CHAPTER 2

DB2 Environment

- z/OS
- Address spaces
- Installation and migration
- System parameters
- Commands
- Utilities
- Catalog and directory
- Distributed data
- Subsystem pools

DB2 operates as a formal subsystem of z/OS. DB2 utilities run in the batch environment, and applications that access DB2 resources can run in the batch, TSO (Time Sharing Option), USS (UNIX System Services), IMS (Information Management System), or CICS (Customer Information Control System) environment. IBM provides attachment facilities to connect DB2 to each of these environments.

This chapter looks at the subsystem architecture that supports DB2 and the attachments available. This chapter also looks briefly at the installation process of the DB2 subsystem and the subsystem parameters known as DSNZPARMs, as well as how to execute the utilities and various commands.

DB2 also has the ability to access other DB2 subsystems and other relational databases. This chapter takes a look at the methods for doing this.

z/OS

The IBM zSeries family of enterprises servers use z/OS, the next generation of operating system replacing OS/390. The core of z/OS is the base control program MVS. Open-standard application server functions that support scalable, secure e-business applications are integrated with the MVS base.

The z/OS platform can provide scalable, secure processing for various types of workloads in a high-performance environment, such as stock trading. The applications can be batch, OLTP (Online Transaction Processing), or DSS (Decision Support Systems). The major strengths of this environment are availability, flexibility, manageability, reliability, scalability, and security.

The DB2 database management system operates as a formal subsystem of the z/OS operating system. DB2 processes can be executed in various address space regions within z/OS, such as IMS or CICS. Because it is required for DB2 version 8, z/OS provides a new architecture that supports a 64-bit operating system. Thus, the z/OS platform can support many diverse applications with increased availability, scalability, and security.

Address Spaces

The address spaces used in the operation of DB2 are

- System services address space (SSAS)
- Database services address space (DSAS)
- Internal resource lock manager (IRLM)
- Distributed Database Facility (DDF)
- Stored procedure address space (SPAS)
- Allied address spaces

Figure 2-1 shows the relationship among the address spaces DB2 uses.

The *system services address space,* also called the Data System Control Facility (DSCF) address space, is the DSN1MSTR address space. The following subcomponents execute in the SSAS:

- General command processor
- Subsystem support
- Agent services manager
- Storage manager

- Message generator
- Initialization procedures
- Instrumentation facilities
- System parameter manager
- Recovery manager
- Recovery log manager
- Group manager
- Distributed transaction manager

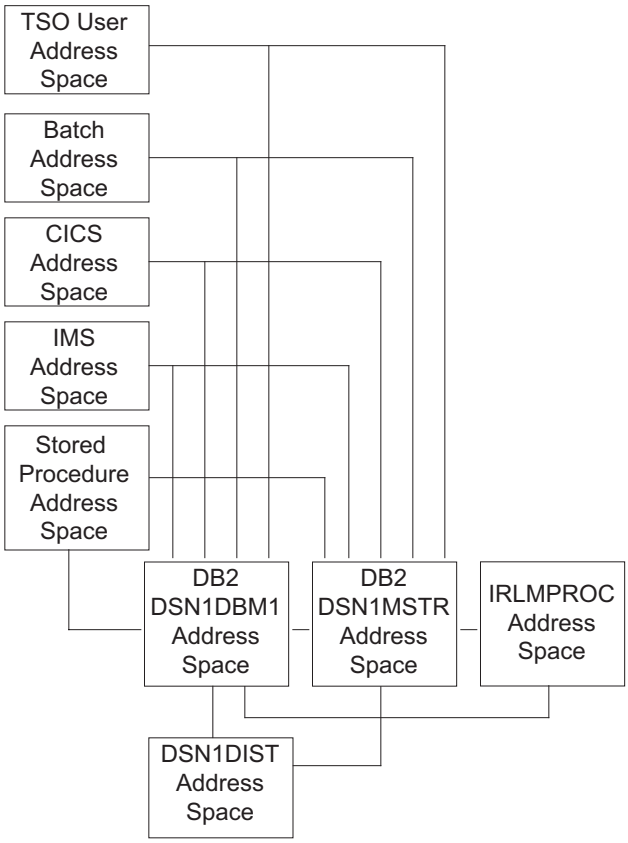

Figure 2-1 Relationship between DB2 users and DB2-related address spaces

This is critical, as it manages most of the activities in DB2. The *database services address space,* also called the Advanced Database Management Facility (ADMF) address space, is the DSNDBM1 address space. The following subcomponents execute in the DSAS:

- Service controller
- Data manager
- Large-object manager (LOBM)
- Data space manager
- Relational data system (RDS)
- Stored-procedures manager
- Utilities (work with associated code in an allied address space)
- Buffer manager

The DBM1 address space uses memory for several operations in DB2. The following output from a DB2 Performance Monitor report shows the DB2 objects that use memory in the DBM1 address space:

```
DBM1 STORAGE STATISTICS (MB)
-----------------------------------
TOTAL GETMAINED STORAGE
  VIRTUAL BUFFER POOLS
  EDM POOL
  COMPRESSION DICTIONARY
  CASTOUT BUFFERS
TOTAL VARIABLE STORAGE
  TOTAL AGENT SYSTEM STORAGE
  TOTAL AGENT LOCAL STORAGE
  RDS OP POOL
  RID POOL
  PIPE MANAGER SUB POOL
  LOCAL DYNAMIC STMT CACHE CTL BLKS
LOCAL DYNAMIC STMT CACHE STMT POOL
BUFFER & DATA MANAGER TRACE TBL
VIRTUAL POOL CONTROL BLOCKS
TOTAL FIXED STORAGE
TOTAL GETMAINED STACK STORAGE
STORAGE CUSHION
TOTAL DBM1 STORAGE
TOTAL NUMBER OF ACTIVE USER THREADS
TOTAL STORAGE FOR ALL THREADS
NUMBER OF PREFETCH ENGINES
NUMBER OF DEFERRED WRITE ENGINES
NUMBER OF CASTOUT ENGINES
NUMBER OF GBP WRITE ENGINES
NUMBER OF P-LOCK/NOTIFY EXIT ENGINES
```

Each DB2 subsystem has its own internal resource lock manager. The IRLM controls access to the application data and is used by DB2 as the lock manager to ensure the integrity of the data.

The Distributed Data Facility allows client applications running in an environment that supports Distributed Relational Database Architecture (DRDA) to access DB2 data. The DDF also allows for one DB2 subsystem to access data on another DB2 subsystem. Other relational database servers can be accessed if they support DRDA. TCP/IP and SNA (Systems Network Architecture) are the supported network protocols.

DDF permits up to 150,000 distributed concurrent threads to be attached to a single DB2 subsystem at a time. The following resource managers execute in the DDF services address space:

- Data communications resource manager (DCRM)
- Distributed data interchange services (DDIS)
- Distributed relational data system manager (DRDS)
- Distributed transaction manager (DTM)

The stored-procedure address space (DSN1SPAS) is established by DB2. SPAS provides an isolated environment in which to execute stored procedures.

