

GETTING STARTED WITH  
**DB2 Express-C**

A book for the community by the community

FOREWORD BY DR. ARVIND KRISHNA

FREE TO BUILD, DEPLOY, DISTRIBUTE



**NO LIMITS.. JUST DATA!**

**First Edition (November 2007)**

**This edition applies to IBM® DB2® Express-C Version 9.1 for Linux®, UNIX® and Windows®.**

**© 2007 Copyright IBM Corporation. All rights reserved.**

---

# Contents

<b>About this book.....</b>	<b>8</b>
Notices and Trademarks.....	8
Who should read this book? .....	9
How is this book structured?.....	9
A book for the community by the community.....	9
Authors and Contributors.....	10
Acknowledgements.....	10
Foreword.....	11
<b>PART I – OVERVIEW AND SETUP .....</b>	<b>12</b>
<b>Chapter 1 – What is DB2 Express-C? .....</b>	<b>13</b>
1.1 Free to develop, deploy, and distribute...no limits!.....	13
1.2 Technical support.....	14
1.3 DB2 servers .....	14
1.4 DB2 clients.....	15
1.5 Application development freedom .....	16
1.6 DB2 versions versus DB2 editions .....	17
1.7 Moving up to another DB2 edition .....	17
1.8 Maintenance of DB2 Express-C .....	18
1.9 Related free software.....	18
1.9.1 DB2 Developer Workbench .....	18
1.9.2 DB2 9 Embedded Application Server .....	19
1.9.3 DB2 9 Net Search Extender .....	19
1.9.4 Starter Toolkit for DB2 on Rails .....	19
1.9.5 Web 2.0 Starter Toolkit for DB2.....	19
<b>Chapter 2 – Related features and products .....</b>	<b>21</b>
2.1 Features included with DB2 Express-C subscription.....	22
2.1.1 Fixpacks.....	22
2.1.2 High Availability Disaster Recovery (HADR) .....	22
2.1.3 Data Replication .....	23
2.2 Features not available with DB2 Express-C .....	24
2.2.1 Database Partitioning .....	24
2.2.2 Connection Concentrator.....	24
2.2.3 Geodetic Extender .....	24
2.2.4 Query Patroller.....	24
2.3 Fee-based products that are related to DB2 .....	25
2.3.1 DB2 Connect .....	25
2.3.2 WebSphere Federation Server .....	25
2.3.3 WebSphere Replication Server .....	26
<b>Chapter 3 – DB2 installation .....</b>	<b>27</b>
3.1 Installation prerequisites .....	27

## 4 Getting Started with DB2 Express-C

---

3.2 Operating system installation authority.....	27
3.3 Installation wizard .....	28
3.4 Silent Install.....	32
QuickLab #1: Install DB2 Express-C & create SAMPLE database .....	34
<b>Chapter 4 – DB2 Tools.....</b>	<b>37</b>
4.1 Control Center.....	38
4.2 Command Editor .....	42
4.3 SQL Assist Wizard.....	44
4.4 Show SQL Button .....	45
Quicklab #2 – Create a New Database with Control Center .....	46
4.5 Scripting .....	48
4.5.1 SQL scripts .....	48
4.5.2 Operating system (shell) scripts .....	50
Quicklab #3 – Create an installation script for EXPRESS Database ...	51
4.6 Task Center .....	54
4.6.1 The Tools Catalog database .....	54
4.7 Journal .....	56
4.8 Health Monitor.....	57
4.8.1 Health Center.....	58
<b>Chapter 5 – DB2 Environment .....</b>	<b>61</b>
5.1 DB2 configuration .....	70
5.1.1 Environment variables .....	71
5.1.2 Database manager configuration file (dbm cfg) .....	71
5.1.3 Database configuration file (db cfg).....	73
5.1.4 DB2 profile registry .....	74
5.2 The DB2 Administration Server .....	75
<b>PART II – LEARNING DB2: DATABASE ADMINISTRATION .....</b>	<b>76</b>
<b>Chapter 6 – DB2 Architecture .....</b>	<b>77</b>
6.1 DB2 process model .....	77
6.2 DB2 memory model.....	78
6.3 DB2 storage model.....	79
6.3.1 Pages and Extents.....	79
6.3.2 Buffer pools.....	79
6.3.3 Table spaces .....	81
<b>Chapter 7 – DB2 Client Connectivity .....</b>	<b>87</b>
7.1 Configuration Assistant.....	87
7.1.1 Setup required at the server .....	88
7.1.2 Setup required at the client.....	90
7.1.3 Creating Client and Server Profiles .....	94
Quicklab #4 – Using the Configuration Assistant .....	97
<b>Chapter 8 – Working with Database Objects .....</b>	<b>101</b>
8.1 Schema.....	101
8.2 Tables .....	101

---

8.2.1 Data Types.....	102
8.2.2 Identity Columns.....	104
8.2.3 SEQUENCE objects.....	105
8.2.4 System catalog tables.....	106
8.2.5 Declared temporary tables.....	106
Quicklab #5 – Creating a new table.....	108
8.3 Views.....	111
8.4 Indexes.....	111
8.4.1 Design Advisor.....	111
8.5 Referential integrity.....	113
<b>Chapter 9 – Data Movement Utilities.....</b>	<b>115</b>
9.1 EXPORT utility.....	116
9.2 IMPORT utility.....	117
9.3 LOAD.....	118
9.4 The db2move utility.....	119
9.5 The db2look utility.....	119
Quicklab #6 – Extracting DDL for the EXPRESS database.....	122
<b>Chapter 10 – Database Security.....</b>	<b>125</b>
10.1 Authentication.....	126
10.2 Authorization.....	127
10.3 DBADM authority.....	130
10.4 The PUBLIC group.....	130
10.5 The GRANT and REVOKE statements.....	131
10.6 Authorization and privilege checking.....	131
10.7 Group privilege considerations.....	133
Quicklab #7 – Granting and revoking user permissions.....	134
<b>Chapter 11 – Backup and Recovery.....</b>	<b>137</b>
11.1 Database Logging.....	137
11.2 Types of logs.....	138
11.3 Types of logging.....	138
11.3.1 Circular logging.....	138
11.3.2 Archival logging or log retain.....	139
11.4 Database logging from the Control Center.....	140
11.5 Logging parameters.....	141
11.6 Database backup.....	142
Quicklab #8 – Scheduling a backup.....	144
11.7 Database recovery.....	147
11.7.1 Recovery types.....	147
11.7.2 Database restore.....	147
11.8 Other operations with BACKUP and RESTORE.....	148
<b>Chapter 12 – Maintenance Tasks.....</b>	<b>149</b>
12.1 REORG, RUNSTATS, REBIND.....	149
12.1.1 The REORG command.....	150
12.1.2 The RUNSTATS command.....	150
12.1.3 BIND / REBIND.....	150

12.1.4 Maintenance tasks from the Control Center .....	152
12.2 Maintenance Choices .....	153
Quicklab #9 – Configuring automated maintenance.....	156
<b>Chapter 13 – Concurrency and Locking .....</b>	<b>159</b>
13.1 Transactions .....	159
13.2 Concurrency.....	160
13.3 Problems without concurrency control.....	161
13.3.1 Lost update .....	161
13.3.2 Uncommitted read .....	162
13.3.3 Non-repeatable read.....	162
13.3.4 Phantom read .....	163
13.4 Isolation Levels .....	164
13.4.1 Uncommitted read .....	164
13.4.2 Cursor stability .....	164
13.4.3 Read stability .....	165
13.4.4 Repeatable read .....	165
13.4.5 Comparing isolation levels.....	165
13.4.6 Setting the isolation level.....	166
13.5 Lock escalation .....	167
13.6 Lock monitoring .....	168
13.7 Lock wait.....	169
13.8 Deadlock causes and detection.....	169
13.9 Concurrency and locking best practices .....	170
<b>PART III – LEARNING DB2: APPLICATION DEVELOPMENT.....</b>	<b>173</b>
<b>Chapter 14 – SQL PL Stored Procedures .....</b>	<b>175</b>
14.1 The DB2 Developer Workbench .....	176
14.1.2 Create a stored procedure in the DWB .....	177
14.2 SQL PL stored procedures basics.....	180
14.2.1 Stored procedure structure .....	180
14.2.2 Optional stored procedure attributes .....	181
14.2.3 Parameters .....	181
14.2.4 Comments in an SQL PL stored procedure.....	182
14.2.5 Compound statements.....	182
14.2.6 Variable declaration .....	183
14.2.7 Assignment statements .....	183
14.3 Cursors .....	183
14.4 Flow control.....	184
14.5 Calling stored procedures.....	184
14.6 Errors and condition handlers.....	186
14.7 Dynamic SQL.....	188
<b>Chapter 15 – Inline SQL PL, UDFs, Triggers .....</b>	<b>189</b>
15.1 Inline SQL PL.....	189
15.2 Triggers.....	190
15.2.1 Types of triggers .....	190
Quicklab #10 – Creating a trigger in the Control Center.....	194

---

15.3 User-defined functions (UDFs) .....	197
15.3.1 Scalar functions .....	197
15.3.2 Table functions .....	198
Quicklab #11 – Creating a UDF using the DB2 Developer Workbench	200
<b>Chapter 16 – SQL/XML and XQuery .....</b>	<b>203</b>
16.1 Using XML with databases .....	204
16.2 XML databases .....	204
16.2.1 XML-enabled databases.....	204
16.2.2 Native XML databases.....	205
16.3 XML in DB2.....	205
16.3.1 DB2 9 pureXML technology advantages .....	206
16.3.2 XPath basics .....	208
16.3.3 XQuery defined.....	211
16.3.4 Inserting XML documents .....	213
16.3.5 Querying XML data .....	216
16.3.6 Joins with SQL/XML .....	222
16.3.7 Joins with XQuery .....	222
16.3.8 Update and delete operations .....	223
16.3.9 XML indexing.....	224
QuickLab #12 - SQL/XML and XQuery.....	226
<b>Chapter 17 –Developing with Java, PHP, and Ruby.....</b>	<b>227</b>
17.1 Application development in Java .....	227
17.1.1 JDBC Type 2 driver .....	227
17.1.2 JDBC Type 4 driver .....	228
17.2 Application development in PHP .....	229
17.2.1 DB2 connection options for PHP .....	229
17.2.2 Zend Core for IBM .....	231
17.3 Application development in Ruby on Rails .....	233
17.3.1 Startup Toolkit for DB2 on Rails .....	233
<b>Appendix A – Troubleshooting.....</b>	<b>235</b>
A.1 Obtaining more information about error codes.....	236
A.2 SQLCODE and SQLSTATE .....	236
A.3 DB2 Administration Notification Log.....	237
A.4 db2diag.log .....	237
A.5 CLI traces .....	238
A.6 DB2 Defects and Fixes.....	238
<b>Resources.....</b>	<b>239</b>
Web sites: .....	239
Books .....	240

---

## About this book

### Notices and Trademarks

© Copyright IBM Corporation 2007  
All Rights Reserved.  
IBM Canada  
8200 Warden Avenue  
Markham, ON  
L6G 1C7  
Canada

Neither this documentation nor any part of it may be copied or reproduced in any form or by any means or translated into another language, without the prior consent of all of the above mentioned copyright owners.

IBM makes no warranties or representations with respect to the content hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. IBM assumes no responsibility for any errors that may appear in this document. The information contained in this document is subject to change without any notice. IBM reserves the right to make any such changes without obligation to notify any person of such revision or changes. IBM makes no commitment to keep the information contained herein up to date.

The information in this document concerning non-IBM products was obtained from the supplier(s) of those products. IBM has not tested such products and cannot confirm the accuracy of the performance, compatibility or any other claims related to non-IBM products. Questions about the capabilities of non-IBM products should be addressed to the supplier(s) of those products.

IBM, the IBM logo, DB2, DB2 Connect, DB2 Universal Database, i5/OS, pureXML, WebSphere, and z/OS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.



## 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, covers stored procedures, user defined functions, triggers, SQL/XML, XQuery, development in Java™, PHP and Ruby.

The Appendix contains useful information about troubleshooting.

Exercises called “Quicklabs” are provided with most chapters; any input files required for these labs are provided in the zip file `expressc_book_quicklabs.zip` accompanying this book, or provided in the IBM® DB2 Express-C Web site: [www.ibm.com/db2/express](http://www.ibm.com/db2/express).

The materials in this book are also used in courses offered as part of the “DB2 on Campus” Program. This book can prepare you to pass the DB2 on Campus exam which provides you with a program completion confirmation that acknowledges receiving 16 hours worth of DB2 training. You can read more about this program at the DB2 Express-C website [www.ibm.com/db2/express/students.html](http://www.ibm.com/db2/express/students.html).

## A book for the community by the community

The initial edition of this book was created by the DB2 Express-C team and released to the DB2 Express-C community at no-charge. Writing and maintaining a book is an arduous task. Our goal is that the content of this book is maintained and enhanced through the DB2 Express-C community. If you would like to provide feedback, contribute new material, improve existing material, or help with translating this book to another language, please send an email of your planned contribution to [db2x@ca.ibm.com](mailto:db2x@ca.ibm.com) with the subject “DB2 Express-C book changes.”

---

## Authors and Contributors

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

Contributor's Name	Company	Job Title	Material contributed	Date
Raul F. Chong	IBM	DB2 on Campus Program Manager	Initial version of all chapters of the book	May 2007
Ian Hakes	IBM	DB2 Express-C Community Facilitator	Complete review and edit of the book	May – June 2007
Rav Ahuja	IBM	DB2 Product Manager	Review, update, edit, layout, and formatting of the entire book	July - Oct 2007

## 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 for designing the cover of this book.
- Susan Visser for assistance with publishing this book.

## Foreword

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

However DB2 is not just for large enterprises anymore. With the release of DB2 Express-C, the award-winning DB2 technology is now available to meet the needs of small and mid-size companies – and that with no mandatory cost! Although there are other free or open-source data servers out there, DB2 Express-C offers unique advantages over these alternatives.

There are many technological innovations present in DB2 Express-C. Some of these innovations are aimed at advanced new capability, some at reducing administrative burdens, some at improving performance, and some at reducing infrastructure cost. We will not discuss most of these here, hoping you will be tempted to read the book – but we'll briefly describe one as a teaser.

DB2 Express-C is built on 'Viper' technology, making it the first hybrid data server for managing both relational and XML data in their native formats. This makes DB2 ideal for powering a new breed of SOA and Web 2.0 applications where XML data flows in abundance. Unlike data servers from other commercial vendors, 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 gained with the help of this book are very relevant to 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 with this book being developed by DB2 Express-C community members and becoming freely available to anyone in the community. I very much encourage you to enrich and update this book with your know-how, experiences, and assist with translating this book into other languages so others can benefit from your knowledge.



Arvind Krishna  
Vice President, Data Servers  
Information Management, IBM Software Group

## **PART I – OVERVIEW AND SETUP**

# 1

## Chapter 1 – What is DB2 Express-C?

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, such as:

- 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 optimized for systems with up to 2 processors, and 4 GB of memory, and does not have any specialized storage or system setup requirements. DB2 Express-C also includes pureXML at no charge. pureXML is DB2's unique technology to store and process XML documents natively.

### 1.1 Free to develop, deploy, and distribute...no limits!

This sentence summarizes the key ideas of 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 database 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.
- **No limits:** While other competitor database offerings set limits on database sizes, with DB2 Express-C there are NO data limits. Your database can grow and grow without violating the licensing agreement. There are also no limits in terms of the number of connections or users per server.

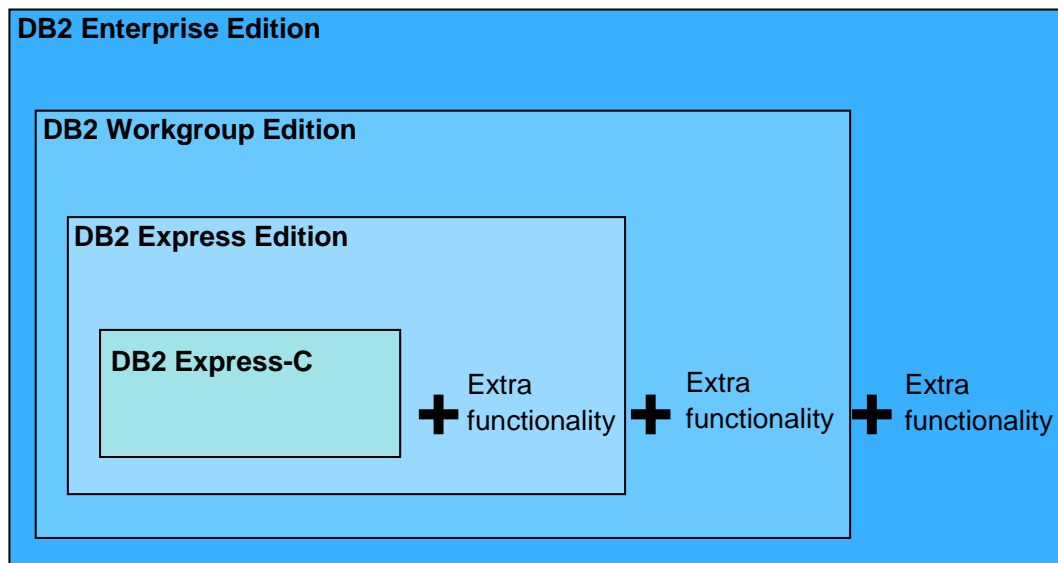
## 1.2 Technical support

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 a dedicated DB2 Express-C team, 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 yearly subscription (also known as 12 Months License and Subscription or Fixed Term License or FTL). This subscription for DB2 Express-C comes with the backing of IBM for 24 x 7 technical support and software updates. For a yearly renewable low cost fee (\$2,995 per Server per Year in the United States – may vary in other countries) you not only get support and software maintenance for your DB2 Express-C server, but you also get to use two key add-on features: HADR (High Availability Disaster Recovery), and SQL replication (for replicating data with other DB2 servers).

## 1.3 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 by Figure 1.1, DB2 Express-C is the same as DB2 Express without a few components. DB2 Express-C is free to the community. As mentioned earlier, 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 12 month subscription license.

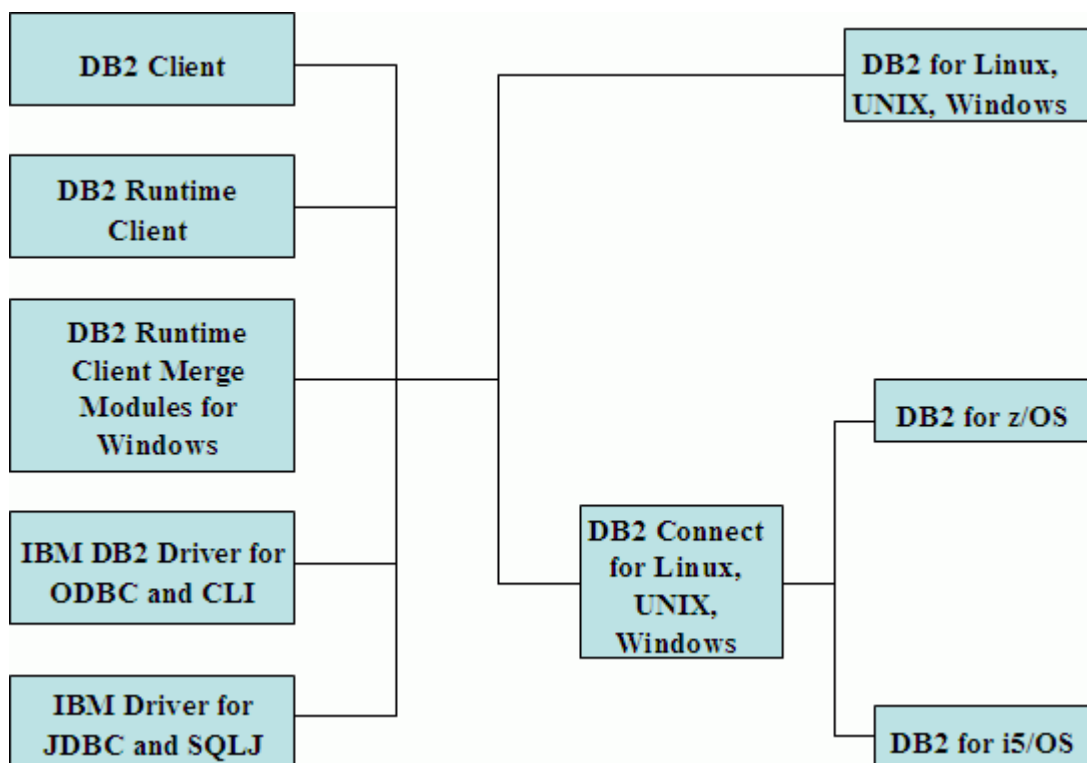
Figure 1.1 also explains why it is so easy to upgrade from DB2 Express-C. If you wish to upgrade to any of the other DB2 servers in the future, all DB2 servers have the same core components. This also means that any application developed for one edition will work, without modification, in other edition.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

## 1.4 DB2 clients

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 can connect directly to a DB2 server, provided that the correct driver is loaded. DB2 Clients come in several different flavors:

- DB2 client: most complete, includes GUI Tools, drivers.
- DB2 runtime client: basic functionality to connect, and includes drivers
- DB2 runtime client Merge Modules for Windows: mainly used to include a DB2 runtime client as part of a Windows application installation

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, with DB2 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 will need to go through a DB2 Connect server (shown in the middle of Figure 1.2). We will discuss the DB2 Connect software in Chapter 2.

## 1.5 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 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 are:

- Ruby on Rails
- C/C++ (ODBC and embedded SQL)



- JDBC and SQLJ
- COBOL
- Borland
- Python
- PHP
- Perl
- .NET languages
- OLE-DB
- ADO
- Web services
- SQL
- MS Office: Excel, Access, Word

## 1.6 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 8 and Version 9 are officially supported by IBM. A Version may also have a few Releases which can include some new functionality but usually not significant enough to warrant a new Version. For example 8.1 and 8.2 are Release levels for DB2 Version 8. Going by the past history, IBM seems to come out with a new Release of DB2 almost every year, however new Versions are typically spaced 2-3 years apart. The most current release, V9.1 (previously code-named as DB2 'Viper') became Generally Available (GA) in the summer of 2006. At the time of writing (September 2007) the next release, code-named DB2 'Viper 2' is available as a public beta. Each release may also have several Modification levels, which typically contain fixes or correspond to Fixpack levels, and seldom contain new functionality. At the time of writing the most current Version, Release, Modification (V,R,M) level of DB2 Express-C is 9.1.2 which corresponds to a code-level of 9.1 with Fixpack 2.

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 (also known as DB2 9) has several editions; for example, DB2 Express-C 9, DB2 Express 9, DB2 Workgroup 9, and DB2 Enterprise 9 (see Figure 1.1).

## 1.7 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 to another DB2 edition on the same computer system, install the new DB2 edition on top of DB2 Express-C, and the corresponding new license. Your databases will not be deleted (but a backup is always recommended)
- If you are upgrading DB2 where the new edition will be installed on a different, larger computer using the same operating system, install the new DB2 edition on the

larger computer, backup your databases from the smaller computer, move the backup images to the larger computer, and restore from the backup images the databases on the larger computer. You may 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 application will not need modification.

## 1.8 Maintenance of DB2 Express-C

As discussed earlier, there are two support options for DB2 Express-C:

1. Buy the 12 month subscription license. This provides you with full time coverage from IBM DB2 Technical support, gives you the ability to install DB2 software updates (also called fixpacks).
2. Use the online DB2 Express-C community forum. This is totally free, but comes with no official support from IBM. Also, under this option, IBM does not commit to provide new features and bug fixes at scheduled dates. The concept of a fixpack, which is discussed in Chapter 2, does not apply either; instead, refreshes of the entire DB2 Express-C image are made available from time to time. As new releases come out, you can typically expect refreshed DB2 Express-C images to be available for the new releases rather than the older ones.

## 1.9 Related free software

All the software that is available for download from the DB2 Express-C download page ([www.ibm.com/db2/express/download.html](http://www.ibm.com/db2/express/download.html)) is free of charge. Besides the images for DB2 Express-C (for Linux and Windows, both 32 and 64-bit architectures), there is other useful software that can be downloaded and used for free:

- DB2 Developer Workbench
- DB2 9 Embedded Application Server
- DB2 9 Net Search Extender

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

- Starter Toolkit for DB2 on Rails
- Web 2.0 Starter Toolkit for DB2

### 1.9.1 DB2 Developer Workbench

The DB2 Developer Workbench (DWB) is a powerful integrated development environment (IDE) that allows you to create, edit, debug, deploy and test Java and SQL PL stored procedures, as well as user-defined functions (UDFs), SQLJ applications and the creation and execution of SQL statements and XML queries. It is based on the Eclipse IDE, and re-

places the IBM Development Center which was available in previous version of DB2. We will discuss the DWB in Chapter 14, SQL PL stored procedures.

### **1.9.2 DB2 9 Embedded Application Server**

The DB2 Embedded Application Server enables you to run the web applications supplied with DB2 Version 9.1, without requiring you to purchase a separate application server. The Web applications supplied with DB2 Version 9.1 are:

- DB2 web tools, for web-based database administration
- DB2WebServices, an application that automates the deployment of .NET web services from Microsoft Visual Studio to the DB2 Embedded Application Server.

### **1.9.3 DB2 9 Net Search Extender**

With DB2 9 Net Search Extender, you can execute fast and detailed full-text searches in text documents, including any XML documents stored natively in DB2 9.

### **1.9.4 Starter Toolkit for DB2 on Rails**

Starter Toolkit for DB2 on Rails is a conveniently-packaged set of products and technologies that enables the quick creation of an environment to build DB2 Web applications using Ruby on Rails technology. All required software is included: DB2 Express-C; DB2 driver for Ruby; DB2 adapter for Rails; along with tutorials, examples, and other learning materials. We will discuss Ruby on Rails further in Chapter 17, Development in Java, PHP and Ruby.

### **1.9.5 Web 2.0 Starter Toolkit for DB2**

Web 2.0 Starter Toolkit for DB2 is an easy way to get started with DB2, PHP, and Dojo. It helps you deploy the required software, links you to tutorials, and includes demo applications. Two of the demo applications are the Atom Feed Control Panel, which generates Atom feeds from DB2 tables, and the Web Services Control Panel, which creates REST web service wrappers around your DB2 tables. Both rely on Dojo for its significant Ajax and widget capabilities.



# 2

## Chapter 2 – Related features and products

This chapter describes DB2 features included with the purchase of a DB2 Express-C 12 month subscription license. It also describes other features **not** available with the DB2 Express-C edition, but as part of other DB2 editions, in some cases, for an additional fee.

Features available with the DB2 Express-C 12 month subscription license are:

- Fixpacks
- High Availability and Disaster Recovery (HADR )
- Data Replication (Homogenous SQL)

Features not available with DB2 Express-C, but with other DB2 editions are:

Chargeable DB2 Enterprise Edition Features

- Database Partitioning Feature (DPF)
- Storage Optimization Feature (includes compression)
- Advanced Access Control (fine grained and advanced security)
- Performance Optimization (Performance Expert, Query Patroller)
- Geodetic Data Management
- DB2 Homogenous Federation

DB2 Enterprise Edition also contains additional no-charge functionality, such as:

- Table (Range) Partitioning
- Materialized Query Tables (MQT)

- Multi-dimensional Clustering (MDC)
- High Availability and Disaster Recovery (HADR )
- Connection Concentrator

#### Chargeable DB2 Workgroup and Express Edition Features

- High Availability
- Workload Management (Connection Concentrator, Query Patroller)
- Performance Optimization (MQT, MDC, Query Parallelism)
- DB2 Homogenous Federation

#### Fee-based products related to DB2:

- DB2 Connect
- WebSphere® Federation Server
- WebSphere Replication Server

## **2.1 Features included with DB2 Express-C subscription**

This section outlines DB2 Fixpacks, HADR and SQL replication.

### **2.1.1 Fixpacks**

A DB2 Fixpack 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, Fixpacks are free to download and install. They are typically available every three months.

To download the latest Fixpack, review the DB2 technical support site at [http://www.ibm.com/software/data/db2/support/db2\\_9/](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 pri-

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

### 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.

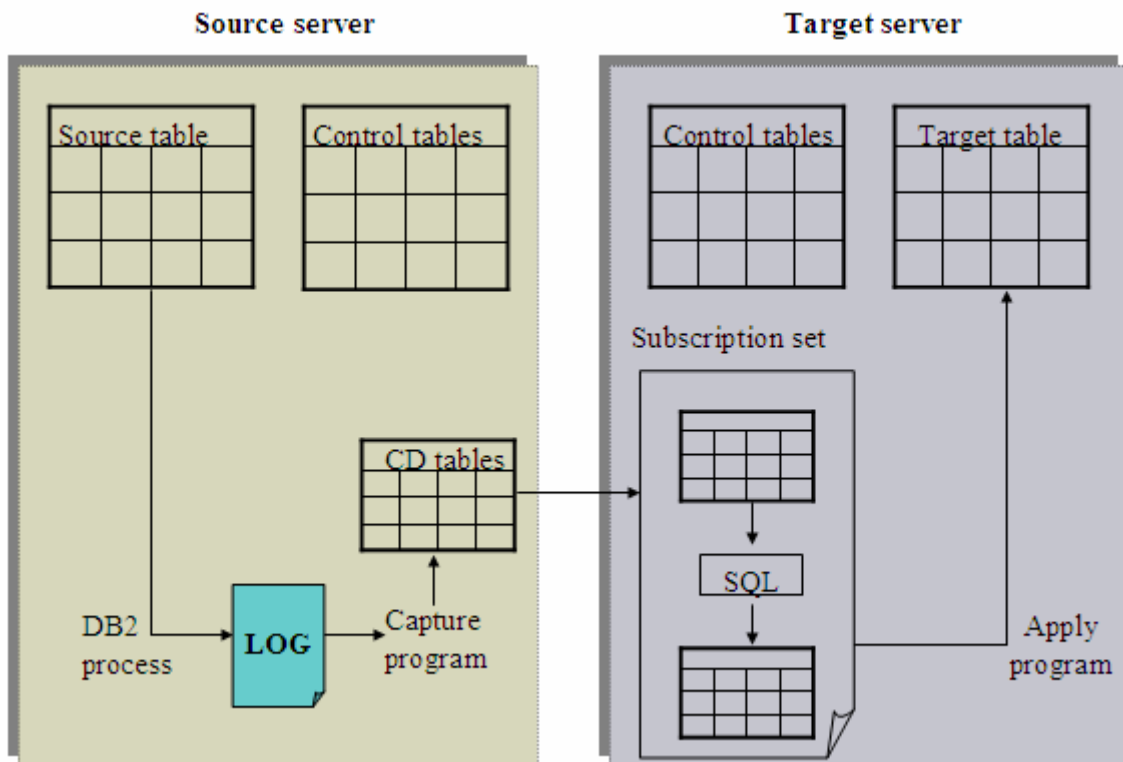


Figure 2.1 –SQL Replication

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-C and other DB2 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.

### **2.2.1 Database Partitioning**

The database partitioning feature (DPF) is only available with DB2 Enterprise Edition for an additional license fee. It allows databases to be spread across multiple partitions which can reside in several computers. DPF is based on a shared-nothing architecture. Each computer, as it is added to the partition group, brings additional data processing power with its own CPUs and memory. DPF is particularly useful in large data server environments like data warehouses where decision support systems (DSS) queries are run.

### **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 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 Spatial 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 Query Patroller**

DB2 Query Patroller is a powerful query management system that can control the flow of queries against your DB2 database. It allows you to regulate your database query workload so that small queries and high-priority queries can run promptly, ensuring that your system resources are used efficiently.



## 2.3 Fee-based products that are related to 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 2.2. 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.

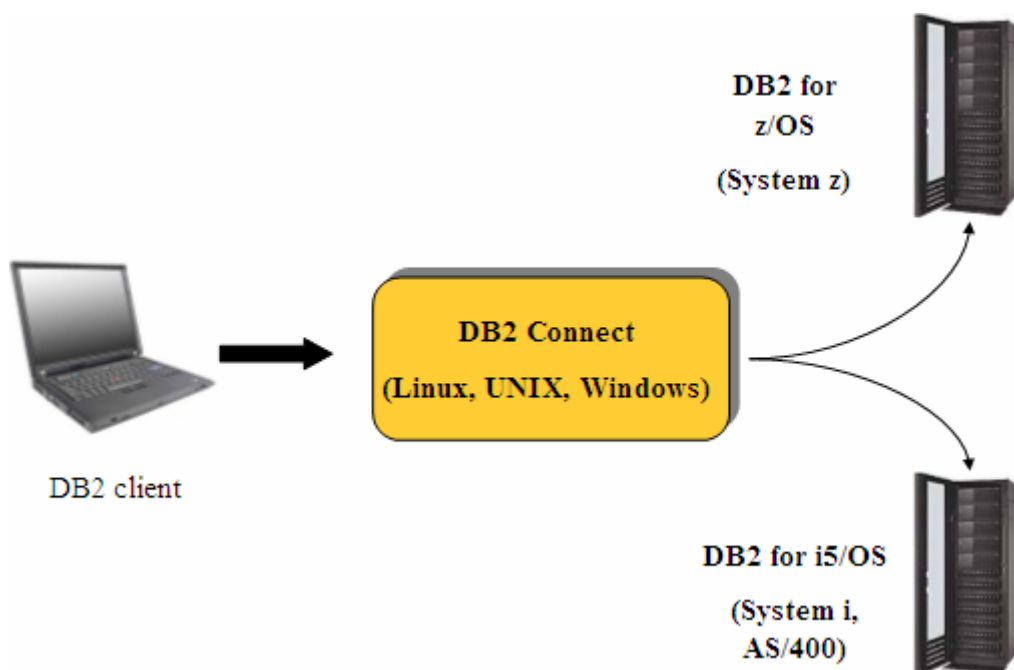


Figure 2.2 – DB2 Connect

### 2.3.2 WebSphere 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 following query:

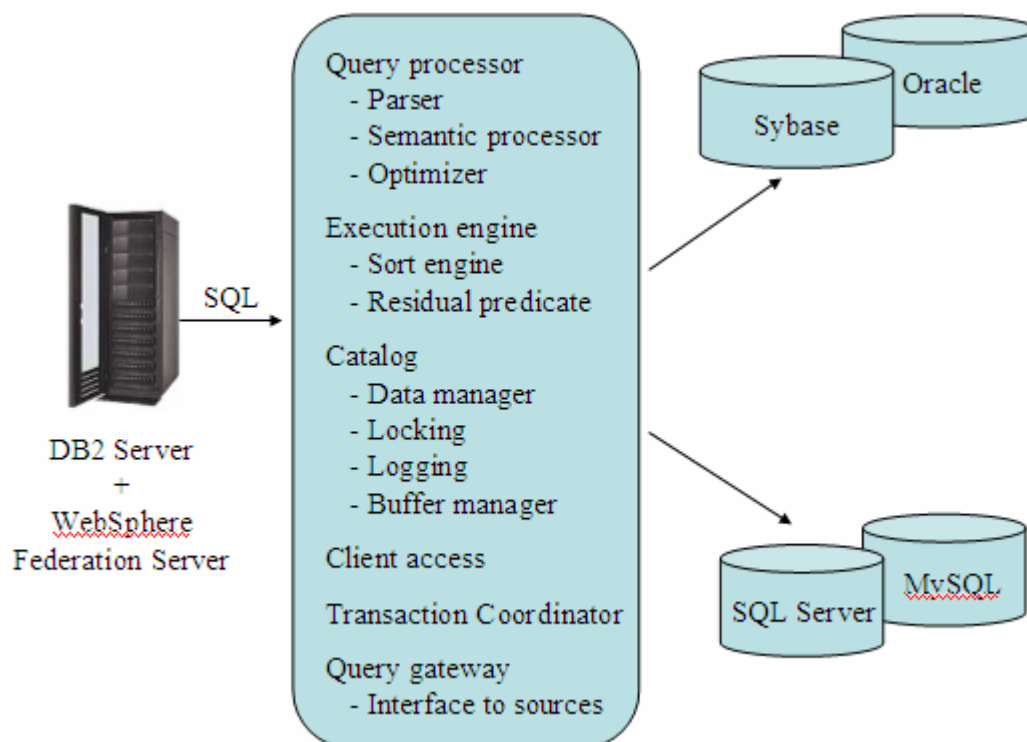
```
SELECT *
FROM   Oracle.Table1 A
       DB2.Table2 B
```

```

SQLServer.Table3 C
WHERE
    A.col1 < 100
    and B.col5 = 1000
    and C.col2 = 'Test'

```

Figure 2.3 provides an illustration where WebSphere Federation Server is used.



**Figure 2.3 – WebSphere 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 WebSphere Replication Server

Formerly known as WebSphere Information Integrator (for replication support), the WebSphere 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.

# 3

## Chapter 3 – DB2 installation

To install the DB2 Express-C edition in either Linux or Windows, ensure that your systems satisfy the installation pre-requisites.

### 3.1 Installation prerequisites

With respect to operating system version and level requirements, DB2 Express-C is available on Linux and Windows 2000, XP, and Vista. 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:

<http://www.ibm.com/software/data/db2/udb/sysreqs.html>

In terms of hardware resources, DB2 Express-C is optimized for systems with up to 2 physical CPUs and 4GB RAM. 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 uni-processor system with 1GB of memory.

For the latest information on DB2 Express-C hardware requirements, review the DB2 Express-C web page:

<http://www-306.ibm.com/software/data/db2/udb/db2express/getstarted.html>

### 3.2 Operating system installation authority

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

For **Linux**, you need to be root (the superuser) to install DB2 Express-C

For **Windows**, the user account must belong to the Administrators group on the machine where you will perform the installation. Alternatively, a non-Administrator user account can be used, provided that a member of the Windows Administrators group first configures the Windows elevation privileges settings to allow a non-Administrator user account to perform an installation.

For Windows domain accounts, to verify user IDs on the DB2 server, the installation user ID must belong to the Domain Administrators group on the domain where the accounts are going to be created. You may also use the built-in Local System account to run the installation for all products.

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

### 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 unzipping the DB2 Express-C image, you can launch the wizard to handle the installation:

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

DB2 Express-C is very easy to install by following the instructions of 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 several times, and click the "Finish" button. After a few minutes, your installation will be complete and DB2 is up and running!

Figure 3.1 shows the DB2 Setup Launchpad where you should choose "Install New" to install a new copy of DB2 Express-C in your system.

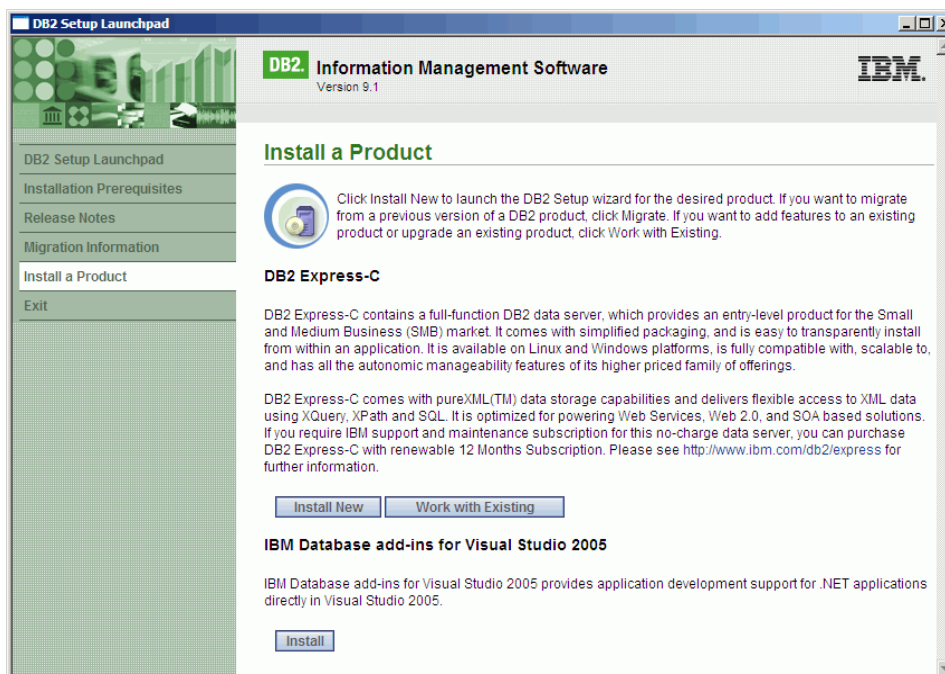
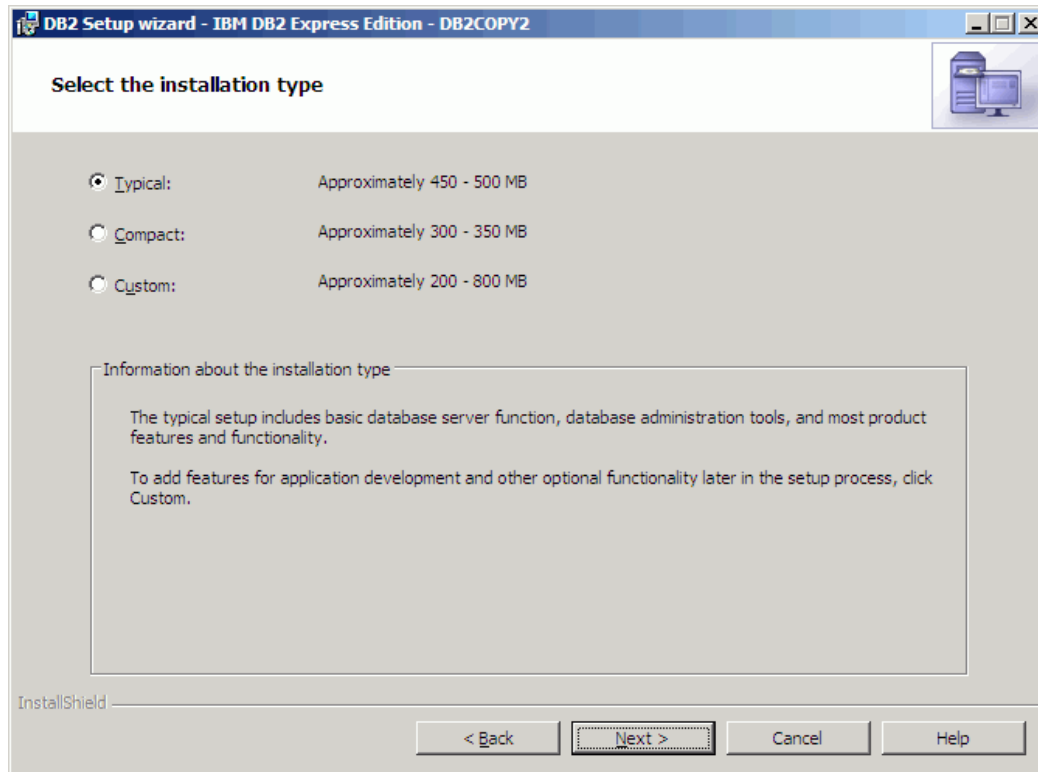


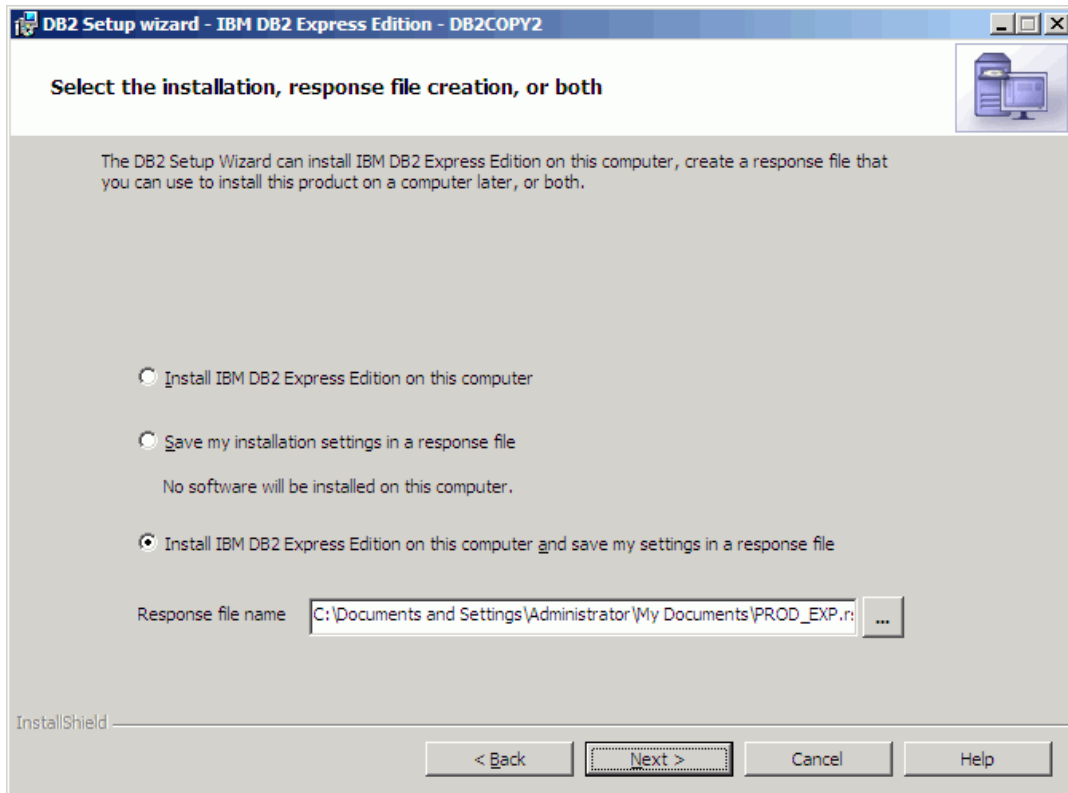
Figure 3.1 – The DB2 Setup Launchpad

After accepting the license, it is usually sufficient to choose the “Typical” installation (default) as shown in Figure 3.2.



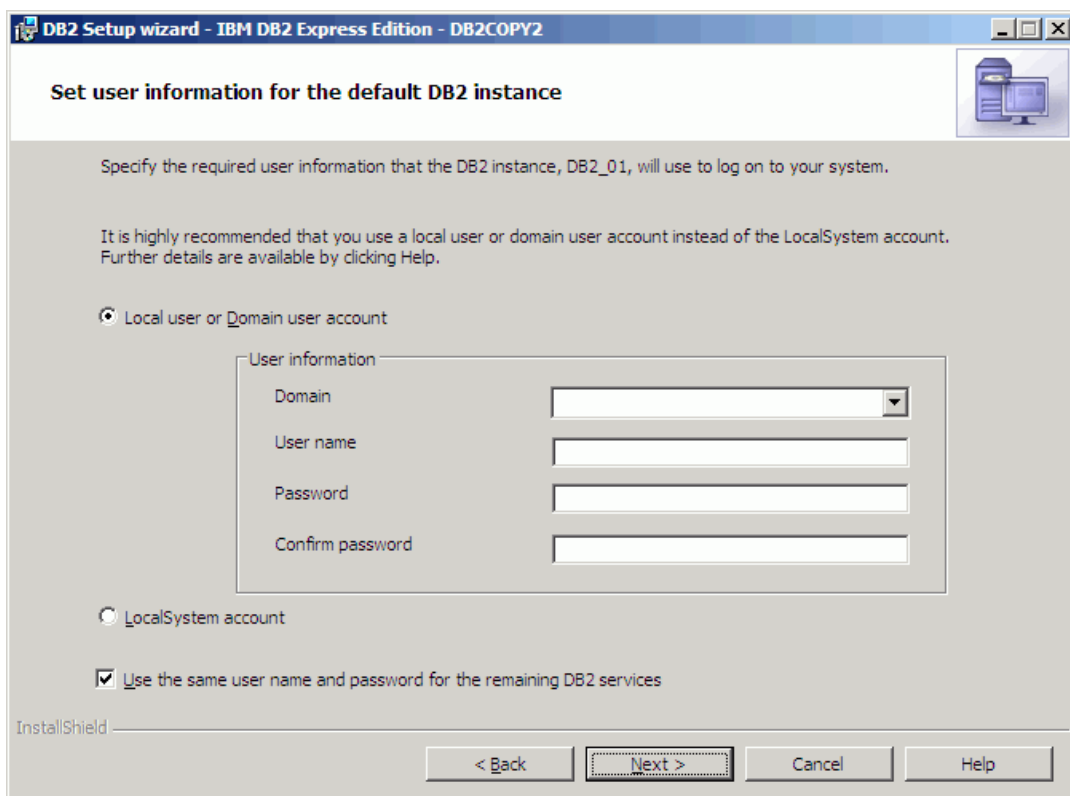
**Figure 3.2 – Installation types**

In Figure 3.3, you have the choice 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 good enough.



**Figure 3.3 – Selecting the installation**

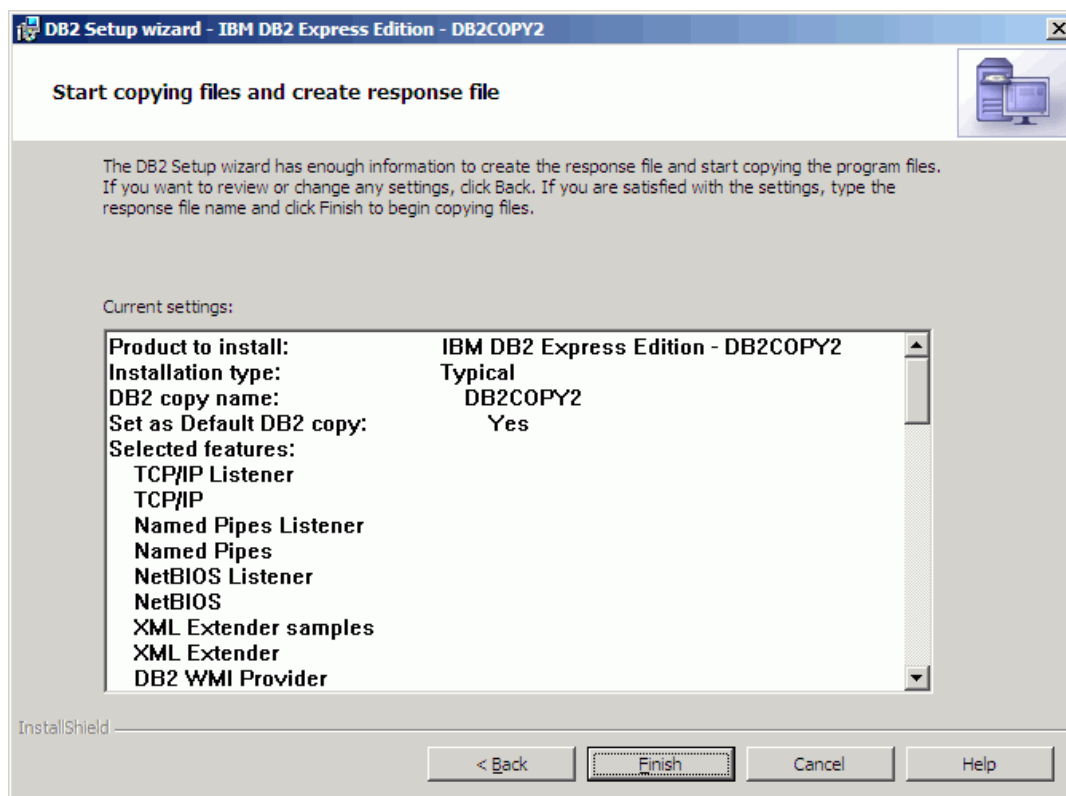
Choose default values for the next few screens. When you get to the window shown in Figure 3.4, you can input an existing user who will be used to work with the instance and other services. This user must be part of the Local Administrator group in Windows. If the user ID you input does not belong to 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 to create in Windows is called db2admin. In the case of Linux, the default user ID created is called db2inst1.



