

Getting Started with DB2 for z/OS™ and OS/390® Version 7 for DB2 Distributed Platform Users

Raul F. Chong
DB2 UDB Consulting Services
IBM Toronto Laboratory
July 2002

©Copyright International Business Machines Corporation 2002. All rights reserved.

Table of Contents

Preface	7
Terminology conventions	7
Why this document was written	7
Comments Welcomed	8
Acknowledgments	8
Trademarks	8
Part I. Prerequisites	9
Chapter I - Basic Mainframe Required Concepts	9
1.1 Mainframe Terminology	9
1.2 Datasets	10
1.2.1 Storage Management Subsystem (SMS)	10
1.2.2 Types of Datasets	11
1.3 TSO and ISPF/PDF	12
1.3.1 ISPF panels most commonly used	12
1.4 JCL (Job Control Language)	12
1.5 Submitting a job and reviewing the output	14
1.6 Summary	14
Chapter 2 - DB2 S/390 Installation	16
2.1 What comes with the DB2 S/390 Software Order	16
2.2 DB2 Libraries	17
2.3 Summary	18
Part II. Introduction to DB2 S/390	19
Chapter 3 - DB2 S/390 Environment	19
3.1 MVS, OS/390 and z/OS	19
3.2 Logical Partition (LPAR)	19
3.3 Virtual Storage and Address Spaces in MVS	19
3.3.1 The DSN command processor	20
3.3.2 DB2 Interactive (DB2I)	20
3.4 The DB2 subsystem	21
3.5 Summary	21
Chapter 4 - Architecture	22
4.1 System Structures	22
4.1.1 The Catalog (DSNDB06)	22
4.1.2 The Directory (DSNDB01)	22
4.1.3 Active and Archive Logs	22
4.1.4 Bootstrap Dataset (BSDS)	22
4.1.5 Bufferpools and Hiperpools	22

4.1.6 Resource Limit Facility Database (DSNRLST)	23
4.1.7 Work file database (DSNDB07)	23
4.1.8 TEMP database	23
4.1.9 Comparing DB2 S/390 and DB2 ULWO System Structures	23
4.2 Data Structures	29
4.2.1 Databases	29
4.2.2 Storage Groups	29
4.2.3 Tablespaces	29
4.2.4 Tables	30
4.2.5 Indexes	30
4.2.6 Views	30
4.2.7 Aliases	30
4.2.8 Synonyms	30
4.3 Comparing DB2 S/390 and DB2 ULWO Data Structures	30
4.4 Schema	37
4.5 Data Types	38
4.5.1 Data Type Comparison Chart	40
4.5.2 User-defined Data Types	42
4.6 IDENTITY columns	42
4.7 Sequence Objects	42
4.8 Special Registers	42
4.9 Unicode Support	43
4.10 Unique Constraints	43
4.11 Referential Integrity	43
4.12 Check Constraints	43
4.13 Comparing SQL Statements	44
4.14 Summary	44
Chapter 5 - Controlling Data Access	47
5.1 DB2 Subsystem Access	47
5.1.1 Kerberos Security Support	47
5.2 Access within the DB2 subsystem	47
5.2.1 Authorization IDs and Privileges	47
5.3 Comparing DB2 ULWO vs DB2 S/390 Authorizations and Privileges	48
5.4 Summary	51
Part III. DB2 S/390 Administration	53
Chapter 6 - DB2 S/390 Utilities and Maintaining Data	53
6.1 DB2 S/390 Utilities at a glance	53
6.2 Loading Data	54
6.2.1 The Cross Loader	54
6.3 Unloading Data	55
6.4 Reorganizing Data	55
6.5 Gathering Statistics	56
6.6 Checking Data Consistency	56
6.7 Other Utilities	57

6.8 Running Online Utilities	57
6.9 Standalone Utilities (or offline utilities)	58
6.10 Resolving Restrictive and Advisory States	59
6.11 Summary	59
Chapter 7 - Database Recovery	61
7.1 Database Recovery Concepts	61
7.2 Logging	61
7.3 Image Copies	62
7.4 Quiesce Utility	63
7.5 Recovery Concepts	63
7.5.1 Tablespace Recovery	63
7.5.2 Indexspace Recovery	64
7.5.3 Partial Object Recovery	64
7.5.4 Multiple Object Recovery	64
7.5.5 Point-inTime Recovery	64
7.6 Backup and Recovery of the DB2 Catalog and Directory	64
7.7 Disaster Recovery	64
7.8 Tracker Site Recovery	65
7.9 Comparing DB2 ULWO with DB2 S/390 Backup and Recovery	65
7.10 Summary	67
Chapter 8 - Data Sharing	69
8.1 Description	69
8.2 Data Sharing Benefits	70
8.3 DB2 ULWO Enterprise Extended Edition (EEE)	70
Chapter 9 - Performance Monitoring and Tuning	71
9.1 EXPLAIN	71
9.2 Query Parallelism	71
9.3 System Monitoring	73
9.4 Summary	74
Part IV. Application Considerations and Connectivity	76
Chapter 10 - DB2 S/390 Connectivity	76
10.1 DB2 S/390 Supported Protocols	76
10.2 DB2 S/390 as a DRDA Server	76
10.3 DB2 S/390 as a DRDA Requester	80
Chapter 11 - Application Development Environment	82
11.1 Application Environment	82
11.2 Precompiling and Binding	83
11.3 Stored Procedures	86
11.5 Triggers	86
11.6 User-defined Functions (UDFs)	86
11.7 DB2 Extenders™	87
11.8 Summary	87

Chapter 12 - Locking and Concurrency	88
12.1 Locking Data	88
12.2 Lock Modes	89
12.3 Lock Durations	89
12.4 Isolation Levels	89
12.5 Avoiding Locks	90
12.6 System Parameters	90
12.7 Claims and Drains	91
12.8 Escalation and Promotion	91
12.9 Summary	92
Part V. DB2 S/390 Tools	94
Chapter 13 - Fee-based optional DB2 Tools	94
13.1 Comparing fee-based optional DB2 tools with DB2 ULWO tools.	94
Chapter 14 - The DB2 Management Clients Package	95
14.1 DB2 Control Center	95
14.2 DB2 Installer	95
14.3 DB2 Visual Explain	95
14.4 DB2 Estimator	95
14.5 Stored Procedure Builder	95
14.6 Summary	95
Appendix A - Comparison of DB2 ULWO vs DB2 S/390 commands	97
Appendix B - DB2 S/390 Versions and OS versions	98
BIBLIOGRAPHY	99

Preface

This document is written for DB2 UNIX, Linux, and Windows® users who would like to leverage their DB2 knowledge in these platforms to understand how DB2 on the mainframe works. With analogies and simple examples using DB2 UDB UNIX, Linux, and Windows terms, we explain the equivalent concepts for DB2 on the mainframe.

DB2 for OS/390 concepts are *not* covered in detail. We provide a short description of the topics, followed by a 'DB2 UNIX, Linux, and Windows Analogy' section. After reading this document, if you need to investigate a topic further, you will be able to scan through a DB2 for OS/390 book and quickly understand the concepts in more detail. By no means is this document intended to be a complete reference, but a starting point for new users of DB2 on the mainframe.

This paper is based on Version 7 of DB2.

Terminology conventions

The full name of DB2 on the mainframe is DB2 Universal Database™ for z/OS™ and OS/390. Many people use the terms “DB2 UDB for OS/390,” “DB2 for OS/390,” “DB2 S/390®,” “DB2 for MVST™” or combinations of all of the above to refer to this same product. In this document we use the term “DB2 S/390” to reference DB2 Universal Database for z/OS and OS/390 version 7.

With respect to DB2 on distributed platforms, DB2 Universal Database Version 7.2 for UNIX, Linux, Windows and OS/2 is referenced in this document as DB2 ULWO (ULWO for “UNIX, Linux, Windows, OS/2”).

The following terms will also be used:

- Tablespace, instead of table space
- Indexspace instead of index space
- Dataset, instead of data set
- Bufferpool, instead of buffer pool

Why this document was written

DB2 S/390 is the leading relational database in the OS/390 and z/OS platforms. The mainframe continues to grow and transform itself while maintaining its reliability, security and speed. Mainframe skills, however, are hard to find; moreover, it is difficult to train new people as the mainframe environment is not flexible enough. Not everyone can install DB2 S/390, and not everyone can have his or her own mainframe to “play” with. If access to a test mainframe machine is provided, this access is limited because you may affect other users.

This working environment can slow the progress of new users attempting to learn more about DB2 on the mainframe, and is one of the reasons why it can be more difficult to train a DB2 ULWO administrator to use DB2 S/390, than the other way around.

Today it is common to see customers running two-tier or three-tier applications using DB2 S/390 as their database server. By reading this document as a “starting guide,” we hope you can gain a foundation of knowledge for DB2 on the mainframe that will help you work effectively in

an environment in which DB2 S/390 plays a major role. As an added bonus, we hope that DB2 S/390 administrators can also gain skills on DB2 ULWO by reading this material.

Comments welcomed

It is difficult enough to write a document about a product for a particular platform; it is even harder to write one that involves several platforms. This is the first version of this document, and as such, there may be omissions, typos and errors. Please feel free to contact me to have this document corrected for future versions. My email address is rfchong@ca.ibm.com

Acknowledgments

I would like to thank the following persons from the IBM Canada DB2 Application Enabling Support Systems team for the feedback they provided about this document: Michael Dang, John De Dominicis, Cheryl Raitakari, Steve Tiffney, and Alain Tremblay. Also, my especial thanks to Danny Padilla and Amyris Rada from the IBM Toronto Lab Consulting Services Team.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries: AIX, CICS, DataJoiner, DB2, DB2 Connect, DB2 Extenders, DB2 Universal Database, DataPropagator DRDA, IBM, IMS, MVS/ESA, OS/390, Parallel Sysplex, S/390, Sysplex Timer, zSeries, z/Architecture, z/OS.

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

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

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

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

Part I. Prerequisites

Chapter 1 - Basic Mainframe Required Concepts

This chapter describes briefly some of the concepts you need to operate in a mainframe environment. You should get familiar with these skills to understand some of the concepts used in subsequent sections of this document. Mainframe terminology will be compared where possible with UNIX/Linux/Windows terminology. We encourage you to review books on Job Control Language (JCL) and ISPF/PDF.

1.1 Mainframe terminology

Table 1 lists a few basic terms that are commonly used in the mainframe environment and compares them to the corresponding terms in the distributed platform (UNIX/Linux/Windows) environment:

Table 1 - Comparison of mainframe and UNIX/Linux/Windows Terms

Mainframe Term	UNIX/Linux/Windows Term
Dataset	File
DASD (Direct Access Storage Devices) Volume	Hard disk
Central storage, main storage, main memory, real storage	Main memory, RAM
Expanded storage	Extended storage
CPU, CP	CPU
CPC (Central Processor Complex)	Several CPUs (e.g., SMP system with multiprocessors)
LPAR (Logical Partition)	Partition (e.g., partitioning a hard disk in a PC)
Datasets using SYS1 as the highest level qualifier	Configuration files (e.g., .ini files, .profile file)

Other mainframe terms and keyboard keys to use:

- JCL: Job Control Language
- Job: A sequence of one or more related programs
- Abend: Abnormal termination of a job or program
- CLIST: Command List - a scripting language
- REXX: Restructured Extended Executor Language - a scripting language
- FMID: Function Modification ID - an ID used to identify a separate product or function of a product
- <ctrl> key: <enter> key. Some terminals allow both <ctrl> and <enter> to work the same.
- <pause>key: Attention key - Interrupts current TSO operation. This can be used to stop the system when it keeps asking the same question.

In ISPF:

F8 key:	Scroll down
F7 key:	Scroll up
F10 key:	Scroll to the left
F11 key:	Scroll to the right

1.2 Datasets

Creating files in a UNIX, Linux or Windows environments is a much simpler task than in the mainframe environment. On the mainframe, disk datasets have to be allocated before they can be used. Allocating a dataset means specifying a dataset name, a primary quantity size, a secondary quantity size, the units of allocation (tracks or cylinders), the volume where the dataset would be located, and several other variables.

When the contents of a disk dataset exceed the primary quantity size, the secondary quantity is used, if one was specified during allocation. There can be up to 255 secondary extents dynamically created before warning or error messages are reported. When this limit is reached, you need to reallocate a disk dataset with a larger primary quantity. At this point the smaller disk dataset can be uncataloged and deleted, and the program or job can be rerun to re-create it. Alternatively, you can create a new larger disk dataset with a different name; copy the contents of the smaller dataset to it (using standard utilities), delete the smaller dataset and then optionally rename the larger dataset if needed.

Specifying a large primary quantity in the first place may cause space to be wasted as this space is allocated on disk immediately. On the other hand, specifying a small primary quantity may cause extents to be dynamically created, which is bad for performance. You should estimate the size of your datasets based on their planned use.

Allocating a dataset can be performed through JCL, CLIST, REXX Execs, TSO, or by using the various menus under ISPF/PDF. This will be explained in a later section. The naming convention of a dataset is as follows:

<high level qualifier>.<qualifier>.<qualifier>...

Each qualifier can use up to eight characters. The total dataset name (including periods) cannot exceed 44 characters. The highest-level qualifier of a dataset name normally corresponds to the ICF catalog or an alias to the catalog name where the dataset is defined. There are different type of catalogs in MVS, but ICF is the most common one. Catalogs in MVS record the location of datasets so that there is no need to specify the volume serial number (vol-ser) of the volume that contains the dataset.

1.2.1 Storage Management Subsystem (SMS)

SMS is an automated storage management system that removes many of the manual procedures that are associated with managing datasets, such as determining which volume a dataset should be stored on, calculating the amount of space to allocate to the dataset, and determining when a dataset is no longer needed and can therefore be deleted or moved to off-line storage. When creating a dataset under SMS, you assign that dataset to a pre-defined storage class, management

class and data class. By specifying the class, SMS will perform the required operations automatically. In addition, an installation can set up automatic class selection (ACS) routines, which will automatically pick the SMS classes based on the dataset name.

1.2.2 Types of datasets

There are several types of datasets available in a mainframe environment. The different types are based on how data is organized within the dataset:

VSAM dataset organization

VSAM stands for Virtual Storage Access Method. This type of data management requires a utility program called Access Method Services (AMS) to create VSAM datasets; however, the Storage Management Subsystem (SMS) of MVS/ESA™ lets you bypass AMS and define these datasets using JCL. The DB2 datasets used to store data are VSAM linear datasets (VSAM LDS).

The following statements added to a JCL job can create a VSAM linear dataset:

```
DEFINE CLUSTER-
      (NAME(DSNCAT.DSNDBC.DSNDB06.SYSUSER.I0001.A001) -
      LINEAR -
      REUSE -
      VOLUMES(DSNV01) -
      RECORDS(100 100) -
      SHAREOPTIONS(3 3) )
      DATA -
      (NAME(DSNCAT.DSNDBD.DSNDB06.SYSUSER.I0001.A001) -
      CATALOG(DSNCAT)
```

A VSAM dataset will always have a 'data' and a 'cluster' part defined. In the above example, DSNCAT.DSNDBC.DSNDB06.SYSUSER.I0001.A001 is for the cluster and DSNCAT.DSNDBD.DSNDB06.SYSUSER.I0001.A001 is for the data.

There are three types of VSAM dataset organizations which are very similar to non-VSAM ones:

- Entry-sequenced dataset (ESDS): Similar to a non-VSAM physical sequential dataset.
- Key-sequenced dataset (KSDS): Similar to a non-VSAM indexed sequential dataset. This is the most commonly used organization in MVS.
- Relative-record dataset (RRDS): Similar to a non-VSAM direct dataset.

Non-VSAM dataset organization.

There are four types of non-VSAM dataset organization:

- Physical sequential: Records are stored one after another in consecutive sequence.
- Index sequential: Records can be accessed sequentially and randomly; an index is used.
- Direct organization: Records can be accessed at random; an index is not used but the record's disk location is.
- Partitioned organization - Partitioned datasets (PDS): This type of dataset is divided into members, each of which can be processed as if it were a separate physical sequential file.

The most useful types are physical sequential and partitioned datasets. The other two are hardly used, as their functions are better handled by VSAM files.

1.3 TSO and ISPF/PDF

TSO stands for Time Sharing Option. It is a MVS subsystem that allows terminal users the ability to invoke MVS facilities interactively. Each TSO user is given a unique address space and can allocate datasets and invoke programs just as a batch job can.

ISPF stands for Interactive System Productivity Facility. It runs under TSO. PDF (Program Development Facility) is part of ISPF, and it is used to develop programs and job streams. After you log on to TSO, an ISPF panel will normally come up automatically. If it doesn't come up automatically, you can type the command `pdf` or `ispf` to have it started. The ISPF panel will provide you with a list of functions and options available at your installation. Type the desired option at the `Option===>` line. This is referred to as the command line on the panel.

1.3.1 ISPF panels most commonly used

Listing datasets

To list datasets, you can type `=3.4` on an ISPF panel command line. This takes you to a screen where you can type in one or more qualifiers of a dataset. Pressing `<ctrl>` lists all the cataloged datasets, starting with the qualifiers provided. An asterisk can be used as a wild character as in the UNIX/Linux/Windows environment. This ISPF option is the equivalent to listing files in UNIX/Linux using the `ls` command or on Windows/DOS prompt using the `dir` command.

Allocating a dataset

To allocate a sequential dataset, type `=3.2` on an ISPF panel command line. Normally you would enter first the name of an existing dataset already allocated and press `<ctrl>` so that the allocation properties of this dataset are brought up. Then you can choose "a" to allocate the new dataset which will use the same allocation properties. In other words, a common method used to allocate a new dataset is by copying the dataset attributes of an existing one and then changing its attributes as required (that is, name, space, volume, etc.).

Copying a dataset

To copy a dataset, type `=3.3` on an ISPF panel command line.

Note: When you use ISPF and enter a dataset name without single quotes, ISPF prefixes your TSO ID to the dataset name. If you don't want this to happen, use single quotes before the start and at the end of the dataset name.

Using the equal (=) symbol is a shortcut to take you to the main ISPF menu. For example, the use of `= in =3.2` means that you will be taken first to the main ISPF menu, and then you will be taken to option 3, followed by option 2.

1.4 JCL (Job Control Language)

The Job Control Language, or JCL, is a set of control statements that provide the specifications necessary to process a job in batch mode. When you submit a job to be processed by MVS, the

job is treated as a whole. The job begins with the execution of the first program and continues until the last program has finished executing, unless an unforeseen error condition occurs.

It is rare to write a JCL job from scratch. Normally you only need to copy (in part or whole) from existing JCL jobs and then modify that copy to perform the desired tasks. Three basic JCL statements are:

JOB

The first statement of any job identifies the name of the job to MVS by supplying a job name. It also provides for accounting information, the job submitter's name, and whom to notify in TSO when the program finishes or abends.

EXEC

Indicates the name of a program to be executed. Because each program to be executed in a job is a job step, the EXEC statement helps you locate quickly the number of steps in your JCL. When "PGM=" follows an EXEC statement, you will be executing a program. If this is not the case, then you would be executing a JCL PROC (JCL procedure). A JCL procedure is similar to a JCL job in that it may contain several steps and call different programs. A JCL procedure can be called from several JCL jobs.

DD

This statement is normally required for every input and output file that is processed by the program.

Here is an example of a JCL job:

```
//RAULDSNT JOB 6230,'DSNTIAUL',CLASS=C,MSGCLASS=X,REGION=0M,
//          NOTIFY=TS56692
//*****
//*
//* DATE      : March 30th, 2002
//* USING DSNTIAUL TO UNLOAD DATA FROM TABLE DSN8710.EMP
//*
//*****
//UNLOAD EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD CYST=*
//SYSTSIN DD *
DSN SYSTEM(NJLU)
RUN PROGRAM(DSNTIAUL) PLAN(DSNTIB71) -
LIB('NJLU710.RUNLIB.LOAD')
//SYSPRINT DD CYST=*
//SYSUDUMP DD CYST=*
//SYSREC00 DD DSN=TS56692.UNLOAD.SYSREC00,
//           UNIT=SYSDA,SPACE=(32760,(1000,500)),DISP=(,CATLG),
//           VOL=SER=SCR03
//SYSPUNCH DD DSN=TS56692.UNLOAD.SYSPUNCH,
//           UNIT=SYSDA,SPACE=(800,(15,15)),DISP=(,CATLG),
//           VOL=SER=SCR03,RECFM=FB,LRECL=120,BLKSIZE=1200
//SYSIN DD *
DSN8710.EMP
```

Figure 1 below shows the exact same JCL previously provided, but each statement is explained in more detail:

the figure presented in the chapter should provide a good summary of the most important JCL statements and their use.

Chapter 2 - DB2 S/390 Installation

Installation of software in a mainframe environment requires coordination with your MVS System Administrator or System Programmer. You will need to be granted different authorizations. SMP/E (System Modification Program/Extended) is used to install/update/remove software. Like the UNIX environment, SMP/E allows you to install the software without committing it permanently. The normal SMP/E steps to follow, in order, are: RECEIVE, APPLY and ACCEPT:

1. RECEIVE copies the software code to the distribution libraries.
2. APPLY copies this code to the Target libraries
3. ACCEPT commits the changes that were applied and makes them permanent.

An SMP/E REJECT can also be performed after an installation to discard any applied changes.

Installing DB2 S/390 can be summarized in the following steps:

1. SMP/E RECEIVE the DB2 software.
2. Execute the CLIST provided in received dataset DSN710.SDSNCLST using as input the member DSNTIDXA of received dataset DSN710.SDSNSAMP.
3. The CLIST will display installation panels that you need to complete.
4. After all installation panels have been completed, the CLIST will create PDS dataset <prefix>.NEW.SDSNSAMP containing tailored JCL installation jobs.
5. Edit each of the JCL installation jobs as required; some jobs may be optionally run. Note that some of the tailored jobs execute a SMP/E APPLY and SMP/E ACCEPT.
6. Submit each installation job following the order provided in the *Installation Guide*.
7. The successful execution of all the installation jobs will determine the success of the DB2 software installation.
8. Run verification programs to confirm that DB2 has been correctly installed.

SMP/E ACCEPT jobs are not normally executed until the product is fully tested in your environment, especially if fixes have been applied on top of the base code.

To simplify the DB2 S/390 installation process, the DB2 Installer GUI tool can be used from a workstation connected to the mainframe. This tool comes with the DB2 Management Client Package, which is described in more detail in a later section.

For detailed information about DB2 S/390 installation, refer to the *DB2 UDB for OS/390 and z/OS v7 Installation Guide*.

2.1 What comes with the DB2 S/390 software order

When you order DB2, you receive standard label 9-track magnetic tapes, 3480 cartridges, or 4mm cartridges, depending on the feature you ordered. If you ordered a custom-built product delivery offering (CBPDO), your order may differ. In addition, you may also receive maintenance tapes containing the latest fixes you may need to apply on top of the base code.

The DB2 Management Clients Package, Net.Data, and DB2 REXX Language Support are free-of-charge features, but they must be ordered separately.

The DB2 Management Clients Package includes: DB2 Installer, Estimator, Visual Explain, DB2 Connect, and DB2 for OS/390 and z/OS Control Center Enablement. Most of the tools are offered to you on a CD-ROM.

2.2 DB2 libraries

During the SMP/E processing, DB2 code is loaded into the distribution and target libraries. The distribution libraries are used to maintain DB2 code and contain the master copy of all elements for your DB2 system. The target libraries contain the various DB2 components. DB2 target libraries are updated by corrective service.

Distribution libraries will normally have a name like *<prefix>.ADSNxxxx* (libraries received by SMP/E), while target libraries will have a name like *<prefix>.SDSNxxxx* (Libraries applied by SMP/E). The *<prefix>* value is determined by the installer. Libraries of the format *<prefix>.NEW.SDSNxxxx* correspond to tailored libraries obtained after executing the installation CLISTs. Table 2 shows the most common target libraries.

Table 2 - Comparison of DB2 S/390 and DB2 ULWO libraries.

Library	Comment	Analogy to DB2 ULWO
prefix.SDSNDBRM	This library contains the system DBRMs for DB2 Version 7.	This library is equivalent to sqllib\bnd storing system bind files.
prefix.SDSNEXIT	This program library is empty when first created. The installation jobs put DSNHDECP, the DSNZPxxx subsystem parameters load module, and user exit modules into this library.	Similar to sqllib<instance name> where db2system file is stored. This is a binary file containing the database manager configuration (dbm cfg).
prefix.SDSNLOAD	This library contains Version 7 load modules.	Similar to sqllib\bin (in Windows) where the DB2 code resides. In UNIX (AIX®,HP,SUN) different directories are used.
prefix.SDSNSAMP	This initialization library contains the sample applications and data, the jobs for installing and migrating, the default installation and migration parameters, and catalog initialization data for DB2. The JCLIN for each FMID is stored in this library.	Somewhat similar to sqllib\samples, but SDSNSAMP contains other samples used for 'system' type operations.
prefix.DBRMLIB.DATA	Library for DB2 sample application DBRMs.	N/A

Library	Comment	Analogy to DB2 ULWO
prefix.RUNLIB.LOAD	DB2 sample application load module library. Two commonly used sample applications are DSNTIAUL (To unload data), and DSNTIAD (to run dynamically SQL from a JCL job).	N/A
prefix.SRCLIB.DATA	DB2 declaration library for sample application include files.	N/A

2.3 Summary

In this chapter, we briefly covered the steps required to install DB2 S/390. We also described some of the libraries created at installation time and compared them to similar libraries in DB2 ULWO. It was important to mention them as they are frequently used in JCL jobs during normal DB2 operations by users.