> **NOTE** To create new stored procedures in version 8 it is recommended that you use WLM (Workload Manager) address spaces.

The Workload Manager (WLM) component of z/OS manages address spaces for the execution of stored procedures and user-defined functions. These address spaces are called application environments. When an application calls a stored procedure or external user-defined function, DB2 schedules this work with WLM to run in a WLM-managed application environment. Each DB2 subsystem may have many defined application environments.

The *allied address spaces* communicate with DB2. The SPASs operate as allied address spaces. The following subcomponents execute in allied address spaces:

- Attachment facilities: call attachments, CICS, IMS, recoverable resource manager services, TSO
- Message generator (standalone only)
- Subsystem support
- Utilities: DDAS; standalone

Address Space Priority

To establish the z/OS address space dispatching priorities, the IRLM address space should be placed above IMS, CICS, and DB2, with DB2's MSTR and DBM1 address spaces placed above CICS. The following is a recommended priority listing:

- MVS monitor with IRLM capabilities
- IRLM
- DB2 performance monitors
- DBM1
- MSTR
- WLM-managed application environments
- CICS

If the IRLM address space is not at the top of the dispatching priorities, DB2 locks will not get set and released without causing excessive wait time (see Chapter 16 for information on DB2 locks). A warning message will be issued if the IRLM is not above DBM1 at DB2 start-up.

It is important to pay attention to the dispatching recommendations of the vendors of the various performance monitors. Each vendor has one or more address spaces and always recommends specific dispatching priorities. If a performance monitor that can analyze problems in the IRLM is being used, dispatching IRLM higher than a DB2 and/or MVS monitor could be a problem. If a performance monitor can't get dispatched ahead of IRLM, you will not be able to find or analyze the problem.

Attachments

Attachment facilities provide the interface between DB2 and other environments. Only one attachment can be active within a program at any given time. Five attachment facilities can be used, depending on the environment in which the program will execute. The attachments are

- Customer Information Control System (CICS)
- Information Management System (IMS)
- Call Attach Facility (CAF)
- Resource Recovery Manager Services Attachment Facility (RRSAF)
- Time Sharing Option (TSO)

Each program runs in an allied address space. The DB2 attachment facility modules are called by the program, execute within the allied address space, and establish communication with the DB2 address spaces. When initialized, the attachment modules make the link to DB2, and agent control blocks are created in the DSNMSTR address space to control the program's DB2 processing.

When a program issues an SQL statement, the attachment modules are given control. The attachment programs prepare the request for DB2 and pass the request over to the DB2 address space. Results are passed back to the attachment programs, and any results are copied into the application storage areas.

Call Attach Facility

The DB2 Call Attach Facility is used for application programs that run under TSO or z/OS batch. The CAF is an alternative to the DSN command processor and provides greater control over the execution environment.

With the CAF, your application program can establish and control its own connection to DB2. Programs that run in z/OS batch, TSO foreground, and TSO background can use CAF.

IMS batch applications too can access DB2 databases through CAF, although that method does not coordinate the commitment of work between the IMS and DB2 systems. It is recommended that you use the DB2 Data Language/I batch support for IMS batch applications.

Programs using CAF can

- Access DB2 from address spaces where there is no TSO, CICS, or IMS
- Access the Instrumentation Facility Interface (IFI)
- Access DB2 from multiple tasks within an address space
- Control the connection to DB2 explicitly
- Connect to DB2 implicitly, using a default subsystem ID and plan name
- Receive a signal from DB2 on start-up or shutdown.

The Call Attach Facility is being slowly replaced by the more powerful RRS (resource recovery services) attachment facility.

Customer Information Control System

CICS is an application server that provides online transaction management for applications. You can use the CICS attachment facility to connect to DB2 from this environment. CICS facilitates access to DB2 data from the CICS environment. A CICS application can access both DB2 and CICS data; in the event of a failure, CICS coordinates the recovery between DB2 and CICS.

Once the DB2 subsystem has been started, DB2 can be operated from a CICS terminal. DB2 and CICS can be started independently of each other. The connection can also be made automatic. Figure 2-2 shows the relationship between DB2 and CICS. The CICS attachment facility manages multiple connections to DB2, called threads, that can be reused for improved performance.

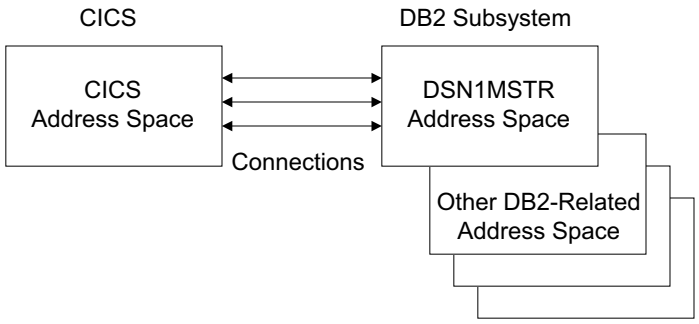

Figure 2-2 Relationship of DB2 to CICS

Information Management System

The IMS database computing system includes a hierarchical database manager, a transaction manager, and middleware products for access to the IMS databases and transactions. In order to connect to DB2, you can use the IMS attachment facility, which receives and interprets requests for access to DB2 data via exit routines in the IMS subsystems. IMS can connect automatically to DB2 without operator intervention. You can make DB2 calls from IMS applications by using embedded SQL statements. DB2 also provides the database services for IMS-dependent regions with the IMS attachment facility. An IMS batch environment has support for DL/1 batch jobs to have access to both IMS (DL/1) and DB2 data.

IMS will also coordinate recoveries between DB2 and IMS in the event of a failure. You also have the option to include DB2 in the IMS Extended Recovery Facility (XRF) recovery scenario. Figure 2-3 shows the relationship between IMS and DB2.

Resource Recovery Services Attachment Facility

RRSAF is a DB2 subcomponent that uses z/OS transaction management and resource recovery manager services to coordinate resource commitment between DB2 and all other resource managers that also use z/OS RRS in a z/OS subsystem.

An application program can use the RRSAF attachment facility to connect to and use DB2 to process SQL statements, commands, or IFI calls. Programs that run in z/OS batch, TSO foreground, and TSO background can use RRSAF.

RRSAF uses z/OS transaction management and resource recovery manager services (z/OS RRS). With RRSAF, you can coordinate DB2 updates with updates made by all other resource managers that also use z/OS RRS in a z/OS system.

