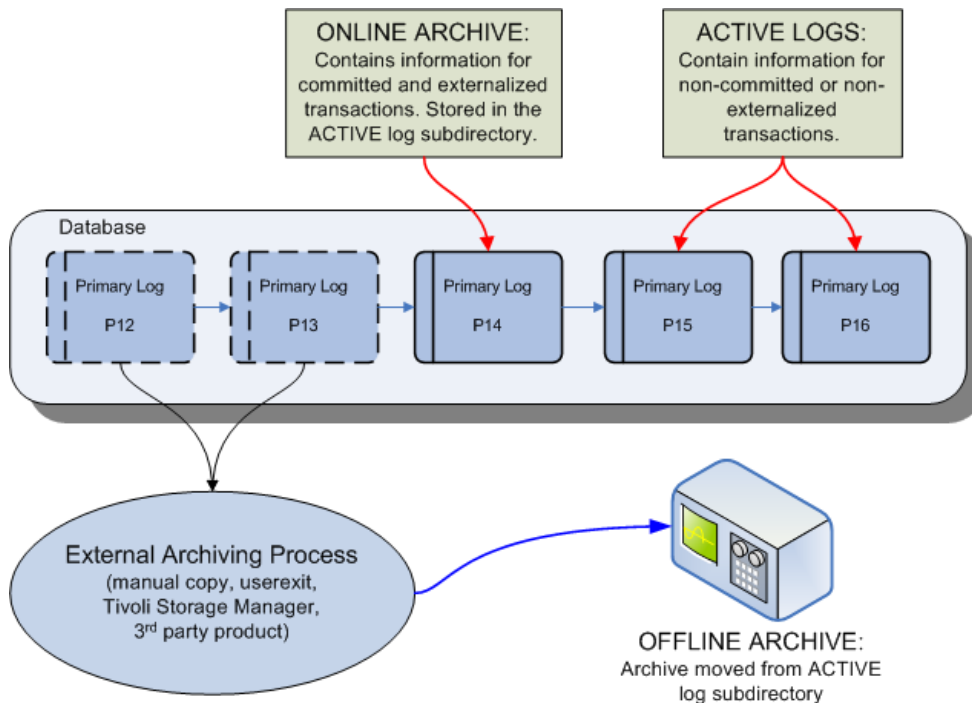
Figure 11.3 – Archive logging

11.4 Database logging from the Control Center

You can configure database logging from the Control Center by right-clicking on the database in question, and choosing *Configure Database Logging.* This is depicted in *Figure 11.4*
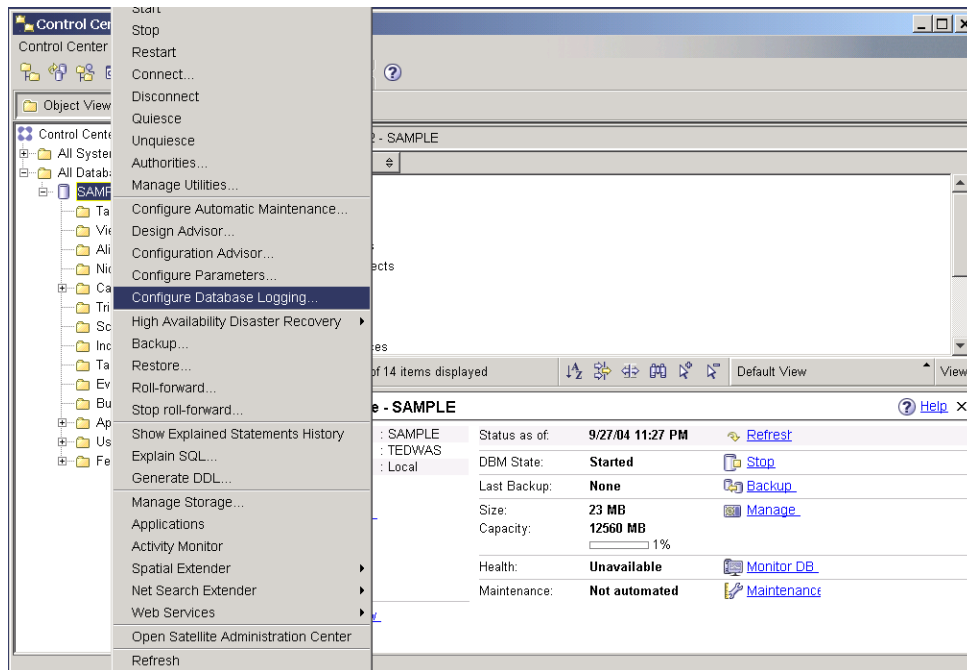
Figure 11.4 – Configuring database logging from the Control Center.

Figure 11.5 shows the Database Logging Wizard, where you can choose circular logging or Archive logging.
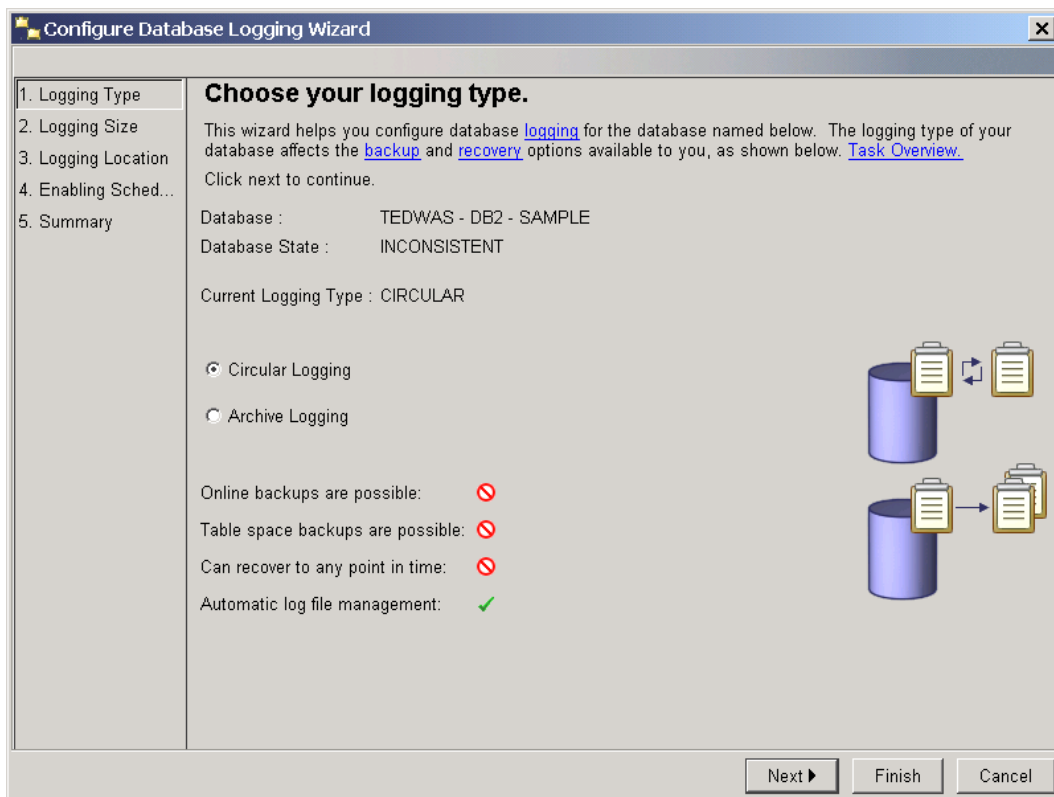
Figure 11.5 – Database Logging Wizard

11.5 Logging parameters

There is a number of DB CFG parameters related to logging. *Table 11.1* lists the main parameters.

Parameter	Description
logbufsz	The amount of memory to use as a buffer for log records before writing these records to disk
logfilsz	The size of each configured log, in number of 4KB pages
logprimary	The number of primary logs of size logfilsz that will be created
logsecond	The number of secondary log files that are created and used for recovery, if needed.
newlogpath	The database active and online archive logs are initially created under your database directory, in subdirectory SQLOGDIR. You can change this location by changing the value of this configuration parameter to

	point to a different directory or to a device.
mirrorlogpath	To protect the logs on the primary log path from disk failure or accidental deletion, you can specify that an identical set of logs be maintained on a secondary (mirror) log path
logarchmeth1 logarchmeth2	Specifies a location other than the active log path to store archive log files. If both of these parameters are specified, each log file is archived twice. This means that you will have two copies of archived log files in two different locations. Possible values are OFF (which means circular logging is enabled), LOGRETAIN, USEREXIT, DISK, TSM, VENDOR
loghead	The name of the log file that is currently active
softmax	Limits cost of crash recovery
overflowlogpath	Specifies a location for DB2 to find log files that are needed for a rollforward operation. Similar to the OVERFLOW LOG PATH option of the ROLLFORWARD command.
blk_log_dsk_ful	Set to prevent disk full errors from being generated when DB2 cannot create a new log file in the active log path. Instead, DB2 will attempt to create the log file every five minutes until it succeeds. Unblocked, read-only SQL may continue.
max_log	Percent of max active log space by one transaction
num_log_span	Number. of active log files for 1 active UOW
mincommit	Number of commits to group before writing to disk

Table 11.1 – Logging parameters

11.6 Database backup

The DB2 backup command allows you to take a snapshot copy of your database at the time the command is executed. The simplest syntax that you can use to run this command is:

```
BACKUP DATABASE <dbname> [ TO <path> ]
```

Most commands and utilities can be performed online or offline. Online implies that other users may be connected and performing operations on the database while you execute your command. Offline means that no other users are connected to the database while you perform your operation. To allow for an online operation, add the keyword ONLINE to the

command syntax, otherwise, by default the command will be assuming you are executing it offline.

For example, if you want to back up the database **SAMPLE** to the path C:\BACKUPS you can issue this command from the DB2 Command Window or Linux shell:

```
db2 BACKUP DB sample TO C:\BACKUPS
```

Note that the C:\BACKUPS directory must exist prior to executing the command. Also ensure there are no connections to the database when you execute the above command, otherwise you will receive an error message since an offline backup cannot be performed when there are connections.

To find out if there are connections to databases in an instance, issue this command from the DB2 command window or Linux shell:

```
db2 list applications
```

To force all the connections from all databases in an instance, issue this command from the DB2 command window or Linux shell:

```
db2 force applications all
```

You may not want to run this last command in a production environment with many users; otherwise you would receive many calls from angry co-workers! Note as well that the last command runs asynchronously. This means than when you try to run the backup command right after, it may still not work. Wait a few seconds, and repeat the backup command if you received an error the first time.

After a successful execution of the backup command, a new file containing the backup database image is created. The name of this file follows the convention shown in *Figure 11.6.*

Linux/UNIX/Windows

Figure 11.6 – Backup image naming convention

A type of "0" means that the backup is a full backup. A type of "3" means that it is a table space backup. The node is fixed to NODE0000 for non-partitioned databases, which is the case for all DB2 editions except DB2 Enterprise Edition with the DPF feature. The catalog node is also fixed to CATN0000. Refer to the DB2 manuals for more details.

When several backups are taken and stored in the same path, the timestamp at the end of the file name is used to distinguish between the backup images. As we will see on the next section, the RESTORE command can use this timestamp to restore from a specific backup.

11.7 Database recovery

A database recovery implies restoring your database from a backup and/or logs. If you just restore from a backup, you would be recreating the database as it existed at the time the backup was taken.

If archive logging was enabled before the backup, you can not only restore using a backup image, but also from the logs. As we will see in the next section, a roll-forward recovery allows you to restore from a backup, and then apply (roll-forward) the logs to the end of the logs, or to a specific point in time.

Note that the term "recovery" is used often in this section, but the command used for recovery is called RESTORE.

11.7.1 Recovery types

There are three types of recovery:

- **Crash or restart recovery**

  Assume you are working on a desktop computer running important transactions to a DB2 database. Suddenly there is a power outage, or someone accidentally unplugs the power cord: what will happen to the database?

  The next time you start your computer, and start DB2, crash recovery will automatically be executed. In crash recovery, DB2 will automatically run the command RESTART DATABASE and will read and redo/undo the transactions based on the active logs. When this command completes, you are guaranteed that your database will be in a consistent state, that is, whatever was committed will be saved, and whatever was not committed will be rolled back.

- **Version or image recovery**

  This type of recovery implies that you are restoring only from a backup image; therefore, your database would be put in the state it was at the time the backup was taken. Any transactions performed on the database after the backup was taken would be lost.

- **Roll-forward recovery**

  With this type of recovery, you not only RESTORE from a backup image, but you also run the ROLLFORWARD command to apply the logs on top of the backup so that you can recover to a specified point in time. This type of recovery minimizes data loss.

11.7.2 Database restore

Use the RESTORE command to recover a database from a backup image. The following syntax is the simplest that can be used for this command:

```
RESTORE DATABASE <dbname> [from <path>] [taken at <timestamp>]
```

For example, if you had a backup image file of the *sample* database with this name:

You could perform the following:

```
RESTORE DB sample FROM <path> TAKEN AT 20060314131259
```

11.8 Other operations with BACKUP and RESTORE

The following lists some of the things that you can also do with the BACKUP and RESTORE commands. We encourage you to review the DB2 manuals for additional details.

- Backup a database in a 32-bit instance, and restore it on a 64-bit instance

- Restore over an existing database

- Use of a redirected restore when restoring into a system where there are a different number of disks than what was specified in the backup image

- Backup or restore just by table space, rather than the entire database

- Perform delta and incremental backups; delta backups record only the changes from one backup to the next, while incremental backups record all the changes and accumulates them on each backup image

- Backup from flash copy (correct hardware required)

- Recover dropped tables (if the option was enabled for a given table)

- Backup from one platform (for example, Windows) and restoring to another platform (for example, Linux) is not possible. Use db2look and db2move for this scenario. For DB2 editions that support UNIX operating systems, be aware that within the UNIX operating system, some platforms allow for backup and restore from one UNIX platform to another.

11.9 Summary

In this chapter we examined the function of logging in DB2, including the two types of logs (primary and secondary) and the two types of logging (circular and archive), and the

various database parameters related to logging. For each type of logging, we discussed when and why it is used, and how to set it up from the Control Center.

We also looked at how to run backup and restore activities using the DB2 command line, including an in-depth look at the three types of database restore: crash, version and roll-forward.

11.10 Exercises

Although DB2 is able to automate several database maintenance activities, sometimes you want to customize when certain activities occur. In this exercise, you will create a customized nightly backup schedule for the **EXPRESS** database.

Procedure

1. From the Control Center object tree, navigate to *Control Center -> All Databases*. Right-click on the **EXPRESS** database and select the *Backup* item. This launches the *Backup Wizard*.

2. The *Introduction* page of the wizard summarizes the current state of the database including the time of the last backup and logging method. Click the *Next* button to move to the next page of the wizard.

3. On the *Image* page of the wizard, select the destination of the backup image. You will typically select a different physical drive than where the existing database is stored. For now, create a new folder in the file system called `C:\db2backup`, and specify that folder as the backup location. In the wizard, select the *File System* item from the *Media Type* drop-down list. Click the *Add* button, select the folder you just created, and then click the *OK* button. Click the *Next* button to move to the next page of the wizard.

4. You can explore the *Options* and *Performance* pages, but the default options are usually sufficient because DB2 automatically performs the database backup in the most optimal way. Navigate to the *Schedule* page when you are finished exploring.

5. On *Schedule* page, if the scheduler has not yet been enabled, choose to enable it now. Select the system to create the tools catalog on and create a new tools catalog. Specify a schema for the tools catalog and choose to create it in the existing **EXPRESS** database. The tools catalog holds metadata about all the scheduled tasks. Click the *OK* button to continue. Click the *Next* button to move to the next page of the wizard once the tools catalog has been created.
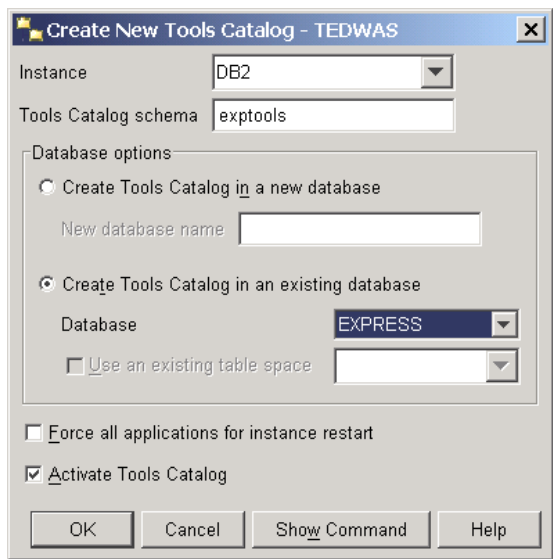
6. On the *Schedule* page, choose to create a schedule for task execution. Schedule the backup to run each day, starting at 1AM. Click the *Next* button to move to the next page.
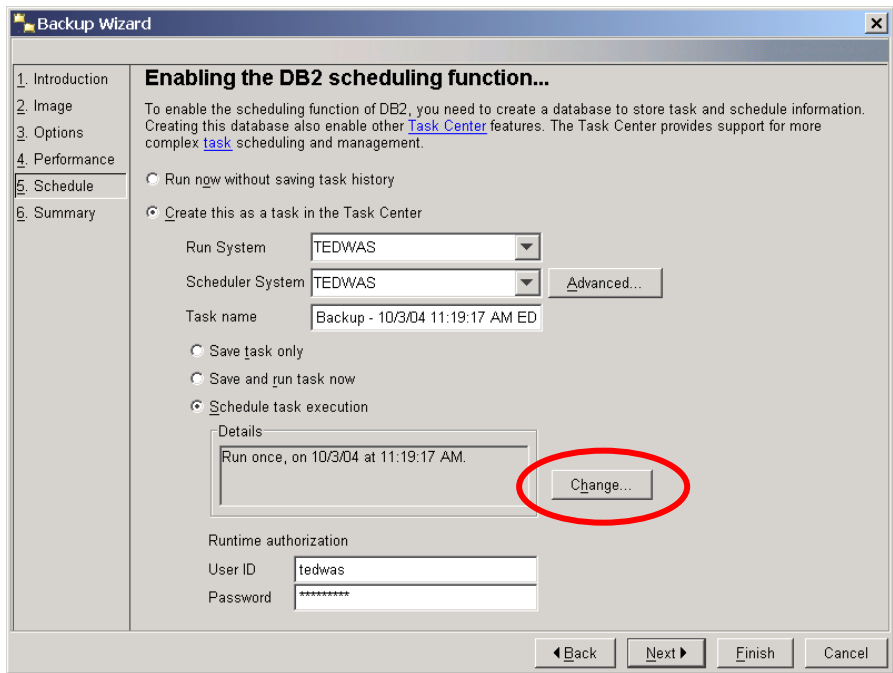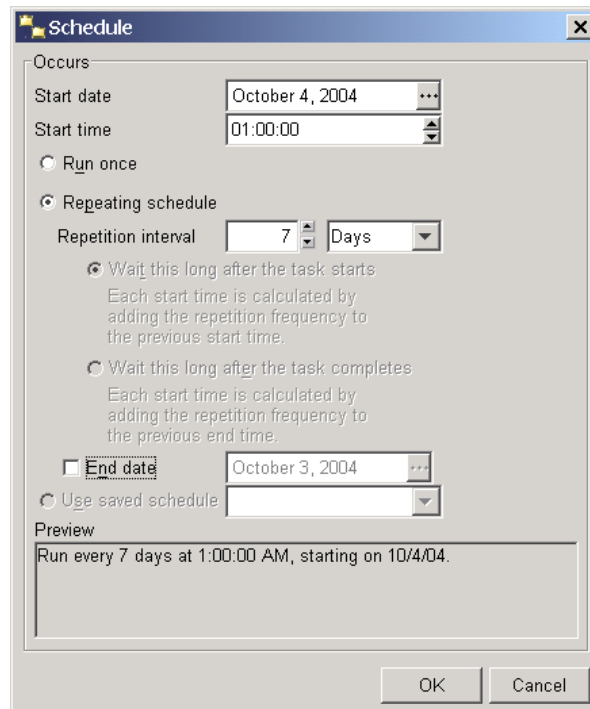
7. On the *Summary* page, you can review the scheduled tasks that will be created. When you have reviewed the changes, click the *Finish* button to create the task.

8. Launch Task Center to view or modify the newly created backup task.

12

Chapter 12 – Maintenance Tasks

This chapter discusses some of the tasks required to keep your database well maintained. The overall direction in DB2 is to automate most of these tasks. DB2 Express-C edition, like all current DB2 editions, includes these automated capabilities. This self management capability is a great benefit to small and medium size companies who cannot hire a full time DBA to manage the data server. On the other hand, if a DBA is hired, he or she will have more free time to perform advanced activities that will add value to a company's bottom line.

> **Note**:
>
> For more information about maintenance tasks, watch this video:
> http://www.channeldb2.com/video/video/show?id=807741:Video:4302

12.1 REORG, RUNSTATS, REBIND

There are three main maintenance tasks in DB2, as depicted in *Figure 12.1*: REORG, RUNSTATS and REBIND.
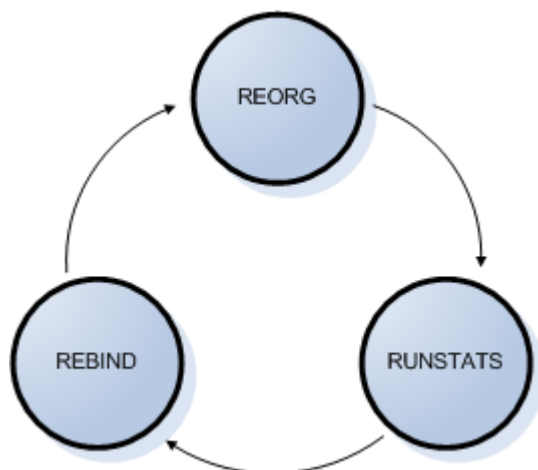
Figure 12.1 – Maintenance tasks: REORG, RUNSTATS, REBIND

Figure 12.1 shows that the maintenance tasks are performed in circular fashion. If a REORG is performed, it is recommended to also run a RUNSTATS, followed by a REBIND. After some time, the tables in a database will be modified due to UPDATE, DELETE and INSERT operations. At that time the cycle will start again with a REORG.