The screenshot shows a Windows-style dialog box titled "DB2 Setup wizard - IBM DB2 Express Edition - DB2COPY2". The main heading is "Set user information for the default DB2 instance". Below the heading, there is instructional text: "Specify the required user information that the DB2 instance, DB2\_01, will use to log on to your system." and "It is highly recommended that you use a local user or domain user account instead of the LocalSystem account. Further details are available by clicking Help." There are two radio button options: "Local user or Domain user account" (which is selected) and "LocalSystem account". Below the selected option is a "User information" section with four input fields: "Domain" (a dropdown menu), "User name", "Password", and "Confirm password". At the bottom, there is a checked checkbox for "Use the same user name and password for the remaining DB2 services". The bottom of the window features four buttons: "< Back", "Next >" (highlighted with a dashed border), "Cancel", and "Help". The "InstallShield" logo is visible in the bottom left corner.

Figure 3.4 – Specifying user information for the default DB2 instance

Finally, in Figure 3.5, the installation wizard displays a summary of what will be installed, and the different configuration information provided in the previous steps. When you click “Finish”, installation will start, and the program files will be laid down on your system.



**Figure 3.5 – Summary of what will be installed**

### 3.4 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 text response files which store installation information. The following shows a snippet of a sample response file.

Sample response file code snippet

```
PROD = UDB_EXPRESS_EDITION
```

```
LIC_AGREEMENT=ACCEPT
```

```
FILE = C:\Program Files\IBM\SQLLIB\
```



```
INSTALL_TYPE = CUSTOM
COMP=TCPIP_DB2_LISTENER_SUPPORT
COMP=TCPIP_DB2_CLIENT_SUPPORT
COMP=DB2_SAMPLE_DATABASE
COMP=DB2_SAMPLE_APPLICATIONS
COMP=OLE_DB_SUPPORT
COMP=ODBC_SUPPORT
COMP = JDBC_SUPPORT
COMP = IBM_JRE
...
```

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 allows you to select the checkbox to save your install responses to a response file. At the end of the wizard, a response file will be generated in a directory and filename that you specify. This is a text file, so you can edit it afterwards. This was shown in Figure 3.3.
- ▶ Edit the sample response file packaged with the DB2 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>`

Then to install DB2 using a response file, for Windows you issue the command:

```
setup -u <response filename>
```

For Linux you issue the command:

```
db2setup -r <response filename>
```

## QuickLab #1: Install DB2 Express-C & create SAMPLE database

Note: If you are using DB2 Express-C V9.1.2 or higher, the DB2 sample database may already be created for you automatically if you installed using the graphical setup wizard.

### 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 Quicklab, you will perform a basic installation of DB2 Express-C on Windows.

### Procedure

1. **Obtain DB2 Express-C images.** Download the appropriate DB2 Express-C image, or order the Discovery Kit DVD with the images from the DB2 Express-C Web site ([ibm.com/db2/express](http://ibm.com/db2/express)). Unzip the files into any directory you wish.
2. **Locate files.** Navigate to the directory (or drive) containing the unzipped DB2 product installation files.
3. **Run Launchpad.** Launch the DB2 Launchpad by double-clicking on the `setup.exe` file. From the Launchpad, click the *Install Product* option on the left pane of the window.
4. **Run DB2 setup wizard.** The DB2 setup wizard checks that all system requirements are met and sees if there are any existing DB2 installations. Click the *Next* button to continue with the installation.
5. **Review license agreement.** Read and accept the license agreement (select the “*I Accept...*” radio button) and click the *Next* button to continue.
6. **Choose 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 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):

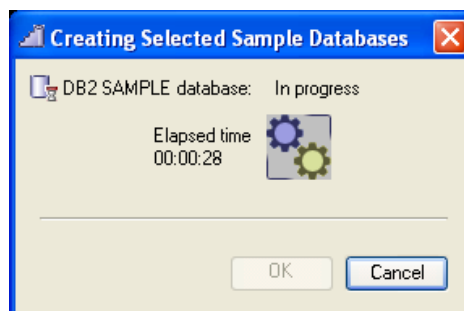
**Drive:** C:

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

Click the *Next* button to continue.

8. **Set 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. Click the *Next* button to continue.
9. **Configure 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 be 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 *Protocols* and *Startup* buttons, respectively. We recommend using the default settings in this example. Click the *Next* button to continue.
10. **Start installation.** Review the installation options previously selected. Click the *Install* button to begin copying the files to the installation location. DB2 will also perform some initial configuration processes.
11. **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`.
12. From First Steps, choose the tab that says “Database Creation”, and then follow the wizard to create the SAMPLE database. Ensure the *XML and SQL objects and data* option is chosen, and click OK.
13. The following progress screen is displayed while the database is being created. (This procedure may take several minutes).



14. When database creation is complete, click the *OK* button and close the First Steps tool. Go back to Control Center and verify that a database called SAMPLE now appears in the Object Tree pane. You may have to refresh Control Center view to see the new changes. To do this, select the *Refresh* item from the Control Center *View* menu.
15. **Restart the computer.** 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. This is **OPTIONAL**.

# 4

## Chapter 4 – DB2 Tools

In this chapter, we describe some of the tools you can use with DB2. The left side of Figure 4.1 shows the different DB2 commands, SQL, SQL/XML and XQuery statements that can be created to interact with a DB2 data server. The middle of the figure (highlighted by the red ellipse) shows the names of the different tools to interact with a DB2 data server. The right side of the figure shows the basic DB2 environment consisting of an instance, a database, and the associated configuration files.

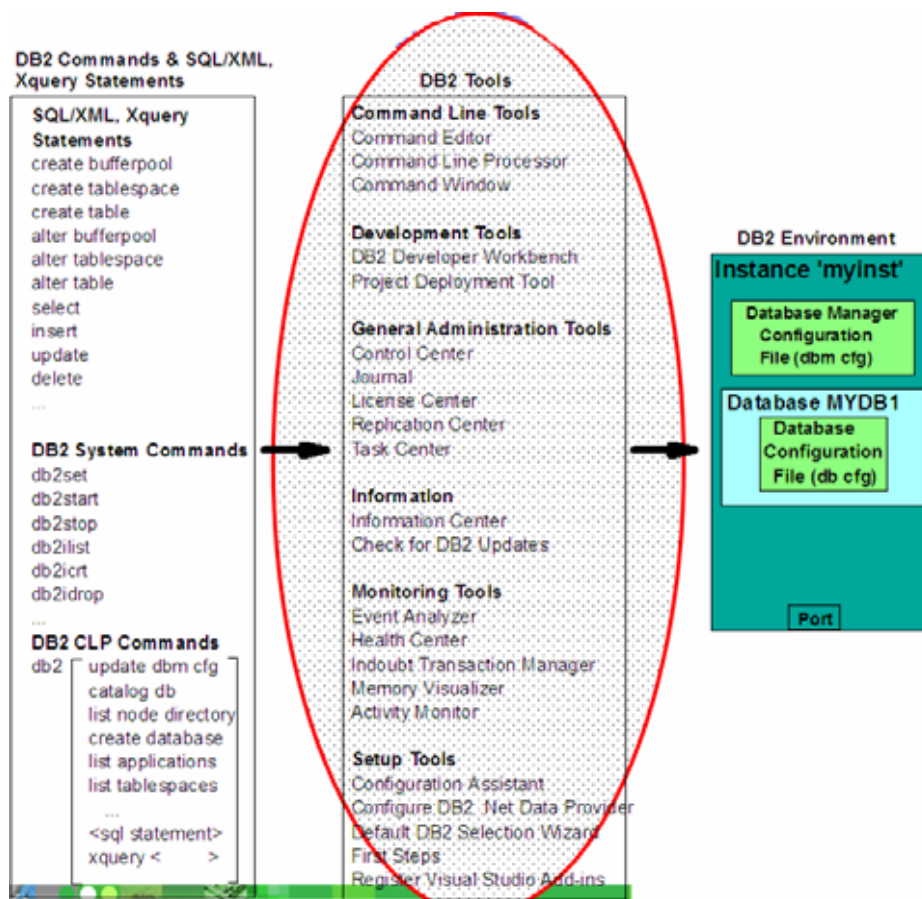
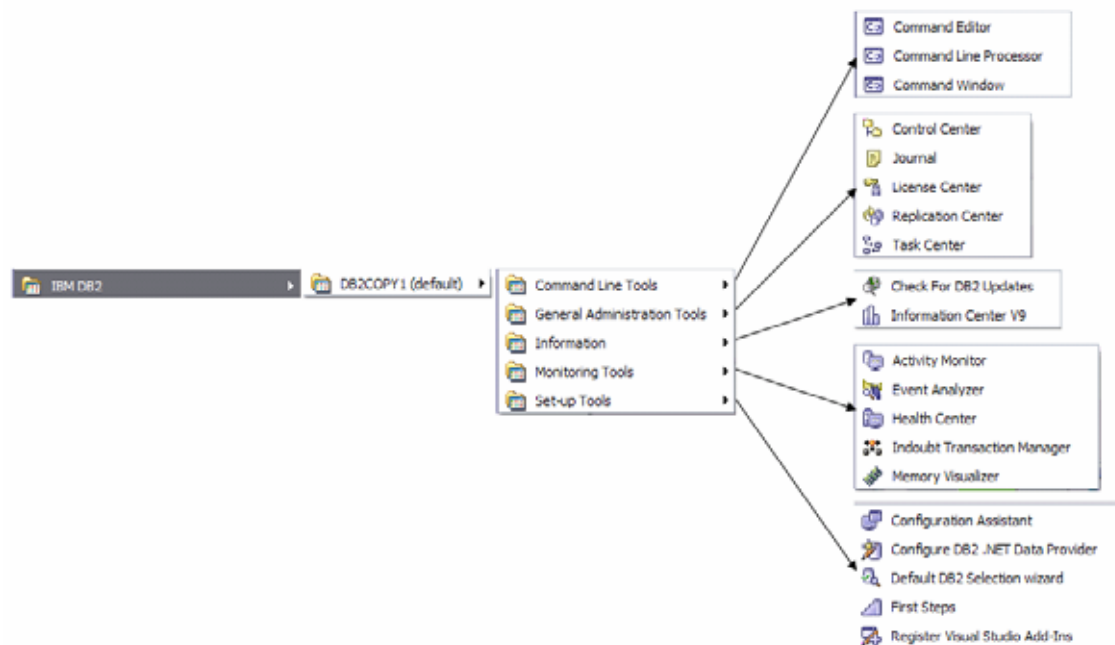


Figure 4.1 – The DB2 big picture: DB2 tools

Figure 4.2 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 4.2 – DB2 tools from the IBM DB2 Start menu**

Table 4.1 provides a list of shortcut commands that can be used to start some of the most popular tools in either Linux or Windows.

Tool name	Command
Command Editor	db2ce
Command Line processor	db2
Command Window (Only on Windows platforms)	db2cmd
Control Center	db2cc
Task Center	db2tc
Health Center	db2hc
Configuration Assistant	db2ca
First Steps	db2fs

**Table 4.1 – Shortcut commands to start some DB2 tools**

### 4.1 Control Center

The primary DB2 tool for database administration is the Control Center, illustrated in Figure 4.3.

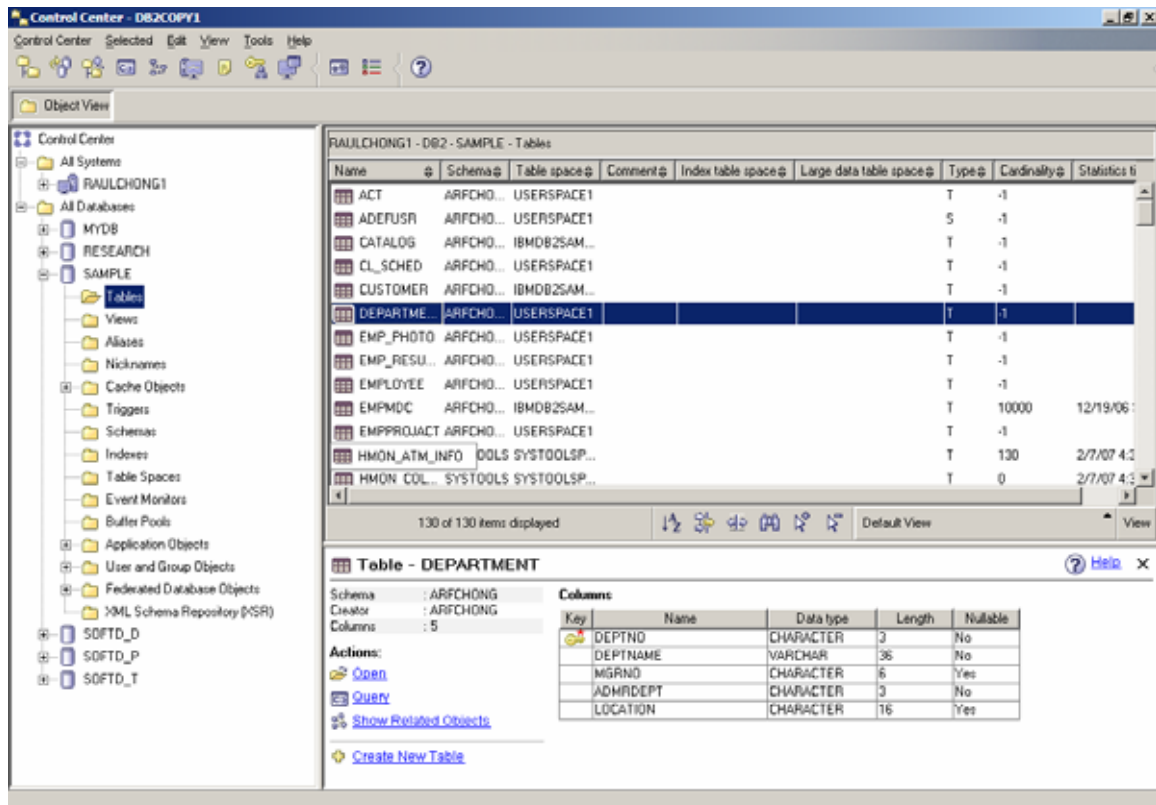


Figure 4.3 - The DB2 Control Center

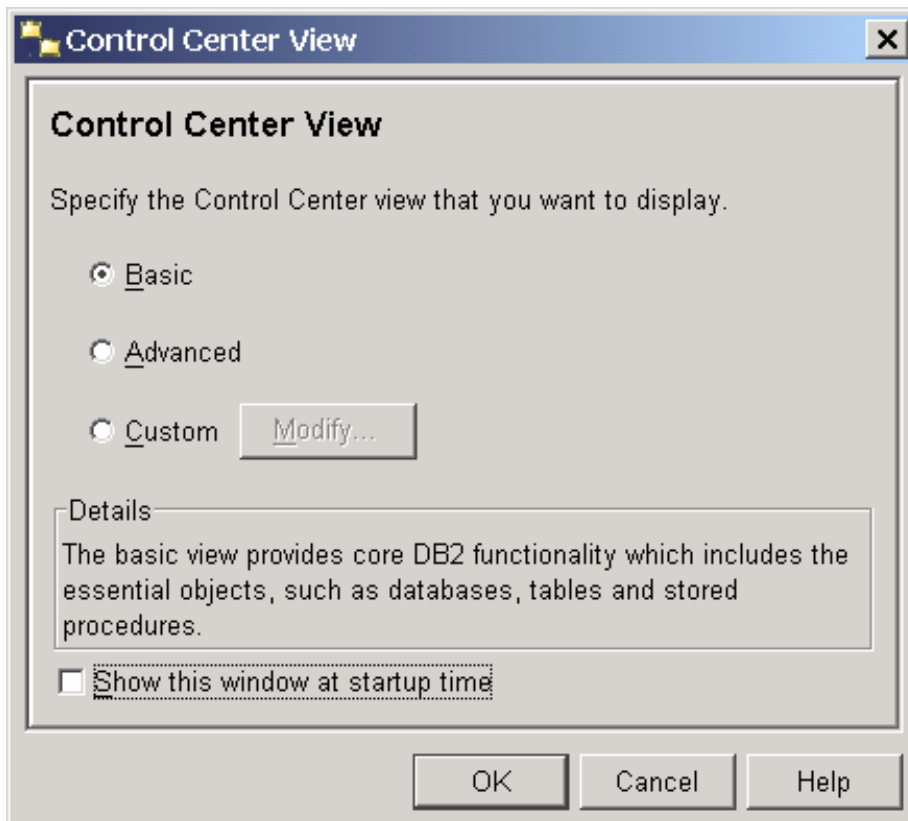
The Control Center is a centralized administration tool that allows you to:

- 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 4.3), 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/objects in the Object tree brings up menus applicable to the given folder/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 4.4 shows the Control Center View dialog box.



**Figure 4.4 - 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 the “Customize Control Center” option from the Tools menu as shown in Figure 4.5.



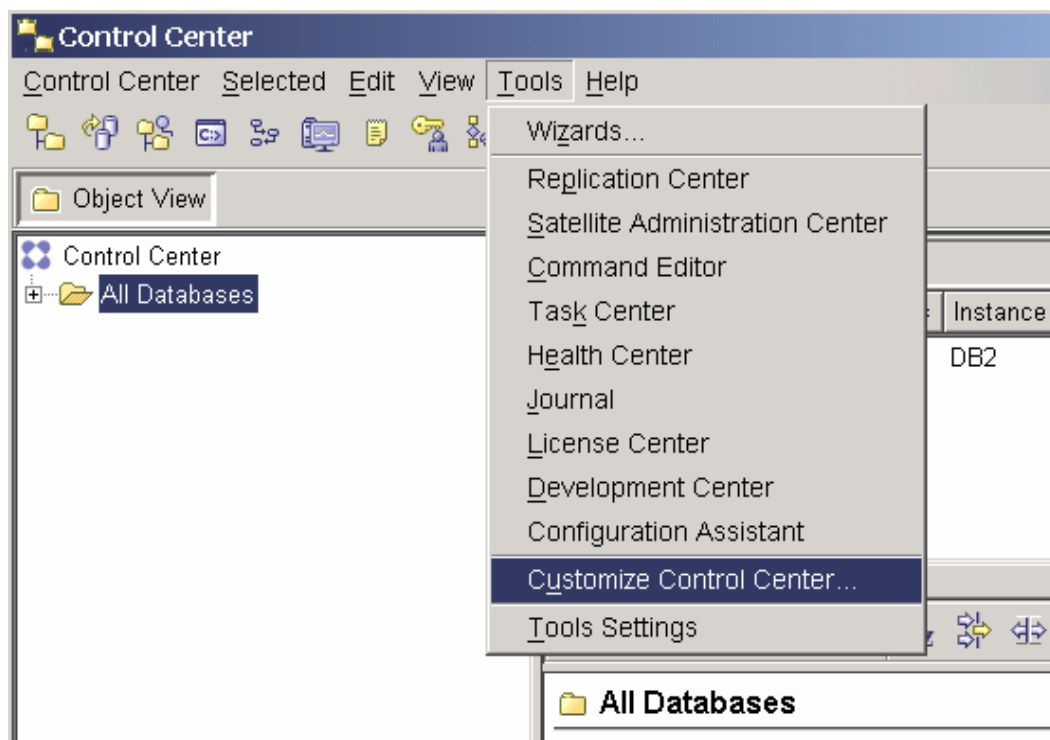


Figure 4.5 – Customizing the Control Center

#### 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  in the toolbar of any of the other DB2 GUI tools
- From the DB2 icon in the Windows system tray as shown in Figure 4.6 (Right click on the DB2 green icon and select the DB2 Control Center menu option)

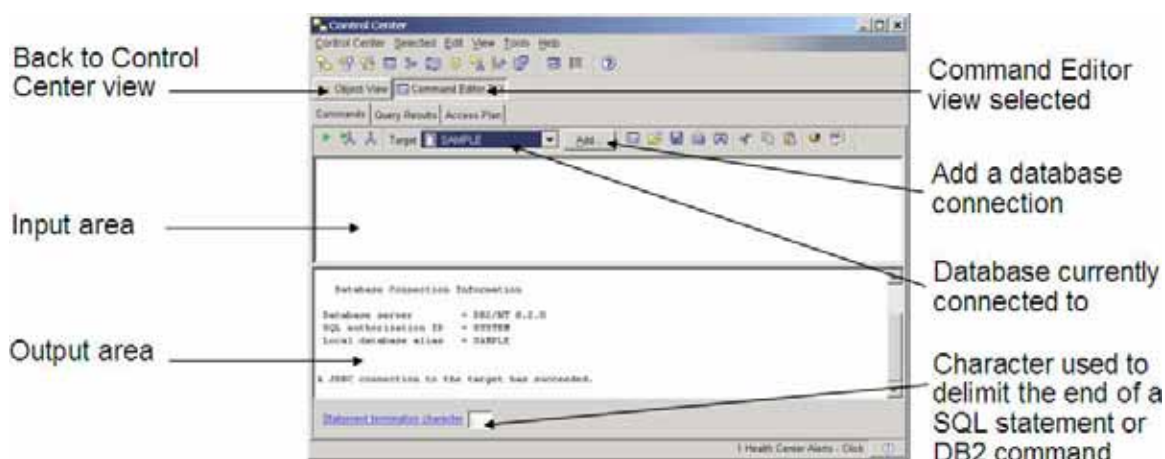


Figure 4.6 – Launching the DB2 Control Center from the Windows system tray

## 4.2 Command Editor

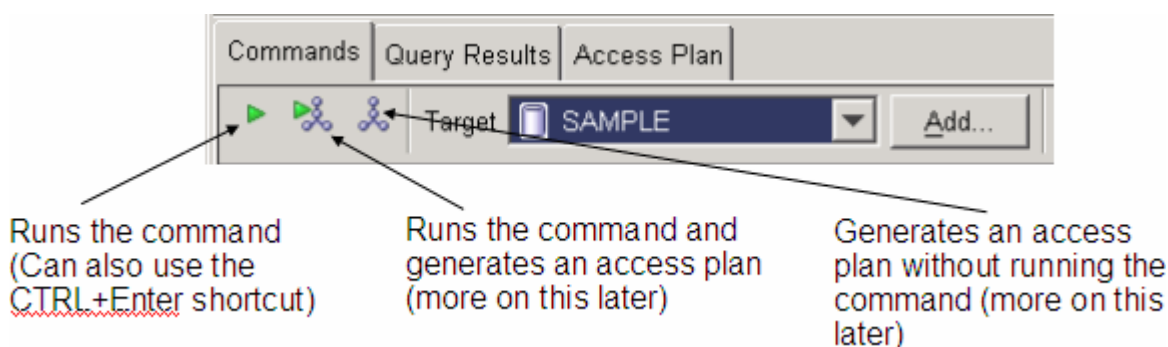
Using the DB2 Command Editor, you can execute DB2 commands, SQL and XQuery statements, analyze the execution plan of a statement, and view or update query result sets.

Figure 4.7 shows the Command Editor with a description of its elements.



**Figure 4.7 – 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 (see Figure 4.8), 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.

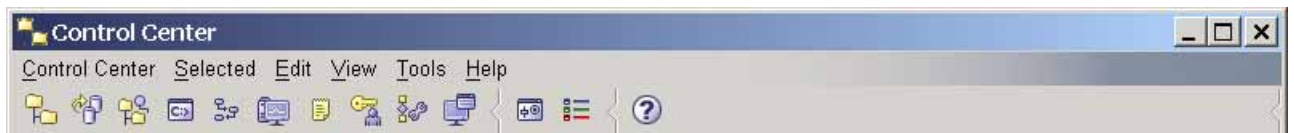


**Figure 4.8 – The Command Editor – Commands tab**

### Launching the Command Editor

You can launch the Command Editor in several ways:

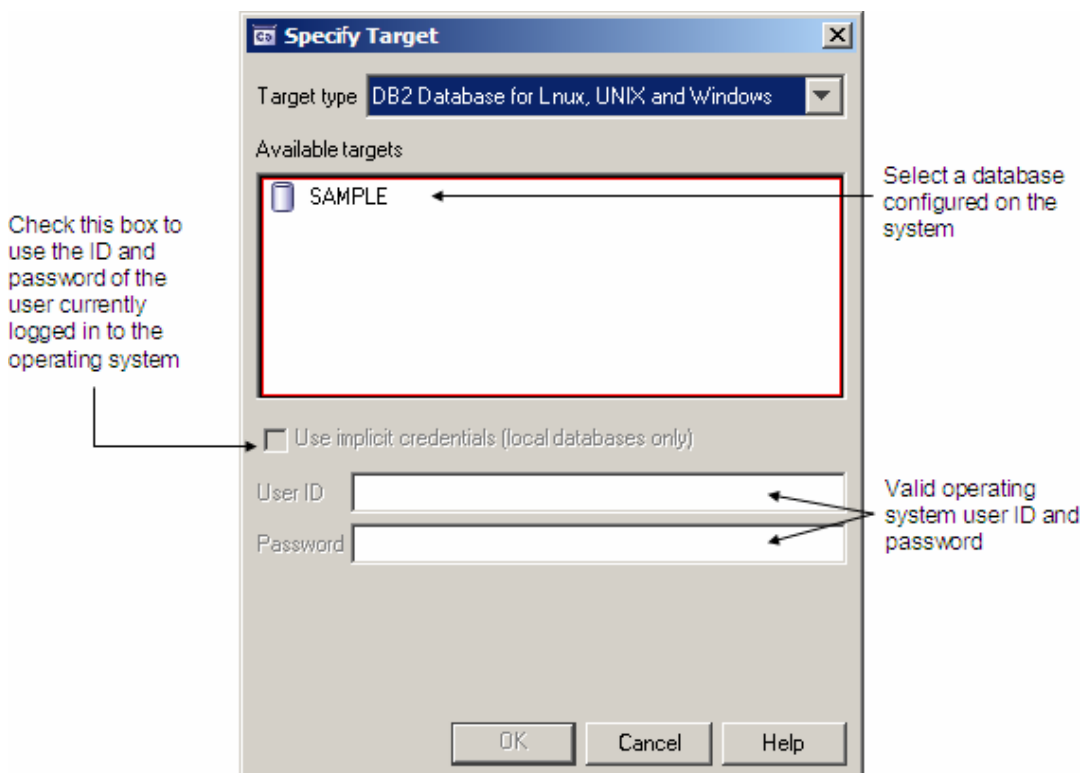
- ▶ From the Windows Start Menu:  
*Start -> Programs -> IBM DB2 -> Command Line Tools -> Command Editor*
- ▶ From a command prompt, type `db2ce`
- ▶ From the Tools menu in the Control Center
- ▶ Embedded within the Control Center
  - Right click on the SAMPLE database icon in the Control Center's Object 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  on the Control Center Toolbar as shown in Figure 4.9



**Figure 4.9 – The Command Editor icon in Control Center**

### Adding a database connection

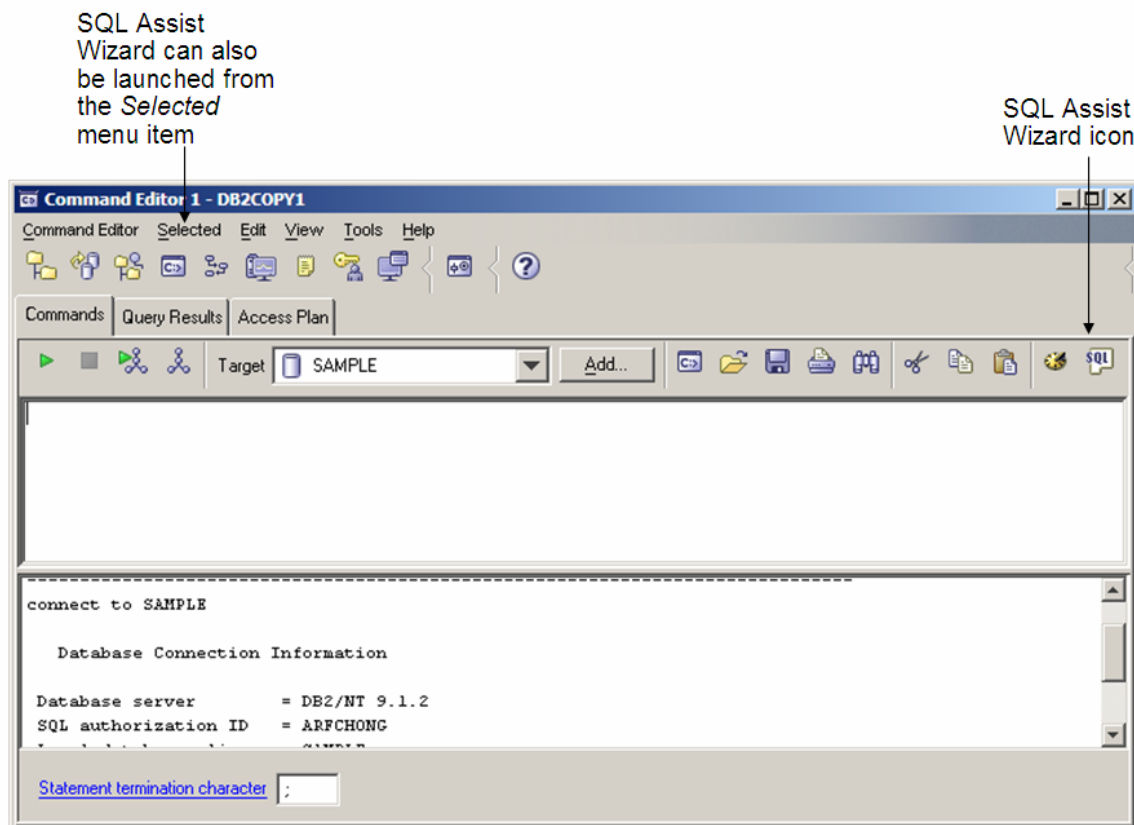
To add a connection to a database, click on the Add button. A dialog as show in Figure 4.10 will appear.



**Figure 4.10 – Add a database connection**

### 4.3 SQL Assist Wizard

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 4.11, you invoke it from the Command Editor by clicking on the last icon with the SQL symbol (highlighted in the figure with a red circle)



**Figure 4.11 – Invoking the SQL Assist Wizard**

Figure 4.12 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.

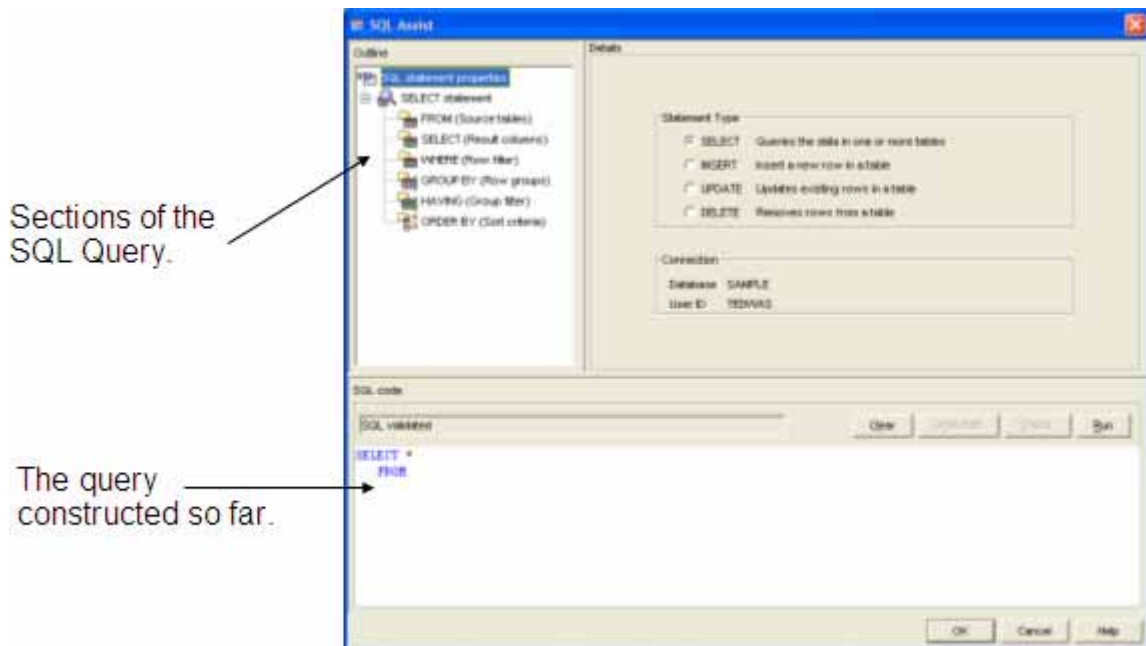


Figure 4.12 – The SQL Assist wizard

#### 4.4 Show SQL Button

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 4.13 and Figure 4.14



Figure 4.13 – The Show SQL button



Figure 4.14 – 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.

## Quicklab #2 – Create a New Database with Control Center

### Objective

In this Quicklab, you will create a new database using the Create Database wizard in the Control Center.

### Procedure

1. From the Control Center Object Tree pane, right-click the *All Databases* folder, select the *Create Database* item, and choose the *With Automatic Maintenance* item. 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 the Enable database for XML checkbox, and then click on the *Next* button to continue to the next page of the wizard.

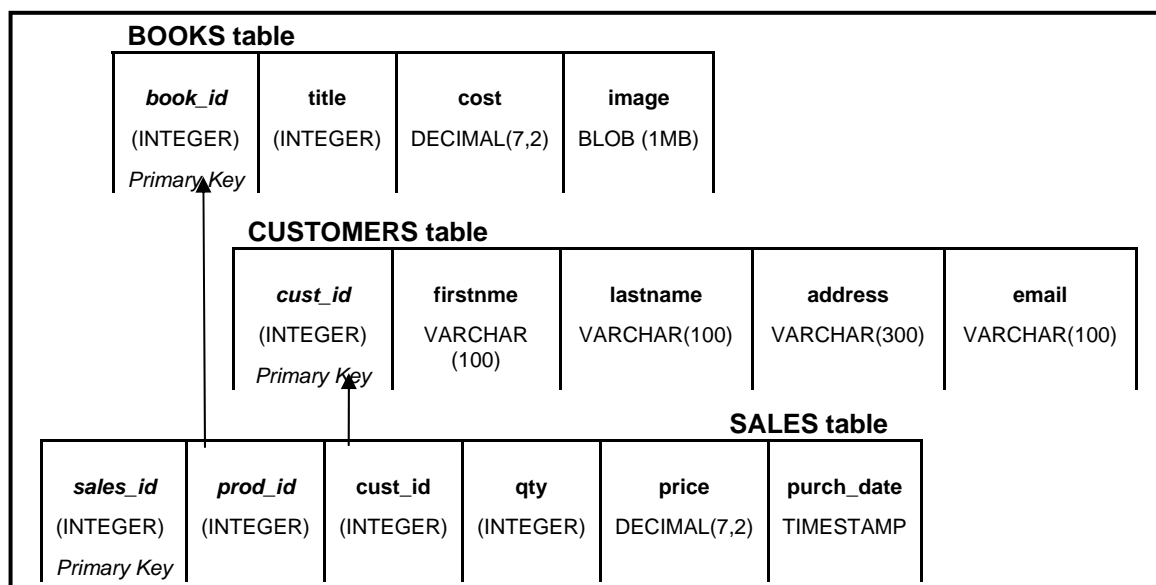
3. In the *Specify where to store your data* page, don't make any changes, and click *Next*.
4. In the *Select your maintenance strategy* page, leave the default (*Yes, I can specify an offline ...*), and click *Next*.
5. Specify the offline maintenance time window in the *Timing* page of the wizard. Specify two or more hours a week when DB2 can perform automated maintenance tasks to preserve the health of your database. For now, configure the window to start at 1AM every Monday through Thursday for a 6 hour period. Click the *Next* button.
6. Configure notification on the *Mail Server* page of the wizard. DB2 can automatically send an email or a page 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 lab we don't have an SMTP server, so leave this blank and click *Next*.

7. 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.
8. Now that the database has been created, populate it with a few tables and some data. For your convenience, two scripts, called `quicklab2.db2` and `quicklab2.dat` are available for you. The `quicklab2.db2` script contains the commands used to create the tables and therefore must be run first. The `quicklab2.dat` script contains statements that insert data into the tables. Both scripts can be found in the zip file `expressc_book_quicklabs.zip` accompanying this book. To run these scripts, use the Command Editor:
  - a. Ensure that the new database you created is selected in the drop-down list in the Command Editor toolbar. If the new database does not appear in the list, add a connection to it using the *Add* button. (See earlier Figure 4.10)
  - b. Click the *Selected* → *Open* menu or the open button  in the Command Editor and navigate to the folder where the scripts are stored. Select the `quicklab2.db2` file and click the *OK* button. The contents of the file should now be displayed in the Command Editor's input area. Click the *Run* button  to run the script. Verify that there were no errors encountered when running the script.
9. Repeat Step (8) for the `quicklab2.dat` file.

This new database 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.



## 4.5 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:

1. SQL scripts
2. Operating system (shell) scripts.

### 4.5.1 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 following commands 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;
```

**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.

### 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 previous script from the DB2 Command Window or Linux shell, you can use the following command:

```
db2 -t -v -f script1.db2 -z script1.log  
or, db2 -tvf script1.db2 -z script1.log
```

In this command:

- t indicates statements use the default statement termination character (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 the following message filename should be 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 below called “functions.db2”, which contains the necessary Data Definition Language (DDL) to create a function, a semi-colon is needed at the end of the SELECT 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 run time conflict from the script, resulting in an error reported by DB2.

```
CREATE FUNCTION f1()  
    SELECT ... ;  
    ...  
END!
```

### 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
```

The description of other flags that can be used can be obtained from the Command Window or Linux shell with the command:

```
db2 list command options
```

### 4.5.2 Operating system (shell) scripts

Operating system scripts provide greater flexibility and power, as they give you the possibility to add additional programming logic. They are platform dependant, but they support parameters and variables. Below is 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
```

#### 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
```

On Windows using the “bat” extension tells the operating system 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.

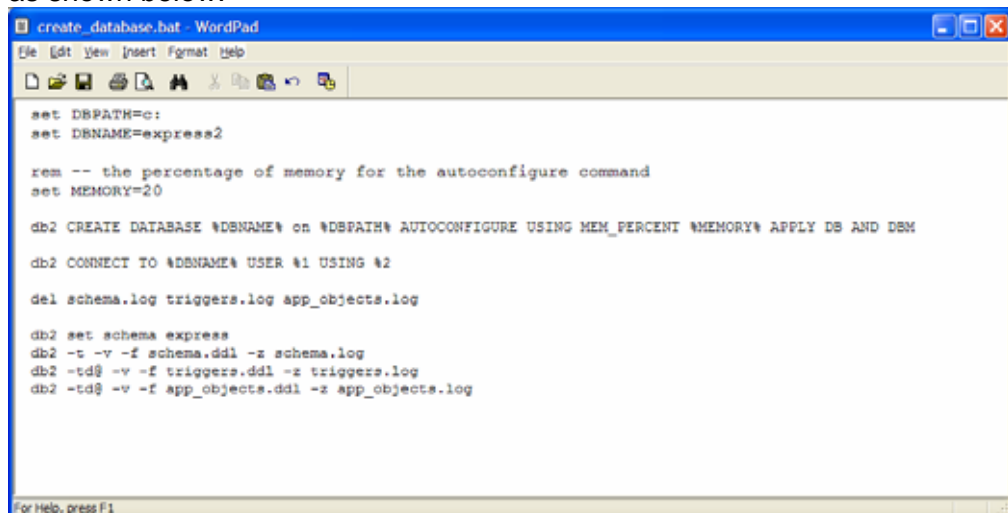
## Quicklab #3 – Create an installation script for EXPRESS Database

### Objective

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 Quicklab, you will create an operating system script to deploy the *EXPRESS* database. The script will call the previously generated SQL scripts for database objects. In order to save space, this quicklab 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, such as Notepad or Wordpad and input the information as shown below.



```
create_database.bat - WordPad
File Edit View Insert Format Help
set DBPATH=c:
set DBNAME=express2

rem -- the percentage of memory for the autoconfigure command
set MEMORY=20

db2 CREATE DATABASE %DBNAME% on %DBPATH% AUTOCONFIGURE USING MEM_PERCENT %MEMORY% APPLY DB AND DSM
db2 CONNECT TO %DBNAME% USER %1 USING %2

del schema.log triggers.log app_objects.log

db2 set schema express
db2 -t -v -f schema.ddl -z schema.log
db2 -td@ -v -f triggers.ddl -z triggers.log
db2 -td@ -v -f app_objects.ddl -z app_objects.log

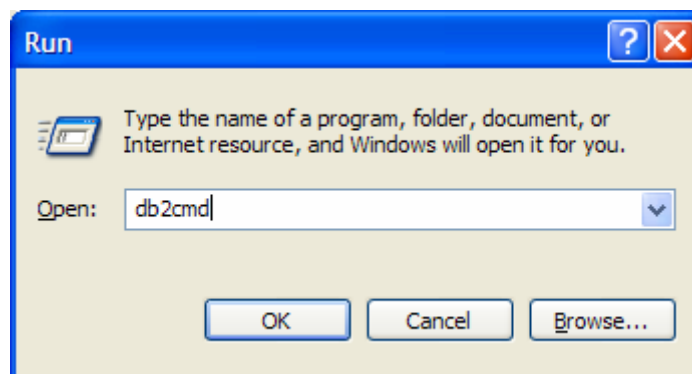
For Help, press F1
```

2. Save the script file in a directory and call it `create_database.bat`. In the *Save As* dialog window, ensure you choose the *MS-DOS Format* option (in Wordpad). If you save the file with a different format, Wordpad may introduce invisible characters which may cause problems in the execution of the script. Also, put quotes around the file name to ensure that Windows does not append a `.TXT` extension to it as shown in the figure below.



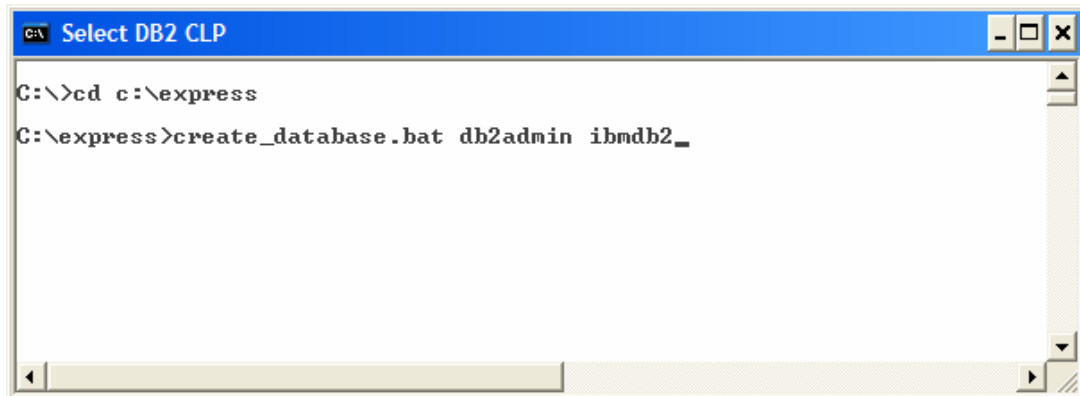
3. To run scripts that interact with DB2, you must have a DB2 command line environment. To open a DB2 Command Window, go to *Start > Program Files > IBM DB2 > DB2COPY1 (default) > Command Line Tools > Command Window*.

Alternatively, you can use *Start > Run*, type `db2cmd` and press *enter* as shown below.