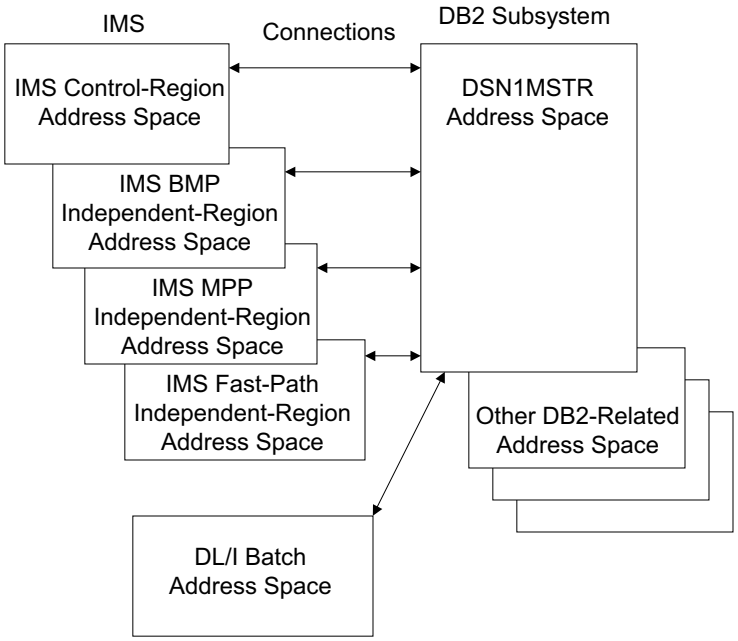

Figure 2-3 Relationship between DB2 and IMS

Programs that use RRSAF can

- Run under UNIX system services, TSO, or batch
- Use dynamic and static SQL Statements
- Sign on to DB2 with an alternative user ID
- Access DB2 from one or more tasks in an address space
- Use the IFI
- Control the exact state of the DB2 connection
- Capture DB2 start-up and termination events

Programs using RRSAF can be run from almost every environment available under z/OS. The programs can be invoked from the command prompt in TSO, from a shell prompt under UNIX system services, and, like any non-DB2 program using the PGM specification, on the EXEC card in Job Control Language (JCL).

Each task control block (TCB) can have only one connection to DB2. Using RRSAF, this connection can be switched between tasks within the address space. It is also possible to directly connect to multiple DB2 susbystems and switch between them during execution, although this would limit the ability to move subsystems to alternate systems in a sysplex.

Time Sharing Option

TSO allows interactive time-sharing capabilities from local and remote terminals. The TSO attachment facility is used by the majority of TSO applications. TSO controls programs, also allowing noninteractive batch program execution. In order to connect to DB2, you can use the TSO CAF or the RRS (resource recovery services). Historically, the TSO attachment facility has been the most commonly used connection for TSO interactive and batch applications.

Using TSO, you can bind application plans and packages and execute several online functions of DB2. Figure 2-4 shows the relationship between DB2 and TSO. Application programs and databases can be created, modified, and maintained via the TSO attach. You can run in either the foreground or batch when accessing DB2.

Access to two command processors is allowed when using TSO:

- DSN command processor
- DB2 Interactive (DB2I)

The DSN command processor runs as a TSO command processor using the attachment facility. This command processor provides an alternative method for executing programs in a TSO environment that accesses DB2. This processor can be invoked from the foreground by issuing a TSO command or from batch by invoking the TMP (terminal monitor program) from an MVS batch job, passing the commands in the SYSTSIN data set to TMP. When DSN is running and DB2 is up, you can issue DB2 or DSN commands (covered later in this chapter).

The DB2I comprises ISPF panels that allow for an interactive connection to DB2 and invokes the DSN command processor behind the scenes. With this processor, you can invoke utilities, issue commands, and run SQL statements. DB2I is discussed in more detail later in this chapter.

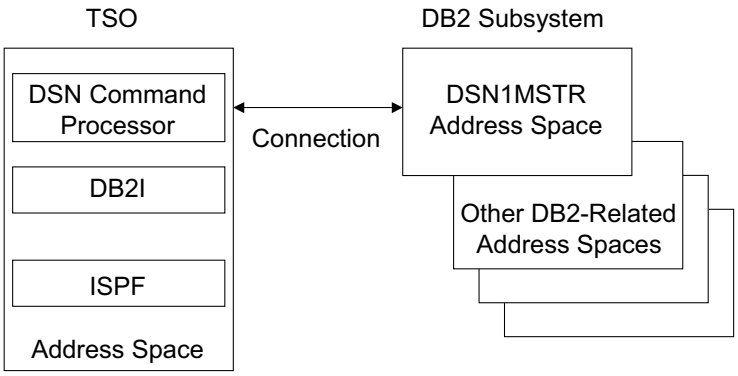

Figure 2-4 Relationship between DB2 and TSO

SECURITY

You can add security to your DB2 subsystem by using the Resource Access Control Facility (RACF). RACF is a component of the SecureWay Security Server for z/OS. Other, equivalent security packages are available on the market.

Use of RACF will prevent unauthorized users from having access to the system. RACF can be used to protect a variety of resources in DB2 and to check the identity of DB2 users. An exit routine that runs as an extension to DB2 is part of the Security Server and can also provide central control authorization to DB2 objects.

For more information on security options, refer to Chapter 3.

PARALLEL SYSPLEX SUPPORT

DB2 has the ability to run in a parallel sysplex environment. This environment is required to support DB2 data sharing (refer to Chapter 9 for more details), allowing us to configure an environment in which several processors can share data and the DB2 subsystems can have concurrent read/write access to this data. A parallel sysplex environment also gives us flexibility for adding new processors for added throughput, seamlessly routing workload away from failed processors, and balancing diverse work across multiple processors.

DFSMS

DFSMS (Data Facility Storage Management Subsystem) can be used to manage DB2 data sets automatically and to reduce the amount of administrative work for database and systems administrators. SMS facilitates allocation and movement of data, better availability and performance management, and automated space management. Most important is the aspect of performance improvements, as performance goals can be set for each class of data, thereby reducing the need for manual tuning. This environment also uses the cache provided by the storage hardware.

INTERFACES

A variety of ways to interface with DB2 are available, such as with programs using TSO, CAF, CICS, and so on. Two of the most common interfaces provided with the DB2 product are DB2 Interactive (DB2I) and Sequential Processing Using File Input (SPUFI).

DB2 Interactive

The interactive program DB2I can be used to run application programs and to perform many DB2 operations by entering values on panels. DB2I runs under TSO, using ISPF (Interactive System Productivity Facility) services. To use DB2I, follow the local procedures for logging on to TSO, and enter **`ISPF`**. Each operation is controlled by entering the parameters that describe it on the panels provided. DB2 also provides help panels to

- Explain how to use each operation
- Provide the syntax for and examples of DSN subcommands, DB2 operator commands, and DB2 utility control statements

The following tasks can be performed using DB2I:

- Execute SQL statements using SPUFI
- Perform a DCLGEN (Declarations Generator)
- Prepare a DB2 program
- Invoke the DB2 precompiler
- Execute BIND/REBIND/FREE commands
- RUN an SQL program
- Issue DB2 commands
- Invoke DB2 utilities
- Change DB2I defaults

An example of the DB2I menu is shown in Figure 2-5.