12.1.1 The REORG command

Over time, as you perform INSERT, UPDATE and DELETE operations on your database, your data starts getting more and more fragmented across the database pages. The REORG command reclaims wasted space and re-organizes data to make retrieval more efficient. Tables that are frequently modified will benefit the most from REORG. You can REORG indexes as well as tables, and a REORG can be performed online or offline.

Offline REORG is faster and more efficient, but does not permit access to the table, while an online REORG allows access to the table, but can consume a lot of system resources; this works best for small tables.

Syntax:

```
REORG TABLE <tablename>
```

Example:

```
REORG TABLE employee
```

The REORGCHK command can be used before a REORG to determine whether a table or index needs to be fixed.

12.1.2 The RUNSTATS command

The DB2 Optimizer is "the brain" of DB2. It finds the most efficient access paths to locate and retrieve data. The optimizer is system cost-aware, and uses statistics of the database objects that are stored in catalog tables to maximize the database performance. For example, catalog tables have statistics about how many columns are present in a table, how many rows there are, how many and what type of indexes are available for a table, and so forth.

Statistics information is not updated dynamically. This is by design, as you would not want DB2 to be updating the statistics constantly for every operation performed to the database; this would negatively affect the entire database performance. Instead, DB2 provides the RUNSTATS command to update these statistics. It is essential to keep database statistics up to date. The DB2 optimizer can make radical changes in the access path if it thinks a table has 1 row versus 1 million rows. When database statistics are up to date, DB2 can choose a better access plan. The frequency of statistics gathering should be determined by how often the data in the table changes.

Syntax:

```
RUNSTATS ON TABLE <schema.tablename>
```

Example:

```
RUNSTATS ON TABLE myschema.employee
```

12.1.3 BIND / REBIND

After successfully running a RUNSTATS command, not all queries will use the latest statistics. Static SQL access plans are determined when you first issue a BIND command, so the statistics used at that time may not be the same as the current ones. *Figure 12.2* helps illustrate this idea.
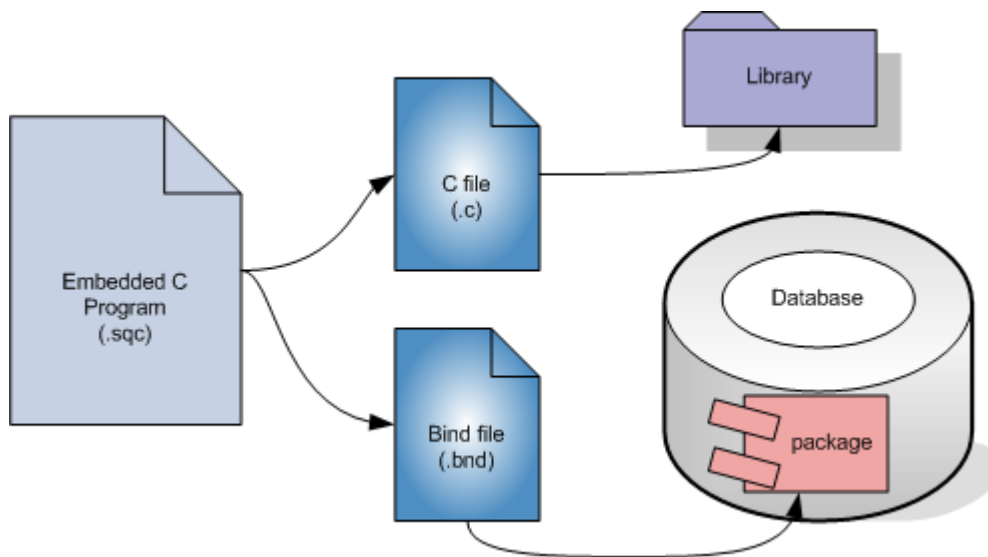
Figure 12.2 – Static SQL bind process

In *Figure 12.2* an embedded C program (stored as a file with a "sqc" extension) is precompiled. After pre-compilation, two files are generated, a ".c" file containing the C code with all the SQL commented out; and a ".bnd" file containing all the SQL statements. The C file with the ".c" extension is compiled as usual with a C compiler, creating a "library" as shown in the top right hand side of the figure. The ".bnd" file is similarly bound, generating a package that is stored in the database. Binding is equivalent to compiling the SQL statements where the best access plan is determined based on the statistics available at the time, and then storing them in the package.

Now, what happens if 1 million rows are inserted into a table used in the SQL for this embedded C program? After the insertion, if a RUNSTATS is performed, the statistics will be updated; however the package will not be automatically updated to recalculate the access path based on the latest statistics. The **db2rbind** command can be used to rebind all the existing packages to take into account the latest stats.

Syntax:

```
db2rbind database_alias -l <logfile>
```

Example:

To rebind all the packages of the **SAMPLE** database and store the output log in the file `mylog.txt`, issue this command:

```
db2rbind sample -l mylog.txt
```

12.1.4 Maintenance tasks from the Control Center

From the Control Center you can REORG and RUNSTATS. *Figure 12.3* shows you how.
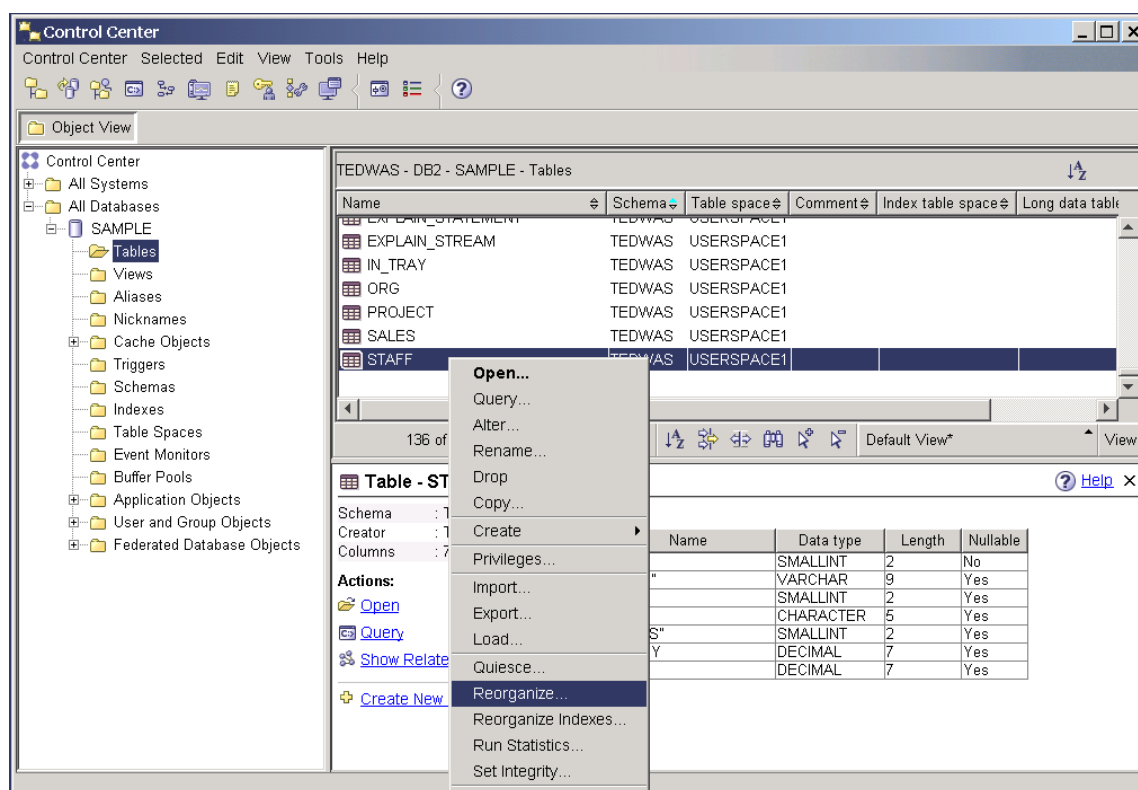
Figure 12.3 – REORG and RUNSTATS from the Control Center

You choose the table you would like to operate against, right-click on it and choose Reorganize (for REORG) or Run Statistics (for RUNSTATS).

12.1.4.1 The database operational view

When you select a database, the database operational view on the bottom right side of the Control Center will provide information about the database, such as its size, when it was backed up last, whether automatic maintenance is set, etc. This view allows you to quickly identify maintenance needs for your database. *Figure 12.4* shows this information.
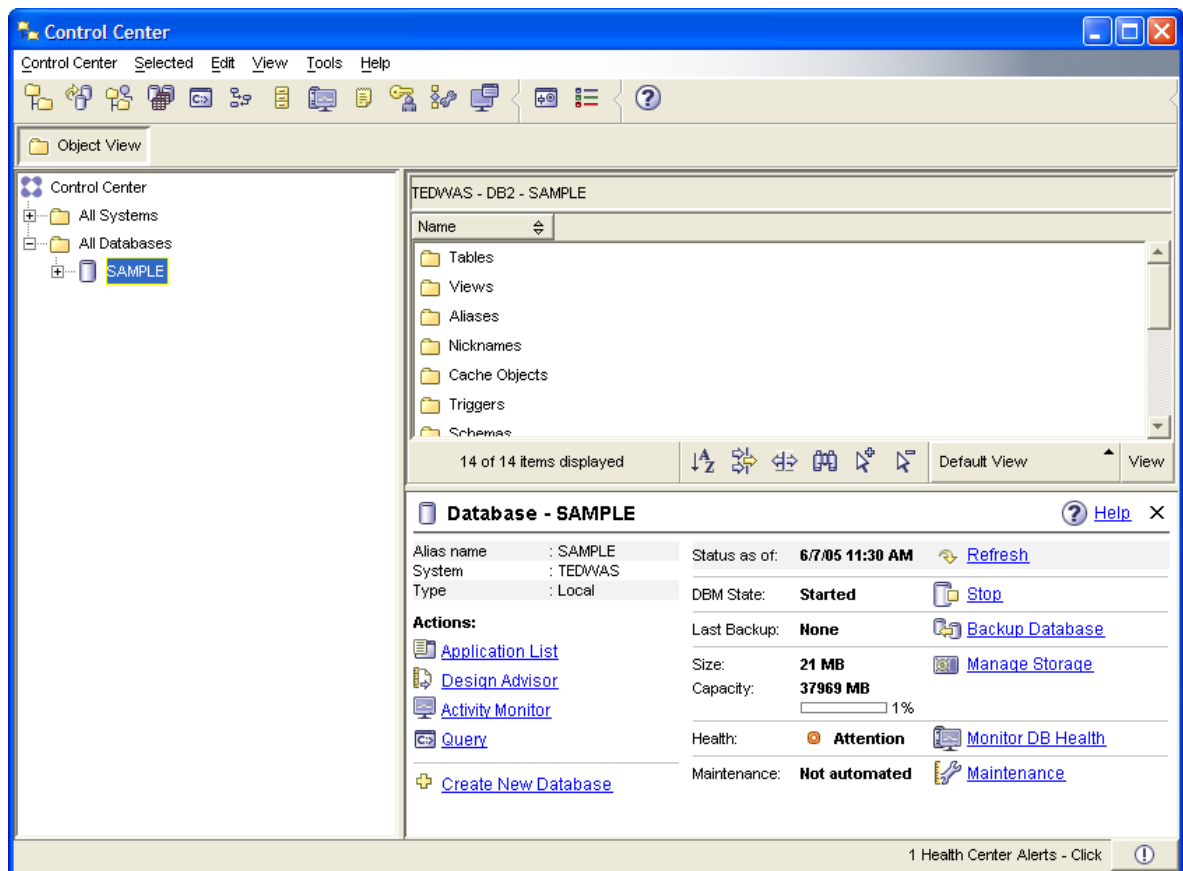
Figure 12.4 – The database operational view from the Control Center

12.2 Maintenance Choices

There are three ways to perform maintenance tasks:

- Manual maintenance

    You perform maintenance activities manually when the need arises

- Create scripts to perform maintenance

    You can create scripts with the maintenance commands, and schedule them regularly for execution.

- Automated maintenance

    Have DB2 automatically look after maintenance for you (REORG, RUNSTATS, BACKUP)

In this section we concentrate on automated maintenance.

Automatic maintenance consists of the following:

- The user defines a *maintenance window* where tasks can be executed with minimal disruption*.* For example, if the system has the least activity on Sundays from 2:00am to 4:00am, this time frame would work as a maintenance window.

- There are two maintenance windows: one for online operations, and another one for offline operations.

- DB2 will perform maintenance operations automatically only when needed during the maintenance windows

From the Control Center, you can launch the *Configure Automated Maintenance* Wizard as shown in *Figure 12.5*.
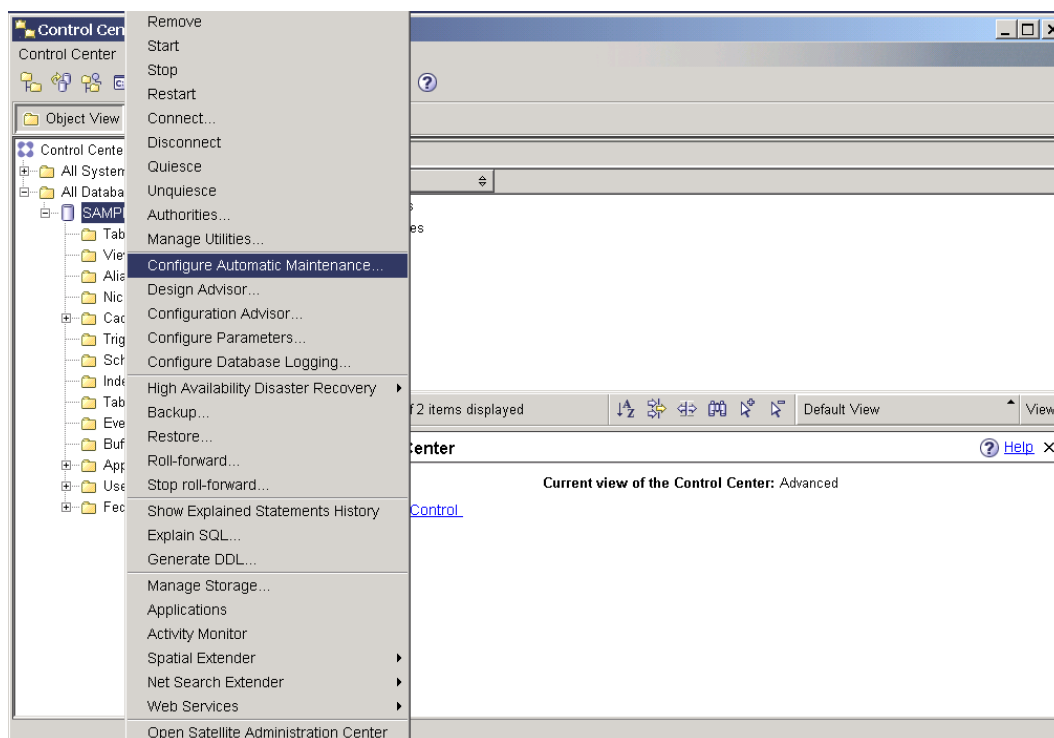
Figure 12.5 – Launching the Configure Automated Maintenance Wizard

Figure 12.6 shows the *Configure Automated Maintenance* Wizard.
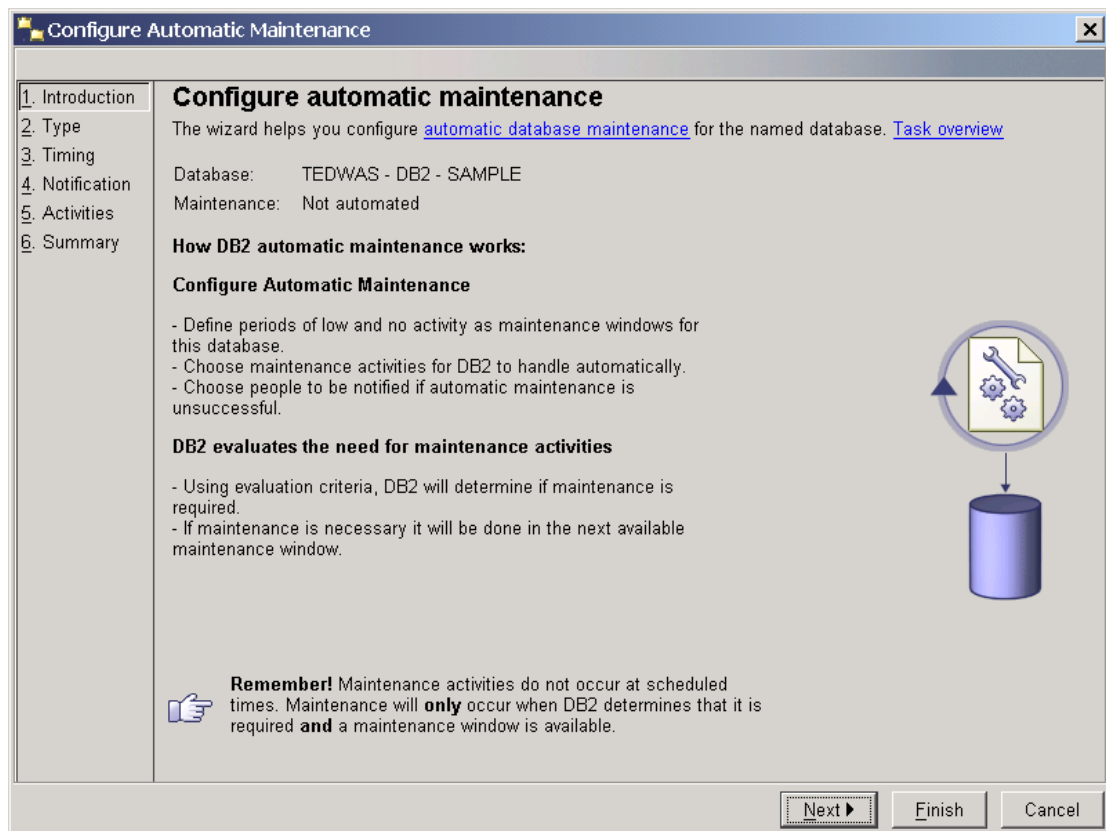
Figure 12.6 –The Configure Automated Maintenance Wizard

12.3 Summary

This chapter examined the importance of maintenance on your databases, including the role of the REORG, RUNSTATS, and REBIND cycle. The REORG command, as its name implies, reorganizes your data to eliminate fragmentation and speed data retrieval. RUNSTATS updates the statistical information used by the DB2 optimization tool to improve data performance. The BIND or REBIND process updates the database packages with the latest access paths.

We also looked at the graphical tools provided in the DB2 Control Center to run maintenance activities in manual, scripted and automatic modes.

12.4 Exercises

In this exercise, you will configure automatic maintenance on the DB2 **SAMPLE** database.

Procedure

1. From the Control Center object tree, right-click on the **SAMPLE** database and select the *Configure Automatic Maintenance* menu item. This launches the *Configure Automatic Maintenance* wizard.

2. The *Introduction* page of the wizard displays the current automated maintenance settings. If you created the database with the automated maintenance option, then automated maintenance is already configured. You can use this wizard to re-configure the automated maintenance options. Click the *Next* button to move to the next page of the wizard.

3. The *Type* page of the wizard asks you to choose between disabling all automated maintenance, and changing your automated maintenance settings. Select the option to change the current automated maintenance settings. Click *Next*.

4. The *Timing* page of the wizard asks you to specify the maintenance windows. Configure the database to be offline Saturday and Sunday night from midnight to 6AM, as shown below. Click the *Change* button beside the offline maintenance window preview pane and choose the desired times. After specifying the required information, click the *OK* button to return to the wizard. Leave the online window as is (online maintenance can occur anytime). Click the *Next* button.
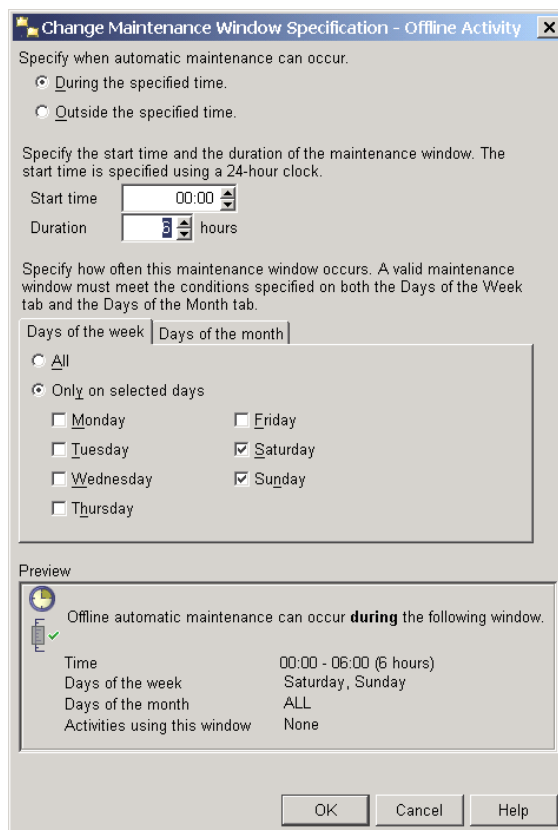
5. On the *Notification* page of the wizard, you can set up a contact in case an automated maintenance activity fails. Skip this step for now. Click the *Next* button

6. On the *Activities* page of the wizard, you can choose to individually automate or not to automate specific activities as well as choose to be notified of particular

activities. In this example, ensure that all the *Automate* checkboxes are checked and the *Notify* checkboxes are unchecked. Click the *Next* button.