Part II. Introduction to DB2 S/390

Chapter 3 - DB2 S/390 Environment

3.1 MVS, OS/390 and z/OS

MVS (Multiple Virtual Storage) has been known for years to be one of the most important operating systems in the mainframe environment. OS/390 bundles MVS with other IBM software products which are now tested jointly. An example is the Language Environment, or LE . This software product provides a runtime environment for programs generated with C/C++ for MVS/ESA, Cobol, etc. Prior to OS/390 there were many concerns from customers about compatibility of different versions of such products with MVS. Now that these products are tested jointly and provided as a bundle with OS/390, the concerns have been diminished.

z/OS is the next generation of OS/390 and is based on the new 64-bit z/Architecture™. z/OS is the foundation for the future of zSeries™ servers. The core of OS/390 and z/OS is still the MVS operating system. For more information about OS/390 and z/OS with the features and functions bundled, refer to this website:

<http://www.ibm.com/servers/eserver/zseries/zos/>

3.2 Logical partition (LPAR)

A LPAR is a logical partition where users can have different images of different operating systems in one mainframe machine. For example, a mainframe machine can have a LPAR with OS/390 installed to be used for development purposes, another LPAR also with OS/390 installed to be used for test purposes, and another LPAR with Linux installed.

3.3 Virtual storage and address spaces in MVS

Virtual storage is a facility that simulates a large amount of main storage (real memory) by treating DASD storage (Direct Access Storage Devices - physical disks) as an extension of real storage. Main storage consists of millions of individual storage locations that are referred to by an address. An address space is simply the complete range of addresses bounded by a beginning and ending address that can be accessed by the computer. MVS not only simulates more storage (virtual storage), but it also uses real memory to simulate several address spaces, each of which is independent of the others. Thus, in order to refer to a particular byte of virtual storage under MVS, you need to know the address, and the address space to where this address applies.

DB2 S/390 uses the following address spaces:

- System Services Address Space (SSAS or MSTR)
Manipulates most of the structures in user-created databases. It is also known as MSTR since the address space name used is *ssnm*MSTR, where *ssnm* stands for the subsystem name.
- Database Service Address Space (DSAS or DBM1)
Performs a variety of system-related functions. It is also known as DBM1 since the address space name used is *ssnm*DBM1, where *ssnm* stands for the subsystem name.

- Internal Resource Lock Manager (IRLM)
Controls DB2 locking
 - Distributed Database Facility (DDF)
Provides support for remote requests
 - DB2-established Stored Procedure Address Space (SPAS)
Provides an isolated execution environment for user-written SQL programs at a DB2 server.
 - WLM (Work Load Manager)- Established Stored Procedure Address Space
Zero to many address spaces for the execution of stored procedures and user-defined functions. WLM-established address spaces are isolated from other stored procedures or user-defined functions running in other address spaces.
 - Allied Address Spaces
 - CICS® (Customer Information Control System). Provides online transaction management for applications.
 - IMS™ (Information Management System). Provides a *hierarchical* database manager as well as a transaction manager.
 - TSO (Time Sharing Option) allows for interactive time sharing capabilities from remote terminals. You can use two different command processors through TSO:
 - . DSN command processor
 - . DB2 Interactive (DB2I)
- Attachment Facilities:
- RRSF (Recoverable Resource Services Attachment Facility). Coordinates resource commitment between DB2 and other resource managers.
 - CAF (Call Attachment Facility). This is used as an alternative to the DSN command processor.

3.3.1 The DSN command processor

The DSN command processor can be invoked from TSO foreground or through a batch job. It allows you to invoke DB2 commands. The example JCL provided in section 1.4 (Figure 1) shows a call to DSN through a batch job.

3.3.2 DB2 Interactive (DB2I)

DB2I allows you to perform several tasks by entering values on its ISPF panels. Through DB2I you can:

- Invoke SPUFI (Sequential Processing Using File Input), which allows you to perform dynamic SQL statements.
- Perform DB2 commands.
- Invoke DCLGEN (declaration generator) to generate SQL and source language data type declarations.
- Prepare a DB2 application program to run (precompile, compile, bind, link, etc.).

- Invoke the DB2 precompiler.
- Bind, rebind, or free plans or packages.
- Run SQL programs.
- Invoke utilities.

3.4 The DB2 subsystem

DB2 is a subsystem of MVS. A subsystem is a software product that operates in its own address space under the control of MVS. Any operation within this address space, however, is of no concern to MVS.

3.5 Summary

In this chapter we explained the differences between MVS, OS/390 and z/OS. We explained the concepts of LPAR, virtual storage and address spaces, as well as a brief description of the DSN processor, DB2I and the DB2 subsystem. Having a better understanding of these concepts will help you learn the subsequent sections.

The table below shows equivalent command processors for DB2 S/390 and DB2 ULWO.

Table 3. DB2 S/390 and DB2 ULWO Command Processors

DB2 S/390 Concept	DB2 ULWO Analogy
DB2 S/390, through TSO, provides the DSN command processor and DB2I.	DB2 ULWO provides the Command Line Processor (CLP), the Command Window (Only on Windows platforms), the Command Center GUI Tool, and the Control Center GUI Tool.
TSO in batch (IKJEFT01) can also invoke the DSN processor.	The CLP background process: db2bp
DSNTEP2 program in a JCL job using other datasets containing SQL as input.	<code>db2 -tvf <script file></code>

The DSN and DB2I command processors are native to DB2 S/390. The Control Center GUI tool, Command Line Processor (CLP), Command Window, and the Command Center GUI tool can be used from a DB2 ULWO client machine connected to a DB2 S/390 system to perform operations from these tools. As we will see in a later chapter, the Control Center GUI tool can be used to administer databases from DB2 S/390 and DB2 ULWO.

Chapter 4 - Architecture

The sections below in this chapter describe the system and data structures of DB2 S/390. Many of these concepts will be covered in more detail in other chapters of this document.

4.1 System structures

4.1.1 The catalog (DSNDB06)

The DB2 catalog consists of tables containing information about everything defined to the DB2 subsystem. It is contained in the system database DSNDB06. System tables start with a qualifier of 'SYSIBM'. There is *one* catalog for the entire DB2 subsystem.

4.1.2 The directory (DSNDB01)

The DB2 directory (DSNDB01) is a database that contains information in synch with the catalog, but it is in internal format and cannot be queried using SQL. The DB2 directory also contains information required to start DB2 and to perform normal DB2 operations. It consists of five tablespaces:

- SCT02: Plans with the internal form of SQL.
- SPT01: Packages with the internal form of SQL.
- SYSLGRNX: Used to speed up reading logs during recovery.
- SYSUTILX: Keeps track of utilities.
- DBD01: Contains the DBDs (database descriptors), which are internal control blocks that describe the databases existing within DB2.

4.1.3 Active and archive logs

Logs are datasets where DB2 keeps track of every data change and significant events as they occur. You can have from two to 31 datasets used for the active logs in single logging mode, and from four to 62 active log datasets in dual logging mode. When the active logs are full, the offload process is started, which copies the active logs to physical sequential datasets (archive logs) on DASD or tape.

4.1.4 Bootstrap dataset (BSDS)

This dataset contains critical information to DB2. It contains a “table of contents” for the logs, including the dataset names and the record ranges those datasets include for which RBAs (relative byte address) or LRSNs (logical sequence number for data sharing) are used. The BSDS is used in recovery, and when starting or stopping DB2. This dataset is so important that two copies are created by default.

4.1.5 Bufferpools and hiperpools

Bufferpools are areas of virtual storage used to cache pages of tablespaces or indexes. There are predefined bufferpools that you can activate with the `-ALTER BUFFERPOOL` command by assigning them a size other than zero.

For example, from the DB2 Commands panel in DB2I execute this command to activate bufferpool BP32K0:

```
-ALTER BUFFERPOOL(BP32K0) VPSIZE(1000) HPSIZE(10000)
```

For each of the four page sizes currently supported (4KB, 8KB, 16KB and 32KB), there are the following predefined bufferpools: BP0-BP49 (for 4KB pages) BP8K0-BP8K9 (for 8KB page size), BP16K0-BP16K9 (for 16KB page size) and BP32K0-BP32K9 (for 32KB page size).

In addition to bufferpools, DB2 S/390 also provides *hiperpools*. A hiperpool is an extension to a bufferpool and must always be associated with a bufferpool. Bufferpools hold the most frequently accessed data, while hiperpools serve as a cache for data that is accessed less frequently. When a page in the bufferpool is no longer needed, it is moved to the hiperpool; thus, it works as a second level of cache.

4.1.6 Resource Limit Facility Database (DSNRLST)

This database contains tables that specify limits on the amount of processor time allowed for the execution of dynamic SELECT, UPDATE, DELETE and INSERT SQL. The START RLIMIT command is used to start this facility.

4.1.7 Work file database (DSNDB07)

This database is used to provide temporary storage for processing SQL statements that require working space. DSNDB07 is used as the work file database in a non-datasharing environment. In a datasharing environment, each DB2 member has its own work file database, and only one of them can use the name 'DSNDB07'.

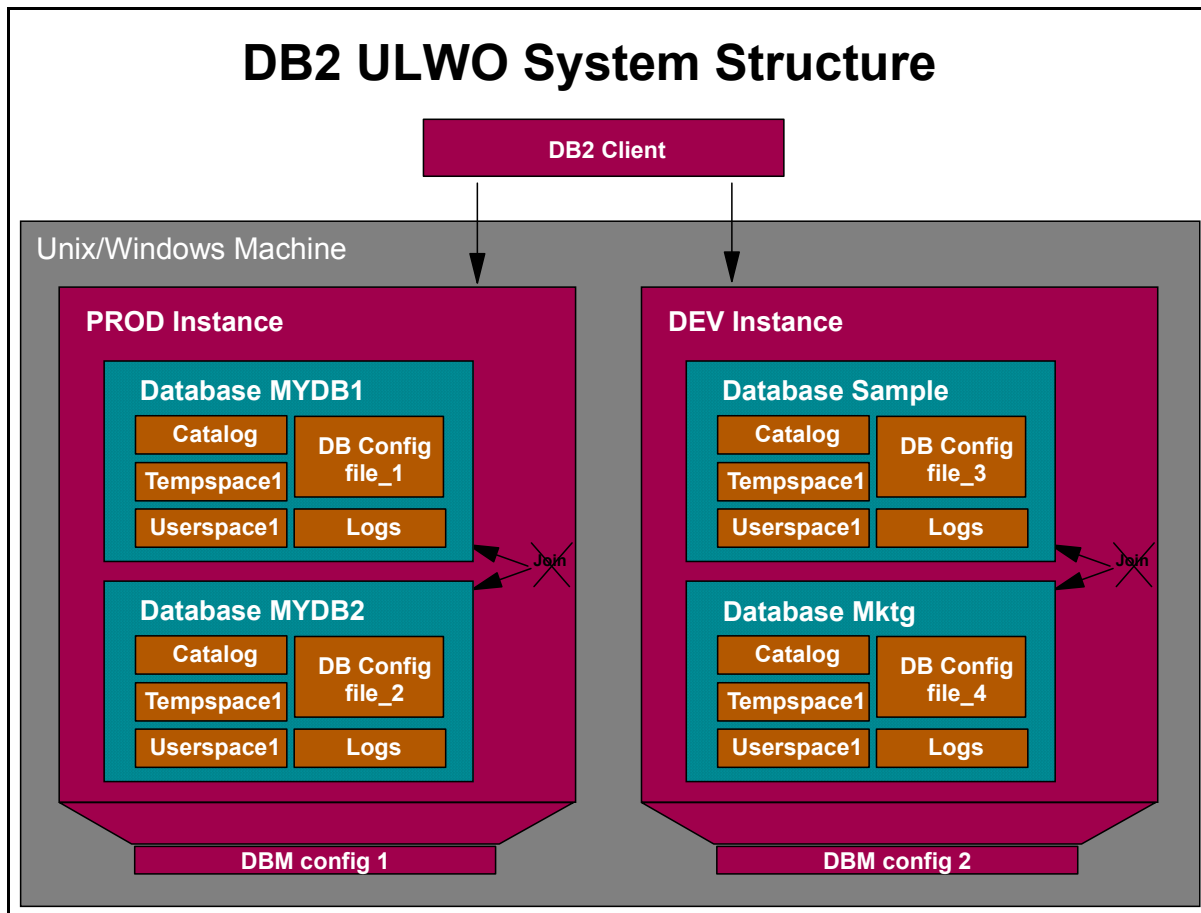
4.1.8 TEMP database

The TEMP database is used for declared temporary tables only. DB2 stores all declared temporary tables in this database. You can create one TEMP database for each DB2 subsystem or datasharing member.

4.1.9 Comparing DB2 S/390 and DB2 ULWO system structures

Review Figures 2 and 3 below. They show you a simplified view of the DB2 ULWO structure and the DB2 S/390 structure, respectively.

Figure 2



Instance vs. subsystem

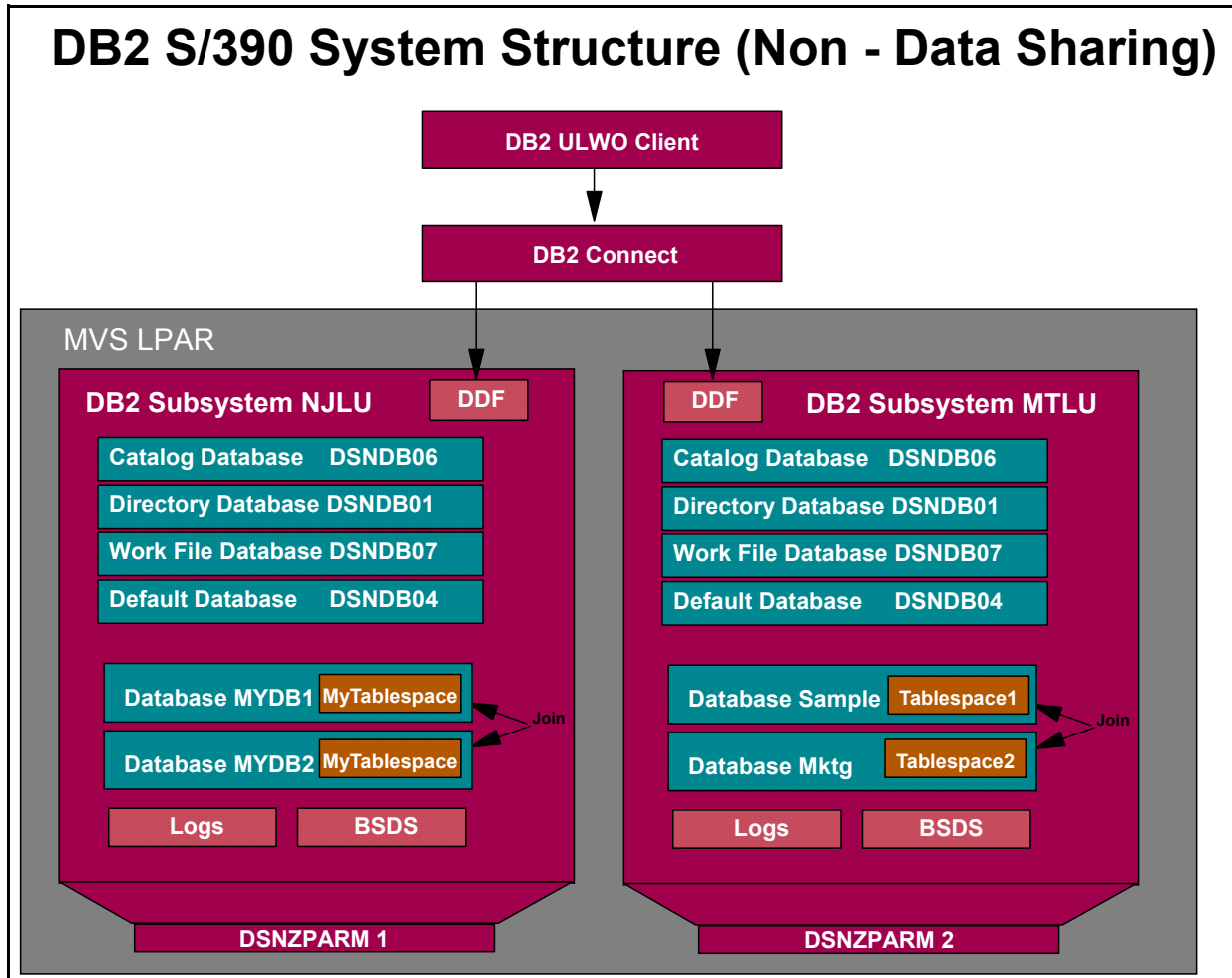
In DB2 ULWO an *instance* provides an independent environment where database objects can be created and applications can be run against them. When an instance is created, links to the DB2 code are generated. Several instances can be created in the same machine. In Windows platforms, you can only install one version of DB2 at a given fixpack level. Thus, all instances created in DB2 UDB for Windows will be linked to the same DB2 code. In UNIX platforms, you can install several versions of DB2 in the same machine as they are installed in different paths; however, only one fixpack level per version is allowed. Thus, in DB2 UDB for UNIX, you may have several instances linking to different code sets.

In DB2 S/390, a DB2 *subsystem* provides a separate DB2 environment similar to a DB2 ULWO instance. Several DB2 S/390 subsystems can be installed in the same machine Logical Partition (LPAR), and they can only communicate with each other through the Distributed Data Facility - DDF (not considering data-sharing systems which will be covered later in this document).

Different DB2 S/390 subsystems at different versions can be installed in the same LPAR. You can also have different DB2 S/390 subsystems at the same version but with different maintenance levels installed in the same LPAR. In both of these cases different code sets are used. For example, an installation may have DB2 S/390 V6 at maintenance level 0112, DB2

S/390 V7 at maintenance level 0106, and DB2 S/390 V7 at maintenance level 0110 installed in the same machine LPAR. DB2 subsystems running with the same version and at the same maintenance level in a LPAR, are also allowed; in this case the DB2 code can be shared.

Figure 3



Services, processes, address spaces

In DB2 UDB for Windows, several services are launched when the DB2 engine is started using the command db2start. In DB2 UDB for UNIX, several processes are started (e.g., db2sysc) when the engine starts with db2start. Similarly, when DB2 S/390 is started the MSTR, DBM1, IRLM and DDF address spaces will start. Each of these address spaces are launched by using a JCL proc, and you can check the MVS system log through ISPF (SDSF, log) to confirm they are running.

In DB2 ULWO there are agents that handle communications between remote clients and the DB2 engine. In DB2 S/390, the distributed data facility needs to be started to allow such communication. To start DDF, you can issue -start ddf from the DB2I command panel. To display its status issue the command -display ddf.

DB2 ULWO does not externalize processes that handle locks other than db2dlock for deadlock detection. DB2 S/390 uses IRLM to handle locks.

Redirecting commands to a specific instance or subsystem

In DB2 ULWO, you can direct commands to a specific instance by setting the value of the DB2INSTANCE variable (`set DB2INSTANCE=<instance name>`) or “attaching” to the instance using a node previously defined (`attach to <nodename>`). In DB2 S/390 you can execute DB2 commands from different places. The `start db2` command is the only command that is normally executed from the MVS console. Because you can have different DB2 subsystems installed, a command prefix is required for MVS to know to which DB2 subsystem this command should be applied to. This command prefix can have 1 to 8 characters with the default of -DSN1, where DSN1 is the default MVS subsystem name for DB2. Many installations use the old default of - as the command prefix. For example, if you would like to start a DB2 S/390 V7 subsystem that is associated with the command prefix of #, you would execute the command `#start db2` from the MVS console. If you would like to start a DB2 S/390 V6 that is associated with a command prefix of -, you would execute the command `-start db2` from the MVS console. Each subsystem is running on its own address space, and can run concurrently.

The above command prefix is only required when performing DB2 commands from a place where you can access different DB2 subsystems (like the MVS console). If you are performing a DB2 command from other applications like CICS attach, or DSN using TSO attach, you would not need to specify different command prefixes because these applications can only connect to one DB2 subsystem at a time. . The hyphen, however, should still be prefixed to the command regardless of which subsystem you are connecting to for this case.

Names for your instance and subsystem

In DB2 ULWO names you may need are:

- The instance name and
- The database name.

When connecting to a database you will also need the TCPIP address and port for the instance (If connecting using TCPIP). Other network protocols will need other information.

In DB2 S/390, several names are used to identify a subsystem:

- Subsystem ID (ssid).
Because DB2 is a subsystem of MVS, it will have a subsystem name or subsystem ID (ssid) that MVS can use to identify it. The default ssid is DSN1.
- The location name
This is the name specified when using the CATALOG DCS DATABASE command from a DB2 Connect machine, which refers to it as the target database name. It can have from one to 16 characters.

- The LU Name

This is the name by which VTAM can recognize the local subsystem. The unique name must be unique within the network of connected systems and can have from one to eight characters.

System databases and system tablespaces

In DB2 ULWO, an instance can have several databases. Each database is one closed and independent unit containing its own logs, catalog and database configuration files. You cannot perform queries that would involve the tables of two different databases (unless using Relational Connect or DataJoiner®, which are not described in this document). The catalog (SYSCATSPACE), temporary space (TEMPSPACE1) and the user space (USERSPACE1) are all tablespaces. These are created automatically when you issue a CREATE DATABASE command.

In DB2 S/390 a DB2 subsystem can contain several databases. Databases in a subsystem interact with each other. In fact, as you can see from Figure 3, the catalog itself (DSNDB06) is a database. The work file database (DSNDB07) would correspond to the temporary space used in DB2 ULWO. The default database (DSNDB04) is used to store objects created without explicitly indicating the database they would belong to. DSNDB04 would correspond to USERSPACE1 in DB2 ULWO. The catalog, directory, and other system structures are created once at DB2 subsystem installation time, not like in DB2 ULWO where a catalog and other system structures are created for every CREATE DATABASE that is executed.

The DB2 ULWO catalog tables use the schema SYSIBM. In addition, DB2 ULWO provides read-only catalog views, which use schema SYSCAT, and updatable catalog views which use schema SYSSTATS. DB2 S/390 catalog tables use the qualifier of SYSIBM.

Some of the DB2 S/390 catalog tables are updatable, like the tables that are part of the communications database (CDB), and some of the columns that hold statistics information about the data are also updatable.

Under the DB2 S/390 structure, you can perform SQL operations using tables from different databases. For example, say you have table TS56692.testtbl in database MYDB1 and table DSN8710.emp in the default database DSNDB04. Then, you can execute the following query:

```
SELECT    B.name, B.salary
FROM      TS56692.testtbl A, DSN8710.emp  B
WHERE     A.level = B.edlevel
```

This would have not been allowed in DB2 ULWO because the two tables are in two different databases.

Connecting to a database vs. connecting to a subsystem

In DB2 ULWO, you *attach* to an instance to perform some administrative operations, and you *connect* to a database to perform database operations. In DB2 S/390 you *connect* to a subsystem, and perform both administrative and database operations. Thus, a DB2 ULWO client connecting to DB2 S/390 (through DB2 Connect™ software and DDF) will not connect to a specific database, but to the entire DB2 S/390 subsystem. In DB2 ULWO, you must connect to a specific database, which has its own catalog. For DB2 S/390, a database does not have its own

catalog; there is only one catalog for the entire DB2 subsystem. Therefore, when you want to access a specific DB2 S/390 database, you actually have to connect to the entire DB2 subsystem.

The directory

DB2 ULWO has a database directory, node directory, and dcs directory whose contents can be reviewed with the list db directory, list node directory, and list dcs directory, respectively. These directories contain connectivity information to other systems, and perform a similar function to DB2 S/390's Communication Database (CDB). Even though the term “directory” is used, this should not be confused whatsoever with the term directory in DB2 S/390. The DB2 S/390 directory is an important piece containing vital information in internal format and not directly useable to end users. There is no similar concept in DB2 ULWO.

Active and archive logs

DB2 ULWO uses active and archive logs for recovery purposes. It keeps track of its logs using the file SQLOGCTL.LFH, which is stored in the same directory as the database files (<instance name>.NODE0000.SQL0000x). Version 7 allows dual logging capability. DB2 S/390 also has the same concepts for active and archive logs. The BSDS dataset is used to keep track of its logs. Dual logging has been available in DB2 S/390 for many versions of the product.

Bufferpools

DB2 ULWO uses bufferpools to improve the performance of a database. In DB2 ULWO, the CREATE BUFFERPOOL command can be used to create a new bufferpool. In DB2 S/390, there are predefined bufferpools, most of them starting with a size of zero. In order to “create” a bufferpool, you have to use the ALTER BUFFERPOOL command and set a size greater than zero. In DB2 ULWO, the page size is entered as part of the CREATE TABLESPACE statement. A bufferpool with the correct page size needs to be created before creating the tablespace that uses this page size; in DB2 S/390, there is no parameter in the CREATE TABLESPACE statement that indicates the page size to be used; however, by specifying the bufferpool to be used, you will determine the page size.

DB2 ULWO uses Extended Storage to provide a second level of caching; DB2 S/390 can use hiperpools.

Configuration parameters

DB2 ULWO has parameters at the instance level (database manager configuration) as well as at the database level (database configuration). Changes to instance-level parameters require that DB2 ULWO is stopped and started. Changes to database-level parameters require that all connections are terminated before the changes take place on the next connections. In DB2 S/390, these parameters are often called “zparms” (for the default name of the parameter module, which is DSNZPARM). There is only one set of parameters that would affect the entire DB2 subsystem and its databases. The job DSNTIJUZ is used to specify the desired values for these parameters. When run, this job will assemble and link-edit the DSNZPARM module as well as the application program's default module DSNHDECP. The assembled zparm module can be specified when starting DB2. If it is not specified, the module with name DSNZPARM will be

used. Prior to Version 7, changes to zparms required DB2 S/390 to be recycled (stopped and started) to load the new parameter module into memory. With V7, this is still the case for some parameters but not for all. The new SET SYSPARM command allows you to load a new parameter module without recycling DB2.

The governor vs. the Resource Limit Facility

DB2 ULWO uses the governor to monitor and limit the activity of applications against a given database. A configuration file with rules is provided and the db2gov command is used to start the governor. Similarly, DB2 S/390 uses the Resource Limit Facility (RLF). This facility is started with the START RLIMIT command, and the rules are stored in database DSNRLST.

Temporary tablespaces

DB2 ULWO has two types of temporary tablespaces: system and user. You must always have a system temporary tablespace available, because this is the work area for the database manager to perform operations like join or sort. User temporary tablespaces, on the other hand, are used to store declared global temporary tables. These tables are not persistent; they only “live” for a given connection, or while the application that declared them is running. Similarly, DB2 S/390 provides two types of temporary space. The work file database (DSNDB07 in a non-data sharing environment) corresponds to DB2 ULWO's system temporary tablespace. DB2 S/390's TEMP database corresponds to DB2 ULWO's user temporary tablespace. The TEMP database is used also for server-side scrollable cursors, so client applications using this type of cursors may get an error if such database has not been created ahead of time. The concept of “global temporary table” is the same in these platforms.