Figure 2-6 shows the defaults for DB2I. These defaults can be changed. Some of the parameters that can be changed are the DB2 subsystem on which to run the requests, application language of choice, number or rows returned to the session, and what string delimiter to use.

```
                          DB2I PRIMARY OPTION MENU

===>

Select one of the following DB2 functions and press ENTER.

1  SPUFI                  (Process SQL Statements)
2  DCLGEN                 (Generate SQL and source language declarations)
3  PROGRAM PREPARATION    (Prepare a DB2 application program to run)
4  PRECOMPILE             (Invoke DB2 precompiler)
5  BIND/REBIND/FREE       (BIND, REBIND, or FREE plans or packages)
6  RUN                    (RUN an SQL program)
7  DB2 COMMANDS           (Issue DB2 commands)
8  UTILTIES               (Invoke DB2 utilities)
D  DB2I DEFAULTS          (Set global parameters)
X  EXIT                   (Leave DB2I)

Press : END to exit       HELP for more information
```

Figure 2-5 DB2I menu

```
                          DB2I DEFAULTS PANEL 1

Command ===> _

Change defaults as desired:

1  DB2 NAME.............  ===> DSN        (Subsystem Identifier)
2  DB2 CONNECTION RETRIES  ===> 0          (How many retries for DB2 connection)
3  APPLICATION LANGUAGE... ===> COB2       (ASM, C, CPP, COBOL, COB2, IBMCOB,
                                            FORTRAN, PLI)
4  LINES/PAGE OF LISTING.  ===> 60         (A number from 5 to 999)
5  MESSAGE LEVEL.........  ===> I          (Information, Warning, Error, Severe)
6  SQL STRING DELIMITER..  ===> DEFAULT    (DEFAULT or ' or ")
7  DECIMAL POINT.........  ===> .          (. or , )
8  STOP IF RETURN CODE >=  ===> 8          (Lowest terminating return code)
9  NUMBER OF ROWS........  ===> 20         (For ISPF Tables)
10 CHANGE HELP BOOK NAMES? ===> NO         (YES to change HELP data set names)


Press : ENTER to process  END to cancel    HELP for more information
```

Figure 2-6 DB2I defaults screen

Sequential Processing Using File Input

SPUFI is a DB2I menu option that allows you to execute SQL statements interactively with DB2. The options on the main menu are as follows.

- **Input data set name** is the data set that contains one or more SQL statements that are to be executed. The data set needs to exist before SPUFI can be used.
- **Output data set name** is the data set that will receive the output from the SQL statement that is executed. This data set does not have to exist before execution.
- **Change defaults** allows for changing control values and characteristics of the output data set and the format of SPUFI settings.
- **Edit input** allows for editing the SQL statements.
- **Execute** indicates whether to execute the SQL statement in the input file.
- **Autocommit** tells DB2 whether to make the changes to the data permanent.
- **Browse output** allows for browsing of the query output.
- **Connect location** specifies the name of the application server to receive the queries.

SPUFI allows you to have multiple queries in one input data set member for execution. The queries can be stacked by using a `;` between the statements. Without an explicit `COMMIT` statement, all the SQL statements are considered a unit of work.

After the SQL has been executed, SPUFI displays both the SQLCODE and the SQLSTATE. Figure 2-7 shows the main screen for SPUFI.

```
SPUFI                              SSID: DSN
===>

Enter the input data set name:        (Can be sequential or partitioned)
 1  DATA SET NAME ... ===>            ISPF.SQL(RECURS1)
 2  VOLUME SERIAL ... ===>            (Enter if not cataloged)
 3  DATA SET PASSWORD ===>            (Enter if password protected)

Enter the output data set name:       (Must be a sequential data set)
 4  DATA SET NAME ... ===>            OUTPUT.DATA

Specify processing options:
 5  CHANGE DEFAULTS   ===> YES        (Y/N - Display SPUFI defaults panel?)
 6  EDIT INPUT ...... ===> YES        (Y/N - Enter SQL statements?)
 7  EXECUTE ......... ===> YES        (Y/N - Execute SQL statements?)
 8  AUTOCOMMIT ...... ===> YES        (Y/N - Commit after successful run?)
 9  BROWSE OUTPUT ... ===> YES        (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===>


PRESS:  ENTER to process    END to exit              HELP for more information
```

Figure 2-7 SPUFI panel

INSTALLATION AND MIGRATION

Installation is the process of preparing DB2 to operate as a z/OS subsystem. *Migration* is the process of upgrading to a more current release of DB2. Both processes require the same steps. This book does not cover the details of each migration step; if you require more information, refer to the *DB2 Version 8 z/OS Installation* manual.

Before you begin installing or migrating, plan the amount of direct-access storage and virtual storage you need. Planning and coordinating with other DB2 subsystems is essential if you plan to install the DDF. For more information, refer to the *IBM DB2 for z/OS Version 8 Installation Guide*. Review what values are needed for the parameters on the installation and migration panels.

Another option for installation is to use the msys (Managed System Infrastructure) for Setup DB2 Customization Center. This reduces the complexity of configuring a DB2 subsystem when installing, migrating to compatibility mode, or enabling new function mode, or enabling data sharing. The msys for Setup workplace provides a GUI tool similar to Windows Explorer, that is used to manage z/OS products. The msys for Setup management directory stores configuration data for all systems.

The DB2 Customization Center can be added to the msys for Setup workplace. When refreshed, it pulls the current DB2 and z/OS settings. These settings can be reviewed and/or changed. An update can be performed to apply the changes. If a CLIST (command list) was previously used to customize a DB2 subsystem, this can be cloned by specifying the name of the output member generated by the CLIST and then performing a refresh to obtain the information.

Once in the DB2 Customization Center, the parameter values can be copied from one DB2 to another. The tasks executed during the update process are equivalent to those performed by the DB2 installation Job Control Language (JCL) jobs. If you wish to have JCL jobs configure DB2 on the host, the DB2 Customization Center can also generate an output member to be used as input to the CLIST.

For more information on the msys for Setup DB2 Customization Center, refer to the *z/OS Managed System Infrastructure for Setup Users Guide*.

DB2 provides a set of tools that automate the process of installing or migrating. These tools include most of the JCL needed to install and migrate the product. This JCL constitutes the *installation and migration job*s. Each of these jobs helps you perform a task when installing or migrating.

DB2 also includes the installation CLIST (command list) to help tailor the installation and migration jobs. This CLIST, also called the *migration CLIST*, or simply the *CLIST*, contains the code necessary to tailor the jobs to your needs. With the Interactive Systems Productivity Facility (ISPF) and Interactive Systems Productivity Facility/Program Development Facility (ISPF/PDF), you can use a series of ISPF panels to pass parameter values to the CLIST. The CLIST uses these values to tailor the installation and migration jobs.