7. Before proceeding to the next page of the wizard, you should configure the backup location of the database. Ideally, you want to store backups on a different physical drive in case of disk failure. From the *Activities* page, select the *Backup database* option, and then click the *Configure Settings* button.

8. On the Backup Criteria tab of the Configure Settings dialog window, choose the *Balance Database Recoverability with Performance* option. On the *Backup Location* tab, select the existing backup location and click the *Change* button. Specify a different location to perform the backup (ensure that enough room exists on the drive). On the *Backup Mode* tab, ensure that *Offline Backup* is selected. Click the *OK* button to close the *Backup Criteria* tab. Click the *Next* button.

9. The *Summary* page of the *Configure Automated Maintenance* wizard contains a summary of the choices you selected. Click the *Finish* button to accept and implement the changes.

13

Chapter 13 – Concurrency and Locking

This chapter discusses how to allow multiple users to access the same database at the same time without interfering with each other, and keeping their operations consistent. We will discuss the concepts of transactions, concurrency and locking.

Note:

For more information about concurrency and locking, watch this video:
http://www.channeldb2.com/video/video/show?id=807741:Video:4322

13.1 Transactions

A transaction or unit of work consists of one or more SQL statements which, when executed, should be considered as a single unit; that is, if one of the statements in the transaction fails, the entire transaction fails, and any statements executed up to the point of failure are rolled back. A transaction ends with a COMMIT statement, which also signifies the start of a new transaction. *Figure 13.1* provides an example of a transaction.
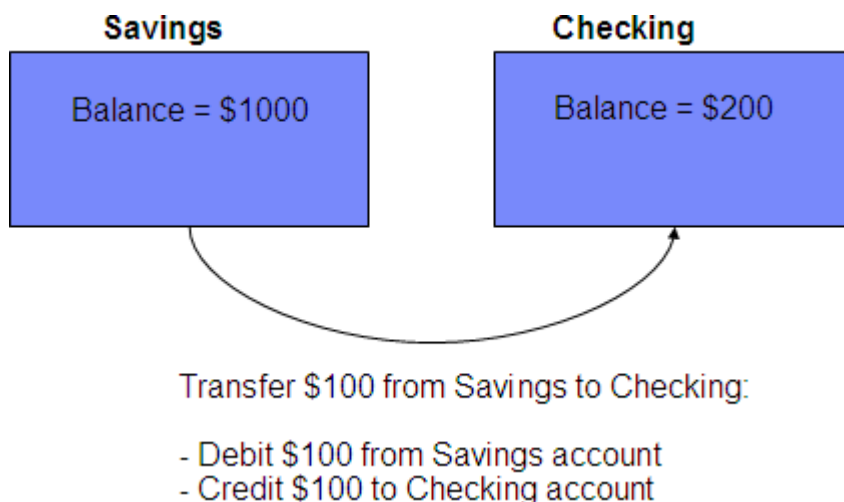
Figure 13.1 –An example of a transaction

In *Figure 13.1*, for example, you want to transfer 100 dollars from your savings account to your checking account. As shown in the figure, the following sequence of events may be required to achieve this task:

- Debit $100 from the savings account

- Credit $100 to the checking account

If the above sequence of events is not treated as a single unit of work, a transaction, imagine what would happen if a power failure occurred after the debit from the savings account, but before the checking account is credited. You would lose $100!

13.2 Concurrency

Concurrency implies that several users can work at the same time on the same database objects. DB2 was designed as a multi-user database. Access to data must be coordinated properly and transparently using a mechanism to ensure data integrity and consistency. Consider *Figure 13.2* as an example.
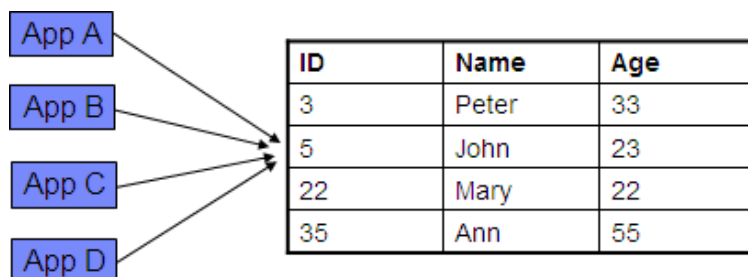
Figure 13.2 –An example of concurrency, and the need for concurrency control

In *Figure 13.2*, there are four applications, App A, App B, App C, and App D that are trying to access the same row (row 2) in a table. Without any concurrency control, all of the applications could perform operations against the same row. Assuming all of the applications are updating the Age column for row 2 with different values, the application which performs the update the last will likely be the "winner" in this situation. It should be obvious in this example that some sort of concurrency control is required to guarantee consistent results. This concurrency control is based on using locks.

Locking and concurrency concepts go hand in hand. Locking temporarily stops other applications from performing their operation until another operation finishes. The more locking there is in a system, the less concurrency is possible. On the other hand, the less locking there is in a system, the more concurrency is possible.

Locks are acquired automatically as needed to support a transaction and are released when the transaction terminates (using either a COMMIT or ROLLBACK command). Locks can be acquired on tables or rows. There are two basic types of locks:

- Share locks (S locks) – acquired when an application wants to read and prevent others from updating the same row

- Exclusive locks (X locks) – acquired when an application updates, inserts, or deletes a row

Now consider *Figure 13.3*, which is similar to *Figure 13.2*, but it now shows a lock.
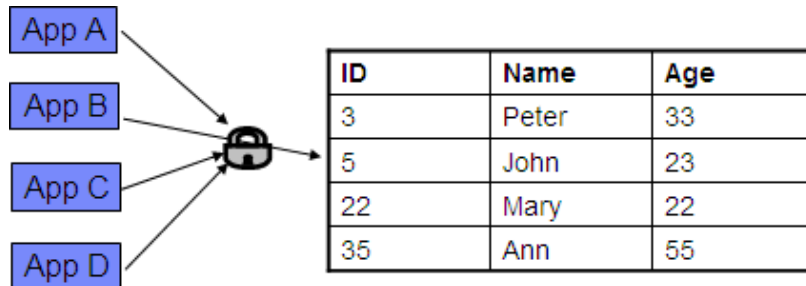
Figure 13.3 –An example of concurrency, and the need for locks

For example, in *Figure 13.2*, if App B is the first one accessing row 2, and is performing an UPDATE, App B holds an X lock on the row. When App A, App C and App D try to access the same row, they won't be able to UPDATE it because of the X lock. This control allows for consistency and integrity of the data.

13.3 Problems without concurrency control

Without some form of concurrency control, the following problems may be encountered

- Lost update

- Uncommitted read

- Non-repeatable read

- Phantom read

13.3.1 Lost update

Lost update is a problem similar to the one explained earlier in this section where the application performing the last update, will be the "winner".
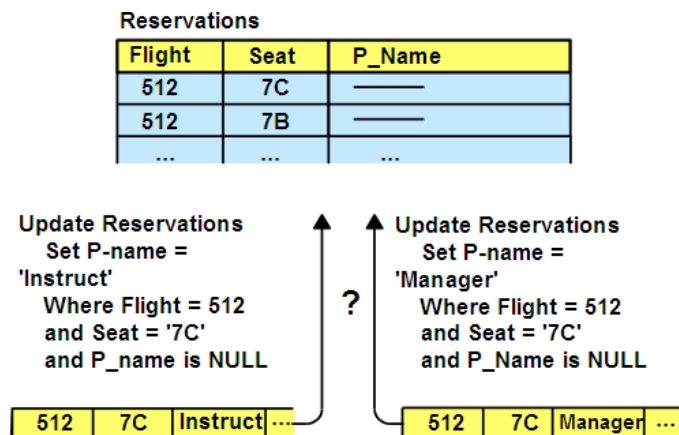
Figure 13.4 – Lost Update

In *Figure 13.4* there are two applications attempting to update the same row. The one on the left is application App1, and the one on the right is application App2. The sequence of events is then:

> 1. App1 updates a row
>
> 2. App2 updates the same row
>
> 3. App1 commits
>
> 4. App2 commits

App1's update is lost when App2 make its update, hence the term "Lost Update".

13.3.2 Uncommitted read

An uncommitted read, or "dirty read" allows for an application to read information that has not been committed, and therefore is not necessarily accurate.

Figure 13.5 – Uncommitted Read

Figure 13.5 follows this sequence of events:

> 1. App1 updates a row
>
> 2. App2 reads the new value from that row
>
> 3. App1 rolls back its changes to that row

App2 is reading uncommitted data, and hence invalid data, which is why this problem is called an "uncommitted read"

13.3.3 Non-repeatable read

A non-repeatable read implies that you cannot obtain the same result after performing the same read in the same operation.

FLIGHT	SEAT	NAME	DESTINATION	ORIGIN
512	7B	——	DENVER	DALLAS
....				
....				
814	8A	——	SAN JOSE	DENVER
....				
134	1C	——	HONOLULU	SAN JOSE
....				....

Figure 13.6 – Non-repeatable Read

In *Figure 13.6*, consider if you are trying to book a flight from Dallas to Honolulu. The sequence of events is:

1. App1 opens a cursor (also known as a result set) obtaining what you see in *Figure 13.6*

2. App2 deletes a row that qualified for the cursor (for example, the row with destination "San Jose")

3. App2 commits changes

4. App1 closes and reopens the cursor

In this case, since App1 would not get the same data on a repeated read, it cannot reproduce the data set; that's why this problem is called "non-repeatable read".

13.3.4 Phantom read

The phantom read problem is similar to the non-repeatable read problem, but the difference is that on subsequent fetches, you may obtain additional rows rather than fewer rows. *Figure 13.7* provides an example of this problem.

Reservations

Flight	Seat	P_Name
512	7B	———
512	7A	P Read
...	...	

① Select seat
   From Reservations
   Where P-name is NULL

② Update Reservations
   Set P-name = 'NULL'
   Where Flight = 512
   and Seat = '7A'
   and P_Name = 'P Read'

| 512 | 7B | ——— |

③ Repeat 1 now 7C is
  now available
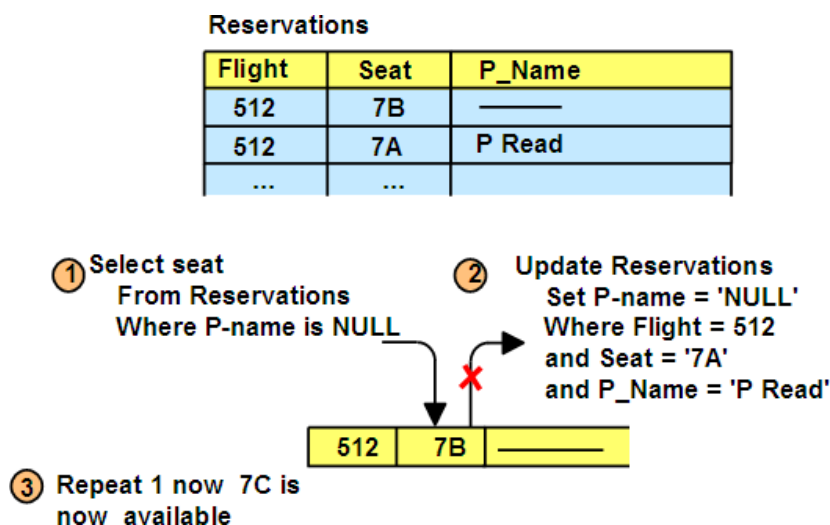
Figure 13.7 – Phantom read

Figure 13.7 shows the following sequence of events:

1. App1 opens a cursor

2. App2 adds a row to the database that would qualify for the cursor

3. App2 commits changes

4. App1 closes and reopens cursor

In this case, App1 would not get the same data on a repeated read, it would get more rows, that's why this problem is called "phantom read".

13.4 Isolation Levels

You can think of isolation levels as locking policies where, depending on the isolation level chosen, you may get different behaviors for database locking with an application.

DB2 provides different levels of protection to isolate data:

- Uncommitted Read (UR)
- Cursor Stability (CS)
- Read Stability (RS)
- Repeatable Read (RR)

13.4.1 Uncommitted read

Uncommitted read is also known as dirty read. It is the lowest level of isolation, and provides the highest degree of concurrency. No row locks are obtained on read operations, unless another application attempts to drop or alter a table; and update operations act as if using the cursor stability isolation level.

Problems still possible with this isolation level:

- Uncommitted read

- Non-repeatable read

- Phantom read

Problems prevented with this isolation level:

- Loss of update

13.4.2 Cursor stability

Cursor stability is the default isolation level. It provides a minimal degree of locking. Basically, with this isolation level the "current" row of a cursor is locked. If the row is only read, the lock is held until a new row is fetched or the unit of work is terminated. If the row is updated, the lock is held until the unit of work is terminated.

Problems still possible with this isolation level:

- Non-repeatable read

- Phantom read

Problems prevented with this isolation level:

- Loss of update

- Uncommitted read

13.4.2.1 Currently committed

Prior to DB2 9.7 when using cursor stability isolation level, a writer (UPDATE operation) would prevent a reader (SELECT operation) from accessing the same row. The logic was that since the writer is making changes to the row, the reader should wait until the update is finished to see the final committed value. In DB2 9.7, there is a new default behavior for the cursor stability isolation level for new databases. This new behavior is implemented using **currently committed (CC)** semantics. With CC, a writer will not prevent a reader from accessing the same row. This behaviour was possible in the past if you used an isolation level of uncommitted read (UR); however, the difference now is that with UR the reader retrieves the **uncommitted** value, while with CC, the reader retrieves the **currently committed** value. The currently committed value is the committed value prior to the start of the write operation.

For example, you have a table T1 with the following contents:

```
FIRSTNAME    LASTNAME
Raul         Chong
Jin          Xie
```

Now your application AppA issues this statement, but does not commit:

```
update T1 set lastname = 'Smith' where firstname = 'Raul'
```

Next, application AppB issues this statement:

```
select lastname from T1 where firstname = 'Raul' with CS
```

Prior to DB2 9.7, this statement would hang because it is waiting for the exclusive lock held by the update statement of AppA (the writer) to be released.

With DB2 9.7 and currently committed enabled (the default for new databases), the statement would return the currently committed value which is **Chong.**

Note that even though CS is the default we are including 'with CS' in the statement for clarity. We discuss this clause later on in the chapter.

If AppB tries this statement:

```
select lastname from T1 where firstname = 'Raul' with UR
```

Since UR isolation is used, the result would be **Smith** which is the uncommitted value.

This example shows that with CC, there is better concurrency for applications, allowing readers to access the row a writer is updating.

Another scenario that would have caused contention prior to DB2 9.7 is a reader preventing a writer to access a row. This scenario was one of the reasons why a COMMIT was recommended even for read operations, as it would ensure the share (S) locks would be released. With CC this is no longer an issue, as a reader will not block a writer.

With respect to INSERT operations that are not committed, the read operation will skip them by default; that is, the result set will not display these rows. For DELETE commands, the read operation should also skip (ignore) the affected rows, but the behavior depends on the value of the DB2 registry variable DB2_SKIPDELETED. Other registry variables and properties in the BIND and PREPARE commands can change the default behavior of CC.

Just remember: Currently committed means it will only show currently committed information, therefore a non-committed INSERT or DELETE operations will be ignored.

As mentioned earlier, CC is enabled by default on new databases. If you would like to turn it off, or enable it for a database created prior to DB2 9.7 and upgraded to DB2 9.7, you can update the database configuration value to CUR_COMMIT. For example, to turn it off for the **SAMPLE** database issue:

```
db2 update db cfg for sample using CUR_COMMIT off
db2stop
db2start
```

13.4.3 Read stability

With read stability, all the rows an application retrieves within a unit of work are locked. For a given cursor, it locks all rows that qualify for the result set. For example, if you have a table containing 10,000 rows and the query returns 10 rows, then only those 10 rows are locked. Read stability uses a moderate degree of locking.

Problems still possible with this isolation level:

- Phantom read

Problems prevented with this isolation level:

- Loss of update

- Uncommitted read

- Non-repeatable read

13.4.4 Repeatable read

Repeatable read is the highest isolation level. It provides the highest degree of locking, and the least concurrency. Locks are held on all rows processed to build the result set; that is, rows not necessarily in the final result set may be locked. No other application can update, delete, or insert a row that would affect the result set until the unit of work completes. Repeatable read guarantees that the same query issued by an application more than once in a unit of work will give the same result each time.

Problems still possible with this isolation level:

- none

Problems prevented with this isolation level:

- Loss of update

- Uncommitted read

- Non-repeatable read

- Phantom read

13.4.5 Comparing isolation levels

Figure 13.8 compares the different isolation levels for a fetch. In the figure, we see that isolation level uncommitted read (UR) takes no locks. Isolation level cursor stability (CS) takes a lock for row 1 when it is fetching it, but releases it as soon as it fetches row 2, and so on. For isolation levels read stability (RS) or repeatable read (RR), any row that is fetched will be locked, and the lock is not released until the end of the transaction (at a commit point).
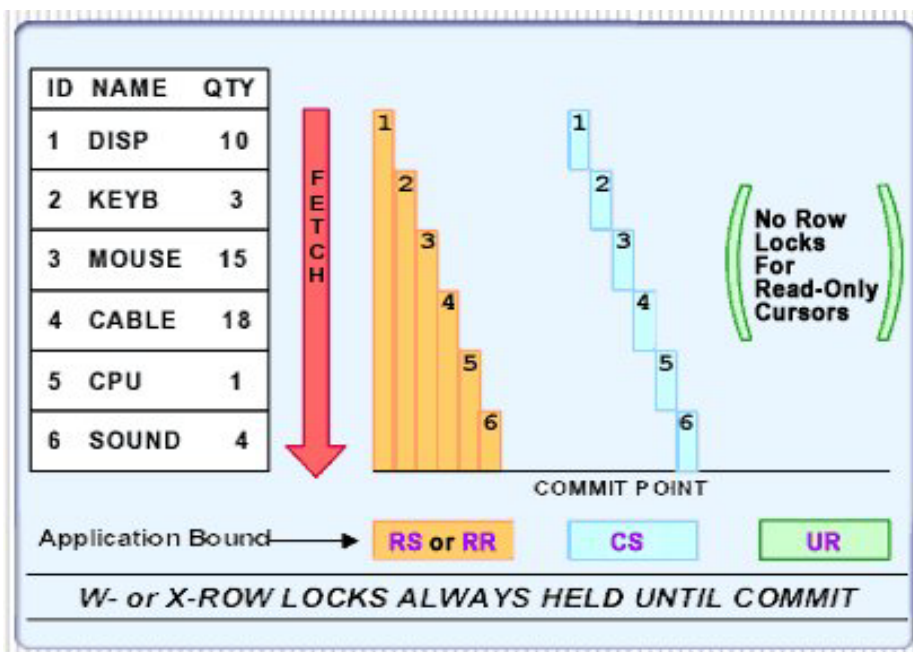
Figure 13.8 – Comparing isolation levels for a fetch

13.4.6 Setting the isolation level

Isolation levels can be specified at many levels:

- Session (application)
- Connection
- Statement

The isolation level is normally defined at the session or *Application* Level. If no isolation level is specified in your application, it defaults to cursor stability. For example, *Table 13.1* shows the possible isolation levels for a .NET or JDBC program and how these properties, when set, match a DB2 isolation level.

DB2	.NET	JDBC
Uncommitted Read (UR)	ReadUncommitted	TRANSACTION_READ_UNCOMMITTED
Cursor Stability (CS)	ReadCommitted	TRANSACTION_READ_COMMITTED
Read Stability (RS)	RepeatableRead	TRANSACTION_REPEATABLE_READ
Repeatable Read (RR)	Serializable	TRANSACTION_SERIALIZABLE

Table 13.1 - Comparison of isolation level terminology

Statement isolation level can be set using the WITH {isolation level} clause. For example:

```
SELECT ... WITH {UR | CS | RS | RR}
```

Example scenario:

An application needs to get a "rough" count of how many rows are in a table. Performance is of utmost importance. Cursor stability isolation level is required with the exception of one SQL statement:

```
SELECT COUNT(*) FROM tab1 WITH UR
```

For embedded SQL, the level is set at bind time, for dynamic SQL, the level is set at run time.

Choosing which isolation level to use depends on your application. If your application does not need exact counts as in the above example, choose UR isolation. If your application requires very tight control on the data it works with, choose RR isolation.

To use currently committed semantics at bind time or prepare time, use this syntax:

BIND:

```
>--+-----------------------------------------------------------+-->
   '--CONCURRENTACCESSRESOLUTION--+--USE CURRENTLY COMMITTED--+--'
                                  '--WAIT FOR OUTCOME---------'
```

PREPARE:

```
concurrent-access-resolution:
|-+-USE CURRENTLY COMMITTED-+-------------------------|
  '-WAIT FOR OUTCOME--------'
```

On a JDBC application using the IBM Data Server Driver for JDBC and SQLJ, you can use the concurrentAccessResolution property to enable currently committed.