4.2 Data structures

4.2.1 Databases

A database includes a collection of tables, their associated indexes, and the tablespaces in which they reside. Databases are treated as single units that can be started and stopped independently of each other.

4.2.2 Storage groups

Storage groups consist of a set of volumes on disk that hold datasets in which tables and indexes are actually stored. The default storage group SYSDEFLT is created after installing DB2. All volumes of a given storage group must have the same device type, but parts of a single database can be stored in different storage groups.

4.2.3 Tablespaces

Tablespaces consist of one or more VSAM LDS datasets. They are used to store tables. The page size of a tablespace is determined by the associated bufferpool. There are four types of tablespaces:

- **Simple**
Can contain more than one table. The rows of different tables are not kept separate (unlike segmented tablespaces).
- **Segmented**
Divides the available space into groups of pages called *segments*. Each segment is the same size. A segment contains rows from only one table.
- **Partitioned**
Can only contain *one* table. This type of tablespace divides the available space into separate units of storage called *partitions*. Each partition resides on a separate physical dataset. You assign the number of partitions (from one to 254) and you can assign partitions independently to different storage groups.
- **Large object (LOB)**
Holds large object data such as graphics, video, or very large text strings. A LOB tablespace is always associated with the tablespace that contains the logical LOB column values. The tablespace that contains the table with the LOB columns is called, in this context, the *base tablespace*.

4.2.4 Tables

All data in a DB2 database is presented in *tables* -- collections of rows all having the same columns. A table that holds persistent user data is a *base table*. A table that stores data temporarily is a global temporary table.

4.2.5 Indexes

An index is an ordered set of pointers to the data in a DB2 table. The index is stored separately from the table.

4.2.6 Views

A view is an alternate way of representing data that exists in one or more tables. A view can include all or some of the columns from one or more base tables.

4.2.7 Aliases

This is a pointer to another table, which can be on the same DB2 subsystem or on another DB2 subsystem. Aliases are not dropped if the table they are pointing to is dropped.

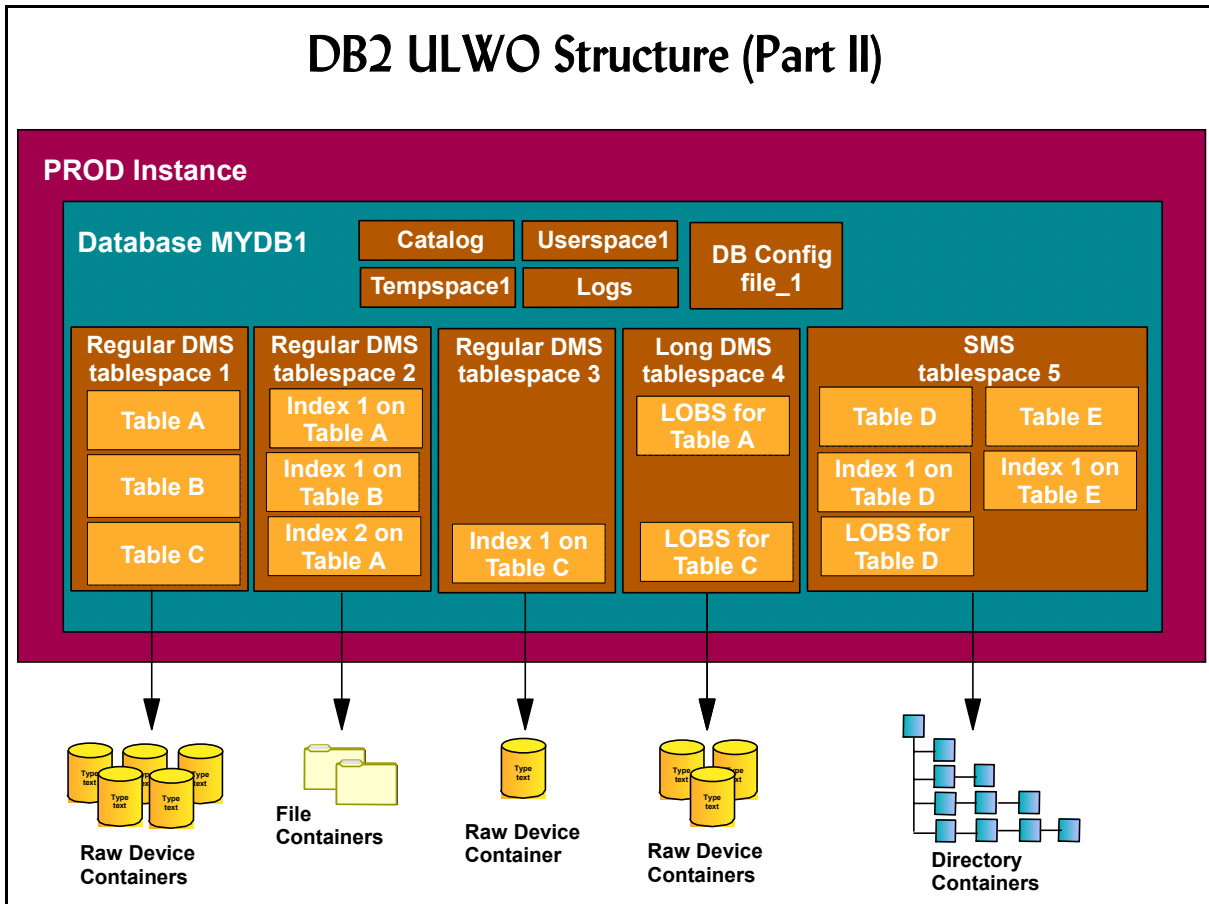
4.2.8 Synonyms

Synonyms are similar to aliases, but they can only refer to a table in the same subsystem. If the table is dropped, so is the synonym.

4.3 Comparing DB2 S/390 and DB2 ULWO data structures

Figures 4 and 5 below show the DB2 ULWO data structure and the DB2 S/390 data structure respectively.

Figure 4



The concept of a “database” in DB2 ULWO and DB2 S/390

A “database” in DB2 ULWO is an independent unit containing tablespaces, tables, indexes and “system” information (that is, catalog, logs, database configuration file). Clients must connect to a database before performing any database operation against them. A database can be deactivated when connections are idle for some time in order to improve performance by avoiding the overhead incurred on a first database connection.

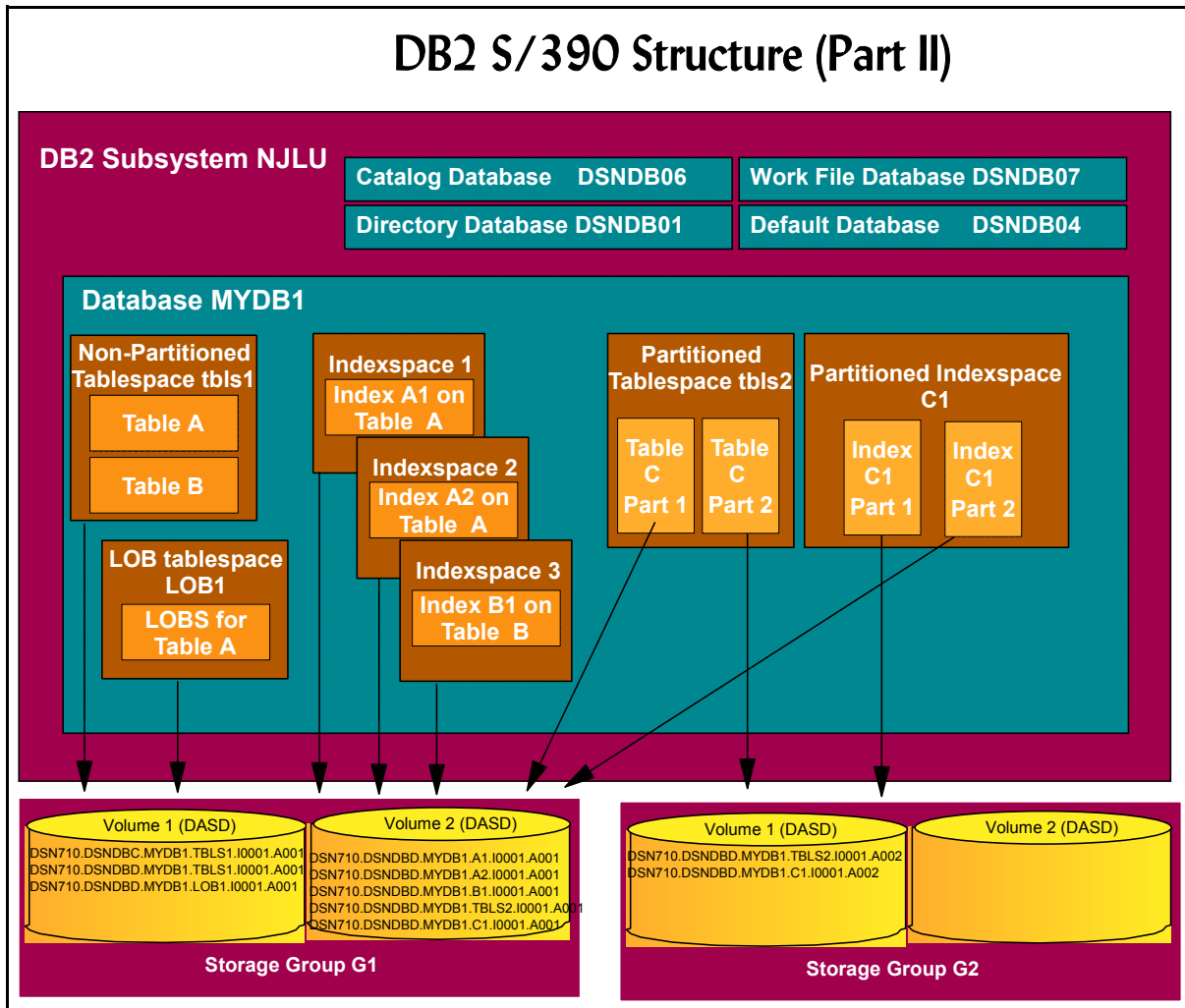
In DB2 S/390, a database is also an independent unit containing tablespaces, tables and indexes; however, “system” information is not included. The catalog, logs and configuration parameters are kept at the DB2 subsystem level, not at the database level. Since a DB2 S/390 database does not contain system information, you can think of it as another logical layer grouping tablespaces. Stopping a database implies stopping all the tablespaces it contains. The -STOP DB command and -START DB command can be used to stop and start a database, respectively.

Even though a DB2 S/390 database is an independent unit, objects from different databases can interact with each other; therefore, a table in one database cannot have the exact same full name (qualifier.tablename) as a table in another database. The same would apply to other objects. If the same full object names were allowed, DB2 S/390 would not be able to distinguish between the two when performing SQL operations using tables from different databases.

In DB2 ULWO, objects from different databases cannot interact with each other; therefore, a table in one database can have exactly the same full name as a table in another database. The same would apply to other objects.

Based on the above descriptions, it could be said that a DB2 S/390 database is most similar to a DB2 ULWO database without the “system” information.

Figure 5



DB2 ULWO containers vs. DB2 S/390 storage groups

A *container* in DB2 ULWO is a physical object that is used to store data. There are three types of containers:

- Directory
- Raw device
- File

A container is associated to a tablespace.

In DB2 S/390, a *storage group* serves a similar purpose as a container in that it is used to store data. A DB2 storage group, however, consists of a collection of physical devices (DASD volumes) managed by DB2. A DB2 storage group is associated to a tablespace. Note that a DB2 storage group is not the same as a SMS storage group. The latter is not covered in this document.

Containers and DB2 storage groups are both physical and are associated with tablespaces. In DB2 ULWO you can specify the containers associated to your tablespace “individually.” In DB2 S/390, you associate a tablespace to a DB2 storage group containing several DASD volumes; you cannot specify an individual volume (unless you create DB2 storage groups that contain only one volume).

Based on the above descriptions, it could be said that a DB2 ULWO raw device container is most similar to a DB2 storage group consisting of one DASD volume.

DB2-managed vs user-defined datasets

In DB2 ULWO, when a tablespace is created, the underlying physical objects (files) are created automatically for you when the objects are used. In DB2 S/390, you have the choice to have these underlying objects (datasets) automatically created by DB2 (“DB2-managed” datasets), or manually created by you (“user-defined” datasets).

By using DB2 S/390 storage groups, your datasets will be “DB2-managed.” If using “user-defined” datasets, you would need to allocate manually each dataset as needed (following DB2 dataset naming conventions) before executing a CREATE TABLESPACE statement. This latter method is not commonly used today, as it requires more manual administration. DB2 S/390 DB2-managed datasets are most similar to the way things work in DB2 ULWO.

The concept of a “tablespace” in DB2 ULWO and DB2 S/390

In DB2 ULWO, a tablespace is a *logical* interface to tables (logical) and containers (physical). It allows associating tables to a specific container, so that DB2 administrators can determine the best place to store data.

When you create a tablespace, you can indicate the containers associated with this tablespace, once created; however, there is no physical object that would map to this tablespace. For example, if you execute this statement:

```
CREATE TABLESPACE TBLSA MANAGED BY SYSTEM USING ('C:\TEMP')
```

Inside container 'c:\temp', files containing table data (.DAT) and index data (.INX) will be created; however, there is no physical object that would represent a tablespace.

In DB2 S/390 a tablespace is a *physical* interface to tables (logical) and Storage Groups (physical). Each tablespace can be mapped directly to one or more physical datasets. You can see this mapping by reviewing the dataset names DB2 uses; they include the tablespace name. Below is the structure used in a DB2 S/390 dataset name:

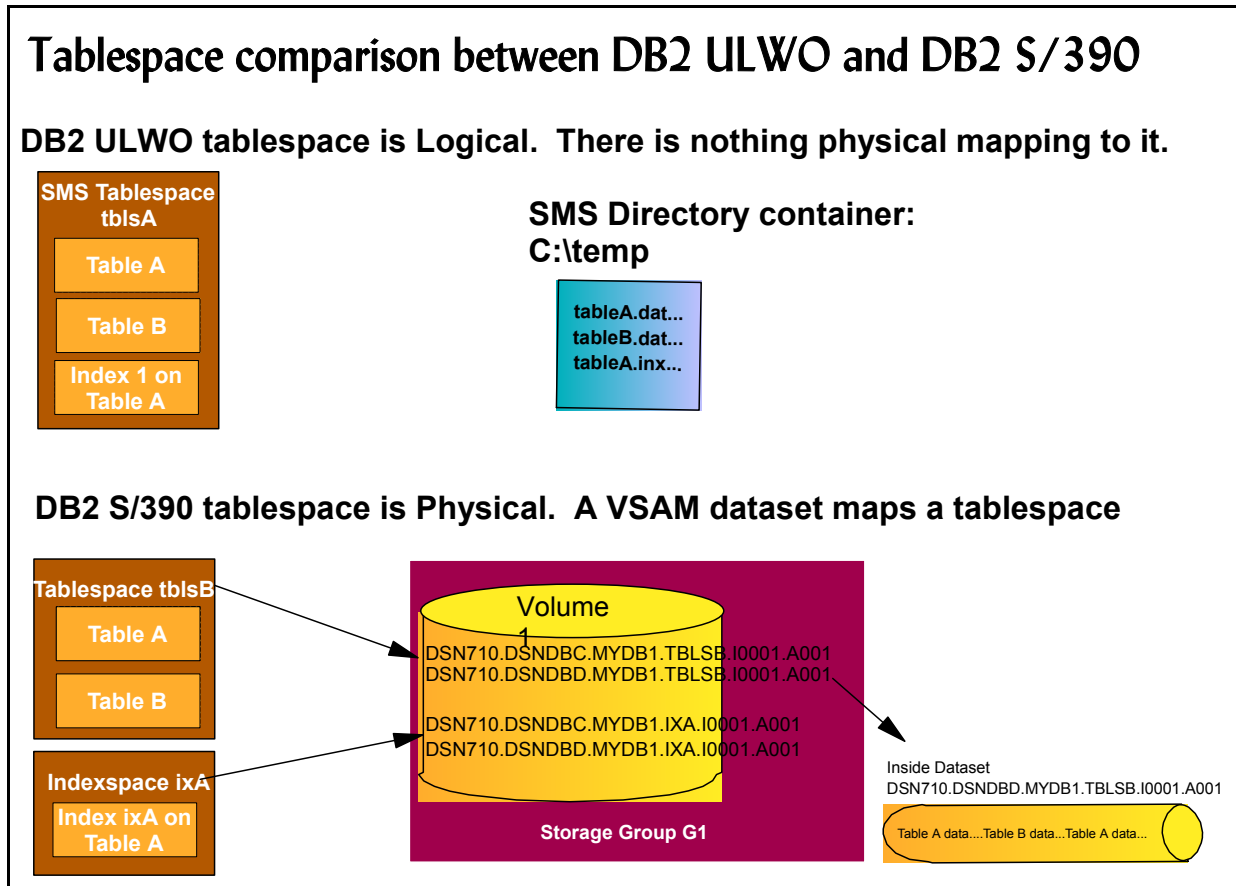
```
catname.DSDBx.dbname.tsname.I0001.Annn
```

tsname corresponds to the tablespace name or index name. Since the tablespace name is used as a qualifier of the dataset name, and since MVS dataset names only allow a maximum of eight characters per qualifier, then a tablespace name can only use eight characters.

Within each of these datasets is where the table or index information is stored, as opposed to DB2 ULWO, where there was no physical representation of a tablespace, and internal table and index files were stored inside the container.

Figure 6 shows the difference between DB2 ULWO tablespaces and DB2 S/390 tablespaces.

Figure 6



Tablespace classification

DB2 ULWO supports two types of tablespaces, SMS (system-managed) and DMS (database-managed). SMS-managed tablespaces are managed by the operating system's file system. They can only use directories as containers. Space is allocated dynamically by the file system. DMS-managed tablespaces are handled by the database system. They can only use files or raw devices as containers, and space is allocated 'manually' by the user.

As we can see, DB2 ULWO classifies tablespaces by the way they are managed (SMS, DMS), and also based on their use (regular, temporary, long). DB2 S/390 classifies tablespaces by the way the data is internally organized (simple, segmented, partitioned, LOBs). These types of tablespaces have no equivalence in DB2 ULWO other than the LOB tablespace.

The DB2 S/390 concept of a “partitioned tablespace” is not quite the same as a DB2 ULWO Enterprise Extended Edition (EEE) tablespace. A partitioned tablespace can only store one (large) table, and the partitions are created in the same machine. DB2 ULWO EEE uses several “nodes” (machines) to store part of tables across the nodes.

DB2 ULWO DMS tablespaces using file containers may be most similar to a DB2 S/390 tablespace. In a DB2 ULWO DMS tablespace using file containers, space needs to be manually specified to indicate how large the file container should be. In a DB2 S/390 tablespace, you specify a primary quantity for your underlying datasets (PRIQTY) as well as a secondary quantity (SECQTY).

Large objects

Both, DB2 ULWO and DB2 S/390 support large objects (LOBs); however, there are differences in the way LOBs are created. For DB2 ULWO, the LOBs can reside in the same tablespace as the table data or, if using DMS tablespaces, the LOBs can be stored in a separate LONG DMS tablespace. LOB columns can be declared in the CREATE TABLE statement as NOT LOGGED (LOBs modifications will not be logged). There is also a COMPACT option that can be specified to avoid allocating extra disk space for possible future appends. In DB2 S/390, depending on the value of special register CURRENT RULES, creating LOB columns may involve more steps:

- **If CURRENT RULES = 'DB2'**
 - A LOB tablespace *must* be created. The NO LOG option of the CREATE TABLESPACE can be used as equivalent to DB2 ULWO NOT LOGGED option in its CREATE TABLE statement
 - The table where the LOB(s) are defined must have a ROWID column defined.
 - An auxiliary table must be created
 - An index to the auxiliary table must be created.

Examples

In DB2 ULWO:

```
CREATE TABLE STRUCTURE_HEAD (  
  SH_PMGROUP          VARCHAR(5)  NOT NULL,  
  SH_INFO             CLOB(1G)   NOT LOGGED,  
  SH_CHANGE_TIME     TIMESTAMP  
);
```

Or, if a LONG DMS tablespace 'LOBTBLS1' was created, the CLOB could be put there with this statement:

```
CREATE TABLE STRUCTURE_HEAD (  
  SH_PMGROUP          VARCHAR(5)  NOT NULL,  
  SH_INFO             CLOB(1G)   NOT LOGGED,  
  SH_CHANGE_TIME     TIMESTAMP  
) IN DATATBLS1  
  LONG IN LOBTBLS1;
```

In DB2 S/390, however, the following DDL is required:

```
CREATE LOB TABLESPACE LOBTBLS1
  IN DSNDB04
  LOG NO;
CREATE TABLE STRUCTURE_HEAD (
  SH_PMGROUP          VARCHAR(5)  NOT NULL,
  SH_INFO             CLOB(1G) ,
  SH_CHANGE_TIME      TIMESTAMP,
  SH_ROWID            ROWID GENERATED ALWAYS NOT NULL
);

-- CREATE AUXILIARY TABLE IN LOB TABLESPACE
CREATE AUX TABLE STRUCTURE_HEAD_AUX
  IN DSNDB04.LOBTBLS1
  STORES STRUCTURE_HEAD
  COLUMN SH_INFO;

-- INDEX REQUIRED ON AUXILIARY TABLE
CREATE UNIQUE INDEX STRUCTURE_HEAD_IX
  ON STRUCTURE_HEAD_AUX;
```

- **If CURRENT RULES = 'STD'**

It is much easier to create LOBs when CURRENT RULES = 'STD' because DB2 will create implicitly the LOB tablespace, auxiliary table, and index to the auxiliary table. The ROWID column, however, must be added to the table definition. In terms of performance, it is preferred to use CURRENT RULES = 'DB2' before connecting to the database; thus, you can opt to create the DDL using CURRENT RULES = 'STD', and then switch the value afterwards to 'DB2'.

Tables, views and indexes

The concept of tables and views is the same in DB2 ULWO as in DB2 S/390. With respect to indexes, in DB2 ULWO, they could be stored in the same tablespace as the table data. Since a tablespace is associated to a container; files for both, table data and indexes would be “mixed” within the container. DB2 ULWO supports only 'Type 1' indexes. Support for 'Type 2' indexes is planned for future versions.

In DB2 S/390 the term *indexspace* is used to refer to tablespaces used by indexes. An indexspace is automatically created when an index is created; this means that underlying physical datasets are created for the index. As mentioned previously, in DB2 S/390, tablespaces as well as indexspaces can be mapped to physical datasets. Inside these datasets is where table and index information is kept in internal format. In DB2 ULWO, there is no physical object for a tablespace, but the table and index information are stored in files directly in the container. Type 1 and Type 2 indexes have been supported in DB2 S/390 for a couple of versions. Type 1 indexes, however, have been phased out and in Version 7 they are no longer supported.

Alias and synonyms

In DB2 ULWO, aliases and synonyms are exactly the same. Thus, the CREATE ALIAS and the CREATE SYNONYM statements are equivalent. In DB2 S/390 as seen previously, an alias has some differences with a synonym.

4.4 Schema

A schema is a collection of named objects. The objects that a schema can contain include distinct types, functions, stored procedures, and triggers. An object is assigned to a schema when it is created.

Schemas extend the concept of *qualifiers* for tables, views, indexes and aliases to enable the qualifiers for distinct types, functions, stored procedures and triggers to be called schema names.

When a distinct type, function, stored procedure, or trigger is created, it is given a qualified two-part name. The first part is the schema name (or the qualifier), which is either implicitly or explicitly specified. The default schema is the authorization ID of the owner of the plan or package. The second part is the name of the object.

You can create a schema with the schema processor by using the CREATE SCHEMA statement. To process the CREATE SCHEMA statement, you must use the schema processor (DSNHSP), by running a job based on the sample JCL provided in member DSNTEJ1S of the SDSNSAMP library. The result of processing a schema definition is identical to the result of executing the SQL statements without a schema definition.

Outside of the schema processor, the order of statements is important. They must be arranged so that all referenced objects have been previously created. This restriction is relaxed when the statements are processed by the schema processor if the object table is created within the same CREATE SCHEMA. The requirement that all referenced objects have been previously created is not checked until all of the statements have been processed. For example, within the context of the schema processor, you can define a constraint that references a table that does not exist yet or GRANT an authorization on a table that does not exist yet.

The schema processor sets the current SQLID to the value of the schema authorization ID before executing any of the statements in the schema definition. Thus, in the following example:

```
CREATE SCHEMA AUTHORIZATION CHONG
CREATE TABLE TEST
  (TESTNO CHAR(4), RESULT CHAR(4), TESTTYPE CHAR(3))
CREATE TABLE STAFF
  (EMPNUM CHAR(3) NOT NULL, EMPNAME CHAR(20), GRADE DECIMAL(4), CITY CHAR(15))
CREATE VIEW SMITH.MYVIEW AS SELECT * FROM STAFF WHERE GRADE >= 12
GRANT INSERT ON TEST TO PUBLIC
GRANT ALL PRIVILEGES ON STAFF TO PUBLIC
```

The fully qualified objects created will be: CHONG.TEST, CHONG.STAFF, SMITH.MYVIEW. For this example, it is assumed that authorization ID 'CHONG' has either a secondary ID of 'SMITH' or an authority of 'SYSADM', which allows the 'CHONG' ID to create a view with an explicit qualifier of 'SMITH'.

The concept of “schema” in DB2 ULWO corresponds to “qualifier” in DB2 S/390 for tables, views, indexes and aliases; and to “schema” for distinct types, functions, stored procedures, and triggers. When using the DB2 S/390 schema processor however, the authorization ID used in the CREATE SCHEMA statement becomes the current SQL ID; therefore, it is also used as the default qualifier for the mentioned objects above.

Examples

In DB2 ULWO we use:

<schema name>.<object name>

where <object name> can be tables, views, stored procedures, triggers, etc.

In DB2 S/390 we use:

- <schema name>.<object name>

where <object name> can only be distinct types, functions, stored procedures, and triggers.