DB2 also provides a set of sample programs and procedures that help you determine whether DB2 is functioning correctly. Because it is distributed as object code, DB2 requires few assemblies. An assembly to specify DB2 initialization parameters must be performed but requires only a few seconds.

DB2 allows you to specify many subsystem characteristics during DB2 operation, thereby providing the ability to defer decisions about DB2 characteristics until after the install or migration. Some of these decisions include authorizing users, defining databases and tables, and tuning DB2.

DB2 allows for updating most of the install options without requiring you to reinstall or remigrate. Defaults can be accepted for certain options; after acquiring experience with DB2, you can tailor them to the needs of a particular environment.

New with version 8 is the Managed System Infrastructure for Setup (msys for Setup), which is a z/OS tool that addresses the difficulties in maintaining z/OS. It can help manage all installation and customization tasks for DB2. It can be used with the DB2 Customization Center to update DB2 parameters and generate output that can be used as input to the installation CLIST.

High-Level Overview

The following procedures need to be performed for both installing and migrating DB2.

1. If using distributed data, install Virtual Telecommunication Access Method (VTAM) and, optionally, TCP/IP.
2. If planning implementation in a data sharing parallel sysplex environment, examine the additional considerations for the parallel sysplex installation before installing or migrating DB2 to data sharing.
3. Load the DB2 libraries (do the SMP/E steps).
4. Execute the additional jobs if you plan to use DB2's call level interface (CLI) or DB2 for z/OS Java Edition. (Refer to the *IBM DB2 ODBC Guide and Reference* or the *Application Programming Guide and Reference for JAVA*.)
5. Customize the installation or migration jobs.
6. Install or migrate DB2.
7. Connect the DB2 attachment facilities.
8. Prepare DB2 for use.
9. Verify installation or migration.

Version 8 Migration Considerations

Version 8 migration is different from all previous DB2 release migrations. You can upgrade only to V8 from V7. Also, the migration process has three modes.

1. **In compatibility (COMPAT) mode (CM)**, you install the new DB2 software and do some catalog updates, but no new functions are available to customers when you start DB2. Fallback to prior releases is still possible during this mode. DB2 members in a data sharing group can be migrated one by one to this mode (mix V7/V8).
2. **In enable new-function mode (ENFM)**, you make extensive changes to the catalog. For the most part, fallback to version 7 is impossible unless a complete restore of the subsystem is performed. To begin this mode, all DB2 members in data sharing have to running in be V8 COMPAT mode.
3. **In new-function mode (NFM)**, you are fully on DB2 V8 and have all new features. Fallback to version 7 is impossible. All DB2 data sharing members have to be in ENFM mode before you can migrate them.

The three modes of migration provide

- **Reduced risk.** During CM, customers can do extensive testing of their existing applications; if fallback needs to occur from CM to V7, you guarantee that no new features are used.

- **More control over doing the migration.** Fallback is allowed without users experiencing too many errors (force the fallback SPE).
- **Customer control of new-function usage.** Full control is provided over the migration process (timing) and when new functions can be used.

Compatibility Mode

In compatibility mode, you will run the `MIGRATE` CLIST to generate jobs, with the fallback SPE applied to all members in the data sharing group. Then start DB2 with V8 software, and run the DSNTIJTC job. Many updates are made during this mode to header pages in the directory, Boot Strap Data Set (BSCS), and Shared Communication Area (SCA). Coded Character Set Identifier (CCSID) updates are also performed at this time.

At this point, DB2 uses V8 code internally, 64-bit addressing (only virtual pools), and Unicode for parsing SQL. The V8 catalog is still in Extended Binary Coded decimal interchange format, and full fallback to V7 is allowed. To prevent `BIND` failures if someone tries to use new V8 features, you can use precompiler option `NEWFUN=NO`. While running in this mode, users have virtually no new V8 functions available.

The data sharing group permits members to be a mix of V7 and V8 in CM. However, it is best to keep the coexistence period short.

Enable New-Function Mode

Only a few new functions are available in this mode. During ENFM, a restructuring of the DB2 catalog is performed. The DSNTIJNE reorganizes 17 catalog table spaces and one directory table with extra functions (`CATENFM`). In a data sharing environment, this is a groupwide event because one catalog is shared by all members. It is recommended that this migration period be kept short for all members in the data sharing group.

NOTE If you do not have BP8K0, BP16K0, and BP32K buffer pools allocated, it is best to create them in compatibility mode.

NOTE At this point, fallback to version 7 is not possible.

During this mode (`CATEFNM`), many catalog changes are made, such as changing almost every `CHAR(8)` and `CHAR(18)` column to `VARCHAR(18)` and `VARCHAR(128)` so that DB2 can accept long names, and changing the catalog to use Unicode.

NOTE Unicode ordering is different from EBCDIC.

An online REORG of most of the catalog and one directory table space must also occur during this mode. The DSNTIJNE job is then executed in several steps, which include the following actions:

- ENFM0000. Terminate any pending DSNENFM.* utilities.
- ENFM0001. Update catalog for new mode (for every table space(18)).
- ENFMnnnn0. Test for conversion, do DDL Alters (* exclusive locking *).
- ENFM00001. Delete old data sets.
- ENFM00003. Define new data sets.
- ENFM00007. REORG and fix. Change record formats, page sizes, and encoding to Unicode; create new V8 indexes; inline image copy; and switch.
- ENFM0009. Delete old data sets.

New-Function Mode

At this time, the subsystem is a true V8 system, with all functions available. The DSNTIJNF job signals the end of ENFM (CATENFM COMPLETE), DSNTIJNG builds a new DSNHDECP module (specifies whether to use new functions), and DSNHDECM specifies NEWFUN=YES. The new V8 IVP sample jobs can be run, including all the new functions.

In order to see the mode that DB2 is in during the migration process, you can use the -DISPLAY GROUP command. The attribute for the MODE will give you this information. Values in the MODE field are as follows:

- C = compatibility
- E = enabling new function
- N = new function

The following output shows the result of a -DISPLAY GROUP command showing new-function mode:

```
DSN7100I  +DSN8 DSN7GCMD
 *** BEGIN DISPLAY OF GROUP(........) GROUP LEVEL(...) MODE(N)
                   PROTOCOL LEVEL(2)  GROUP ATTACH NAME(....)
 -------------------------------------------------------------------
 DB2                                       DB2 SYSTEM    IRLM
 MEMBER   ID  SUBSYS CMDPREF   STATUS   LVL NAME      SUBSYS IRLMPROC
 -------- --- ----   --------  -------- --- --------  ----   --------
 ........   0 DSN8   +DSN8     ACTIVE   810 P390      BRLM   BRLMPROC
 -------------------------------------------------------------------
 *** END DISPLAY OF GROUP(........)
 DSN9022I  +DSN8 DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION
```

SYSTEM PARAMETERS

DSNZPARMs

At installation time you supply values for the DB2 DSNZPARM parameters via the install panels. Table 2-1 lists all the DSNZPARMs, along with a description of each, the allowable values, and whether they can be changed online.