4. Then to run the script, enter the commands:

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



```
C:\>cd c:\express
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 isn't called with any parameters?
  - f. Why don't the SQL scripts being called contain CONNECT TO statements? How do they connect to the database?

## 4.6 Task Center

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 4.15 shows the Task Center.

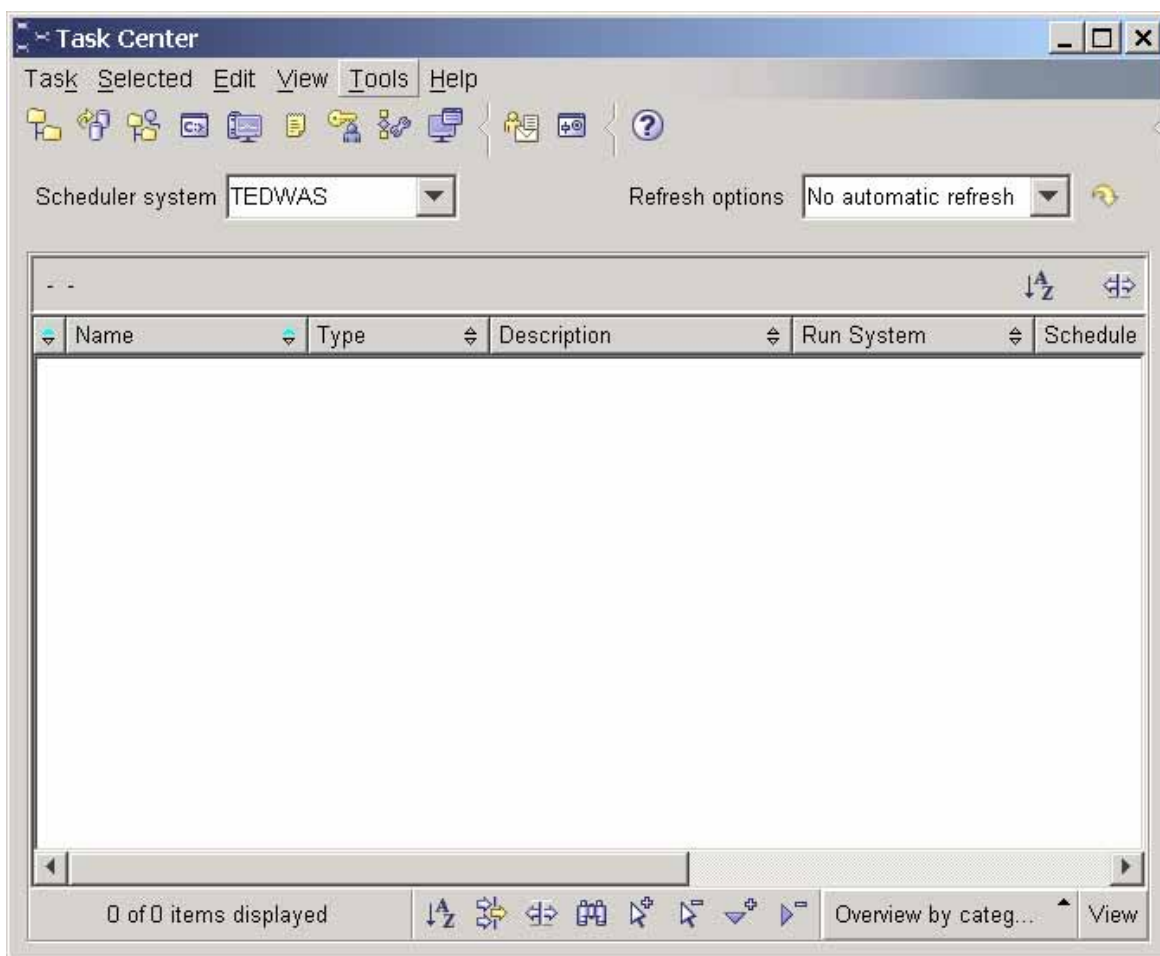


Figure 4.15 – The Task Center

### 4.6.1 The Tools Catalog database

All the details about your tasks and task scheduling are stored in a separate DB2 database called the Tools Catalog database. This database must already exist 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, `sysools` 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.

### Launching the Task Center

You can launch the Task Center from the Control Center by clicking on *Tools > Task Center*, as shown in Figure 4.16. Alternatively, you can start this tool from the Windows Start menu: *Start > Programs > IBM DB2 > DB2COPY1 > General Administration Tools > Task Center*

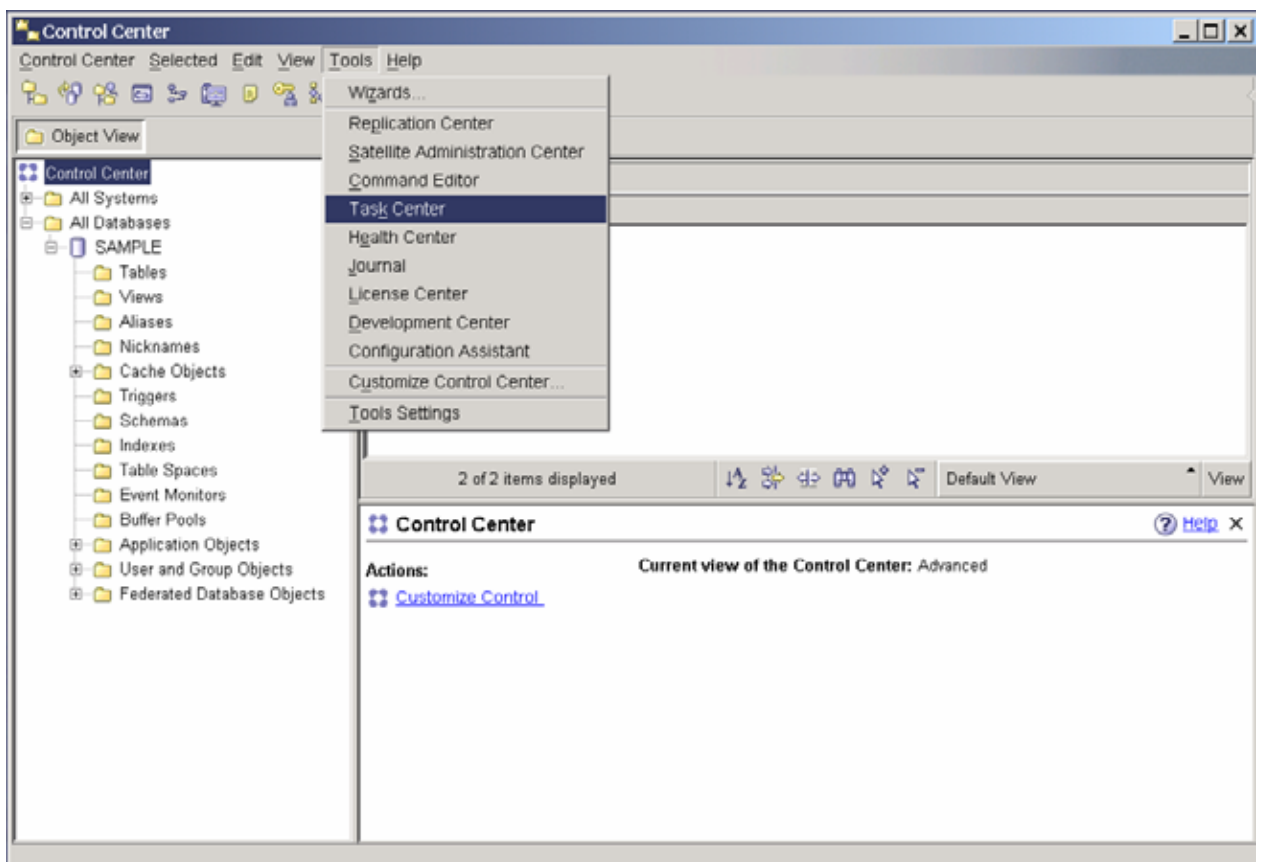


Figure 4.16 – Launching the Task Center

### Scheduling with the Task Center

Any type of script can be scheduled using the Task Center (whether or not it was created through a DB2 GUI tool). 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.

## 4.7 Journal

The DB2 Journal GUI tool provides a DBA with a journal of activities in online form. Figure 4.17 shows the Journal and Table 4.2 shows the information you can obtain from the Journal.

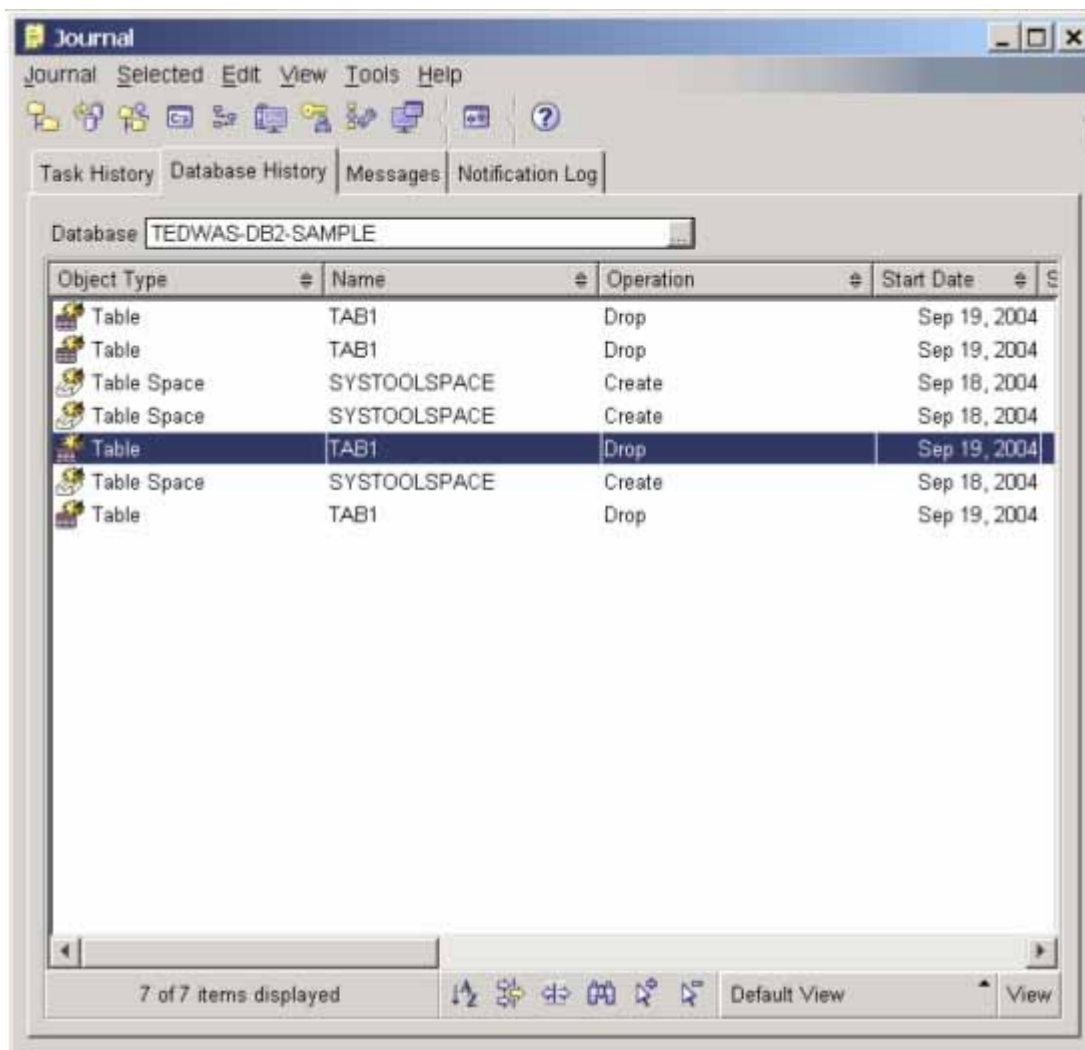


Figure 4.17 –The Journal

Type of Information	Description
Task History	All attempted scheduled tasks and their success status
Database History	A record of database activities performed (backup, restore, RE-ORGs, 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



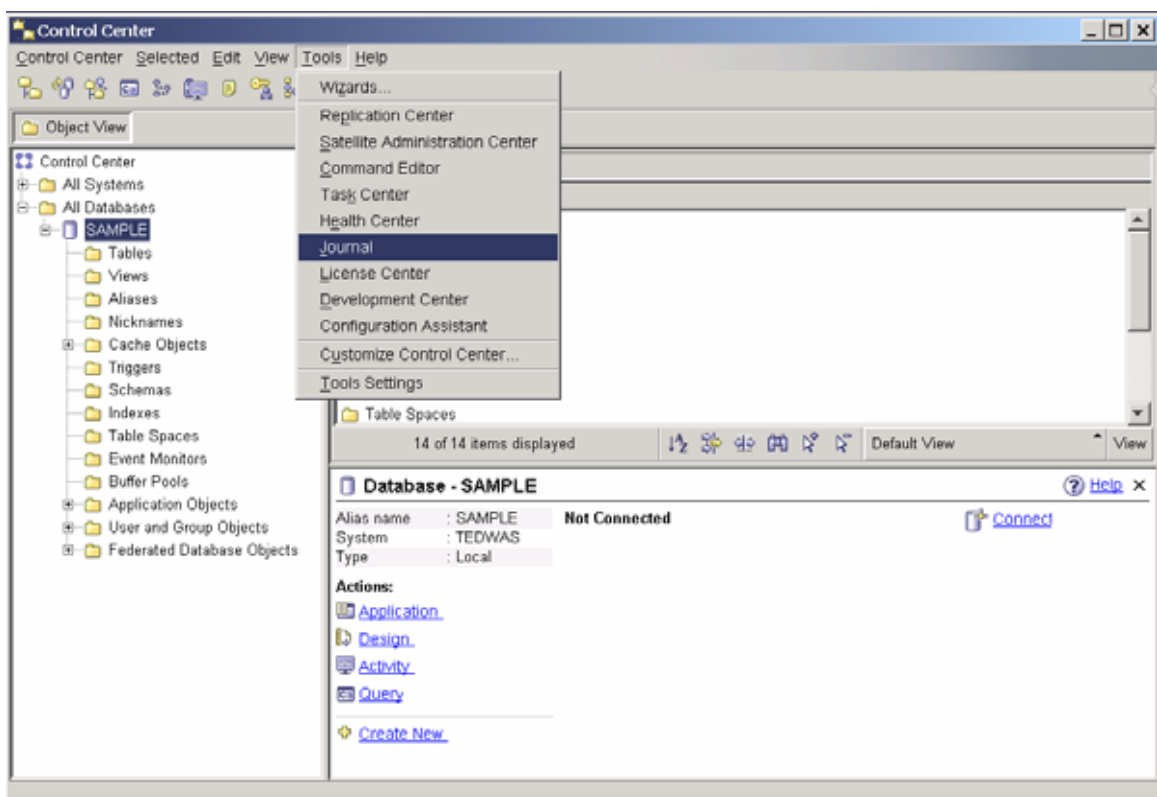
dialog box too quickly or by accident.

Notification Log Contains system-level message. Critical errors are recorded here

**Table 4.2 - Information provided in the Journal**

### Launching the Journal

You can launch the Journal from the Control Center by clicking on *Tools > Journal*, as shown in Figure 4.18. Alternatively, you can start this tool from the Windows Start menu: *Start > Programs > IBM DB2 > DB2COPY1 > General Administration Tools > Journal*.



**Figure 4.18 – Launching the Journal**

## 4.8 Health Monitor

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 aspect 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:

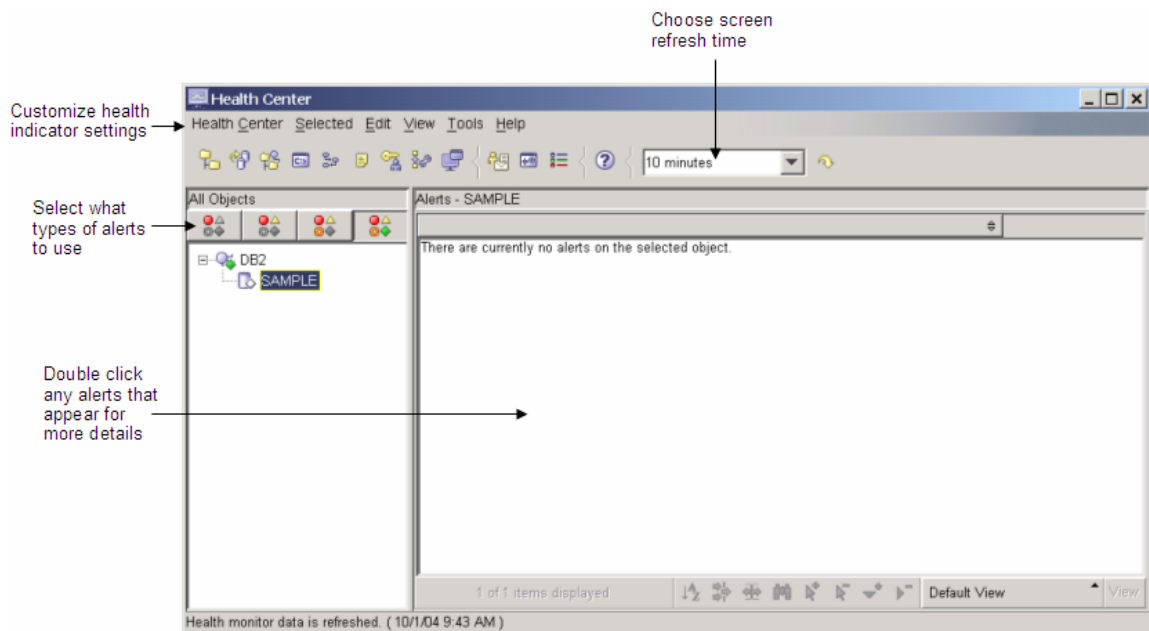
- ▶ Attention: a non-normal state

- ▶ 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.

#### 4.8.1 Health Center

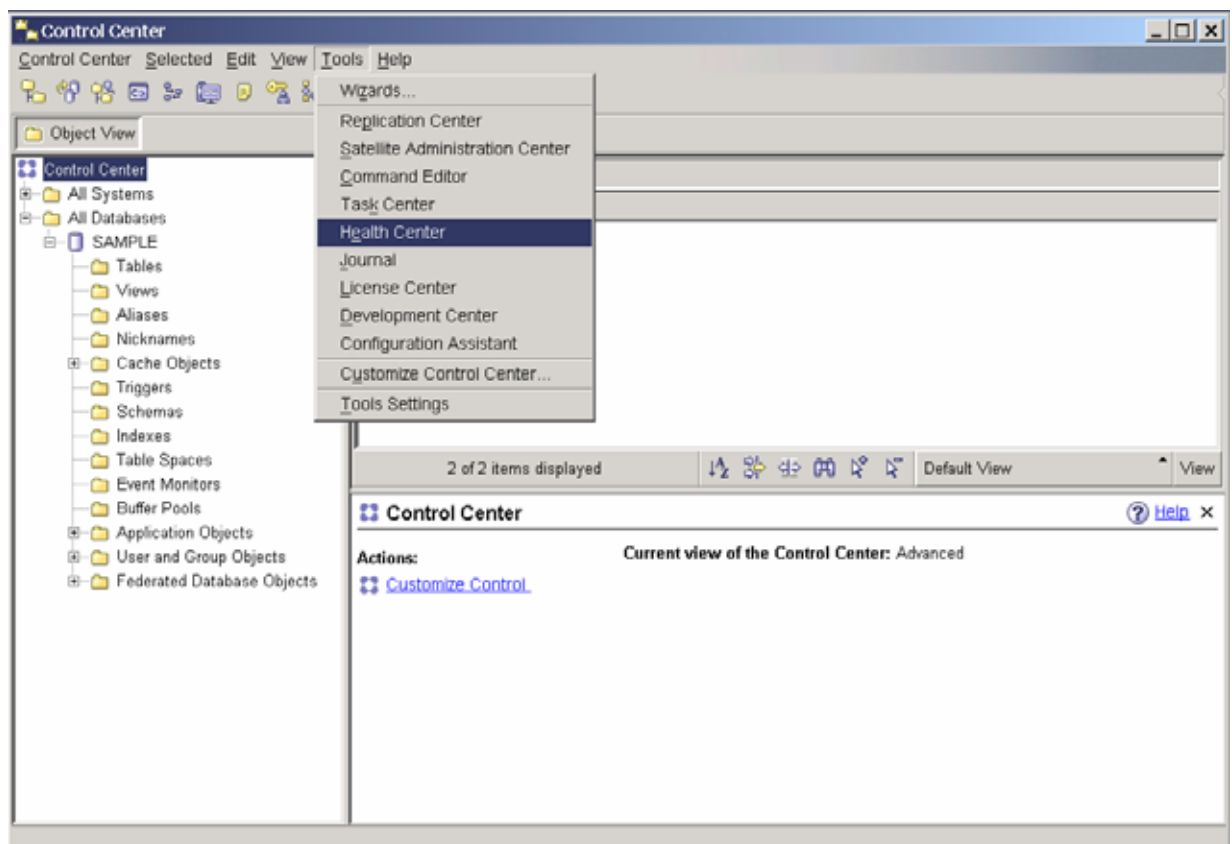
The Health Center is a graphical tool for interacting with the Health Monitor. The Health Center breaks down health alerts on a system by instance, database, and table space levels. Figure 4.19 shows the Health Center.



**Figure 4.19 – The Health Center**

#### Launching the Health Center

You can launch the Health Center from the Control Center by clicking the Tools Menu and choosing Health Center. This is shown in Figure 4.20. You can also start this tool from *Start > Programs > IBM DB2 > DB2COPY1 > Monitoring Tools > Health Center*



**Figure 4.20 – Launching the Health Center**

### **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 4.21. 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.

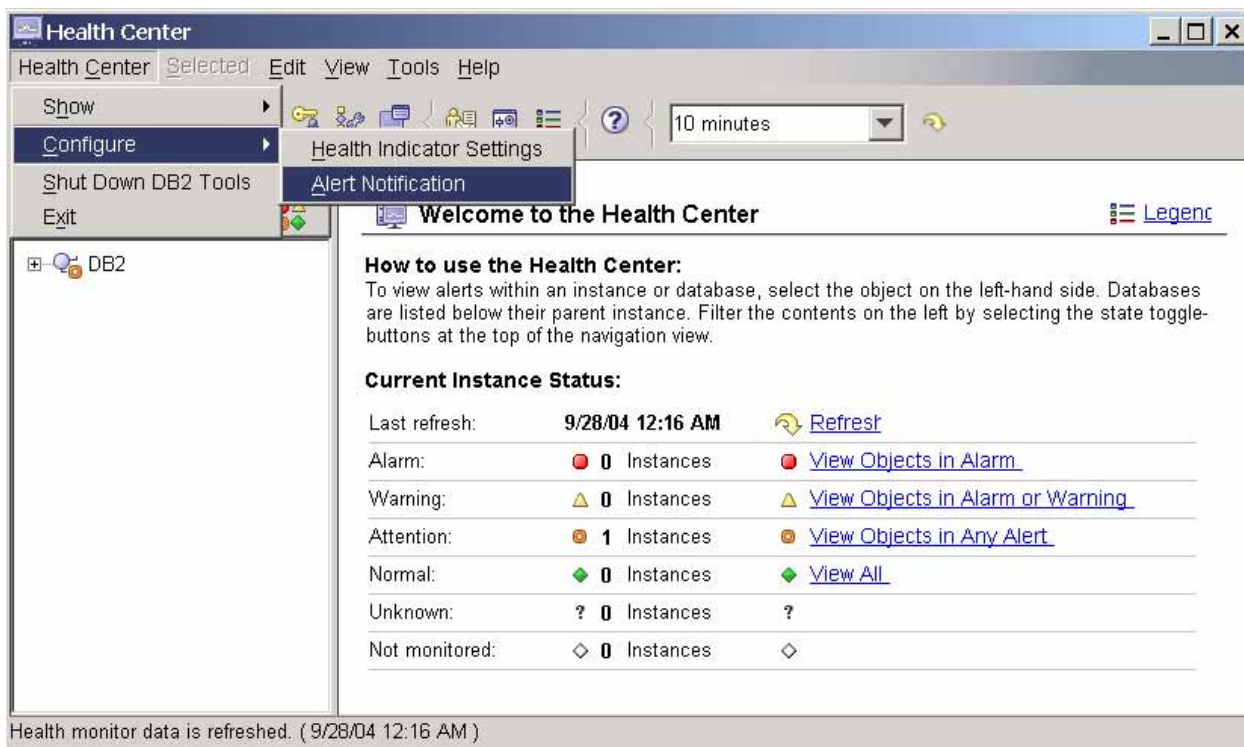


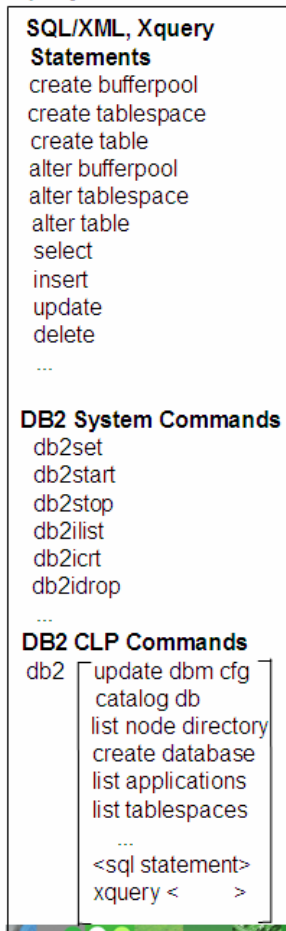
Figure 4.21 – Alert Notification

# 5

## Chapter 5 – DB2 Environment

In this chapter we will discuss the DB2 environment: instances, databases, and configuration files. Figure 5.1 shows with a red ellipse the area we focus in this chapter.

### DB2 Commands & SQL/XML, Xquery Statements



### DB2 Tools

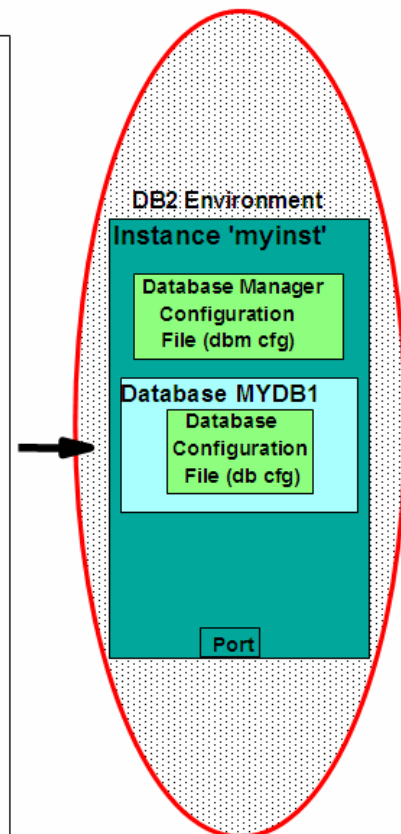
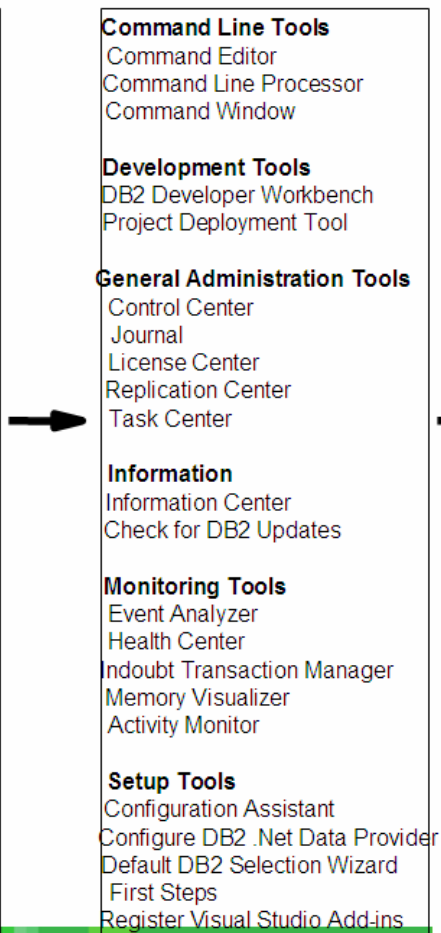
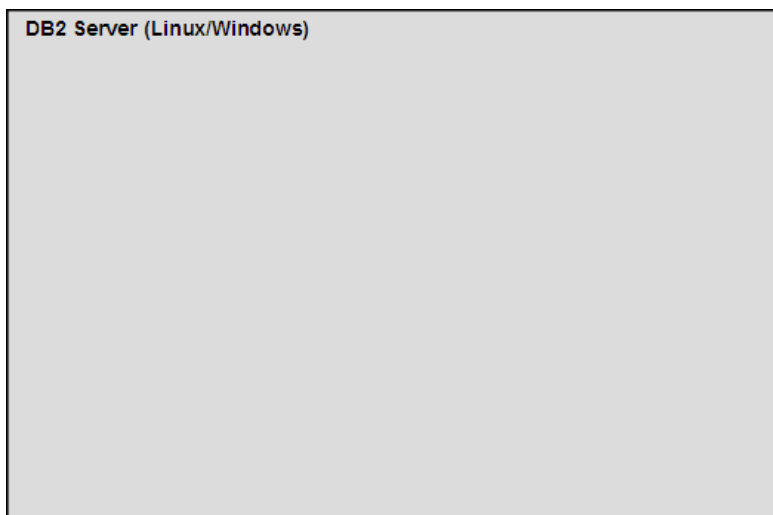


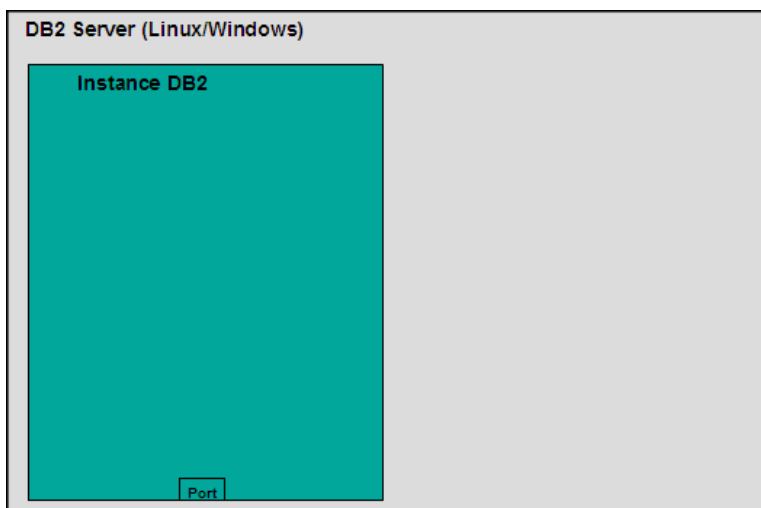
Figure 5.1 – The DB2 big picture: DB2 environment

To describe the DB2 environment, let's describe each component element step by step. Figure 5.2 shows a representation of aDB2 data server after installing DB2 Express-C 9.



**Figure 5.2 – Representation of a DB2 Server after installing DB2 Express-C 9**

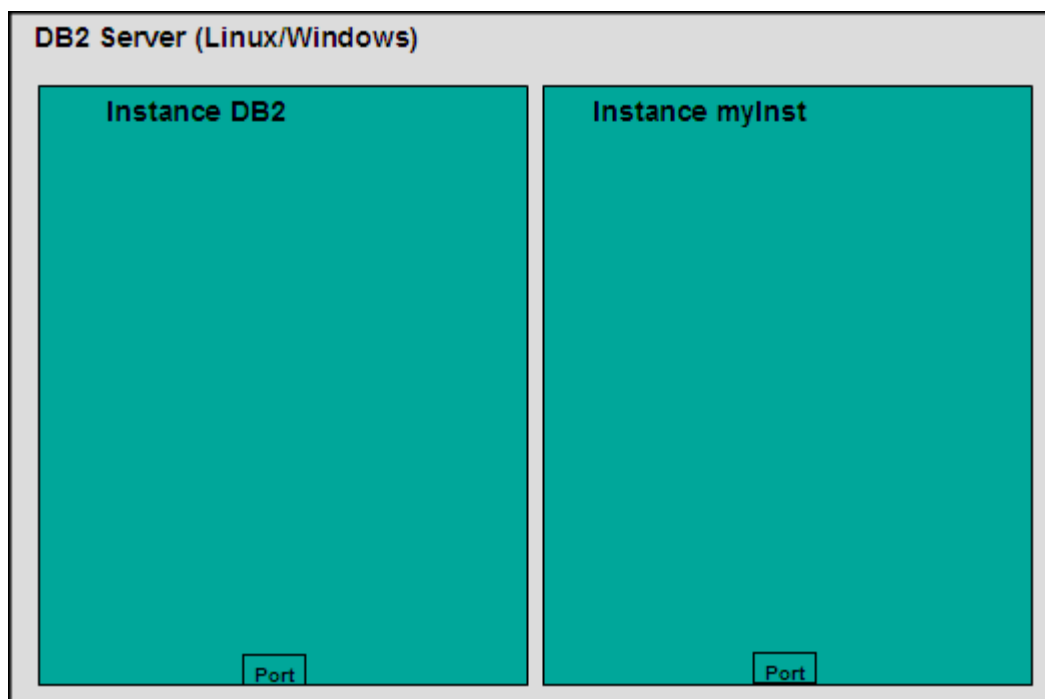
As part of the installation in Windows, a default instance called “DB2” (“db2inst1” on Linux) is created. This is represented by the green box in Figure 5.3. An instance is simply an independent environment where applications can run, and databases can be created. You can create multiple instances on a data server, and use them for different purposes. For example, one instance can be used to hold databases for production, 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 5.3 – 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 5.4 shows a new instance called *myinst* as a separate green box.



**Figure 5.4 – 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. 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
```

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, run the command:

```
db2ilist
```

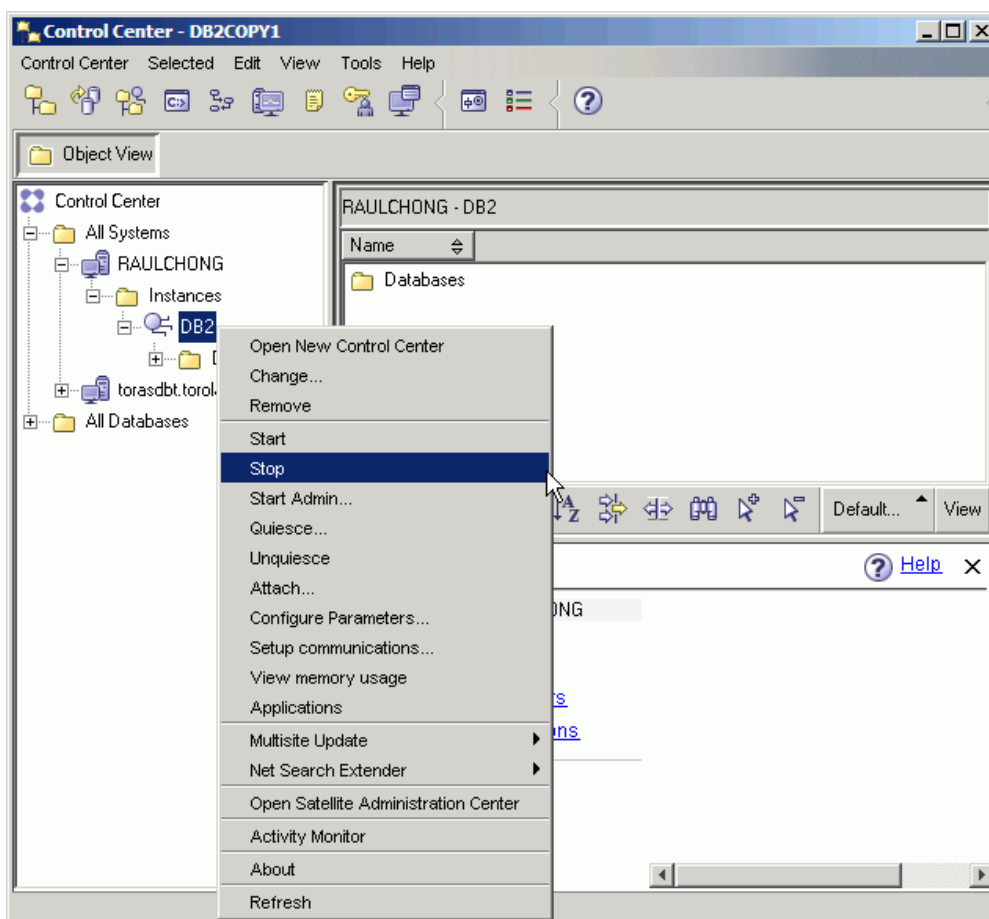
On Linux, an instance must match a Linux operating system user; therefore, to switch between instances you can simply switch users (with the `su` command).

Table 5.1 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 5.1 – 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 5.5.



**Figure 5.5 – Instance commands from the Control Center**



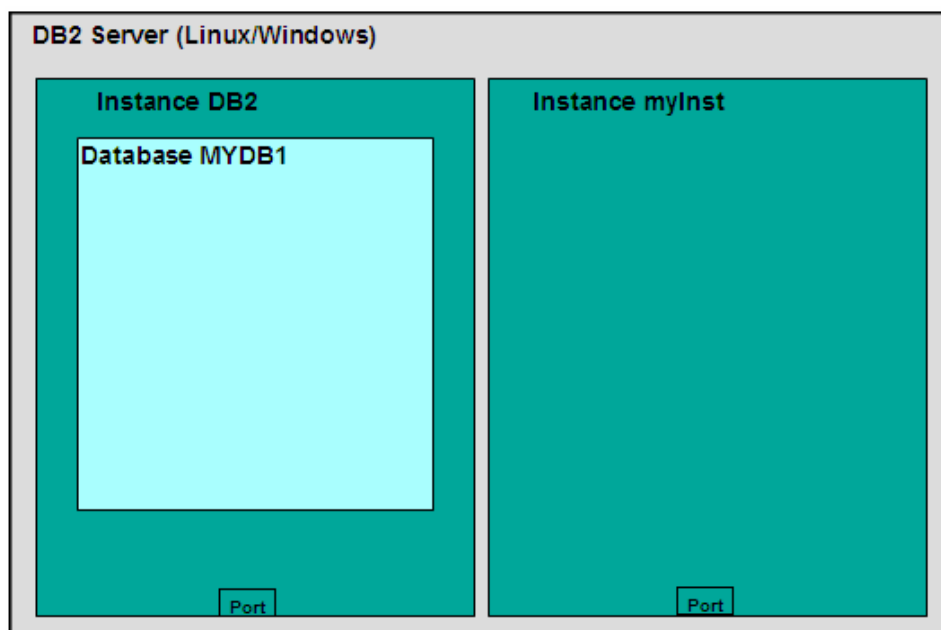
To create a database in the active instance, issue this command from the DB2 Command Window:

```
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 5.6 shows a representation of a database “MYDB1” created inside instance “DB2”.



**Figure 5.6 – Database “MYDB1” created in instance “DB2”**

Table 5.2 shows some commands you can use at the database level.

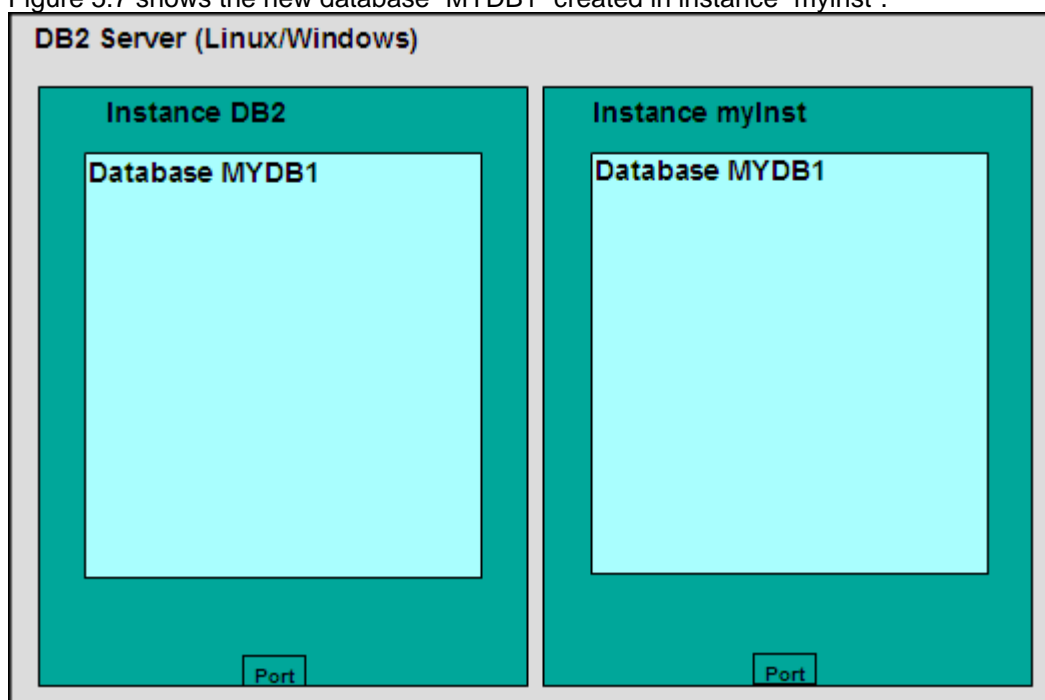
Command/SQL statement	Description
db2 create database	Creates a new database
db2 drop database	Drops a database
db2 connect to <database_name>	Connects to a database
db2 create table/create view/create index	SQL statements to create table, views, and indexes respectively

**Table 5.2 - 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:

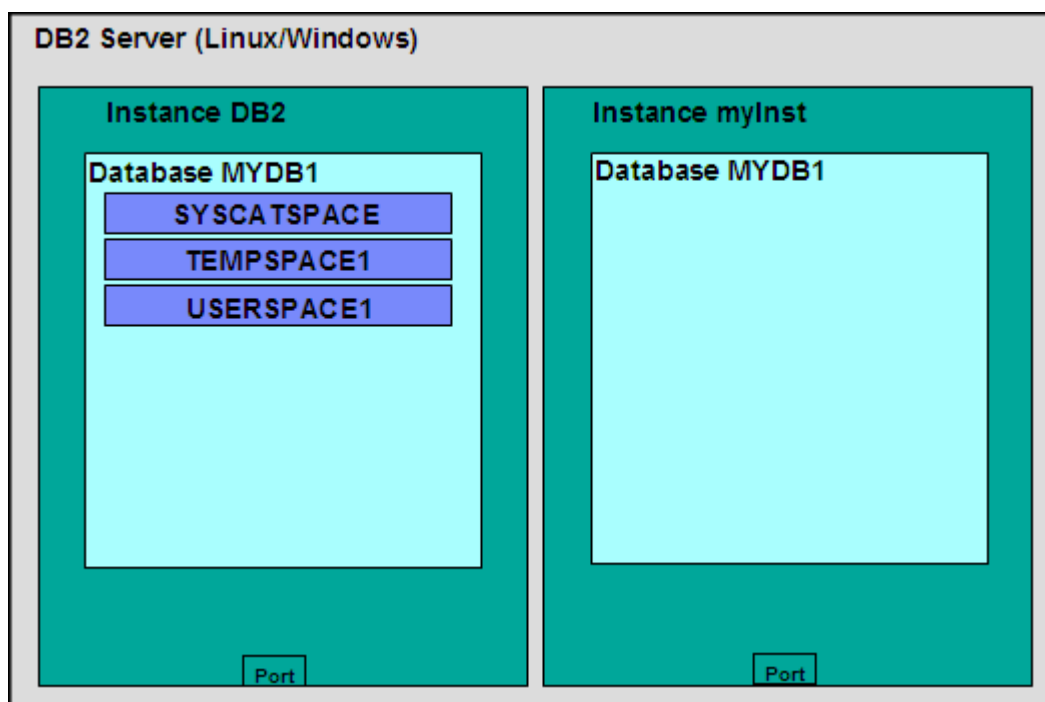
```
db2 list db directory
set db2instance=myinst
db2 create database mydb1
set db2instance=db2
```

Figure 5.7 shows the new database “MYDB1” created in instance “myinst”.



**Figure 5.7 – 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 5.8 shows three table spaces created by default. Table spaces will be discussed in more detail in Chapter 6, DB2 Architecture; but for now, think of table spaces as the logical layer between logical tables, and physical resources such as disks and memory.



**Figure 5.8 –Table spaces created by default when a database is created**

Table space SYSCATSPACE contains the Catalog tables. The Catalog is also known in other relational database management systems as the data dictionary. 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.

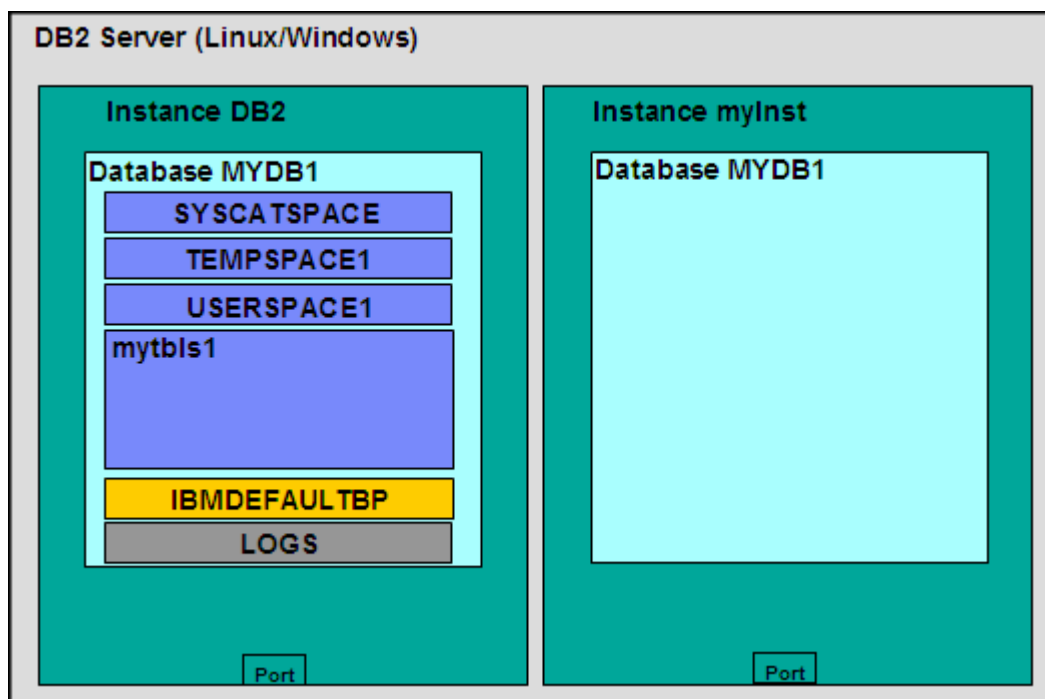
You can also create your own table spaces using the CREATE TABLESPACE statement. Figure 5.9 shows the table space MYTBLS1 created inside database MYDB1 on instance DB2. When you create a table space, you specify the disks to use and the memory (buffer pool) to use. Therefore, if you have a “hot” table, 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 5.9, we show two other objects created by default: A buffer pool called IBMDEFAULTBP, and the log files.

A buffer pool is basically 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 the 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 5.9 – 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 5.10 shows the table space “mytbls1” inside both the database MYDB1 and the database SAMPLE, within instance DB2. This is valid because the databases are independent units. Note that Figure 5.10 does not show the other default objects of database SAMPLE due to space constraints in the figure.

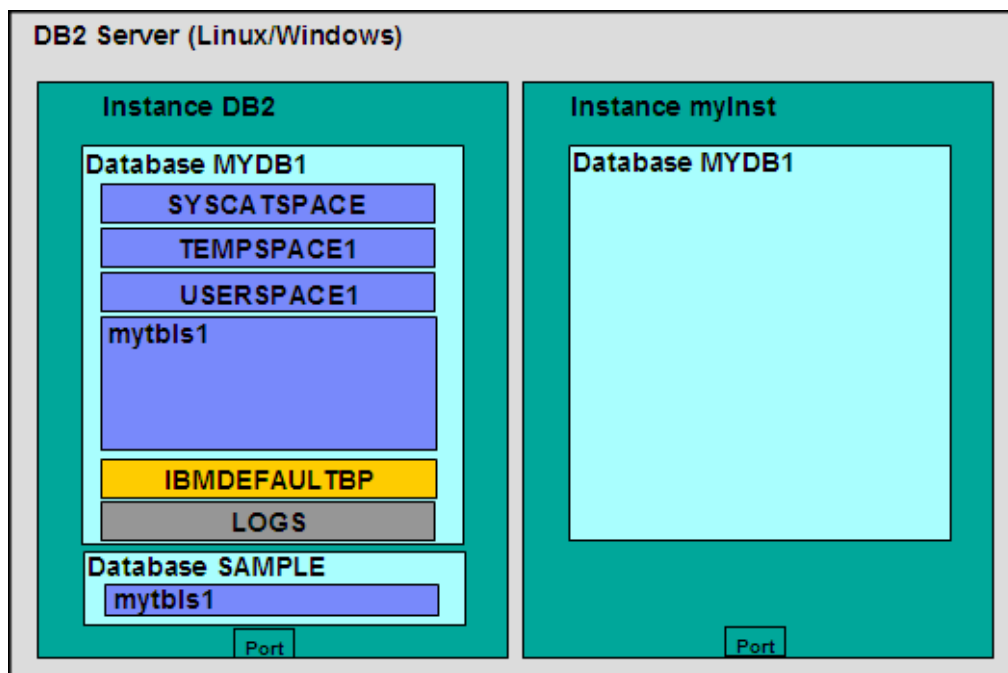


Figure 5.10 – Table spaces with the same name in different databases.

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 5.11.

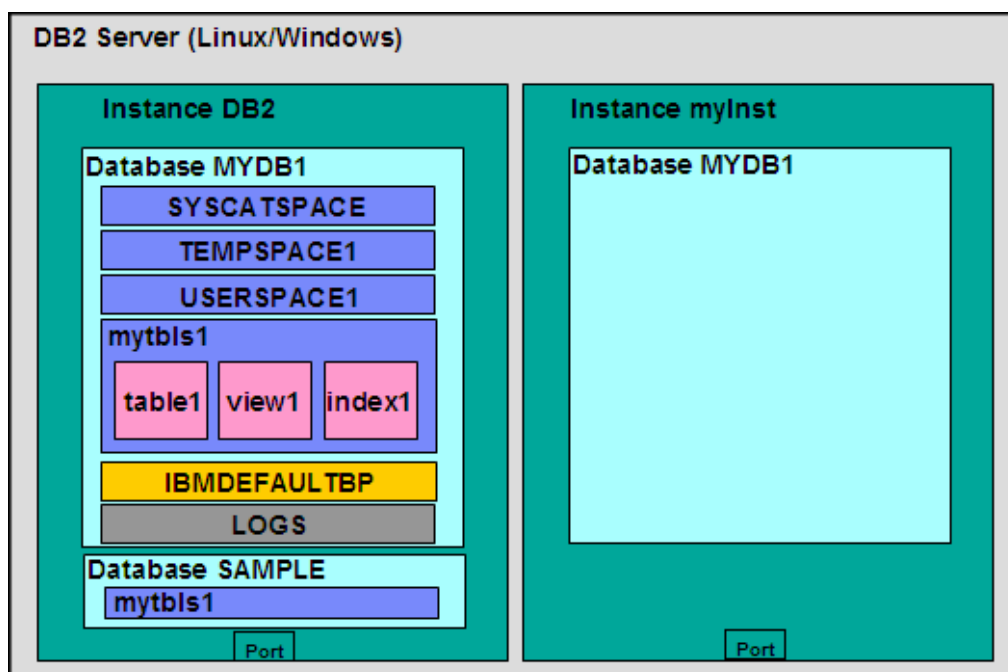


Figure 5.11 – Tables, views, indexes created inside the table space

## 5.1 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 5.12. 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.

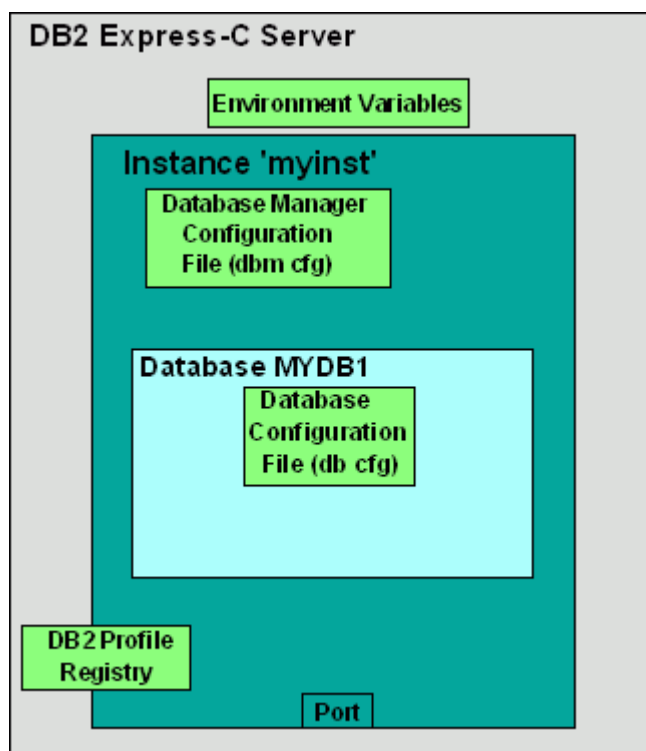


Figure 5.12 – DB2 Configuration

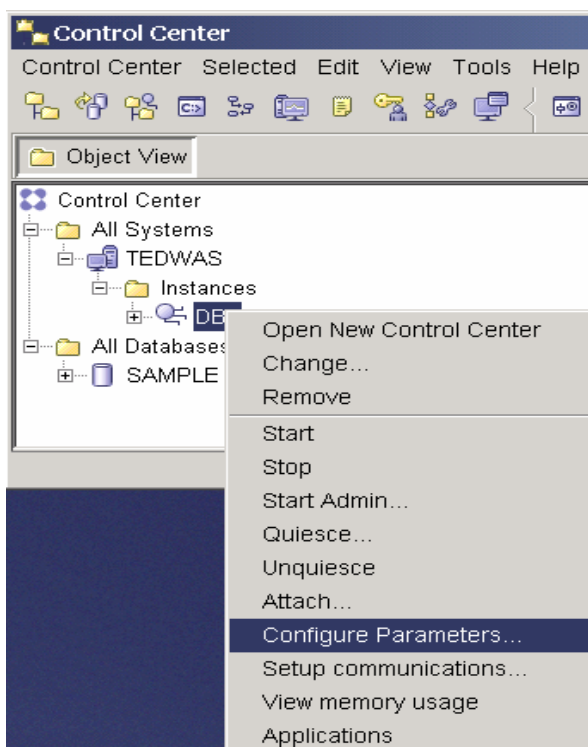
### 5.1.1 Environment variables

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`

### 5.1.2 Database manager configuration file (dbm cfg)

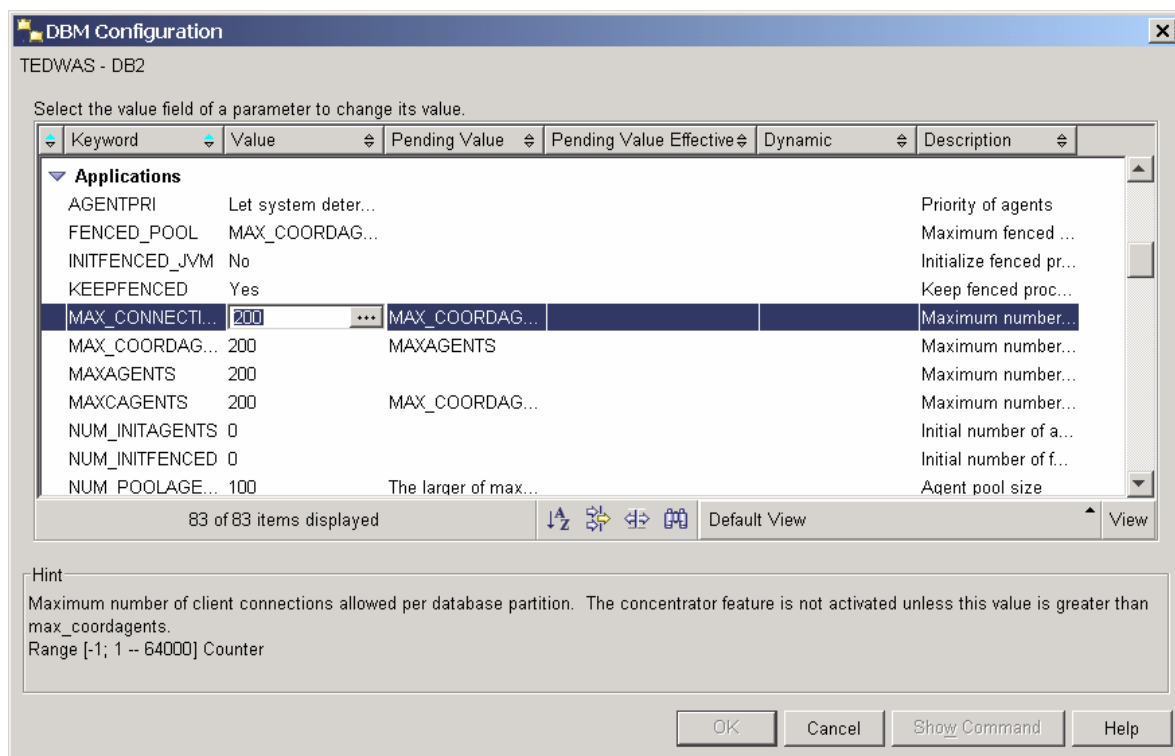
The Database manager configuration file (dbm cfg) includes parameters that affect the instance and all the databases contained within. The database manager configuration file can be viewed or modified using the command line, or through the DB2 Control Center.

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 5.13.



**Figure 5.13 – Configuring the dbm cfg from the Control Center.**

After choosing Configure Parameters, the screen shown in Figure 5.14 will be displayed with the list of dbm cfg parameters.



**Figure 5.14 –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 5.3 shows some useful commands to manage the dbm cfg from the Command Line.

Command	Description
<code>db2 get dbm cfg</code>	Retrieves information about the dbm cfg
<code>db2 update dbm cfg using &lt;parameter_name&gt; &lt;value&gt;</code>	Updates the value of a dbm cfg parameter

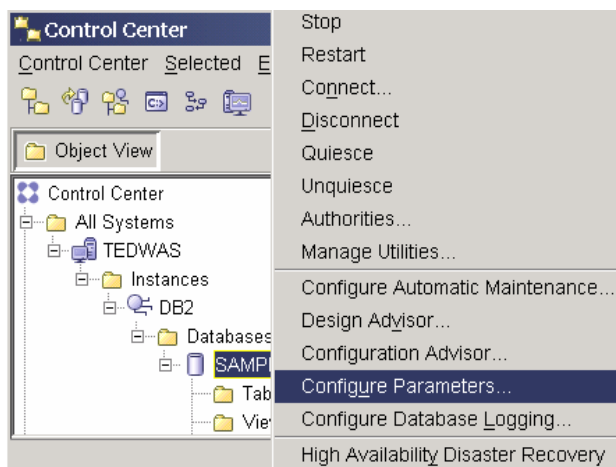
**Table 5.3 - Commands to manipulate the dbm cfg**



### 5.1.3 Database configuration file (db cfg)

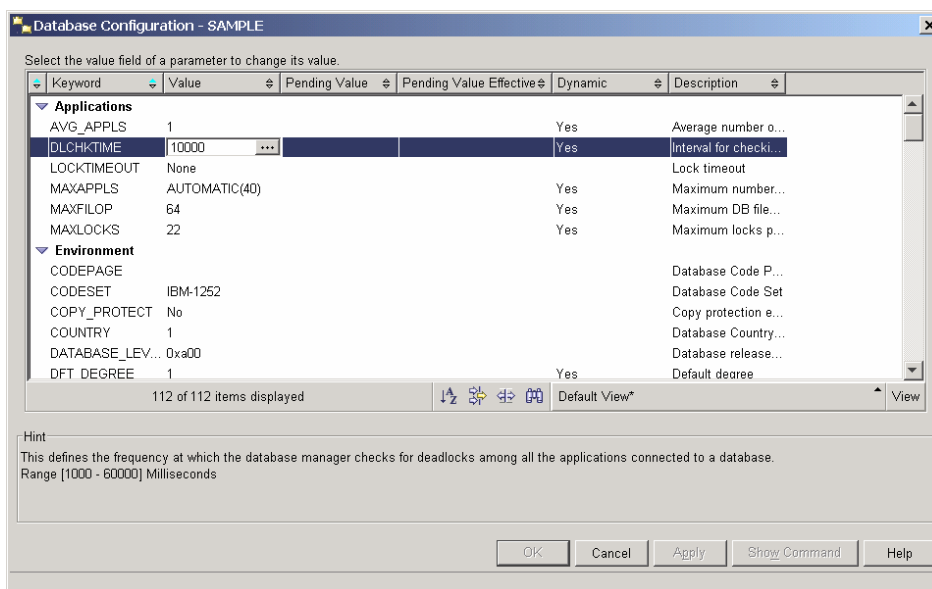
The database configuration file (db cfg) includes parameters that affect a particular database. The database configuration file can be viewed or modified using the command line, or through the DB2 Control Center.

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 5.15.



**Figure 5.15 – Configuring the db cfg from the Control Center.**

After choosing Configure Parameters, the screen shown in Figure 5.16 will be displayed with the list of db cfg parameters.



**Figure 5.16 –The db cfg**

Table 5.4 shows some useful commands to manage the db cfg from the Command Line.

Command	Description
<code>get db cfg for &lt;database_name&gt;</code>	Retrieves information about the db cfg for a given database
<code>update db cfg for &lt;database_name&gt; using &lt;parameter_name&gt; &lt;value&gt;</code>	Updates the value of a db cfg parameter

**Table 5.4 - Commands to manipulate the db cfg**

#### 5.1.4 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 5.5 shows some useful commands to manipulate the DB2 profile registry

Command	Description
<code>db2set -all</code>	Lists all the DB2 profile registry variables that are currently set
<code>db2set -lr</code>	Lists all the DB2 profile registry variables
<code>db2set &lt;parameter&gt;=&lt;value&gt;</code>	Assigns a parameter with a given value

**Table 5.5 - Commands to manipulate the DB2 profile registry**

Table 5.6 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 <code>install-path\bin\db2envvars.bat</code> command. This variable cannot be used for this purpose.

**Table 5.6 – 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
```

### 5.2 The DB2 Administration Server

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. There is only one DAS per physical computer as shown in Figure 5.16.