- <qualifier>.<object name>

where <object name> can be tables, views, indexes, aliases.

As in DB2 ULWO, in DB2 S/390 you can create an object with an *explicit* schema name/qualifier or you can create one implicitly. When created explicitly, the creator of the object has to have the correct authority to use the schema name/qualifier; when created implicitly, the schema name/qualifier to be used is the current SQL ID.

4.5 Data types

We will not explain in detail the data types supported with DB2 S/390. The two figures below show the data types supported with DB2 ULWO and DB2 S/390, respectively.

Figure 7

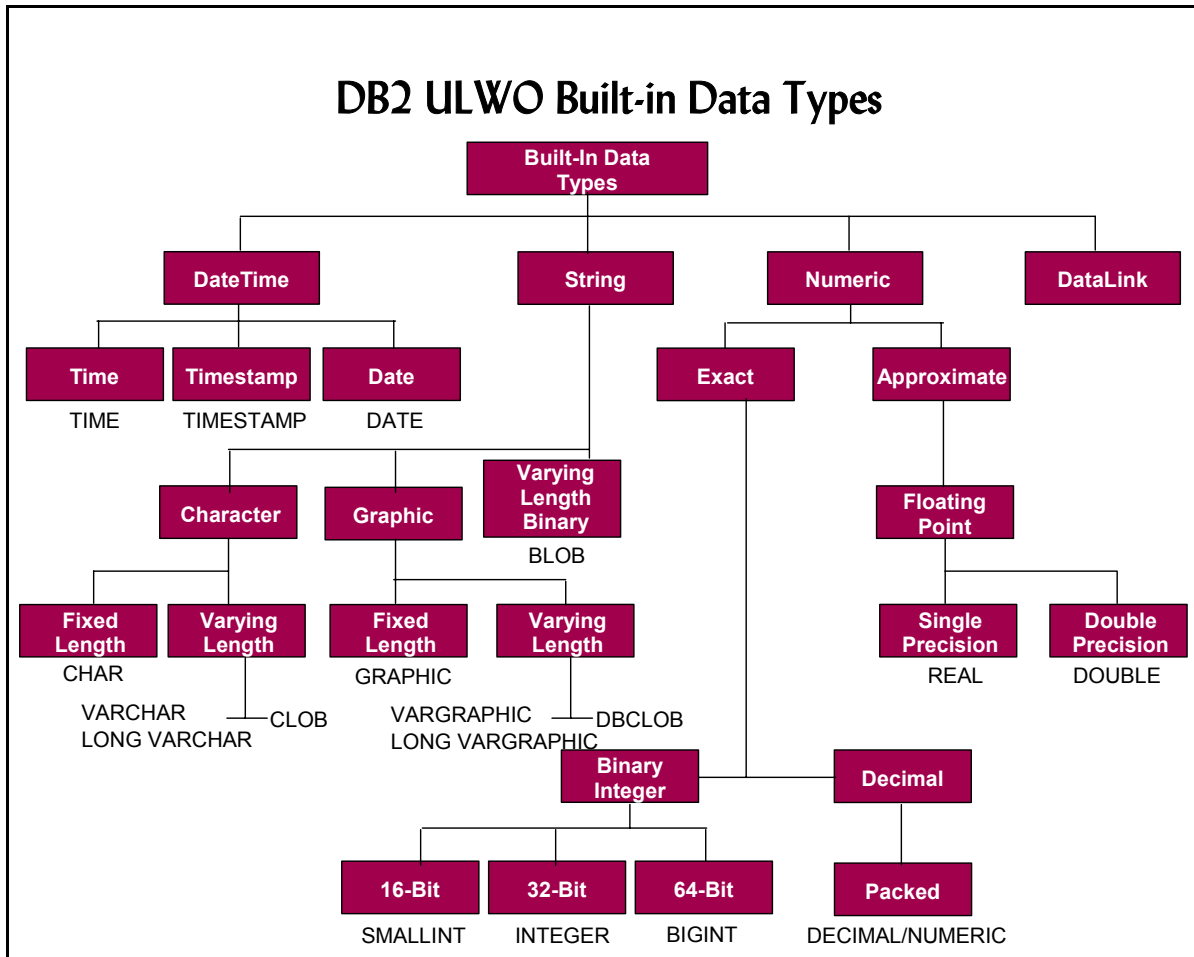
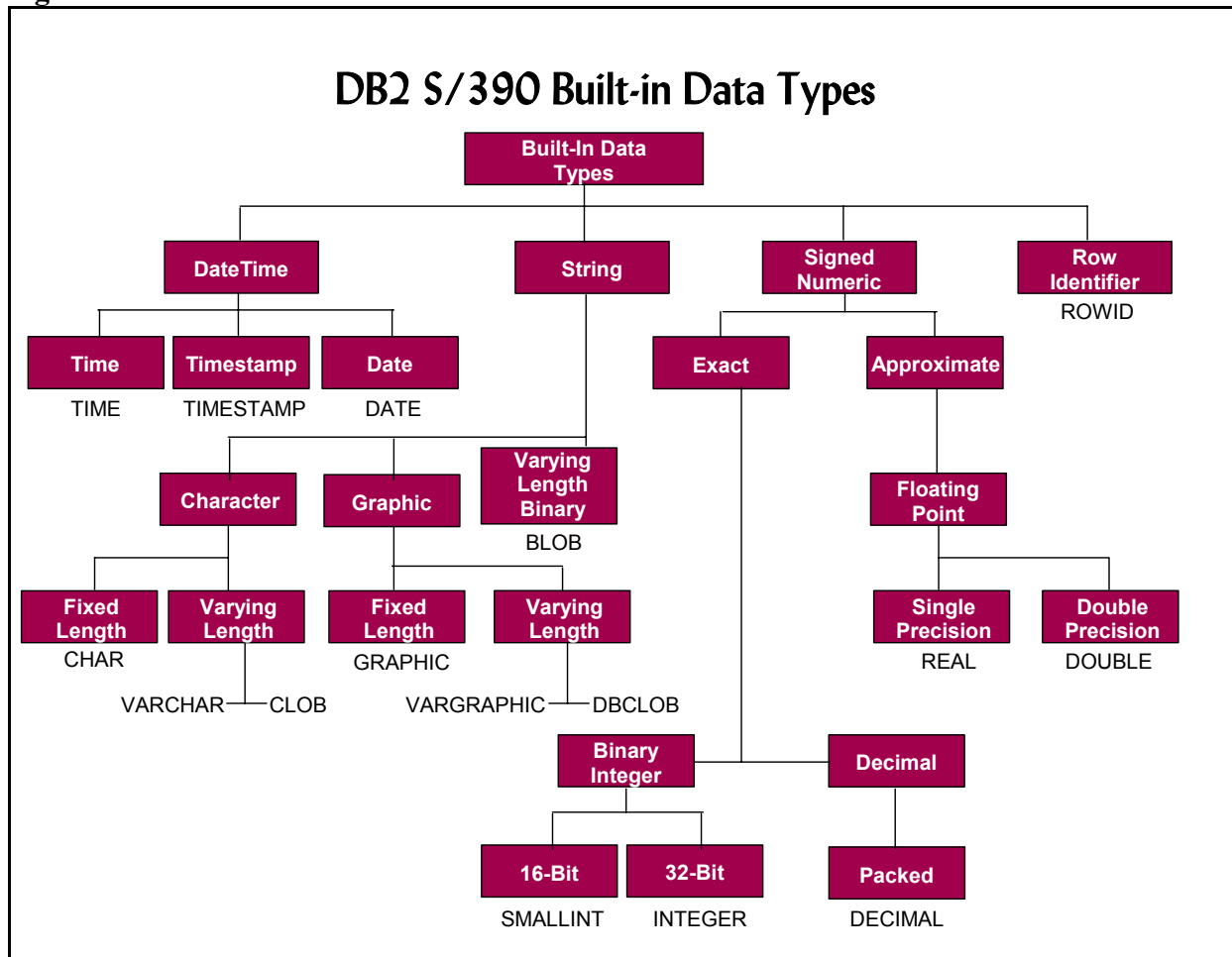


Figure 8



4.5.1 Data type comparison chart

The following tables map DB2 S/390 data types with DB2 ULWO data types.

Table 4. Mapping String Data Types from DB2 S/390 to DB2 ULWO

DB2 S/390 Data Type	Notes	DB2 ULWO Data Type	Notes
CHAR(n)	1 ≤ n ≤ 255	CHAR(n)	If n ≤ 254
VARCHAR(n)	n ≤ 255 ¹	VARCHAR(n)	n ≤ 32672
VARCHAR(n)	n ≤ 4056 for 4k page ²	VARCHAR(n)	n ≤ 32672
VARCHAR(n)	n ≤ 8138 for 8k page ²	VARCHAR(n)	n ≤ 32672
VARCHAR(n)	n ≤ 16330 for 16k page ²	VARCHAR(n)	n ≤ 32672
VARCHAR(n)	n ≤ 32714 for 32k page ²	LONG VARCHAR(n)	If 32672 < n ≤ 32700
CLOB(n)	1 ≤ n < 2GB ¹	CLOB(n)	n ≤ 2GB
GRAPHIC	1 ≤ n ≤ 127	GRAPHIC	1 ≤ n ≤ 127 DBCS

DB2 S/390 Data Type	Notes	DB2 ULWO Data Type	Notes
	DBCS characters		characters
VARGRAPHIC	1 <= n <= 32704 DBCS characters	VARGRAPHIC	1 <= n <= 16336 DBCS characters
VARGRAPHIC	1 <= n <= 32704 DBCS characters	LONG VARGRAPHIC	16336 < n <= 16350 DBCS characters
DBCLOB	If 32672 < n <= 2GB	DBCLOB	If 32672 < n <= 2GB
BLOB	n <= 2GB	BLOB(n)	n <= 2GB
CHAR(n) FOR BIT DATA	If n <= 255	CHAR(n) FOR BIT DATA	If n <= 254
VARCHAR(n) FOR BIT DATA	If n <= 32672	VARCHAR(n) FOR BIT DATA	If n <= 32672
VARCHAR(n) FOR BIT DATA	If 32672 < n <= 32700	LONG VARCHAR(n) FOR BIT DATA	If 32672 < n <= 32700

¹ Any CLOB and varchar (n) where n > 255 is considered a long string column and it may have some SQL statement restrictions.

² Subtract 10 bytes if using EDITPROC = YES

Table 5. Mapping Numeric Data Types from DB2 S/390 to DB2 ULWO

DB2 S/390 Data Type	Notes	DB2 ULWO Data Type	Notes
SMALLINT	-32768 to 32767	SMALLINT	-32768 to 32767
INTEGER	-2147483648 to 2147483647	INTEGER	-2147483648 to 2147483647
DECIMAL/NUMERIC (p,s)	-10 ³¹ to 10 ³¹ -1 (p+s <= 31)	DECIMAL(p,s)	-10 ³¹ to 10 ³¹ -1 (p+s <= 31)
REAL, FLOAT	1 <= p <= 21	REAL, FLOAT(p) 0<p<25	0, from -3.402E+38 to -1.175E-37, from 1.175E-37 to 3.402E+38
DOUBLE, FLOAT	22 <= p <= 53	DOUBLE, FLOAT(p) 24<p<54	0, from -1.79769E+308 to -2.225E-307, from 2.225E-307 to 1.79769E+308
ROWID		VARCHAR(40) FOR BIT DATA	ROWID is not supported in DB2 ULWO, however if data from DB2 S/390 is being loaded into DB2 ULWO, you can define a column in DB2 ULWO as varchar(40) for bit data to hold this data
DECIMAL(19,0)	BIGINT is not currently supported in DB2 S/390. DECIMAL(19,0) is the closest match	BIGINT	-9223372036854775808 to 9223372036854775807

Table 6. Mapping Datetime Data Types from DB2 S/390 to DB2 ULWO

DB2 S/390 Data Type	Notes	DB2 ULWO Data Type	Notes
DATE	MM-DD-YYYY ¹	DATE	MM-DD-YYYY ²
TIME	HH-MM-SS ¹	TIME	HH-MM-SS ²
TIMESTAMP	YYYY-MM-DD-HH -MM-SS-NNNNNN	TIMESTAMP	YYYY-MM-DD-HH-M M-SS-NNNNNN

¹ This representation can vary depending on DSNHDECP setting at installation time.

² This representation can vary and is dependent on the country code.

4.5.2 User-defined data types

User-defined distinct types allow a user to extend the data types that DB2 understands in a database. DB2 ULWO supports distinct types in a similar way as DB2 S/390. In addition, DB2 ULWO also supports user-defined reference types (to define a type hierarchy), and user-defined structured data types to support structured type columns. These are not currently available in DB2 S/390.

4.6 IDENTITY columns

Identity columns are supported exactly the same for both DB2 ULWO and DB2 S/390.

4.7 Sequence objects

Sequence objects are not currently supported in DB2 S/390, but are planned to be supported in the near future. IDENTITY columns can be used instead. DB2 ULWO does support sequence objects.

4.8 Special registers

DB2 S/390 as well as DB2 ULWO have special registers that can be used in SQL statements. In DB2 ULWO, the VALUES statement can be used to test the current value of a special register. The following example would return the value of the CURRENT TIMESTAMP special register:

```
db2 values (CURRENT TIMESTAMP)
```

In DB2 S/390, the VALUES statement is not currently supported. Instead, one can execute the following SQL (which can also be executed in DB2 ULWO):

```
SELECT CURRENT TIMESTAMP FROM SYSIBM.SYSDUMMY1
```

Note:

VALUES INTO as part of a program does work in DB2 S/390. For example:

```
EXEC SQL VALUES(CURRENT APPLICATION ENCODING SCHEME) INTO :HV;
```

Refer to the DB2 S/390 manuals for more detail about these special registers.

DB2 ULWO has special registers of its own. The table below compares these for both DB2 ULWO and DB2 S/390.

Table 7 - DB2 S/390 vs DB2 ULWO Special Registers

DB2 S/390 Special Register	DB2 ULWO Analogy
CURRENT APPLICATION ENCODING SCHEME	N/A
CURRENT DEGREE	CURRENT DEGREE
CURRENT LOCALE LC_CTYPE	N/A
CURRENT MEMBER	N/A
CURRENT OPTIMIZATION HINT	N/A
CURRENT PACKAGESET	N/A
CURRENT PATH	CURRENT FUNCTION PATH
CURRENT PRECISION	N/A
CURRENT RULES	N/A
CURRENT SERVER	CURRENT SERVER
CURRENT SQLID	CURRENT SQLID CURRENT SCHEMA
USER	USER
CURRENT DATE	CURRENT DATE
CURRENT TIME	CURRENT TIME
CURRENT TIMESTAMP	CURRENT TIMESTAMP

4.9 Unicode support

Unicode support is the same for both DB2 ULWO and DB2 S/390. Both support the UCS-2/UTF-8 standard. However, the CREATE DATABASE command used to indicate this support is different:

```
CREATE DATABASE <dbname> USING CODESET UTF-8 TERRITORY US (in DB2 ULWO)
CREATE DATABASE <dbname> CCSID UNICODE (in DB2 S/390)
```

4.10 Unique constraints

The concept of unique constraints is the same for both DB2 ULWO and DB2 S/390. Unique constraints can be defined in the CREATE TABLE statement or the ALTER TABLE statement using the PRIMARY KEY clause or the UNIQUE clause.

In DB2 ULWO, when a unique constraint is defined, the database manager will automatically create a unique index if needed. In DB2 S/390, when a unique constraint is defined, the table is marked as unavailable until you *explicitly* create an index for the unique or primary key constraint. Only if these constraints are processed by the schema processor will DB2 implicitly create all the necessary indexes.

4.11 Referential integrity

Referential integrity is supported in the same way for DB2 ULWO as it is for DB2 S/390. Insert, delete and update rules are basically the same.

4.12 Check constraints

Check constraints are supported similarly in DB2 ULWO and DB2 S/390. If you use the ALTER TABLE statement to add a check constraint, your table may be placed in check pending state. To reset this state, you can run the SET INTEGRITY command in DB2 ULWO, or execute the CHECK DATA utility in DB2 S/390.

4.13 Comparing SQL statements

Most SQL statements supported in these platforms are the same; differences will be found for statements that refer to a particular architecture difference. For example, DB2 S/390 supports different types of tablespaces; thus the CREATE TABLESPACE syntax will be different in DB2 S/390 from that in DB2 ULWO .

For a handy, single source of SQL that is compatible across the DB2 Family, see *SQL Reference for Cross-Platform Development*, available at <http://www7b.software.ibm.com/dmdd/library/techarticle/0206sqlref/0206sqlref.html>

4.14 Summary

The table below summarizes and compares the system structure concepts described in this chapter.

Table 8 - DB2 S/390 and DB2 ULWO System Structure Comparison

DB2 S/390 Concept	DB2 ULWO Analogy
DB2 address spaces started when the command <code>-start db2</code> is performed from an MVS console	DB2 services (In Windows), and DB2 processes (In UNIX) started when the command <code>db2start</code> is executed
DDF handles remote communications. To start DDF execute the command <code>-start ddf</code> from the DB2I command panel	DB2 ULWO agents handle remote communications. Eg: In UNIX we have <code>db2tpcm</code> , <code>db2snacm</code> , <code>db2tcpdm</code>
DB2 S/390 uses IRLM to handle locks.	DB2 ULWO does not externalize processes that handle locks other than <code>db2dlock</code> for deadlock detection.
In the mainframe you are allowed to have all types of environments in an LPAR: <ul style="list-style-type: none"> - Several DB2 subsystems with the same version and maintenance level, thus running the same shared code. - Several DB2 subsystems with the same version but at different maintenance level (different sets of DB2 code) - Several DB2 subsystems with different versions (different sets of DB2 code) 	In Windows, only one version at a given fixpack level of DB2 can be installed. In UNIX, different versions of DB2 can be installed, but only one fixpack level is allowed per version.
A command prefix is required for MVS to know to which DB2 subsystem should a command be applied to in the case where the command is executed from a place where access to the different DB2 subsystems is allowed. Otherwise, if you can only connect to one subsystem at a time, use the prefix '-' for commands.	Direct commands to a specific instance by setting the value of the DB2INSTANCE variable (<code>set DB2INSTANCE=<instance name></code>) or "attaching" to the instance (<code>attach to <nodename></code>)

DB2 S/390 Concept	DB2 ULWO Analogy
DB2 S/390 subsystem	DB2 ULWO instance
Catalog database (DSNDB06)	SYSCATSPACE tablespace
Directory database (DSNDB01)	N/A
Communications Database (CDB), part of the catalog.	Database directory, node directory, DCS directory
Active and archive Logs concept	Similar concept as in DB2 S/390
Dual logging supported	Dual logging supported
Bootstrap dataset (BSDS)	SQLOGCTL.LFH
Predefined bufferpools are “created” with -ALTER BUFFERPOOL.	Bufferpools are created with CREATE BUFFERPOOL.
Hiperpools	Extended Storage (ESTORE)
Resource Limit Facility (DSNRLST)	- The DB2 Governor (db2gov) - Query Patroller
Work file database (DSNDB07)	TEMPSPACE1 tablespace (system temporary tablespace)
TEMP database, for global temporary tables.	User temporary tablespace, for global temporary tables
Default database (DSNDB04)	USERSPACE1 tablespace
Objects from different databases can interact with each other; therefore, a table in one database cannot have exactly the same full name as a table in another database. The same would apply to other objects.	Objects from different databases cannot interact with each other. A table in one database can have exactly the same full name as a table in another database. The same would apply to other objects.
Can execute queries involving tables of different databases.	Cannot execute queries involving tables of different databases.
Client connects to a DB2 subsystem, not to a particular database.	Client connects to a particular database.
DSNZPARM (SET SYSPARM command allows DSNZPARM module to be loaded in memory while DB2 is up, but for some parameters, a -stop db2, -start db2 is still required).	DBM CFG (db2stop, db2start required for new values to be in effect) and DB CFG (all connection need to be terminated for the new values to be in effect on next connection).

The table below summarizes and compares the data structure concepts described in this chapter.

Table 9 - DB2 S/390 and DB2 ULWO Data Structure Comparison

DB2 S/390 Concept	DB2 ULWO Analogy
DB2 S/390 database	A DB2 S/390 database is like a DB2 ULWO database without system information (logs, db cfg, catalog).
DB2 storage group	A DB2 S/390 storage group is similar to having a <i>group</i> of DB2 ULWO raw device containers.
Tablespace	A DB2 S/390 tablespace holds tables as a

DB2 S/390 Concept	DB2 ULWO Analogy
	<p>DB2 ULWO tablespace. In both cases, a tablespace is an interface between tables and the physical container/storage group. DB2 S/390 tablespaces map to a physical dataset, while DB2 ULWO tablespaces don't map to anything, because they are "logical." DB2 S/390 supports simple, segmented, partitioned and LOB tablespaces. DB2 ULWO tablespaces can be SMS or DMS.</p> <p>DB2 S/390 tablespaces are classified by how data is internally organized; DB2 ULWO tablespaces are classified by how they are managed.</p> <p>A DB2 S/390 tablespace may be most similar to a DB2 ULWO DMS tablespace with file containers.</p>
Partitioned tablespace	<p>The DB2 S/390 partitioned tablespace concept is not the same as DB2 ULWO EEE. A partitioned tablespace will divide one table into several partitions within the same machine. DB2 ULWO EEE partitions a table across several machines (nodes).</p>
Tables	Same concept as in DB2 S/390
Indexes	<p>Same concept as in DB2 S/390; however, DB2 S/390 only supports Type 2 indexes; Type 1 indexes have been phased out. DB2 ULWO only supports Type 1 indexes, with Type 2 indexes coming up in a future version.</p>
Indexspace	Tablespace for indexes
View	Same concept as in DB2 S/390.
Alias	Same concept as in DB2 S/390.
Synonym	<p>DB2 ULWO does not differentiate between an alias and a synonym. Both are different terms for the same concept, not as in DB2 S/390. Thus, there is no equivalence to a DB2 S/390 synonym in DB2 ULWO.</p>
Qualifier (for tables, views, indexes, aliases)	Schema
Schema (for distinct types, functions, stored procedures, triggers)	Schema

Chapter 5 - Controlling Data Access

Access to DB2 can be divided in two parts: Access to the DB2 subsystem and access within the DB2 subsystem.

5.1 DB2 subsystem access

Access to the DB2 subsystem is controlled outside of DB2. Typically RACF (Resource Access Control Facility), a.k.a OS/390 SecureWay Security Server or other similar third-party vendor software (e.g., TopSecret, ACF2), are used to control this access. Security softwares are also typically used to protect the underlying data (VSAM LDS datasets) in a DB2 subsystem.

5.1.1 Kerberos security support

DB2 S/390 supports the use of Kerberos security to authenticate remote users. RACF is required for this support to be available.

5.2 Access within the DB2 subsystem

Once you are allowed to access a DB2 subsystem, DB2 will control access to specific DB2 objects through the use of privileges and authorities.

5.2.1 Authorization IDs and privileges

Every process that connects or signs on to DB2 is represented by a set of one or more DB2 identifiers called authorization IDs. An authorization ID can be assigned to a process by user exit routines. Every process has exactly one ID called the *primary authorization ID*. All other IDs are *secondary authorization IDs*.

Furthermore, one ID (either primary or secondary) is designated as the *current SQL ID*. You can change the value of the SQL ID during your session with the SET CURRENT SQLID statement.

For example, a user may connect to a DB2 subsystem with a primary authorization ID of JOHN but as he connects, a user exit routine is executed which will assign this process a secondary ID of PROJECT. Then you can make your primary ID be your current SQL ID by executing the command:

```
SET CURRENT SQLID = 'JOHN' ;
```

Having a secondary ID allows a user to be part of a group for which privileges have been assigned. For this example, several primary authorization IDs can be mapped to the secondary ID of PROJECT. Rather than providing privileges individually to each primary authorization ID, a system administrator can now provide specific privileges to the PROJECT ID. The mapping of primary and secondary IDs is performed with an exit routine. Setting the current SQLID to the primary ID will give you access to the objects allowed for this ID only; likewise, setting the current SQLID to the secondary ID will give you access to the objects allowed for this secondary ID only.

The GRANT and REVOKE SQL statements are used to provide and reject privileges and authorities to users.