Table 2-1 DSNZPARMs

Parameter	Description	Acceptable Values (defaults are in boldface)	Online Updatable
ABEXP	EXPLAIN processing	**YES**, NO	Yes
ABIND	Autobind	**YES**, NO	Yes
ACCUMACC	DDF/RRSAF accumulated data	NO, 2–65,535 **10**	Yes
AGCCSID	ASCII-coded character set (graphic)	**0**–65,533	—
ALCUNIT	Allocation units	**BLK**, TRK, CYL	Yes
ALL/dbname	Start names	**ALL**, space names	—
AMCCSID	ASCII-coded character set (mixed)	**0**–65,533	—
APPENSCH	Application encoding	ASCII, **EBCDIC**, Unicode, CCSID	—
ARCPFX1	Copy 1 prefix	1–34 characters	Yes
ARCPFX2	Copy 2 prefix	1–34 characters	Yes
ARCRETN	Retention period	0–**9,999**	Yes
ARCWRTC	WTOR route code	1–16 **1,3,4**	Yes
ARCWTOR	Write to operator	NO**, YES**	Yes
ARC2FRST	Read Copy 2 archive	**NO,** YES	Yes
ASCCSID	ASCII-coded character set (single byte)	**0**–65,533	—
ASSIST	Assistant	**YES**, NO	No
AUDITST	Audit trace	**NO,** YES, list, *	No
AUTH	Use protection	**YES**, NO	No

continues

Table 2-1 DSNZPARMs (Continued)

Parameter	Description	Acceptable Values (defaults are in boldface)	Online Updatable
AUTHCACH	Plan authorization cache	0–4,096 **3,072**	Yes
BACKODUR	Backout duration	0–255 **5**	No
BINDNV	Bind new package	**BINDADD**, BIND	Yes
BLKSIZE	Block size	8,192–28,672 **24,576**	Yes
BMPTOUT	IMS BMP Timeout	1–254 **4**	Yes
CACHEDYN	Cache dynamic SQL	NO, **YES**	Yes
CACHEPAC	Package authorization cache	0–2M **32K**	No
CACHERAC	Routine authorization cache	0–2M **32K**	No
CATALOG	Catalog alias	1–8 characters **DSNCAT**	Yes
CDSSRDEF	Current degree	**1**, ANY	Yes
CHARSET	CCSID used	**ALPHANUM**, KATAKANA (if SCCSID = 930 or 5,026)	—
CHKFREQ	Checkpoint frequency	200–16M records (**500K**), or 1–60 minutes	Yes
CHGDC	Drop support	**1**,2,3	Yes
CMTSTAT	DDF threads	ACTIVE, **INACTIVE**	No
COMPACT	Compact data	**NO**, YES	Yes
COMPAT	IBM service	OFF	—
CONDBAT	Maximum remote connected	0–25,000 **10,000**	Yes
CONTSTOR	Contract thread storage	**NO**, YES	Yes
COORDNTR	Coordinator	**NO**, YES	No
CTHREAD	Maximum users	1–2,000 **200**	Yes
DBACRVW	DBADM can create view for other authid	YES, **NO**	Yes
DBPROTCL	Database protocol	**DRDA**, Private	Yes
DATE	Date format	**ISO,** USA, EUR, JIS, LOCAL	—
DATELEN	Local date length	**0**, 10 –254	—
DDF	DDF start-up option	**NO**, AUTO, COMMAND	No
DEALLCT	Deallocate period	**0**–1,439 minutes, 0–59 seconds, NOLIMIT	Yes

Table 2-1 DSNZPARMs (Continued)

Parameter	Description	Acceptable Values (defaults are in boldface)	Online Updatable
DECARTH	Decimal arithmetic	DEC15, DEC31, 15, 31	—
DECDIV3	Minimum divide scale	**NO**, YES	No
DECIMAL	Decimal point	**,** .	—
DEFLANG	Language default	ASM, C, CPP, COBOL, COB2, **IBMCOB**, FORTRAN, PL1	—
DEFLTID	Unknown authid	**IBMUSER**, or authid	No
DELIM	String delimiter	**DEFAULT**, “, ‘	—
DESCSTAT	Describe for static	NO, **YES**	Yes
DISABSCL	SQLWARN1 and 5 for nonscrollable cursors	**NO**, YES	—
DLDFREQ	Level ID update frequency	0–32,767 **5**	Yes
DLITOUT	DL/I batch timeout	1–254 **6**	Yes
DSHARE	Data sharing	**YES**, NO, blank	No
DSMAX	Data set maximum	1–**100,000**	Yes
DSQLDELI	Dist SQL string delimiter	**‘**, ‘’	—
DSSTIME	Data set stats time	1–1,440 **5**	Yes
DSCVI	Vary DS control interval	**YES,** NO	—
DYNRULES	Use for dynamic rules	**YES**, NO	—
EDMBFIT	Algorithm for free chain search	YES, **NO**	Yes
EDMDBDC	EDM DBD cache	**5,000**–2,097,152K	Yes
EDMPOOL	EDMPOOL storage size	1K–2,097,152K **32,768**	Yes
EDMSTMTC	EDM statement cache size	0–1,048,576K **5,000**	Yes
EDPROP	Drop support	**1**, 2, 3	Yes
ENSCHEME	Default encoding scheme	**EBCDIC**, ASCII	—
EVALUNC	Predicate evaluation with UR and RS	YES, **NO**	Yes
EXTRAREQ	Extra blocks requester	0–**100**	Yes
EXTRASRV	Extra blocks server	0–**100**	Yes
EXTSEC	Extended security	NO, **YES**	Yes

continues

Table 2-1 DSNZPARMs (Continued)