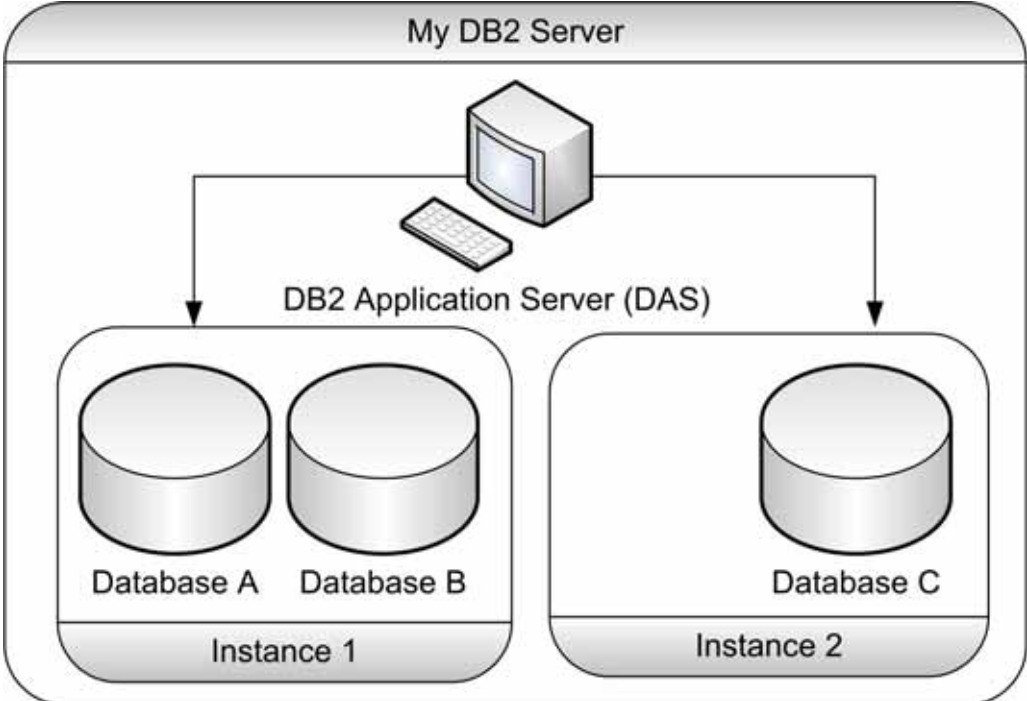


Figure 5.16 –The DB2 Administration Server (DAS)

## **PART II – LEARNING DB2: DATABASE ADMINISTRATION**

# 6

## Chapter 6 – DB2 Architecture

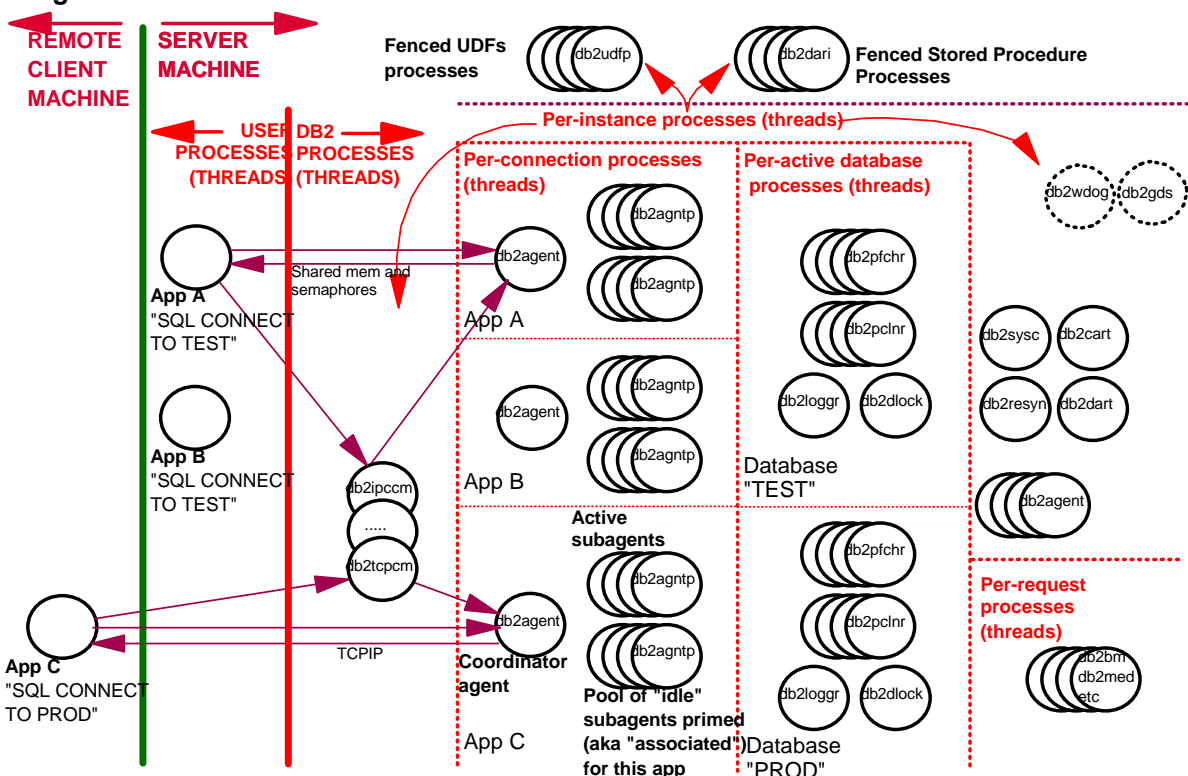
In this chapter we briefly discuss the DB2 architecture:

- The DB2 process model
- The DB2 memory model
- The DB2 storage model

### 6.1 DB2 process model

Figure 6.1 depicts the DB2 Process Model. In the figure, the left of the green vertical line represents a remote client machine, and on the right side of this line, a server machine. When an application (App C) on the client machine tries to connect to the server using an SQL CONNECT statement, the communication 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 db2tpcm process. In some cases, like when parallelism is enabled, a db2agent may spawn other agents (db2agntp). Other agents such as db2pfchr, db2loggr, db2dlock may also be used for different purposes. Most common processes are described in Table 6.1.

Figure 6.1 – The DB2 Process Model



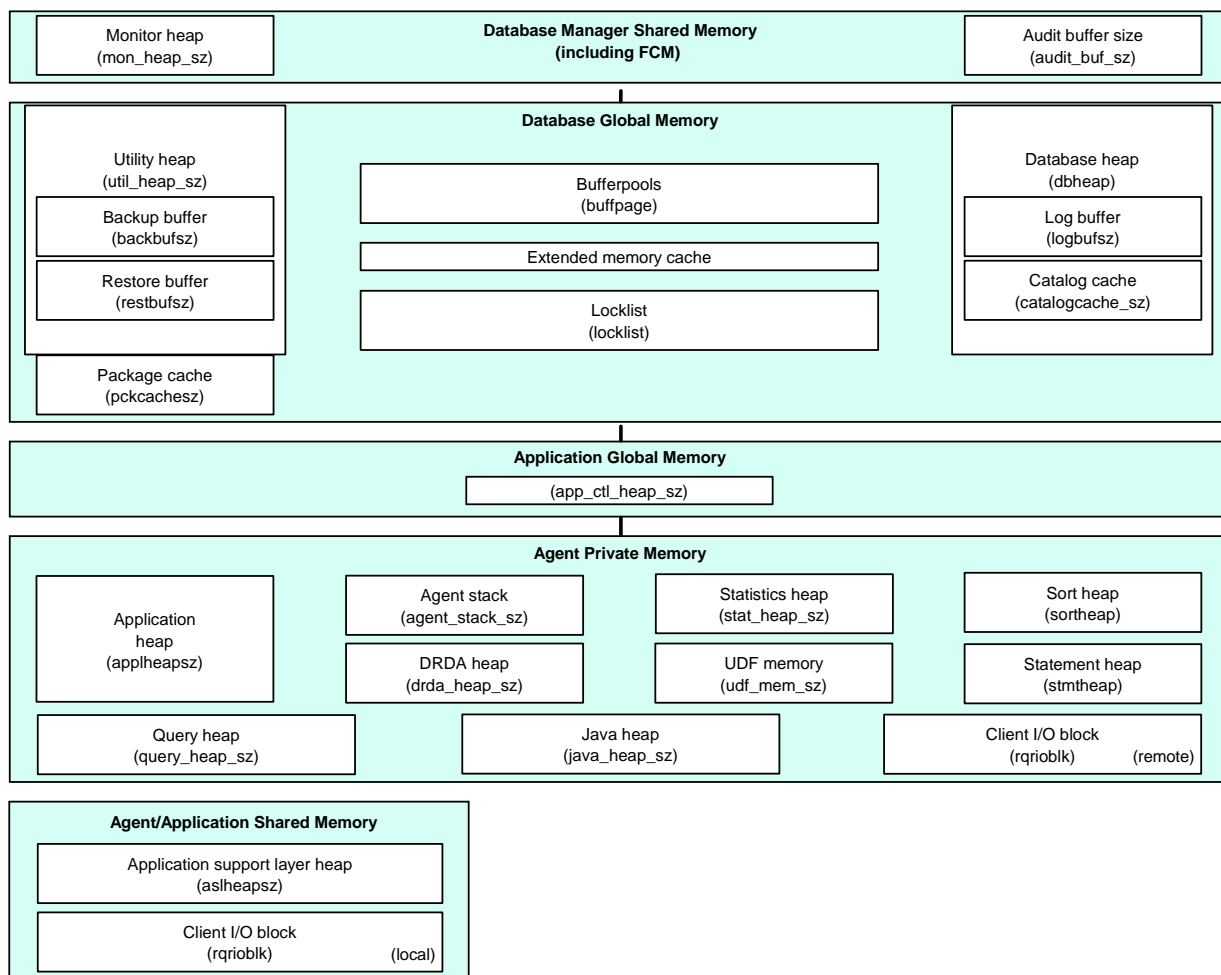
Process Name	Description
db2sysc	database instance process (one per INSTANCE)
db2tccpm	TCPIP communication/listener
db2agent	Coordinator agent that performs database operations on behalf of applications (at least 1 per connection, depending if Connection Concentrator is enabled).
db2agntp	Active subagent spawned if INTRA_PARALLEL is set to YES. Will perform 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 6.1 – Common DB2 processes

## 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 how large the entire Database Global Memory will be.

Agent private memory is the memory used by each DB2 agent. Without using connection concentrator, each connection requires one agent. Typically an agent can use approximately 3 to 5 MB. With 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 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.

#### 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

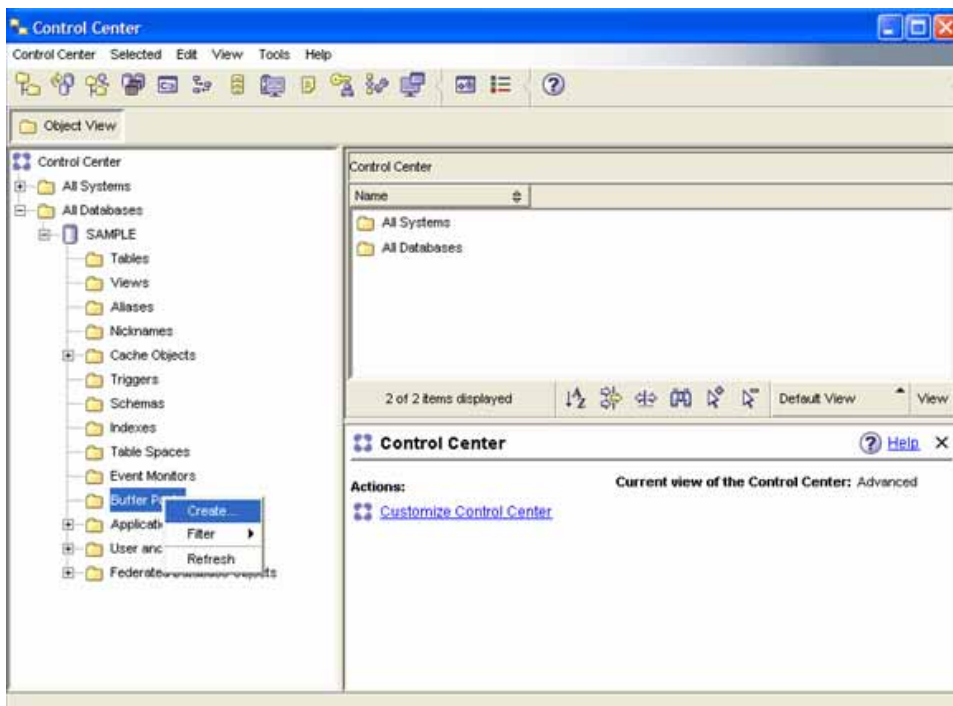


Figure 6.3 – Creating a buffer pool

After clicking on Create, the Create Buffer Pool Dialog will appear as shown in Figure 6.4

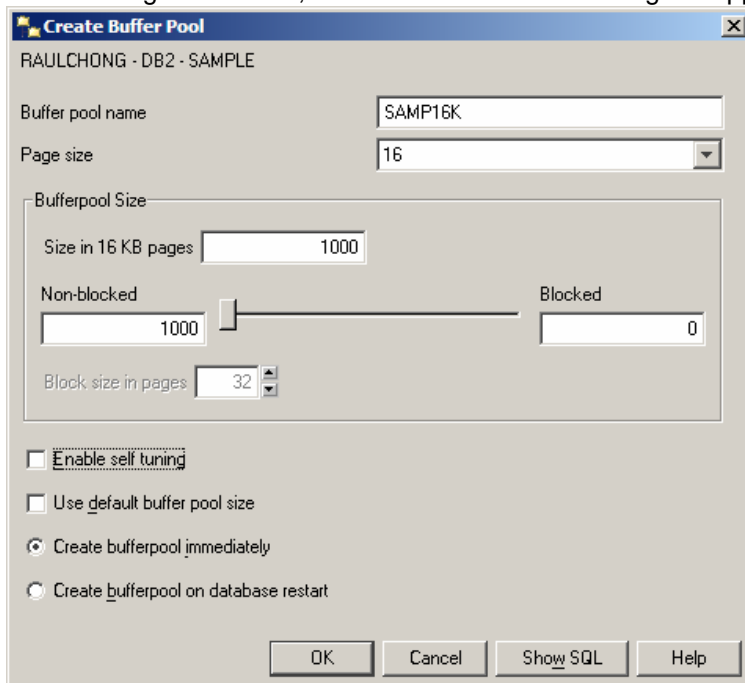


Figure 6.4 – Create a buffer pool dialog box



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 0 disables block I/O.

Once the buffer pool has been created, it would be displayed in the Control Center as shown in Figure 6.5.

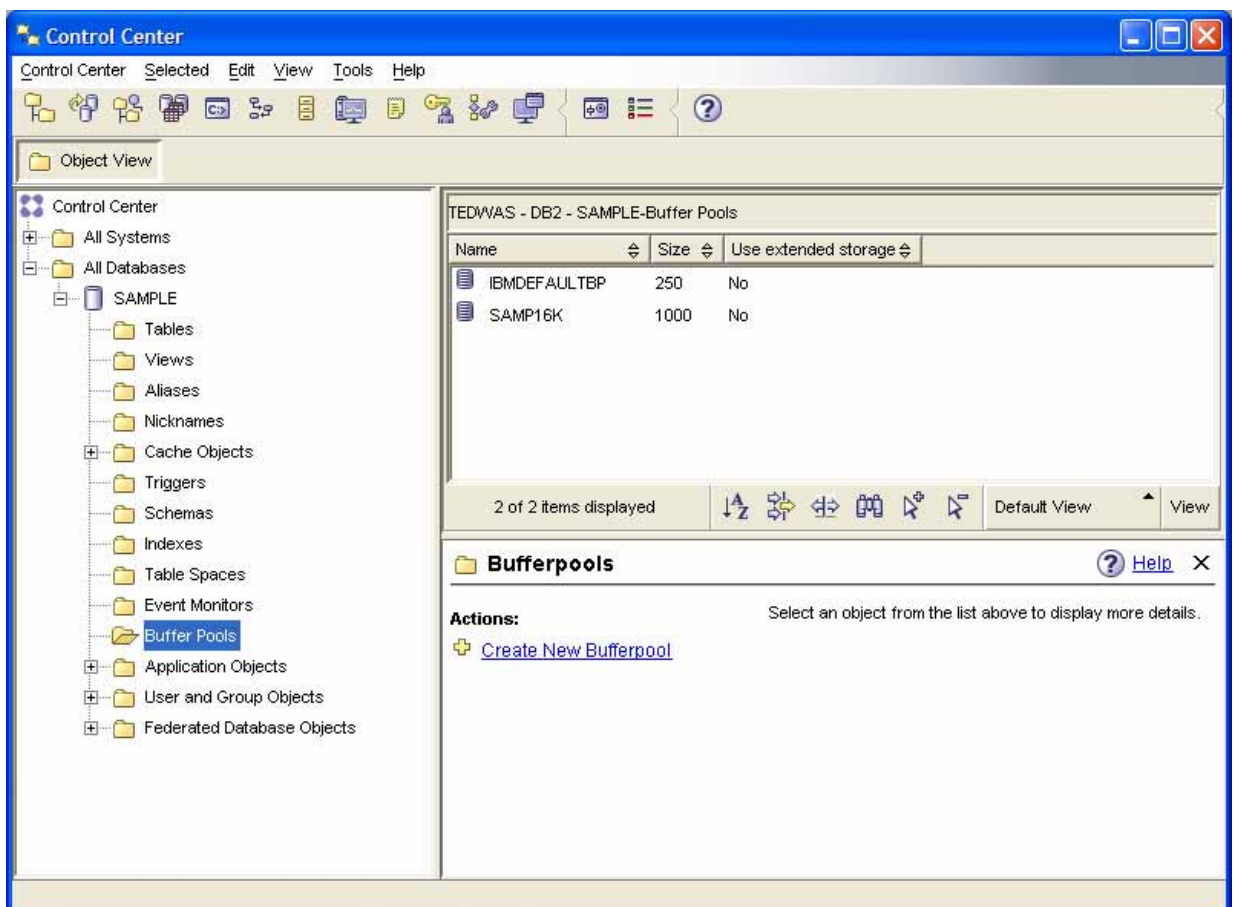


Figure 6.5 – The Control Center after the creation of buffer pool “SAMP16K”

### 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.

### Table space types

There are three types of table spaces:

- Regular  
These are for user tables. For example, the USERSPACE1 table space create 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 -- where the database was created as UNICODE and 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.
  - ▶ User temporary  
These are used to create User Defined Global Temporary tables (temporary tables in-memory). They are often confused with system temporary table spaces.

### 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 System Managed Storage (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 Database Managed Storage (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. 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 in DB2 9 this is the default type of table space. For these table spaces, a user first specifies a logical group of storage devices for. 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.

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).

### Automatic storage example

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:  
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
```

### 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 row 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.

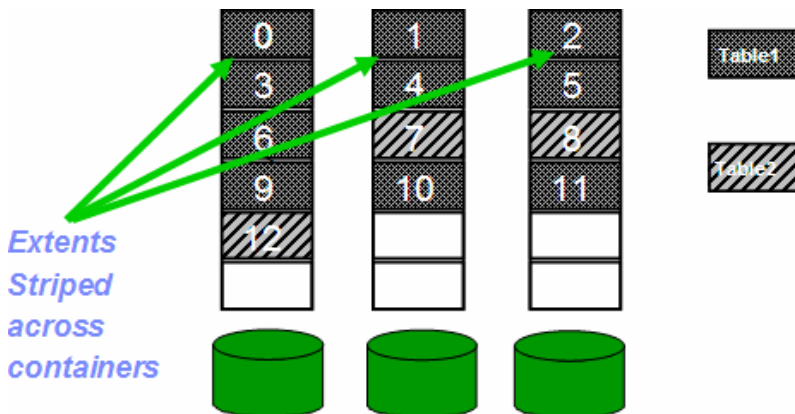


Figure 6.6 – Writing data in table spaces

### 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.

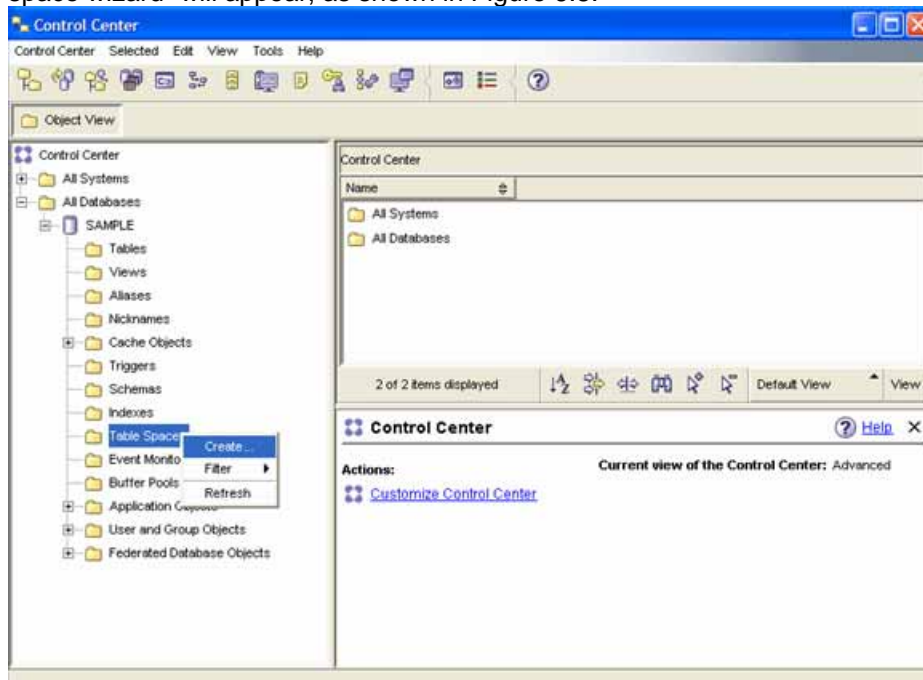


Figure 6.7 – Creating a Table Space from the Control Center

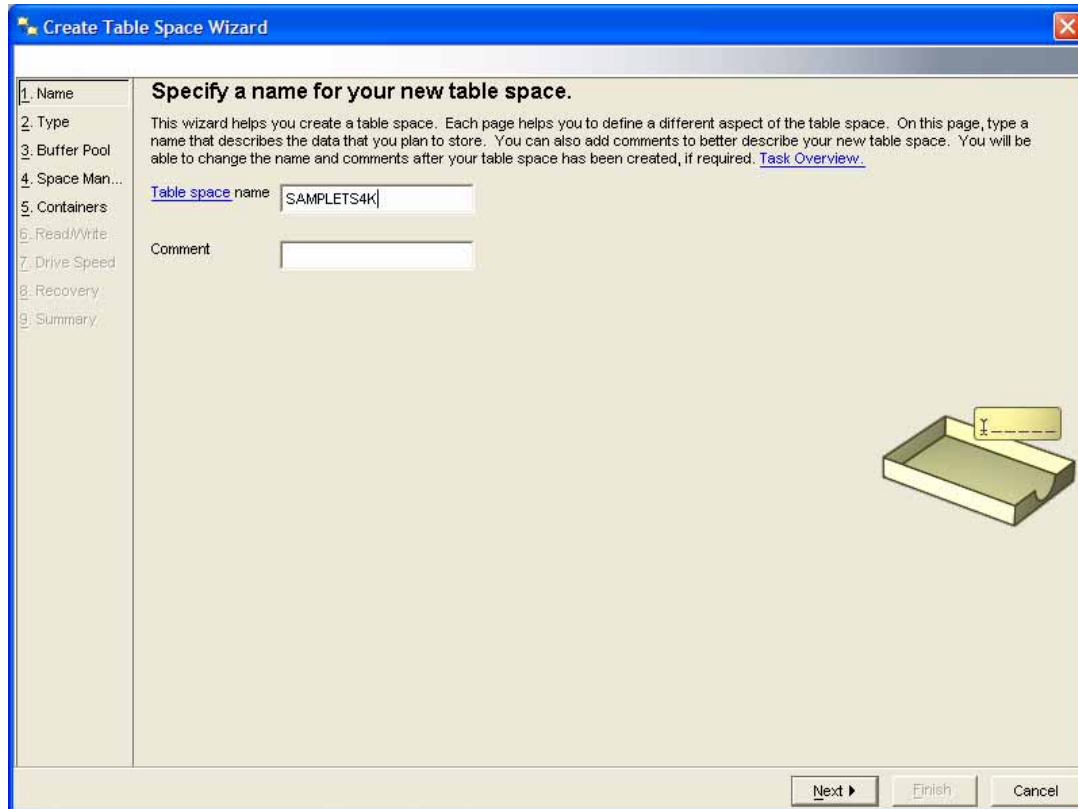


Figure 6.8 – Create Table Space Wizard



# 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.

### 7.1 Configuration Assistant

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.

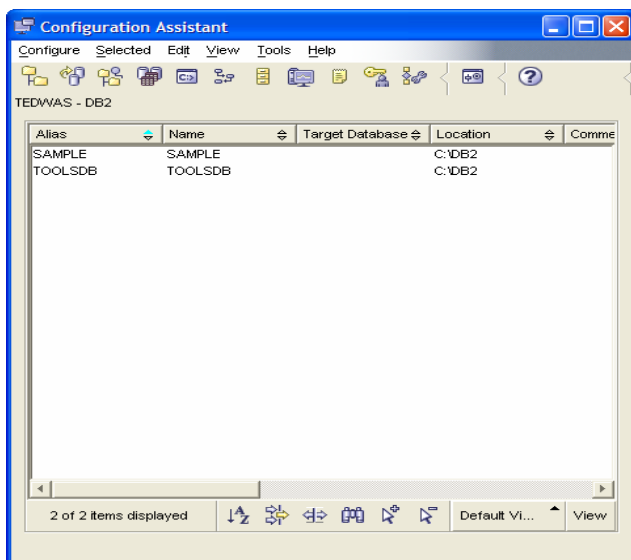


Figure 7.1 – The Configuration Assistant

### 7.1.1 Setup required at the server

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

#### 1) DB2COMM

This registry variable determines which communication protocol listeners should be monitoring requests from clients. Typically TCPIP is the communication protocol most 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.

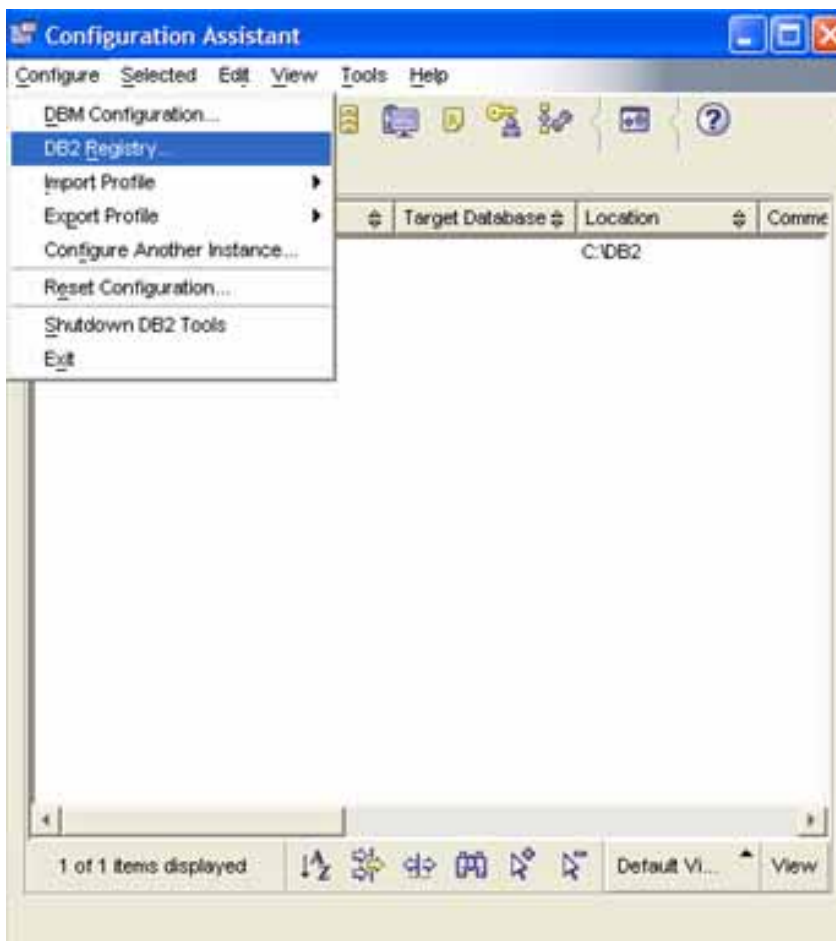


Figure 7.2 – Accessing the DB2 Registry



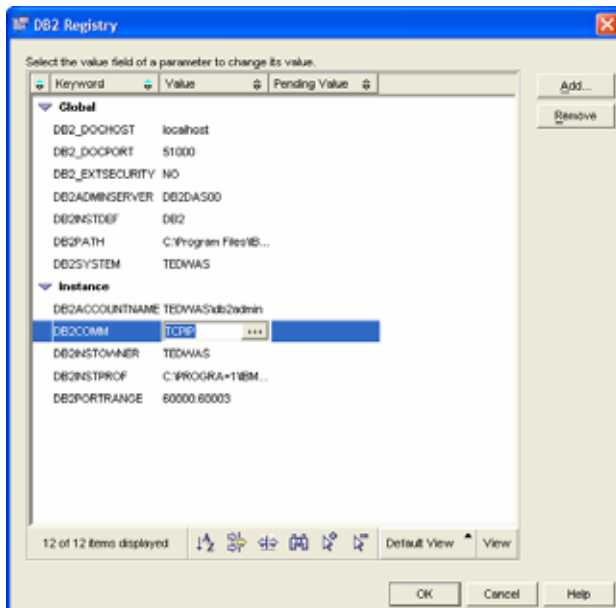


Figure 7.3 –Verifying the value of the DB2COMM DB2 Registry variable

## 2) SVCENAME

This dbm cfg parameter should be set to the service name (as defined in the TCPIP 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

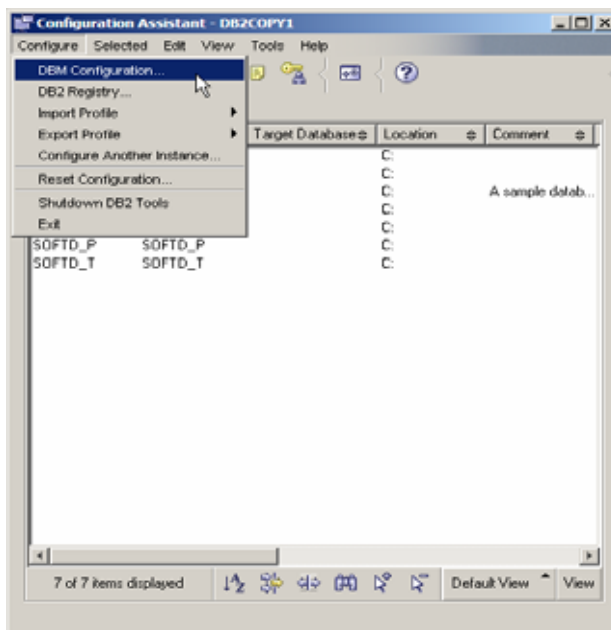
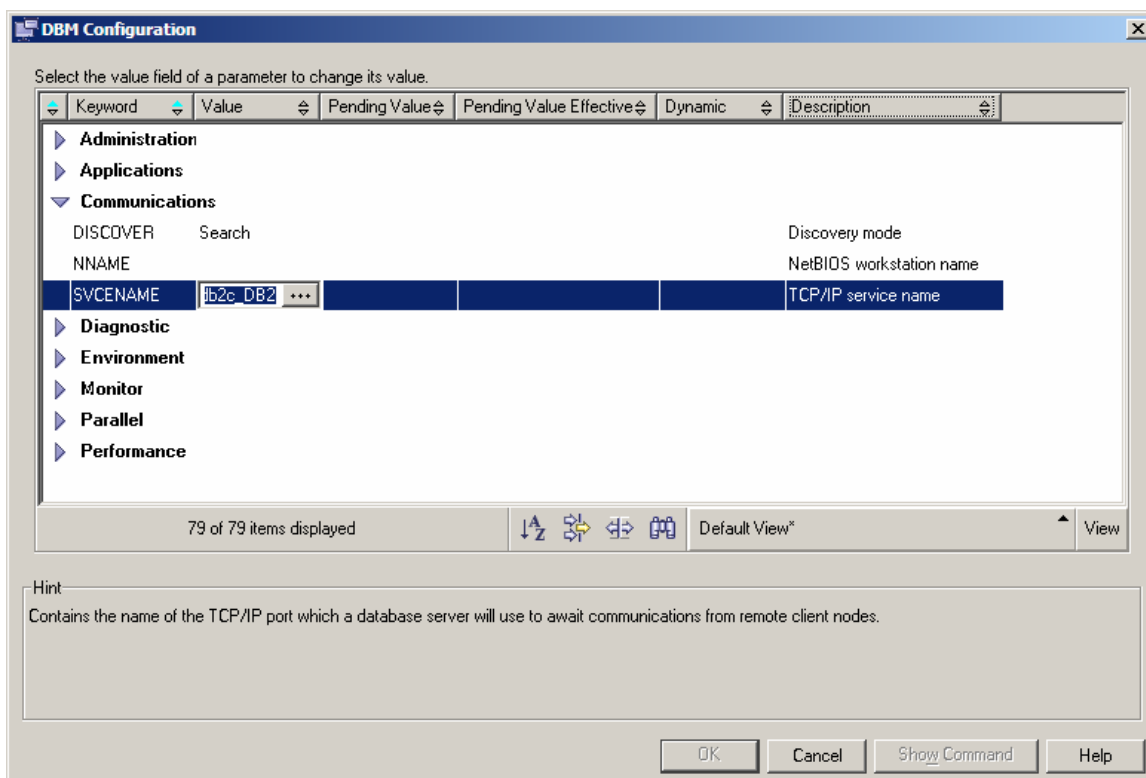


Figure 7.4 –Reviewing the dbm cfg from the Configuration Assistant

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**

### 7.1.2 Setup required at the client

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

1. The name of the database you want to connect to
2. 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 TCPIP services file
3. 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:

### 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.

### 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

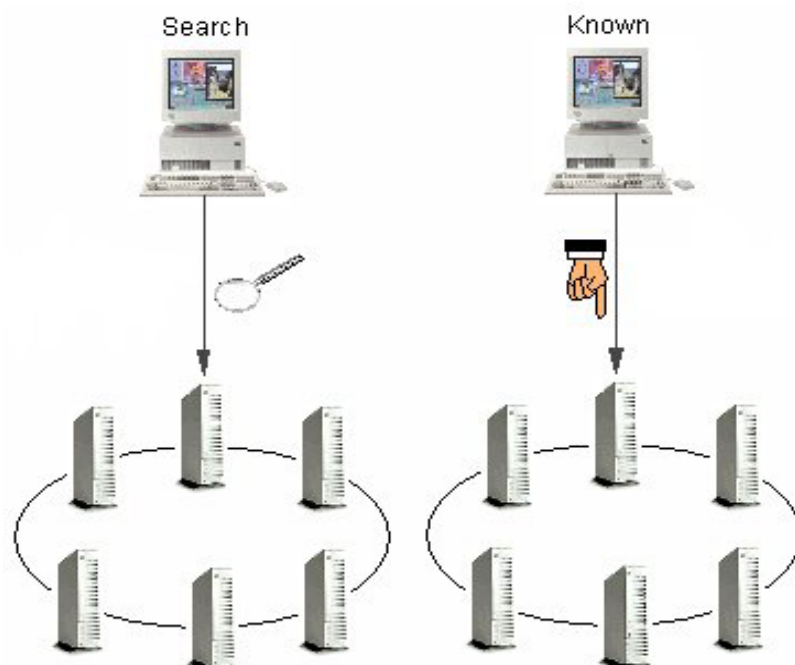
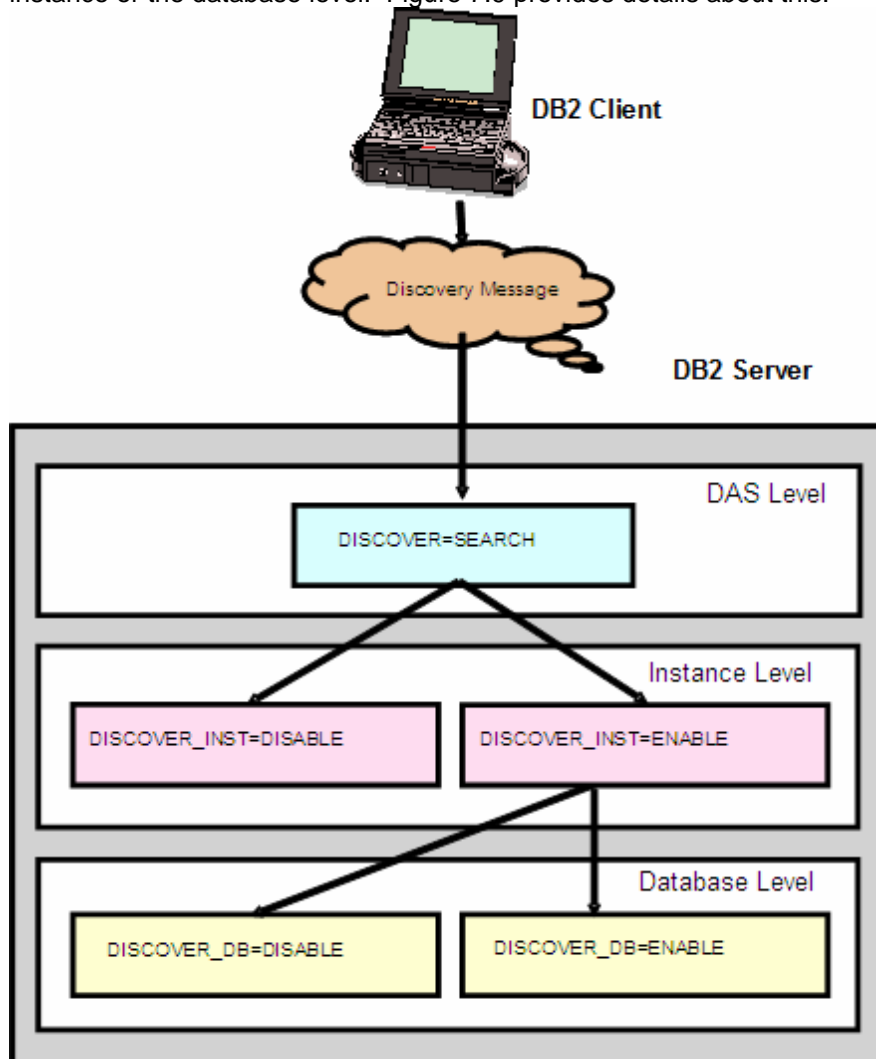


Figure 7.8 –The Search and Known search (or Discovery) methods

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 dbm cfg 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.

#### **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.

### 7.1.3 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 (i.e. configuration file) 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

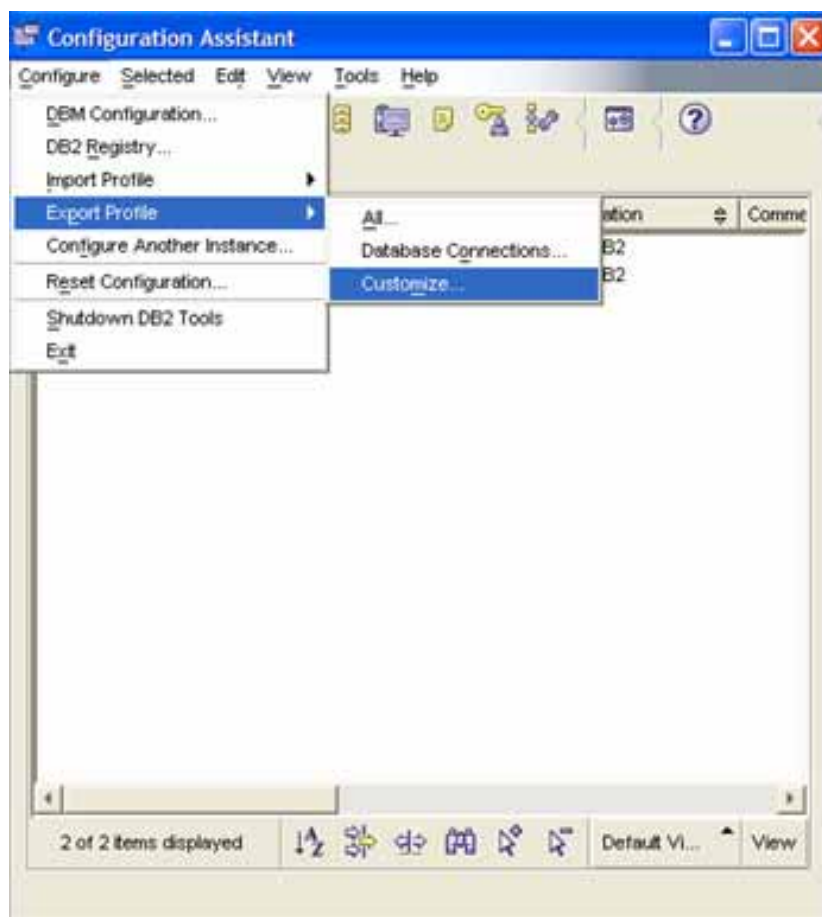


Figure 7.10 – Exporting a Profile

Figure 7.11 shows the fields that need to be completed to export a profile

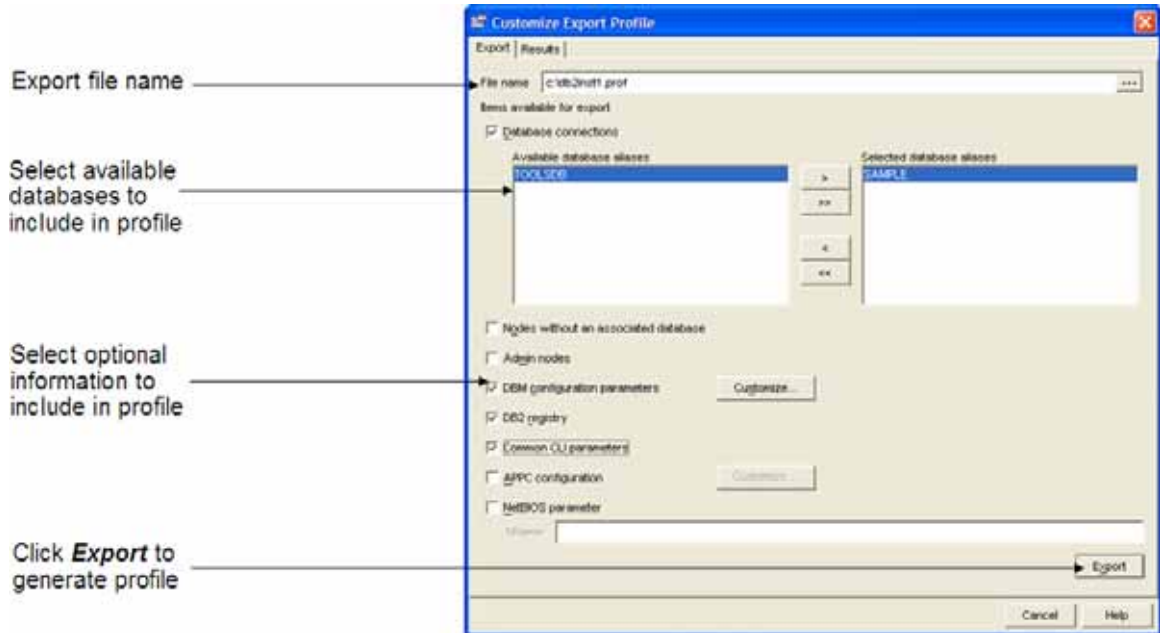


Figure 7.11 – Customize Export Profile dialog

Figure 7.12 show the results after clicking “Export” in the Customize Export Profile dialog.

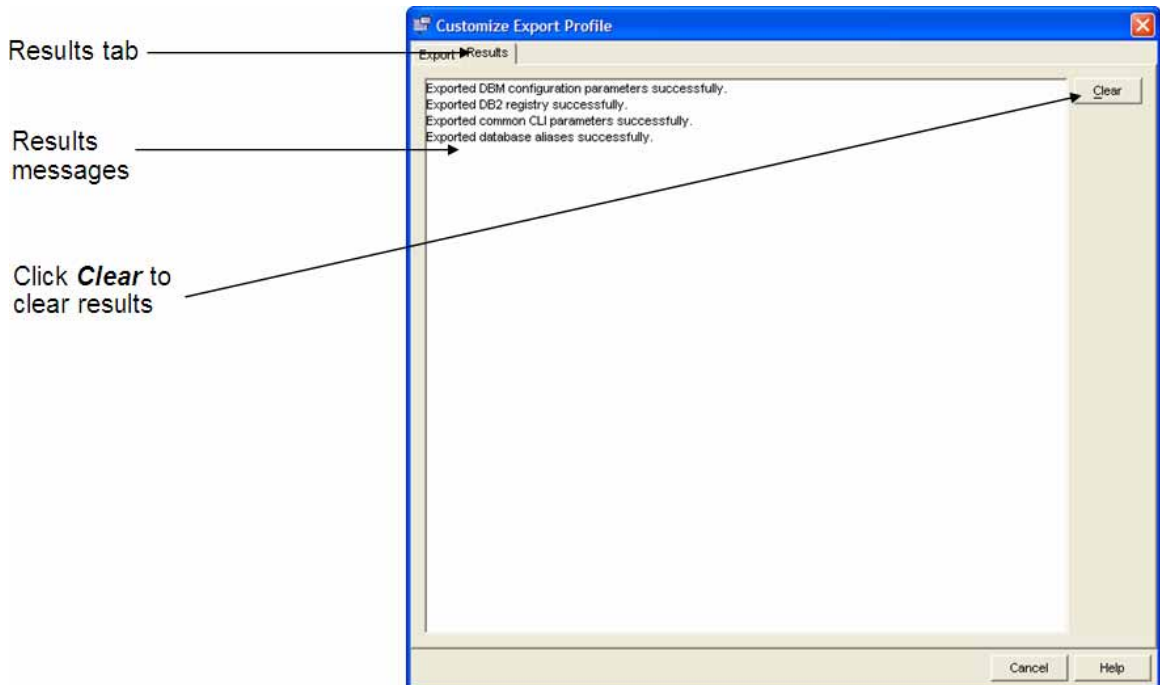


Figure 7.12 – Export Profile results

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

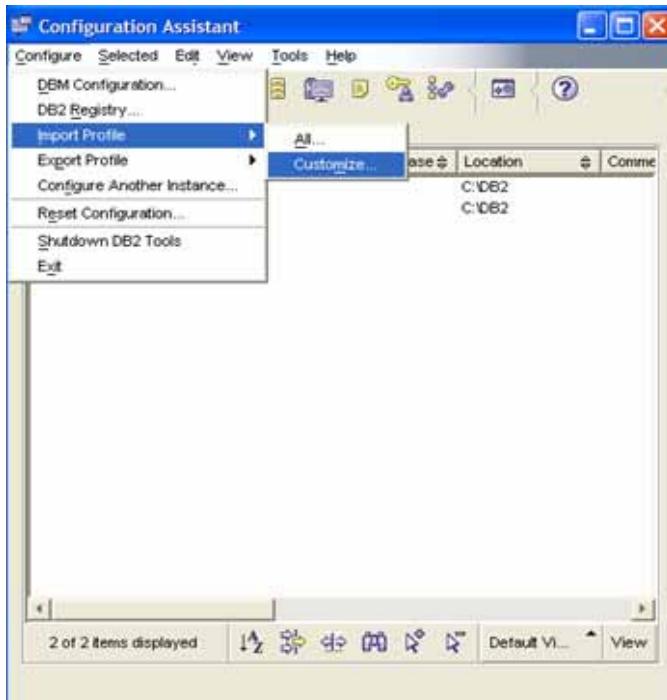


Figure 7.13 – Importing a profile

Figure 7.14 shows the fields that need to be completed to import a profile

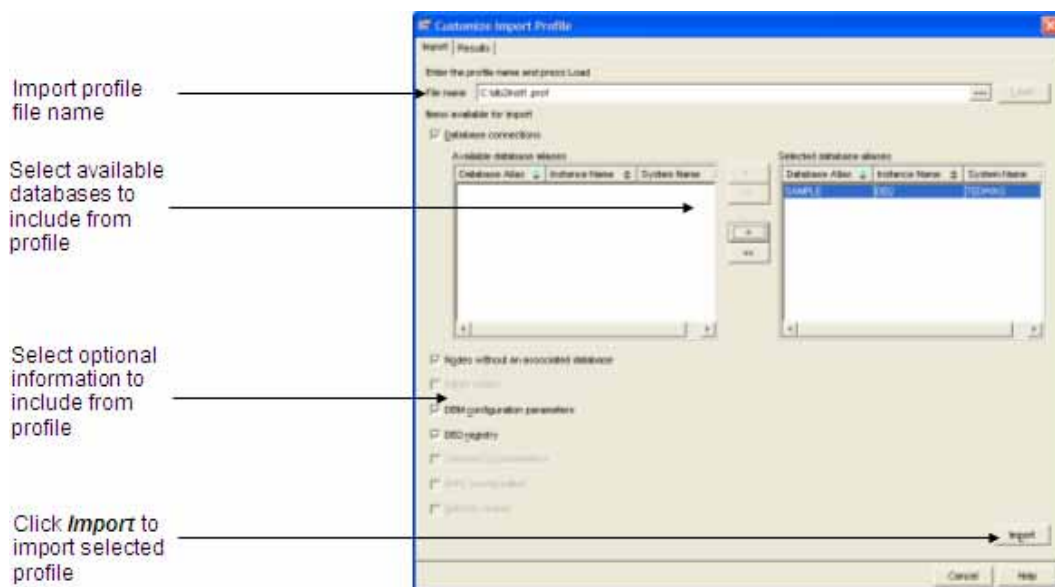


Figure 7.14 – Customize Import Profile





6. 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.

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 local machine, not on the server. You must add an entry to this file if you want to use this option.

7. 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.
8. 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.
9. 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.
10. 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.
11. 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 ap-

pear. 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).

12. Open Control Center and try viewing the different tables in the newly cataloged remote database.
13. 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, SQL PL stored procedures, and Chapter 15, Inline SQL PL, UDFs, Triggers.

### 8.1 Schema

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 have a two part fully qualified name; the schema is the first half of that name:

<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 a table using the CREATE TABLE statement:

```
CREATE TABLE artists ...
```

The fully qualified name of the created table is actually `arfchong.artists`.

### 8.2 Tables

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

```

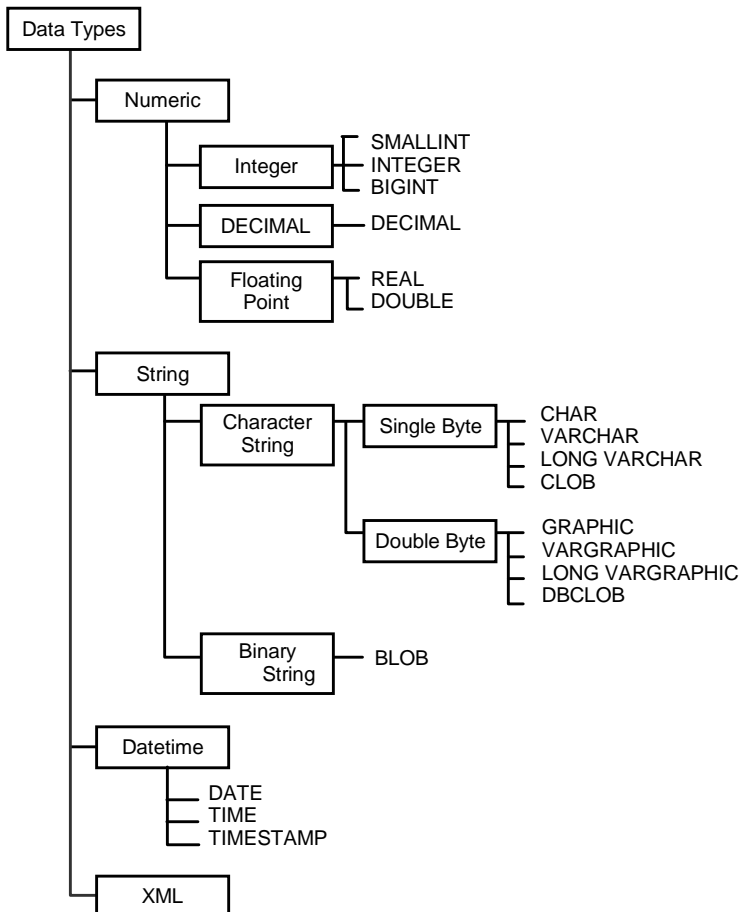
CREATE TABLE artists
(artno          SMALLINT    not null,
 name          VARCHAR(50)  with default 'abc',
 classification CHAR(1)    not null,
 bio          CLOB(100K)   logged,
 picture      BLOB(2M)    not logged compact
)
IN mytbls1

```

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

### 8.2.1 Data Types

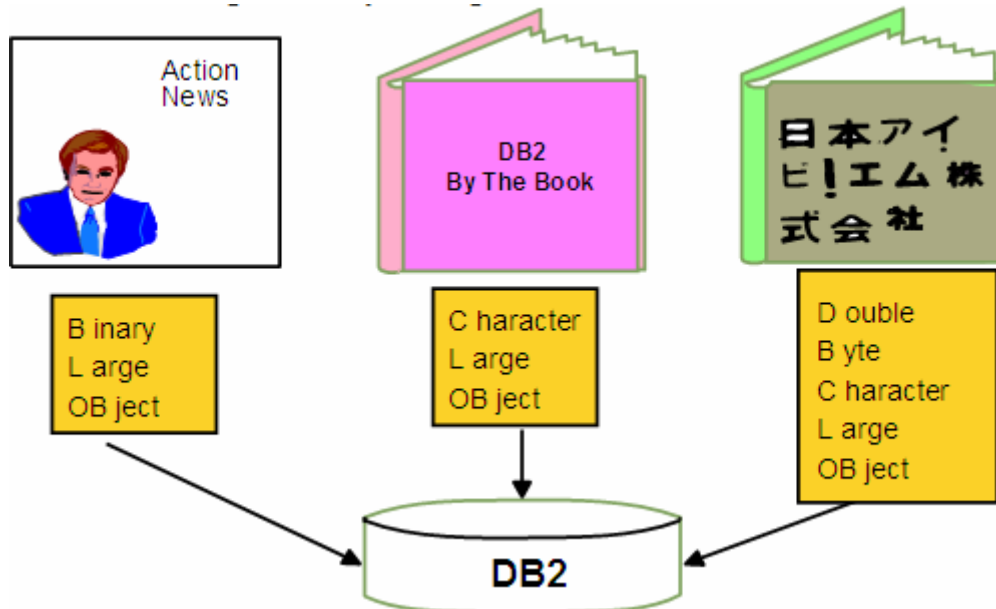
Figure 8.1 lists the data types supported in DB2



**Figure 8.1 – DB2 built-in data types**

#### Large Object (LOB) data types

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.

### User-defined types

DB2 allows you to define your own data types, based on the built-in data types. These are known as user-defined types (UDTs). UDTs are useful when:

- there is a need to establish context for values
- there is a need to have DB2 enforce data typing

The following statements illustrate an example of how and when to use 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 )
```

In this example, two 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, what would a comparison between pounds and kilograms mean? It 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 invoking the `WITH COMPARISONS` clause of the `CREATE DISTINCT TYPE` statement.