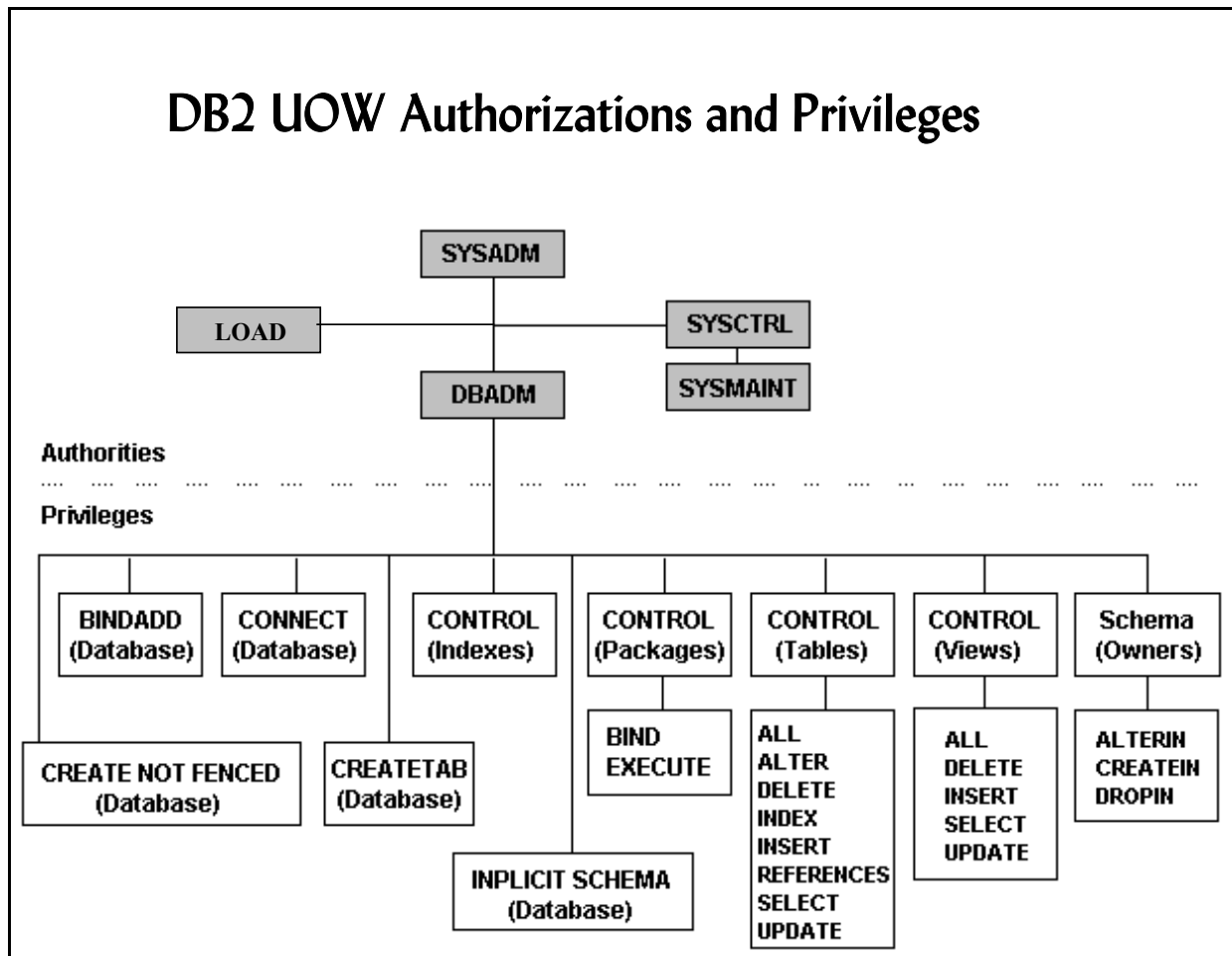
5.3 Comparing DB2 ULWO vs DB2 S/390 authorizations and privileges

In DB2 ULWO, users do not exist within the database, but rather are managed by the operating system. The operating system is also responsible for authentication. Within the database, privileges on specific database objects are assigned to those users provided by the operating system. It will be necessary to create in the operating system any users that your application requires to connect to the database and then provide database access to those users from within the database. DB2 ULWO authorization is defined by means of a system of authorities and privileges. Authority levels provide a method of grouping privileges and control. In DB2 ULWO, there are system-specific and database-specific authorities. System authorities are recorded by group membership and are stored in the database manager configuration file for a given instance (dbm cfg). These authorities are SYSADM, SYSCTRL and SYSMAINT. Each group name assigned to these authorities is managed by the operating system facility. Privileges are assigned within DB2 by using GRANT and REVOKE statements.

In DB2 S/390, users do not exist within the database either, but are managed through TSO/RACF (or other security software). As well, as indicated in the previous section, a similar concept is used with respect to authentication. Access to the DB2 subsystem itself is left to the operating system (or the Security software which is in some cases part of the operating system). Access to objects *within* DB2 is handled by DB2. Like DB2 ULWO, DB2 S/390 also uses authorities and privileges.

In DB2 ULWO, the System Administration authority (SYSADM) is the highest level of authority within the database manager, and controls all database objects. The database manager configuration parameter SYSADM_GROUP defines the group name with SYSADM authority for the database. In UNIX, the initial value is null and defaults to the primary group of the instance owner. In Windows, the value defaults to the Administrator Group. Following installation, a different group name can be assigned to SYSADM_GROUP within DB2 ULWO. Figure 9 shows the DB2 ULWO authorizations and privileges.

Figure 9

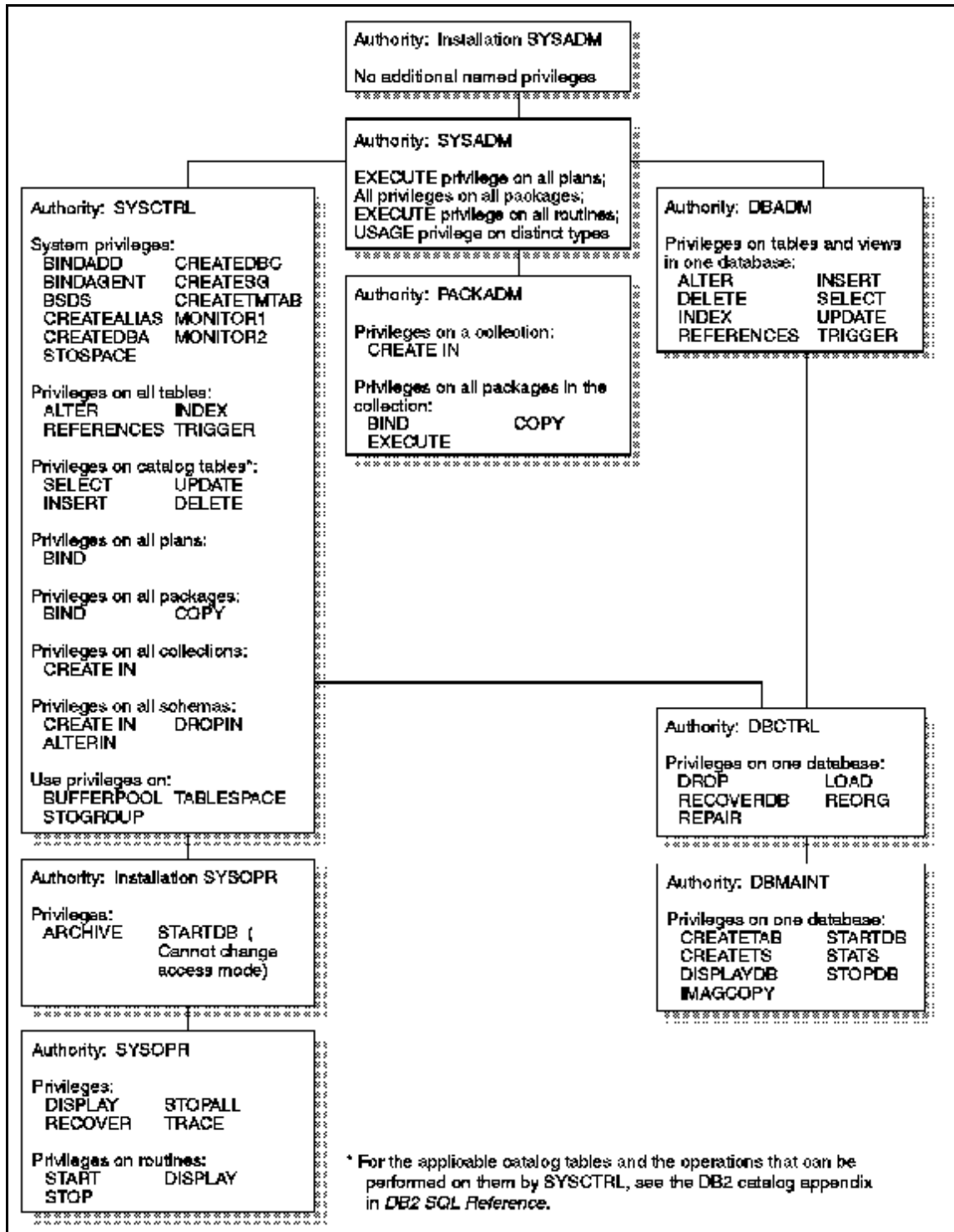


In DB2 S/390, there are two types of system administrators:

- The **Installation SYSADM** has the highest level of authority within DB2. This parameter is defined in the DSNZPARM module (through job DSNTIJUZ), and thus, nobody can REVOKE this authority within DB2.
- The **SYSADM** system administrator is the second-highest level of authority. As opposed to the Installation SYSADM, this authority is stored in the DB2 catalog, and it can be granted and revoked.

Other levels of authority are SYSCTRL, SYSOPR, Installation SYSOPR, PACKADM, DBMAINT, DBCTRL, DBADM. Figure 10 shows DB2 S/390 authorizations. For a list of DB2 S/390 privileges, refer to the *DB2 UDB for z/OS and OS/390 V7 Administration Guide*.

Figure 10 - DB2 S/390 Authorizations



In DB2 S/390, a secondary authorization ID can be used to map users to a shared ID (a group); privileges or authorities can then be given to this secondary ID. RACF also classifies users into groups, and DB2 can also grant or revoke authorizations and privileges to these groups. In DB2

ULWO, the secondary authorization ID concept is not used. UNIX and Windows operating systems also classify users into a group. Within DB2 ULWO, privileges and authorities can be granted or revoked to this operating system group.

In DB2 ULWO, any user can execute the statement SET CURRENT SQLID <schema name>. The schema name can be any string of up to 30 characters. The statement SET CURRENT SCHEMA <schema name> is equivalent. In DB2 S/390 only the SET CURRENT SQLID <id> statement can be used; SET CURRENT SCHEMA is not allowed. The ID can be either the primary authorization ID, or any secondary authorization ID of the current process. Only a SYSADM can specify any string of up to eight characters whether or not it is an authorization ID or associated with the process that is running.

5.4 Summary

In this chapter, we compared security between DB2 ULWO and DB2 S/390. For these platforms, access to DB2 is controlled by the operating system or the security software; and access within DB2 is controlled by DB2 itself through the use of authorizations and privileges. Though similar, these authorizations and privileges are not the same. The table below shows some analogies between these platforms with respect to security.

Table 10 - Comparing DB2 ULWO and DB2 S/390 Security

DB2 S/390 Concept	DB2 ULWO Analogy
Primary authorization ID	ID used to connect to a database or attach to an instance.
Secondary authorization ID	N/A
CURRENT SQLID	CURRENT SQLID or CURRENT SCHEMA
SET CURRENT SQLID	SET CURRENT SQLID SET CURRENT SCHEMA
Access to the DB2 subsystem is controlled by RACF or other security software (which may be part of the operating system).	Access to DB2 is controlled by the operating system.
Access within DB2 is controlled by DB2 through authorizations and privileges.	Access within DB2 is controlled by DB2 through authorizations and privileges.
Authorization/privileges to groups is allowed by using secondary authorization IDs, or a RACF group.	Authorization/privileges to groups is allowed by using a group created by the operating system.
Installation SYSADM is the highest authorization allowed.	SYSADM is the highest authorization allowed.
Other authorizations: SYSADM (different than installation SYSADM), SYSCTRL, DBADM, PACKADM, DBCTRL, DBMAINT, SYSOPR, Installation SYSOPR	Other authorizations: SYSCTRL, SYSMOINT, DBADM, LOAD.
Installation SYSADM and Installation SYSOPR are set in DSNZPARM. These authorizations cannot be granted or revoked	SYSADM authority group, SYSCTRL authority group, and SYSMOINT authority group are set in the dbm cfg. These

DB2 S/390 Concept	DB2 ULWO Analogy
with the GRANT and REVOKE statements respectively. All others can.	authorizations cannot be granted or revoked. All others can.
Kerberos security is supported for remote users. RACF is required.	Kerberos security is supported only for clients and servers running Windows 2000.

Part III. DB2 S/390 Administration

Chapter 6 - DB2 S/390 Utilities and Maintaining Data

6.1 DB2 S/390 Utilities at a glance

The following two tables list the utilities available with DB2 S/390. Some of these utilities are covered in more detail in subsequent sections of this document.

Table 11 - DB2 S/390 Operational Utilities Summary

Utility Name	Description
EXEC SQL	Executes non-select dynamic SQL statements in a DB2 utility job.
LOAD	Loads tables. This includes the DB2 family cross loader functionality.
REBUILD INDEX	Creates or rebuilds indexes.
RECOVER	Restores tablespaces and indexes from backups and applies log records.
REORG TABLESPACE	Physically reorders data.
REORG INDEX	Physically reorders indexes.
RUNSTATS	Collects space and access path information on tablespaces and indexes.
STOSPACE	Collects space usage information on DB2-managed tablespaces and indexes.
UNLOAD	Unloads tables, tablespaces and backup copies

Table 12 - DB2 S/390 Diagnostic and Recovery Utilities Summary

Utility Name	Description
CHECK DATA	Checks consistency among referentially related tables
CHECK INDEX	Checks consistency between data and indexes.
CHECK LOB	Checks consistency of LOB tablespaces.
COPY	Creates full or incremental backups of tablespaces or indexes.
COPYTOCOPY	Makes additional backups of tablespaces and indexes asynchronously.
MERGECOPY	Merges incremental backup copies to full backup copies.
MODIFY RECOVERY	Removes obsolete backup information from the DB2 catalog.
MODIFY STATISTICS	Removes obsolete historical information from the DB2 catalog.
REBUILD INDEX	Creates or rebuilds indexes.
RECOVER	Restores tablespaces and indexes from backups and applies log records.

6.2 Loading data

The SQL INSERT statement can be used to move or copy data from one subsystem to another, or from one table to another within the same subsystem; however, for better performance in the case of large amounts of data, the LOAD utility should be used.

The LOAD utility will load the data of one or more tables of a tablespace. The target table(s) to be loaded must exist prior to executing this utility. LOAD will also build or extend any indexes defined on them. All integrity checking is performed (referential integrity, check integrity).

In DB2 ULWO, a LOAD utility is also available. This utility is also used for performance reasons when loading large amounts of data. DB2 ULWO's LOAD utility uses UNIX/Linux/Windows files of format ASCII, delimited ASCII or IXF as input, and the target is a table. The LOAD utility is used on a per table basis; however, the db2move utility with the LOAD option can be used to invoke the LOAD utility against several tables of a database. The LOAD utility in DB2 ULWO will rebuild or extend indexes defined on the table to be loaded, and will check for primary key or unique key constraints; however, as opposed to DB2 S/390's LOAD utility, it will not check for any other type of constraints including referential constraints or check constraints.

As in DB2 S/390, the table to be loaded must exist before the LOAD is executed.

6.2.1 The Cross Loader

In the past, loading data from a DB2 S/390 to a DB2 ULWO database and vice versa could only be performed through the DB2 ULWO Export and Import utilities. DB2 Connect software had to be used to establish the connection between these databases. Because Export and Import are executing DML SELECTS and INSERTS under the covers, this was the only method supported for these platforms. DB2 S/390 does not have an Export or Import utility.

Example

Assuming that connectivity has been set up with a DB2 S/390 subsystem, you can execute the following from the DB2 ULWO CLP:

```
db2 connect to <DB2 S/390 subsystem> user <TSO id> using <TSO password>
db2 "export to d:\tempD\staff.ixf of ixf select * from staff390"
db2 connect to <DB2 ULWO database> user <UNIX/Win id> using <UNIX/Win
password>
db2 "import from d:\tempD\staff.ixf of ixf insert into staff"
```

The above example will export (unload) data from the staff390 table in the DB2 S/390 subsystem to a UNIX/Linux/Windows file. The data will be later imported (loaded) from this file into the staff table on DB2 ULWO.

With DB2 S/390 V7, the LOAD utility was extended to load the output of any SELECT statement directly into a table on DB2 S/390. This new extension is known as the 'Cross Loader'. The Cross Loader can only be used when loading a DB2 S/390 table, so it can only work one way. If you would like to load a DB2 ULWO table with data from a DB2 S/390 database, you would still need to use the EXPORT and IMPORT utilities.

Using LOAD REPLACE to delete ALL records of a table

To quickly delete all the records of a table in DB2 S/390, a method of using the LOAD utility with the REPLACE option and an empty input file can be used. This same method can also be used with DB2 ULWO.

DB2 ULWO Load Query command

DB2 ULWO also has a command called LOAD QUERY, which is used to monitor the activity of the LOAD utility. In DB2 S/390 the -DISPLAY UTILITY command can be used for a similar purpose. Alternatively, the progress of the LOAD in DB2 S/390 can be monitored through SDSF under ISPF, by looking at the LOAD job.

6.3 Unloading data

There are different methods that can be used to unload data on DB2 S/390:

- **DSNTIAUL**

This is a sample program provided with DB2 that allows you to unload data into a mainframe sequential dataset. Figure 1 in section 1.4 “JCL (Job Control Language)” provides an example of how DSNTIAUL is invoked in a JCL job.

- **UNLOAD utility**

The UNLOAD utility unloads data rows from an entire tablespace, specific partitions, or individual tables to one or more sequential datasets. It can also just unload specific columns of a table by using a field specification list.

The output records written by UNLOAD are compatible as input to the LOAD utility. This provides the ability to reload the original table or different tables with the data from the UNLOAD.

In DB2 ULWO, the utilities used to unload data are EXPORT and db2move. The EXPORT utility, as mentioned earlier, uses SELECT statements to unload the data into UNIX/Linux/Windows files. This utility is used on a per table basis. The db2move utility with the EXPORT option invokes the EXPORT utility for several tables.

- **REORG utility**

The REORG utility will be explained in more detail in the next section; however, there is an option in this utility that allows you to unload data: UNLOAD EXTERNAL. With this option, the REORG utility will unload selected data and place it into a dataset that can then be loaded into a table with the LOAD utility. Another REORG option that allows you to delete data from a table is DISCARD.

6.4 Reorganizing data

The REORG TABLESPACE utility is used to reorganize data in tablespaces and indexes. Alternatively, if a REORG of only indexes is required, you should use the REORG INDEX option. By reorganizing data, DB2 is able to reclaim space from tables previously dropped, eliminate overflow pointers, allocate free space per current settings, etc.

There is an *offline* REORG, and an *online* REORG. An offline REORG implies that limited or no access is allowed during the execution of this utility. Using SHRLEVEL option set to NONE or starting the tablespace in UT (utility mode) would correspond to an offline REORG. An online REORG provides users with read-only or read-write access while the REORG is executing. The SHRLEVEL option set to REFERENCE corresponds to a read-only REORG, while the SHRLEVEL option set to CHANGE corresponds to a read-write REORG.

DB2 ULWO also has a REORG utility with a similar purpose. This REORG, however, is performed at the table level rather than the tablespace level. All indexes will also be reorganized. Currently online REORG for tables is not supported. Online REORG for indexes *is* supported and performed automatically for you; the MINPCTUSED option must be set to a value greater than zero when creating the index, however.

In addition, DB2 ULWO provides the REORGCHK utility to check the physical organization of tables and report if a REORG is required. The REORGCHK will obtain its information from the catalog; therefore, the catalog statistics have to be up to date.

6.5 Gathering statistics

DB2 S/390 uses the RUNSTATS utility to gather statistics about columns, tables, tablespaces and indexes. These statistics are stored in catalog tables and are used to provide information about the physical organization of the data and to provide information that the DB2 optimizer needs in order to select the best access path for executing SQL statements. Given that this type of information is constantly changing in a system, RUNSTATS should be run frequently.

The RUNSTATS utility can be executed on a tablespace and its indexes, or for each object independently, or even for a specific column. It will also run without any locking or interference to other process (i.e., it's an 'online' utility).

DB2 ULWO also has a RUNSTATS utility. This utility is run at the table level as opposed to the tablespace level.

For both DB2 ULWO and DB2 S/390, it is recommended to RUNSTATS the catalog. It is also recommended to perform a RUNSTATS after a REORG, and to perform a BIND/REBIND after a RUNSTATS. In DB2 ULWO, RUNSTATS acts as a read-write online utility or as a read-only online utility. This is determined by the option SHRLEVEL. When this option is set to CHANGE (default), RUNSTATS behaves as a read-write online utility. When this parameter is set to REFERENCE, other applications will have read-only access.

6.6 Checking data consistency

Three utilities are used to check the integrity of data and indexes. These utilities are often required to remove the restrictive check pending status. They are:

- **CHECK DATA**
This online utility checks tablespaces for violations of referential and table check constraints.
- **CHECK INDEX**
This online utility checks indexes for consistency with its data.

- **CHECK LOB**

This utility checks a LOB tablespace for structural defects in the LOB tablespace and any invalid LOB values.

The REPAIR utility with the SET TABLESPACE NOCHECKPEND option can also be used to reset CHECK PENDING status.

In DB2 ULWO, the SET INTEGRITY statement (formerly known as SET CONSTRAINTS) is used to check data consistency and to reset CHECK PENDING status. This statement is performed at the table level, not at the tablespace level, as it is in DB2 S/390.

6.7 Other utilities

- **MODIFY**

The SYSIBM.SYSCOPY catalog table is used to keep track of backup operations (image copies) as well as other operations performed against DB2 S/390 databases. Related entries are also stored in other system tables like SYSIBM.SYSLGRNX. These tables grow as operations are performed on the databases and space problems may occur if they are not cleaned up. The MODIFY online utility with the RECOVERY option can be used to delete outdated information on these tables.

DB2 ULWO uses the 'History file' (db2rhist.asc) to store similar information. This is a binary file located where the database is created. In order to view the file, you can execute the command LIST HISTORY. To clean up the file so that it doesn't grow too much, use the PRUNE HISTORY command.

- **REPAIR**

The REPAIR online utility repairs data. It also can be used to reset pending status. Users of this utility should be extremely careful.

DB2 ULWO provides the db2dart utility, which verifies that the architectural integrity of a database is correct. It mainly inspects the data, but does not repair it. Normally, customers would contact DB2 Service support when db2dart reports problems.

6.8 Running online utilities

The sections above described several DB2 S/390 online utilities. In order to execute them, a JCL job needs to be submitted. This JCL job can be generated using option 8, 'Utilities' in the DB2I primary option menu. Depending on the type of utility selected, different datasets will be required. After you have entered information into the DB2I panels, you will be taken to a screen where you can edit the JCL. Alternatively, you can save this JCL elsewhere and create your own library of JCL jobs for the different utilities. The JCL used will execute program DSNUTILB, and the input to this program will be the utility statement cards.

As an example, the following JCL job can be used to execute the RUNSTATS utility:

```

//RUNSRAUL JOB 6230, 'RUNSTATS', CLASS=C, MSGCLASS=X, REGION=0M,
//          NOTIFY=TS56692
// *
//UTIL EXEC PGM=DSNUTILB, PARM='NJLU, RUNSTEMP'
//STEPLIB DD DSN=SHARE.DSN710.PROD.SDSNLOAD, DISP=SHR
//SYSPRINT DD CYST=*
//SYSIN DD *
RUNSTATS TABLESPACE DSNDB04.EMP
          INDEX (ALL)
//

```

6.9 Standalone utilities (or offline utilities)

The following utilities are considered standalone or offline utilities, because DB2 does not need to be running when they are executed:

- **DSNJLOGF (Preformat Active Log)**
This utility preformats the datasets to be used for active logs so they can be used for writing. In DB2 ULWO this type of operation is left to the operating system.
- **DSNJU003 (Change Log Inventory)**
This utility is used to edit the BSDS (Bootstrap dataset).
In DB2 ULWO there is no equivalent utility available to users. The SQLOGCTL.LFH file is normally not manipulated.
- **DSNJU004 (Print Log Map)**
This utility is used to list the contents of the BSDS.
In DB2 ULWO there is no equivalent utility available to users. The SQLOGCTL.LFH file is normally not manipulated. DB2 service personnel may use internal utility db2pdlog.
- **DSN1CHKR**
This utility verifies the integrity of DB2 directory and catalog tablespaces. It will detect broken links, broken hash chains and orphans (records that are not part of any link or chain). DB2 ULWO uses db2dart to check the integrity of all tablespaces including the catalog and user tablespaces.
- **DSN1COMP**
This utility estimates the space savings to be achieved by DB2 data compression in tablespaces. You can compress data in a tablespace or partition by specifying COMPRESS YES on the CREATE TABLESPACE or ALTER TABLESPACE statements and then running the LOAD or REORG utility. In DB2 ULWO there is no compression capability within the product.
- **DSN1COPY**
This utility allows you to copy data and restore it under the same or different DB2 subsystem, in other words, it lets you move data between subsystems. This utility also lets you print hexadecimal dumps of DB2 datasets and databases, and can check the validity of data or index pages. DB2 ULWO uses EXPORT, IMPORT, LOAD, and db2move to

move data between databases or machines. The 'backup' command can also be used to copy an entire database which could be restored in a different machine. With respect to data and index page validity checking, DB2 ULWO uses the db2dart utility.

- **DSN1LOGP**

This utility formats the contents of the recovery log for display. DB2 ULWO does not have something similar to this utility.

- **DSN1PRNT**

This utility is useful when you want to identify the contents of a tablespace or index. It allows you to print:

- DB2 VSAM datasets that contain tablespaces or indexspaces (including dictionary pages for compressed data)
- Image copy datasets
- Sequential datasets that contain DB2 tablespaces or indexspaces.

DB2 ULWO does not currently have something similar to this utility.

- **DSN1SDMP**

This utility, under the direction of an IBM Support Center specialist lets you:

- Force dumps when selected DB2 trace events occur.
- Write DB2 trace records to a user-defined MVS dataset.

DB2 ULWO does not currently have something similar to this utility.

6.10 Resolving restrictive and advisory states

We discussed previously the check pending state. There are many other restrictive states that could be set for a DB2 object, and many different actions to take to reset these states. Refer to the *DB2 UDB for OS/390 Version 7 Certification Guide* for a chart with a description of these other states.

6.11 Summary

The following table shows DB2 S/390 utilities and their corresponding DB2 ULWO ones.