13.5 Lock escalation

Every lock made by DB2 consumes some memory. When the optimizer thinks it is better to have one lock on the entire table, rather than multiple row locks, lock escalation occurs. *Figure 13.9* illustrates this.
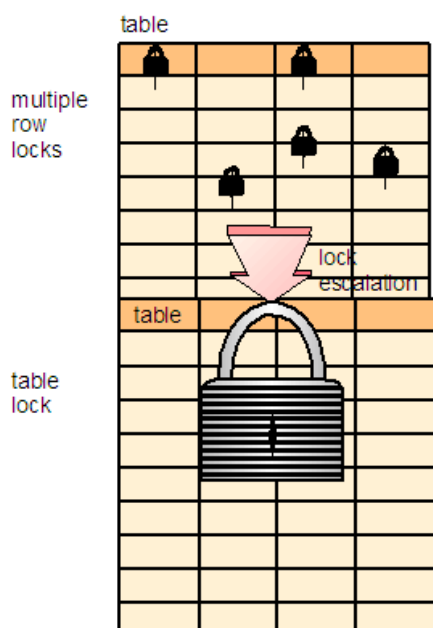
Figure 13.9 – Lock escalation

There are two main database configuration parameters related to lock escalation:

- **LOCKLIST** – The amount of memory (in 4k pages) reserved to manage locks for all connected applications.
- **MAXLOCKS** – Maximum percentage of the entire lock list that a single application can use up. .

The default for both parameters is AUTOMATIC, meaning that the self-tuning memory manager (STMM) will modify the size. If you don't enable STMM, and set the value your self, these values will have an impact on when lock escalation occurs. For example, if you set LOCKLIST to 200K and MAXLOCKS to 22%, then lock escalation occurs when a single application requires more than 44K of lock memory (200 K * 22% = 44K). If lock escalation occurs frequently with these settings, increase the value of LOCKLIST and MAXLOCKS. Lock escalation is not good for performance as it reduces concurrency. The DB2 diagnostic log file (`db2diag.log`) can be used to determine whether lock escalation is occurring. See Appendix A to learn more about this file.

13.6 Lock monitoring

You can monitor the use of locks using DB2 application lock snapshots. To turn on the snapshots for locks, issue this command:

```
UPDATE MONITOR SWITCHES USING LOCK ON
```

After the switch is turned on, monitoring information will be collected. To obtain a report of the locks at a given time, issue this command:

```
GET SNAPSHOT FOR LOCKS FOR APPLICATION AGENTID <handle>
```

Figure 13.9 shows the output for a sample application lock snapshot.

```
              Application Lock Snapshot


Snapshot  timestamp                                 = 11-05-2002
00:09:08.672586


Application handle                       = 9
Application ID                           = *LOCAL.DB2.00B9C5050843
Sequence number                          = 0001
Application name                         = db2bp.exe
Authorization ID                         = ADMINISTRATOR
Application status                       = UOW Waiting
Status change time                       = Not Collected
Application code page                    = 1252
Locks held                               = 4
Total wait time (ms)                     = 0


List Of Locks
 Lock Name                  = 0x05000700048001000000000052
 Lock Attributes            = 0x00000000
 Release Flags              = 0x40000000
 Lock Count                 = 255
 Hold Count                 = 0
 Lock Object Name           = 98308
 Object Type                = Row
 Tablespace Name            = TEST4K
 Table Schema               = ADMINISTRATOR
```

Figure 13.9 – Application Lock Snapshot

new in V9.7

> **Note**:
>
> In DB2 9.7, an effort is being made to move database monitoring away from the system monitor and snapshot technology towards having SQL access to internal memory such as workload management table functions and IBM Data Studio tools. See the official DB2 documentation for more information.

13.7 Lock wait

When two or more applications need to perform an operation on the same object, one of them may have to wait to obtain the needed lock. By default, an application will wait indefinitely. The time an application waits for a lock is controlled by the database configuration parameter LOCKTIMEOUT. The default value of this parameter is -1 (infinite wait).

The CURRENT LOCK TIMEOUT register can be used to set the lock wait for a given connection. By default, this register is set to the value of LOCKTIMEOUT. Use the SET LOCK TIMEOUT statement to change its value. Once the value of this register is set for a connection, it will persist across transactions.

Example:

```
SET LOCK TIMEOUT=WAIT n
```

13.8 Deadlock causes and detection

A deadlock occurs when two or more applications connected to the same database wait indefinitely for a resource. The waiting is never resolved because each application is holding a resource that the other needs. Deadlocks are an application design issue most of the time. *Figure 13.10* illustrates a deadlock.

within a UNIT OF WORK (UOW)   INSERT CEREAL AND MILK into BOWL
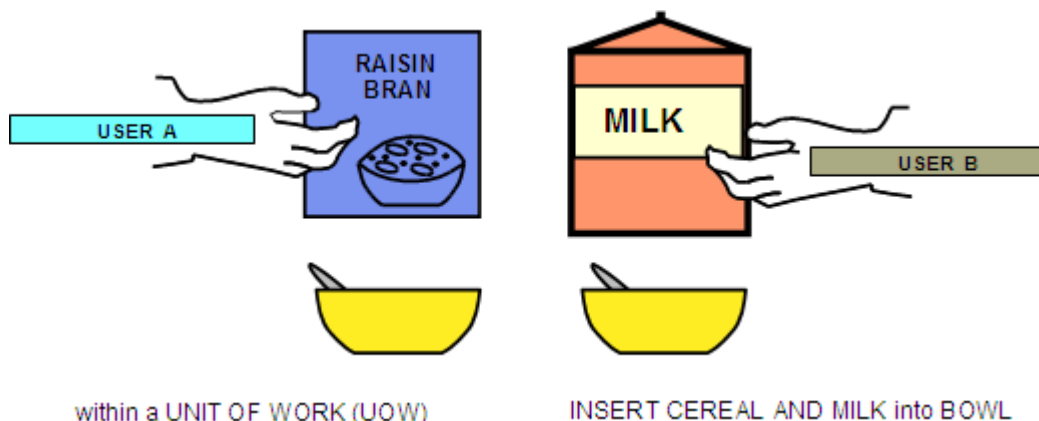
Figure 13.10 – Deadlock scenario

In *Figure 13.10*, user A is holding the cereal and will not let go until he gets the milk. On the other hand, user B is holding the milk, and will not let go until he gets the cereal. Therefore, we have a deadlock situation.

With DB2 9.7, the use of currently committed has considerably reduced the occurrence of deadlocks, as one application does not need to wait for the other application to release its lock, but instead accesses the currently committed value.

To simulate a deadlock situation in DB2, follow these steps:

1.   Turn off currently committed:

```
db2 update db cfg for sample using cur_commit off
db2stop force
db2start
```

2.   Open two DB2 Command Windows (which we will call "CLP1" and "CLP2". respectively) representing two different applications connecting to the database

3.   From CLP1 issue the commands:

```
db2 connect to sample
db2 +c update employee set firstnme = 'Mary' where empno = '000050'
```

First we are connecting to the **SAMPLE** database, and then issuing an update statement on the row with "empno = 000050" on the employee table. The "+c" option in the statement indicates that we do not want the DB2 Command Window to automatically commit the statement. We are doing this on purpose so we hold the locks.

4. From CLP2 issue the commands:

```
db2 connect to sample
db2 +c update employee set firstnme = 'Tom' where empno = '000030'
```

In the CLP2 window, which represents the second application, we are also connecting to the SAMPLE database, but are updating another row in the employee table.

5. From CLP1 issue:

```
db2 +c select firstnme from employee where empno = '000030'
```

After pressing Enter to execute the above SELECT statement, the SELECT may seem to hang. It actually is not hanging, but waiting for the release of the exclusive lock that was taken by CLP2 on this row in step 3. At this point, if LOCKTIMEOUT has been left with its default value of -1, the CLP1 application would wait forever.

6. From CLP2 issue:

```
db2 +c select firstnme from employee where empno = '000050'
```

By issuing the above SELECT statement, we are now creating a deadlock. This SELECT statement will also seem to hang, as it is waiting for the release of the exclusive lock that was taken by CLP1 on this row in step 2.

In the above deadlock scenario, DB2 will check for the database configuration parameter DLCHKTIME. This parameter sets the time interval for checking for deadlocks. For example, if this parameter is set to 10 seconds, DB2 will check every 10 seconds if a deadlock has occurred. If indeed a deadlock happened, DB2 will use an internal algorithm to determine which of the two transactions should be rolled back, and which one should continue..

If you are experiencing numerous deadlocks, you should re-examine your existing transactions and see if any re-structuring is possible.

13.9 Concurrency and locking best practices

The following are some tips to follow in order to allow for the best possible concurrency:

1. Ensure you enable currently committed (CC), if your application logic allows it

2. Keep transactions as short as possible. This can be achieved by issuing frequent COMMIT statements (even for read-only transactions) when your application logic allows it.

3. Log transaction information only when required.

4. Purge data quickly. You can execute this command

```
ALTER TABLE ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE
```

new in
V9.7

or now with DB2 9.7, use the TRUNCATE command:

```
TRUNCATE <table name>
```

5. Perform data modifications in batches/groups. For example:

```
DELETE FROM (
    SELECT * FROM tedwas.t1 WHERE c1 = … FETCH FIRST 3000 ROWS ONLY)
```

6. Use concurrency features in DB2 data movement tools.

7. Set the database level LOCKTIMEOUT parameter (suggested times are between 30-120 seconds). Don't leave it to the default of -1. You can also use session-based lock timeout.

8. Do not retrieve more data than is required. For example, use the FETCH FIRST n ROWS ONLY clause in SELECT statements.

Note:

For more information on concurrency and locking best practices, see the Best Practices documents available at http://www.ibm.com/developerworks/data/bestpractices/

13.10 Summary

In this chapter we looked at maintaining data integrity through transaction control, concurrent user access and locking levels. The different concurrency levels all have issues that can affect how you access and manage your data.

We also looked in detail at the role of setting the isolation levels to deal with these issues, and how the isolation level can be manipulated to provide the maximum flexibility required by your application and data needs.

We also looked at lock escalation, lock waits and lock monitoring, along with the causes, detection and handling of database deadlocks.

Finally, we looked at some best practice ideas for getting the best possible results for your concurrency needs.

13.11 Exercises

In this exercise you will practice with the concurrency and locking concepts discussed in this chapter using the DB2 Command Window. This tool uses a lock level of isolation CS by default. After executing an SQL statement, the Command Window will automatically issue a commit (this is also known as autocommit). For illustration purposes in this exercise, we will use the **+c** flag to turn off autocommit, and use the **WITH <isolation level>** clause after some SQL statements to override the CS default isolation.

Part 1: Testing the Phantom Read problem and isolation RR

Procedure:

1. Open two DB2 Command Windows as shown in the figure below. We will call the window at the top "DB2 Command Window #1", and the one at the bottom "DB2 Command Window #2"

2. From DB2 Command Window #1 issue:

    ```
    db2 connect to sample
    db2 +c select * from staff
    ```
    This will return 35 records

3. From DB2 Command Window #2 issue:

    ```
    db2 connect to sample
    db2 +c insert into staff (id,name) values (400, 'test')
    ```

4. From DB2 Command Window #1 issue:

    ```
    db2 +c select * from staff
    ```

    This should still return 35 records

5. From DB2 Command Window #2 issue:

    ```
    db2 commit
    ```

6. From DB2 Command Window #1 issue:

    ```
    db2 +c select * from staff
    ```

    This now returns 36 records!

    DB2 Command Window #1 represents one application which opens a cursor or result set (select * from staff) obtaining 35 records. Within the same transaction (because in this window we are not issuing any **commit** statements), the application opens the same cursor, and still sees 35 records, even after the application in DB2 Command Window #2 inserts (but did not commit) a new record.

Next, the DB2 Command Window #2 application does commit the insert, and so the third time DB2 Command Window #1 application opens the cursor, the result set returns one more row (a phantom read) obtaining 36 records. This example illustrates the phantom read problem: Within the same transaction, opening the same cursor returns more rows. We are using a CS isolation level, and as mentioned earlier in the chapter, CS does not prevent phantom reads.

7. Clean up the record that was inserted before we try the next steps:

From DB2 Command Window #1:

```
db2 rollback
```

From DB2 Command Window #2:

```
db2 delete from staff where id = 400
```

```
db2 select * from staff
```

This returns 35 records again.

8. Now let's see if an RR isolation level can prevent this phantom read problem.

From DB2 Command Window #1 issue:

```
db2 connect to sample
```

```
db2 +c select * from staff with RR
```
This returns 35 records

From DB2 Command Window #2:

```
db2 connect to sample
db2 +c insert into staff (id,name) values (400, 'test')
```

This statement will hang which is expected.

Since the clause WITH RR is added on the SELECT statement for DB2 Command Window #1, this isolation level prevents an INSERT on a row that would alter the result set. This example illustrates that the isolation level of RR does prevent phantom read problems.

9. Clean up before proceeding to part 2 of this exercise:

From DB2 Command Window #2:

```
Ctrl-C (to interrupt)
```
Close the window

From DB2 Command Window #1:

```
db2 rollback
```
Close the window

Part 2: Testing currently committed (CC) and uncommitted read (UR) levels

Procedure 1: Analyze the behavior of isolation CS without currently committed

1. Open a DB2 Command Window and issue the following statements:

    ```
    db2 connect to sample
    db2 select * from staff
    ```

    Inspect the contents of table *STAFF*, specifically look for *ID* value of '10'. The corresponding *NAME* column has a value of *Sanders*. Close the window.

2. Currently committed is the default for new databases. Verify it is in fact turned on for database sample:

    ```
    db2 get db cfg for sample
    ```

    Look for the line close to the end of the output that says:
    ```
    Currently Committed                        (CUR_COMMIT) = ON
    ```

    If the value is ON, turn it OFF first so we can analyze the behavior of isolation CS as it would be prior to DB2 9.7:
    ```
    db2 update db cfg for sample using CUR_COMMIT off
    db2 force applications all
    ```
    Adding the `force` option ensures that there are no connections so the update to CUR_COMMIT will take effect on the next connection)

    Confirm CUR_COMMIT is disabled. It should say:
    ```
    Currently Committed                        (CUR_COMMIT) = DISABLED
    ```

3. Open two DB2 Command Windows as in part 1, so one is on top of the other. Let's see how isolation CS works without currently committed, when an update (writer) and a select (reader) have an interest on the same row. Note we don't need to put "WITH CS" after the statements (as this is the default).

    From DB2 Command window #1 (this is the writer):

    ```
    db2 connect to sample
    db2 +c update staff set name = 'Chong' where id = 10
    ```

    From DB2 Command window #2 (this is the reader):

    ```
    db2 connect to sample
    db2 +c select * from staff
    ```

This SELECT hangs waiting for the exclusive (X) lock to be released by the DB2 Command Window #1 application. As you can see, the default behavior of CS prior to DB2 9.7 allows for less concurrency

From DB2 Command window #2:

```
CTRL-C (press these keys to interrupt)
```
Close the window

From DB2 Command window #1:

```
db2 rollback
```
Close the window

Procedure 2: Analyze the behavior of isolation CS with currently committed

1. Turn currently committed to ON:

    Open a DB2 Command Window and issue:
    ```
    db2 update db cfg for sample using CUR_COMMIT on
    db2 force applications all
    ```

    Close the window.

2. Open two DB2 Command Windows as in part 1 so one is on top of the other. Then issue the following:

    From DB2 Command window #1:

    ```
    db2 connect to sample
    db2 +c update staff set name = 'Chong' where id = 10
    ```

    From DB2 Command window #2:

    ```
    db2 connect to sample
    db2 +c select * from staff
    ```

    Now this SELECT statement works! .It does not hang, and the value displayed is *Sanders*, which is the currently committed value.

    From DB2 Command window #1:

    ```
    db2 rollback
    ```
    Close the window.

From DB2 Command window #2:

```
db2 rollback
```
Close the window.

Procedure 3: Analyze the behavior of isolation UR

1. Open two DB2 Command Windows as in part 1 so one is on top of the other. Then issue the following:

    From DB2 Command window #1:

    ```
    db2 connect to sample
    db2 +c update staff set name = 'Chong' where id = 10
    ```

    From DB2 Command window #2:

    ```
    db2 connect to sample
    db2 +c select * from staff with UR
    ```

    This SELECT works, but note that the value displayed is *Chong* which is the uncommitted value.

    From DB2 Command window #1:

    ```
    db2 rollback
    ```
    Close the window.

    From DB2 Command window #2:

    ```
    db2 rollback
    ```
    Close the window.

PART III – LEARNING DB2: APPLICATION DEVELOPMENT

14

Chapter 14 – Introduction to DB2 Application Development

IBM DB2 is powerful data server software for managing both relational and XML data. It offers flexibility not only to database administrators, but also to database developers. No matter which language you use to develop your programs, DB2 provides the drivers, adapters, and extensions you need to work with databases as part of your application. Moreover with DB2 Express-C, you can develop your applications at no cost, with no database size limits, and with the same level of programming language support as the other versions of DB2. Develop once using DB2 Express-C, and you can run on any DB2 edition without any modification required to your application.

> **Note**:
>
> This section is only an introduction to DB2 application development. A series of more than 25 free online books is being developed as part of the ***DB2 on Campus Book Series*** including a book specific to DB2 application development. Other books in the series include non-DB2 topics such as Java, PHP, Ruby on Rails, Python, Perl, Web 2.0, SOA, Eclipse, Open Source development, Cloud Computing and much more!. Other books cover IBM technologies such as pureQuery, Data Studio, InfoSphere Data Architect in more detail. These books will be introduced starting on October of 2009.

14.1 DB2 Application Development: The big picture

DB2 offers database developers the flexibility to take advantage of server-side development features such as stored procedures and user-defined functions, while, application developers can develop client applications using the programming language of their choice. This flexibility is illustrated in *Figure 14.1*.

Figure 14.1 - DB2 is for everyone: Database and Application developers

In Figure 14.1 the left side represents a client machine where an application programmer develops and runs his program. In this client machine, in addition to the operating system, an IBM Data Server Client may be installed depending on the type of application being developed. An IBM Data Server client includes the required connection drivers such as the JDBC drivers and the ODBC/CLI drivers. These drivers can also be downloaded independently by visiting the IBM DB2 Express-C Web site at ibm.com/db2/express

Using programming tools such as IBM Data Studio, InfoSphere Data Architect (IDA), Rational Software Architect (RSA), Rational Application Developer (RAD), and so on, you can develop your application in your desired programming language. The API libraries supporting these languages are also included with the IBM Data Server Client, so that when you connect to a DB2 Server, all the program instructions are translated appropriately using these APIs into the SQL or XQuery statements understood by DB2. Table 1.1 provides a short description of the tools mentioned earlier.

Tool name	Description
IBM Data Studio	IBM Data Studio is an Eclipse-based tool that allows users to manage their data servers and develop stored procedures, functions and Data Web services. IBM Data Studio was covered earlier in the book.
InfoSphere Data Architect (IDA)	IDA is a modeling tool for your data. It helps you build your database logical design and physical design.
Rational Software Architect (RSA)	RSA is an Eclipse-based tool for software engineering to help you develop UML

	diagrams
Rational Application Developer (RAD)	RAD is an Eclipse-based rapid application development tool for software developers
Visual Studio	Microsoft Visual Studio is an IDE that allows you to develop applications in the Windows platform using Microsoft's technology.
ZendCore	Formerly called *ZendCore for IBM*, this is a free IDE for developing PHP applications.

Table 14.1 - Tools that can help you develop applications with DB2

On the right side of *Figure 14.1* a DB2 server is illustrated containing one database. Within this database there are stored procedures, user-defined functions and triggers. We describe all of these objects in more detail in the next sections.

14.2 Server-side development

Server-side development in DB2 implies that application objects are developed and stored on the DB2 database. The following application objects will be discussed briefly in this section:

- Stored Procedures
- User-defined Functions (UDFs)
- Triggers

14.2.1 Stored Procedures

A **stored procedure** is a database application object that can encapsulate SQL statements and business logic. Keeping part of the application logic in the database provides performance improvements as the amount of network traffic between the application and the database is reduced. In addition, stored procedures provide a centralized location to store your code, so other applications can reuse the same stored procedures. To call a stored procedure, use the **CALL** statement. In DB2 you can develop stored procedures in several languages including SQL PL, Java, C/C++, CLR, OLE, and COBOL. A simple example of how to create and call a SQL PL stored procedure from the DB2 Command Window or Linux shell is shown below:

```
db2 create procedure P1 begin end
db2 call P1
```

In the example, procedure P1 is an empty stored procedure which is not doing anything. The example illustrates how easy you can create a stored procedure. To develop stored

procedures with more complex logic, we recommend you use IBM Data Studio which includes a debugger.

14.2.2 User-defined functions

A *user-defined function (UDF)* is a database application object that allows users to extend the SQL language with their own logic. A function always returns a value or values normally as a result of the business logic included in the function. To invoke a function, use it within a SQL statement, or with the **values** function. In DB2 you can develop UDFs in several languages including SQL PL, Java, C/C++, OLE, CLR.

This simple example shows how to create and call a SQL PL UDF from the DB2 Command Window or Linux shell:

```
db2 create function F1() returns integer begin return 1000; end
db2 values F1
```

In the example, function F1 is a function returning an integer value of 1000. The **VALUES** statement can be used to invoke the function. Like in the case of stored procedures, we recommend you create functions using IBM Data Studio.

14.2.3 Triggers

A *trigger* is an object that automatically performs an operation on a table or view. A triggering action on the object where the trigger is defined causes the trigger to be fired. A trigger is normally not considered an application object; therefore, database developers normally don't code triggers, but database administrators do. Because some coding is required, we have included triggers in this section. Below is an example of a trigger:

```
create trigger myvalidate no cascade before insert on T1
      referencing NEW as N
      for each row
      begin atomic
         set (N.myxmlcol) = XMLVALIDATE(N.myxmlcol
            according to xmlschema id myxmlschema);
      end
```

In this example, the trigger is fired before an INSERT operation on table T1. The trigger will insert the value (which is an XML document), but will invoke the XMLVALIDATE function to validate this XML document with a given schema. *Chapter 15, DB2 pureXML* talks more about XML and XML schemas.