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

### Null Values

A null value represents an unknown state. However, the `CREATE TABLE` statement can define a column using the `NOT NULL` clause. This ensures 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
)
```

### 8.2.2 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.

### 8.2.3 SEQUENCE objects

Sequence objects generate a unique number across the database. Unlike identity columns, sequences are independent of tables. 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
```

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 a query requires to return an output based on only one value. System catalog tables are described in the next section.

#### 8.2.4 System catalog tables

Each database has its own system catalog tables and views. These store *metadata* about the database objects. 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

#### 8.2.5 Declared temporary tables

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. No entry exists in any DB2 catalog table. 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 following statements 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 tempemployees
  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;
```

When a declared temporary table is created, its schema is SESSION, and must be specified. The user ID used to create a temporary table will have all privileges on the table. Each application which creates a temporary table will have its own independent copy as shown in Figure 8.5.



Figure 8.5 – Scope of declared global temporary tables

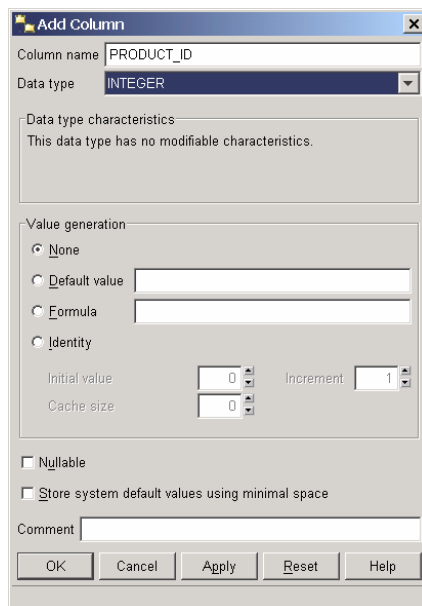
## Quicklab #5 – Creating a new table

### Objective

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 a database. In this Quicklab, 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 presentation. (Control Center => All Databases => *SAMPLE* => (right-click) *Tables* object => *Create ...* option)
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.



Enter the column name “*product\_id*” and select the data type: *INTEGER*. Uncheck *Nullable*, and click the *Apply* button to define the column.

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 no-

tice that the image column is the only one defined as a “NULLABLE” column. Why do you think that the column was defined like this?

5. At this point, all the mandatory information for creating a table has been provided. By skipping the other pages, you are choosing the default values for those options. You can always add keys and constraints after a table has been created.
6. 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.

7. 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.
8. 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.

## 8.3 Views

A view is a representation of the data in tables. The data for the view is not stored separately, but obtained when the view is invoked. Nested views, that is, a view created based on other views, are supported. All information about views is kept in the following DB2 catalog views: SYSCAT.VIEWS, SYSCAT.VIEWDEP, and SYSCAT.TABLES. Here is 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
...		

## 8.4 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)
```

### 8.4.1 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 8.6.

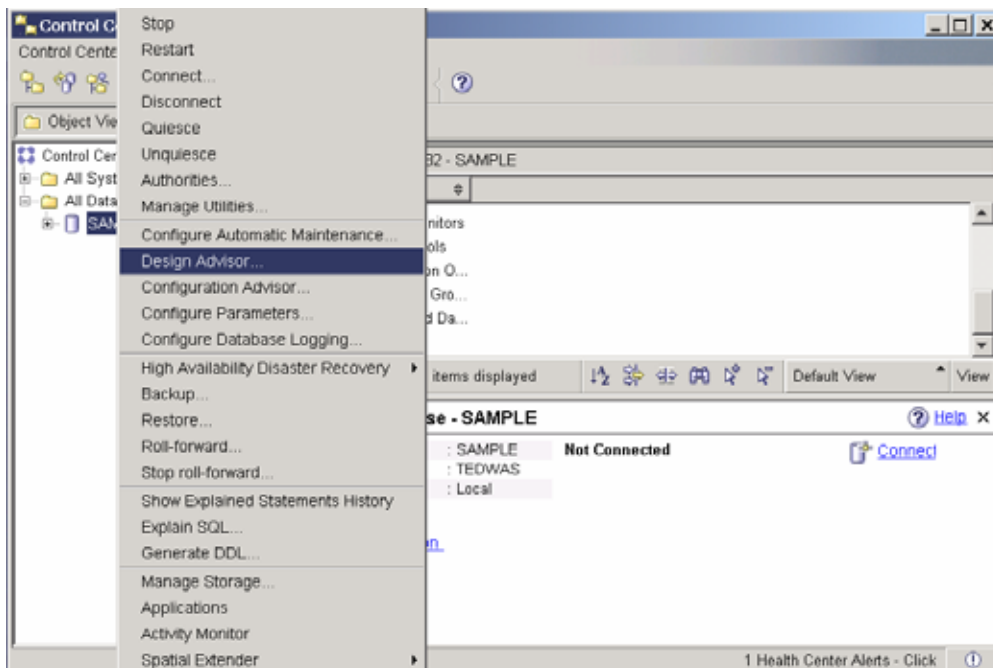


Figure 8.6 – Invoking the Design Advisor from the Control Center

Figure 8.7 shows the Design Advisor. Follow the steps in this wizard to obtain the design recommendations from DB2.



Figure 8.7 –The Design Advisor



## 8.5 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 8.8. 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.

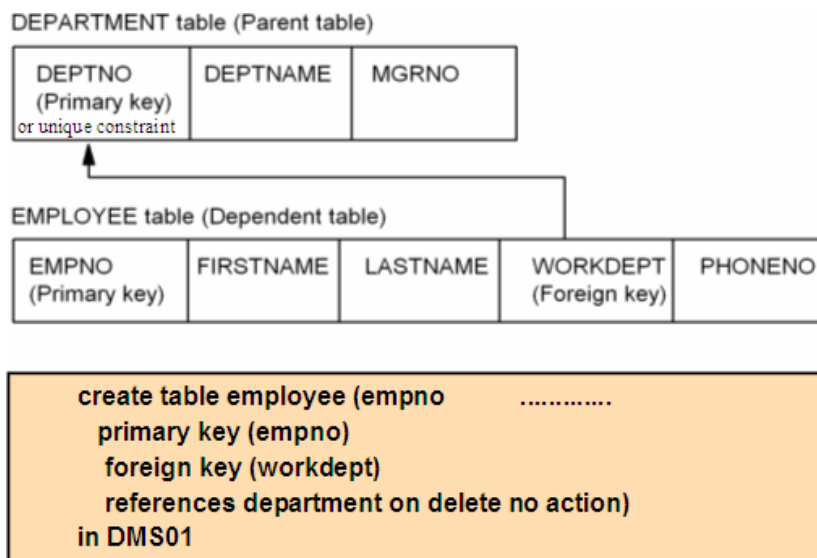


Figure 8.8 –An example of referential integrity between tables

In referential integrity, the following concepts 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 foreign key of a parent table

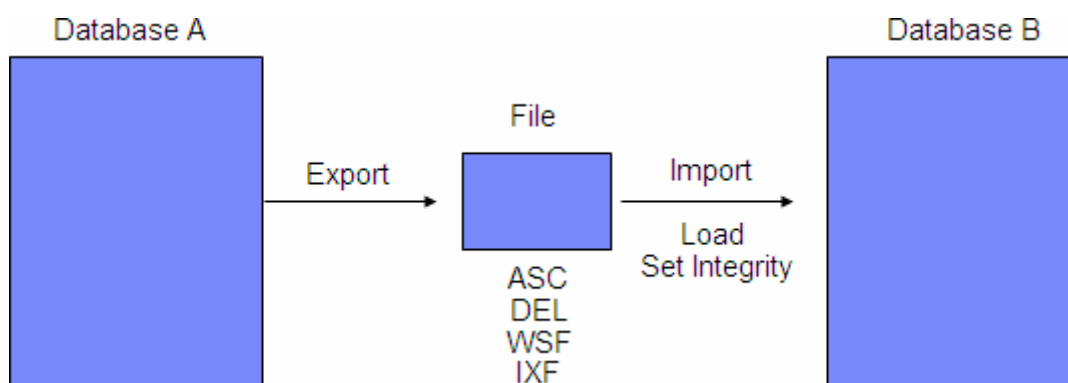
Data in tables can be related to data in one or more tables with referential integrity. Constraints can also be imposed on data values so that they conform to a certain property or business rule. For example, if a table column stores the sex of a person, the constraint can enforce that the only values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



# 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 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. This is convenient because when the table needs to be reconstructed, it can be done directly from a file with an IXF format, while this would not be 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 will not be 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.

## 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.

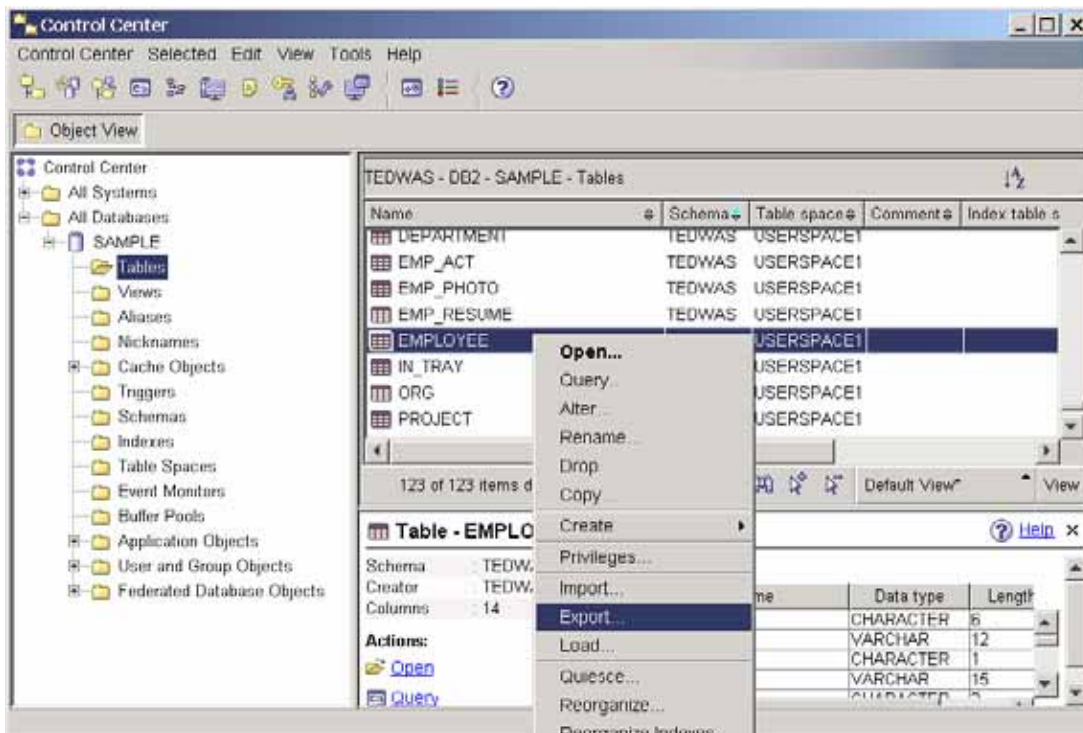


Figure 9.2 – Launching the EXPORT table dialog

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*. The REPLACE\_CREATE option is one of many options available with the IMPORT utility. This option will replace 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. We encourage you to try the example below, but you need to have run the EXPORT utility in the previous section.

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

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

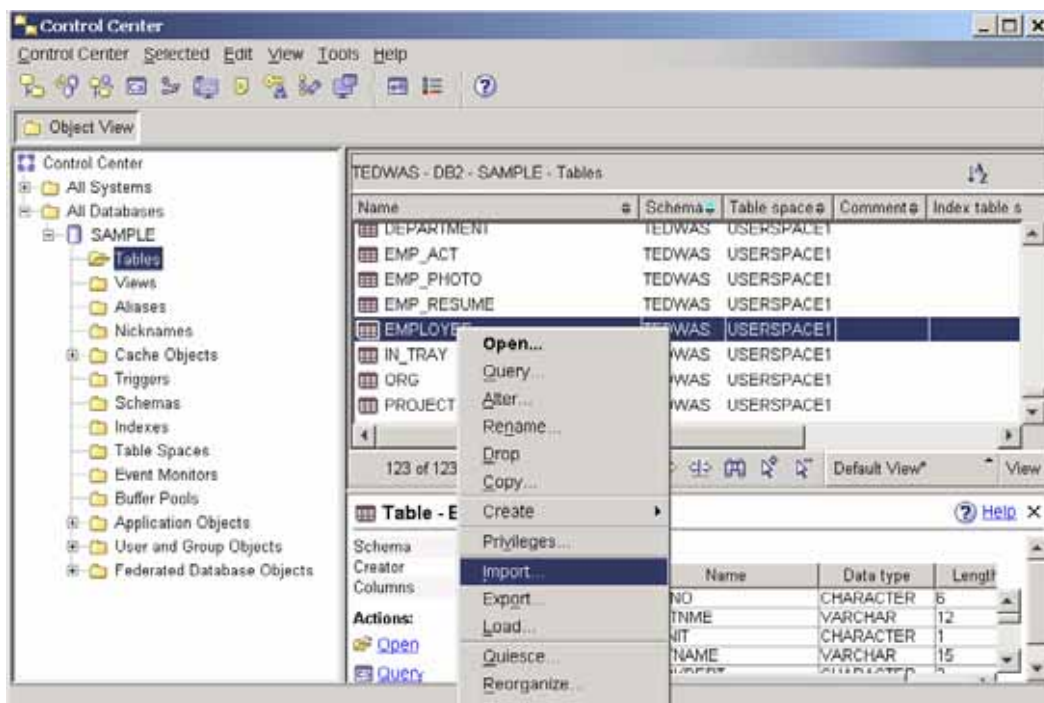


Figure 9.3 – Launching the IMPORT dialog

### 9.3 LOAD

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, so 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 the contents of the *employee\_copy* table.

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

After executing the above command (which you can try out), the table space where your table resides may have been place 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.

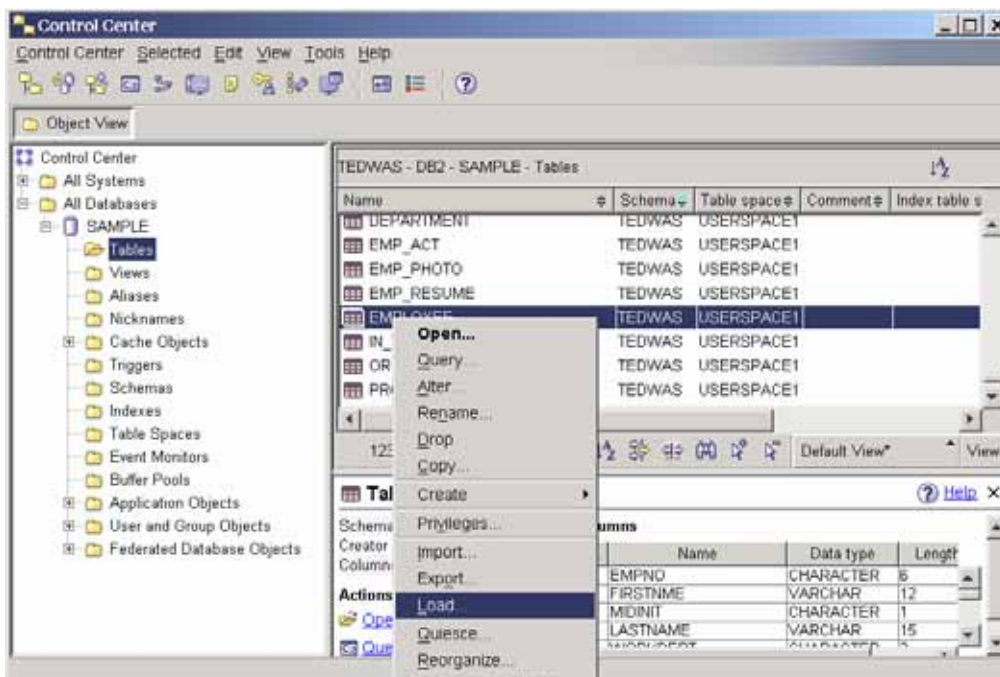


Figure 9.4 – Launching the LOAD wizard

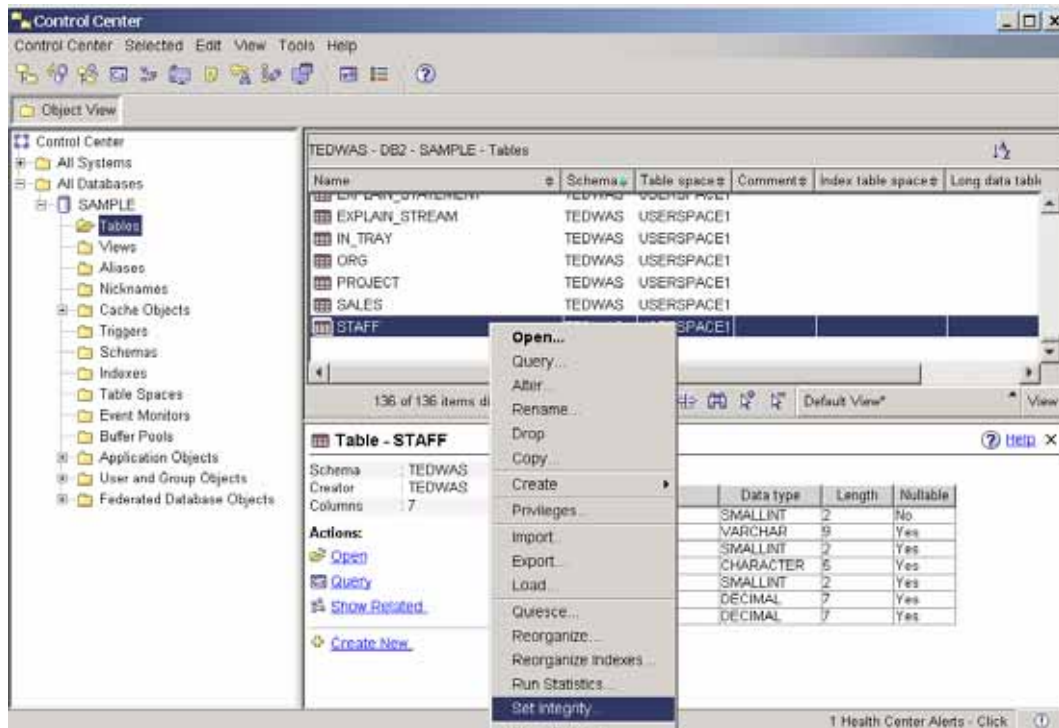


Figure 9.5 – Launching the SET INTEGRITY wizard

## 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*.

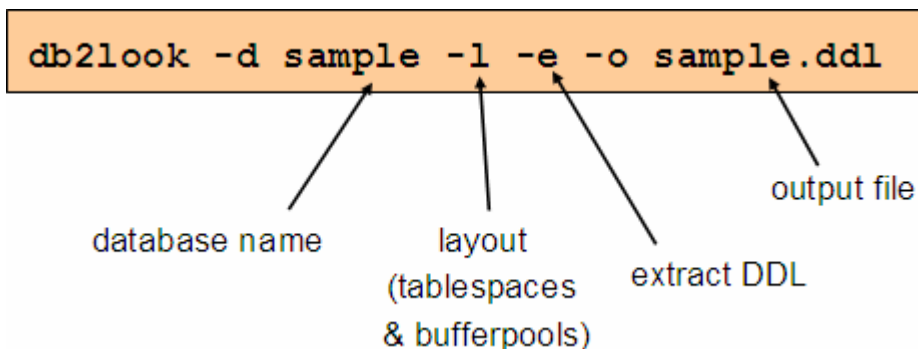
## 9.5 The db2look utility

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 would like to clone a database from a DB2 server running on Linux to a

DB2 server running on Windows; you could 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 the import or load options. Once this is done, your database would be fully cloned from one server to another on different platforms.

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



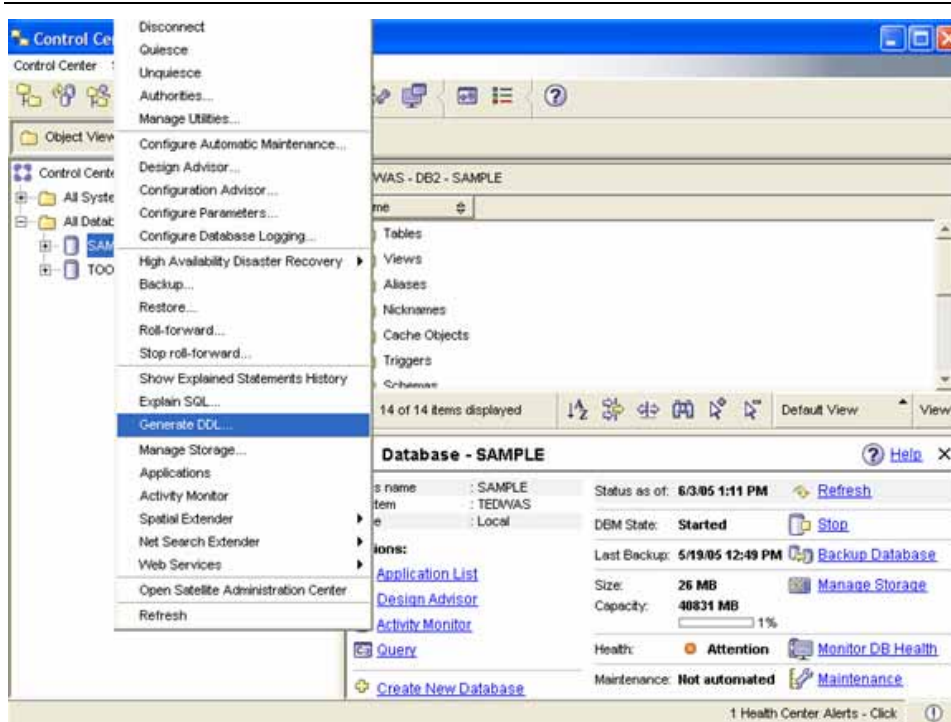


Figure 9.6 - Extracting DDL from the Control Center

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.

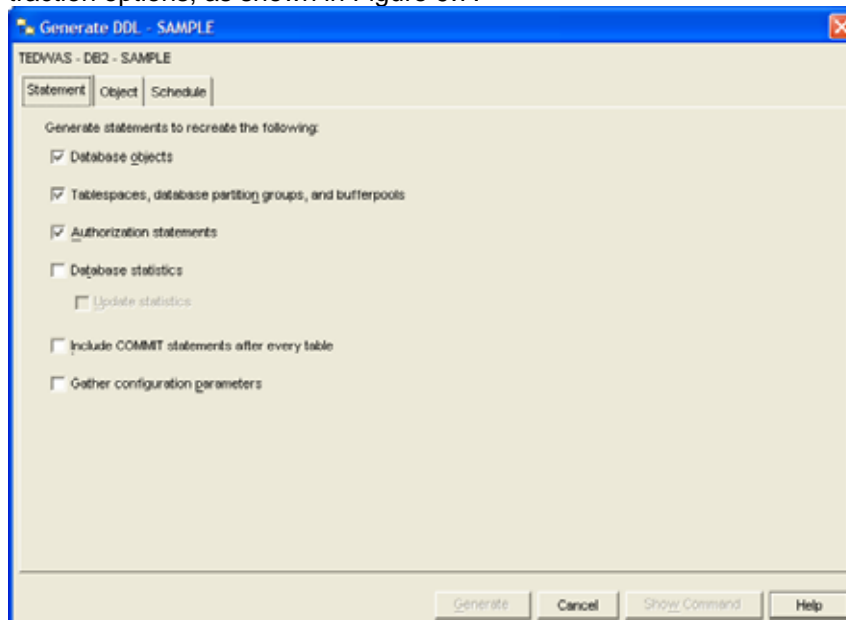


Figure 9.7 - Extracting DDL from the Control Center

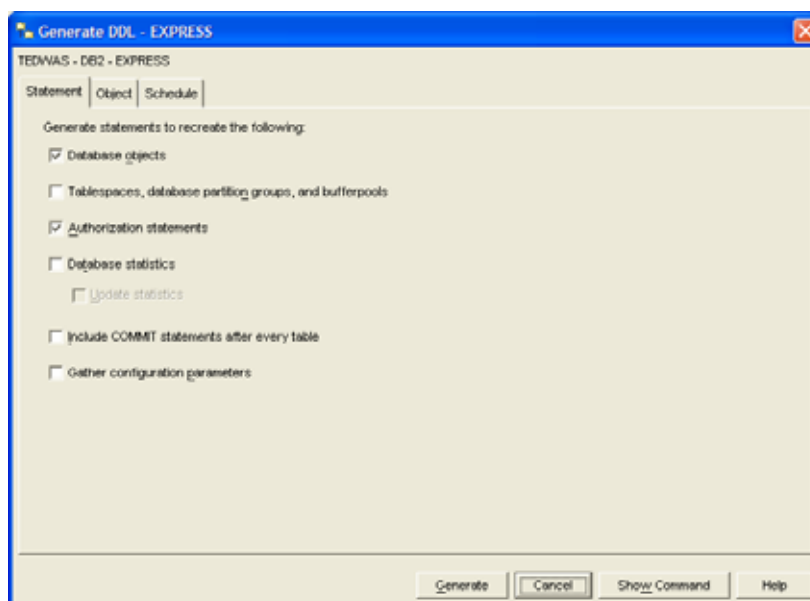
## Quicklab #6 – Extracting DDL for the EXPRESS database

### Objective

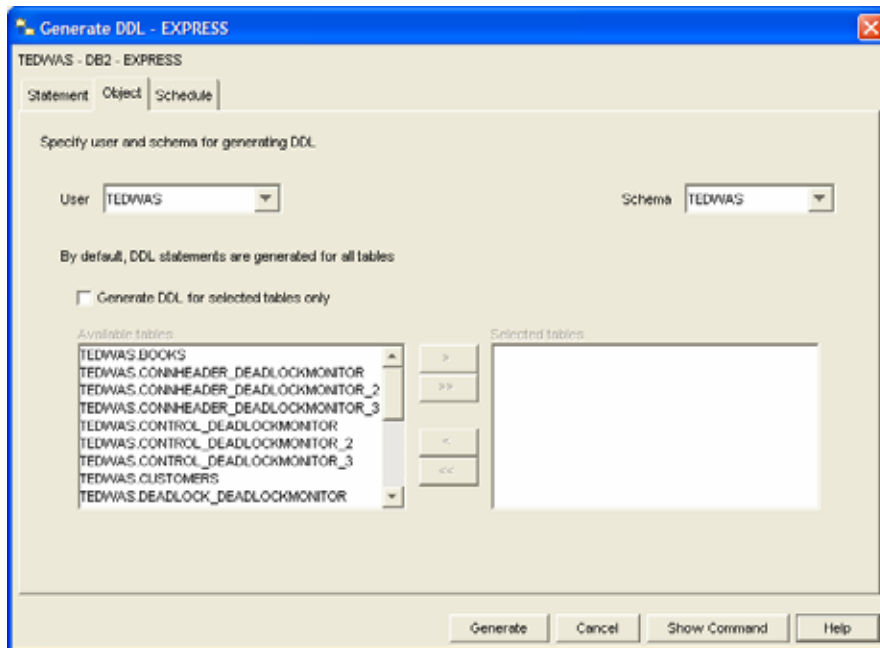
When you duplicate a database, your goal should be to make the re-creation of the database as straightforward and repeatable as possible. This is usually done using SQL scripts, which can be immediately executed after DB2 has been installed. In this Quicklab, you will extract the object definitions from the *EXPRESS* database (created in Quicklab #3) using the Control Center.

### Procedure

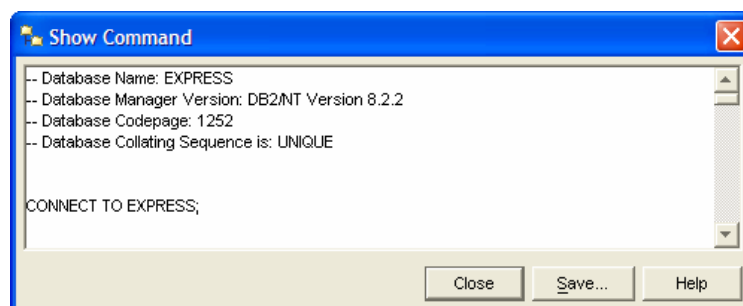
1. Open the Control Center.
2. Right-click on the *EXPRESS* database in the object tree and select the *Generate DDL* menu item. This launches the *Generate DDL* dialog window.
3. In the *Generate DDL* window, specify options for the generated DDL, as shown below. If you created additional objects in 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 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:

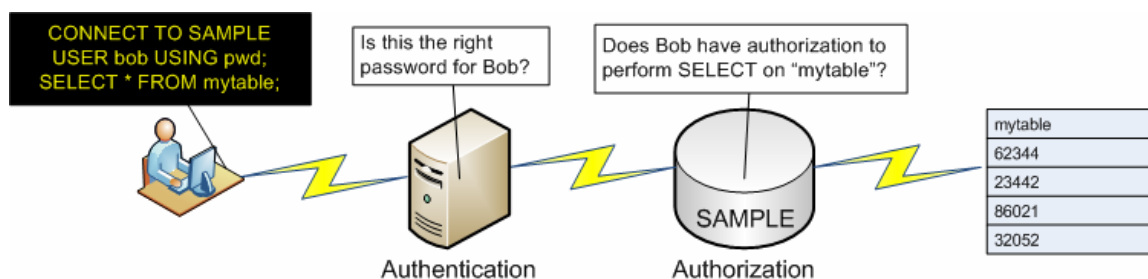
<code>C:\express\schema.ddl</code>	DDL for tables, views, indexes, and constraints
<code>C:\express\triggers.ddl</code>	DDL for triggers

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 (typically by an operating system, some network authentication method, or a custom-built authentication plug-in). OS-based authentication is the default. When using OS-based authentication, the userid and password are flown 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 “psw” 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 op-

erating system will return security control to DB2. Next, when user “bob” executes a statement such as:

```
SELECT * FROM mytable
```

Now 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”.

### 10.1 Authentication

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

The DBM CFG 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/external security facility on the server. However, if AUTHENTICATION is set to CLIENT, the authentication is performed by the operating system/external security facility at the client. This is shown in Figure 10.2.

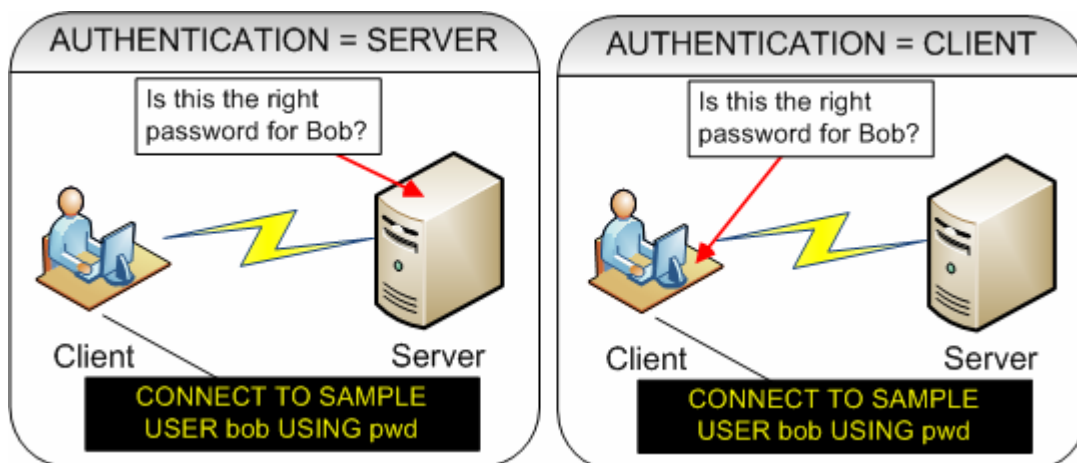


Figure 10.2 – Where authentication takes place

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 and authorities 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.

An authority is a predefined role consisting of several privileges. Figure 10.3 shows the different authorities and privileges in DB2.

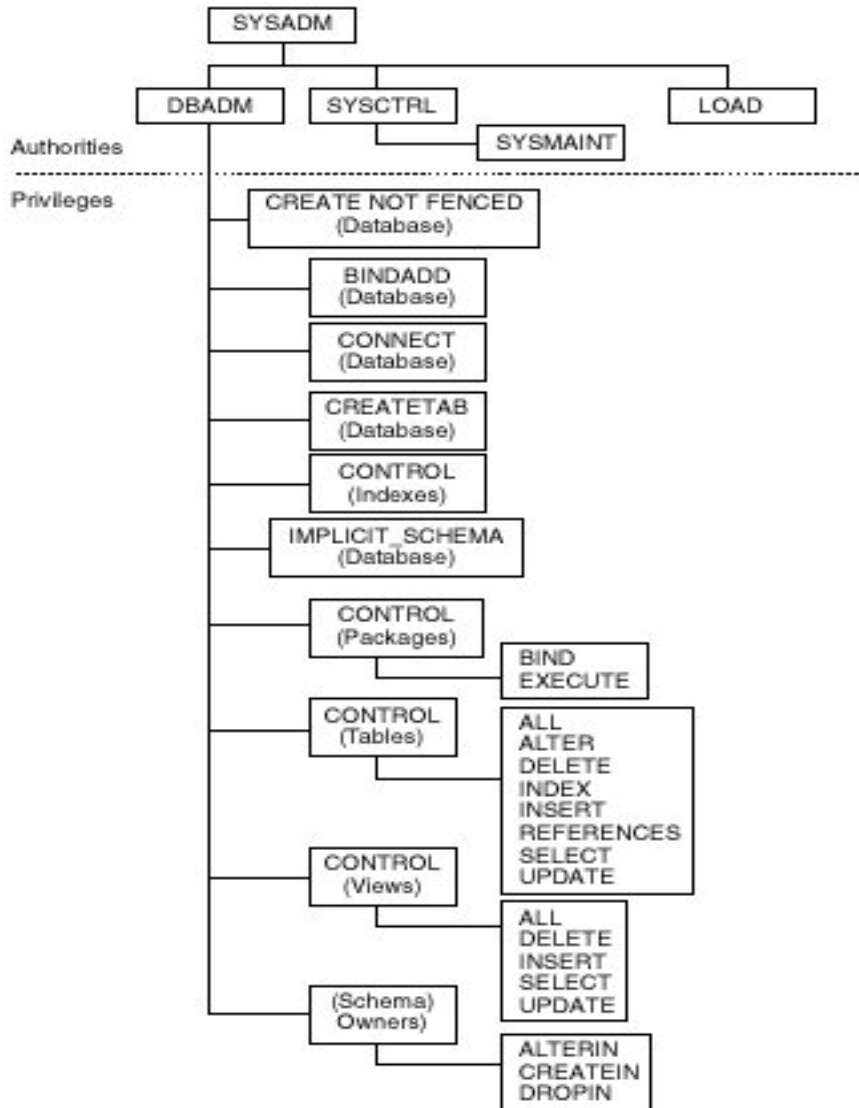


Figure 10.3 – Authorities and privileges



Table 10.2 shows the different functions that each authority can perform. As you can see, SYSADM has the most authority while SYSMON has the least.

<i>Function</i>	<i>SYSADM</i>	<i>SYSCTRL</i>	<i>SYSMAINT</i>	<i>SYSMON</i>	<i>DBADM</i>	<i>LOAD</i>
<b>Update DBM CFG</b>	Y					
<b>Grant/revoke DBADM</b>	Y					
<b>Establish/change SYSCTRL</b>	Y					
<b>Establish/change SYSMAINT</b>	Y					
<b>Establish/change SYSMON</b>	Y					
<b>Force users off database</b>	Y	Y				
<b>Create/drop database</b>	Y	Y				
<b>Restore to new database</b>	Y	Y				
<b>Update DB CFG</b>	Y	Y	Y			
<b>Backup database/table space</b>	Y	Y	Y			
<b>Restore to existing database</b>	Y	Y	Y			
<b>Perform roll-forward recovery</b>	Y	Y	Y			
<b>Start/stop instance</b>	Y	Y	Y			
<b>Restore table space</b>	Y	Y	Y			
<b>Run trace</b>	Y	Y	Y	Y		
<b>Obtain monitor snapshots</b>	Y	Y	Y			
<b>Query table space state</b>	Y	Y	Y			
<b>Prune log history files</b>	Y	Y	Y			
<b>Quiesce table space</b>	Y	Y	Y		Y	Y
<b>LOAD tables</b>	Y				Y	Y
<b>Set/unset check pending state</b>	Y				Y	
<b>Create/drop event monitors</b>	Y				Y	

**Table 10.2 - DB2 authorities and privileges**

In order to grant SYSADM, SYSCTRL or SYSMANT authority to a group, the DBM CFG parameters SYSADM\_GROUP, SYSCTRL\_GROUP, and SYSMANT\_GROUP can be assigned to an operating system group.

For example, to give SYSADM authority to the operating system group “db2admns”, you can issue this command:

```
update dbm cfg using SYSADM_GROUP db2admns
```

Each DB2 instance has its own authority group definitions.

On Windows, these parameters are empty by default, which means the local Windows Administrators group will be SYSADM. On Linux the instance owner group is the default SYSADM group.

### 10.3 DBADM authority

The DBADM (DataBase ADMinistrator) authority is a super user for the database. It is not an authority at the instance level; therefore it is not listed in the previous section. In order to grant DBADM authority, use the GRANT statement as shown in the example below.

```
connect to sample
grant DBADM on database to user <userid>
```

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 you need to be SYSADM

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

### 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 standards, and are used to give or remove privileges to a user or group. 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 10.4 shows how to launch the Table Privileges dialog for the EMPLOYEE table from the Control Center.

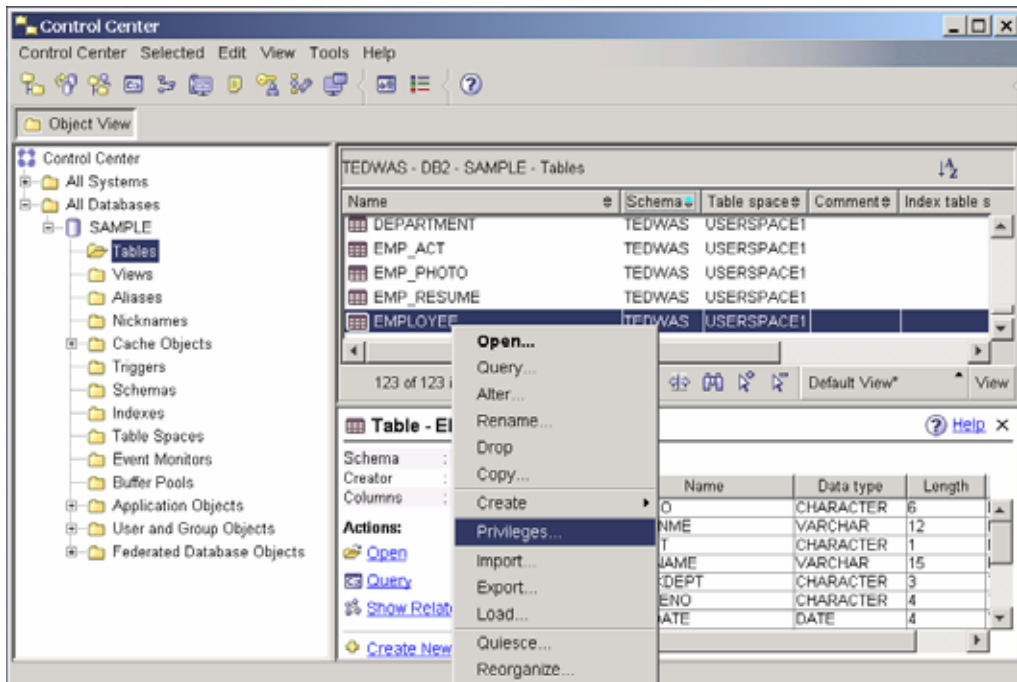


Figure 10.4 - Launching the Table Privileges dialog

As shown by Figure 10.4, you select the desired table, right-click on it, and choose Privileges. Once selected, the Table Privileges dialog box appears as shown in Figure 10.5. This figure also explains the different fields and elements of the dialog box.

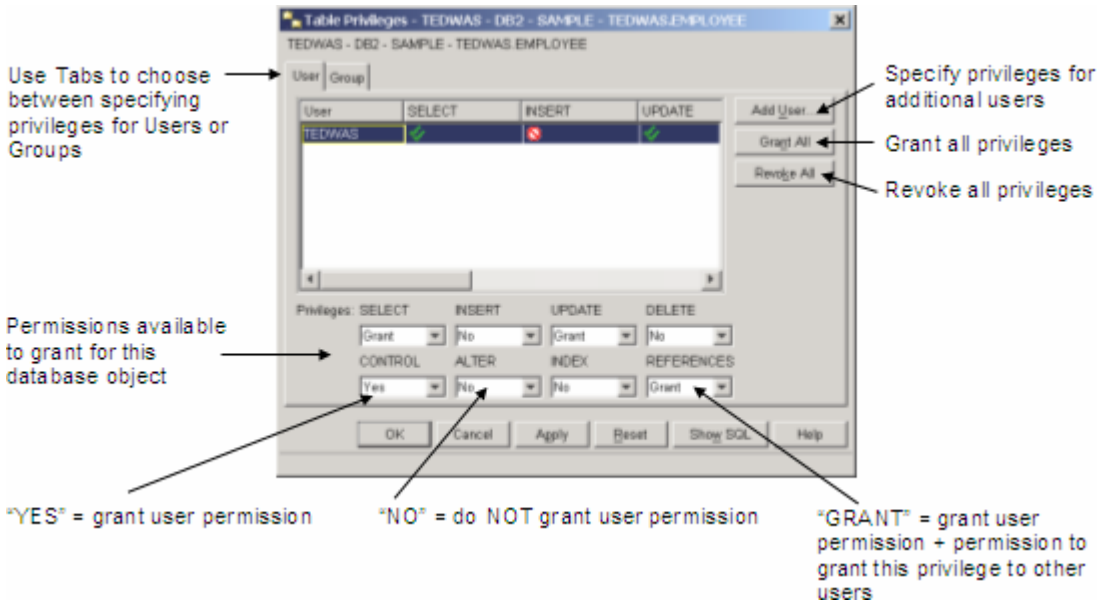


Figure 10.5 – The Table Privileges Dialog box

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 Group privilege considerations

To make DB2 administration easier, place users into groups, and then grant those groups the required privileges.

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.

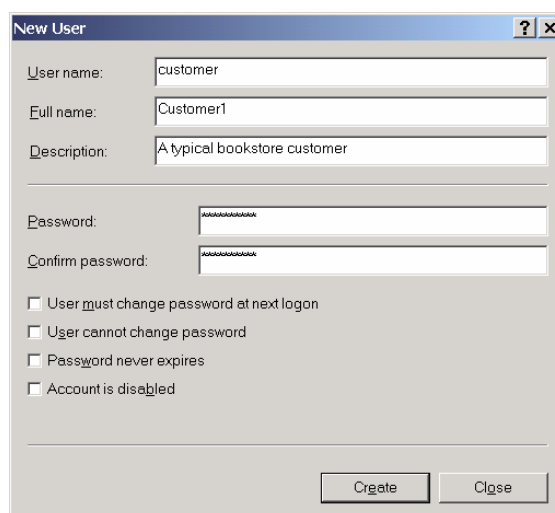
## Quicklab #7 – Granting and revoking user permissions

### Objective

So far, you have been using the instance administrator account (SYSADM) to issue all the database commands. This account has full 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 lab, you will create a new user account, then assign it specific privileges.

### Procedure

1. Open the Windows Computer Management console by right-clicking on the *My Computer* icon on the desktop, and selecting the *Manage* menu item.
2. 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.
3. 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 “ibmdb2”. Remove the checkmark from the *User must change password on next logon* option, and click the *Create* button to create the new user.

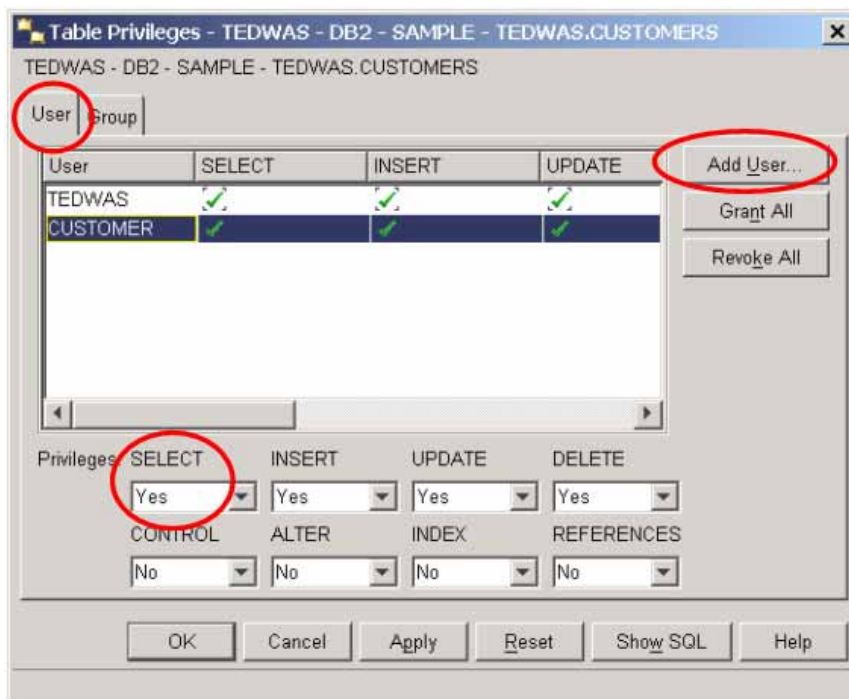


The screenshot shows the 'New User' dialog box with the following fields and options:

- User name: customer
- Full name: Customer1
- Description: A typical bookstore customer
- Password: ibmdb2
- Confirm password: ibmdb2
- User must change password at next logon
- User cannot change password
- Password never expires
- Account is disabled

Buttons: Create, Close

4. Ensure the advanced view is being used in the DB2 Control Center. To switch to the advanced view, select the *Customize Control Center* menu item from the Control Center *Tools* menu. Select the *Advanced* option and click the *OK* button.
5. Expand the Control Center object tree in the left object tree pane to *All Databases > EXPRESS > Tables*.
6. 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.
7. Click the *Add User* button and select the *customer* user just created. Click the *OK* button to close the *Add User* dialog box.
8. 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.



9. 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.
  
10. Connect to the database using the *customer* user ID created above. Try to SELECT data from the customers table. What happens? Try to DELETE or UPDATE data in the *SALES* table. What happens?

*In this Quicklab, we only created one user; however, your application may contain many different types of users. Experiment with creating other users and assigning them privileges. You can also create groups of users and assign privileges to those groups, rather than to each individual user specifically.*



# 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.

### 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.

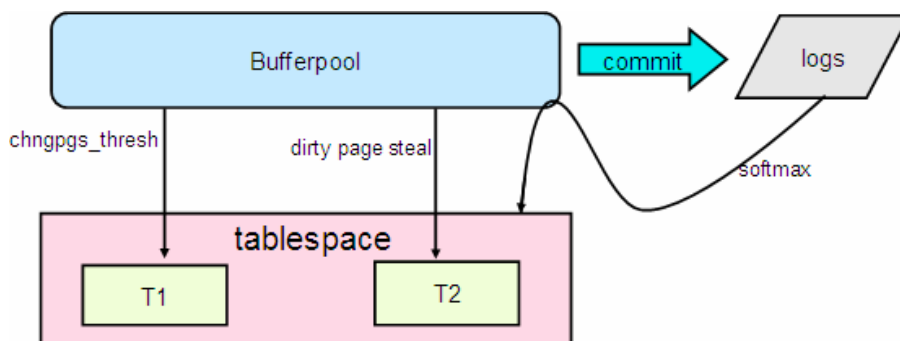


Figure 11.1 – Database logging

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 will be 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 parameter CHNGPGS\_THRES, will the pages in the buffer pool be “externalized” or 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 db cfg parameter.