Table 13 - Comparing DB2 S/390 and DB2 ULWO Utilities

DB2 S/390 Utility	DB2 ULWO Analogy
LOAD (Can load one or more tables of a tablespace).	LOAD (Can only load one table at a time unless the db2move with the load option is used).
An extension of the LOAD utility called the cross loader allows using SQL statements and DRDA to be used as input to the LOAD. This is helpful when loading data from a DB2 ULWO server (or other DRDA server) to DB2 S/390.	There is no such capability in DB2 ULWO. Currently, EXPORT/IMPORT utilities are used to transfer data from DB2 S/390 to DB2 ULWO and vice versa.
-DISPLAY UTILITY	LOAD QUERY TABLE

DB2 S/390 Utility	DB2 ULWO Analogy
DSNTIAUL (SAMPLE PROGRAM)	EXPORT (Can only export one table at a time unless the db2move with the export option is used).
REORG UNLOAD EXTERNAL	EXPORT (Can only export one table at a time unless the db2move with the export option is used).
REORG UNLOAD DISCARD	EXPORT (Can only export one table at a time unless the db2move with the export option is used).
UNLOAD (Can unload one or more tables of a tablespace)	EXPORT (Can only export one table at a time unless the db2move with the export option is used).
REORG TABLESPACE	REORG TABLE
N/A	REORGCHK
RUNSTATS TABLESPACE	RUNSTATS TABLE
CHECK DATA	SET INTEGRITY (a.k.a., SET CONSTRAINTS)
CHECK INDEX	SET INTEGRITY (a.k.a., SET CONSTRAINTS)
CHECK LOB	SET INTEGRITY (a.k.a., SET CONSTRAINTS)
MODIFY	PRUNE HISTORY
REPAIR (Inspects and can also fix)	db2dart (Inspects only)
DSNJLOGF	N/A
DSNJU003	N/A
DSNJU004	db2pdlog (internal utility)
DSN1CHKR	db2dart
DSN1COMP	N/A, compression is not available in DB2 ULWO
DSN1COPY	EXPORT, IMPORT, LOAD, db2move for moving data. db2dart for checking page validity
DSN1LOGP	N/A
DSN1PRNT	N/A
DSN1SDMP	N/A
DB2PLI8 (internal utility)	db2look
DB2EXPL (internal utility, formats EXPLAIN table output in a way needed for DB2 Service)	N/A

Chapter 7 - Database Recovery

7.1 Database recovery concepts

There are different methods DB2 S/390 can use to recover data when there are failures. DB2 allows you to recover data to the current state or to the state at an earlier point in time. You can recover the following:

- tablespaces
- indexes
- partitions of a tablespace
- individual datasets
- pages within an error range
- individual pages

DB2 will save only data and not the layout of your structures. Also, if you drop your structures, then all recovery information is lost and recovery cannot be performed.

7.2 Logging

Changes made to the database are logged in active logs. The DML statements are recorded in the log as follows:

- INSERT: Entire after image of the record is logged called a *redo* record.
- DELETE: The before image of the record is recorded and called an *undo* record.
- UPDATE: Both. The before and after images (undo and redo record) are recorded.

LRSNs and RBAs are unique identifiers of each log. RBA (relative byte address) is used in non-data sharing environments. It is the offset of the record in the log from the beginning of the log. LRSNs (log record sequence number) is equivalent to a RBA but for data sharing environments. In this type of environment, all the members of the data sharing group have to coordinate their own separate logs, so LRSNs help track the sequence of events occurring over multiple members. LRSNs are based on timestamps.

Active logs are stored physically on log datasets. There can be up to 31 *predefined* log datasets available for active logs; however, you can specify a number lower than this on installation panel DSNTIPL. Active logs are written to the active log datasets as changes are made to the database. When a dataset becomes full, it will be offloaded into an archive log dataset, and the next active log dataset will be used to continue logging database activity. The offload process may prompt an operator to mount a tape or prepare a disk unit. The operator may not reply right away, but this will not affect DB2, as it will continue logging in the next active log dataset. This process will continue in a round-robin fashion; thus, when the last allocated active log dataset becomes full, DB2 will trigger the offload for this dataset, and will continue with the first active log dataset.

If the operator never replies to the offload processing prompt, DB2 may run out of its active logs. If all the active logs are full, DB2 will attempt an offload and will halt processing until the offload is completed. If the offload processing fails, then DB2 cannot continue work that requires writing to the log.

Archive logs are created *dynamically* and can be stored on disk or tape, but recovery will be faster if they are stored on disk. DB2 allows for many archive log datasets. Their retention period depends on your image copy frequency and how far you want to go back in time during a point-in-time recovery.

DB2 S/390 also provides **dual logging** capability to ensure that there are two copies of the active log datasets; in the event that one is lost, the other can be used for recovery. During a recovery, typically DB2 will start from an image copy (a backup) and then apply the logs. If the records needed to recover are not in the active logs any longer, DB2 will call for the appropriate archive log(s).

The **BSDS** (bootstrap dataset), as explained in a previous section, is like a table of contents that keeps track of log datasets and their contents. It also stores information about bufferpool attributes. The DB2 system will record all of the current active and archive log datasets in the BSDS. At DB2 installation time, by default DB2 will create two copies of the BSDS datasets. These datasets will be used when DB2 is started and will be updated during normal DB2 operations whenever an archive log is created. The BSDS can record up to 1000 archive log dataset records; when this number (or the one set at installation time) is reached, DB2 will continue recording in the BSDS from the beginning of the dataset, thus overwriting older records. Note that during the offloading process, when an archive log is created, a copy of the BSDS is also stored with the archive log dataset. This will be useful whenever the BSDS datasets are corrupted and you need to recover them.

SYSIBM.SYSLGRNX is a table in the DB2 directory that keeps track of update periods for tablespaces and recoverable indexes. Thus, if you are attempting to recover these objects, there will not be a need to go through all the log records. DB2 will review the SYSLGRNX table and determine which log datasets (or parts of the log) it requires and will speed up the recovery process because logs that contain no updates for the object being recovered are skipped.

7.3 Image copies

The **COPY utility** is used to obtain image copies, which are backups of your tablespaces or indexspaces. The more frequently you take image copies, the less the number of logs to be processed, and thus the faster the recovery can be. Note that after a drop and recreate of an object, all system retained recovery information is removed. Also, after you run utilities that recreate the data (like REORG, LOAD REPLACE without the LOG YES option), you have to run an image copy to establish a new base for recovery. A record in the SYSIBM.SYSCOPY catalog table will be inserted every time an image copy is performed.

Image copies can be taken as part of other utilities like LOAD or REORG. These are called inline copies, and it is faster to take them that way than it is to run a separate COPY utility afterwards.

New to version 7 is the **COPYTOCOPY utility**, which allows you to make a copy from an image copy.

Full and incremental image copies

A full image copy copies all pages of a tablespace whether it has changed or not (this is the default). An incremental image copy copies only the pages that have changed since the last image copy. It is common to see administrators use incremental copies for its speed and then run the **MERGECOPY utility** to merge the last full copy and the latest incremental copy into a new full copy.

Index image copies

It is not necessary to have index image copies, as these can be rebuilt from the tables (using the REBUILD utility); however, in some cases, it may be faster to recover from an image copy of an index, especially when the index has little or no change activity.

Image copies for the catalog and directory

Without a catalog or directory you will not have an operational DB2 subsystem. Thus, these system structures should be image copied frequently so that their recovery is fast.

7.4 Quiesce utility

To ensure that it is possible to recover to a point in time where everything is consistent and in synch for related objects, a point of consistency or a common QUIESCE point must be established. This can be obtained by executing the QUIESCE utility. The utility will wait for units of work to complete and will prevent new ones from beginning. The quiesce point will be recorded for each tablespace and indexspace in a recovery set, and the START_RBA of the quiesce point will be recorded in SYSIBM.SYSCOPY.

You should backup and quiesce objects together if they are related by Referential Integrity. To determine these relationships, you can use the **REPORT utility** with the TABLESPACESET option.

7.5 Recovery concepts

With image copies and logs, a DB2 user can recover different types of objects. We explain this procedure in this section.

7.5.1 Tablespace recovery

In order to recover a tablespace, the **RECOVER utility** is used. It will first perform a restore phase, where it will determine which full image copy is needed by looking at the rows in the SYSIBM.SYSCOPY catalog table. The appropriate full image copy will be taken and merged with any incremental copies that are found in the SYSIBM.SYSCOPY table replacing any updated pages in the full copy. The RECOVER utility will then perform a LogApply phase. In this phase any changes that are on the DB2 log that were made after the image copies were taken will be applied.

The REPORT RECOVERY utility can help plan for recovery as it provides information about image copies, active logs, and archive logs needed to recover a tablespace.

7.5.2 Indexspace recovery

Indexes can also be recovered if they were enabled for image copies in their DDL definition. Using image copies, an index is recovered rather than being rebuilt from the table using the REBUILD utility. There are only full image copies on indexes, and no incremental copies. The LogApply phase works the same way as for the tablespace recovery.

7.5.3 Partial object recovery

It is possible to recover part of an object. You can perform recoveries against:

- A partition
- A dataset of a nonpartitioned Index (NPI)
- A single page
- An error page range

A single table, however, cannot be independently recovered if there is more than one table in the tablespace.

7.5.4 Multiple object recovery

As you can back up multiple objects in the same job execution, it is also possible to recover multiple objects at the same time. The parallel option will allow the restore phase to be performed in parallel.

7.5.5 Point-in-time recovery

The RECOVER utility with the TOLOGPOINT <RBA/LRSN> parameter can be used to recover to any previous point in time. The RBA or LRSN provided should be a quiesce point that was established for the copy recover set. You can get this information from the SYSIBM.SYSCOPY table or the REPORT RECOVERY utility.

7.6 Backup and recovery of the DB2 catalog and directory

The DB2 catalog and directory is also comprised of tablespaces and indexspaces, and they can be subjected to failures (like media failures) like any other DB2 object. Thus, these objects may need to be recovered. There is a specific order to recover catalog datasets due to relationships and dependencies in the catalog.

7.7 Disaster recovery

This is another type of recovery that happens when an entire data center suffers a catastrophic event (fire, earthquake, terrorism). In order to prepare for such type of disasters, many steps need to be carefully planned and followed. We will only mention here the type of objects you would need to back up frequently and send to the disaster recovery site:

- Image copies of all user data
- Image copies of the catalog and directory
- Archive logs
- ICF (Integrated Catalog Facility) EXPORT and list

-BSDS lists

7.8 Tracker site recovery

The tracker site option allows for the creation of a separate DB2 subsystem (or data sharing group) that exists only for keeping shadow copies of the primary site's data. This is supported by transferring the BSDS and archive logs from the primary site to the tracker site, where LOGONLY recoveries are periodically run to keep the shadow data current. This method is advantageous for disaster recovery as it minimizes data loss and the time required to recreate a functional system at the recovery site.

7.9 Comparing DB2 ULWO with DB2 S/390 backup and recovery

In general, DB2 ULWO backup and recovery is simpler than DB2 S/390's. Many of the actions required or allowed in DB2 S/390's are automated or hidden in DB2 ULWO. For example, though the SQLOGCTL.LFH file is equivalent to DB2 S/390's BSDS, a DB2 ULWO user need not be concerned with this file (other than making sure it is not deleted). Adding new active log files doesn't require manual updates of SQLOGCTL.LFH. In DB2 S/390, adding active logs datasets require manual changes to the BSDS which can be performed by the DSNJU003 utility. In addition, the BSDS also keeps a DDF communication record (location name, TCP port, etc). You can use the DSNJU004 utility to view this information and DSNJU003 to make changes. Alternatively, installation panel DSNTIPR can be used to specify this DDF information.

DB2 ULWO, like DB2 S/390, uses similar concepts with respect to active logs and archive logs, but these are not quite the same. In DB2 ULWO there are two types of logging: circular logging and archival logging. Circular logging (default) does not use archive logs, and as its name implies, it will use the active logs in round-robin fashion, and will overwrite older logs. Because of this, with circular logging you are limited to crash recovery only; this means that you cannot restore the database to a given point in time. In DB2 S/390 there is no circular logging.

The second type of logging in DB2 ULWO is archival logging and this is similar to DB2 S/390's logging. In DB2 ULWO, either or both of these two database configuration parameters must be turned on in order to enable archival logging: LOGRETAIN and USEREXIT. In DB2 S/390, archival logging is the default.

DB2 ULWO has a database configuration parameter called LOGPRIMARY, which is used to specify how many primary log files are created when first connecting to the database (similar to DB2 S/390 in the sense that it has a specific number of predefined active log datasets that cannot exceed 31 in single logging mode). In addition, DB2 ULWO has another database configuration parameter called LOGSECOND, which will specify how many active log files should be *dynamically* created when there are not enough primary log files. Secondary active logs are not available in DB2 S/390.

In DB2 ULWO, when an active log becomes full, the next available active log will be used to continue logging. There is no offloading process that is initiated automatically when the log becomes full, unless userexits have been enabled.

In DB2 ULWO, an active log will be considered active until the transactions in that log file are no longer needed for crash recovery; that is, when the unit of work has committed or rolled back.

Only when this happens will the active log become an online archive log first (meaning it remains in the same directory or location as the active logs), and when these files are moved to another location or media, they become offline archive logs. In DB2 S/390, active logs are the ones that are stored in active log datasets, and when the offloading process takes place, the full active log becomes an archive log. This archive log, however, may potentially have uncommitted transactions.

In DB2 ULWO, when userexits are enabled, DB2 will trigger the userexit to copy an active log into an archival location or media as soon as it becomes full. This is similar to the DB2 S/390 offloading process. The original copy of the active log will be overwritten once it becomes an archive log. Based on the above descriptions, it could be said that DB2 ULWO archival logging with userexits enabled is most similar to DB2 S/390 logging. Dual logging is supported in DB2 ULWO as in DB2 S/390.

If there is a very long running transaction that does not commit, there is a possibility that DB2 ULWO has run out of active primary and secondary logs. So if for example a long transaction X is running and has used 95% of the active logs, if another transaction Y is started which uses the 5% left, then since this transaction was the first one to hit the 'active log full' condition, DB2 ULWO will rollback this transaction and report an error. In DB2 S/390, however, this situation would only happen when all the active *and* archive logs allowed are used. Thus, you can have transaction X filling up all the active logs, which are offloaded to an archive log and ready for reuse on the next pass of this same transaction.

DB2 ULWO uses the terms "backup" and "restore" while DB2 S/390 uses the terms "image copy/backup" and "recover." The BACKUP command in DB2 ULWO can back up an entire database or just specific tablespaces; DB2 S/390 works at the tablespace level, so image copies can only be performed at that level. For example, the DB2 ULWO command:

```
BACKUP DATABASE DB2CERT TO C:\DBBACKUP
```

will back up the entire database DB2CERT into directory C:\DBBACKUP. In DB2 S/390, you would have to image copy all the tablespaces of a database, and also the tablespaces of the catalog and directory, though using wildcards could make this process easier.

In this example:

```
BACKUP DATABASE DB2CERT TABLESPACE (SYSCATSPACE,FILETS) TO C:\DBBACKUP
```

would back up the catalog tablespace, SYSCATSPACE, and a user tablespace, FILETS, and would store the backup in C:\DBBACKUP.

DB2 ULWO also provides the ability to create incremental backups; this is a feature introduced in Version 7. DB2 S/390 can use incremental image copies.

In a similar way, the RESTORE command in DB2 ULWO is used to restore the database or tablespaces back to the way they were. The corresponding examples for the previous backup examples would be:

```
RESTORE DATABASE DB2CERT FROM C:\DBBACKUP  
RESTORE DATABASE DB2CERT TABLESPACE (SYSCATSPACE,FILETS) FROM C:\DBBACKUP
```

When restoring the database or tablespace, you can apply the logs to the backup if desired. Alternatively, you can use the option WITHOUT ROLLING FORWARD, which will prevent

this from happening. The ROLLFORWARD command can be later used to apply the logs. A timestamp can be specified with the ROLLFORWARD command, which would allow you to apply the logs to a specific point in time. This would be equivalent to DB2 S/390's "point-in-time recovery." While DB2 S/390 uses RBAs/LRSNs to recover to a point in time, in DB2 ULWO, you explicitly specify a timestamp.

DB2 ULWO also uses the QUIESCE TABLESPACES FOR TABLE command to create a point of consistency that can be used for a subsequent roll-forward recovery. The recovery history file can be used to find quiesce points and check whether they are past the minimum recovery time to determine a desirable time to stop the rollforward. This history file contains information about BACKUPS, RESTORES, ROLLFORWARDS, QUIESCEs, DROPs, etc. It is similar to SYSIBM.SYSCOPY in DB2 S/390. DB2 ULWO 's PRUNE HISTORY command can be used to clean up this history file in the same way that DB2 S/390's MODIFY RECOVERY command is used to clean up SYSIBM.SYSCOPY.

DB2 ULWO uses also LSNs (Log Sequence Number) which should be the equivalent to RBAs (Relative Byte Address)/LRSNs (Logical Sequence Number); however, LSNs are not externalized to normal DB2 ULWO users, while RBAs/LRSNs are needed in DB2 S/390 for point-in-time recovery.

With respect to database access, DB2 ULWO's BACKUPS can be performed for databases and tablespaces in online mode (using the ONLINE keyword) or offline mode (default). RESTORES can be performed for databases in offline mode only, while they can be performed in offline and online mode for tablespaces. ROLLFORWARDS for databases can be performed in offline mode ONLY, while they can be performed in offline and online mode for tablespaces, except for SYSCATSPACE (similar to RESTORE). In DB2 S/390, image copies and RECOVER can only be performed at the tablespace level, and both allow online and offline modes.

7.10 Summary

The following table summarizes the similarities and differences between DB2 ULWO's backup and restore and DB2 S/390's backup and recover concepts.

Table 14 - Comparing DB2 ULWO with DB2 S/390 Backup and Recovery

DB2 S/390 Concept	DB2 ULWO Analogy
Active logs	Active logs
Archive logs	Archive logs in DB2 S/390 may still contain transactions not committed or rolled back that may still be used for crash recovery. DB2 ULWO's archive logs contain only transactions that have completed (committed/rolled back).
Offloading process, which can be triggered when an active log is full.	Manual or userexit move of logs to another media or directory location.
BSDS	SQLOGCTL.LFH
RBA/LRSN	LSN
N/A	Circular logging (default)
Having logs on (default)	Archival logging with userexits enabled.

DB2 S/390 Concept	DB2 ULWO Analogy
Dual logging supported	Dual logging supported
Image copy (COPY utility) at the tablespace or indexspace level.	Backup utility at the database or tablespace level.
Image copy can be taken online or offline.	Backup utility can be taken online or offline.
Incremental Image copies	Incremental backups
RECOVER utility at the tablespace level	RESTORE utility at the database or tablespace level.
RECOVER can be run online or offline	RESTORE can be run online or offline for tablespaces, offline only for databases
RECOVER Utility includes a LOGAPPLY phase	RESTORE can include a LOGAPPLY phase, or can use the ROLLFORWARD command.
There is no ROLLFORWARD command.	ROLLFORWARD command can be run online or offline for tablespaces, offline only for databases.
REPORT RECOVERY utility	LIST HISTORY command
MODIFY RECOVERY utility	PRUNE HISTORY command
QUIESCE TABLESPACESET TABLESPACE <tablespace name>	QUIESCE TABLESPACES FOR TABLE <table name>

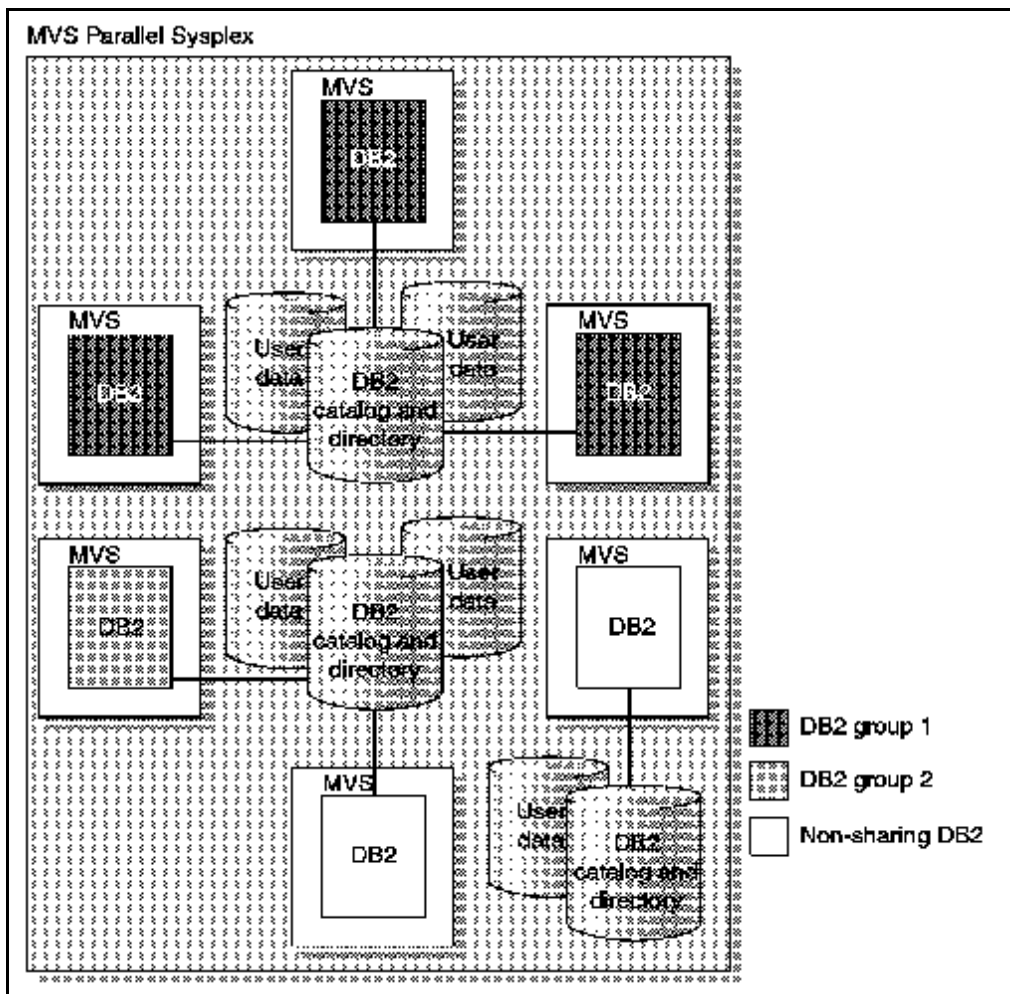
Chapter 8 - Data Sharing

8.1 Description

Data sharing allows an application to run on one or more DB2 subsystems in a Parallel Sysplex® environment. A parallel sysplex is a cluster of OS/390 systems that communicate and cooperate with each other. It consists of a **Coupling Facility** (specialized hardware, specialized high-speed links, adaptors, and so on), and a **Sysplex Timer®** (common time source across all of the systems in the cluster).

In a data-sharing environment, an application can read and write to the same data concurrently. In the past, this could only be done using DDF to access data on other subsystems. The subsystems that can share data must belong to a data-sharing group. The subsystems in the group are known as members. Shared DASD is required to share the MVS catalog, the DB2 catalog, DB2 directory, shared databases, etc. There is one catalog and one directory for the entire data sharing group. The following graph shows what a data-sharing system looks like.

Figure 11 - An Example of a Data-Sharing Environment



8.2 Data-sharing benefits

Data sharing provides many benefits, some of them are:

- Improves availability of data - if a member is down, other members are still available.
- Enables scalability - Processors can be added without disruption.
- Supports flexible configurations. You can have more than one DB2 data-sharing group on the same OS/390 Parallel Sysplex. For example, you might want one group for testing and another for production data.
- Leaves application interface unchanged. Your investment in people and skills is protected because existing SQL interfaces and attachments remain intact when sharing data. You can bind a package or plan on one DB2 subsystem and run that package or plan on any other DB2 subsystem in a data-sharing group.

8.3 DB2 ULWO Enterprise Extended Edition (EEE)

Data sharing as implemented in DB2 S/390 does not have an equivalent in DB2 ULWO. Some of the benefits it provides, like scalability, is provided through DB2 ULWO Enterprise Extended Edition (EEE).

Chapter 9 - Performance Monitoring and Tuning

9.1 EXPLAIN

The EXPLAIN tool allows you to analyze the access path chosen by the DB2 S/390 optimizer. You first need to create the PLAN_TABLE, DSN_STATEMNT_TABLE and DSN_FUNCTION_TABLE tables. The DDL for these tables can be found in the DB2 sample library SDSNSAMP, under the member DSNTESC. The last two tables mentioned above are optional, but if created, the optimizer may insert records in them too. The PLAN_TABLE can also be used as input to the optimizer in order to provide 'optimization hints'.

In order to populate the PLAN_TABLE, the EXPLAIN tool should be invoked. This is done in either one of two ways:

- Executing the EXPLAIN SQL statement in SPUFI or QMF.
- Binding with the option EXPLAIN(YES) and using the EXPLAIN SQL statement.

For example, assuming that no set of rows in the PLAN_TABLE has the value 23 for the QUERYNO column, we can do:

```
EXPLAIN PLAN SET QUERYNO = 23
FOR SELECT X.ACTNO, X.PROJNO
FROM DSN8710.EMPPROJECT X
WHERE X.EMPTIME > 0.5;
```

To retrieve and analyze the information stored in the PLAN_TABLE, query it using the queryno column as shown below:

```
SELECT * FROM PLAN_TABLE A, DSN_STATEMNT_TABLE B
WHERE A.QUERYNO = 23 and B.QUERYNO = 23
ORDER BY A.QBLOCKNO, A.PLANNO, A.MIXOPSEQ;
```

DB2 ULWO also has an EXPLAIN facility, and several EXPLAIN tables; however, the DDL for these tables are different from the ones for DB2 S/390. This means that each product has a version of Visual Explain, a workstation-based tool for graphically displaying access paths.

Visual Explain for S/390 comes as part of the DB2 Management Clients Package which is explained in more detail in a later section. In DB2 ULWO, optimization hints are not provided through the EXPLAIN tables; however, the catalog views with schema SYSSTAT can be modified to change the statistics and thus influence the optimizer that way.

9.2 Query parallelism

DB2 S/390 provides three types of query parallelism:

- **Query I/O parallelism.** This type of parallelism is used only when one of the other parallelism modes cannot be used. It allows for parallel fetch of pages into the bufferpool for a single query. This processing can significantly improve the performance of I/O-bound queries.
- **Query CP parallelism.** This type of parallelism reduces the elapsed time for a query by breaking it up into multiple smaller queries. These smaller queries run simultaneously on multiple processors accessing data in parallel.

- **Sysplex query parallelism.** This type of parallelism is only possible in a data-sharing environment. For processor-intensive queries, DB2 can split a large query across the different DB2 members of the data-sharing group.