14.3 Client-side development

As the name suggests, in client-side development, the application developers code their programs on a client and then connect and access the DB2 database using the application program interfaces (APIs) that are provided with DB2. In this section we discuss:

- Embedded SQL

- Static SQL vs Dynamic SQL

- CLI and ODBC

- JDBC, SQLJ and pureQuery

- OLE DB

- ADO.NET

- PHP

- Ruby on Rails

- Perl

- Python

14.3.1 Embedded SQL

Embedded SQL applications are applications where SQL is embedded into a host language such as C, C++, or COBOL. The embedded SQL application can include static or dynamic SQL as described in the next section. *Figure 14.2* shows how an embedded SQL application is built.

Figure 14.2 - Building embedded SQL applications

In the figure, the C program `hello.sqc` contains embedded SQL. The embedded SQL API for the C language uses EXEC SQL (highlighted in *Figure 14.2*) to allow a precompilation process to distinguish between the embedded SQL statements and the actual C code. You may also note in the `hello.sqc` listing that some variables are prefixed with a colon, as in ***:dbname, :userID,*** and ***:psw***. These are called host variables. Host variables are variables from the host language that are referenced in the embedded SQL statements.

Issuing the **precompile** command (also known as the **prep** command) with the **bindfile** option generates two files, the `hello.bnd` bind file containing only SQL statements and the `hello.c` file containing only C code. The bind file will be compiled using the **bind** command to obtain a *package* that is stored in the database. A package includes the compiled/executable SQL and the access path DB2 will follow to retrieve the data. To issue the **bind** command, a connection to the database must exist. At the bottom of the figure, the `hello.c` file is compiled and linked like any regular C program. The resulting executable file `hello.exe` has to match the package stored in the database to successfully execute.

14.3.2 Static SQL vs. Dynamic SQL

Static SQL statements are the ones where the SQL structure is fully known at precompile time. For example:

```
SELECT lastname, salary FROM employee
```

In this example, the names for the columns (**lastname, salary**) and table (**employee**) referenced in a statement are fully known at precompile time. The following example is also a static SQL statement:

```
SELECT lastname, salary FROM employee WHERE firstnme = :fname
```

In this second example, a host variable ***:fname*** is used as part of an embedded SQL statement. Though the value of the host variable is unknown until runtime, its data type is known from the program, and all the other objects (column names, table names) are fully known ahead of time. DB2 uses estimates for these host variables to calculate the access plan ahead of time; therefore, this case is still considered static SQL.

You precompile, bind, and compile statically executed SQL statements before you run your application. Static SQL is best used on databases whose statistics do not change a great deal. Now let's take a look at one more example:

```
SELECT ?, ? FROM ?
```

In this example, the names for the columns and table referenced by the statement are not known until runtime. Therefore the access plan is calculated only at runtime and using the statistics available at the time. These types of statements are considered ***Dynamic SQL*** statements.

Some programming APIs, like JDBC and ODBC, always use dynamic SQL regardless of whether the SQL statement includes known objects or not. For example, the statement **SELECT lastname, salary FROM employee** has all the columns and table names known ahead of time, but through JDBC or ODBC, you do not precompile the statements. All the access plans for the statements are calculated at runtime.

In general, two statements are used to treat a SQL statement as dynamic:

- **PREPARE**: This statement prepares or compiles the SQL statement calculating the access plan to use to retrieve the data

- **EXECUTE**: This statement executes the SQL

Alternatively you can execute a **PREPARE** and **EXECUTE** in one single statement: **EXECUTE IMMEDIATELY**

Listing 14.1 shows an example on an embedded C dynamic SQL statement that is prepared and executed.

```
strcpy(hVStmtDyn, "SELECT name FROM emp WHERE dept = ?");
PREPARE StmtDyn FROM :hVStmtDyn;
EXECUTE StmtDyn USING 1;
EXECUTE StmtDyn USING 2;
```

Listing 14.1 - An embedded C dynamic SQL statement using PREPARE and EXECUTE

Listing 14.2 shows the same example as *Listing 14.1*, but using the EXECUTE IMMEDIATELY statement

```
EXECUTE IMMEDIATELY SELECT name from EMP where dept = 1
EXECUTE IMMEDIATELY SELECT name from EMP where dept = 2
```

Listing 14.2 - An embedded C dynamic SQL statement using EXECUTE IMMEDIATELY

In many dynamic programming languages such as PHP or Ruby on Rails, where SQL is run dynamically, programmers tend to write the same SQL statements with different field values as follows:

SELECT lastname, salary FROM employee where firstnme = 'Raul'

SELECT lastname, salary FROM employee where firstnme = 'Jin'

...

In this example, the statements are identical except for the value of the column **firstnme.** DB2 considers these two dynamic SQL statements as different ones, and therefore at runtime, it prepares and then executes each statement independently. The overhead of preparing the same statement several times can cause performance degradation, therefore prior to DB2 9.7, the recommendation was to code statements as follows:

SELECT lastname, salary FROM employee where firstnme = ?

The question mark (?) in the statement is known as a ***parameter marker***. Using parameter markers, the program could prepare the statement only once, and then issue `EXECUTE` statements providing the different values for the parameter marker.

In DB2 9.7, DB2 introduced a technology called **statement concentrator** where all the statements that are the same except for the field values are automatically lumped together into one single statement with parameter markers, and then EXECUTE statements are performed with the different values. The statement concentrator does have the intelligence to determine when not to lump some statements together; for example, when you purposely add some clauses to influence the DB2 optimizer.

With respect to performance, Static SQL will normally perform better than dynamic SQL since the access plan in static SQL is performed at precompile time and not at runtime. However, for environments where there is a lot of activity such as INSERTs and DELETEs, the statistics calculated at precompile time may not be up-to-date, and therefore, the access plan of the static SQL may not be optimal. In this case, dynamic SQL may be a better choice, assuming a RUNSTATS command is frequently executed to collect current statistics.

> **Note**:
>
> Many users think embedded SQL is only static. In reality, it can be both, static or dynamic.

14.3.3 CLI and ODBC

Call Level Interface (CLI) was originally developed by the X/Open Company and the SQL Access Group. It was a specification for a callable SQL interface with the purpose of developing portable C/C++ applications regardless of the RDBMS vendor. Based on a preliminary draft of X/Open Call Level Interface, Microsoft developed ***Open Database Connectivity (ODBC)***, and later on, the ISO CLI International Standard accepted most of the X/Open Call Level Interface specification. DB2 CLI is based on both: ODBC and the International Standard for SQL/CLI as shown in *Figure 14.3*.

Figure 14.3 - DB2 CLI is based on ODBC and ISO CLI International standard

DB2 CLI conforms to ODBC 3.51 and can be used as the ODBC Driver when loaded by an ODBC Driver Manager. Figure 14.4 can help you picture DB2 CLI support for ODBC.

Figure 14.4 - DB2 CLI conforms to ODBC 3.51

CLI/ODBC has the following characteristics:

- The code is easily portable between several RDBMS vendors

- Unlike embedded SQL, there is no need for a precompiler or host variables

- It runs dynamic SQL

- It is very popular

To _**run**_ a CLI/ODBC application all you need is the DB2 CLI driver. This driver is installed from either of the following clients and drivers which can be downloaded and used for free from www.ibm.com/db2/express:

- IBM Data Server Client

- IBM Data Server Runtime Client
- IBM Data Server Driver for ODBC and CLI

To **_develop_** a CLI/ODBC application you need the DB2 CLI driver and also the appropriate libraries. These can be found only on the IBM Data Server Client.

Let's take a look at the following example so you understand better how you can set up the DB2 CLI driver for your applications. *Figure 14.5* depicts three different machines, one in Indonesia, the other one in Brazil, and the other one in Canada.

Figure 14.5 - DB2 CLI/ODBC sample scenario

The figure shows two cases:

On the left let's say the machine in Indonesia is running an *ODBC application* which could work with any RDBMS such as Oracle, Microsoft SQL Server or DB2. An ODBC Driver Manager will load the appropriate ODBC driver depending on the database that is being accessed. In the case where the application accesses a DB2 database in Canada, the connection needs to go through a DB2 Client which has the components to connect remotely.

On the right side, let's say a *CLI application* is running in a machine in Brazil. It's a CLI application because it may be using some specific functions not available in ODBC, and also because the application will only work for a DB2 database. The CLI application will go through the DB2 CLI Driver. The application can connect to the local DB2 database in Brazil. When it needs to connect to the remote database in Canada, it will go through a DB2 client.

One last point to be made in this section is a comparison between a CLI/ODBC application and an embedded SQL C dynamic application. *Figure 14.6* illustrates this comparison.

Figure 14.6 - CLI/ODBC application versus Embedded SQL C dynamic application

As shown in *Figure 14.6*, the only difference between CLI/ODBC vs. Embedded SQL C dynamic SQL is that for CLI/ODBC your code is portable and can access other RDBMS simply by changing the connection string, while in the embedded SQL C dynamic version, you may be coding specific elements for DB2. Of course the other difference is the way the different functions for **PREPARE**, and **EXECUTE** are invoked.

14.3.4 JDBC, SQLJ and pureQuery

Java Database Connectivity (JDBC) is a Java programming API that standardizes the means to work and access databases. In JDBC the code is easily portable between several RDBMS vendors. The only changes required to the code are normally which JDBC driver to load and the connection string. JDBC uses only dynamic SQL and it is very popular.

SQLJ is the standard for embedding SQL in Java programs. It is mainly used with static SQL, though it can inter-operate with JDBC as shown in *Figure 14.7*. Though it is normally more compact than JDBC programs and provides better performance, it has not been widely accepted. SQLJ programs must be run through a preprocessor (the SQLJ translator) before they can be compiled.

Figure 14.7 - Relationship between SQLJ and JDBC applications

In *Figure 14.7*, a DB2 client may or may not be required depending on the type of JDBC driver used as discussed later on this section.

pureQuery is an IBM Eclipse-based plug-in to manage relational data as objects. Available since 2007, pureQuery can automatically generate the code to establish an object-relational mapping (ORM) between your object oriented code and the relational database objects. You start by creating a Java project with Optim Development Studio (ODS), connect to a DB2 database, and then have ODS discover all the database objects. Through the ODS GUI you can pick a table and then choose to generate the pureQuery code which would transform any of the underlying relational table entities into a Java object. Code is generated to create the relevant SQL statements and parent Java objects that encapsulate those statements. The generated Java objects and the contained SQL statements can be further customized. With pureQuery, you can decide at runtime whether you want to run your SQL in static or dynamic mode. pureQuery supports both Java and .NET.

14.3.4.1 JDBC and SQLJ drivers

Though there are several types of JDBC drivers such as type 1, 2, 3 and 4; type 1 and 3 are not commonly used, and DB2's support of these types has been deprecated. For type 2, there are two drivers as we will describe shortly, but one of them is also deprecated.

Type 2 and type 4 are supported with DB2, as shown in *Table 14.2*. Type 2 drivers need to have a DB2 client installed, as the driver uses it to establish communication to the database. Type 4 is a pure Java client, so there is no need for a DB2 client, but the driver must be installed on the machine where the JDBC application is running.

Driver Type	Driver Name	Packaged as	Supports	Minimum level of SDK for Java required
Type 2	DB2 JDBC Type 2 Driver for Linux, UNIX and Windows **(Deprecated*)**	db2java.zip	JDBC 1.2 and JDBC 2.0	1.4.2
Type 2 and Type 4	IBM Data Server Driver for JDBC and SQLJ	db2jcc.jar and sqlj.zip	JDBC 3.0 compliant	1.4.2
		db2jcc4.jar and sqlj4.zip	JDBC 4.0 and earlier	6

Table 14.2 - DB2 JDBC and SQLJ drivers

* Deprecated means it is still supported, but no longer enhanced

As mentioned earlier and shown also in *Table 14.2*, Type 2 is provided with two different drivers; however the DB2 JDBC Type 2 Driver for Linux, UNIX and Windows, with filename db2java.zip is deprecated.

When you install a DB2 server, a DB2 client or the IBM Data Server Driver for JDBC and SQLJ, the db2jcc.jar and sqlj.zip files compliant with JDBC 3.0 are automatically added to your classpath.

14.3.5 OLE DB

Object Linking and Embedding, Database (OLE DB) is a set of interfaces that provides access to data stored in diverse sources. It was designed as a replacement to ODBC, but extended to support a wider variety of sources, including non-relational databases, such as object oriented databases and spreadsheets. OLE DB is implemented using the Component Object Model (COM) technology.

OLE DB consumers can access a DB2 database with the IBM OLE DB Provider for DB2. This provider has the following characteristics:

- Provider name: IBMDADB2

- Supports level 0 of the OLE DB provider specification, including some additional level 1 interfaces

- Complies with Version 2.7 or later of the Microsoft OLE DB specification

- An IBM Data Server Client with the Microsoft Data Access Components (MDAC) must be installed

- If IBMDADB2 is not explicitly specified, Microsoft's OLE DB driver (MSDASQL) will be utilized by default. MSDASQL allows clients utilizing OLE DB to access non-

Microsoft SQL server data sources using the ODBC driver but does not guarantee full functionality of the OLE DB driver.

14.3.6 ADO.NET

The ***.NET Framework*** is the Microsoft replacement for Component Object Model (COM) technology. Using the .NET Framework, you can code .NET applications in over forty different programming languages; the most popular ones being C# and Visual Basic .NET.

The .NET Framework class library provides the building blocks with which you build .NET applications. This class library is language agnostic and provides interfaces to operating system and application services. Your .NET application (regardless of language) compiles into Intermediate Language (IL), a type of bytecode.

The Common Language Runtime (CLR) is the heart of the .NET Framework, compiling the IL code on the fly, and then running it. In running the compiled IL code, the CLR activates objects, verifies their security clearance, allocates their memory, executes them, and cleans up their memory once execution is finished.

As an analogy to how Java works, in Java, a program can run in multiple platforms with minimal or no modification: one language, but multiple platforms. In .NET, a program written in any of the forty supported languages can run in one platform, Windows, with minimal or no modification: multiple languages, but one platform.

ADO.NET is how data access support is provided in the .NET Framework. ADO.NET supports both connected and disconnected access. The key component of disconnected data access in ADO.NET is the `DataSet` class, instances of which act as a database cache that resides in your application's memory.

For both connected and disconnected access, your applications use databases through what is known as a ***data provider***. Various database products include their own .NET data providers, including DB2 for Windows.

A .NET data provider features implementations of the following basic classes:

- Connection: establishes and manages a database connection.
- Command: executes an SQL statement against a database.
- DataReader: reads and returns result set data from a database.
- DataAdapter: links a DataSet instance to a database. Through a DataAdapter instance, the DataSet can read and write database table data.

Three data providers that can work with DB2 are shown in *Table 14.3*

Data Provider	Characteristics
ODBC .NET Data provider (not recommended)	- Makes ODBC calls to a DB2 data source using the DB2 CLI driver. - It has same keyword support and restrictions as that

	of DB2 CLI driver
	▪ Can be used with .NET Framework Version 1.1, 2.0, or 3.0.
OLE DB .NET Data provider (not recommended)	▪ Uses IBM DB2 OLE DB Driver (IBMDADB2). ▪ It has same keyword support and restrictions as that of DB2 OLE DB driver ▪ Can be used only with .NET Framework Version 1.1, 2.0, or 3.0.
DB2 .NET Data provider (recommended)	▪ Extends DB2 support for the ADO.NET interface. ▪ The DB2 managed provider implements the same set of standard ADO.NET classes and methods ▪ It is defined under IBM.DATA.DB2 namespace. ▪ Can be obtained by downloading any of: 　- Data Server Driver for ODBC, CLI, and .NET 　- IBM Data Server Runtime Client 　- DB2 Data Server

Table 14.3 - ADO.NET data providers

14.3.7 PHP

PHP Hypertext Preprocessor (PHP) is an open source, platform independent scripting language designed for Web application development. It can be embedded within HTML, and generally runs on a Web server which takes the PHP code and creates Web pages as output.

PHP is a modular language. You can use extensions to customize the available functionality. Some of the most popular PHP extensions are those used to access databases. IBM supports access to DB2 databases through two extensions:

- **`ibm_db2`**: The ibm_db2 extension offers a procedural application programming interface to create, read, update and write database operations in addition to extensive access to the database metadata. It can be compiled with either PHP 4 or PHP 5.

- **`pdo_ibm`**: The pdo_ibm is a driver for the PHP Data Objects (PDO) extension that offers access to DB2 database through the standard object-oriented database interface introduced in PHP 5.1. It can be compiled directly against DB2 libraries.

The PHP extensions and drivers are available for free from the PECL repository at http://pecl.php.net/, or with the IBM Data Server Client. Both, ibm_db2 and pdo_ibm are based on the IBM DB2 CLI Layer

IBM has also partnered with Zend Technologies Inc. to support ZendCore, a free environment ready toolkit for developing with PHP and DB2 Express-C. The ZendCore bundle includes PHP libraries, the Apache Web server, and DB2 Express-C. To download ZendCore, go to http://www.ibm.com/software/data/info/zendcore

14.3.8 Ruby on Rails

Ruby is an open source object oriented language. Rails is a Web framework created using Ruby. Ruby on Rails (RoR) is an ideal means to develop database backed web-based applications. This hot new technology is based on the Model, View, Controller (MVC) architecture and follows the principles of agile software development.

Rails requires no special file formats or integrated development environments (IDEs); you can get started with a text editor. However, various IDEs are available with Rails support, such as RadRails, which is a Rails environment for Eclipse. For more information about RadRails, visit http://www.radrails.org/.

DB2 supports Ruby 1.8.5 and later and Ruby on Rails 1.2.1 and later. The IBM_DB gem includes the IBM_DB Ruby driver and Rails adapter which allows you to work with DB2 and is based on the CLI layer. This gem must be installed along with an IBM Data Server Client. To install the IBM_DB driver and adapter you can use Ruby gem or as a Rails plug-in.

14.3.9 Perl

Perl is a popular interpreted programming language that is freely available for many operating systems. It uses dynamic SQL, and it is ideal for prototyping applications.

Perl provides a standard module called the Database Interface (DBI) module for accessing different databases. It is available from http://www.perl.com. This module "talks" to drivers from different database vendors. In the case of DB2, this is the DBD::DB2 driver which is available from http://www.ibm.com/software/data/db2/perl.

14.3.10 Python

Python is a dynamic language often used for scripting. It emphasizes code readability and supports a variety of programming paradigms, including procedural, object-oriented, aspect-oriented, meta, and functional programming. Python is ideal for rapid application development.

Table 14.4 shows the extensions that are available for accessing DB2 databases from a Python application.

Extension	Description
ibm_db	Defined by IBM

	Provides the best support for advanced features.
	Allows you to issue SQL queries, call stored procedures, use pureXML, and access metadata information.
ibm_db_dbi	Implements the Python Database API Specification v2.0.
	It does not offer some of the advanced features that the ibm_db API supports.
	If you have an application with a driver that supports Python Database API Specification v2.0, you can easily switch to ibm_db. The ibm_db and ibm_db_dbi APIs are packaged together.
ibm_db_sa	Supports SQLAlchemy, a popular open source Python SQL toolkit and object-to-relational mapper (ORM).

Table 14.4 - IBM Data Server - Python extensions

14.4 XML and DB2 pureXML

Extensible Markup Language (XML) is the underlying technology for Web 2.0 tools and techniques, as well as for ***Service Oriented Architecture (SOA)***. IBM recognized early on the importance of XML, and large investments were made to deliver pureXML technology -- a technology that provides for better storage support XML documents in DB2.

Introduced in 2006, DB2 9 is a hybrid data server: it allows native storage of relational data, as well as hierarchical data. While previous versions of DB2 and other data servers in the marketplace could store XML documents, the storage method used in DB2 9 has improved performance and flexibility. With DB2 9's pureXML technology, XML documents are stored internally in a parsed hierarchical manner, as a tree; therefore, working with XML documents is greatly enhanced. Newer releases of DB2 such as DB2 9.5 and DB2 9.7 have further improved the support for pureXML. *Chapter 15, DB2 pureXML* is devoted to this subject in detail.

14.5 Web Services

As a simple definition, think of a Web service as a function you can invoke through the network, where you don't need to know the programming language used to develop it, you don't need to know the operating system where the function will run, and you don't need to know the location where it will run. Web services allow one application to exchange data with another application using extensible industry standard protocols based on XML. This is illustrated in *Figure 14.8*.

Figure 14.8 – How an example Web service works

In the figure, let's say the left side represents the system of a fictitious airline, Air Atlantis which is using DB2 on Linux, and stores its flight information in XML format in the DB2 database. On the right side we have a system from another fictitious airline, Air Discovery which is using SQL Server running on Windows. Now let's say that Air Atlantis and Air Discovery sign a partnership agreement where the two companies want to share scheduling and pricing information in order to coordinate their flights. Sharing information between the two may be a challenge given that the two companies are using different operating systems (Linux, Windows), and different data servers (DB2, SQL Server). When Air Atlantis changes its flight schedule for a trip going from Toronto to Beijing, how can Air Discovery automatically adjust its own flight schedule for a connecting flight from Beijing to Shanghai? The answer lies on Web services. Air Atlantis can expose some of its flight information by creating a ***Data Web service*** that returns the output of a stored procedure (the `getFlightInfo` stored procedure) with flight information from the DB2 database. A ***Data*** Web service is a Web service based on database information. When this Data Web service is deployed to an application server such as WebSphere Application Server; then a client or partner like Air Discovery can use a browser to access Air Atlantis' flight information very easily. In this example, Air Atlantis behaves as the Web service provider as it developed and made available the Web service, while Air Discovery behaves as the Web service consumer since it consumes or uses the Web service.