### Secondary logs

These are dynamically allocated when needed by DB2. The maximum number of secondary logs is set by the db cfg parameter LOGSECOND. Dynamically allocating a log is costly; therefore, for day to day operations, ensure that you stay within your primary log allocation. Secondary log files are deleted when all the connections to a database are terminated.

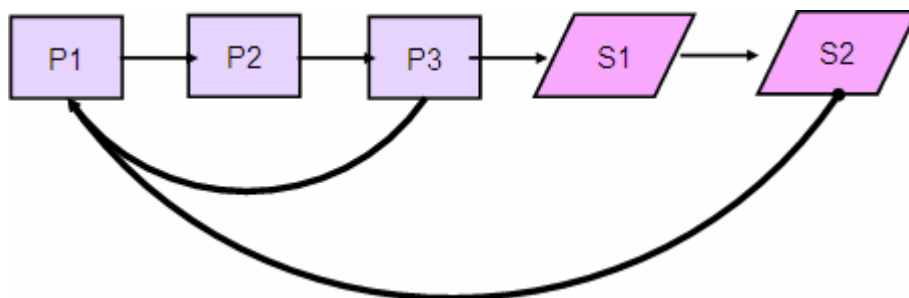
Infinite logging is possible if you set LOGSECOND to a value of -1; however, this is not recommended, and you may run out of file system space.

## 11.3 Types of logging

There are two types of logging: circular logging (default) and archive logging.

### 11.3.1 Circular logging

Figure 11.2 demonstrates how circular logging works.



**Figure 11.2 – Working with primary and secondary logs**

In Figure 11.2 there are 3 primary logs, therefore we can assume that LOGPRIMARY was set to 3. For simplicity, let's just say there is only one transaction being performed in this example. As the transaction is performed, space starts filling up the log file P1, 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) would be dynamically allocated. If the transaction still continues, more secondary logs are allocated until the maximum LOGSECOND logs are allocated. If 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.

### 11.3.2 Archival logging or log retain

In archival logging, also known as log retain logging, the log files are not overwritten, but are kept online or offline. Online archive logs are kept 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. To enable archival logging set the LOGRETAIN db cfg parameter to YES.

Archival logging is normally used in production systems, and because the logs are kept, it allows for database recovery back to as early as the oldest log file in most situations. With archival logging, a DBA can recover (to some extent) from errors caused by humans. For example, if a user of a system inadvertently starts performing an incorrect transaction that lasts for days, when the problem is detected later, the DBA could restore the system back to the time before the problem was introduced. However, there may be some manual manipulation required to rerun the transaction correctly.

Archival logging is required for roll forward recovery and on-line backup. Figure 11.3 depicts the archival logging process.

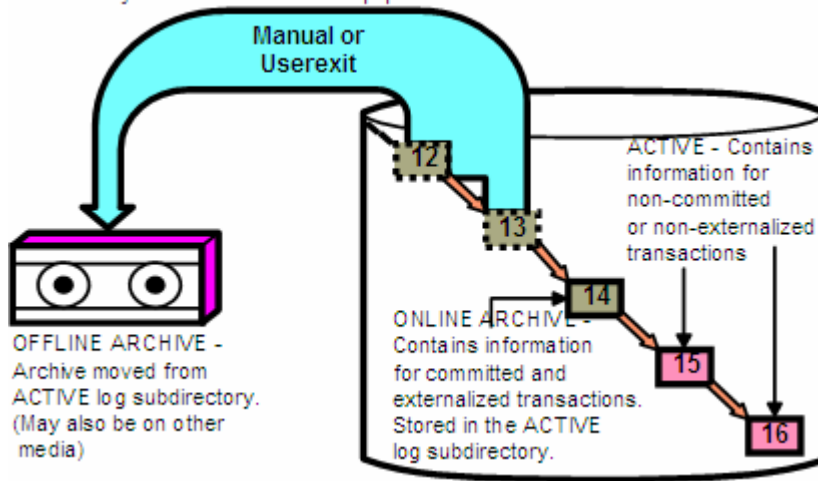


Figure 11.3 – Archival logging

### 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

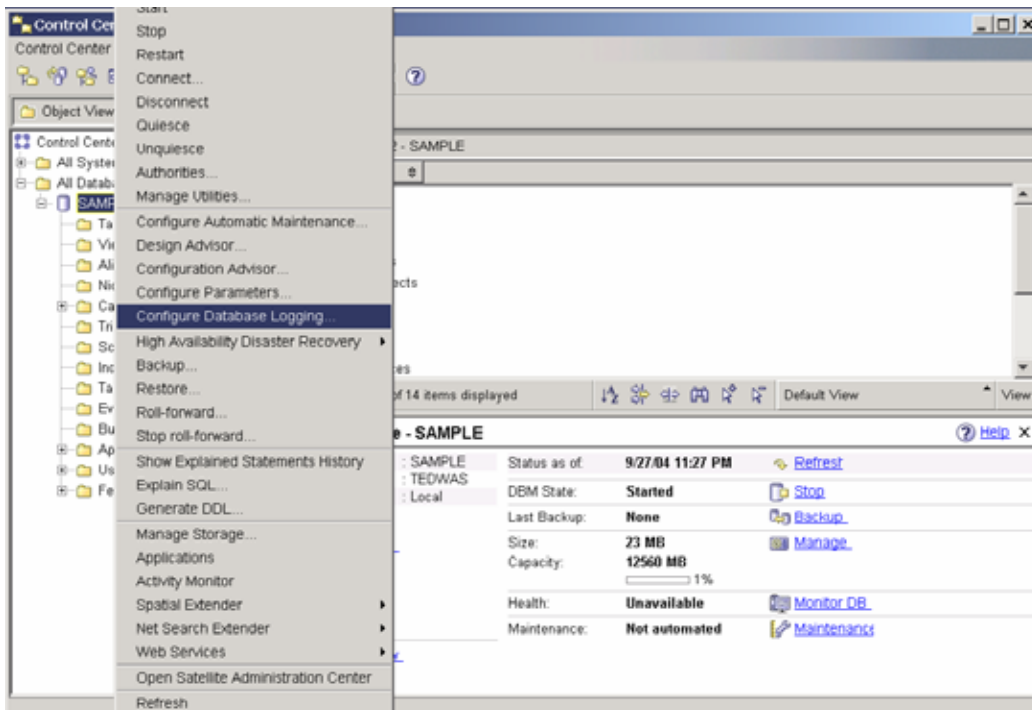


Figure 11.4 – Configuring database logging from the Control Center.

Figure 11.5 shows the Database Logging Wizard, where you can choose circular logging or archival logging.

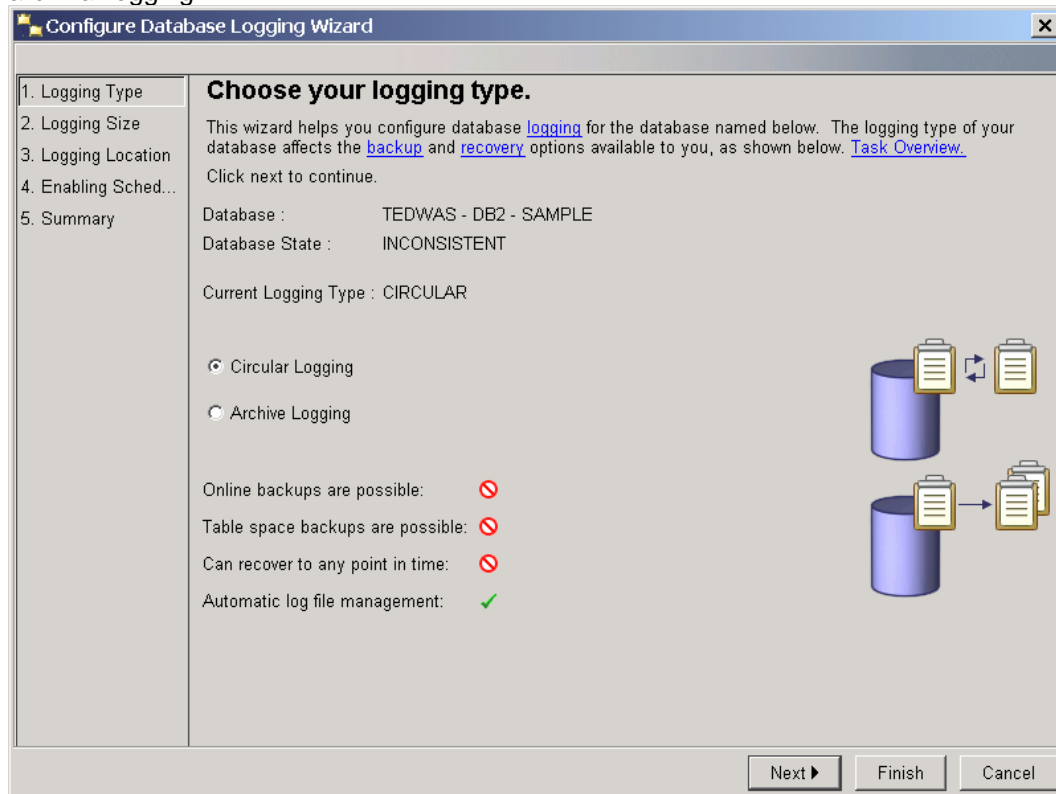


Figure 11.5 – Database Logging Wizard

## 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
logfilisz	The size of each configured log, in number of 4KB pages
logprimary	The number of primary logs of size logfilisz that will be created
logsecond	The number of secondary log files that are created and used for recovery, if needed.
log-path/newlogpath	The location in which active logs and future archived logs are placed.
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
loghead	The name of the log file that is currently active

---

userexit	Enable userexit program to copy logs offline
softmax	Limits cost of crash recovery
logretain	Enables Archive Logging mode
overflowlogpath	Similar to the OVERFLOW LOG PATH option of the ROLLFORWARD command; however, instead of specifying the OVERFLOW LOG PATH option for every ROLLFORWARD command issued, you can set this configuration parameter once.
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 transaction
num_log_span	Number. of active log files for 1 active UOW

**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/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
```

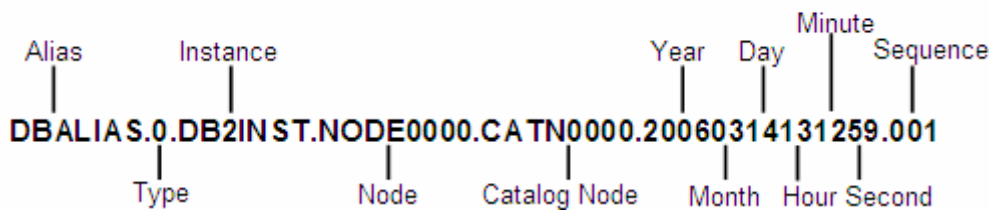
To force all the connections from all databases in an instance, issue this command from the DB2 command window or Linux shell:

```
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 that 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” for example, would mean that it is just 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.

## Quicklab #8 – Scheduling a backup

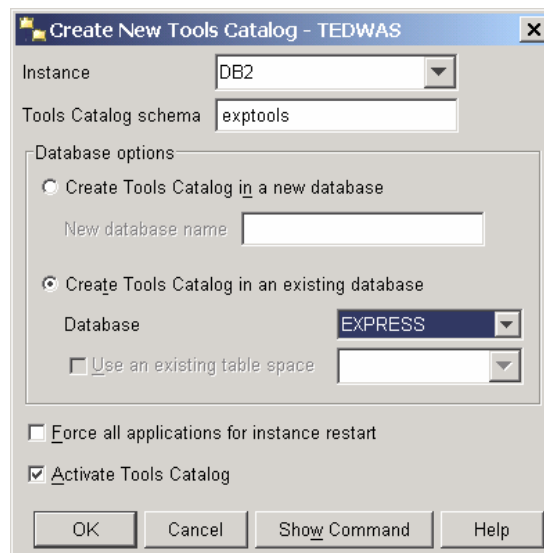
### Objective

Although DB2 is able to automate several database maintenance activities, sometimes you will want to customize when certain activities occur. In this Quicklab, 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.





**Create New Tools Catalog - TEDWAS**

Instance: DB2

Tools Catalog schema: exptools

Database options:

- Create Tools Catalog in a new database
- Create Tools Catalog in an existing database

New database name: [ ]

Database: EXPRESS

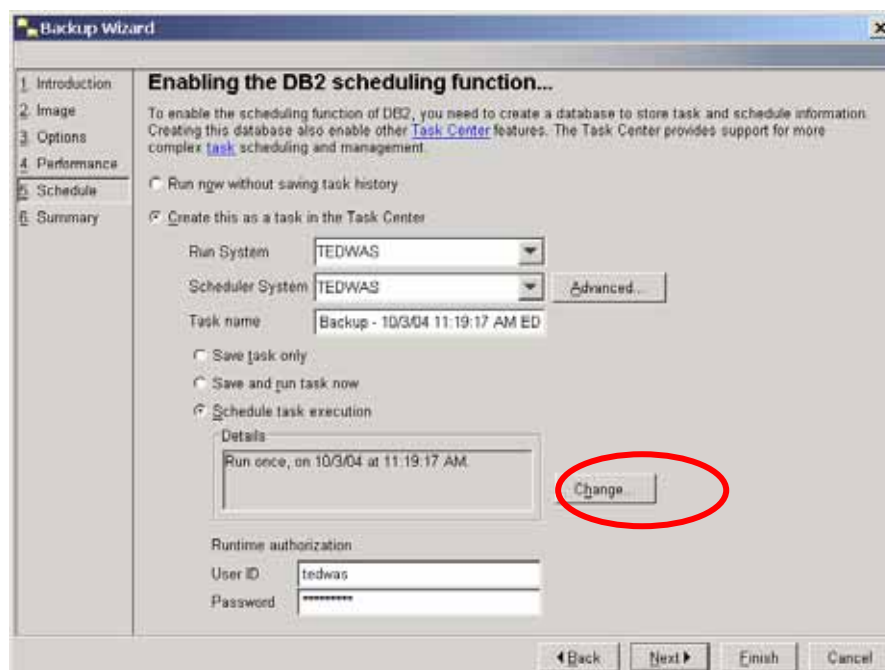
Use an existing table space: [ ]

Force all applications for instance restart

Activate Tools Catalog

OK Cancel Show Command Help

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.



**Backup Wizard**

**Enabling the DB2 scheduling function...**

To enable the scheduling function of DB2, you need to create a database to store task and schedule information. Creating this database also enable other [Task Center](#) features. The Task Center provides support for more complex [task](#) scheduling and management.

- Run now without saving task history
- Create this as a task in the Task Center

Run System: TEDWAS

Scheduler System: TEDWAS [Advanced...]

Task name: Backup - 10/3/04 11:19:17 AM ED

- Save task only
- Save and run task now
- Schedule task execution

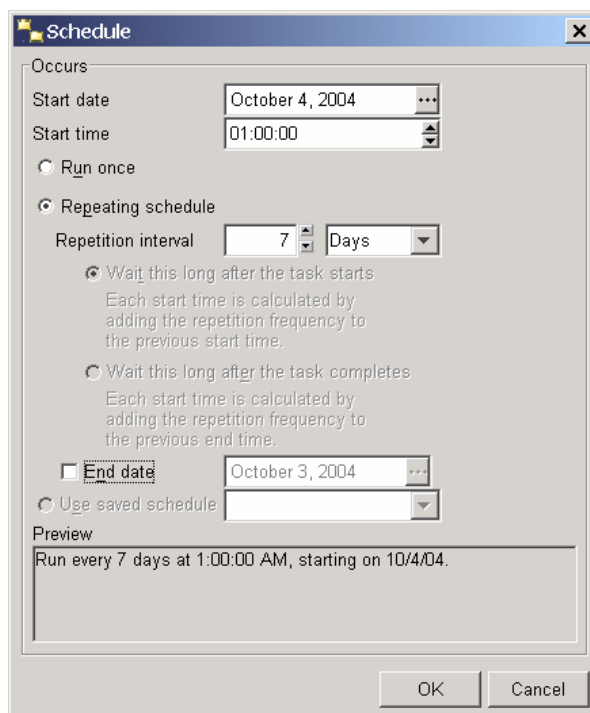
Details: Run once, on 10/3/04 at 11:19:17 AM. [Change...]

Runtime authorization:

User ID: tedwas

Password: [ ]

Back Next Finish Cancel



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.

## 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 archival 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:

Alias Instance Year Day Minute Sequence  
SAMPLE.0.DB2INST.NODE0000.CATN0000.20060314131259.001  
Type Node Catalog Node Month Hour Second

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
- Delta and incremental backups are allowed; delta backups record only the changes from 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 (e.g. Windows) and restoring to another platform (e.g. Linux) is not possible. Use db2look and db2move for this scenario

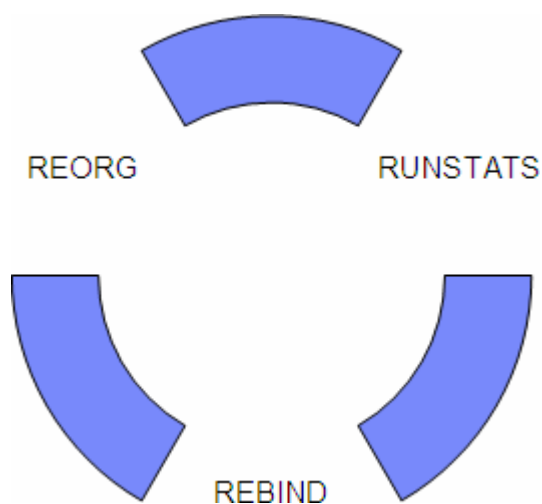
# 12

## 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.

### 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 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>
```

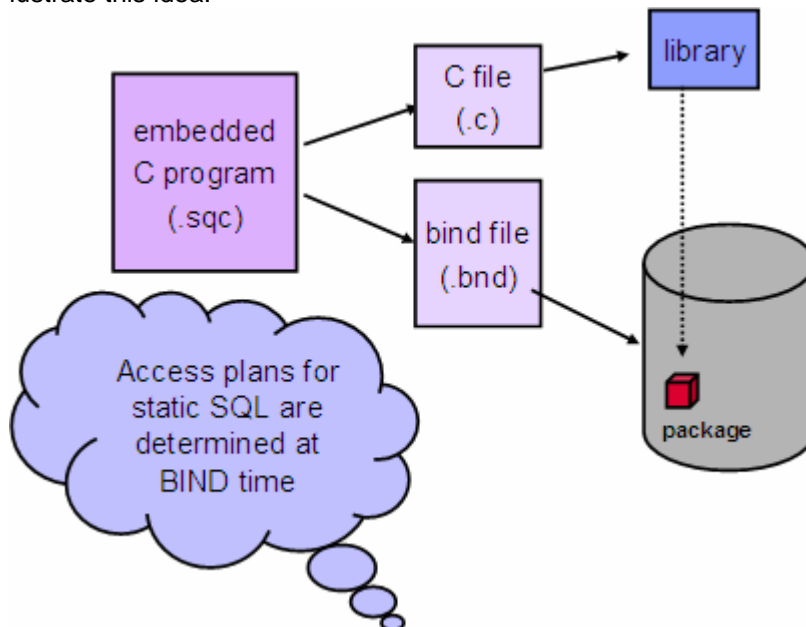
**Example:**

```
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 issue a BIND command, so the

statistics used at the 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>
```

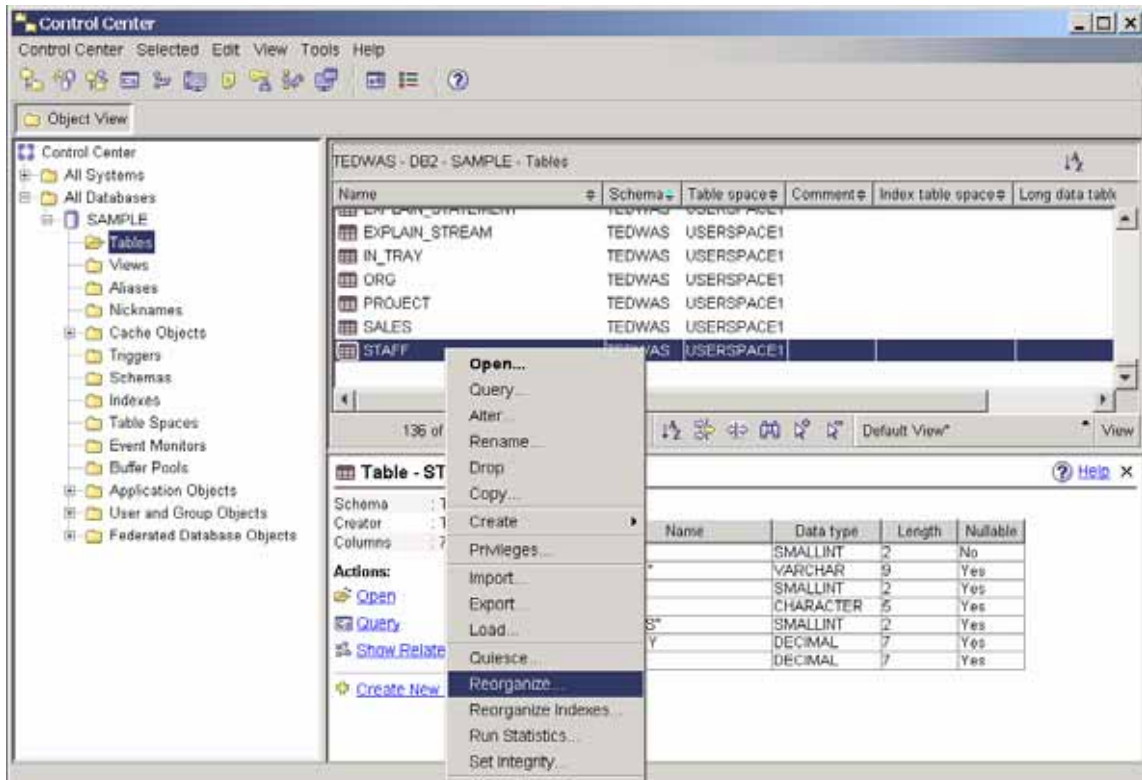
**Example:**

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).

#### 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.



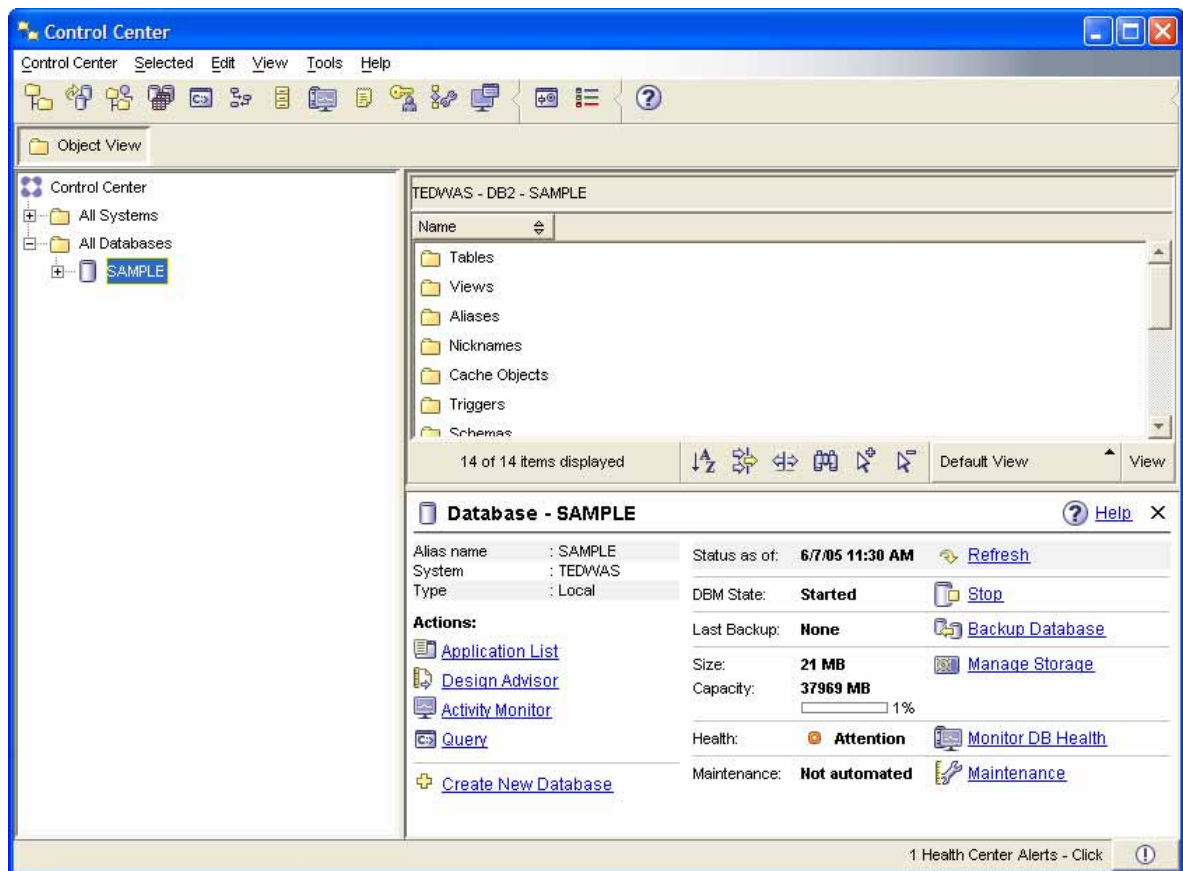


Figure 12.4 – The database operational view from the Control Center

## 12.2 Maintenance Choices

There are three ways to perform maintenance tasks:

1. Manual maintenance  
You perform maintenance activities manually when the need arises
2. Create scripts to perform maintenance  
You can create scripts with the maintenance commands, and schedule them regularly for execution.
3. 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 window

From the Control Center, you can launch the Configure Automated Maintenance Wizard as shown in Figure 12.5.

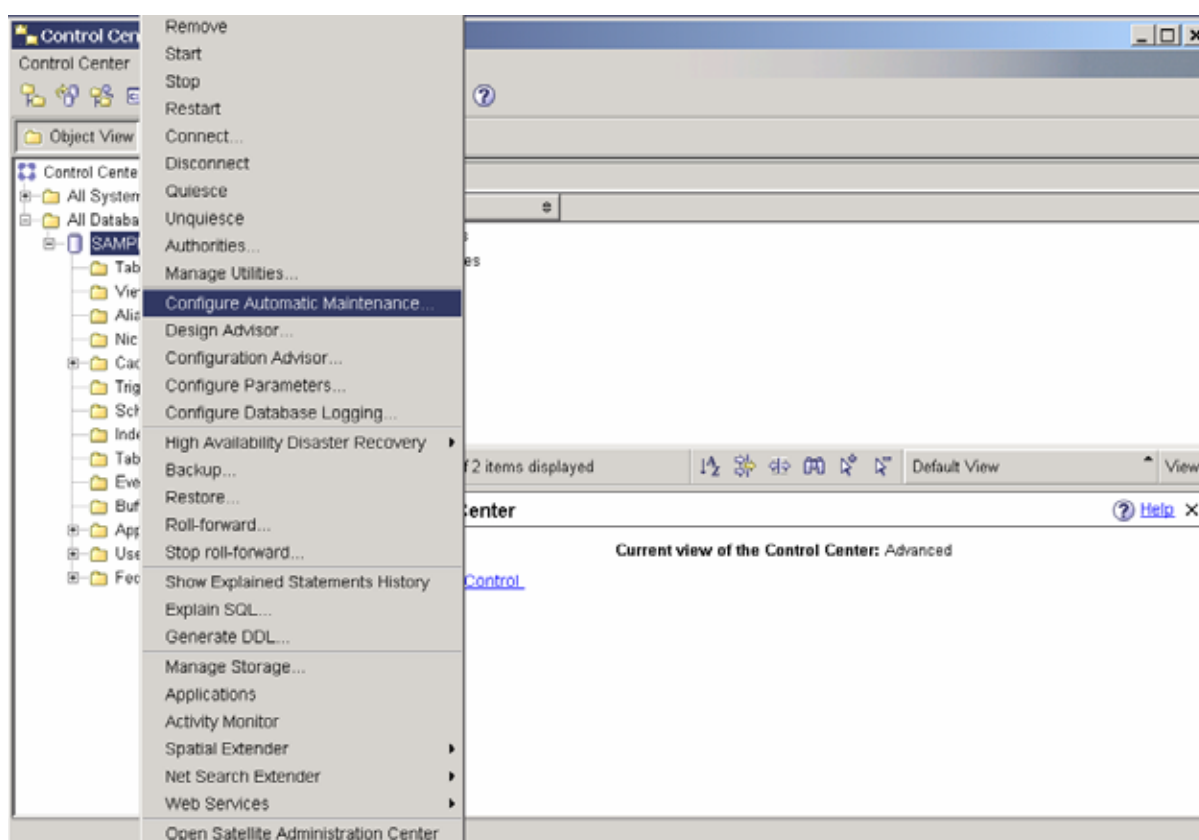


Figure 12.5 – Launching the Configure Automated Maintenance Wizard

Figure 12.6 shows the Configure Automated Maintenance Wizard.

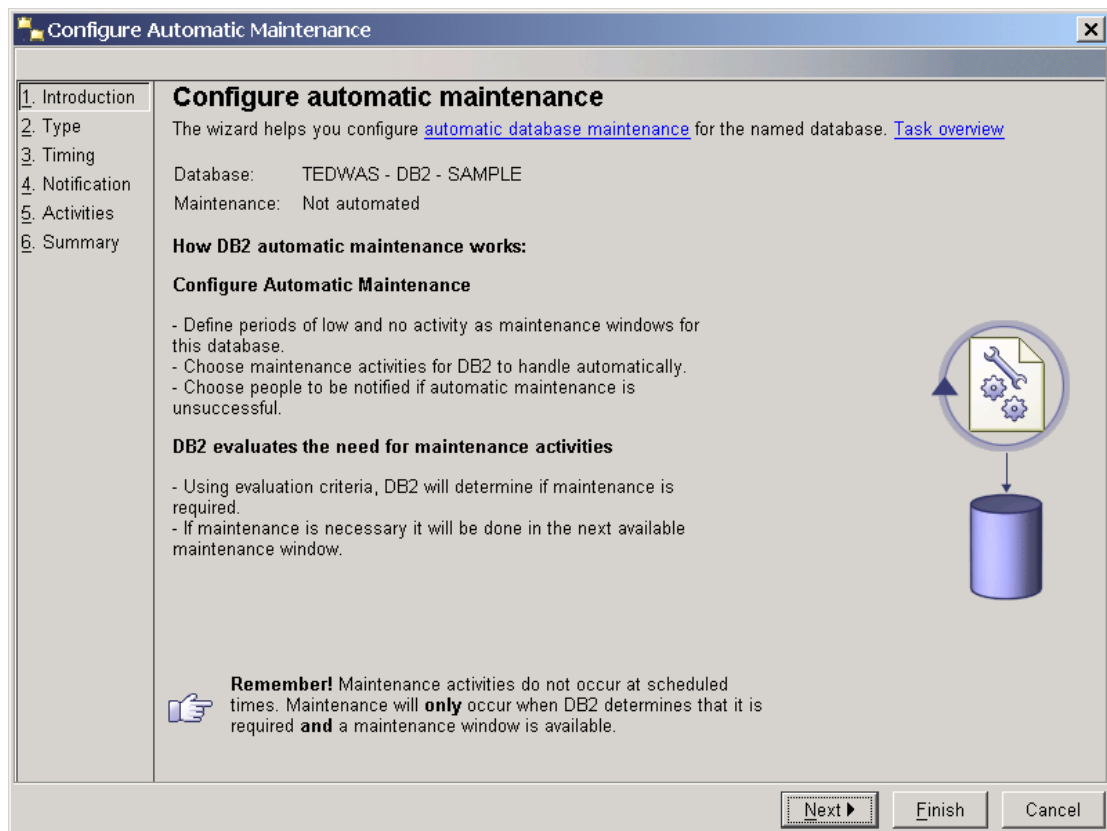


Figure 12.6 –The Configure Automated Maintenance Wizard

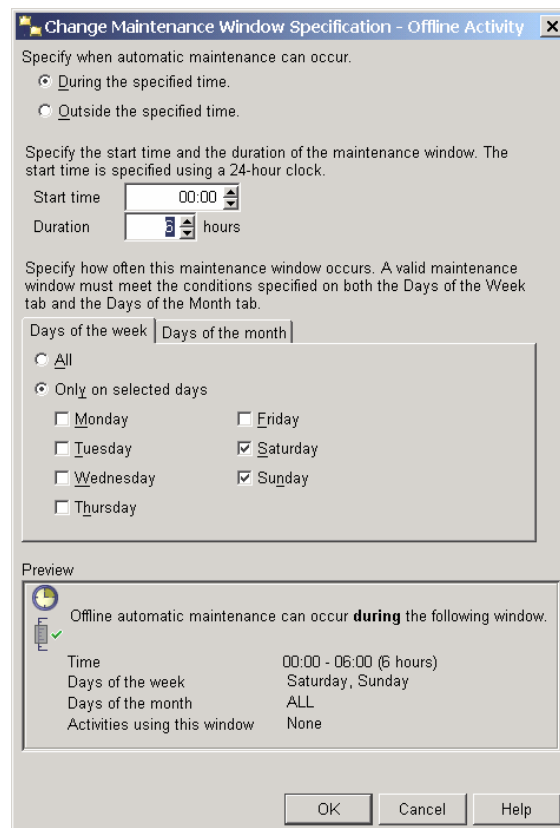
## Quicklab #9 – Configuring automated maintenance

### Objective

In this Quicklab, with a few easy steps, you will configure automatically 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, or changing your automated maintenance settings. Select the option to change the current automated maintenance settings. Click the *Next*.
4. The *Timing* page of the wizard asks you to specify the maintenance windows. Configure the *Offline* window to be every 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.

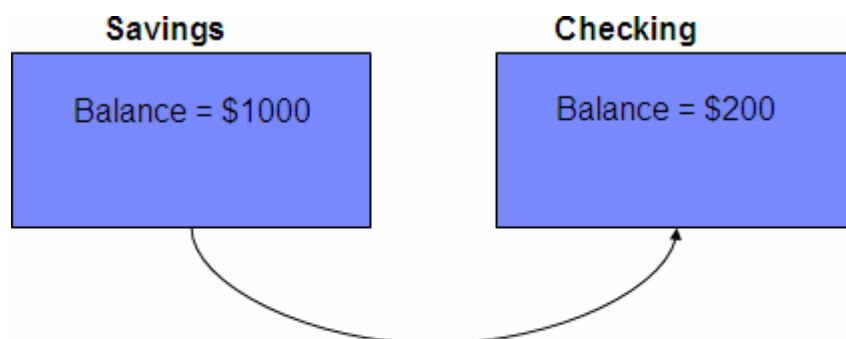
# 13

## 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.

### 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.



Transfer \$100 from Savings to Checking:

- Debit \$100 from Savings account
- Credit \$100 to Checking account

**Figure 13.1 –An example of a transaction**

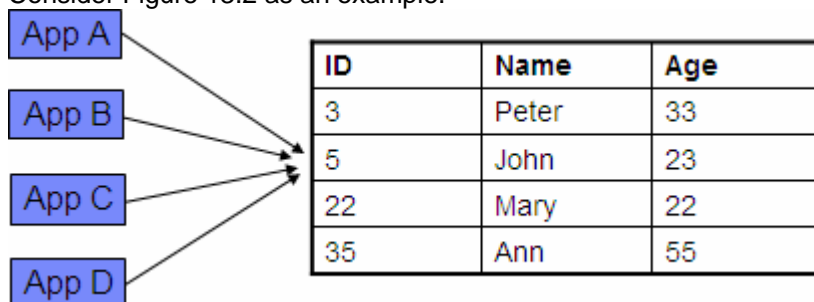
In Figure 13.1, for example, you want to transfer 100 dollars from your savings account to your checking account. The following sequence of events may be required to achieve this task:

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.

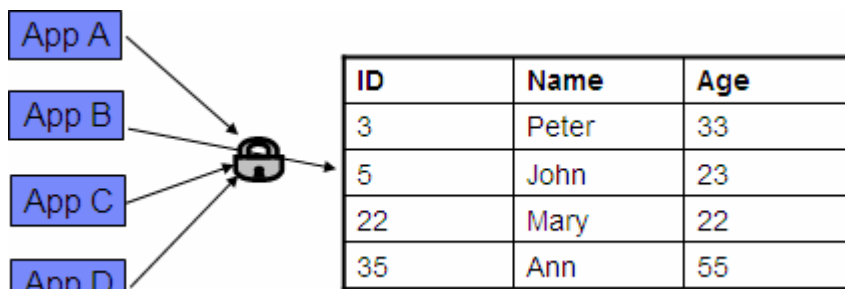
Locking and concurrency concepts go hand in hand. Locking temporarily stops other applications from performing their operation until another operation finishes. The more locking there is in a system, the less concurrency is possible. On the other hand, the less locking there is in a system, the more 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.

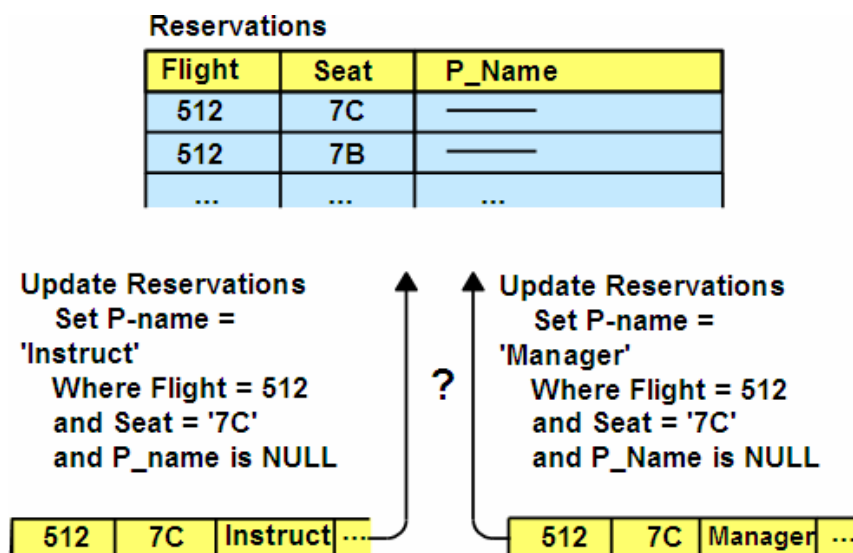
### 13.3 Problems without concurrency control

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.

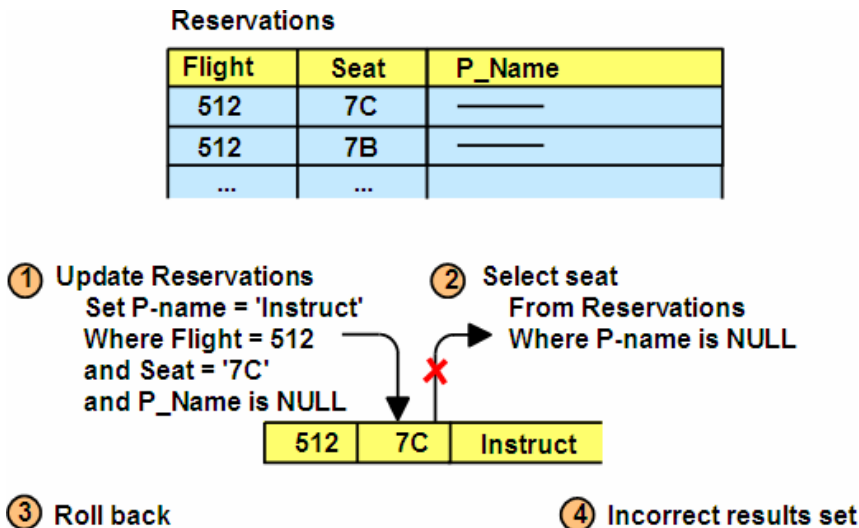


Figure 13.5 – Uncommitted Read

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**

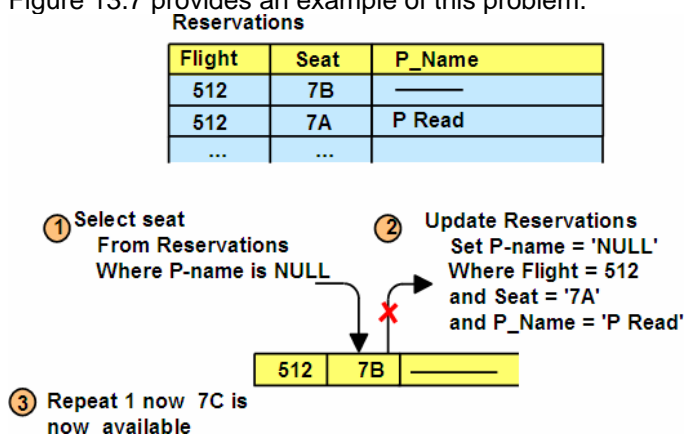
In Figure 13.6, consider if you are trying to book a flight from Dallas to Honolulu. The sequence of events is:

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.



**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

### 13.4.3 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 (A commit point).

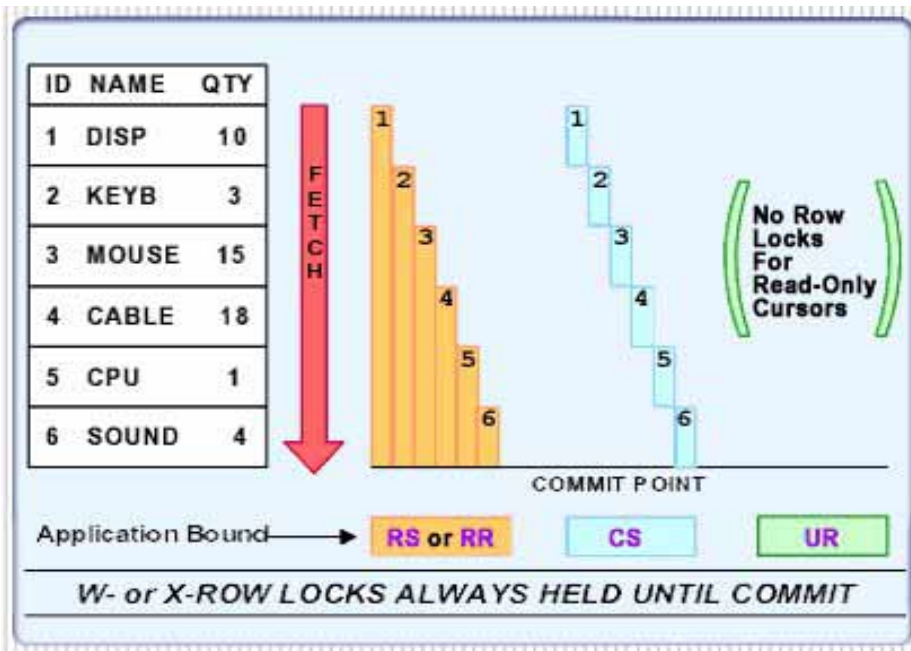


Figure 13.8 – Comparing isolation levels for a fetch

### 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.

### 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.

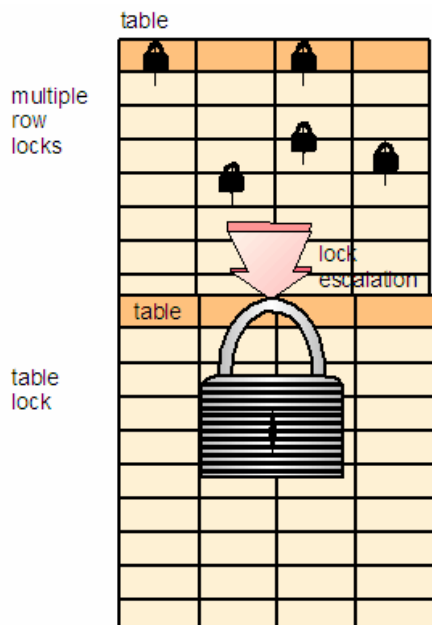


Figure 13.9 – Lock escalation

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. The default is fifty 4K (200 K)pages on Windows
- ▶ **MAXLOCKS** –Maximum percentage of the entire lock list that a single application can use up. The default is 22%.

Therefore, if the default values are used, 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, which is normally located in the C:\Program Files\IBM\SQLLIB\DB2 directory) can be used to determine whether lock escalation is occurring.

### 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 AGENT ID <handle>
```

Figure 13.9 shows the output for a sample application lock snapshot.