Parameter	Description	Acceptable Values (defaults are in boldface)	Online Updatable
GCCSID	EBCDIC coded character set (graphic byte)	0–65,533	—
GRPNAME	Group name	1–8 characters **DSNCAT**	No
HOPAUTH	Authorization at hop site	**BOTH**, RUNNER	No
IDBACK	Maximum batch connect	1–2,000 **50**	Yes
IDFORE	Maximum TSO connect	1–2,000 **50**	Yes
IDTHTOIN	Idle thread timeout	0–9,999 **120**	Yes
IDXBPOOL	Default buffer pool for user indexes	**BP0**–BP*x*	Yes
IMMEDWRI	Immediate write	NO, YES, **PH1**	Yes
IRLMAUT	Autostart	**YES**, NO	No
IRLMPRC	Procedure name	**IRLMPROC**, IRLM proc name	No
IRLMRWT	Resource timeout	1–3,600 **60**	No
IRLMSID	Subsystem name	**IRLM**, IRLM name	No
IRLMSWT	Time to autostart	1–3,600	Yes
IXQTY	Index space default size	**0**–4,194,304	Yes
LBACKOUT	Postpone backward log processing	**AUTO**, YES, NO	No
LC_CTYPE	Locale LC_CTYPE	Valid locale 0–50 characters	—
LEMAX	Maximum LE Tokens	0–50, **20**	No
LOBVALA	User LOB value storage	1–2,097,152 **10,240**	Yes
LOBVALS	User LOB value storage	1–510,002 **2,048**	Yes
LOGAPSTG	Log apply storage	1–100M **100**	No
LRDRTHLD	Long-running reader threshhold	**0**–1,439 minutes	Yes
MAINTYPE	Current maintenance types for MQTs	NONE, **SYSTEM**, USER, ALL	Yes
MAXARCH	Recording maximum	10–**1,000**	No
MAXDBAT	Maximum remote active	0–1,999 **200**	Yes
MAX_NUM_CUR	Maximum open cursors	0–99,999 **500**	Yes

Table 2-1 DSNZPARMs (Continued)

Parameter	Description	Acceptable Values (defaults are in boldface)	Online Updatable
MAXKEEPD	Maximum kept dynamic statements	0–65,535 **5000**	Yes
MAXRBLK	RID pool size	0, 16K–1,000,000K **calculated**	Yes
MAXRTU	Read tape units	1–99 **2**	Yes
MAX_ST_PROC	Maximum number of stored procedures	0–99,999 **2,000**	Yes
MAXTYPE1	Maximum Type 1 inactive	**0**–MAX REMOTE CON	Yes
MCCSID	EBCDIC coded character set (mixed byte)	**0**–65,533	—
MEMBNAME	Member name	1–8 characters **DSN1**	No
MGEXTSZ	Optimize extent sizing	YES, **NO**	Yes
MINDVSCL	Minimum scale for decimal division	**NONE**, 3, 6	—
MINRBLK	Number of RID lists for each RID map	**1**, *n*	—
MINSTOR	Thread management	YES, **NO**	Yes
MIXED	Mixed data	**NO**, YES	—
MON	Monitor trace	**NO**, YES	No
MONSIZE	Monitor size	**8K** to 1M	No
NPGTHRSH	Use of index after table growth	**0**, –1, *n*	Yes
NUMLKTS	Locks per table space	0–50,000 **1,000**	Yes
NUMLKUS	Locks per user	0–100,000 **10,000**	Yes
OFFLOAD	Offload active logs online	NO**, YES**	—
OJPERFEH	Disable outer join performance enhancements		Yes
OPTPREF		**ON,** OFF	—
OPTHINTS	Optimization hints	**NO**, YES	Yes
OUTBUFF	Output buffer	40K–400MB **400K**	No
PADIX	Pad index by default	YES**, NO**	Yes
PADNTSTR	Pad null-terminated strings	YES**, NO**	Yes

continues

Table 2-1 DSNZPARMs (Continued)

Parameter	Description	Acceptable Values (defaults are in boldface)	Online Updatable
PARAMDEG	Degree of parallelism	**0**–no upper limit	Yes
PARTKEYU	Allow partitioning keys to be updated	**YES**, NO, or same	Yes
PCLOSEN	Read-only switch checkpoints	1–32,767 **5**	Yes
PCLOSET	Read-only switch time	1–32,767 **10**	Yes
POOLINAC	Pool thread timeout	0–9,999 **120**	Yes
PRIQTY	Primary quantity	**Blank**, 1–9,999,999	Yes
PROTECT	Archive logs protected with RACF	**NO**, YES	Yes
PTASKROL	Include accounting traces for parallel tasks	**YES**, NO	Yes
QUIESCE	Quiesce period	0–999 **5**	Yes
RECALL	Recall database	**YES**, NO	No
RECALLD	Recall delay	0–32,767 **120**	Yes
REFSHAGE	Current refresh age	**0,** ANY	Yes
RELCURHL	Release locks	**YES**, NO	Yes
RESTART/DEFR	Restart or defer	**RESTART**, DEFER	—
RESYNC	Resync interval	1**, 2** –99	Yes
RETLWAIT	Retained lock timeout	**0**–254	Yes
RETVLCFK	Varchar from index	**NO**, YES	Yes
RGFCOLID	Registration owner	1–8 characters **DSNRGCOL**	No
RGFDBNAM	Registration database	1–8 characters **DSNRGFDB**	No
RGFDEDPL	Control all applications	**NO**, YES	No
RGFDEFLT	Unregistered DDL default	APPL**, ACCEPT**, REJECT	No
RGFESCP	ART/ORT escape character	Nonalphanumeric character	No
RGFFULLQ	Require full names	**YES**, NO	No
RGFINSTL	Install DD control support	**NO**, YES	No
RGFNMORT	OBJT registration table	1–17 characters **DSN_REGISTER_OBJT**	No

Table 2-1 DSNZPARMs (Continued)

Parameter	Description	Acceptable Values (defaults are in boldface)	Online Updatable
RGFNMPRT	APPL registration table	1–17 characters **DSN_REGISTER_APPL**	No
RLF	RLF autostart	**NO**, YES	No
RLFAUTH	Resource Authid	**SYSIBM**, or authid	Yes
RLFERR	RLST access error	**NOLIMIT**, NORUN,1–50,000,000	Yes
RLFERRD	RLST access error	**NOLIMIT**, NORUN,1–50,000,000	Yes
RLFTBL	RLST name suffix	**01**, 2 alphanumeric characters	Yes
ROUTCDE	WTO route codes	1, 1–14 route codes	No
RRULOCK	U lock for RR/RS	**NO**, YES	Yes
SCCSID	EBCDIC coded character set (single byte)	**0**–65,533	—
SECQTY	Secondary quantity	Blank, CLIST calculated, 1–9,999,999	Yes
SECLCACH			
SEQCACH	Sequential cache	**BYPASS**, SEQ	Yes
SEQPRES	Utility cache option	**NO**, YES	Yes
SITETYP	Site type	**LOCALSITE**, RECOVERYSITE	No
SJMXPOOL	Star join maximum pool	0–1,024 **20**	Yes
SJTABLES	Number of tables in star join		Yes
SMFACCT	SMF accounting	NO, **YES(1)**, list (1–5,7,8), *	No
SMFSTAT	SMF statistics	**YES (1,3,4)**, NO, list(1–5) , *	No
SMSDCFL	SMS data class for file table space	**Blank**, 1–8 characters	Yes
SMSDCIX	SMS data class for index table space	**Blank**, 1–8 characters	Yes
SPRMEDX			Yes
SPRMLTD	Size threshhold for compression	**10**	—
SQLDELI	SQL string delimiter	**Default**, ‘, “	—
SRTPOOL	Sort pool size	240K–64,000K	Yes

continues

Table 2-1 DSNZPARMs (Continued)