Air Discovery can also invoke the Web service from its own JDBC application so that it can perform calculations that use data from its SQL Server database. For example, if a flight

from Toronto to Beijing takes an average of 12 hours, Air Discovery can compute the connecting flight from Beijing to Shanghai by adding the departure time the Air Atlantis flight left Toronto, and adding the flight duration plus a few buffer hours. The amount of hours to use as buffer may be stored in the SQL Server database at Air Discovery's system, and the simple equation to use in the JDBC application would look like this:

Air Discovery Flight Departure time (Beijing to Shanghai)	=	Air Atlantis Flight Departure time (Toronto to Beijing)	+	Air Atlantis Flight Duration (Toronto to Beijing) 12 hours average	+	Air Atlantis Flight Buffer time (Toronto to Beijing) 2 hours

If Air Atlantis changes its flight departure time, this information is automatically communicated to the Air Discovery system when it invokes the Web service.

14.6 Administrative APIs

DB2 provides a large amount of administrative APIs that developers can use to build their own utilities or tools. For example, to create a database you can invoke the `sqlecrea API;` to start an instance, use the `db2InstanceStart API`; or to import data into a table, use the `db2Import API`. The complete list is available from the DB2 Information Center. See the *Resources* section for the DB2 Information Center URL.

14.7 Other development

Some users of DB2 Express-C also interact with third party products such as MS Excel and MS Access to create simple forms that connect to DB2. In this section we describe how to work with these products and DB2 Express-C.

DB2 Express-C is available also on Mac OS X, so you can use DB2 natively to develop database applications on a Mac. This may be especially appealing to the RoR community who has embraced the Mac platform.

14.7.1 Working with Microsoft Access and Microsoft Excel

Microsoft Excel and Microsoft Access are popular tools to generate reports, create forms, and develop simple applications that provide some business intelligence to your data. DB2 interacts very easily with these tools. A DBA can store the company data in a secure DB2 server, and regular users with Access or Excel can access this data and generate reports. This is illustrated in Figure 14.9

Figure 14.9 - Working with Excel, Access and DB2

In the figure, Excel and Access can be used to develop a front-end application, while DB2 takes care of data security, reliability and performance as the back-end of the application. Having all the data centralized in DB2 creates a simplified data storage model.

In the case of Excel, the easiest way to get access to the DB2 data is to use an OLE DB driver such as the IBM OLE DB Provider for DB2. This is included when you install the free IBM Data Server Client which can be downloaded from the DB2 Express-C web site at www.ibm.com/db2/express. Once installed, you need to select your data source with the appropriate OLE DB provider to use from the MS Excel menu. Choose *Data* ➔ *Import External Data* ➔ *Import Data*. The next steps are documented in the article *IBM® DB2® Universal Database™ and the Microsoft® Excel Application Developer… for Beginners* [1]. See the *References* section for details.

In the case of Microsoft Access, you should also have either of the following installed:

- IBM Data Server client, or

- IBM Data Server Driver for ODBC, CLI and .Net, or

- IBM Data Server Driver for ODBC and CLI

The IBM Data Server Driver for ODBC, CLI and .Net, and the IBM Data Server Driver for ODBC and CLI is also known as the IBM DB2 ODBC Driver, which is the same as the DB2 CLI driver. This is the driver to use to connect from Access to DB2. After the driver is installed, create an Access 2007 project, and choose the *ODBC Database* option available within the *External Data tab* in the *Table Tools* ribbon. The next steps are documented in the article *DB2 9 and Microsoft Access 2007 Part 1: Getting the Data...*[2]. When using **linked tables** in Microsoft Access, the data is available to Access 2007 users, but the data resides on the DB2 data server.

For versions of Access prior to 2007, the setup is a bit different, but you can review the article *Use Microsoft Access to interact with your DB2 data* [3]. See the *References* section for details.

14.8 Development Tools

Microsoft Visual Studio and Eclipse are two of the most popular Integrated Development Environments (IDEs) used by developers today. Both IDEs work well with DB2.

For Microsoft Visual Studio, DB2 provides a Visual Studio Add-in so that after installed, IBM tools are added to the Visual Studio menus. This way, a developer does not need to switch to other tools to work with DB2 databases. You can download the Visual Studio Add-in from the DB2 Express-C web site at www.ibm.com/db2/express.

With respect to Eclipse, IBM has released IBM Data Studio, a free Eclipse-based tool that allows you to develop SQL and XQuery scripts, stored procedures, UDFs, and Web services. Because it is based on the Eclipse platform, many developers can leverage their existing knowledge to work with this tool.

14.9 Sample programs

To help you learn how to program in different languages using DB2 as the data server, you can review the sample applications that come with the DB2 server installation in the SQLLIB\samples directory. *Figure 14.10* below shows some sample programs provided with DB2 on a Windows platform.

Figure 14.10 - Sample programs that come with DB2

14.10 Summary

In this chapter, we looked at how DB2 provides the flexibility to program database applications either inside the database on the server, or via client side applications with connections to the DB2 data server.

The server side application coverage included stored procedures, user-defined functions and triggers.

On the client side, we discussed the myriad of programming interfaces and methods permitted by DB2 application development, once again displaying the remarkable flexibility and capacity of DB2 as a database server.

Chapter 15 – DB2 pureXML

In this chapter we discuss pureXML, the new technology introduced in DB2 9 to support XML native storage. Many of the examples and concepts discussed in this chapter have been taken from the IBM Redbook: *DB2 9: pureXML overview and fast start*. See the Resources section for more information on this title. *Figure 15.1* outlines which section of the DB2 "Big Picture" we discuss in this chapter.

Figure 15.1 – The DB2 Big Picture: DB2 commands, SQL/XML and XQuery

Note:

For more information about pureXML, watch this video:
http://www.channeldb2.com/video/video/show?id=807741:Video:4382

15.1 Using XML with databases

XML documents can be stored in text files, XML repositories, or databases. There are two main reasons why many companies opt to store them in databases:

- Managing large volumes of XML data is a database problem. XML is data like other data, just in a different overall format. The same reasons to store relational data on databases apply to XML data: Databases provide efficient search and retrieval, robust support for persistence of data, backup and recovery, transaction support, performance and scalability.

- Integration: By storing relational and XML documents together, you can integrate new XML data with existing relational data, and combine SQL with XPath or XQuery in one query. Moreover, relational data can be published as XML, and vice versa. Through integration, databases can better support Web applications, SOA, and Web services.

15.2 XML databases

There are two types of databases for storing XML data:

- XML-enabled databases
- Native XML databases

15.2.1 XML-enabled databases

An XML-enabled database uses the relational model as its core data storage model to store XML. This requires either a mapping between the XML (hierarchical) data model and the relational data model, or else storing the XML data as a character large object. While this can be considered as "old" technology, it is still being used by many database vendors. *Figure 15.2* explains in more detail the two options for XML-enabled databases.

Figure 15.2 – Two options to store XML in XML-enabled databases

The left side of *Figure 15.2* shows the **CLOB and varchar method** of storing XML documents in a database. Using this method, an XML document is stored as an unparsed string in either a CLOB or a varchar column in the database. If the XML document is stored as a string, when you want to retrieve part of the XML document, your program will have to retrieve the entire string, and parse it to find what you want. Think of parsing as building the XML document tree in memory so you can navigate through this tree. This method is memory-intensive and not very flexible.

The other option for XML-enabled databases is called **shredding or decomposition** and is illustrated on the right hand side of *Figure 15.2*. Using this method, an entire XML document is shredded into smaller parts which are stored in tables. Using this method, you are literally forcing an XML document, which is based on the hierarchical model, into the relational model. This method is not flexible because if the XML document is changed, this change is not easily propagated into the corresponding tables and many other tables may need to be created. This method is also not good for performance: if you need to get the original XML document back, you need to perform an expensive SQL join operation, which can become even more expensive when more tables are involved.

15.2.2 Native XML databases

Native XML databases use the hierarchical XML data model to store and process XML internally. The storage format is the same as the processing format: there is no mapping to the relational model, and XML documents are not stored as unparsed strings (CLOBs or varchars). When XPath or XQuery statements are used, they are processed natively by the engine, and not converted to SQL. This is why these databases are known as "native" XML databases. DB2 is currently the only commercial data server providing this support.

15.3 XML in DB2

Figure 15.3 below outlines how relational data and hierarchical data (XML documents) are both stored in a DB2 hybrid database. The figure also shows the **CREATE TABLE** statement that was used to create the table dept.

```
CREATE TABLE dept (deptID CHAR(8),..., deptdoc XML);
```


Figure 15.3 – XML in DB2

Note that the table definition uses a new data type, XML, for the deptdoc column. The left arrow in the figure indicates the relational column deptID stored in relational format (tables), while the XML column deptdoc is stored in parsed hierarchical format.

Figure 15.4 illustrates that in DB2 9, there are now four ways to access data:

- Use SQL to access relational data

- Use SQL with XML extensions (SQL/XML) to access XML data

- Use XQuery to access XML data

- Use XQuery to access relational data

Figure 15.4 – Four ways to access data in DB2

Thus, depending on your background, if you are an SQL person you may see DB2 as a world class RDBMS that also supports XML. If you are an XML person, you would see DB2 as a world class XML repository that also supports SQL.

Note that IBM uses the term *pureXML* instead of *native XML* to describe this technology. While other vendors still use the old technologies of CLOB/varchar or shredding to store XML documents, they call those old technologies "native XML". To avoid confusion, IBM decided to use the new term pureXML, and to trademark this name so that no other database or XML vendor could use this same term to denote some differing technology. pureXML support is provided for databases created as Unicode or non-Unicode.

15.3.1 pureXML technology advantages

Many advantages are provided by pureXML technology.

1. You can seamlessly leverage your relational investment, given that XML documents are stored in columns of tables using the new XML data type.

2. You can reduce code complexity. For example, *Figure 15.5* illustrates a PHP script written with and without using pureXML. Using pureXML (the smaller box on the left side) the lines of code are reduced. This not only means that the code is less complex, but the overall performance is improved as there are fewer lines to parse and maintain in the code.

Figure 15.5 – Code complexity with and without pureXML

3. Changes to your schema are easier using XML and pureXML technology. *Figure 15.6* illustrates an example of this increased flexibility. In the figure, assume that

you had a database consisting of the tables *Employee* and *Department*. Typically with a non-XML database, if your manager asked you to store not only one phone number per employee (the home phone number), but also a second phone number (a cell phone number), then you could add an extra column to the *Employee* table and store the cell phone number in that new column. However, this method would be against the normalization rules of relational databases. If you want to preserve these rules, you should instead create a new *Phone* side table, and move all phone information to this table. You could then also add the cell phone numbers as well. Creating a new *Phone* table is costly, not only because large amounts of pre-existing data needs to be moved, but also because all the SQL in your applications would have to change to point to the new table.

Instead, on the left side of the figure, we show how this could be done using XML. If employee *Christine* also has a cell phone number, a new tag can be added to put this information. If employee *Michael* does not have a cell phone number, we just leave it as is.

Figure 15.6 – Increased data flexibility using XML

4. You can improve your XML application performance. Tests performed using pureXML technology showed huge improvements in performance for XML applications. *Table 15.1* shows the test results for a company that switched to pureXML from older technologies. The second column shows the results using the old method of working with XML using another relational database, and the third column shows the results using DB2 with pureXML.

Task	Other relational DB	DB2 with pureXML
Development of search and	CLOB: 8 hrs	30 min.

retrieval business processes	Shred: 2 hrs	
Relative lines of I/O code	100	35 (65% reduction)
Add field to schema	1 week	5 min.
Queries	24 - 36 hrs	20 sec - 10 min

Table 15.1 – Increased performance using pureXML technology

15.3.2 XPath basics

XPath is a language that can be used to query XML documents. *Listing 15.1* shows an XML document, and *Figure 15.7* illustrates the same document represented in parsed-hierarchical (also called "node" or "leaf") format. We will use the parsed-hierarchical format to explain how XPath works.

```
<dept bldg="101">
    <employee id="901">
            <name>John Doe</name>
<phone>408 555 1212</phone>
<office>344</office>
</employee>
<employee id="902">
<name>Peter Pan</name>
<phone>408 555 9918</phone>
<office>216</office>
</employee>
</dept>
```
Listing 15.1 – An XML document

Figure 15.7 – Parsed-hierarchical representation of the XML document in Listing 15.1

A quick way to learn XPath is to compare it to the `change directory (cd)` command in MS-DOS or Linux/UNIX. Using the `cd` command, you traverse a directory tree as follows:

```
cd /directory1/directory2/…
```

Similarly, in XPath you use slashes to go from one element to another within the XML document. For example, using the document in *Listing 15.1* in XPath you could retrieve the names of all employees using this query:

```
/dept/employee/name
```

15.3.2.1 XPath expressions

XPath expressions use fully qualified paths to specify elements and attributes. An "@" sign is used to specify an attribute. To retrieve only the value (text node) of an element, use the **text()** function. *Table 15.2* shows XPath queries and the corresponding results using the XML document from *Listing 15.1*.

XPath	Result
/dept/@bldg	101
/dept/employee/@id	901 902
/dept/employee/name	\<name>Peter Pan\</name> \<name>John Doe\</name>
/dept/employee/name/text()	Peter Pan John Doe

Table 15.2 – XPath expression examples

15.3.2.2 XPath wildcards

There are two main wildcards in XPath:

- "*" matches any tag name
- "//" is the "descendent-or-self" wildcard

Table 15.3 provides more examples using the XML document from *Listing 15.1*

XPath	Result
/dept/employee/*/text()	John Doe 408 555 1212 344 Peter Pan

	408 555 9918 216
/dept/*/@id	901 902
//name/text()	Peter Pan John Doe
/dept**//**phone	<phone>408 555 1212</phone> <phone>408 555 9918</phone>

Table 15.3 – XPath wildcard examples

15.3.2.3 XPath predicates

Predicates are enclosed in square brackets [ ]. As an analogy, you can think of them as the equivalent to the WHERE clause in SQL. For example `[@id="902"]` can be read as: "WHERE attribute id is equal to 902". There can be multiple predicates in one XPath expression. To specify a positional predicate, use [n] which means the nth child would be selected. For Example, `employee[2]` means that the second employee should be selected. *Table 15.4* provides more examples.

XPath	Result
/dept/employee[@id="902"]/name	<name>Peter Pan</name>
/dept[@bldg="101"]/employee[office >"300"]/name	<name>John Doe</name>

//employee[office="344"   OR
office="216"]/@id
	901
	902

/dept/employee[2]/@id	902

Table 15.4 – XPath predicate examples

15.3.2.4 The parent axis

Similar to MS-DOS or Linux/UNIX, you can use a "**.**" (dot) to indicate in the expression that you are referring to the current context, and a "**..**" (dot dot) to refer to the parent context. *Table 15.5* provides more examples.

XPath	Result
/dept/employee/name[../@id="902"]	<name>Peter Pan</name>
/dept/employee/office[.>"300"]	<office>344</office>
/dept/employee[office > "300"]/office	<office>344</office>

`/dept/employee[name="John Doe"]/../@bldg`	101
`/dept/employee/name[.="John Doe"]/../../@bldg`	101

Table 15.5 – XPath parent axis

15.3.3 XQuery basics

XQuery is a query language created for XML. XQuery supports path expressions to navigate the XML hierarchical structure. In fact, XPath is a subset of XQuery; therefore, everything we learned earlier about XPath applies to XQuery too. XQuery supports both typed and untyped data. XQuery lacks null values because XML documents omit missing or unknown data. XQuery and XPath expressions are case sensitive, and XQuery returns sequences of XML data.

XQuery supports the FLWOR expression. If we use SQL for an analogy, it is equivalent to a SELECT-FROM-WHERE expression. The next section describes FLWOR in more detail.

15.3.3.1 XQuery: FLWOR expression

FLWOR stands for:

- FOR: iterates through a sequence, binds a variable to items
- LET: binds a variable to a sequence
- WHERE: eliminates items of the iteration
- ORDER: reorders items of the iteration
- RETURN: constructs query results

It is an expression that allows manipulation of XML documents, enabling you to return another expression. For example, assume you have a table with this definition:

```
CREATE TABLE dept(deptID CHAR(8),deptdoc XML);
```

And the XML document in *Listing 15.2* is inserted in the **deptdoc** column

```
<dept bldg="101">
    <employee id="901">
        <name>John Doe</name>
        <phone>408 555 1212</phone>
        <office>344</office>
    </employee>
    <employee id="902">
        <name>Peter Pan</name>
        <phone>408 555 9918</phone>
        <office>216</office>
    </employee>
```
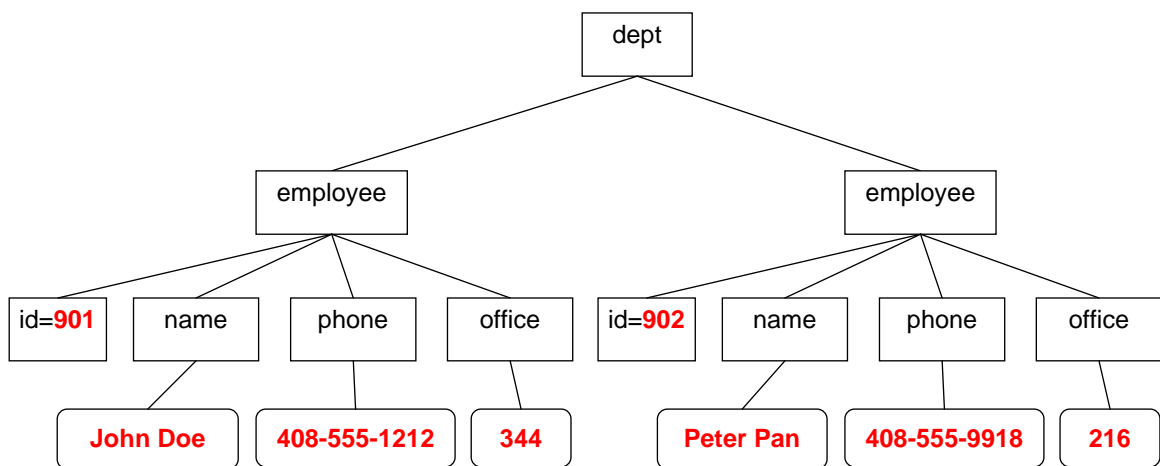
```
</dept>
```
Listing 15.2 - A sample XML document

Then the XQuery statement in *Listing 15.3* using the FLWOR expression could be run:

```
xquery
for $d in db2-fn:xmlcolumn('dept.deptdoc')/dept
let $emp := $d//employee/name
where $d/@bldg > 95
order by $d/@bldg
return
   <EmpList>
   {$d/@bldg, $emp}
   </EmpList>
```
Listing 15.3 - A sample XQuery statement with the FLWOR expression

This would return the output shown in *Listing 15.4*

```
<EmpList bldg="101">
  <name>
      John Doe
  </name>
  <name>
      Peter Pan
  </name>
</EmpList>
```
Listing 15.4 - Output after running the XQuery statement in Listing 15.3

15.3.4 Inserting XML documents

Inserting XML documents into a DB2 database can be performed using the SQL INSERT statement, or the IMPORT utility. XQuery cannot be used for this purpose as this has not yet been defined in the standard.

Let's examine the script `table_creation.txt` shown in *Listing 15.5* below, which can be run from the DB2 Command Window or Linux shell using this statement:

```
      db2 –tvf table_creation.txt


-- (1)
drop database mydb
;


-- (2)
create database mydb using codeset UTF-8 territory US
;


-- (3)
connect to mydb
```

```
;


-- (4)
create table items (
  id         int primary key not null,
  brandname  varchar(30),
  itemname   varchar(30),
  sku        int,
  srp        decimal(7,2),
  comments   xml
);


-- (5)
create table clients(
  id         int primary key not null,
  name       varchar(50),
  status     varchar(10),
  contact    xml
);


-- (6)
insert into clients values (77, 'John Smith', 'Gold',
   '<addr>111 Main St., Dallas, TX, 00112</addr>')
;


-- (7)
IMPORT FROM "D:\Raul\clients.del" of del xml from "D:\Raul" INSERT INTO
CLIENTS (ID, NAME, STATUS, CONTACT)
;


-- (8)
IMPORT FROM "D:\Raul\items.del" of del xml from "D:\Raul" INSERT INTO
ITEMS (ID, BRANDNAME, ITEMNAME, SKU, SRP, COMMENTS)
;
```

Listing 15.5 - Contents of the file table_creation.txt

Note that this script file and related files are provided in the compressed file **Expressc_book_exercises_9.7.zip** that accompanies this book. Follow along as we describe each line in the script of *Listing 15.5.*

1. Drop the database *mydb*. This is normally done in script files to perform cleanup. If *mydb* didn't exist before, you will receive an error message, but this is OK.

2. Create the database *mydb* using the codeset UTF-8. This creates a Unicode database. pureXML is supported in both Unicode and non-Unicode databases.

3. Connect to the newly created database **mydb**. This is necessary to create objects within the database.

4. Create the table **items**. Note that the last column in the table (column **comments**) is defined as an XML column using the new XML data type.

5. We create the table **clients**. Note that the last column in the table (column **contact**) is also defined with the new XML data type.