**Figure 13.9 – 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                    = 0x050007000480010000000000052
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
Table Name                    = T2
Mode                          = X

```



### 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).

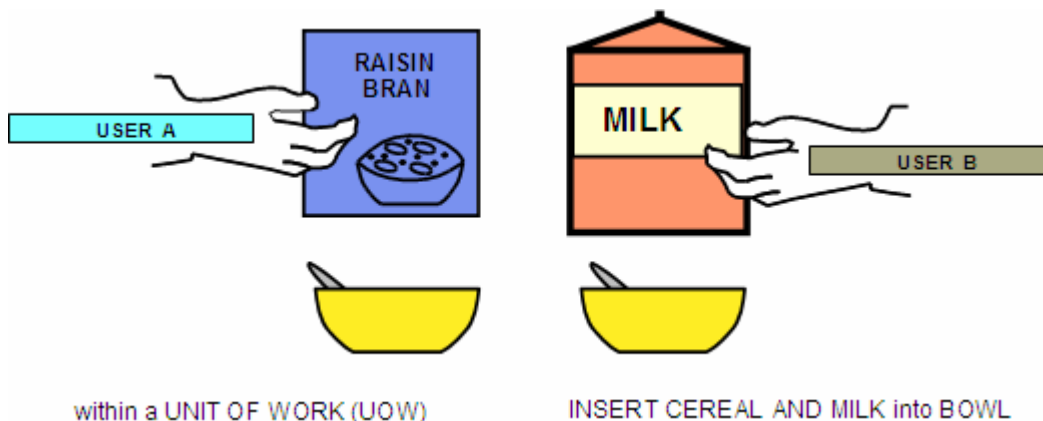
The CURRENT LOCK TIMEOUT register can be used to set the lock wait for a given connection. By default, this register is set to the value of LOCKTIMEOUT. Use the SET LOCK TIMEOUT statement to change its value. Once the value of this register is set for a connection, it will persist across transactions.

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.



**Figure 13.10 – Deadlock scenario**

In Figure 13.10, user A is holding the raisin bran 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 raisin bran. Therefore, we have a deadlock situation.

To simulate a deadlock situation in DB2, follow these steps:

1. Open two DB2 Command Windows (which we will call “CLP1” and “CLP2”, respectively) representing two different applications connecting to the database
2. 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 = 50000” 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.

3. 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.

4. 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.

5. 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. 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.

2. Log transaction information only when required.

3. Purge data quickly using:

```
ALTER TABLE ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE
```

4. Perform data modifications in batches/groups. For example:

```
DELETE FROM (  
  SELECT * FROM tedwas.t1 WHERE c1 = ... FETCH FIRST 3000 ROWS  
  ONLY)
```

5. Use concurrency features in DB2 data movement tools.

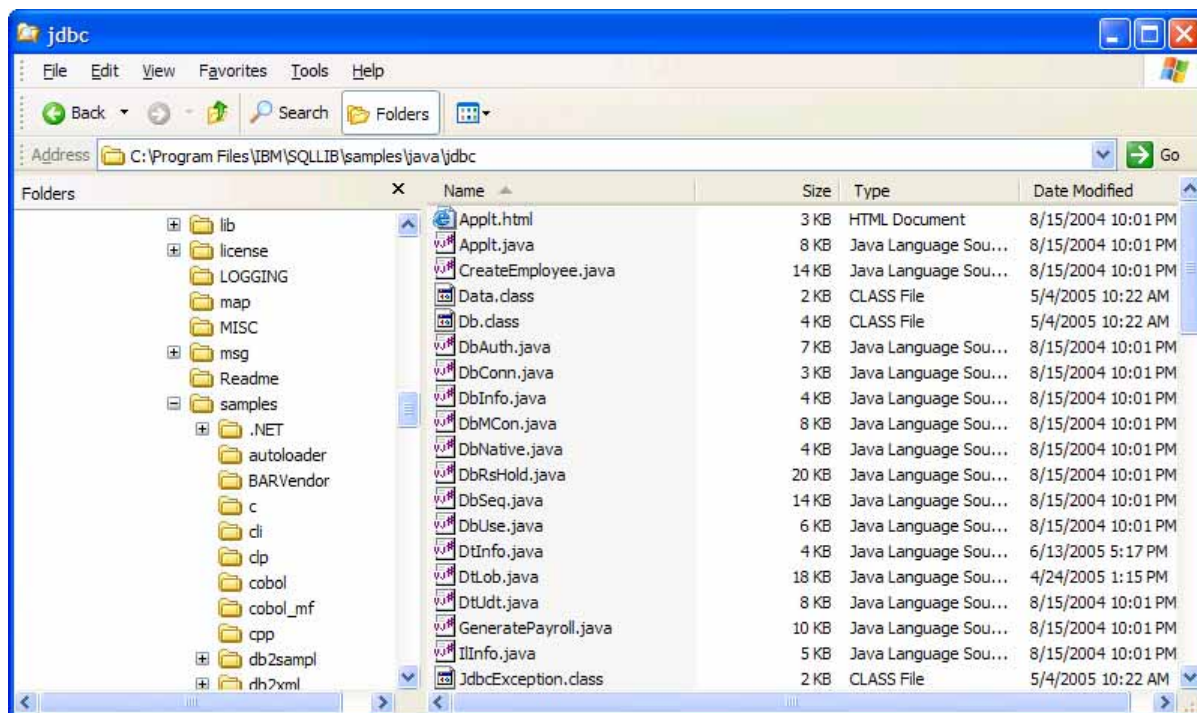
6. Set the database level LOCKTIMEOUT parameter (suggested times are between 30-120 seconds). Don't leave it to the default of -1. You can also use session-based lock timeout.

7. Do not retrieve more data than is required. For example, use the FETCH FIRST n ROWS ONLY clause in SELECT statements.



## **PART III – LEARNING DB2: APPLICATION DEVELOPMENT**

In part III of this book, we discuss in depth application database objects such as: stored procedures, user-defined functions (UDFs), and triggers. Note that you can learn how to program in different languages using DB2 as the data server by reviewing the sample applications that come with the DB2 server installation in the `SQLLIB\samples` directory. The figure below shows the sample Java programs provided with DB2 on a Windows platform.



**Sample Java programs that come with DB2**

# 14

## Chapter 14 – SQL PL Stored Procedures

In this chapter we discuss 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 considerably reduced. In addition, stored procedures provide a centralized location to store your code, so other applications can reuse the same procedures.

DB2 stored procedures can be written using SQL PL, C/C++, Java, Cobol, CLR (Common Language Runtime) supported languages, and OLE. In this chapter, we focus on SQL PL procedures because of their popularity and simplicity.

Figure 14.1 illustrates how stored procedures work.

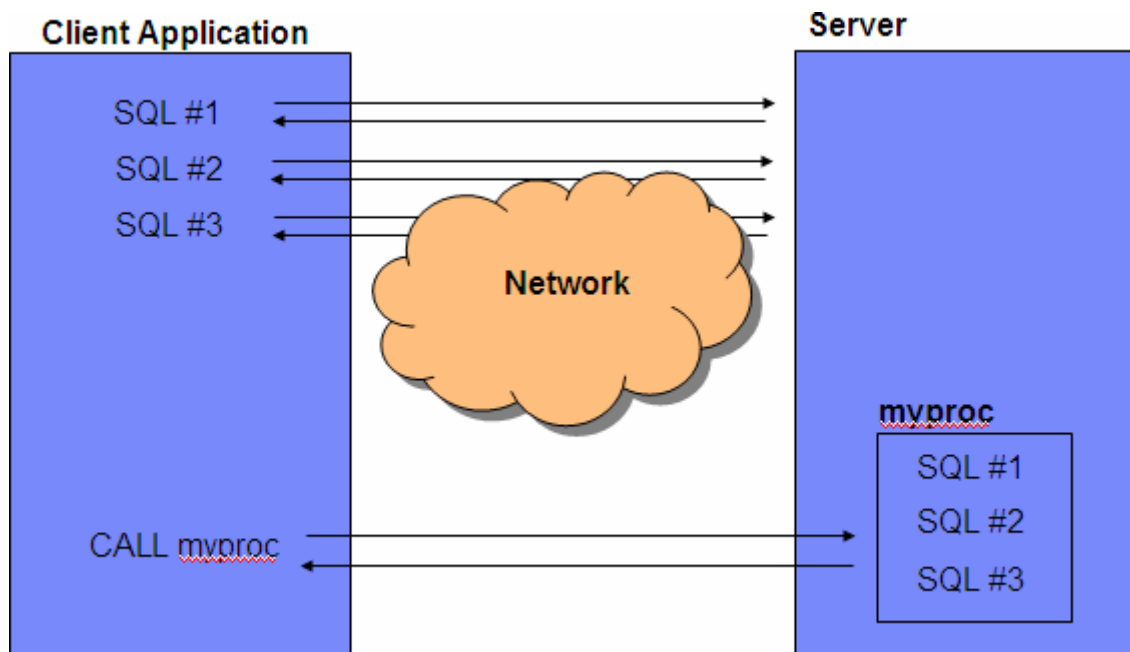


Figure 14.1 – Network traffic reduction with stored procedures

At the top left corner of the figure, you see several SQL statements executed one after the other. Each SQL is sent from the client to the server, and the server returns the result back to the client. If many SQL statements are executed like this, network traffic increases. On the other hand, at the bottom right corner, you can see the stored procedure “myproc” held on the server, which contains the same SQL; and then at the client (on the left side), the CALL statement is used to call the procedure. This second method of calling the procedure is more efficient, as there is only one call statement that goes through the network, and one result returned to the client.

Stored procedures can also be helpful for security purposes in your database. For example, you can let users access tables or views only through stored procedures; this helps lock down the server and keep users from accessing information they are not supposed to access. This is possible because users do not require explicit privileges on the tables or views they access through stored procedures; they just need to be granted sufficient privilege to invoke the stored procedures.

## 14.1 The DB2 Developer Workbench

The DB2 Developer workbench (DWB) is an Eclipse-based tool for development of stored procedures, functions, XML queries, SQLJ applications, amongst others. DWB comes as a separate image, (it is not part of the DB2 installation image), but it is free of charge. DWB images can be downloaded from the “Download” tab on the [ibm.com/db2/express](http://ibm.com/db2/express) web site. Figure 14.2 shows the DWB.

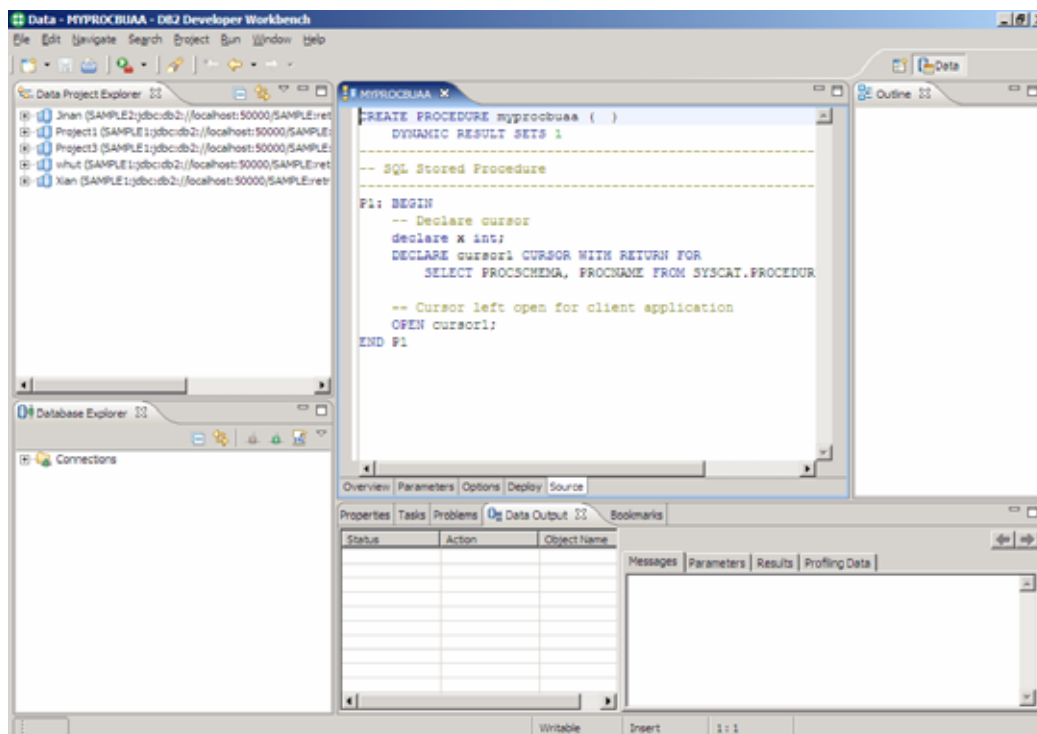


Figure 14.2 – The DB2 Developer Workbench



### 14.1.2 Create a stored procedure in the DWB

To create a Java or SQL PL stored procedure in the DWB, follow the following steps. Note that stored procedures written in other languages cannot be created from the DWB.

#### Step 1: Create a DWB project

From the DWB menu, choose File => New => Project and choose “Data Development Project”. This is shown in Figure 14.3

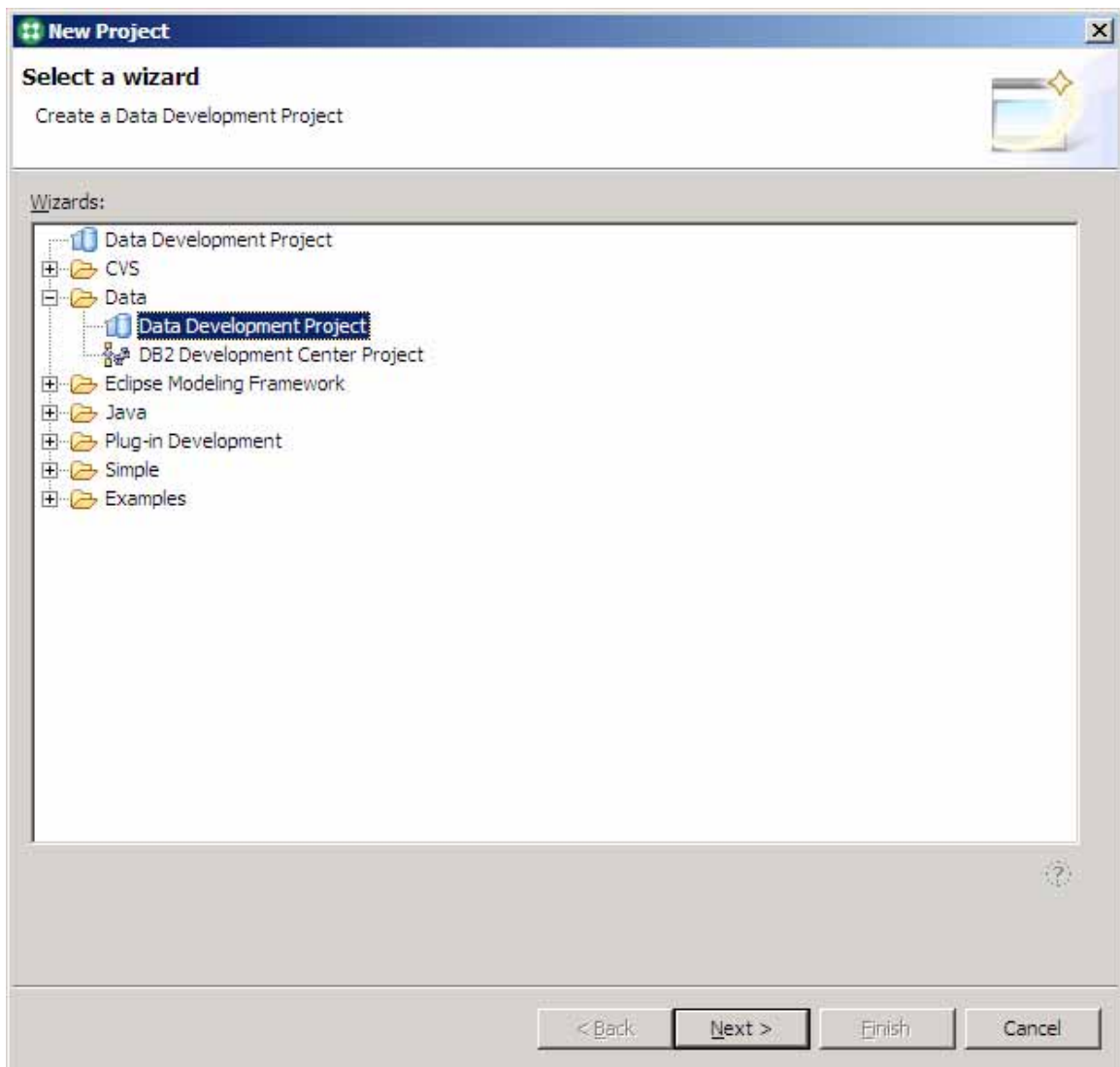
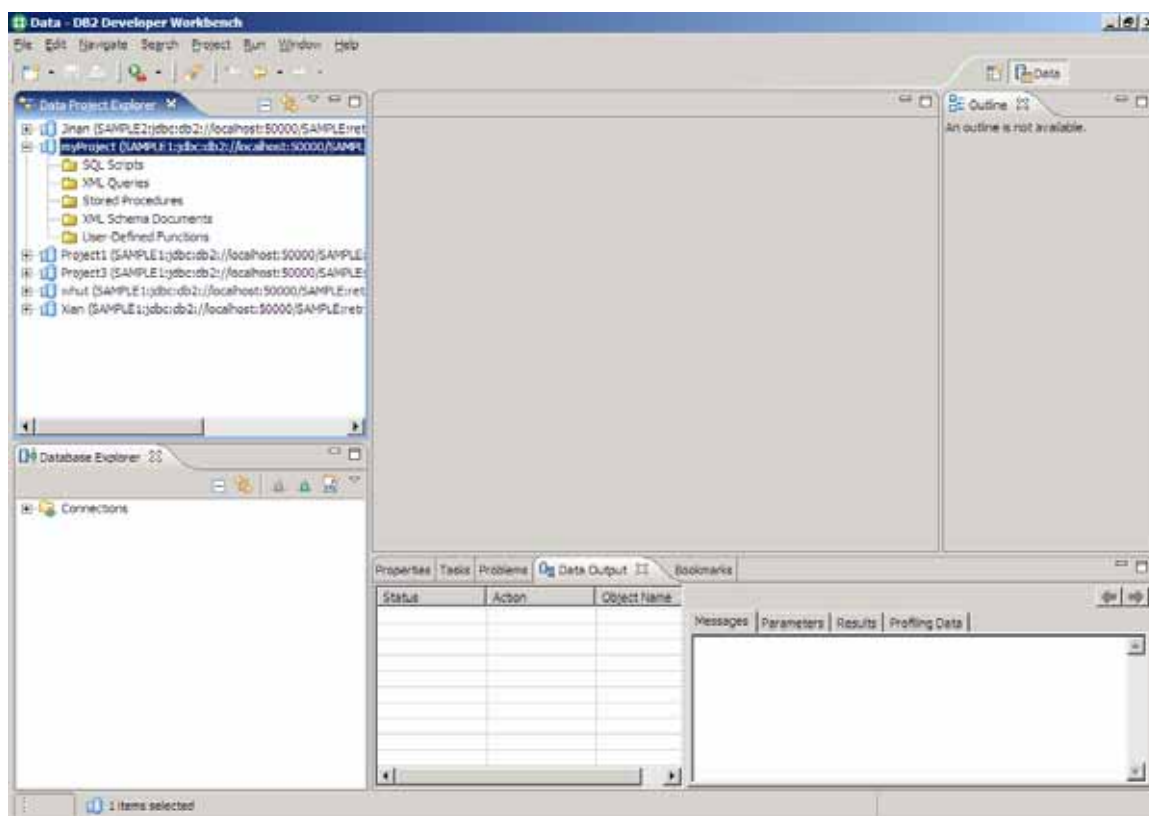


Figure 14.3 – The data development project

Follow the steps from the wizard to input a name for your project, indicate which database you want to connect to as part of your project, and specify the JDK directory (the one provided by default is usually the correct one).

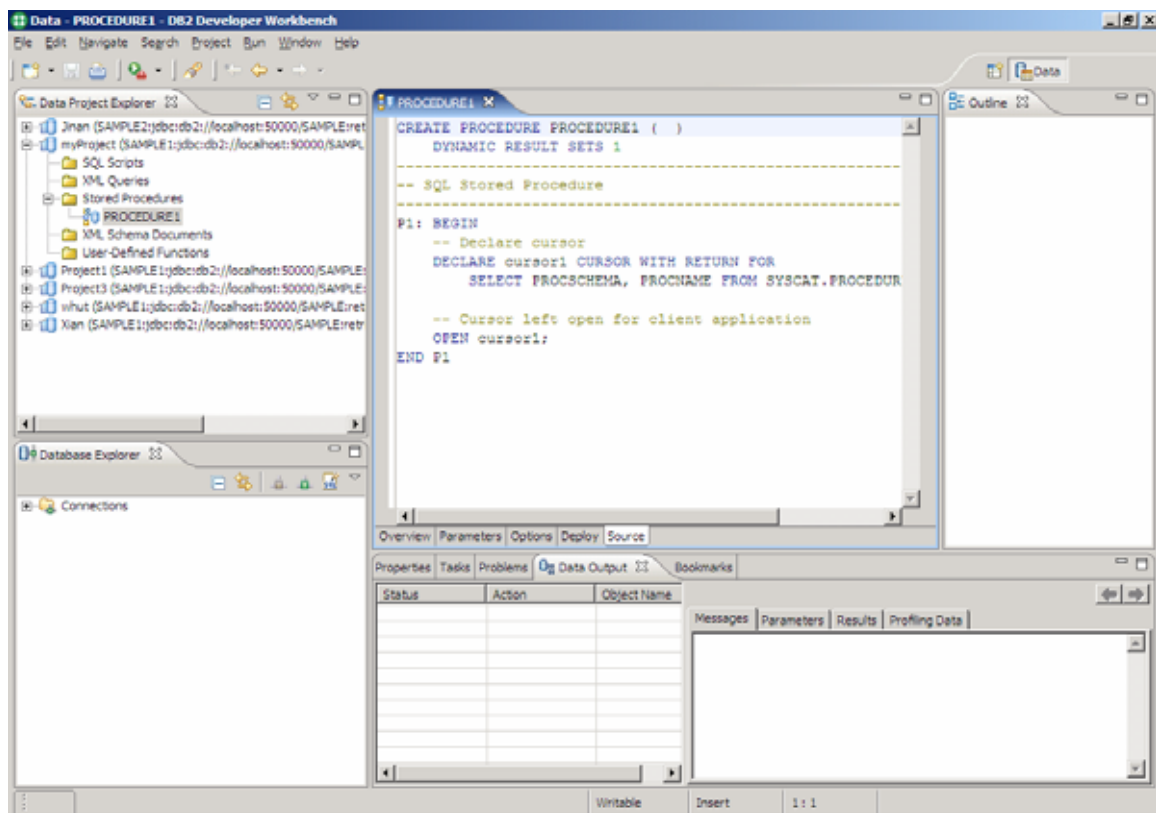
**Step 2: Create a stored procedure**

When the project is created, the left side of the data perspective will show your project. In Figure 14.4 you can see the “myProject” project created and expanded.



**Figure 14.4 – The “myProject” project**

Figure 14.4 shows different folders for your project. When you want to create a stored procedure, right-click on the Stored Procedures folder and choose New => Stored Procedure. Complete the information requested by the “New Stored Procedure” wizard such as the project to associate the procedure with, the name and language of the procedure (Note that only SQL PL and Java are supported within the DWB), and the SQL statements to use in the procedure. By default, DWB gives you an example SQL statement. At this point, you can click Finish and a stored procedure is created using some template code and the SQL statement provided before as an example. This is shown in Figure 14.5.



**Figure 14.5 – A sample stored procedure**

In Figure 14.5, the code for the sample stored procedure “PROCEDURE1” was generated. You can replace all of this code with your own code. For simplicity, we will continue in this book using the above sample stored procedure as if we had written it.

### **Step 3: Compile (deploy) a stored procedure**

Once the stored procedure is created, you compile and deploy it by right-clicking on it in the left panel, and choosing “Deploy”. Figure 14.6 illustrates this step.

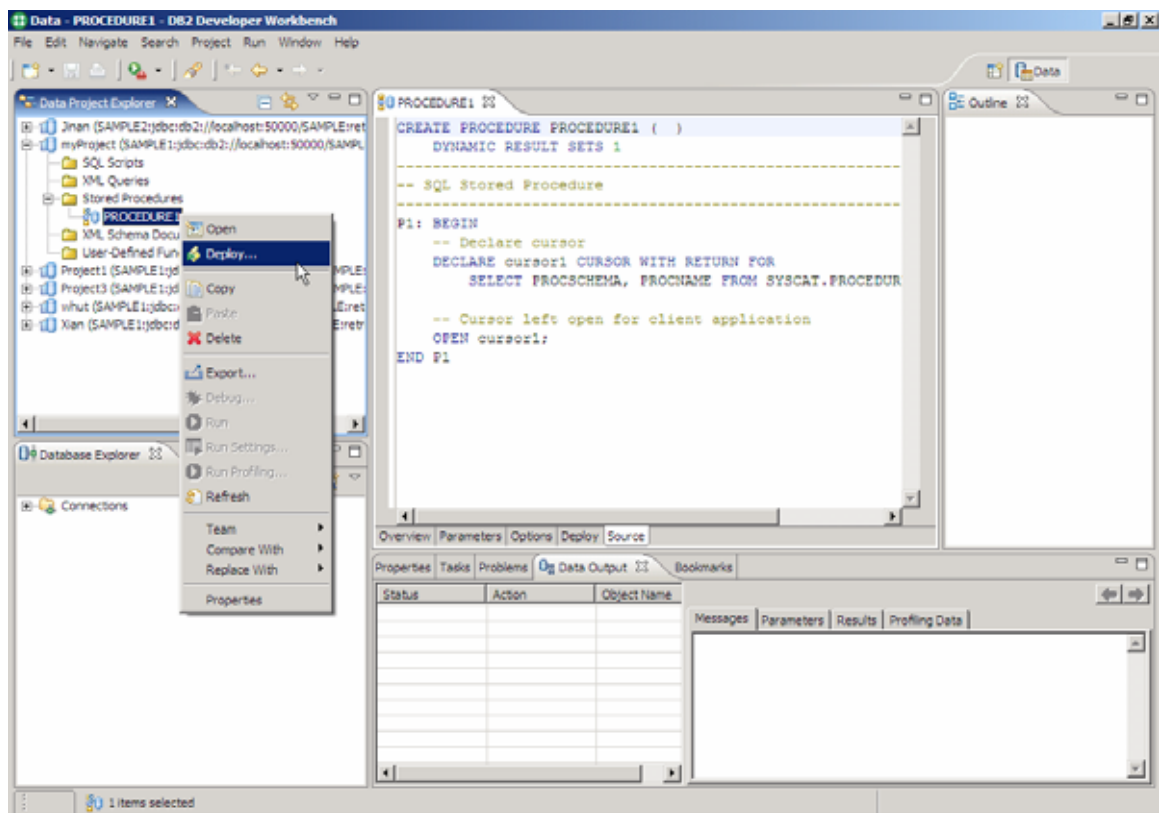


Figure 14.6 – Deploying a stored procedure

#### Step 4: Run a stored procedure

Once the stored procedure has been deployed, you can run it by right-clicking on it and choosing “Run”. The results would appear in the “Results” tab at the bottom right corner of the window.

To run a stored procedure from the Command Window or the Command Editor, you can use `CALL <procedure name>`

## 14.2 SQL PL stored procedures basics

SQL Procedural Language (SQL PL) stored procedures are easy to create and learn. They have the best performance in DB2. SQL PL stored procedures (or simply “SQL stored procedures”) are the focus of this chapter.

### 14.2.1 Stored procedure structure

The basic store procedure syntax is shown here.

```
CREATE PROCEDURE proc_name [( {optional parameters} )]
                        [optional procedure attributes]      <statement>
```

Where `<statement>` is a single statement, or a set of statements grouped by `BEGIN [ATOMIC] ... END`

### 14.2.2 Optional stored procedure attributes

The following describes some of the optional stored procedure attributes:

- `LANGUAGE SQL`  
This attribute indicates the language the stored procedure will use. `LANGUAGE SQL` is the default value. For other languages, such as Java or C use `LANGUAGE JAVA` or `LANGUAGE C`, respectively.
- `RESULT SETS <n>`  
This is required if your stored procedure will be returning `n` result sets.
- `SPECIFIC my_unique_name`  
This is a unique name that can be given to a procedure. A stored procedure can be overloaded, that is, several stored procedures can have the same name, but with different number of parameters. By using the `SPECIFIC` keyword you can provide one unique name for each of these stored procedures, and this can ease management of stored procedures. For example, to drop a stored procedure using the `SPECIFIC` keyword, you can issue this statement: `DROP SPECIFIC PROCEDURE`. If the `SPECIFIC` keyword had not been used you would have had to use a `DROP PROCEDURE` statement and put the name of the procedure with the parameters so DB2 would know which of the overloaded procedures you wanted to drop.

### 14.2.3 Parameters

There are three types of parameters in an SQL PL stored procedure:

- `IN` - Input parameter
- `OUT` - Output parameter
- `INOUT` - Input and Output parameter

For example:

```
CREATE PROCEDURE proc(IN p1 INT, OUT p2 INT, INOUT p3 INT)
```

When calling the procedure, all the parameters must be provided in the `CALL` statement. For example, to call the above stored procedure you would specify:

```
CALL proc (10,?,4)
```

The question mark (?) is used for `OUT` parameters in the `CALL` statement.

Here is another example of a stored procedure with parameters that you can try:

```

CREATE PROCEDURE P2 ( IN    v_p1 INT,
                    INOUT v_p2 INT,
                    OUT   v_p3 INT)

LANGUAGE SQL
SPECIFIC myP2
BEGIN
  -- my second SQL procedure
  SET v_p2 = v_p2 + v_p1;
  SET v_p3 = v_p1;
END

```

To call the procedure from the Command Editor use:

```
call P2 (3, 4, ?)
```

#### 14.2.4 Comments in an SQL PL stored procedure

There are two ways to specify comments in an SQL PL stored procedure:

- Using two dashes. For example:  
-- This is an SQL-style comment
- Using a format similar to the C language. For example:  
/\* This is a C-style coment \*/

#### 14.2.5 Compound statements

A compound statement in a stored procedure is a statement consisting of several procedural instructions and SQL statements encapsulated by the keywords BEGIN and END. When the ATOMIC keyword follows the BEGIN keyword, the compound statement is treated as one unit, that is, all of the instructions or statements in the compound statement must be successful in order for the entire compound statement to be successful. If one of the statements is not, then everything is rolled back. Figure 14.7 shows a compound statement structure.

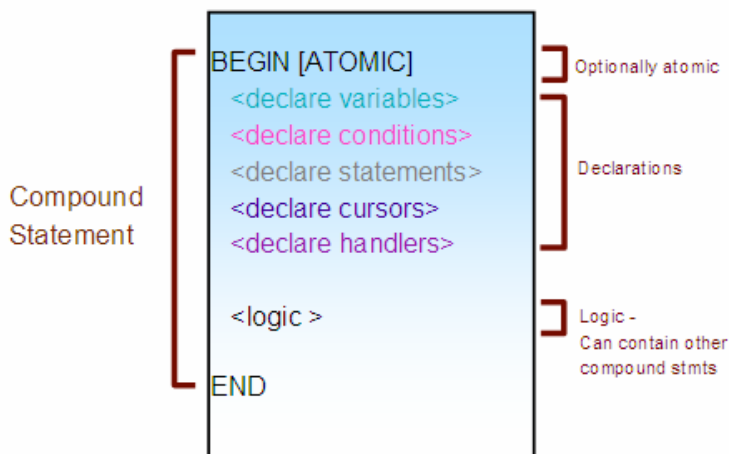


Figure 14.7 – Compound statements

### 14.2.6 Variable declaration

To declare a variable, use the DECLARE statement:

```
DECLARE var_name <data type> [DEFAULT value];
```

Here are some examples:

```
DECLARE temp1 SMALLINT DEFAULT 0;
DECLARE temp2 INTEGER DEFAULT 10;
DECLARE temp3 DECIMAL(10,2) DEFAULT 100.10;
DECLARE temp4 REAL DEFAULT 10.1;
DECLARE temp5 DOUBLE DEFAULT 10000.1001;
DECLARE temp6 BIGINT DEFAULT 10000;
DECLARE temp7 CHAR(10) DEFAULT 'yes';
DECLARE temp8 VARCHAR(10) DEFAULT 'hello';
DECLARE temp9 DATE DEFAULT '1998-12-25';
DECLARE temp10 TIME DEFAULT '1:50 PM';
DECLARE temp11 TIMESTAMP DEFAULT '2001-01-05-12.00.00';
DECLARE temp12 CLOB(2G);
DECLARE temp13 BLOB(2G);
```

### 14.2.7 Assignment statements

To assign a value to a variable, use the SET statement. For example:

```
SET total = 100;
```

The above statement is equivalent to

```
VALUES(100) INTO total;
```

Additionally, any variable can be set to NULL:

```
SET total = NULL;
```

A condition is raised if more than one row fetches only the first row from a table

```
SET total = (select sum(c1) from T1);
SET first_val = (select c1 from T1 fetch first 1 row only)
```

You can also set variables according to external database properties:

```
SET sch = CURRENT SCHEMA;
```

## 14.3 Cursors

A cursor is a result set holding the result of a SELECT statement. The syntax to declare, open, fetch, and close a cursor follows:

```
DECLARE <cursor name> CURSOR [WITH RETURN <return target>]
    <SELECT statement>;
OPEN <cursor name>;
FETCH <cursor name> INTO <variables>;
CLOSE <cursor name>;
```

When a cursor is declared, the WITH RETURN clause can be used with these values:

- CLIENT: the result set will return to client application

- CALLER: the result set is returned to client or stored procedure that made the call

Here is an example of a stored procedure using a cursor:

```
CREATE PROCEDURE set ()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
DECLARE cur CURSOR WITH RETURN TO CLIENT
FOR SELECT name, dept, job
FROM staff
WHERE salary > 20000;
OPEN cur;
END
```

## 14.4 Flow control

Like in many other languages, SQL PL has several statements that can be used to control the flow of the logic. Below we list some of the flow control statements supported:

```
CASE (selects an execution path (simple search))
IF
FOR (executes body for each row of table)
WHILE
ITERATE (forces next iteration. Similar to CONTINUE in C)
LEAVE (leaves a block or loop. "Structured Goto")
LOOP (infinite loop)
REPEAT
GOTO
RETURN
CALL (procedure call)
```

## 14.5 Calling stored procedures

The following code snippets show how to CALL stored procedures using different programming languages.

### Example calling a stored procedure from a CLI/ODBC application

```
SQLCHAR *stmt = (SQLCHAR *)
"CALL MEDIAN_RESULT_SET( ? )" ;
SQLDOUBLE sal = 20000.0; /* Bound to parameter marker in
stmt */
SQLINTEGER salind = 0; /* Indicator variable for sal */

sqlrc = SQLPrepare(hstmt, stmt, SQL_NTS);
sqlrc = SQLBindParameter(hstmt, 1, SQL_PARAM_OUTPUT,
SQL_C_DOUBLE, SQL_DOUBLE, 0, 0, &sal, 0, &salind);
SQLExecute(hstmt);
```



```

if (salind == SQL_NULL_DATA)
    printf("Median Salary = NULL\n");
else
    printf("Median Salary = %.2f\n\n", sal );

/* Get first result set */
sqlrc = StmtResultPrint(hstmt);
/* Check for another result set */
sqlrc = SQLMoreResults(hstmt);
if (sqlrc == SQL_SUCCESS) {
    /* There is another result set */
    sqlrc = StmtResultPrint(hstmt);
}

```

For more details, see the DB2 sample file: `sqllib/samples/sqlproc/rsultset.c`

#### Example calling a stored procedure from a VB.NET application

```

Try
    ` Create a DB2Command to run the stored procedure
    Dim procName As String = "TRUNC_DEMO"
    Dim cmd As DB2Command = conn.CreateCommand()
    Dim parm As DB2Parameter

    cmd.CommandType = CommandType.StoredProcedure
    cmd.CommandText = procName

    ` Register the output parameters for the DB2Command
    parm = cmd.Parameters.Add("v_lastname",
DB2Type.VarChar)
    parm.Direction = ParameterDirection.Output
    parm = cmd.Parameters.Add("v_msg",
DB2Type.VarChar)
    parm.Direction = ParameterDirection.Output

    ` Call the stored procedure
    Dim reader As DB2DataReader = cmd.ExecuteReader

Catch myException As DB2Exception
    DB2ExceptionHandler(myException)
Catch
    UnhandledExceptionHandler()
End Try

```

#### Example calling a stored procedure from a Java application

```

try
{
    // Connect to sample database
    String url = "jdbc:db2:sample";
    con = DriverManager.getConnection(url);

```

```

        CallableStatement cs = con.prepareStatement("CALL
trunc_demo(?, ?)");

        // register the output parameters
        callStmt.registerOutParameter(1, Types.VARCHAR);
        callStmt.registerOutParameter(2, Types.VARCHAR);

        cs.execute();
        con.close();
    }
    catch (Exception e)
    {
        /* exception handling logic goes here */
    }

```

## 14.6 Errors and condition handlers

In DB2, the SQLCODE and SQLSTATE keywords are used to determine the successful or unsuccessful execution of an SQL statement. These keywords need to be explicitly declared in the outermost scope of the procedure as follows:

```

DECLARE SQLSTATE CHAR(5);
DECLARE SQLCODE INT;

```

DB2 will set the values of the above keywords automatically after each SQL operation. For the SQLCODE, the values are set as follows:

- = 0, successful.
- > 0, successful with warning
- < 0, unsuccessful
- = 100, no data was found. (i.e.: FETCH statement returned no data)

For the SQLSTATE, the values are set as follows:

- success: SQLSTATE '00000'
- not found: SQLSTATE '02000'
- warning: SQLSTATE '01XXX'
- exception: all other values

The SQLCODE is RDBMS specific, and more detailed than the SQLSTATE. The SQLSTATE is standard among RDBMSs but is very general in nature. Several SQLCODEs may match one SQLSTATE. SQLCODEs and SQLSTATEs are discussed in more detail in Chapter 18, Troubleshooting.

A condition can be raised by any SQL statement and would match an SQLSTATE. For example, a specific condition like SQLSTATE '01004' is raised when a value is truncated during an SQL operation. Rather than using SQLSTATE '01004' to test for this condition, names can be assigned. In this particular example, the name "trunc" can be assigned to condition SQLSTATE '01004' as shown below.

```

DECLARE trunc CONDITION FOR SQLSTATE '01004'

```

Other predefined general conditions are:

- SQLWARNING
- SQLEXCEPTION
- NOT FOUND

### Condition handling

To handle a condition, you can create a condition handler which must specify:

- which conditions it handles
- where to resume execution (based on the type of the handler: CONTINUE, EXIT or UNDO)
- the actions to perform to handle the condition. The actions can be any statement, including control structures.

If an SQLEXCEPTION condition is raised, and there is no handler, the procedure terminates and returns to the client with an error.

### Types of handlers

There are three types of handlers:

**CONTINUE** – This handler is used to indicate that after an exception is raised, and the handler handles the condition, the flow will CONTINUE to the next statement after the statement that raised the condition.

**EXIT** – This handler is used to indicate that, after an exception is raised, and the handler handles the condition, the flow will go to the end of the procedure.

**UNDO** – This handler is used to indicate that after an exception is raised, and the handler handles the condition, the flow will go to the end of the procedure, and will undo or roll back any statements performed.

Figure 14.8 illustrates the different condition handlers and their behavior.

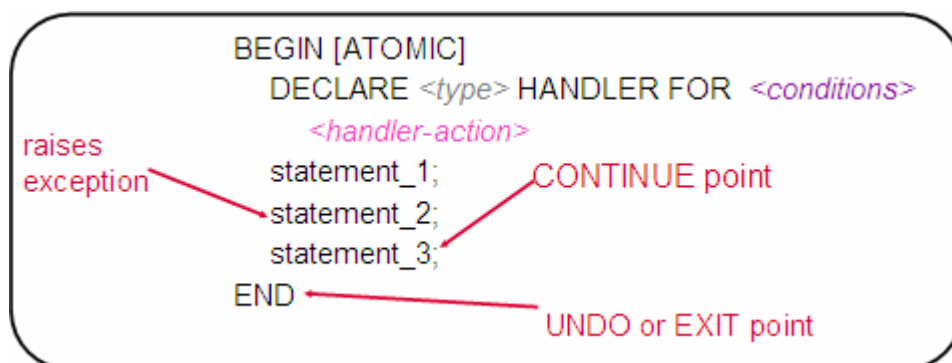


Figure 14.8 – Type of condition handlers

## 14.7 Dynamic SQL

In dynamic SQL, as opposed to static SQL, the entire SQL statement is not known at run time. For example if `col1` and `tablename` are variables in this statement, then we are dealing with dynamic SQL:

```
'SELECT ' || col1 || ' FROM ' || tablename;
```

Dynamic SQL is also recommended for DDL to avoid dependency problems and package invalidation. It is also required to implement recursion.

Dynamic SQL can be executed using two approaches:

- Using the EXECUTE IMMEDIATE statement – this is ideal for single execution SQL
- Using the PREPARE statement along with the EXECUTE statement - ideal for multiple execution SQL

The following code snippet provides an example of Dynamic SQL using the two approaches. The example assumes a table T2 has been created with this definition:

```
CREATE TABLE T2 (c1 INT, c2 INT)

CREATE PROCEDURE dyn1 (IN value1 INT, IN value2 INT)
SPECIFIC dyn1
BEGIN
  DECLARE stmt varchar(255);
  DECLARE st STATEMENT;

  SET stmt = 'INSERT INTO T2 VALUES (?, ?)';

  PREPARE st FROM stmt;

  EXECUTE st USING value1, value1;
  EXECUTE st USING value2, value2;

  SET stmt = 'INSERT INTO T2 VALUES (9,9)';
  EXECUTE IMMEDIATE stmt;
END
```

# 15

## Chapter 15 – Inline SQL PL, UDFs, Triggers

In this chapter, we discuss inline SQL PL and other database application objects such as user-defined functions (UDFs) and triggers.

### 15.1 Inline SQL PL

In Chapter 14 we discussed creating stored procedures using the SQL PL language. The SQL PL language can also be used “inline” meaning that it can stand alone without the need to create a stored procedure. The SQL PL used in UDFs and triggers is also inline because the code is added inline with the UDF/trigger code, and it is dynamic SQL in nature. Inline SQL PL supports only a subset of all the SQL PL statements. Here are inline SQL PL supported keywords:

```
DECLARE <variable>
SET
CASE
FOR
GET DIAGNOSTICS
GOTO
IF
RETURN
SIGNAL
WHILE
ITERATE
LEAVE
```

Here are keywords not supported in inline SQL PL:

```
ALLOCATE CURSOR
ASSOCIATE LOCATORS
DECLARE <cursor>
DECLARE ...HANDLER
PREPARE
EXECUTE
EXECUTE IMMEDIATE
LOOP
REPEAT
RESIGNAL
CALL
COMMIT/ROLLBACK
```

Here is an example of dynamic compound SQL using inline SQL PL. If you want to try it out, you can put it as part of a script file, and ensure you create the following tables:

```
CREATE TABLE T1 (c1 INT)
CREATE TABLE T3 (c1 INT)

BEGIN ATOMIC
  DECLARE cnt          INT DEFAULT 0;
  DECLARE sumevens    INT DEFAULT 0;
  DECLARE err_msg     VARCHAR(1000) DEFAULT '';
  WHILE (cnt < 100) DO
    IF mod(cnt,2) = 0 THEN
      SET sumevens = sumevens + cnt;
    END IF;
    SET cnt=cnt+1;
  END WHILE;
  INSERT INTO T3 values (sumevens);
  SET cnt = (SELECT 0 FROM SYSIBM.SYSDUMMY1);
  FOR cur1 AS SELECT * FROM T1 DO
    IF cur1.c1 > 100 THEN
      SET cnt = cnt + 1;
    END IF;
  END FOR;

  SET err_msg = 'Rows with values > 100 is:' ||
char(cnt);
  SIGNAL SQLSTATE '80000' SET MESSAGE_TEXT = err_msg;
END!
```

If you save the above inline SQL PL into a script file called “myScript.txt” you could execute it as follows:

```
db2 -td! -vf myScript.txt
```

## 15.2 Triggers

Triggers are database objects associated with a table that define operations to occur when an INSERT, UPDATE, or DELETE operation is performed on the table. They are activated (or “fired”) automatically. The operations that cause triggers to fire are called *triggering* SQL statements.

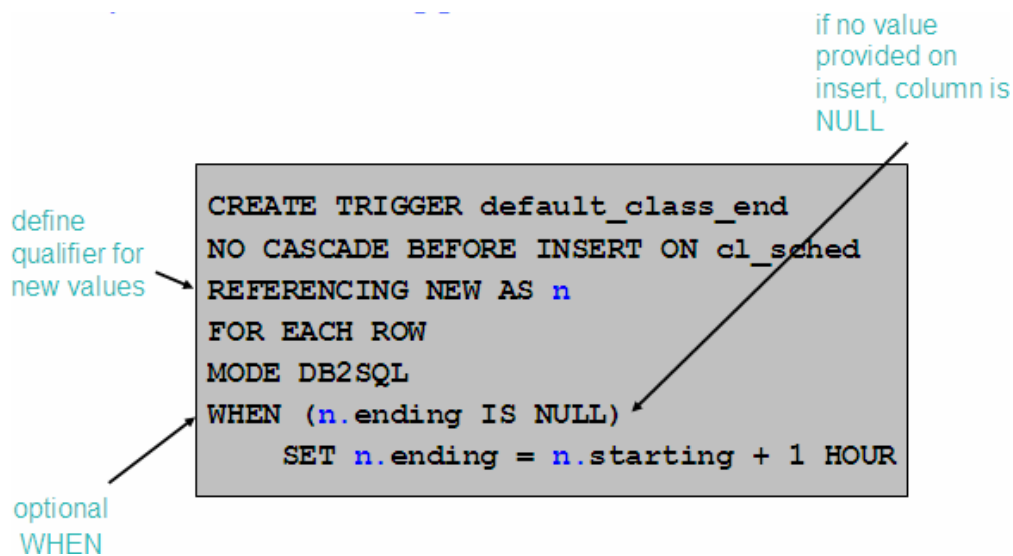
### 15.2.1 Types of triggers

There are three types of triggers: “before” triggers, “after” triggers, and “instead of” triggers.

#### Before triggers

Before triggers are activated before a row is inserted, updated or deleted. The operations performed by this trigger cannot activate other triggers (so INSERT, UPDATE, and DELETE operations are not permitted)

An example of simple before trigger is shown in Figure 15.1.



**Figure 15.1 – Example of a before trigger**

In Figure 15.1 the trigger “default\_class\_end” will be triggered before an INSERT SQL statement is performed on the table cl\_sched. This table is part of the SAMPLE database, so you can create and test this trigger yourself while connected to this database. The variable “n” in the trigger definition will represent the new value in an INSERT, that is, the value being inserted. The trigger will check the validity of what is being inserted into the table. If the column “ending” has no value during an insert, the trigger will ensure it has the value of the column starting plus 1 hour.

The following statements show how to test the trigger.

```

C:\Program Files\IBM\SQLLIB\BIN>db2 insert into cl_sched
(class_code, day, starting) values ('abc',1,current time)
DB20000I The SQL command completed successfully.

C:\Program Files\IBM\SQLLIB\BIN>db2 select * from cl_sched

CLASS_CODE DAY    STARTING ENDING
-----
042:BF      4 12:10:00 14:00:00
553:MJA     1 10:30:00 11:00:00
543:CWM     3 09:10:00 10:30:00
778:RES     2 12:10:00 14:00:00
044:HD     3 17:12:30 18:00:00
abc        1 11:06:53 12:06:53

6 record(s) selected.
  
```

The trigger “validate\_sched” shown below extends the functionality of the “default\_class\_end” trigger previously described. Again, you can create it and test it out against the SAMPLE database.

```
CREATE TRIGGER validate_sched
NO CASCADE BEFORE INSERT ON cl_sched
REFERENCING NEW AS n
FOR EACH ROW
MODE DB2SQL
BEGIN ATOMIC
-- supply default value for ending time if null
IF (n.ending IS NULL) THEN
    SET n.ending = n.starting + 1 HOUR;
END IF;

-- ensure that class does not end beyond 9pm
IF (n.ending > '21:00') THEN
    SIGNAL SQLSTATE '80000'
    SET MESSAGE_TEXT='class ending time is beyond 9pm';
ELSEIF (n.DAY=1 or n.DAY=7) THEN
    SIGNAL SQLSTATE '80001'
    SET MESSAGE_TEXT='class cannot be scheduled on a
weekend';
END IF;
END
```

### After triggers

After triggers are activated after the triggering SQL statement has executed to successful completion. The operations performed by this trigger may activate other triggers (cascading is permitted up to 16 levels). After triggers support INSERT, UPDATE and DELETE operations. Below is an example of an after trigger.

```
CREATE TRIGGER audit_emp_sal
AFTER UPDATE OF salary ON employee
REFERENCING OLD AS o NEW AS n
FOR EACH ROW
MODE DB2SQL
INSERT INTO audit VALUES (
    CURRENT TIMESTAMP, ' Employee ' || o.empno || ' sal-
ary changed from ' || CHAR(o.salary) || ' to ' ||
CHAR(n.salary) || ' by ' || USER)
```

In this example, the trigger audit\_emp\_sal is used to perform auditing on the column “salary” of the “employee” table. When someone makes a change to this column, the trigger will be fired to write the information about the changed made to the salary into another table called “audit”. The “OLD as o NEW as n” line indicates that the prefix “o” will be used to represent the old or existing value in the table, and the prefix “n” will be used to represent the new value coming from the UPDATE statement. Thus, “o.salary” represents the old or existing value of the salary, and “n.salary” represents the updated value for the column salary data.



**Instead of triggers**

Instead of triggers are defined on views. The logic defined in the trigger is executed instead of the triggering SQL statement. For example, if you perform an update operation on a view, the instead of trigger will be fired to actually perform the update on the base tables that form the view.

Triggers cannot be created from the DB2 Developer Workbench. They can be created from the Control Center or from the Command line tools (Command Window, Command Line Processor, or the Command Editor).

## Quicklab #10 – Creating a trigger in the Control Center

### Objective

Triggers are a database object used to perform business logic when a data modification operation occurs on a table. In this Quicklab, you will create a trigger using the Control Center. This trigger will keep a log of changes made to the *SALES* table for auditing purposes. You will log the user ID who made the change, as well as the time of day the change was made.

### Procedure

1. Open the Control Center.
2. For this Quicklab, you will need to create an additional table to be used for logging. Create a table with the following characteristics:

Table name: *saleslog*

First column:

Name: *userid*

Data type: *VARCHAR(128)*

Other attributes: *NOT NULL*

Second column

Name: *daytime*

Data type: *TIMESTAMP*

Other attributes: *NOT NULL*

Hint: Create this table using the *CREATE TABLE* statement in Command Editor, or use the *Create Table* wizard from Control Center.

3. From Control Center, expand the *EXPRESS* database folder. Right-click on the *Triggers* folder and select the *Create* option. The *Create Trigger* dialog window opens.
4. Fill in the following information in the dialog window:

Trigger schema: User ID of the user you are logged in as (should be the default setting)

Trigger name: *audit\_sales*

Table or view schema: User ID of the user you are logged in as (should be the default setting)

Table or view name: *SALES*

Time to trigger action: *After*

Operation that causes the trigger to be executed: *Update of columns* (do not specify any columns because we want the trigger to fire when any of the columns are updated).

Comment: *Logs all update actions on Sales table.*

The screenshot shows the 'Create Trigger' dialog box with the following configuration:

- Trigger schema: TEDWAS
- Trigger name: audit\_sales
- Table or view schema: TEDWAS
- Table or view name: SALES
- Time to trigger action:  After
- Operation that causes the trigger to be executed:  Update of columns
- Columns: BOOK\_ID, CUST\_ID, QTY, PRICE, PURCH\_DATE
- Comment: Logs all update actions on Sales table.

- On the *Triggered action* tab, select the *For Each STATEMENT* option. Use the following code for the triggered action:

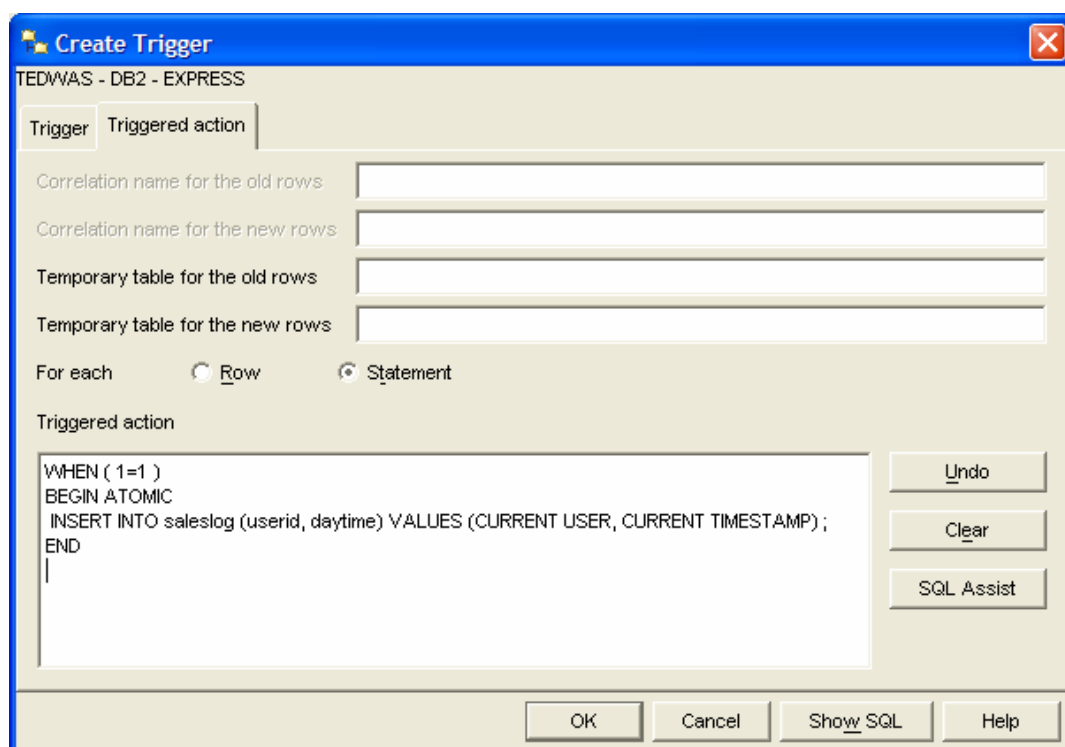
```
WHEN ( 1=1 )
BEGIN ATOMIC
```

```

INSERT INTO saleslog (userid, daytime) VALUES (CURRENT
USER, CURRENT TIMESTAMP);
END

```

(Note: A statement trigger fires once after the statement activating the trigger has completed. A row trigger specifies that the triggered action will execute every time the triggering SQL statement affects a row.)



Click the *OK* button to create the trigger.

6. You should now be able to view the trigger in the *Triggers* folder in the Control Center.
7. Query the *saleslog* table to ensure there is no data in it. Delete any rows that may be in it (`DELETE FROM saleslog`).
8. Try to update a record in the *sales* table. (Hint: use the Command Editor or the SQL Assist Wizard).
9. Check the contents of the *saleslog* table again. How many rows are in it?

## 15.3 User-defined functions (UDFs)

A user-defined function (UDF) is a database application object that maps a set of input data values into a set of output values. For example, a function may take a measurement in inches as input, and return the result in centimeters.

DB2 supports creating functions using SQL PL, C/C++, Java, CLR (Common Language Runtime), and OLE (Object Linking and Embedding). In this book, we focus on SQL PL functions because of their simplicity, popularity, and performance.

There are four types of functions: scalar, table, row, and column functions. In this chapter, we focus only on scalar and table functions.

### 15.3.1 Scalar functions

Scalar functions return a single value. Scalar functions cannot include SQL statements that will change the database state; that is, INSERT, UPDATE, and DELETE statements are not allowed. Some built-in scalar functions are SUM(), AVG(), DIGITS(), COALESCE(), and SUBSTR().

DB2 allows you to build customized user-defined functions where you can encapsulate frequently used logic. For example, consider a migration of your application from Oracle to DB2. In your application, you widely invoke Oracle's NVL() function. The equivalent built-in function in DB2 is called COALESCE. Rather than renaming all the NVL functions in your application to COALESCE, you could create a user-defined function in DB2, call it NVL and have it invoke the COALESCE function as shown below.

```
CREATE FUNCTION NVL (p_var1 VARCHAR(30),
                    p_var2 VARCHAR(30))
SPECIFIC nvlvarchar30
RETURNS VARCHAR(30)
RETURN COALESCE(p_var1, p_var2)
```

The COALESCE function then returns the first argument that is not null.

Below is another example of a scalar function. The function is called "deptname" and it returns the department number of an employee based on the employee id.