DB2 S/390 will determine what type of query parallelism is used and when it will be used. However, in order to enable or disable it, the following actions should be performed:

Table 15 - Query Parallelism in DB2 S/390

To Enable DB2 S/390 Query Parallelism	To Disable DB2 S/390 Query Parallelism
For static SQL , specify DEGREE(ANY) on BIND or REBIND The default value is obtained from zparm CURRENT DEGREE	For static SQL , specify DEGREE(1) on BIND or REBIND The default value is obtained from zparm CURRENT DEGREE
For dynamic SQL : - Set the CURRENT DEGREE special register to 'ANY'. E.g., SET CURRENT DEGREE='ANY'; - Make sure RLST does not disable parallelism for the package, plan, authid.	For dynamic SQL : - Set the CURRENT DEGREE special register to '1' SET CURRENT DEGREE='1'; or - Use the RLST to disable parallelism for the package, plan or authid.
Use CURRENTDATA(NO) for DB2 to consider parallelism on ambiguous cursors.	Use CURRENTDATA(YES) to disable DB2 from considering parallelism on ambiguous cursors.
The virtual bufferpool parallel sequential threshold (VPPSEQT) value must be large enough to provide adequate bufferpool space for parallel processing.	Set VPPSEQT to 0.
Maximum degree of parallelism is determined by zparm parameter MAX DEGREE.	Setting MAX DEGREE to 1 will NOT disable parallelism. Follow any of the above methods to disable parallelism.

DB2 ULWO also supports query parallelism. It defines two types:

1. **Inter-query parallelism.** This simply means that DB2 ULWO allows multiple applications to query a database at the same time. DB2 S/390 also supports this type of parallelism.
2. **Intra-query parallelism.** This type of parallelism refers to the processing of parts of a single query at the same time. It can be classified in two ways:
 - **Intra-partition parallelism.** This type of parallelism will break up a query into multiple parts and it is best suited to take advantage of Symmetric Multiprocessors (SMP) systems. This type of parallelism is similar to DB2 S/390's Query CP parallelism.

- **Inter-partition parallelism.** This type of parallelism will break up a query into multiple parts across multiple partitions of a partitioned database on one machine or multiple machines. This type of parallelism is best suited to take advantage of the massively parallel processing (MPP) system; therefore, DB2 ULWO Enterprise Extended Edition needs to be installed to support this type of parallelism. It is closest in concept to DB2 S/390's Query Sysplex parallelism.

You can also use intra-partition parallelism and inter-partition parallelism at the same time. This combination provides two dimensions of parallelism, increasing the speed of queries even further.

Though I/O parallelism was not mentioned in the above classification, in DB2 ULWO this type of parallelism also happens, and it is determined mainly by the configuration of your hardware (CPU and disks) and some configuration parameters.

In order to enable intra partition parallelism in DB2 ULWO , the database manager configuration parameter INTRA_PARALLEL needs to be set to YES.

With respect to the DEGREE of the parallelism, there are several parameters that deal with it:

- **MAX_QUERYDEGREE:** Database manager configuration parameter that specifies the maximum degree of parallelism a query can use. A value of -1 (or ANY) means that DB2 will determine this value. If set to 1, parallelism will not take place; however, this is not the recommended approach to turn off parallelism as some memory will still be allocated for it. This is equivalent to DB2 S/390's zparm MAX DEGREE.
- **DFT_DEGREE:** Database configuration parameter that specifies the default degree to use. If set to 1 you will be disabling parallelism; however, this is not the recommended way to disable parallelism, but using the INTRA_PARALLEL parameter. This is equivalent to DB2 S/390's zparm CURRENT DEGREE.
- **CURRENT DEGREE:** Special register that sets the degree of parallelism for dynamic SQL and defaults to DFT_DEGREE. The statement SET CURRENT DEGREE is used. This is equivalent to DB2 S/390's CURRENT DEGREE register.
- **DEGREE:** Precompile or bind option that sets the degree of parallelism for static SQL. This is equivalent to DB2 S/390's DEGREE() bind option.
- **RUNTIME DEGREE (SET RUNTIME DEGREE statement):** Sets the degree of parallelism for running applications.
- **DB2DEGREE (CLI configuration file):** Sets the degree of parallelism for CLI applications.

9.3 System monitoring

In order to monitor a DB2 S/390 subsystem, the `-start trace` command is used to turn on traces needed to collect specific monitor information. This information can later be dumped into datasets that are used as input to performance monitor tools. For a fee, IBM can provide the IBM DB2 Performance Monitor (DB2 PM) tool. This tool is sold separately from DB2 and is used to generate performance reports and to monitor the system.

DB2 PM batch reports present the data you select in comprehensive reports or graphs containing system-wide and application-related information for both single DB2 subsystems and DB2 members of a data-sharing group.

Batch reports can be used to examine performance problems and trends over a period of time. Two batch reports typically used are the statistics report and the accounting report. The statistics report provides an excellent source of information of DB2 memory usage at the subsystem level. The Accounting report provides information about applications for specific time periods. There are also some more specific reports like the Lock Detail Analysis report for monitoring locks.

DB2 PM also comes with an Online Monitor that gives you a current snapshot view of a running DB2 subsystem. This monitor also comes with a GUI interface that has a similar look to the DB2 Control Center.

In DB2 ULWO switches need to be turned on in order to capture information about the system. This can be performed by turning on some database manager configuration parameters, or alternatively, by using the UPDATE MONITOR SWITCHES command, which would only affect its own session. Once the switches are on, the `get snapshot` command is used to report the information accumulated. This is known as snapshot monitoring.

For other type of problems like deadlocks, DB2 ULWO provides event monitoring. While snapshot monitoring takes a snapshot view of the current system, event monitoring will report problems based on an event that triggers it.

DB2 S/390's DB2 PM tool is similar to these two tools. DB2 S/390 has other new tools that provide further monitoring capabilities. These are mentioned in a later section of this document.

9.4 Summary

The table below compares and summarizes the topics covered in this chapter.

Table 16 - Comparing Performance Monitoring concepts in DB2 S/390 vs DB2 ULWO

DB2 S/390 Concept	DB2 ULWO Analogy
EXPLAIN tool	EXPLAIN tool
PLAN_TABLE	Explain tables
DDL to create PLAN_TABLE stored in: SDSNSAMP(DSNTESC)	DDL to create EXPLAIN tables stored in: sqllib\misc\explain.ddl
Visual Explain tool	Visual Explain tool. Note however that these two Visual Explain tools are different, as they analyze Explain tables that have different formats in DB2 S/390 vs DB2 ULWO .
N/A	db2exfmt (This is like a text version of Visual Explain)
Optimization hints	Update SYSSTAT catalog views
Query I/O parallelism	I/O parallelism based mainly on hardware configuration, and some DB2 parameters.
Query CP parallelism	Intra-partition parallelism
Sysplex parallelism	Inter-partition parallelism

DB2 S/390 Concept	DB2 ULWO Analogy
DEGREE bind/rebind option	DEGREE precompile/bind option
CURRENT DEGREE zparm	DFT_DEGREE db cfg
CURRENT DEGREE special register	CURRENT DEGREE special register
MAX DEGREE zparm	MAX_QUERYDEGREE dbm cfg
N/A	INTRA_PARALLEL dbm cfg
N/A	RUNTIME DEGREE
N/A	DB2DEGREE
MAX DEGREE = 1 should not be used to disable parallelism.	MAX_QUERYDEGREE = 1 or DFT_DEGREE = 1 should not be used to disable parallelism as memory is allocated. Specify INTRA_PARALLEL = NO instead
DB2 PM	Snapshot monitor
DB2 PM	Event monitor
-start trace	Turn on monitor switches

Part IV. Application Considerations and Connectivity

Chapter 10 - DB2 S/390 Connectivity

10.1 DB2 S/390 supported protocols

DB2 S/390 supports two protocols:

- **DRDA**

This is the recommended protocol. The application connects to a server at another location and executes packages that have been previously bound at that server. The application uses a CONNECT statement, a three-part name, or an alias (if bound with DBPROTOCOL (DRDA) to access the server.

- **DB2 private protocol**

This protocol is being phased out. The application must connect using an alias or three-part name to direct the SQL statement to a given location. Private protocol works only between application requesters and servers that are both DB2 S/390 subsystems; thus this protocol cannot be used to connect to a DB2 ULWO server. A statement is executed using DB2 private protocol access if it refers to objects that are not at the current server and is implicitly or explicitly bound with DBPROTOCOL(PRIVATE). A three-part name consists of a location, an authorization ID, and an object name. For example, for the following SQL statement, NYSERVER is the location name, DB2USER is the authorization ID, and TEST is the table name:

```
SELECT * FROM NYSERVER.DB2USER.TEST
```

10.2 DB2 S/390 as a DRDA Server

When connecting from a DRDA Requester to the DB2 S/390, no setup is required at the target DB2 S/390 DRDA Server other than making sure DDF is started. DB2 ULWO does not currently have by itself DRDA Requester capability, thus another software component -- DB2 Connect -- must be used to provide such capability.

DB2 Connect comes in two flavors:

- DB2 Connect Personal Edition (PE), and
- DB2 Connect Enterprise Edition (EE).

DB2 Connect PE must be installed in each DB2 ULWO client that connects to the mainframe, or alternatively, you can install DB2 Connect EE on a gateway machine and have all DB2 ULWO clients go through this gateway to connect to the mainframe. There are no code differences between PE and EE versions of DB2 Connect other than the number of licenses. When DB2 S/390 is the DRDA Requester accessing a DB2 ULWO DRDA Server, as we will see in the next section, DB2 Connect is *not* required. DB2 S/390 has DRDA Server and DRDA Requester capability. DB2 ULWO only has DRDA Server capability, DB2 Connect provides DRDA Requester capability to DB2 ULWO. Note also that DB2 ULWO Enterprise Edition (EE) comes with DB2 Connect EE included.

The tables below show the steps to follow when connecting from a DB2 ULWO client to DB2 S/390 using DB2 ULWO's Command Line Processor (CLP). Only basic command options will be used. TCPIP is the most common network protocol used today, so this is the only one described in this section. The Client Configuration Assistant (CCA) GUI tool can also be used to set up the connectivity to the mainframe.

Three cases are explained. The first case is mentioned for completeness, but it is not related to DB2 S/390; it describes connectivity between a DB2 ULWO client and DB2 ULWO server:

1) From DB2 UNIX/Linux/Windows machine to DB2 UNIX/Linux/Windows machine (No need to have DB2 connect installed)

Machine 1 ('libra') UNIX/Linux/Windows	Machine 2 ('mpower') UNIX/Linux/Windows
Commands to run on this machine:	Information you need to obtain from this machine, to perform the commands on machine 1:
db2 catalog tcpip node mpwrnode remote 158.228.20.100 server 50000 Note: 'mpwrnode' is an arbitrary name chosen for the node	For this example: •158.228.20.100 = IP address of machine 2. •50000 = The port used for DB2. To find out the port used, go to file /etc/services (UNIX) and grep for your DB2 instance name. For example, if your instance name is 'db2inst1', you will normally find entries like this: db2cdb2inst1 50000/tcp db2idb2inst1 50001/tcp Pick the port associated with db2cdb2inst1 (note 4th letter 'c') For Windows, review X:\WINNT\System32\drivers\etc\services
db2 catalog db a42ds1 at node mpwrnode	•a42ds1 = Name of the database you want to access in machine 2.
db2 terminate	
db2 connect to a42ds1 user <userid> using <password>	

2) From DB2 UNIX/Linux/Windows machine to DB2 for OS/390 machine (DB2 Connect MUST be installed on UNIX/Linux/Windows machine)

Machine 1 ('libra') UNIX/Linux/Windows	Machine 2 ('bigblue') DB2 for OS/390
Commands to run on this machine:	Information you need to obtain from this machine, to perform the commands on machine 1:
db2 catalog tcpip node bignode remote 158.228.20.3 server 446 Note: 'bignode' is an arbitrary name chosen for the node.	For this example: <ul style="list-style-type: none"> •158.228.20.3 = IP address of machine 2. •446 = The port used for DB2. Port 446 is normally the default value. To find out the port used, contact your DB2 for OS/390 DBA who can check the MVS syslog for message DSNL004I. "TCPPORT" in that message contains the port to use. Also, the -DISPLAY DDF command provides this info.
db2 catalog db d42d1 at node bignode authentication des	<ul style="list-style-type: none"> •'d42d1' is the name you will use to connect to the DB2 for OS/390 subsystem. This name is chosen arbitrarily.
db2 catalog dcs db d42d1 as S/390loc	<ul style="list-style-type: none"> •S/390loc = The LOCATION NAME of the DB2 subsystem you want to access on machine 2. To find out the LOCATION NAME, contact your DB2 for OS/390 DBA who can check the MVS syslog for message DSNL004I. "LOCATION" in that message contains the LOCATION NAME to use. Also, the -DISPLAY DDF command provides this info.
db2 terminate	
db2 connect to d42d1 user <userid> using <password>	

3) From DB2 UNIX/Linux/Windows machine (client, machine 0) to DB2 UNIX/Linux/Windows machine (With DB2 Connect as a gateway, machine 1) to DB2 for OS/390 machine (machine 2)

From machine one to machine two, follow the same instructions as in case #2 . There is no need to catalog a database from the DB2 Connect gateway unless you want to connect from that machine.

Machine 0 ('myClientPC') UNIX/Windows	Machine 1 ('libra') UNIX/Windows
Commands to run on this machine:	Information you need to obtain from this machine, to perform the commands on machine 0:
db2 catalog tcpip node gatewayn remote 9.82.24.10 server 50000 Note: 'gatewayn' is an arbitrary name chosen for the node.	For this example: •9.82.24.10 = IP address of machine 1. •50000 = The port used for DB2. To find out the port used, go to file /etc/services and grep for your DB2 instance name (same as case #1).
db2 catalog db d42d1 at node gatewayn authentication dcs_encrypt	•d42d1 = Name of the database you want to access as specified in the "catalog dcs" command in this gateway machine (machine 1).
db2 terminate	
db2 connect to d42d1 user <userid for DB2 OS/390> using <password DB2 OS/390>	

What to check if you cannot connect:

Client Machine	Database Server
ping 9.82.24.10	•9.82.24.10 = IP address of database server This will confirm if there are problems or not with the network.
	If the database server is DB2 UDB UNIX/Linux/Windows: •Is DB2 started? If not run db2start •Is DB2COMM set to TCPIP? Check by executing: db2set -all. If this registry variable is not set, you should execute: db2set db2comm=tcPIP •Is SVCENAME set to the port number or service name in /etc/services? Check this parameter by issuing: db2 get dbm cfg
	If the Database server is DB2 UDB for OS/390: •Is DB2 started? If not, execute -start db2 •Is DDF started? If not, execute -start ddf

Other related commands that you may want to use:

Command	Command Description
db2 list db directory	Lists the databases you have cataloged
db2 list node directory	Lists the node directories you have cataloged
db2 list dcs directory	Lists the dcs databases you have cataloged
Uncatalog database <dbname>	Uncatalogs a database entry (in case you incorrectly cataloged a database)
Uncatalog dcs database <dbname>	Uncatalogs a dcs database entry
Uncatalog node <node name>	Uncatalogs a node entry
db2licm -l	Lists which DB2 product(s) you have installed and licensed
db2 ? <command name>	Displays the syntax to use for <command name> Eg: db2 ? catalog ==> Will show the syntax of the 'catalog' command

10.3 DB2 S/390 as a DRDA Requester

DB2 S/390 has the DRDA Application Requester function built into DDF. DDF will get its configuration data for outbound connections only from the Communication Database (CDB) tables, which are part of the catalog. The following steps are required to set up the connectivity to a server using TCPIP:

- Configure TCPIP

This is normally performed by your TCPIP network specialist in the mainframe environment. Hosts file you may check are: TCPIP.HOSTS.LOCAL and TCPIP.ETC.SERVICES

- Populate the CDB tables

DDF needs at least two entries: One for the new location (in SYSIBM.LOCATIONS) and one that holds the remote IP address or hostname (in SYSIBM.IPNAMES). For example, if you would like to connect to a DB2 ULWO database called 'SAMPLNT', which uses PORT 50000 and has a hostname (or IP address) of ntserver.ibm.com, the following INSERTs should be performed:

```
INSERT INTO SYSIBM.LOCATIONS
(LINKNAME, PORT)
VALUES ('SAMPLNT', 'NTSERVER', '50000');

INSERT INTO SYSIBM.IPNAMES
(LINKNAME, IPADDR)
VALUES ('NTSERVER', 'ntserver.ibm.com');
```

- Restart DDF

DDF reads the CDB tables only during processing of -start ddf; therefore, if you make changes to the CDB tables, you need to restart DDF (-stop ddf, -start ddf) so they become active.

- Bind packages to the application server

Unlike DB2 ULWO , DB2 S/390 sample applications do not automatically bind their packages to the Application Server, you have to manually bind them. The BIND PACKAGE option in DB2I can help you perform this task. Make sure to input the correct value for the LOCATION NAME of the server.

To connect to the DB2 ULWO Application Server, you can use the CONNECT TO statement.

```
CONNECT TO <location name/host variable> USER <host variable> USING <host variable>
```

Chapter 11 - Application Development Environment

DB2 ULWO users may be mostly interested in writing UNIX/Linux/Windows client applications to access a DB2 S/390 subsystem through DB2 Connect. For this, all that needs to be set up is the connectivity to the mainframe, as well as optionally configuring some keywords required for ODBC/CLI programs. In this section, we will briefly describe some application issues when coding in the mainframe environment as well as issues when coding applications on the client to access DB2 on the mainframe.

11.1 Application environment

DB2 S/390 supports applications written with any of these methods:

- Static SQL
- Dynamic SQL
- CLI/ODBC
- JDBC
- SQLJ

Declaring your tables and view definitions within your program can be done using the SQL DECLARE statement, but most developers use the DCLGEN tool, which generates the declaration statements for you.

ODBC (FMID JDB7717) and JDBC/SQLJ (FMID JDB7712) can be optionally installed in DB2 S/390. These will provide the drivers to write ODBC/JDBC/SQLJ applications in the mainframe. If you have any of these types of applications written in a DB2 ULWO client, there is no need to install the above support, since the drivers you would be using are the ones from the client.

When working with client or server CLI/ODBC programs, you may want to consider the following CLI Keywords applicable to DB2 S/390:

Table 17 - CLI Keywords in db2cli.ini file related to DB2 S/390

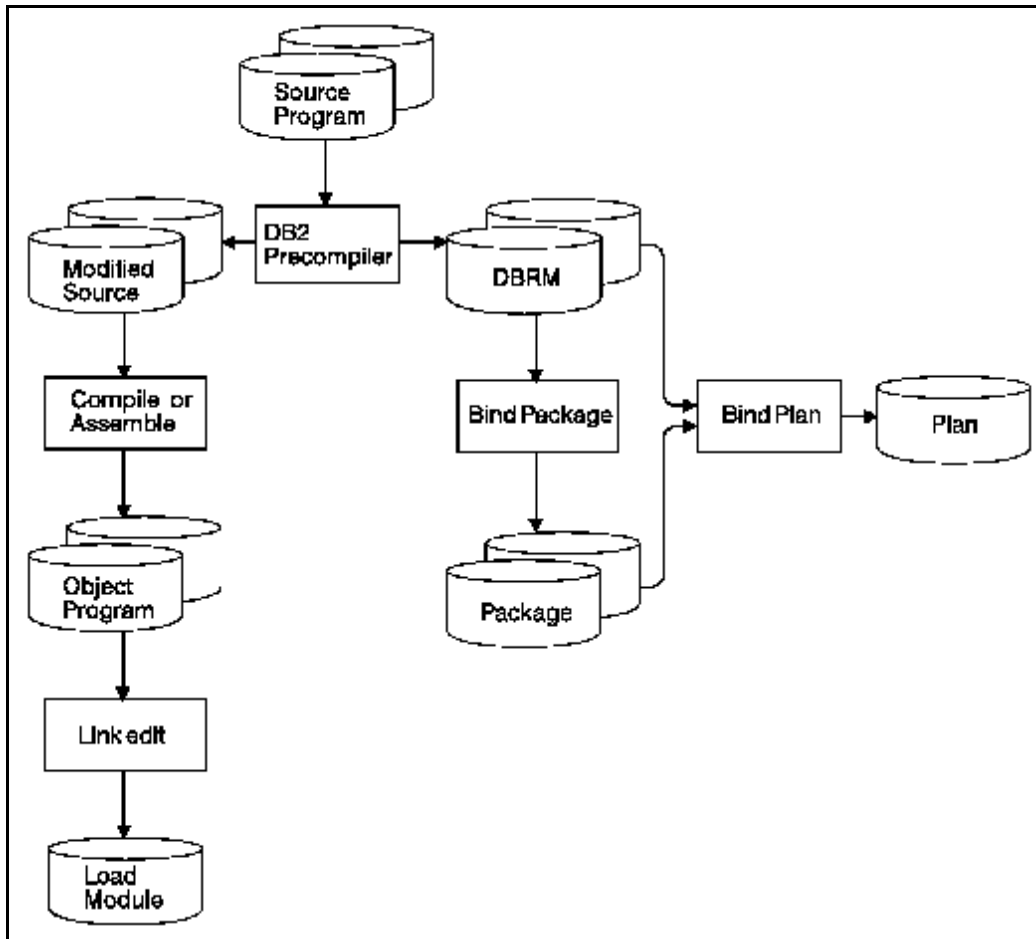
CLI Keyword	Description
DisableUnicode	Disable underlying Unicode support. The default is to use Unicode; thus when connecting to a DB2 S/390 database that does not have Unicode support installed, you may get conversion errors (for example, a carriage return or line feed character may be stored incorrectly into the database, and thus when retrieved it causes problems). Disabling Unicode at the client, or creating the DB2 S/390 as a Unicode database/tablespace/table would resolve this conversion issue.
ConnectCodepage	Indicates a specific code page to use when connecting to the data source to avoid extra connection overhead. Related to the Unicode problem described in 'DisableUnicode'.
UnicodeServer	Indicates that the data source is a unicode server. Equivalent to setting ConnectCodepage=1208. Related to the Unicode problem described in 'DisableUnicode'.

CLI Keyword	Description
DisableKeysetCursor	Disables keyset-driven scrollable cursors. This may be useful when having problems with server-side scrollable cursors.
Patch1	This keyword set to specific values will indicate to the CLI/ODBC program that a workaround should be used for known CLI/ODBC application problems. An applicable value when working with a client application accessing the mainframe is: PATCH1=32768. This value allows Microsoft Query to work with DB2 MVS synonyms.
Patch2	Similar to PATCH1, this keyword is used to indicate a workaround for a known CLI/ODBC application problem. Applicable values are: PATCH2=19 DB2 S/390 V4.1, allow parenthesis in the ON clause in an Outer join PATCH2=20 DB2 S/390 rewrite BETWEEN predicates with parameter markers as both operands PATCH2=44 Revert back to previous cursor state after the cursor was downgraded (MVS specific)
JDBCTrace	Turn on the DB2 JDBC trace facility.
JDBCTracePathName	Subdirectory used to store individual DB2 JDBC trace files.
Trace	Turn on the DB2 CLI/ODBC trace facility.
TraceFileName	File used to store the DB2 CLI/ODBC trace information.
TracePathName	Subdirectory used to store individual DB2 CLI/ODBC trace files.

11.2 Precompiling and binding

Figure 12 below shows a picture of how programs are prepared in DB2 S/390. The steps are similar as in DB2 ULWO; the difference is mainly in the DB2 S/390 concept of plans, packages, package lists and collections. These objects provide more complexity but at the same time more flexibility to applications in the mainframe. DB2 ULWO only uses packages, and collections (schema) in some cases.

Figure 12 - DB2 S/390 Program Preparation



A DB2 S/390 database request module (DBRM) corresponds to a DB2 ULWO bind file. In DB2 ULWO one can use the precompile command without the bindfile option, which will create a package directly in the database; alternatively, if the 'bindfile' option is used, a bind file will be created, and the bind command will need to be run to create a package in the database. In DB2 S/390 you will always obtain a DBRM (that is, you cannot create a package directly), and the bind of the program can be to a **package** or to an application **plan**. In addition, packages can be bound into logical groups called **collections**, which can also be bound into a plan.

When binding a DBRM into a package, this DBRM should be for only a single SQL statement, for a subset of the program, or for an entire program. The package created will contain executable forms of optimized SQL.

A package name will have the format <location name>.<collection_id>.<dbrm or package_id>, where the location name is optional. The collection_id is like a qualifier or 'schema'. A package must be bound into a plan before it is usable. The following can be bound into a plan:

- DBRM
- Packages

- Collections

A **collection** is a group of associated packages that perform a specific function.

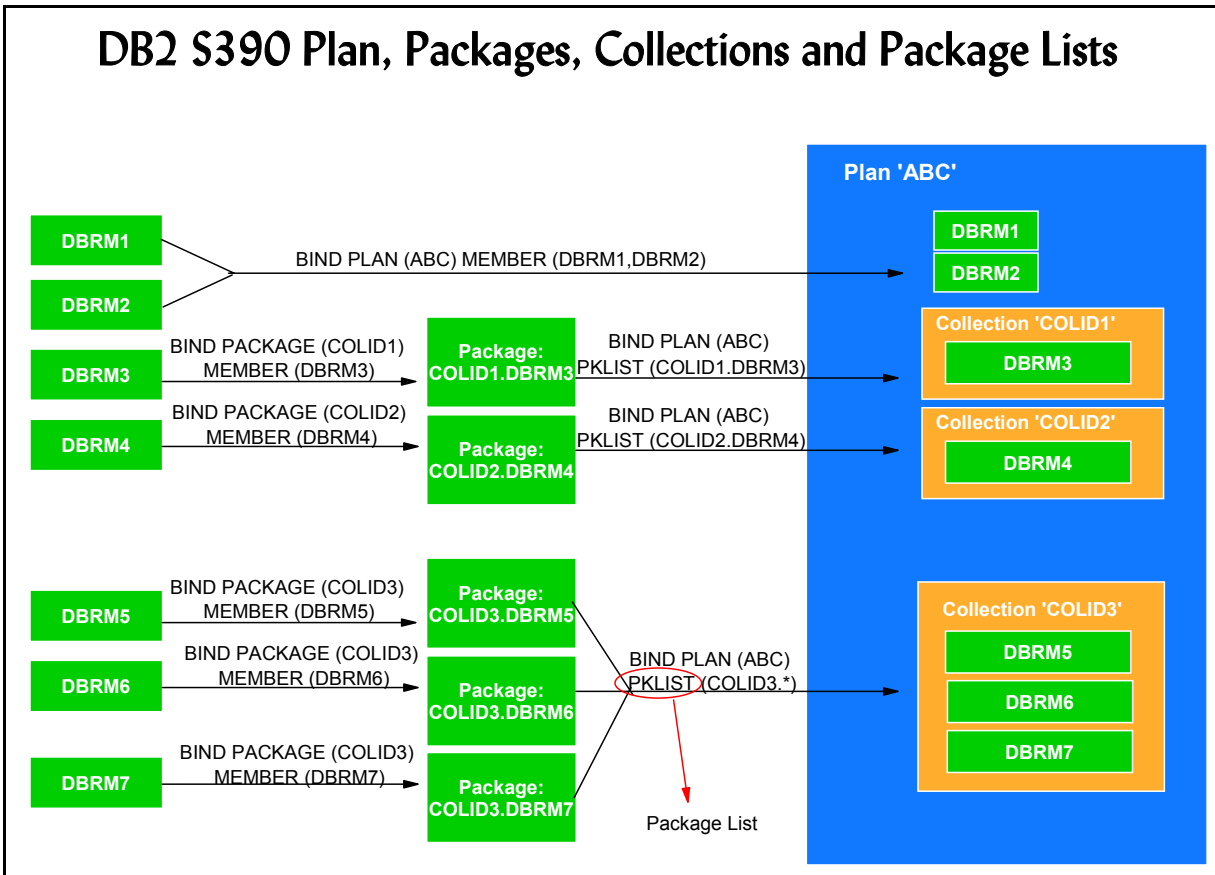
A **package list** is simply a list of packages or collections to be added to a plan. For example:

```

BIND PLAN (ABC) PKLIST (COLID3.* )
    
```

will bind into plan named ABC a package list containing all packages that start with a collection name of COLID3. Figure 13 below shows the relationship between packages, plans and collections.