6. Using this SQL INSERT statement, you can insert an XML document into an XML column. In the INSERT statement you pass the XML document as a string enclosed in single quotes.

7. Using an IMPORT command, you can insert or import several XML documents along relational data into the database. In (7) you are importing the data from the `clients.del` file (a delimited ascii file), and you also indicate where the XML data referenced by that `clients.del` file is located (for this example, in `D:\Raul`).

We will take a more careful look at file `clients.del`, but first, let's see the contents of directory `D:\Raul` first. *Figure 15.8* provides this information.

Figure 15.8 - Contents of D:\Raul directory with XML documents

Listing 15.6 shows the contents of the text file `clients.del`.

```
3227,Ella Kimpton,Gold,<XDS FIL='Client3227.xml' />,
8877,Chris Bontempo,Gold,<XDS FIL='Client8877.xml'/>,
9077,Lisa Hansen,Silver,<XDS FIL='Client9077.xml' />
```

```
9177,Rita Gomez,Standard,<XDS FIL='Client9177.xml'/>,
5681,Paula Lipenski,Standard,<XDS FIL='Client5681.xml' />,
4309,Tina Wang,Standard,<XDS FIL='Client4309.xml'/>
```
Listing 15.6 - Contents of the file clients.del

In the `clients.del` file, "`XDS FIL=`" is used to point to a specific XML document file.

Figure 15.9 shows the Control Center after running the above script.

Figure 15.9 – The Control Center after running the table_creation.txt script

Note that in the figure, we show the contents of the **CLIENTS** table. The last column **contact** is an XML column. When you click on the button with three dots, another window opens showing you the XML document contents. This is shown in the bottom right corner of the *Figure 15.9*.

15.3.5 Querying XML data

There are two ways to query XML data in DB2:

  ▪ Using SQL with XML extensions (SQL/XML)
  ▪ Using XQuery

In both cases, DB2 follows international XML standards.

15.3.5.1 Querying XML data with SQL/XML

Using regular SQL statements allows you to work with rows and columns. An SQL statement can be used to work with full XML documents; however, it would not help when attempting to retrieve only part of the document. In such cases, you need to use SQL with XML extensions (SQL/XML).

Table 15.6 describes some of the SQL/XML functions available with the SQL 2006 standard

Function name	Description
XMLPARSE	Parses character or large object binary data, produces XML value
XMLSERIALIZE	Converts an XML value into character or large object binary data
XMLVALIDATE	Validates XML value against an XML schema and type-annotates the XML value
XMLEXISTS	Determines if an XQuery returns a results (i.e. a sequence of one or more items)
XMLQUERY	Executes an XQuery and returns the result sequence
XMLTABLE	Executes an XQuery, returns the result sequence as a relational table (if possible)
XMLCAST	Cast to or from an XML type

Table 15.6 – SQL/XML Functions

The following examples can be tested using the *mydb* database created earlier.

Example 1

Imagine that you need to locate the names of all clients who live in a specific zip code. The **clients** table stores customer addresses, including zip codes, in an XML column. Using XMLEXISTS, you can search the XML column for the target zip code and then restrict the return result set accordingly. *Listing 15.7* below illustrates the query required.

```
SELECT name FROM clients
 WHERE xmlexists(
     '$c/Client/Address[zip="95116"]'
     passing clients.contact as "c"
```

```
        )
```

Listing 15.7 - An example using XMLEXISTS

In *Listing 15.7*, the first line is an SQL clause specifying that you want to retrieve information in the `name` column of the `clients` table.

The WHERE clause invokes the XMLEXISTS function, specifying the XPath expression that prompts DB2 to navigate to the `zip` element and check for a value of 95116

The `$c/Client/Address` clause indicates the path inside the XML document hierarchy where DB2 can locate the `zip` element. A dollar sign ($) is used to specify a variable; therefore "c" is a variable.   This variable is then defined by this line: `passing clients.contact as "c"`. Here, `clients` is the name of the table and `contact` is the name of the column with an XML data type. In other words, we are passing the XML document to the variable "c".

DB2 inspects the XML data contained in the `contact` column, navigates from the root `Client` node down to the `Address` node, then to the `zip` node and finally determines if the customer lives in the target zip code. The XMLEXISTS function evaluates to "true" and DB2 returns the name of the client associated with that row.

Starting with DB2 9.5, the above query could be simplified as shown in Listing 15.8 below.

```
SELECT name FROM clients
 WHERE xmlexists(
     '$CONTACT/Client/Address[zip="95116"]'
     )
```

Listing 15.8 - Simplified version of the query shown in Listing 15.7

A variable with the same name as an XML column is created automatically by DB2. In the above example, the variable *CONTACT* is created automatically by DB2. Its name matches the name of the XML column `CONTACT`.

Example 2

Let's consider how to solve the problem of how to create a report listing the e-mail addresses of "Gold" status customers. The query in *Listing 15.9* below could be run for this purpose.

```
SELECT xmlquery('$c/Client/email' passing contact as "c")
  FROM  clients
  WHERE status = 'Gold'
```

Listing 15.9 - An example using XMLQUERY

The first line indicates we want to return the email address which is an element of the XML document (not a relational column). As in the previous example, "$c" is a variable that contains the XML document. In this example we use the XMLQUERY function which can be used after a SELECT, while the XMLEXISTS function can be used after a WHERE clause.

Example 3

There may be situations when you would like to present XML data as tables. This is possible with the XMLTABLE function as shown in *Listing 15.10* below.

```
SELECT t.comment#, i.itemname, t.customerID, Message
  FROM items i,
  xmltable('$c/Comments/Comment' passing i.comments as "c"
      columns Comment# integer path 'CommentID',
      CustomerID integer path 'CustomerID',
      Message varchar(100) path 'Message') AS t
```
Listing 15.10 - An example using XMLTABLE

The first line specifies the columns to be included in your results set. Columns prefixed with the "t" variable are based on XML element values.

The third line invokes the XMLTABLE function to specify the DB2 XML column containing the target data (`i.comments`) and the path within the column's XML documents where the elements of interest are located.

The `columns` clause, spanning lines 4 to 6, identifies the specific XML elements that will be mapped to output columns in the SQL result set specified on line 1. Part of this mapping involves specifying the data types to which the XML element values will be converted. In this example all XML data is converted to traditional SQL data types.

Example 4

Now let's explore a simple example in which you include an XQuery FLWOR expression inside an XMLQUERY SQL/XML function. This is illustrated in *Listing 15.11*.

```
SELECT name, xmlquery(
    'for $e in $c/Client/email[1] return $e'
    passing contact as "c"
    )
  FROM clients
  WHERE status = 'Gold'
```
Listing 15.11 - An example using XMLQUERY and FLWOR

The first line specifies that the customer names and the output from the XMLQUERY function will be included in the result set. The second line indicates the first *email* sub-element of the *Client* element is to be returned. The third line identifies the source of our XML data (`contact` column). The fourth line tells us that this column is coming from the `clients` table; and the fifth line indicates that only `Gold` customers are of interest.

Example 5

The example illustrated in *Listing 15.12* demonstrates again the XMLQUERY function which takes an XQuery FLWOR expression; however, note that this time we are returning not only XML, but also HTML.

```
SELECT xmlquery('for $e in $c/Client/email[1]/text()
    return <p>{$e}</p>'
    passing contact as "c")
  FROM clients
  WHERE status = 'Gold'
```
Listing 15.12 - An example returning XML and HTML

The return clause of XQuery enables you to transform XML output as needed. Using the `text()`function in the first line indicates that only the text representation of the first email address of qualifying customers is of interest. The second line specifies that this information is to be surrounded by HTML paragraph tags.

Example 6

The following example uses the XMLELEMENT function to create a series of item elements, each of which contain sub-elements for the ID, brand name, and stock keeping unit (SKU) values obtained from corresponding columns in the `items` table. Basically, you can use the XMLELEMENT function when you want to convert relational data to XML data. This is illustrated in Listing 15.13.

```
SELECT
  xmlelement (name "item", itemname),
  xmlelement (name "id", id),
  xmlelement (name "brand", brandname),
  xmlelement (name "sku", sku)
  FROM items
  WHERE srp < 100
```
Listing 15.13 - An example using XMLELEMENT

The query in *Listing 15.13* would return the output as shown in *Listing 15.14*

```
<item>
      <id>4272</id>
      <brand>Classy</brand>
      <sku>981140</sku>
</item>
…
<item>
      <id>1193</id>
      <brand>Natural</brand
      <sku>557813</sku>
</item>
```
Listing 15.14 - Output of the query in Listing 15.13

15.3.5.2 Querying XML data with XQuery

In the previous section, we looked at how to query XML data using SQL with XML extensions. SQL was always the primary query method, and XPath or XQuery was embedded inside SQL. In this section, we discuss how to query XML data using XQuery. This time, XQuery will be the primary query method, and in some cases, we will use SQL embedded inside XQuery (using the **db2-fn:sqlquery** function). When using XQuery, we will invoke a few functions, and will also use the FLWOR expression.

Example 1

This is a simple XQuery to return customer contact data. In the example, **CONTACT** is the name of the XML column, and **CLIENTS** is the name of the table.

```
xquery db2-fn:xmlcolumn('CLIENTS.CONTACT')
```

Always prefix any XQuery expression with the **xquery** command so that DB2 knows it has to use the XQuery parser, otherwise DB2 will assume you are trying to run an SQL expression. The **db2-fn:xmlcolumn** function is a function that retrieves the XML documents from the column specified as the parameter. It is equivalent to the following SQL statement, as it is retrieving the entire column contents:

```
SELECT contact FROM clients
```

Example 2

In this example shown in *Listing 15.15*, we use the FLWOR expression to retrieve client fax data

```
xquery
  for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/fax
  return $y
```
Listing 15.15 - XQuery and the FLWOR expression

The first line invokes the XQuery parser. The second line instructs DB2 to iterate through the fax sub-elements contained in the **CLIENTS.CONTACT** column. Each fax element is bound to the variable $y. The third line indicates that for each iteration, the value "$y" is returned.

The output of this query is illustrated in *Listing 15.16* (We omitted the namespace in the output, otherwise it would be harder to read as it may span several lines):

```
<fax>4081112222</fax>
<fax>5559998888</fax>
```
Listing 15.16 - Output of the query show in Listing 15.15

Example 3

The example in Listing 15.17 queries XML data and returns the results as HTML.

```
xquery
  <ul> {
    for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/Address
    order by $y/zip
    return <li>{$y}</li>
  }
  </ul>
```

Listing 15.17 - XQuery statement with the FLWOR expression returning HTML

The sample HTML returned would look as shown in *Listing 15.18.*

```
<ul>
<li>
<address>
    <street>9407 Los Gatos Blvd.</street>
    <city>Los Gatos</city>
    <state>ca</state>
    <zip>95302</zip>
</address>
</li>
<address>
<street>4209 El Camino Real</street>
    <city>Mountain View</city>
    <state>CA</state>
    <zip>95302</zip>
</address>
</li>
...
</ul>
```

Listing 15.18 - Output of the query ran in Listing 15.17

Example 4

The following example shows how to embed SQL within XQuery by using the **db2-fn:sqlquery** function. The **db2-fn:sqlquery** function executes an SQL query and returns only the selected XML data. The SQL query passed to **db2-fn:sqlquery** must only return XML data. This XML data can then be further processed by XQuery. This is illustrated in Listing 15.19.

```
xquery
  for $y in
  db2-fn:sqlquery(
    'select comments from items where srp > 100'
  )/Comments/Comment
```

```
where $y/ResponseRequested='Yes'
return (
  <action>
    {$y/ProductID
     $y/CustomerID
     $y/Message}
  </action>
)
```

Listing 15.19 - An example of the `db2-fn:sqlquery` function embedding SQL within XQuery

In the example, the SQL query filters rows based on the condition that the **srp** column has a value greater than 100. From those rows filtered, it will pick the **comments** column, which is the XML column. Next XQuery (or XPath) is applied to go to sub-elements.

Note:

SQL is case insensitive and DB2 stores all table and column names in uppercase by default. XQuery on the other hand, is case sensitive. The above functions are XQuery interface functions so all the table names and column names should be passed to these functions in uppercase. Passing the object names in lowercase may result in an undefined object name error.

15.3.6 Joins with SQL/XML

This section describes how to perform JOIN operations between two XML columns of different tables, or between one XML column and one relational column. Assume you have created two tables with the statements shown in *Listing 15.20*

```
CREATE TABLE dept (unitID CHAR(8), deptdoc XML)
CREATE TABLE unit (unitID  CHAR(8) primary key not null,
                   name    CHAR(20),
                   manager VARCHAR(20),
                   ...
                   )
```

Listing 15.20 - DDL of tables to use in the JOIN examples

You can perform a JOIN operation in either of two ways. The first method is shown in *Listing 15.21*.

```
SELECT u.unitID
  FROM dept d, unit u
  WHERE XMLEXISTS (
    '$e//employee[name = $m]'
    passing d.deptdoc as "e", u.manager as "m")
```

Listing 15.21 - First method to perform a JOIN with SQL/XML

Line 4 of the statement in the above listing shows that the JOIN operation occurs between the element **name**, which is a sub-element of the **deptdoc** XML column in table **dept**, and the **manager** relational column in the table **unit**.

Listing 15.22 shows the second method to perform the JOIN.

```
SELECT u.unitID
  FROM dept d, unit u
  WHERE u.manager = XMLCAST(
    XMLQUERY('$e//employee/name '
      passing d.deptdoc as "e")
    AS char(20))
```
Listing 15.22 - Second method to perform a JOIN with SQL/XML

In this second method, the relational column is on the left side of the JOIN. If the relational column is on the left side of the equation, a relational index may be used instead of an XML index.

15.3.7 Joins with XQuery

Assume the following tables have been created:

```
    CREATE TABLE dept(unitID CHAR(8), deptdoc XML)
    CREATE TABLE project(projectDoc XML)
```

If we use SQL/XML, a JOIN would look as shown in *Listing 15.23*.

```
SELECT XMLQUERY (
  '$d/dept/employee' passing d.deptdoc as "d")
  FROM dept d, project p
  WHERE XMLEXISTS (
    '$e/dept[@deptID=$p/project/deptID]'
      passing d.deptdoc as "e", p.projectDoc as "p")
```
Listing 15.23 - A JOIN with SQL/XML

The equivalent JOIN using XQuery is shown in *Listing 15.24*.

```
xquery
  for $dept in db2-fn:xmlcolumn("DEPT.DEPTDOC")/dept
  for $proj in db2-fn:xmlcolumn("PROJECT.PROJECTDOC")/project
  where $dept/@deptID = $proj/deptID
  return $dept/employee
```
Listing 15.24 - A JOIN with XQuery

This second method is easier to interpret -- variable $*dept* holds the XML document of the XML column **deptdoc** in table **dept**. The variable $*proj* holds the XML document of the XML column **projectdoc** in table **project**. Then line 4 performs the JOIN operation between an attribute of the first XML document and an element of the second XML document.

15.3.8 Update and delete operations

Update and delete operations on XML data can be performed in one of two ways:

- Using SQL UPDATE and DELETE statements

- Using the TRANSFORM expression

For the first way using SQL UPDATE and DELETE statements, the update or delete occurs at the document level; that is, the entire XML document is replaced with the updated one. For example, in the UPDATE statement in *Listing 15.25* below, if you'd only like to change the <state> element, the entire XML document is actually replaced.

```
UPDATE clients SET contact=(
  xmlparse(document
    '<Client>
      <address>
        <street>5401 Julio ave.</street>
        <city>San Jose</city>
        <state>CA</state>
        <zip>95116</zip>
      </address>
      <phone>
        <work>4084633000</work>
        <home>4081111111</home>
        <cell>4082222222</cell>
      </phone>
      <fax>4087776666</fax>
      <email>newemail@someplace.com</email>
    </Client>')
  )
  WHERE id = 3227
```
Listing 15.25 - An example of an SQL UPDATE

For the second way, you can perform sub-document updates using the TRANSFORM expression, which is a lot more efficient. This allows you to replace, insert, delete or rename nodes in an XML document. You can also change the value of a node without replacing the node itself, typically to change an element or attribute value–which is a very common type of update. This support was added in DB2 9.5.

The TRANSFORM expression is part of the XQuery language, you can use it anywhere you normally use XQuery, for example in a FLWOR expression or in the XMLQUERY function in an SQL/XML statement. The most typical use is in an SQL UPDATE statement to modify an XML document in an XML column.

Listing 15.26 shows the syntax of the TRANSFORM expression.

```
>>-transform--| copy clause |--| modify clause |--| return clause |-><


copy clause
```

```
      .-,-------------------------------------.
      V                                       |
|--copy----$VariableName--:=--CopySourceExpression-+-----------|

modify clause

|--modify--ModifyExpression-----------------------------------|

return clause

|--return--ReturnExpression-----------------------------------|
```

Listing 15.26 - The syntax of the TRANSFORM expression

The `copy` clause is used to assign to a variable the XML documents you want to process.

In the `modify` clause, you can invoke an `insert`, `delete`, `rename`, or `replace` expression. These expressions allow you to perform updates to your XML document.

For example:

- If you want to add new nodes to the document, you would use the `insert` expression

- To delete nodes from an XML document, use the `delete` expression

- To rename an element or attribute in the XML document, use the `rename` expression

- To replace an existing node with a new node or sequence of nodes, use the `replace` expression. The replace value of the expression can only be used to change the value of an element or attribute.

The return clause returns the result of the transform expression.

Listing 15.27 shows an example of an UPDATE statement using the TRANSFORM expression.

```
(1)-- UPDATE customers
(2)-- SET contactinfo = xmlquery( 'declare default element namespace
(3)--                       "http://posample.org";
(4)--    transform
(5)--    copy $newinfo := $c
(6)--          modify do insert <email2>my2email.gm.com</email2>
(7)--                   as last into $newinfo/customerinfo
(8)--    return $newinfo' passing contactinfo as "c")
(9)-- WHERE id = 100
```
Listing 15.27 - An UPDATE using the TRANSFORM expression

In the above example, lines (1), (2), and (9) are part of the SQL UPDATE syntax. In Line (2) the XMLQUERY function is invoked, which calls the transform expression in line (4). The transform expression block goes from line (4) to line (8), and it is used to insert a new node into the XML document containing the **email2** element. Note that updating the elements in an XML document through a view is not supported.

Deleting entire XML documents from tables is as straightforward as when using the SELECT statement in SQL/XML. Use the SQL DELETE statement and specify any necessary WHERE predicates.

15.3.9 XML indexing

In an XML document, indexes can be created for elements, attributes, or for values (text nodes). Below are some examples. Assume the table below was created:

```
CREATE TABLE customer(info XML)
```

And assume the XML document in Listing 15.28 is one of the documents stored in the table.

```
<customerinfo Cid="1004">
   <name>Matt Foreman</name>
   <addr country="Canada">
       <street>1596 Baseline</street>
       <city>Toronto</city>
       <state>Ontario</state>
       <pcode>M3Z-5H9</pcode>
   </addr>
   <phone type="work">905-555-4789</phone>
   <phone type="home">416-555-3376</phone>
   <assistant>
      <name>Peter Smith</name>
      <phone type="home">416-555-3426</phone>
   </assistant>
</customerinfo>
```
Listing 15.28 - The XML document to use in the examples related to XML indexes

The statement shown in *Listing 15.29* creates an index on the attribute **Cid**

```
CREATE UNIQUE INDEX idx1 ON customer(info)
  GENERATE KEY USING
  xmlpattern '/customerinfo/@Cid'
  AS sql DOUBLE
```
Listing 15.29 - An index on attribute Cid

The statement shown in *Listing 15.30* creates an index on the element **name**

```
CREATE INDEX idx2 ON customer(info)
  GENERATE KEY USING
  xmlpattern '/customerinfo/name'
```

```
   AS sql VARCHAR(40)
```
Listing 15.30 - An index on element name

The statement in *Listing 15.31* creates an index on all elements *name*

```
CREATE INDEX idx3 ON customer(info)
  GENERATE KEY USING
  xmlpattern '//name'
  AS sql VARCHAR(40);
```
Listing 15.31 - An index on all elements name

The statement in *Listing 15.32* creates an index on all text nodes (all values). This is not recommended, as it would be too expensive to maintain the index for update, delete and insert operations, and the index would be too large.

```
CREATE INDEX idx4 ON customer(info)
  GENERATE KEY USING
  xmlpattern '//text()'
  AS sql VARCHAR(40);
```
Listing 15.32 - An index on all text nodes (Not recommended)

15.4 Working with XML Schemas

DB2 allows you to insert an XML document into the database if it is well-formed. If it's not, you will receive an error at insertion time. On the other hand, DB2 does not force you to validate a XML document. If you wish to have an XML document validated, you have several alternatives as we will discuss in this section.

15.4.1 Registering your XML Schemas

XML Schemas are stored in the DB2 databases in what is called an XML Schema repository. To add an XML Schema to a repository, you use the REGISTER XMLSCHEMA command.