```
CREATE FUNCTION deptname(p_empid VARCHAR(6))
RETURNS VARCHAR(30)
SPECIFIC deptname
BEGIN ATOMIC
  DECLARE v_department_name VARCHAR(30);
  DECLARE v_err VARCHAR(70);
  SET v_department_name = (
    SELECT d.deptname FROM department d, employee e
    WHERE e.workdept=d.deptno AND e.empno= p_empid);
  SET v_err = 'Error: employee ' || p_empid || ' was not
```

```

found';
  IF v_department_name IS NULL THEN
    SIGNAL SQLSTATE '80000' SET MESSAGE_TEXT=v_err;
  END IF;
RETURN v_department_name;
END

```

To test the function, try the following from the Command Window or Linux/UNIX shell:

```
db2 "values (deptname ('000300'))"
```

### Invoking scalar UDFs

Scalar UDFs can be invoked in SQL statements wherever a scalar value is expected, or in a VALUES clause. Here are two examples showing a call to the COALESCE scalar function:

```

SELECT DEPTNAME, COALESCE(MGRNO, 'ABSENT') FROM DEPARTMENT

VALUES COALESCE('A', 'B')

```

### 15.3.2 Table functions

Table functions return a table of rows. You can call them using the FROM clause of a query. Table functions, as opposed to scalar functions, can change the database state; therefore, INSERT, UPDATE, and DELETE statements are allowed. Some built-in table functions are SNAPSHOT\_DYN\_SQL() and MQREADALL(). Table functions are similar to views, but since they allow for data modification statements (INSERT, UPDATE, and DELETE) they are more powerful. Typically they are used to return a table and keep an audit record.

Below is an example of a table function that enumerates a set of department employees:

```

CREATE FUNCTION getEnumEmployee(p_dept VARCHAR(3))
RETURNS TABLE
  (empno CHAR(6),
   lastname VARCHAR(15),
   firstnme VARCHAR(12))
SPECIFIC getEnumEmployee
RETURN
  SELECT e.empno, e.lastname, e.firstnme
  FROM employee e
  WHERE e.workdept=p_dept

```

To test the above function, try:

```
db2 "SELECT * FROM table(getEnumEmployee('D11')) AS t"
```

**Invoking table UDFs**

A table UDF has to be invoked in the FROM clause of an SQL statement. The TABLE() function must be applied and must be aliased. Figure 15.2 shows an example of how to invoke the function “getEnumEmployee” which we had tested earlier as well.

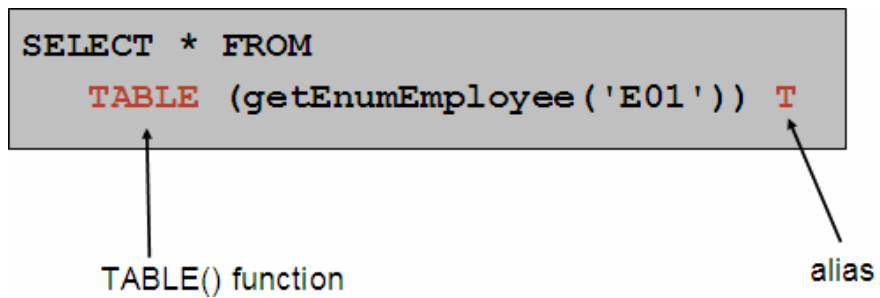


Figure 15.2 – Invoking a table function.

## Quicklab #11 – Creating a UDF using the DB2 Developer Workbench

### Objective

The DB2 Developer Workbench (DWB) is a powerful IDE for writing stored procedures and functions. In this Quicklab, you will create a scalar user-defined function in DWB. This will give you more experience with the DWB, as well as improving your familiarity with the SQL PL language for user-defined functions.

### Procedure

1. Open the DWB (Hint: it is available through the Start menu).
2. From the Data Project Explorer window, choose the project you created in the previous Quicklab and select *Open Project*.
3. Right-click the *User-Defined Functions* folder. Select the *New* menu item. Select the *SQL User-Defined Function* menu item. You could have alternatively selected the *User-Defined Function using Wizard* item if you wanted to be guided through the process using a GUI wizard.
4. The *Editor* view should open with a skeletal function. Modify the code as follows:

```
CREATE FUNCTION booktitle(p_bid INTEGER)
RETURNS VARCHAR(300)
-----
SQL UDF (Scalar)
-----
SPECIFIC booktitle
F1: BEGIN ATOMIC
DECLARE v_book_title VARCHAR(300);
DECLARE v_err VARCHAR(70);
SET v_book_title = (SELECT title FROM books WHERE p_bid =
book_id);
SET v_err = 'Error: The book with ID ' || CHAR(p_bid) || '
was not found.';
IF v_book_title IS NULL THEN SIGNAL SQLSTATE '80000' SET
MESSAGE_TEXT=v_err;
END IF;
RETURN v_book_title;
END
```

5. Build the function by right-clicking on the function and choosing *Deploy*.
6. Run the function by clicking the *Run* button in the toolbar.



7. Since the function accepts one input parameter, a dialog window appears asking you to fill in a value for the parameter.

Enter the value: 80002

What is the result?

Try again with the value: 1002

What happens this time? (Hint: Look in the *Messages* section of the *Output* view).

8. Close the DWB when you are finished.



# 16

## Chapter 16 – SQL/XML and XQuery

In this chapter we discuss pureXML, the new technology provided 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. Figure 16.1 shows the section of the DB2 “Big Picture” we will discuss in this chapter.

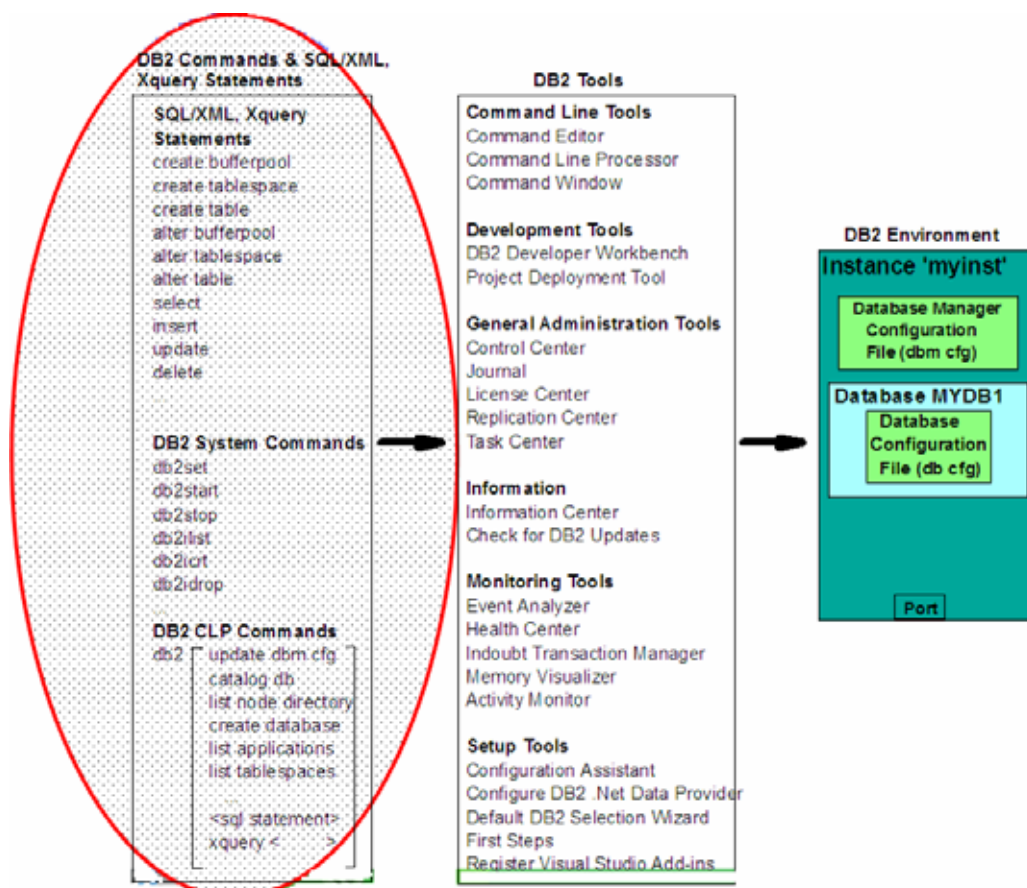


Figure 16.1 – The DB2 Big Picture: DB2 commands, SQL/XML and XQuery

## 16.1 Using XML with databases

XML documents can be stored in text files, XML repositories, or databases. There are two main reasons why many companies propose 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.

## 16.2 XML databases

There are two types of databases for storing XML data:

- XML-enabled databases
- Native XML databases

### 16.2.1 XML-enabled databases

An XML-enabled database uses a relational model for the core data storage model. This requires 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 an "old" technology, it is still being used by many database vendors. Figure 16.2 explains in more detail the two options for XML-enabled databases.

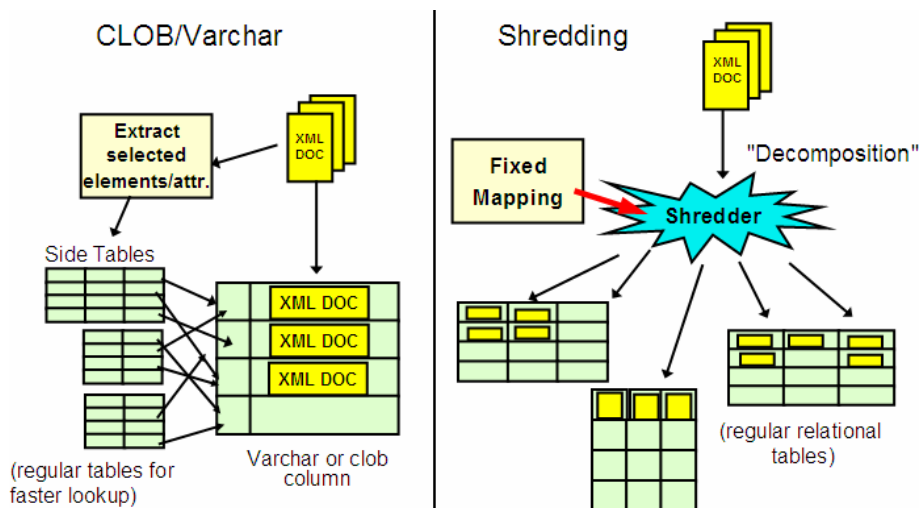


Figure 16.2 – Two options to store XML in XML-enabled databases

The left side of Figure 16.2 shows the “CLOB and varchar” method of storing XML documents in a database. Using this method, an XML document is stored as either a CLOB image or a varchar column in the database. This method is not flexible because it is hard to look for specific data elements inside an image. It is also not good for performance since the XML images are normally large, and bringing them to memory is costly. Many data servers, including DB2, do not bring images to memory, but work directly on disk. Using images is obviously not good for performance.

The other option for XML-enabled databases is called shredding or decomposition and is illustrated on the right hand side of Figure 16.2. Using this method, an entire XML document is shredded into smaller parts which are stored in tables. Using this method, the hierarchical model of an XML document is forced into the relational model. This is also not good for flexibility: a change in the XML document 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 operation, which grows even more expensive when more tables are joined.

### 16.2.2 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 images. 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 9 is currently the only commercial data server providing this support.

## 16.3 XML in DB2

Figure 16.3 below outlines how relational data and hierarchical data (XML documents) are stored in DB2 9. In the figure, assume the table `dept` is defined as follows:

```
CREATE TABLE dept (deptID CHAR(8), ..., deptdoc XML);
```

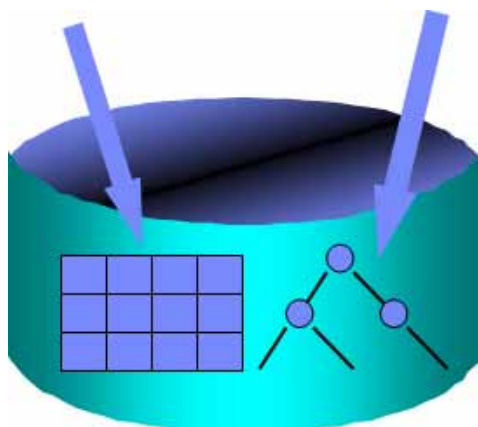


Figure 16.3 – XML in DB2

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 16.4 illustrates that with DB2 9, there are now four ways to access data:

- Use SQL to access relational data
- Use SQL with XML extensions to access XML data
- Use XQuery to access XML data
- Use XQuery to access relational data

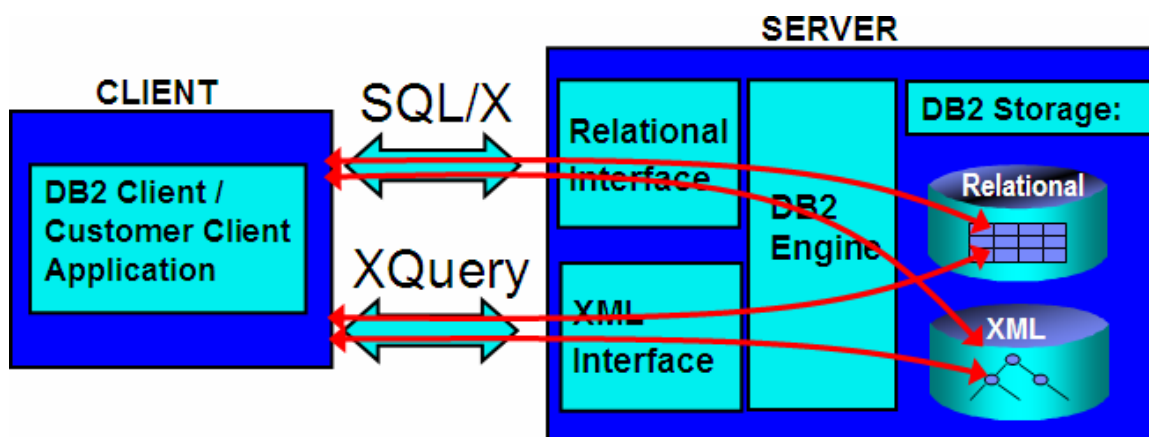


Figure 16.4 – Four ways to access data in DB2

Therefore, while an SQL person may see DB2 as a world class RDBMS that also supports XML, an XML person 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 can use this same term to denote some differing technology. Note as well that pureXML support is only provided when the database is created as a Unicode database.

### 16.3.1 DB2 9 pureXML technology advantages

Many advantages are provided by using 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 16.5 illustrates a PHP script written both with and without using pureXML. Using pureXML (the green box) the lines of code are reduced. This not only means that the code is less com-

plex, but the overall performance is improved as there are fewer lines to parse and maintain in the code.

```

<?php
$conn = db2_connect($dbname, $dbuser, $dbpass);

/* Insert Customer Documents */

$stmt = db2_prepare($conn, "VALUES (NEXT VALUE FOR
Cid)");
db2_execute($stmt);
list($Cid) = db2_fetch_array($stmt);

$fileContents = file_get_contents
("customers/c1.xml");

$stmt = db2_prepare($conn, "INSERT INTO xmlicustomer
(Cid, Info) VALUES (?, ?)");
if(!db2_execute($stmt, array($Cid, $fileContents)))
    echo db2_stmt_errormsg($stmt);
}

/* Insert Product Documents */

$fileContents = file_get_contents
("products/p1.xml");
$dom = simplexml_load_string($fileContents);

$prodID = (string) $dom["pid"];

$stmt = db2_prepare($conn, "INSERT INTO xmliproduct
(Pid, Description) VALUES (?, ?)");
if(!db2_execute($stmt, array($prodID,

```

```

db2_execute($stmt);
list($Cid) = db2_fetch_array($stmt);

$fileContents = file_get_contents
("customers/c1.xml");
$dom = simplexml_load_string($fileContents);

$custName = (string) $dom->name;
$custCountry = (string) $dom->addr["country"];
$custStreet = (string) $dom->addr->street;
$custCity = (string) $dom->addr->city;
$custProvince = (string) $dom->addr->{"prov-state"};
$custZip = (string) $dom->addr->{"pcode-zip"};
$custPhone = (string) $dom->phone;

$stmt = db2_prepare($conn, "INSERT INTO sqlcustomer
(Cid, Name, Country, Street, City, Province, Zip,
Phone, Info) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)");
if(!db2_execute($stmt, array($Cid, $custName,
$custCountry, $custStreet, $custCity, $custProvince,
$custZip, $custPhone, $fileContents) )) {
    echo db2_stmt_errormsg($stmt);
}

/* Insert Product Documents */

$fileContents = file_get_contents
("products/p1.xml");
$dom = simplexml_load_string($fileContents);

$prodID = (string) $dom["pid"];

```

Figure 16.5 – Code complexity with and without pureXML

- Changes to your schema are easier using XML and pureXML technology. Figure 16.6 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.

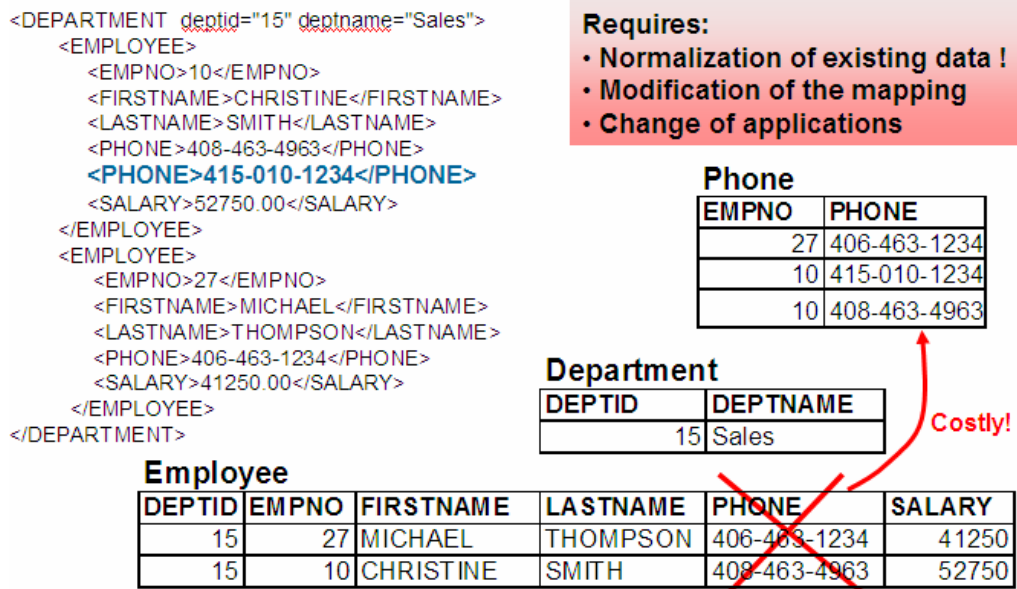


Figure 16.6 – Increased data flexibility using XML

- You can improve your XML application performance. Tests performed using pureXML technology showed huge improvements in performance for several XML applications. Figure 16.7 shows the test results for a company that switched to pureXML from older technologies. The middle column shows the results using the old method of working with XML, and the third column shows the results using pureXML in DB2 9.

Task	Other relational DB	DB2 9
Development of search and retrieval business processes	CLOB: 8 hrs Shred: 2 hrs	30 min.
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

Figure 16.7 – Increased performance using pureXML technology

### 16.3.2 XPath basics

XPath is a language that can be used to query XML documents. Figure 16.8 shows an XML document, and Figure 16.9 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.

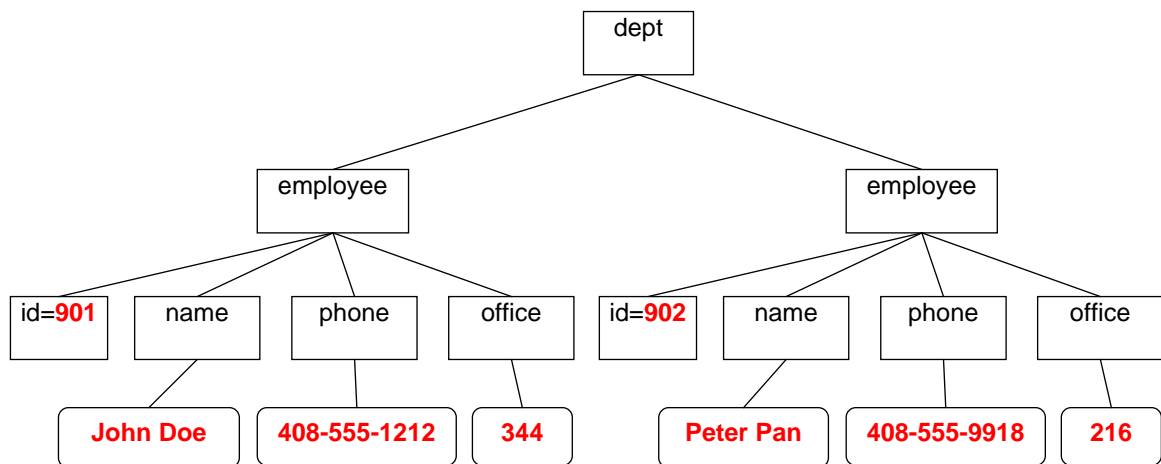


```

<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>

```

**Figure 16.8 – An XML document**



**Figure 16.9 – Parsed-hierarchical representation of the XML document in Fig 16.8**

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 Figure 16.9 in XPath you could retrieve the names of all employees using this query:

```
/dept/employee/name
```

### **XPath expressions**

XPath expressions use fully qualified paths to specify elements and/or attributes. An “@” sign is used to specify an attribute. To retrieve only the value (text node) of an element,

use the “text()” function. Table 16.1 shows XPath queries and the corresponding results using the XML document from Figure 16.9.

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 16.1 – XPath expression examples**

#### XPath wildcards

There are two main wildcards in XPath:

- “\*” matches any tag name
- “//” is the “descendent-or-self” wildcard

Table 16.2 provides more examples using the XML document from Figure 16.9

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 16.2 – XPath wildcard examples**

#### 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 n<sup>th</sup> child would be selected. For Example, *employee[2]* means that the second employee should be selected. Table 16.3 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 16.3 – XPath predicate examples

**XPath: 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 16.4 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 16.4 – XPath parent axis

**16.3.3 XQuery defined**

XQuery is a query language created for XML. XQuery supports path expressions to navigate 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 returns sequences of XML data

It's important to note that XQuery and XPath expressions are case sensitive.

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.

## 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 following XML document 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>
```

Then the following XQuery statement 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>
```

This would return the following:

```
<EmpList bldg="101">
  <name>
    John Doe
  </name>
  <name>
    Peter Pan
  </name>
</EmpList>
```

### 16.3.4 Inserting XML documents

Inserting XML documents into a DB2 database can be performed using the INSERT SQL 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 following script, which can be run from the DB2 Command Window or Linux shell using this statement:

```
db2 -tvf table_creation.txt
```

#### **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)
;
```

Note that this script file and related files are provided in the zip file **expressc\_book\_quicklabs.zip** that accompanies this book. Follow along as we describe each line in the script.

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. A UNICODE database is required to support pureXML, so this step is necessary to create the database as a UNICODE database.
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 (Figure 16.10)

Name	Size	Type
Client3227.xml	1 KB	XML Document
Client4309.xml	1 KB	XML Document
Client5681.xml	1 KB	XML Document
Client8877.xml	1 KB	XML Document
Client9077.xml	1 KB	XML Document
Client9177.xml	1 KB	XML Document
ClientInfo.xsd	2 KB	XML Schema
clients.del	1 KB	DEL File
Comment3926.xml	1 KB	XML Document
Comment4023.xml	1 KB	XML Document
Comment4272.xml	1 KB	XML Document
items.del	1 KB	DEL File

Figure 16.10 - Contents of D:\Raul directory with XML documents

These are the contents of the file clients.del

#### 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' />
```

In the clients.del file, “XDS FIL=” is used to point to a specific XML document file.

Figure 16.11 shows the Control Center after running the above script.

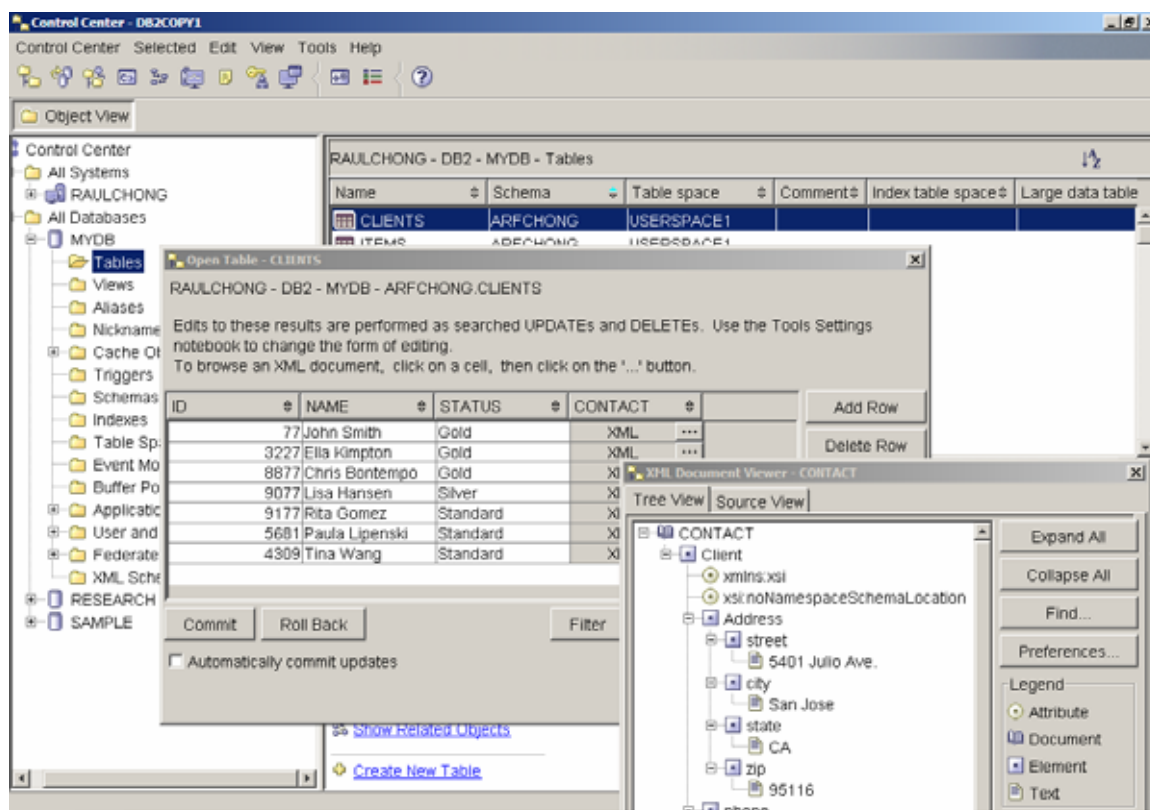


Figure 16.11 – The Control Center after running table\_creation.txt

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 16.11.

### 16.3.5 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.

#### Querying XML Data with SQL/XML

Using plain 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 16.5 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 16.5 – SQL/XML Functions**

The following examples can be tested using the “mydb” database created earlier.

#### Example 1

This is an example of a sample query problem. 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.

```
SELECT name FROM clients
WHERE xmlexists(
    '$c/Client/Address[zip="95116"]'
    passing clients.contact as "c"
)
```

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 and 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.

### 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 following query could be run:

```
SELECT xmlquery('$c/Client/email' passing contact as "c")
FROM clients
WHERE status = 'Gold'
```

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 the example 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
```

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 second line invokes 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

```
SELECT name, xmlquery(
'for $e in $c/Client/email[1] return $e'
passing contact as "c"
)
FROM clients
WHERE status = 'Gold'
```

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

This example again demonstrates the XMLQUERY function which takes an XQuery FLWOR expression, but 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'
```

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.

```
SELECT
  xmlelement (name "item", itemname),
  xmlelement (name "id", id),
  xmlelement (name "brand", brandname),
  xmlelement (name "sku", sku)
FROM items
WHERE srp < 100
```

The above query would be provide output like this:

```
<item>
  <id>4272</id>
  <brand>Classy</brand>
  <sku>981140</sku>
</item>
...
<item>
  <id>1193</id>
  <brand>Natural</brand>
  <sku>557813</sku>
</item>
```

## Query 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 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

A simple XQuery to return customer contact data

```
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, we use the FLWOR expression to retrieve client fax data

```
xquery
  for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/fax
  return $y
```

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 similar to this one (it may include the namespace by default, but we don’t show it below, otherwise this output would be harder to read as it may span several lines):

```
<fax>4081112222</fax>
<fax>5559998888</fax>
```

### Example 3

This next example 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>
```

The sample HTML returned looks like this:

```
<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>
```

#### 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

```
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>
  )
```

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:** DB2 is case insensitive and treats all the table and column names in uppercase while XQuery is case sensitive. The above resource functions are XQuery interface functions so all the table names and column names should be passed to these function as uppercase. Passing the object names in lowercase may result in an undefined object name error.

### 16.3.6 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 these statements:

```
CREATE TABLE dept (unitID CHAR(8), deptdoc XML)

CREATE TABLE unit (unitID CHAR(8) primary key not null,
                  name CHAR(20),
                  manager VARCHAR(20),
                  ...
                  )
```

You can perform a JOIN operation in either of two ways:

#### Method 1:

```
SELECT u.unitID
FROM dept d, unit u
WHERE XMLEXISTS (
  '$e//employee[name = $m]'
  passing d.deptdoc as "e", u.manager as "m")
```

In line 3 of this statement 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".

#### Method 2:

```
SELECT u.unitID
FROM dept d, unit u
WHERE u.manager = XMLCAST(
  XMLQUERY('$e//employee/name '
  passing d.deptdoc as "e")
  AS char(20))
```

In this alternate 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.

### 16.3.7 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, the query would look as follows:

```
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")
```

The equivalent query using XQuery would be:

```
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
```

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.

### 16.3.8 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
- ▶ Invoking the stored procedure DB2XMLFUNCTIONS.XMLUPDATE

In both cases, the update or delete occurs at the document level; that is, the entire XML document is replaced with the updated one. For example, if in the example below all we'd like to change is 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
```

### 16.3.9 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 this was one of the XML documents stored:

```
<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>
```

1) This statement creates an index on the attribute "Cid"

```
CREATE UNIQUE INDEX idx1 ON customer(info)
  GENERATE KEY USING
  xmlpattern '/customerinfo/@Cid'
  AS sql DOUBLE
```

2) This statement creates an index on the element "name"

```
CREATE INDEX idx2 ON customer(info)
  GENERATE KEY USING
  xmlpattern '/customerinfo/name'
  AS sql VARCHAR(40)
```

3) This statement creates an index on all elements "name"

```
CREATE INDEX idx3 ON customer(info)
  GENERATE KEY USING
  xmlpattern '//name'
  AS sql VARCHAR(40);
```

4) This statement 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);
```

---

## QuickLab #12 - SQL/XML and XQuery

### Objective

You have seen several examples of SQL/XML and XQuery syntax and have been introduced to the DB2 Command Editor and the DB2 Developer Workbench. In this lab, 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.

### Procedure

1. Create the “mydb” database and load the XML data, as discussed earlier in the chapter.
2. Using either the Command Editor or the Developer Workbench:
  - a) Retrieve all the comments XML documents from the ITEMS table in two ways, but only using XQuery
  - b) Why would issuing this SQL statement not return the exact same output?:  

```
SELECT comments FROM items
```
  - c) Retrieve the ID and BRANDNAME for the records whose XML documents have a ResponseRequested element with a value of “No”

---

### SOLUTIONS:

2a)

```
xquery db2-fn:xmlcolumn('ITEMS.COMMENTS')
xquery db2-fn:sqlquery("select comments from
items")
```

2b)

The output is different because SQL returns NULL values when a value is not present, while XQuery doesn't return anything.

2c)

```
SELECT id, brandname FROM items WHERE
XMLEXISTS('$c/Comments/Comment[ResponseRequested="No"]'
passing ITEMS.COMMENTS as "c")
```

# 17

## Chapter 17 –Developing with Java, PHP, and Ruby

This chapter discusses the basics of application development in Java, PHP and Ruby on Rails using a DB2 server. The purpose of this chapter is not to teach these languages, but to provide information pertinent to using them with DB2.

### 17.1 Application development in Java

The IBM DB2 driver for JDBC (also known as the JCC driver) has been optimized for all DB2 servers in all platforms. The jar file `db2jcc.jar` (`com.ibm.db2.jcc`) includes the type 2 and type 4 drivers. The `db2jcc.jar` file is included with any DB2 client, or can be obtained on its own (IBM DB2 Driver for JDBC and SQLJ) from the DB2 Express-C website ([ibm.com/db2/express](http://ibm.com/db2/express))

#### 17.1.1 JDBC Type 2 driver

The JDBC type 2 driver requires a DB2 client to be installed where the JDBC application is running. Figure 17.1 illustrates a JDBC application using the type 2 driver.

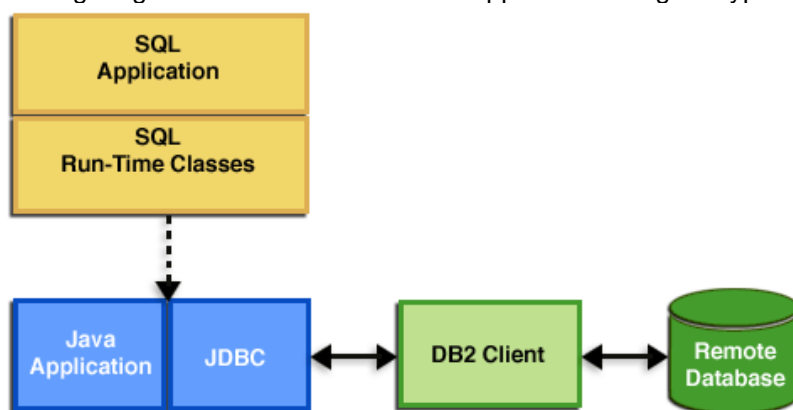


Figure 17.1 – The JDBC type 2 driver

Figure 17.2 provides a listing that shows how to establish a connection using the JDBC Type 2 driver. Note that the URL does not include hostname or port information because this is taken from the DB2 client.

```

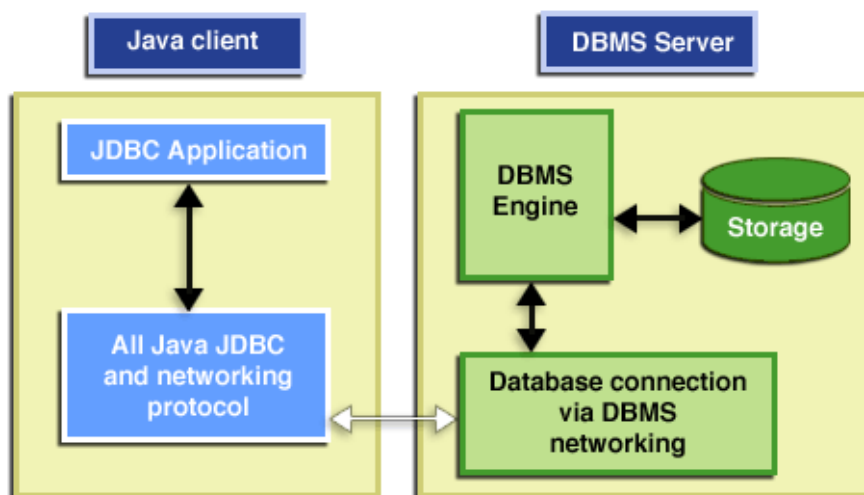
...
public static final String DB_URL = "jdbc:db2:sample";
Properties connectProperties = new Properties();
connectProperties.put("user", "db2admin");
connectProperties.put("password", "ibmdb2");
Connection connection = null
try
{
    Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
    connection = DriverManager.getConnection(url, connectProperties)
}
catch (Exception e)
throw e;
}
...

```

**Figure 17.2 – Establishing a connection using the JDBC type 2 driver**

#### 17.1.2 JDBC Type 4 driver

The JDBC type 4 driver does not require a DB2 client to connect to a DB2 server. Figure 17.3 illustrates a JDBC application using the type 4 driver.



**Figure 17.3 – The JDBC type 4 driver**

Figure 17.4 provides a listing showing how to establish a connection using the JDBC Type 4 driver. Note that the URL does include the hostname or port information.

```
...
public static final String DB_URL = "jdbc:db2://server1:50000/sample";
Properties connectProperties = new Properties();
connectProperties.put("user", "db2admin");
connectProperties.put("password", "ibmdb2");
Connection connection = null
try
{
    Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
    connection = DriverManager.getConnection(url,connectProperties)
}
catch (Exception e)
    throw e;
}
...
```

Figure 17.4 – Establishing a connection using the JDBC type 4 driver

## 17.2 Application development in PHP

PHP (PHP Hypertext Preprocessor) is an open source, platform independent scripting language that is designed for Web application development. It is one of the most widely deployed web languages in the world today. The popularity of PHP is based on the following characteristics of the language:

- Rapid, iterative development cycles with a low learning curve.
- Robust, high-performance and scalable;
- Stable and secure.
- An alternative to J2EE™ and .NET on the Web.
- Easily integrated into heterogeneous environments/systems
- Proven through widespread deployment
- Vibrant well established community

PHP is part of the LAMP stack (Linux, Apache HTTP Server, MySQL, **PHP** / Perl / Python). This is an open source web technology stack, often available on ISPs for reasonable monthly fees.

### 17.2.1 DB2 connection options for PHP

IBM supports access to a DB2 Database from PHP applications 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 to work with either PHP 4 or PHP 5. The extension is available from the PECL repository under the Apache 2.0 license. It was developed and is supported by IBM. It has full featured support for stored procedures and LOBs, and is fast, as it has been optimized for DB2.

**PDO\_ODBC:**

The PDO\_ODBC is a driver for the PHP Data Objects (PDO) extension, and offers access to DB2 databases through a standard object-oriented database interface introduced in PHP 5.1. It can be compiled directly against DB2 libraries. It provides a standard data access interface for PHP. It is fast, light weight, and object oriented. The PDO\_ODBC extension uses DB2 libraries for native access, and has been built into PHP 5.1. For more information, see these sites:

- <http://pecl.php.net/package/pdo>
- [http://pecl.php.net/package/PDO\\_ODBC](http://pecl.php.net/package/PDO_ODBC)

**Connecting to an uncatalogued DB2 database**

Listing 17.1 shows how to connect to a DB2 database using either of the two extensions previously described.

```
$host = 'localhost';
$port = 50000;
$DSN = "DRIVER={IBM DB2 ODBC DRIVER}; PORT=$port;
        HOSTNAME=$host; DATABASE=$database; PROTOCOL=TCPIP;
        USER=$user; PWD=$password";

-- If using the ibm_db2 extension --
$uconn = db2_connect($DSN, null, null);

-- If using the PDO_ODBC extension --
try {
    $uconn = new PDO("odbc:$DSN", null, null);
}
catch (PDOException $e) { print $e->errmsg(); }
```

**Listing 17.1 – Connecting to an uncatalogued DB2 database**

Listing 17.2 provides an example of a simple PHP application using the `ibm_db2` extension.

```
<?php
$sql = "SELECT name, breed FROM ANIMALS WHERE weight < ?";
$conn = db2_connect($database, $user, $password);
$stmt = db2_prepare($conn, $sql);
$res = db2_execute($stmt, array(10));
while ($row = db2_fetch_assoc($stmt)) {
    print "{$row['NAME']} is a {$row['BREED']}. \n";
}
?>
```

**Listing 17.2 – A simple PHP application using the `ibm_db2` extension**

**Configuring PHP for `ibm_db2`**

In Linux or UNIX you may need to modify the `php.ini` file as follows:

```
extension=ibm_db2.so
ibm_db2.instance_name=<instance name>
```

On Windows, modify the php.ini file as follows:

```
extension=php_ibm_db2.dll
```

Alternatively, you can download and install the **Zend Core for IBM** application suite as described in the next section, and not have to worry about these configuration issues.

### 17.2.2 Zend Core for IBM

Zend Core is a seamless out-of-the-box PHP development and production environment for business-critical web applications. It delivers reliability, productivity and flexibility needed for running PHP applications. It can be downloaded for free from <http://ibm.com/software/data/info/zendcore>

Zend Core for IBM installs DB2 and IDS clients, an optional Apache HTTP Server, PHP 5, and popular PHP extensions including ibm\_db2, and PDO\_INFORMIX. Zend Core for IBM can optionally install DB2 Express-C server, IBM Cloudscape™ server, the complete PHP manual, and sample applications for DB2. It comes with an easy to use and configure PHP environment, as shown in Figure 17.5, 17.6 and 17.7.

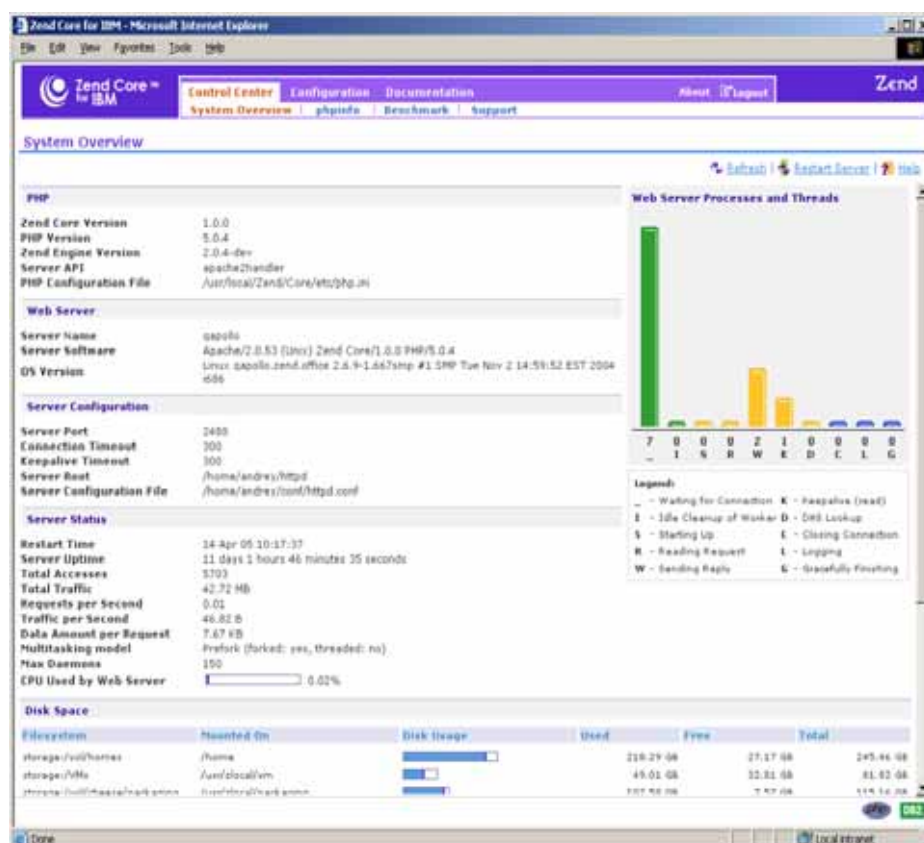


Figure 17.5 - Zend Core management and control interface



Figure 17.6 - Zend Core PHP configuration interface

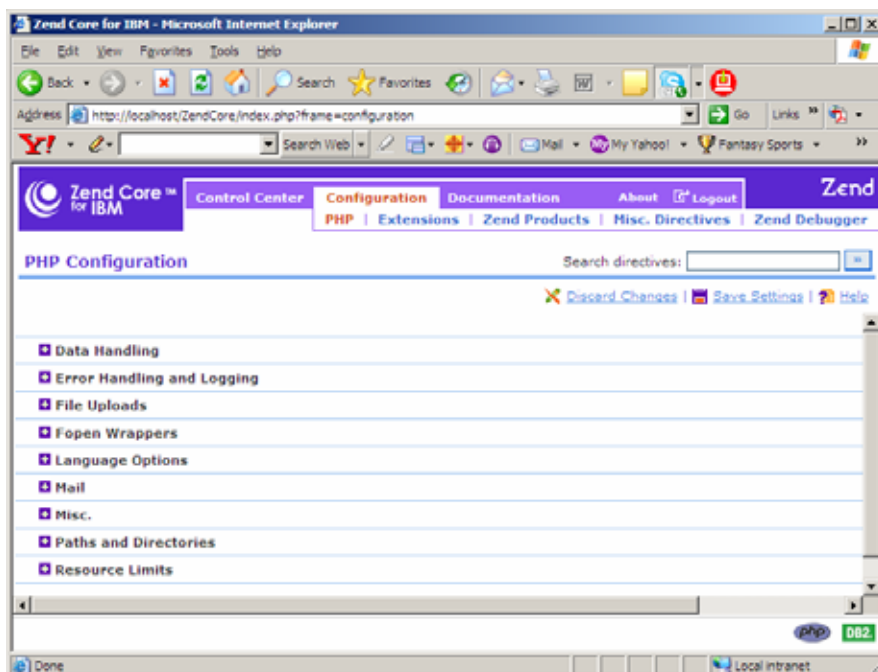


Figure 17.7 - Zend Core PHP configuration interface



## 17.3 Application development in Ruby on Rails

Ruby is an object-oriented, dynamic, cross platform scripting language. It facilitates rapid development and includes a rich library. Ruby is a simple and fun oriented programming language invented by Yukihiro Matsumoto (“Matz”) in 1995.

Rails is a full-stack framework for database-backed web applications written in Ruby. It implements the model-view-control (MVC) architecture. It is incredibly productive and easy to use. Rails is one of the fastest emerging web frameworks since 2004 and was invented by David Heinemeier Hansson.

### 17.3.1 Startup Toolkit for DB2 on Rails

IBM recognizes the importance of Ruby on Rails in the development community; therefore, it has created a package called **Startup Toolkit for DB2 on Rails**. This is an integrated installer that creates a complete DB2 Ruby on Rails development environment. It can be downloaded and used for free from the IBM alphaWorks Web site:  
<http://www.alphaworks.ibm.com/tech/db2onrails>.

The Startup Toolkit for DB2 on Rails:

- ▶ Includes an integrated installer
- ▶ Helps you to easily install and configure Ruby and Rails
- ▶ Installs DB2 Express – C 9 and tools
- ▶ Includes an IBM developed DB2 Ruby driver and a DB2 Rails Adapter
- ▶ Includes various demos and tutorials



# A

## Appendix A – Troubleshooting

This appendix discusses how to troubleshoot problems that may be encountered when working with DB2. Figure A.1 provides an brief overview of the actions to take should a problem arise.

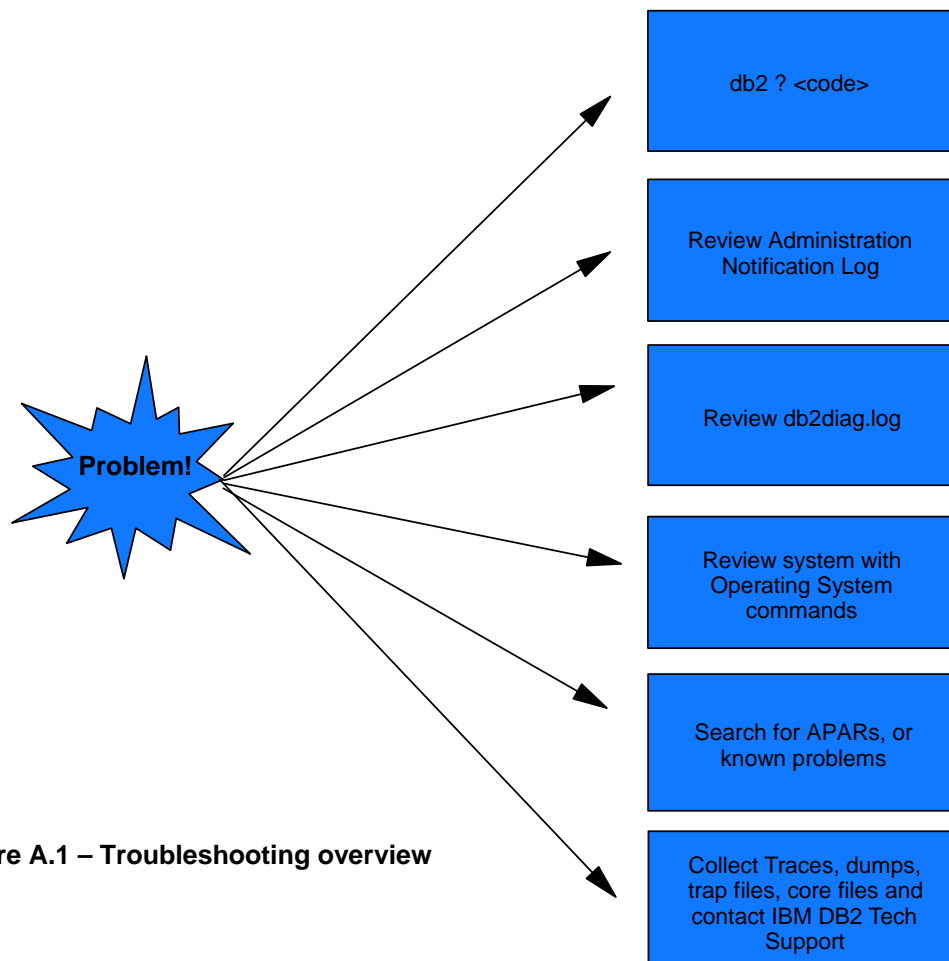
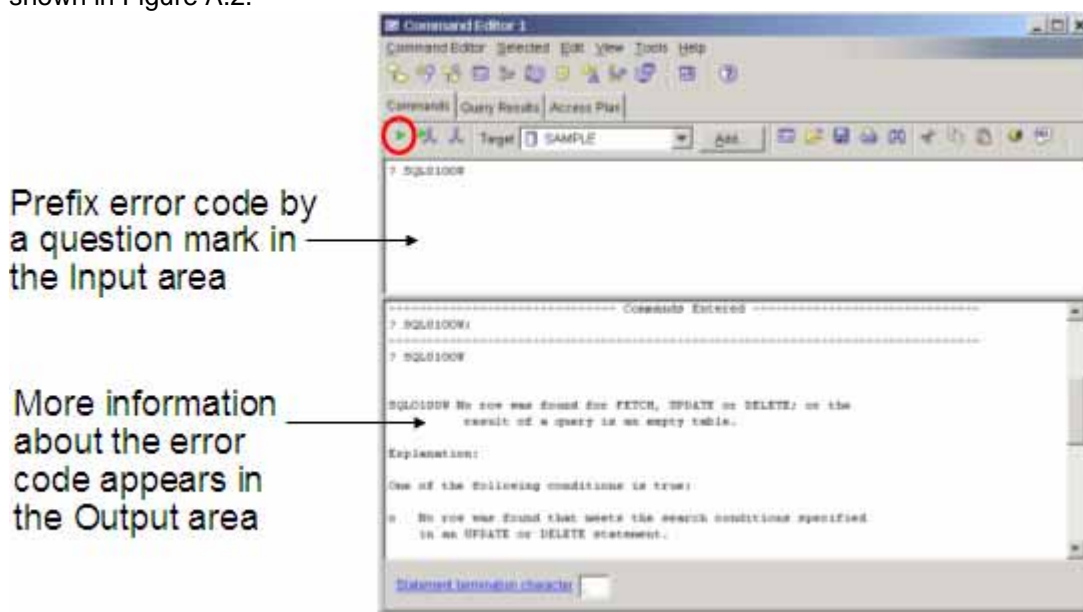


Figure A.1 – Troubleshooting overview

## A.1 Obtaining 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/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, the db2diag.log is located by default under the directory:

```
C:\Program Files\IBM\sqlllib\<instance name>\db2diag.log
```

On Linux/UNIX, the db2diag.log is located by default under the directory:

```
/home/<instance_owner>/sqlllib/db2dump/db2diag.log
```

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 and Java 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.

```
[common]
trace=0
tracerefreshinterval=300
tracepathname=/path/to/writeable/directory
traceflush=1
```

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 Fixpacks which contain code fixes for defects (APARs). The Fixpack documentation contains a list of the fixes contained in the Fixpack. When developing new applications, we always recommend using the latest Fixpack to benefit from the latest fixes. To view your current version and Fixpack level: from Control Center, select the **About** option from the **Help** menu; from the Command Window, type "db2level". Note that Fixpacks and official IBM DB2 technical support are only offered in DB2 Express-C if you purchase the 12 months subscription license.

---

## Resources

### Web sites:

1. DB2 Express-C web site:  
[www.ibm.com/db2/express](http://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](http://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  
<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp>  
The information center provides access to the online manuals. It is the most up to date source of information.
4. developerWorks  
<http://www-128.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](http://www.planetDB2.com)  
This is a blog aggregator from many contributors who blog about DB2.
7. DB2 Technical Support  
If you purchased the 12 months subscription license of DB2 Express-C, you can download fixpacks from this Web site.  
[http://www.ibm.com/software/data/db2/support/db2\\_9/](http://www.ibm.com/software/data/db2/support/db2_9/)

## 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. Understanding DB2 – Learning Visually with Examples V8.2  
Raul F. Chong, Clara Liu, Sylvia Qi, Dwaine Snow. January 2005  
ISBN: 0-13-185916-1
3. DB2 9: pureXML overview and fast start by Cynthia M. Saracco, Don Chamberlin,  
Rav Ahuja June 2006 SG24-7298  
<http://www.redbooks.ibm.com/abstracts/sg247298.html?Open>
4. 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
5. 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>
6. Information on Demand - Introduction to DB2 9 New Features  
Paul Zikopoulos, George Baklarz, Chris Eaton, Leon Katsnelson  
ISBN-10: 0071487832  
ISBN-13: 978-0071487832
7. 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>

## Contact emails

General DB2 Express-C mailbox: [db2x@ca.ibm.com](mailto:db2x@ca.ibm.com)

General DB2 on Campus program mailbox: [db2univ@ca.ibm.com](mailto:db2univ@ca.ibm.com)



## Getting started with DB2 couldn't be easier. Read this book to:

- Find out what DB2 Express-C is all about
- 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
- Prepare for the DB2 on Campus exam

The rapid adoption of XML for application integration, Web 2.0, and SOA is driving the need for innovative hybrid data servers. DB2 Express-C from IBM is a no-charge, no-limits, hybrid data server capable of managing both XML and traditional relational data with ease. No-charge means DB2 Express-C is free to download, free to build your applications, free to deploy into production, and free to redistribute with your solution. And, DB2 does not place any artificial limits on the size of database, 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 DB2 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.