Figure 13



11.3 Bind, rebind, free

In DB2 S/390 you can bind, rebind and free a plan or package. These operations mean the following:

- **Bind**
Builds a new plan or package.
- **Rebind**
Rebuilds a plan or package when changes to it affect the plan or package, but when the

SQL statements in the program have not changed. For example, you should rebind when you change authorizations, create a new index that the package uses, or use RUNSTATS.

- **Free**

Lets you delete plans and packages from DB2

In DB2 ULWO the bind and rebind commands perform the same functions as the corresponding commands in DB2 S/390. DB2 ULWO only binds to create packages (plans cannot be created). The DB2 ULWO db2rbind command can be used to bind more than one package at a time. DB2 ULWO doesn't have a command similar to DB2 S/390's FREE command; instead, you would use DROP PACKAGE <package name>.

11.3 Stored procedures

DB2 S/390 provides two types of stored procedures: DB2-established and WLM-established. DB2-established stored procedures run on a single address space (SPAS), while WLM-established stored procedures can run separately, each on its own WLM address space. This second type of stored procedure is the recommended one, as it isolates your procedures and avoids crashing your other running procedures if there is a problem with one of them.

In DB2 ULWO, there are *fenced* and *unfenced* stored procedures. Fenced stored procedures run in an address space that is different from the one used by DB2 engine, while unfenced stored procedures run in the same address space. DB2 S/390 stored procedures correspond to DB2 ULWO's fenced stored procedures, because they do not run in any of the DB2 main address spaces.

Stored procedures can be written in different languages (e.g., REXX, Java™, SQL Procedure Language). The redbook *Cross-Platform DB2 Stored Procedures: Building and Debugging* (SG245485), published on May of 2001, is an excellent source of information about stored procedures. The redbook has 537 pages, and can be accessed online at:

<http://www.redbooks.ibm.com/pubs/pdfs/redbooks/sg245485.pdf>

11.4 Triggers

A trigger is a set of actions that will be executed when a defined event occurs. The triggering events can be the following SQL statements: INSERT, UPDATE, and DELETE. Trigger usage and activation (before trigger, after trigger) is exactly the same as in DB2 ULWO. Trigger information in DB2 S/390 can be obtained from catalog tables SYSIBM.SYSTRIGGERS and SYSIBM.SYSPACKAGE. In DB2 ULWO, trigger information can be obtained from catalog views SYSCAT.TRIGGERS and SYSCAT.TRIGDEP.

11.5 User-defined functions (UDFs)

DB2 S/390 comes with many built-in functions; however, it is possible to create your own scalar and table functions, these are called user-defined functions (UDFs). UDF programs execute in an OS/390 Workload Manager (WLM) environment. All UDFs in DB2 S/390 execute as fenced in their own address space.

DB2 ULWO also supports UDFs in a similar way to DB2 S/390. UDFs can be run as fenced or unfenced.

11.6 DB2 Extenders™

DB2 S/390 supports Image, Audio, Video, Text and XML extenders. This is similar to DB2 ULWO .

11.7 Summary

The following table summarizes and compares some of the topics covered in this chapter.

Table 18 - Application Development in DB2 ULWO and DB2 S/390

DB2 S/390 Concept	DB2 ULWO Analogy
DBRM	Bind file
PACKAGE	PACKAGE
COLLECTION	COLLECTION (schema)
PLAN	N/A
PACKAGE LIST	N/A
BIND	BIND
REBIND	REBIND
FREE	DROP PACKAGE <package name>
Stored Procedures supported	Stored Procedures Supported
Two types of stored procedures (SPs): DB2-established: All SPs run in the SPAS address space. WLM-Established: Each SP can run on its own WLM address space. The two types of DB2 S/390 SPs would mostly be similar to DB2 ULWO fenced SPs as they don't use the address space used by the DB2 engine.	Two types of Stored Procedures (SPs): Fenced: The SP does not run in the same area where the DB2 engine runs. Unfenced: The SP runs in the same area where the DB2 engine runs (better performance, but at a risk of crashing DB2).
Triggers are supported	Triggers are supported
UDFs are supported	UDFs are supported
Image, Audio, Video, Text and XML extenders are supported.	Image, Audio, Video, Text and XML extenders are supported.

Chapter 12 - Locking and Concurrency

12.1 Locking data

DB2 uses several mechanisms to lock different objects. IRLM is used to manage transaction locks, latches (a kind of internal “fast” lock) are used to manage indexes, and drains and claims are used to control utilities and commands.

The following objects, listed in hierarchical order starting with the largest, can be locked in DB2 S/390:

- Tablespace
- Table (only if the table is in a segmented tablespace)
- Partition (for partitioned tablespaces)
- Page in a tablespace, row in a table, LOB lock

Locking at the tablespace level provides the least concurrency and the best use of resources, while locking at the row level provides the most concurrency, but the most use of resources (overhead in terms of processing time and storage). Row locks are not recommended in a data sharing environment because of this extra overhead. Note that a row lock and a page lock are at the same level, which means that a row lock will not escalate to a page lock and vice-versa, but they would escalate to a table lock.

Use the LOCKSIZE option of CREATE TABLESPACE or ALTER TABLESPACE to indicate which object will be used as the lock size. Possible values for LOCKSIZE are ANY, TABLESPACE, TABLE, PAGE, ROW, and LOB. ANY indicates that DB2 will be the one to choose the appropriate lock size. Normally, DB2 chooses LOCKSIZE PAGE.

Locking is handled by DB2 based on the isolation level selected for your application; the only explicit statement that a user can issue to lock a table or partition is the LOCK TABLE statement. This statement can be invoked with the IN EXCLUSIVE MODE option or the IN SHARE MODE option. This type of lock will normally be released at either COMMIT or ROLLBACK, but is also dependant on the settings of the ACQUIRE/RELEASE parameters to be covered in a later section.

With respect to LOB locks, these have different characteristics from regular locks. There is no concept of page locking or row locking with a LOB.

In DB2 ULWO, lockable objects are:

- Tables
- Rows

The default is ROW level locking. In DB2 ULWO the locksize is specified at the TABLE level rather than at the tablespace level as in DB2 S/390, and you can only use the ALTER TABLE statement with the LOCKSIZE option; the CREATE TABLE statement cannot be used. In DB2 S/390 as mentioned earlier, the CREATE TABLESPACE or ALTER TABLESPACE with the LOCKSIZE parameter could be used.

Tablespaces can be locked when utilities are performed against them. Locking at the database level can be achieved when connecting in exclusive mode, for example:

```
db2 connect to <dbname> in exclusive mode
```

In DB2 ULWO like DB2 S/390, the only statement that can be used explicitly to lock an object is LOCK TABLE.

12.2 Lock modes

DB2 S/390 has the following lock modes:

- IS Intent Share
- S Share
- IX Intent Exclusive
- U Update
- SIX Share with intent exclusive
- X Exclusive

An 'I' in a table/tablespace lock stands for INTENT. This means that row or page locks are in use on the individual pages. In all other cases, the table/tablespace lock will be the only lock used. The intent locks act as an indicator to DB2 to identify what is occurring within the table/tablespace.

DB2 ULWO has some of the same lock modes, but also some other ones:

- For table locks: IN, IS, S, IX, SIX, U, X, Z
- For row locks: NS, S, U, NX, NW, X, W

12.3 Lock durations

Tablespace and table lock duration is determined by the BIND parameters ACQUIRE and RELEASE

- **ACQUIRE**

This parameter determines when the locks are taken. If ACQUIRE ALLOCATE is used, when the *first* SQL statement is issued, the maximum required lock is taken on *all* the objects in the plan or package. In the case of ACQUIRE USE, when a SQL statement is issued, the required lock is taken on the involved object at this time.

- **RELEASE**

This parameter determines when the locks are released. If RELEASE DEALLOCATE is used, the locks will be released at the end of the program; when RELEASE COMMIT is used, the locks will be released at commit time.

In DB2 ULWO, isolation levels determine the duration of a lock. In DB2 S/390, isolation levels and the above parameters determine the duration of a lock.

12.4 Isolation levels

Isolation levels are set at bind time, or for individual SQL statements with the WITH <isolation level> clause. The isolation levels supported with DB2 S/390 are:

- UR Uncommitted Read

- CS Cursor Stability
- RS Read Stability
- RR Repeatable Read

DB2 ULWO supports the exact same isolation levels. It also sets the isolation level at bind time or using the WITH <isolation level> clause as in DB2 S/390.

12.5 Avoiding locks

DB2 S/390 can avoid using page and row locks in different situations. Lock avoidance is normally the default for most use, as it removes a lot of locking overhead. DB2 executes a small instruction set to determine if an IRLM lock is truly needed for read-only cursors or ambiguous cursors. Using isolation level UR will also provide for lock avoidance. For more details about this topic review the *DB2 UDB for z/OS and OS/390 V7 Administration Guide*.

In DB2 ULWO, lock avoidance concepts are not externalized to users.

12.6 System parameters

The following DSNZPARMs are used to control locking:

- **RECURHL**

The use of the RECURHL=YES will release locks that are held by a cursor defined WITH HOLD. In DB2 ULWO there is no equivalent parameter.

- **IRLMRWT**

This is the number of seconds that a transaction will wait for a lock before a timeout is detected. The IRLM uses this for timeout and deadlock detection. The default value is 60 seconds. In DB2 ULWO, the database configuration parameter LOCKTIMEOUT is used for timeouts. The default value for this parameter is -1, which means that DB2 ULWO will wait forever to obtain a lock. The database configuration parameter DLCHKTIME is used for deadlock detection, the default is 10 seconds.

- **XLKUPDLT**

This parameter allows you to specify the locking method to use when a searched UPDATE or DELETE is performed. In DB2 ULWO there is no equivalent parameter.

- **NUMLKTS**

This represents the maximum number of locks on an object. If you are turning off lock escalation (LOCKMAX 0), you will need to increase this number. If you are using LOCKMAX SYSTEM, then the value here will be the value for SYSTEM, that is, the value specified in installation panel DSNTIPJ for the LOCKS PER TABLE(SPACE) field. In DB2 ULWO there is no parameter that would indicate the maximum number of locks for an object. The LOCKLIST database configuration parameter can provide a limit based on the amount of storage (in 4 KB) that is allocated for locks, but this is more similar to the DB2 S/390 MAXIMUM ECSA field in panel DSNTIPJ, which is not a DSNZPARM.

- **NUMLKUS**

This is the maximum number of page or row locks that a single application can hold concurrently on *all* tablespaces. If you were to specify 0 then there will be no limit on the number of locks, in which case there would be a chance of running out of storage if you do not commit frequently (DB2 uses approximately 250 bytes for each lock). This parameter is related to the **LOCKMAX** parameter which is set in the CREATE TABLESPACE or ALTER TABLESPACE statement. LOCKMAX specifies the maximum number of page, row, or LOB locks an application process can hold simultaneously *for a given tablespace*. The value for LOCKMAX is related to LOCKSIZE. When LOCKMAX is not specified, the following default values are used:

LOCKSIZE	Resultant LOCKMAX
ANY	SYSTEM
TABLESPACE, TABLE, PAGE, ROW, or LOB	0

In DB2 ULWO, the MAXLOCKS database configuration parameter is the most similar one. MAXLOCKS indicates a percentage of LOCKLIST per application, while DB2 S/390 NUMLKUS indicates an exact amount for the maximum number of locks. DB2 ULWO uses 72 bytes to hold a lock on an object that has no other locks held on it, and 36 bytes are required to record a lock on an object that has an existing lock held on it.

12.7 Claims and drains

Claim and drain locks are used to control concurrency between SQL processes and utilities, with partition independence being a major focus. This means utilities and SQL can concurrently access and update different partitions.

- **Claims**

When an application first accesses an object within a unit of work, it makes a claim on the object. It releases the claim at the next commit point. Unlike a transaction lock, the claim cannot persist past the commit point. To access the object in the next unit of work, the application must make a new claim.

- **Drains**

Drain locks are used to serialize access to partitions and page sets among utilities, commands, and SQL applications. The drain is initiated at any time, but the actual takeover of an object occurs only when all access to the object has been quiesced. The drain process acquires a lock to prevent subsequent access from occurring until the lock is released.

In DB2 ULWO the concept of claims and drains is not externalized.

12.8 Escalation and promotion

Lock escalation can occur when the value for LOCKMAX is reached. At this time, DB2 will release all of the locks held in favor of taking a larger lock. Lock escalation can happen for

objects defined with LOCKSIZE ANY, PAGE or ROW. The value of LOCKMAX is set on the CREATE TABLESPACE statement. In DB2 ULWO, lock escalation happens when either the database configuration parameter MAXLOCKS is reached, or when the database configuration parameter LOCKLIST (lock storage) is exceeded.

Lock promotion happens when a lock changes from one mode to another. For example a lock can change from IX (Intent exclusive) to X (exclusive). This is the same concept used in DB2 ULWO.

12.9 Summary

The table below summarizes and compares the topics covered in this chapter.

Table 19 - Locking and Concurrency in DB2 S/390 vs DB2 ULWO

DB2 S/390 Concept	DB2 ULWO Analogy
Lockable Objects in hierarchical order: - Tablespace - Table (only if the table is in a segmented tbls) - Partition - Page, Row, LOB Other 'internal' structures can also be locked like DBDs, the SKCT, SKPT	Lockable objects in hierarchical order: - Table - Row Other objects that can be locked are: Database (connecting in exclusive mode) Tablespaces (when performing tablespace operations)
CREATE TABLESPACE/ALTER TABLESPACE with the LOCKSIZE option. Possible values for this LOCKSIZE parameter are: ANY, TABLESPACE, TABLE, PAGE, ROW, LOB. ANY indicates that DB2 will be the one to choose the appropriate lock size. Normally, DB2 chooses LOCKSIZE PAGE.	ALTER TABLE ... LOCKSIZE. Possible values for LOCKSIZE: Table or row. The default value is row.
MAXROW - Option of CREATE TABLESPACE that specifies the maximum rows per page.	N/A
LOCKPART - Option of CREATE TABLESPACE used for partitioned tables.	N/A
LOCK TABLE statement	LOCK TABLE statement
Lock modes: IS, S, IX, U, SIX, X	Lock modes: For Table locks: IN, IS, S, IX, SIX, U, X, Z For Row locks: NS, S, U, NX, NW, X, W
Lock duration based on BIND parameters ACQUIRE and RELEASE and on isolation levels	Lock duration based on the isolation level.
Isolation levels supported: UR, CS, RS, RR	Isolation levels supported: UR, CS, RS, RR
The isolation level can be set for an individual SQL with the clause WITH <isolation level>	The isolation level can be set for an individual SQL with the clause WITH <isolation level>
Avoiding locks - Use of CURRENTDATA bind parameter.	N/A

DB2 S/390 Concept	DB2 ULWO Analogy
RECURHL dsnzparm	N/A
IRLMRWT dsnzparm, handles timeouts and deadlocks.	<ul style="list-style-type: none"> - LOCKTIMEOUT - DLCHKTIME
XLKUPDLT dsnzparm	N/A
NUMLKTS dsnzparm - Maximum number of locks for an object	N/A
MAXIMUM ECSA field in panel DSNTIPJ	LOCKLIST (amount of memory used for locks in 4 KB)
NUMLKUS dsnzparm -Maximum number of page or row locks that a single application can hold concurrently on all tablespaces.	MAXLOCKS, as a percentage of LOCKLIST
LOCKMAX - Option of CREATE TABLESPACE, similar to NUMLKUS but only for one tablespace as opposed to all tablespaces.	N/A
Claims and drains	N/A
Escalation can happen when: - LOCKMAX is reached	Escalation can happen when either: - MAXLOCKS is reached or - LOCKLIST is exceeded.
Concept of lock promotion	Same concept as in DB2 S/390

Part V. DB2 S/390 Tools

Chapter 13 - Fee-based optional DB2 Tools

13.1 Comparing fee-based optional DB2 tools with DB2 ULWO tools

Fee-based DB2 tools do not come with the DB2 S/390 software, but can be ordered separately for a fee. Refer to the Redbook *DB2 for z/OS and OS/390 Data Management Tools Update* (SG24-6406-00), published in February 2002 for more details about these tools.

The following table briefly compares these tools with DB2 ULWO tools. Most of the tools in the table are specific to the DB2 S/390 architecture, so the DB2 ULWO analogy provided may be 'forced'.

Table 20 - Fee-based optional DB2 tools

Classification	Tool Name	DB2 ULWO Analogy
Database Administration	DB2 Administration Tool	~ Control Centre
	DB2 Automation Tool	~ Script Center + Journal (can create a script to run REORG for example, and schedule it for execution)
	DB2 High Performance Unload	~ Export, db2move
	DB2 Object Comparison Tool	N/A
Performance Management	DB2 bufferpool Analyzer	N/A
	DB2 Performance Monitor	Snapshot and Event Monitor
	DB2 Query Monitor	~ db2gov (Governor)
	DB2 SQL Performance Analyzer	~ db2advis (Index Advisor)
Recovery and Replication	DB2 Archive Log Compression Tool	N/A
	DB2 Change Accumulation Tool	N/A
	DB2 Data Propagator	DB2 Data Propagator
	DB2 Log Analysis Tool	N/A
	DB2 Object Restore	N/A
	DB2 Recovery Manager	N/A
Application Management	DB2 Bind Manager	N/A. - db2rbind can be used to bind several packages at once
	DB2 Data Export Facility	db2move, export/import, load, db2look
	DB2 Path Checker	N/A
	DB2 Table Editor	N/A
	DB2 Web Query Tool	N/A

Chapter 14 - The DB2 Management Clients Package

The DB2 Management Clients Package, formerly called DB2 Management Tools Package, is a no-charge feature of DB2 S/390. It delivers GUI tools that are run from a client machine. This client package is free of charge, but must be ordered separately. Here is a brief description of these client tools:

14.1 DB2 Control Center

The DB2 UDB Control Center is the same graphical tool used in DB2 ULWO. This tool has been extended to support DB2 S/390. Once connectivity has been set up with a DB2 S/390 subsystem, access to it through the Control Center will be available. There will be items that only apply to DB2 S/390 databases; for example, you can enter commands like DISPLAY THREAD, DISPLAY LOCATIONS, etc.

14.2 DB2 Installer

The DB2 Installer helps new DB2 users to install and migrate DB2 in a user-friendly fashion. DB2 Installer also helps when installing DB2 Performance Monitor (DB2 PM) and Data Propagator.

14.3 DB2 Visual Explain

This tool provides an easy-to-understand graph of the access paths of SQL statements. Visual Explain can invoke EXPLAIN for dynamic SQL statements, provides suggestions for improving the access paths, and allows you to browse DB2's subsystem parameter values.

Note that DB2 S/390 Visual Explain is *different* from DB2 ULWO Visual Explain, but both are used for the same purpose. DB2 S/390 Visual Explain will not work with a DB2 ULWO database and DB2 ULWO Visual Explain will not work with a DB2 S/390 database.

14.4 DB2 Estimator

DB2 Estimator estimates performance, resource usage, efficiency, and the cost of both, existing and planned DB2 S/390 applications and presents them graphically.

14.5 Stored Procedure Builder

The Stored Procedure Builder is a graphical tool for rapid application development on Windows that supports the coding and debugging of DB2 stored procedures written in Java and SQL Procedures language across the various platforms. It supports Java Database Connectivity (JDBC) and SQL Java (SQLJ) interfaces. This tool is explained in detail in the redbook *Cross-platform DB2 Stored Procedures: Building and Debugging'* (SG24-5485-01).

14.6 Summary

The following table summarizes the GUI Tool information described in this chapter.

Table 21 - The DB2 Management Clients Package GUI Tools

GUI Tool Name	DB2 ULWO Analogy
DB2 Control Center	DB2 Control Center. This tool is the exact same tool for DB2 S/390 as for DB2 ULWO, however depending on which system is selected, menu options may vary.
DB2 Installer	N/A - DB2 ULWO installation is straight forward.
DB2 S/390 Visual Explain	DB2 ULWO Visual Explain. These two tools have the same name, and perform the same task; however, they are not the same tool. One is for DB2 S/390, and the other for DB2 ULWO .
DB2 Estimator	N/A
DB2 Stored Procedure Builder	DB2 Stored Procedure Builder - This is the exact same tool for DB2 S/390 as for DB2 ULWO. Depending on which system you are creating the stored procedure, different options will show up.

Appendix A - Comparison of DB2 ULWO vs DB2 S/390 commands

The following table shows some of the commands used in DB2 S/390 and the corresponding ones in DB2 ULWO .

Table 22 - Comparison of some DB2 ULWO vs DB2 S/390 Commands

DB2 S/390 Command	DB2 ULWO Analogy
-STOP DB2	db2stop
-STOP DB2 MODE(FORCE)	db2stop force
-START DB2	db2start
-TERM UTILITY(utility id)	db2 force application <appl handle>
-CANCEL THREAD (token id)	db2 force application <appl handle>
-DISPLAY DATABASE (LOCKS)	db2 get snapshot for locks on <dbname>
To check the tablespaces that are in restricted status:	To check the status of tablespaces:
-DISPLAY DATABASE (*) SPACENAM (*) RESTRICT	db2 list tablespaces show detail
-DISPLAY THREAD (*) TYPE(*)	db2 list applications show detail
-DISPLAY UTILITY (*)	db2 list applications show detail
SELECT NAME, TBNAME, COLTYPE, LENGTH, NULLS, DEFAULT FROM SYSIBM.SYSCOLUMNS WHERE TBNAME='<tablename>' AND TBCREATOR = '<creator name>';	db2 describe table
SELECT * FROM SYSIBM.SYSTABLES WHERE CREATOR = '<creator name>';	db2 list tables for schema <schema name>

Appendix B - DB2 S/390 Versions and OS versions

Table 23 - DB2 S/390 Versions and Minimum OS versions required

DB2 Version	MVS, OS/390 or z/OS Minimum level required	Notes
3.1	MVS/ESA SP 5.2.2	Out of support
4.1	MVS/ESA SP 5.2.2	Out of support
5.1	MVS/ESA SP 5.2.2	No longer marketed, but still supported. End Support date has not yet been announced.
6.1	OS/390 Version 1, Release 3	End Support date has not yet been announced.
7.1	- z/OS Release 1 - OS/390 Version 2, Release 7	End Support date has not yet been announced.

BIBLIOGRAPHY

	Title	Website
1	<u>DB2 for z/OS and OS/390 Data Management Tools Update</u> , SG24-6406-00 <i>Redbook</i> , published February-27-2002	http://www.redbooks.ibm.com/redbooks/SG246406.html
2	<u>New Tools for DB2 for OS/390 and z/OS Presentation Guide</u> , SG24-6139-00 <i>Redbook</i> , published March-4-2001	http://www.redbooks.ibm.com/redbooks/SG246139.html
3	<u>DB2 UDB for OS/390 Version 6 Management Tools Package</u> , SG24-5759-00 <i>Redbook</i> , published May-18-2000	http://www.redbooks.ibm.com/redbooks/SG245759.html
4	<u>Cross-Platform DB2 Stored Procedures: Building and Debugging</u> , SG24-5485-01 <i>Redbook</i> , published May-7-2001	http://www.redbooks.ibm.com/redbooks/SG245485.html
5	<u>Converting from Oracle AIX to DB2 for OS/390</u> , SG24-5478-00 <i>Redbook</i> , published December-22-1999	http://www.redbooks.ibm.com/redbooks/SG245478.html
6	<u>Migrating Siebel Database from DB2/Oracle for NT to DB2 for OS/390</u> , SG24-6236-00 <i>Redbook</i> , published November-26-2001	http://www.redbooks.ibm.com/redbooks/SG246236.html
7	<u>DB2 Universal Database V7.1 for UNIX, Linux, Windows, and OS/2 Database Administration Certification Guide</u> by George Baklarz and Bill Wong. Prentice Hall PTR, 2001	
8	<u>DB2 Universal Database for OS/390 Certification Guide Version 7.1</u> by Richard Yevich and Susan Lawson. Prentice Hall PTR, 2002	
9	<u>An Introduction to DB2 for OS/390 Version 7</u> by Susan Graziano Sloan and Ann Kilty Hernandez. Prentice Hall PTR, 2001	
10	DB2 ULWO Manuals	http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v7pubs.d2w/en_main
11	DB2 S/390 Manuals	http://www-3.ibm.com/software/data/db2/oS/390/v7books.html
12	z/OS and OS/390 Manuals	http://www-1.ibm.com/servers/eserver/zseries/zos/bkserv/