For example, let's say you have an XML document stored in file `order.xml` as shown in *Figure 15.10*

```
<?xml version="1.0" encoding="UTF-8"?>
  <po:PurchaseOrder xmlns:po="http://www.test.com/po">
     <Header>
          <Id>1</Id>
          <date>2004-01-29</date>
          <description>purchase order</description>
          <value>20</value>
          <status>shipped</status>
     </Header>
     <Items>
          <Item>
               <ItemDescription color="red" weight="5">
                        <Name>Widget C</Name>
                        <SKU>1</SKU>
                        <Price>30</Price>
                        <Comment>no comment</Comment>
               </ItemDescription>
               <NumberOrdered>1</NumberOrdered>
          </Item>
     </Items>
     <Customer type="regualar">
          <Name>Manoj K Sardana</Name>
          <Address>ring road, bangalore</Address>
          <Phone>918051055109</Phone>
          <email>msardana@in.ibm.com</email>
     </Customer>
  </po:PurchaseOrder>
```

Figure 15.10 - The order.xml file containing an XML document

Now, let's say you have an XML Schema document stored in file `order.xsd` as shown in *Figure 15.11*

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.test.com/po"
            xmlns:po="http://www.test.com/po"
            xmlns:head="http://www.test.com/header"
            xmlns:prod = "http://www.test.com/product"
            xmlns:cust = "http://www.test.com/customer">

<xsd:import namespace="http://www.test.com/product" schemaLocation="product.xsd" />
<xsd:import namespace="http://www.test.com/customer" schemaLocation="customer.xsd" />
<xsd:import namespace="http://www.test.com/header" schemaLocation="header.xsd" />
<xsd:complexType name="itemType">
<xsd:sequence>
  <xsd:element name="Item" minOccurs="1" maxOccurs="unbounded" >
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="ItemDescription" type="prod:prodType" />
        <xsd:element name="NumberOrdered" type="xsd:integer" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="potype">
<xsd:sequence>
  <xsd:element name="Header"    type="head:headerType"  />
  <xsd:element name="Items"     type="po:itemType" />
  <xsd:element name="Customer" type="cust:customerType" />
</xsd:sequence>
</xsd:complexType>

<xsd:element name="PurchaseOrder" type="po:potype" />

</xsd:schema>
```

Figure 15.11 - The order.xsd file containing an XML schema

In this XML Schema document we highlight with an ellipse the following:

- <xsd:schema ….>: Indicates it's a XML Schema document

- <xsd:import …>: We import other xsd files (other XML Schemas) that would be part of this bigger XML Schema.

- minOccurs="1": An example of an XML Schema "rule", where for element **Item** we say that it should occur at least one time, or in other words, there should be at least one **Item** element.

Next, in order to register the XML Schema to the database, a script similar to the one shown in *Listing 15.33* below could be used. The script includes comments that make it self-explanatory.

```
-- CONNECT TO THE DATABASE
CONNECT TO SAMPLE;


-- REGISTER THE MAIN XML SCHEMA
REGISTER XMLSCHEMA http://www.test.com/order FROM D:\example3\order.xsd AS
order;
```

```
-- ADD XML SCHEMA DOCUMENT TO MAIN SCHEMA
ADD  XMLSCHEMA  DOCUMENT  TO  order  ADD  http://www.test.com/header  FROM
D:\example3\header.xsd;


-- ADD XML SCHEMA DOCUMENT TO MAIN SCHEMA
ADD  XMLSCHEMA  DOCUMENT  TO  order  ADD  http://www.test.com/product  FROM
D:\example3\product.xsd;


-- ADD XML SCHEMA DOCUMENT TO MAIN SCHEMA
ADD  XMLSCHEMA  DOCUMENT  TO  order  ADD  http://www.test.com/customer  FROM
D:\example3\customer.xsd;


-- COMPLETE THE SCHEMA REGISTRATION
COMPLETE XMLSCHEMA order;
```
Listing 15.33 - A sample script showing the steps to register an XML schema

To review this information later you can SELECT the information from the Catalog tables as shown in *Listing 15.34* below.

```
SELECT CAST(OBJECTSCHEMA AS VARCHAR(15)), CAST(OBJECTNAME AS VARCHAR(15))
  FROM syscat.xsrobjects
  WHERE OBJECTNAME='ORDER';
```
Listing 15.34 - Retrieving XML schema information from the DB2 Catalog tables

15.4.2 XML Schema validation

Once your XML Schemas have been registered in DB2, you can validate your XML documents in two ways:

- Use the XMLVALIDATE function during an INSERT
- Use a BEFORE Trigger

Figure 15.12 shows an example where the XML document shown in *Figure 15.10* is validated according to the XML Schema shown in *Figure 15.11*.

```
DROP TABLE t1;
CREATE TABLE t1 (po xml);

INSERT INTO t1 VALUES(xmlvalidate(xmlparse(document('<?xml version="1.0" encoding="UTF-8"?>
            <po:PurchaseOrder xmlns:po="http://www.test.com/po">
              <Header>
                  <Id>1</Id>
                  <date>2004-01-29</date>
                  <description>purchase order</description>
                  <value>20</value>
                  <status>shipped</status>
              </Header>
              <Items>
                  <Item>
                      <ItemDescription color="red" weight="5">
                          <Name>Widget C</Name>
                          <SKU>1</SKU>
                          <Price>30</Price>
                          <Comment>no comment</Comment>
                      </ItemDescription>
                      <NumberOrdered>1</NumberOrdered>
                  </Item>
              </Items>
              <Customer type="regular">
                  <Name>Manoj K Sardana</Name>
                  <Address>ring road, bangalore</Address>
                  <Phone>918051055109</Phone>
                  <email>msardana@in.ibm.com</email>
              </Customer>
            </po:PurchaseOrder>')) ACCORDING TO XMLSCHEMA ID order));
```

Figure 15.12 - XML Schema validation using XMLVALIDATE

To test if an XML document has been validated, you can use the "IS VALIDATED" predicate on a CHECK constraint.

You can validate XML documents in a column using different XML schemas. This is important for easy migration from version 1 to version 2 of an XML schema. In the same XML column, you may also find XML documents with no validation at all. This is useful if documents are received from trusted and non-trusted sources where only the later require schema validation.

15.4.3 Other XML support

Small XML documents can now be in-lined with the base table. This means that the XML data is stored in the same place as the relational data, and can take advantage of the same compression mechanisms as regular relational data. Larger XML documents are stored in a separate internal object, which can also be compressed.

DB2 also supports XML Schema evolution. This means that if your XML Schema changes, you can update the XML Schema easily with the UPDATE XMLSCHEMA command. If the changes to the XML Schema are too drastic, you are likely to get some errors.

In DB2 XML decomposition or "shredding" is also supported. This is the "old" method to store XML in databases, and is what other vendors use to store XML. DB2 still supports this method if you wish to use it; but we recommend pureXML. DB2 also supports the XML Extender, also using the old method to store XML, but this extender will no longer be enhanced.

With DB2 9.7 all the benefits of pureXML has been extended to database partitions commonly used for data warehouses. Database Partitioning Feature (DPF) is offered with DB2 Enterprise Edition.

15.6 Summary

This chapter introduced you to XML and pureXML technology. XML document usage is growing exponentially due to Web 2.0 tools and techniques as well as SOA. By storing XML documents in a DB2 database you can take advantage of security, performance, and coding flexibility using pureXML. pureXML is a technology that allows you to store the XML documents in parsed-hierarchical format, as a tree, and this is done at database insertion time. At query time, there is no need to parse the XML document in order to build a tree before processing. The tree for the XML document was already built and stored in the database. In addition, pureXML technology uses a native XML engine that understands XQuery; therefore, there is no need to map XQuery to SQL which is what is done in other RDBMS products.

The chapter also talked about how to insert, delete, update and query XML documents using SQL/XML and XQuery. It also discussed XML indexes, XML Schema, and other features such as compression and XML Schema evolution.

15.7 Exercises

Throughout this chapter, you have seen several examples of SQL/XML and XQuery syntax and have been introduced to the DB2 Command Editor and IBM Data Studio. In this exercise, you will test your SQL/XML and XQuery knowledge while gaining experience with these tools. We will use the **mydb** database created using the **table_creation.txt** script file which was explained earlier in this chapter (*Listing 15.5*).

Procedure

1. Create the **mydb** database and load the XML data, as discussed earlier in the chapter. The file table_creation.txt is included in the accompanying file **Expressc_book_exercises_9.7.zip** under the Chapter 2 folder. Run the table_creation.txt script file from a DB2 Command Window or Linux shell as follows:

   ```
   db2 -tvf table_creation.txt
   ```

2. If the script fails in any of the steps, try to figure out the problem by reviewing the error messages. A typical problem when running the script is that you may need to change the paths of the files as they may be located in different directories. You can always drop the database and start again issuing this command from the DB2 Command Window or Linux shell:

```
db2 drop database mydb
```

3. If you receive an error while trying to drop the database because of active connections, issue this command first:

```
db2 force applications all
```

4. After successfully running the script, use the DB2 Control Center, or IBM Data Studio to verify that the **items** and **clients** tables are created and that they contain 4 and 7 rows respectively.

5. With the **mydb** database created and with the two tables loaded, you can now connect to it, and perform the queries shown in *Listings 15.7* through *15.19*

A

Appendix A – Troubleshooting

This appendix discusses how to troubleshoot problems that may be encountered when working with DB2. *Figure A.1* provides a brief overview of the actions to take should a problem arise.
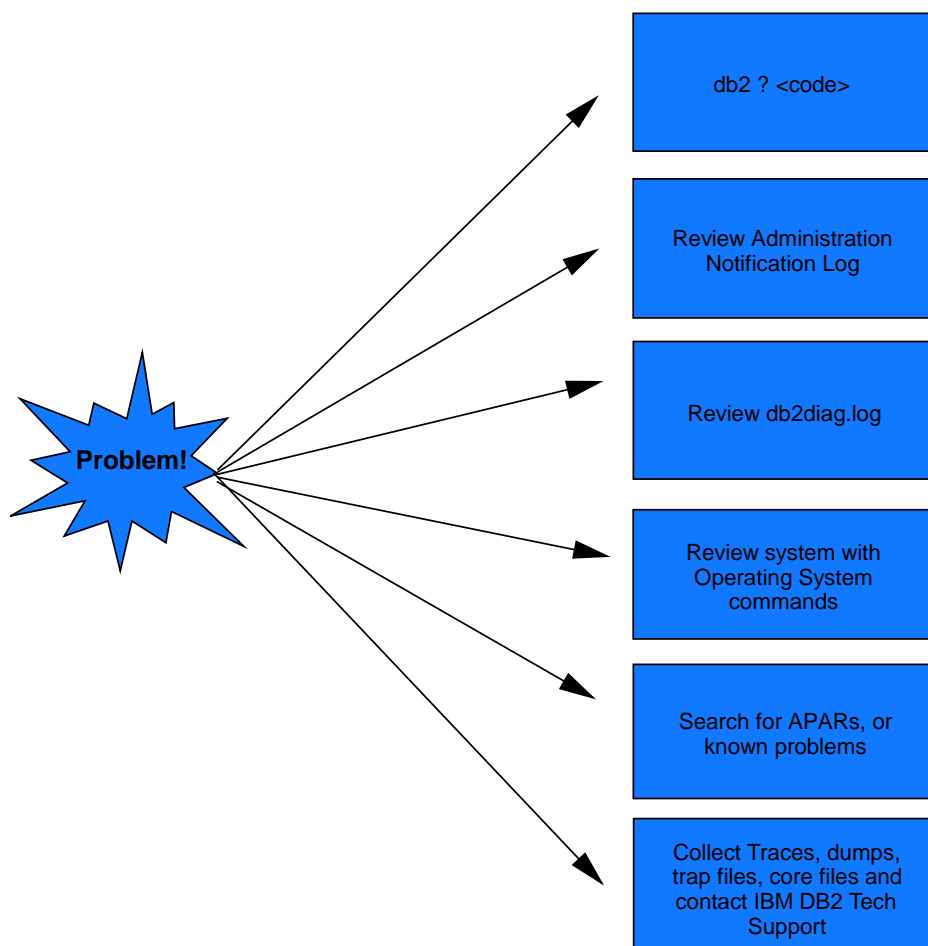

Figure A.1 – Troubleshooting overview

Note:

For more information about troubleshooting, watch this video:
http://www.channeldb2.com/video/video/show?id=807741:Video:4462

A.1 Finding more information about error codes

To obtain more information about an error code received, enter the code prefixed by a question mark in the Command Editor input area and click the *Execute* button. This is shown in *Figure A.2*.

Figure A.2 – Finding more information about DB2 error codes

The question mark (?) invokes the DB2 help command. Below are several examples of how to invoke it for help if you receive, for example, the SQL error code "-104". All of the examples below are equivalent.

```
db2 ? SQL0104N
db2 ? SQL104N
db2 ? SQL-0104
db2 ? SQL-104
db2 ? SQL-104N
```

A.2 SQLCODE and SQLSTATE

An SQLCODE is a code received after every SQL statement is executed. The meanings of the values are summarized below:

- SQLCODE = 0; the command was successful
- SQLCODE > 0; the command was successful, but returned a warning

- SQLCODE < 0; the command was unsuccessful and returned an error

The SQLSTATE is a five-character string that conforms to the ISO/ANSI SQL92 standard. The first two characters are known as the SQLSTATE class code:

- A class code of 00 means the command was successful.

- A class code of 01 implies a warning.

- A class code of 02 implies a not found condition.

- All other class codes are considered errors.

A.3 DB2 Administration Notification Log

The DB2 administration notification log provides diagnostic information about errors at the point of failure. On Linux and UNIX platforms, the administration notification log is a text file called <instance name>.nfy (e.g. "db2inst.nfy"). On Windows, all administration notification messages are written to the Windows Event Log.

The DBM configuration parameter `notifylevel` allows administrators to specify the level of information to be recorded:

- 0 -- No administration notification messages captured (not recommended)

- 1 -- Fatal or unrecoverable errors

- 2 -- Immediate action required

- 3 -- Important information, no immediate action required (the default)

- 4 -- Informational messages

A.4 db2diag.log

The db2diag.log provides more detailed information than the DB2 Administration notification log. It is normally used only by IBM DB2 technical support or experienced DBAs. Information in the db2diag.log includes:

- The DB2 code location reporting an error.

- Application identifiers that allow you to match up entries pertaining to an application on the db2diag.logs of servers and clients.

- A diagnostic message (beginning with "DIA") explaining the reason for the error.

- Any available supporting data, such as SQLCA data structures and pointers to the location of any extra dump or trap files.

On Windows (other than Vista), the db2diag.log is located by default under the directory:

```
C:\Documents and Settings\All Users\Application
Data\IBM\DB2\DB2COPY1\<instance name>
```

On Windows Vista, the db2diag.log is located by default under the directory:

```
C:\ProgramData\IBM\DB2\DB2COPY1\<instance name>
```

On Linux/UNIX, the db2diag.log is located by default under the directory:

```
/home/<instance_owner>/sqllib/db2dump
```

The verbosity of diagnostic text is determined by the dbm cfg configuration parameter DIAGLEVEL. The range is 0 to 4, where 0 is the least verbose, and 4 is the most. The default level is 3.

A.5 CLI traces

For CLI, Java, PHP, and Ruby on Rails applications, you may turn on the CLI trace facility to troubleshoot your application. This can be done by making changes to the db2cli.ini file at the server where your application is running. Typical entries in the db2cli.ini file are shown below in *Listing A.1*.

```
[common]
trace=0
tracerefreshinterval=300
tracepathname=/path/to/writeable/directory
traceflush=1
```
Listing A.1 - db2cli.ini file entries to turn on CLI Tracing

Low level tracing (db2trc) is also available, but this is typically only useful for DB2 technical support.

A.6 DB2 Defects and Fixes

Sometimes a problem you encounter may be caused by a defect in DB2. IBM regularly releases fix packs which contain code fixes for defects (APARs). The fix pack documentation contains a list of the fixes contained in the fix pack. When developing new applications, we always recommend using the latest fix pack to benefit from the latest fixes. To view your current version and fix pack level: from the Control Center, select the **About** option from the **Help** menu; or from the Command Window, type **db2level**. Note that fix packs and official IBM DB2 technical support are not offered with DB2 Express-C, With DB2 Express-C, fixes are incorporated into the image itself rather than applied as fix packs.

B

Appendix B – References and Resources

B.1 References

[1] ZIKOPOULOS, P. *IBM® DB2® Universal Database™ and the Microsoft® Excel Application Developer… for Beginners*, dbazine.com article, April 2005
http://www.dbazine.com/db2/db2-disarticles/zikopoulos15

[2] ZIKOPOULOS, P. *DB2 9 and Microsoft Access 2007 Part 1: Getting the Data...*, Database Journal article, May 2008
http://www.databasejournal.com/features/db2/article.php/3741221

[3] BHOGAL, K. *Use Microsoft Access to interact with your DB2 data*, developerWorks article, May 2006. http://www.ibm.com/developerworks/db2/library/techarticle/dm-0605bhogal/

[4] SARACCO, C. et all. IBM Redbook *DB2 9: pureXML overview and fast start*

July 2006. http://www.redbooks.ibm.com/abstracts/sg247298.html

B.2 Web sites:

1. DB2 Express-C web site: www.ibm.com/db2/express

   Use this web site to download the image for DB2 Express-C servers, DB2 clients, DB2 drivers, manuals, access to the team blog, mailing list sign up, etc.

2. DB2 Express forum:
   www.ibm.com/developerworks/forums/dw_forum.jsp?forum=805&cat=19

   Use the forum to post technical questions when you cannot find the answers in the manuals yourself.

3. DB2 Information Center

   The information center provides access to the online manuals. It is the most up to date source of information. For each version of DB2 there is a corresponding DB2 Information Center:

   - DB2 9.1: http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp

   - DB2 9.5: http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp

   - DB2 9.7: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp

4. developerWorks: http://www.ibm.com/developerworks/db2

   This Web site is an excellent resource for developers and DBAs providing access to current articles, tutorials, etc. for free.

5. alphaWorks: http://www.alphaworks.ibm.com/

   This Web site provides direct access to IBM's emerging technology. It is a place where one can find the latest technologies from IBM Research.

6. planetDB2: www.planetDB2.com

   This is a blog aggregator from many contributors who blog about DB2.

7. DB2 Technical Support: http://www.ibm.com/software/data/db2/support/db2_9/

   You can look her for defects and problem reports (APARs) and other technical information.

8. ChannelDB2: http://www.ChannelDB2.com/

   ChannelDB2 is a social network for the DB2 community. It features content such as DB2 related videos, demos, podcasts, blogs, discussions, resources, etc. for Linux, UNIX, Windows, z/OS, and i5/OS.

B.3 Books

1. Free Redbook: DB2 Express-C: The Developer Handbook for XML, PHP, C/C++, Java, and .NET
   Whei-Jen Chen, John Chun, Naomi Ngan, Rakesh Ranjan, Manoj K. Sardana, August 2006 - SG24-7301-00

http://www.redbooks.ibm.com/abstracts/sg247301.html?Open

2. Free Redbook: DB2 pureXML Guide
   Whei-Jen Chen, Art Sammartino, Dobromir Goutev, Felicity Hendricks, Ippei Komi,
   Ming-Pang Wei, Rav Ahuja, Matthias Nicola. August 2007
   http://www.redbooks.ibm.com/abstracts/sg247315.html?Open

3. Free Redbook: Developing PHP Applications for IBM Data Servers.
   Whei-Jen Chen, Holger Kirstein, Daniel Krook, Kiran H Nair, Piotr Pietrzak
   May 2006 - SG24-7218-00
   http://www.redbooks.ibm.com/abstracts/sg247218.html?Open

4. Understanding DB2 – Learning Visually with Examples V9.5
   Raul F. Chong, et all. January 2008
   ISBN-10: 0131580183

5. DB2® SQL PL: Essential Guide for DB2® UDB on Linux™, UNIX®, Windows™,
   i5/OS™, and z/OS®, 2nd Edition
   Zamil Janmohamed, Clara Liu, Drew Bradstock, Raul Chong, Michael Gao, Fraser
   McArthur, Paul Yip
   ISBN: 0-13-100772-6

6. DB2 9: pureXML overview and fast start
   Cynthia M. Saracco, Don Chamberlin, Rav Ahuja June 2006 SG24-7298
   http://www.redbooks.ibm.com/abstracts/sg247298.html?Open

7. Information on Demand - Introduction to DB2 9 New Features
   Paul Zikopoulos, George Baklarz, Chris Eaton, Leon Katsnelson
   ISBN-10: 0071487832
   ISBN-13: 978-0071487832

B.4 Contact emails

General DB2 Express-C mailbox (For administrative type of questions): db2x@ca.ibm.com

General DB2 on Campus program mailbox: db2univ@ca.ibm.com

Getting started with DB2 9.7 couldn't be easier. Read this book to:

- **Discover DB2 using its free edition - Express-C**
- **Understand DB2 architecture, tools, security**
- **Learn how to administer DB2 databases**
- **Write SQL, XQuery, stored procedures**
- **Develop database applications for DB2**
- **Practice using hands-on exercises**

DB2 Express-C from IBM is a no-charge edition of DB2 data server for managing relational and XML data with ease. No-charge means DB2 Express-C is free to download, free to build your applications, free to deploy into production, and even free to embed and distribute with your solution. And, DB2 does not place any artificial limits on the size of databases, number of databases, or number of users.

DB2 Express-C runs on Windows and Linux systems and provides application drivers for a variety of programming languages including C/C++, Java, .NET, PHP, Perl, and Ruby. Optional low-cost subscription and support with additional capabilities is available. If you require even greater scalability or more advanced functionality, you can seamlessly deploy applications built using DB2 Express-C to other DB2 editions such as Workgroup and Enterprise.

This free edition of DB2 is ideal for developers, consultants, ISVs, DBAs, students, or anyone who intends to develop, test, deploy, or distribute database applications. Join the growing DB2 Express-C user community today and take DB2 Express-C for a test drive. Start discovering how you can create next generation applications and deliver innovative solutions.

To learn more or download DB2 Express-C visit:
ibm.com/db2/express

To socialize with the DB2 community and watch DB2-related videos and blogs, visit:
ChannelDB2.com

9 780986 628351

Price: 24.99USD