Parameter	Description	Acceptable Values (defaults are in boldface)	Online Updatable
SSID	Subsystem name	**DSN**, *SSID*	—
STARJOIN	Enabling star join	**Disable**, enable, 1, 2–32,768	Yes
STATHIST	Collect historical statistics	SPACE, **NONE**, ALL, ACCESSPATH	Yes
STATSINT	Time to write RTS stats	1–1,440 minutes **30**	Yes
STATROLL	Run stats aggregates partition-level statistics	YES, **NO**	Yes
STATIME	Statistics time	1–1,440 minutes **30**	Yes
STDSQL	Standard SQL language	**NO**, YES	—
STORMXAB	Maximum abend count	**0**–225	Yes
STORPROC	DB2 procedure name	1–8 characters **ssnmSPAS**	No
STORTIME	Timeout value	5–1,800 seconds **180**	Yes
SUPERRS	Suppress logrec recording during soft errors	**YES**, NO	Yes
SVOLARC	Single volume	YES**, NO**	Yes
SYNCVAL	Statistics sync	**NO**, 0–59	Yes
SYSADM	System administrator 1	**SYSADM** or authid	Yes
SYSADM2	System administrator 2	**SYSADM** or authid	Yes
SYSOPR1	System operator 1	**SYSOPR** or authid	Yes
SYSOPR2	System operator 2	**SYSOPR** or authid	Yes
TBSBPOOL	Default buffer pool for user data	**BP0** – BPx	Yes
TCPALVER	TCP/IP already verified	**NO**, YES	Yes
TCPKPALV	TCP/IP keep alive	ENABLE, DISABLE, 1–65,524 **120**	Yes
TIME	Time format	**ISO**, JIS, USA, EUR, LOCAL	—
TIMELEN	Local time length	0, 8–254	—
TRACLOC	Size of local trace table	**16** (4K bytes)	
TRACSTR	Trace autostart	NO, YES (1–3), list (1–9)	No
TRACTBL	Trace size	4–396K **64K**	No

Table 2-1 DSNZPARMs (Continued)

Parameter	Description	Acceptable Values (defaults are in boldface)	Online Updatable
TRKRSITE	Remote tracker site usage	**NO**, YES	No
TSQTY	Define allocation for table space	**0**–4,194,304	Yes
TSTAMP	Timestamp archives	**NO**, YES	Yes
TWOACTV	Number of active copies	**2**, 1	No
TWOARCH	Number of archive copies	**2**, 1	No
TWOBSDS	Number of BSDSs	**YES**, NO	—
UGCCSID	Unicode CCSID (graphic)	1,208	—
UIFCIDS	Unicode IFCIDS	YES, **NO**	Yes
UMCCSID	Unicode CCSID (mixed)	1,208	—
UNIT	Device Type 1	**TAPE** or any device	Yes
UNIT2	Device Type 2	Device or unit name	Yes
URCHKTH	UR check frequency	0–255	Yes
URLGWTH	UR log write check	0**–1,000K**	Yes
USCCSID	Unicode CCSID (single byte)	1,208	—
UTIMOUT	Utility timeout	1–254	Yes
VOLTDEVT	Temporary unit name	**SYSDA,** valid name	Yes
WLMENV	WLM environment	Valid name, 1–18 characters	Yes
XLKUPDT	X Lock for searched U/D	YES**, NO**	Yes

After installation, these DSNZPARMs can be changed without your having to go back through the installation process. You can do this by changing the DSNTIJUZ member and reassembling and bouncing the DB2 subsystem, or performing an online change, for them to take effect.

Online DSNZPARMs

DB2 also allows for about 123 of the most popular DSNZPARMs to be dynamically changed without your having to stop and restart the DB2 subsystem and thereby cause an unwanted outage. The advantage to changing DSNZPARMs dynamically is to be able to tailor parameters to the current workload. For example, an approved change to the Environmental Descriptor Manager pool size can be implemented sooner without a DB2 outage. It may also be desirable to

change not only buffer pool size but also EDM pool size and checkpoint frequency for overnight batch processing.

The DSNZPARM member is changed dynamically in its entirety by activating a different DSNZPARM member. The `-SET SYSPARM` command is used from a z/OS console, a DSN session under TSO, a DB2I panel, a CICS or IMS terminal or via an application or product using the Instrumentation Facility. The issuer must have SYSADM, SYSCTRL, or SYSOPR authority; for more on these authorities, refer to Chapter 3. The following forms of the `SET SYSPARM` statement can be used for controlling dynamic DSNZPARM settings:

```
-SET SYSPARM LOAD (modname)
Loads the named parameter module
Default is DSNZPARM

-SET SYSPARM RELOAD
Last named subsystem parameter module is loaded into storage

-SET SYSPARM STARTUP
Loads the initial parameters from DB2 startup
```

COMMANDS

Commands in the DB2 environment are divided into the following categories:

- The DSN command and its subcommands
- DB2 commands
- IMS commands
- CICS Attachment Facility Commands
- MVS IRLM commands
- TSO CLISTs

This chapter considers only the DSN commands and the DB2 commands.

DSN Commands

DSN, the DB2 command processor, executes as a TSO command. All subcommands except SPUFI run under DSN in either the foreground or the background. All subcommands except `END` also run under DB2 Interactive (DB2I). SPUFI runs only in the foreground under ISPF. The DSN commands/subcommands and their functions are listed in Table 2-2.

Table 2-2 DSN Commands/Subcommands

Command/Subcommand	Function
ABEND	Causes the DSN session to terminate with an X'04E'
BIND	Builds an application package or plan
DB2 command	Executes a DB2 command
DCLGEN (declarations generator)	Produces declarations for tables or views
DSN	Starts a DSN session
END	Ends a DSN session
FREE	Deletes an application package or plan
REBIND	Updates an application package or plan
REBIND TRIGGER PACKAGE	Updates an application trigger package
RUN	Executes an application program
SPUFI	Executes the SQL processor using file input
*	Comment

DB2 Commands

The command START DB2 can be issued only from a z/OS console or Authorized Program Facility (APF) passing it to the console. All other DB2 commands can be issued from the following:

- APF authorized program
- CICS terminal
- DB2I panel
- IFI application program
- IMS terminal
- TSO terminal session
- z/OS console or z/OS application program

An application program may use the DB2 Instrumentation Facility Interface to issue DB2 commands. DB2 commands issued from an MVS console are not associated with any secondary authorization IDs. The DB2 commands and their functions are listed in Table 2-3.

Table 2-3 DB2 Commands

Command	Function
-ALTER BUFFERPOOL	Alters attributes for the buffer pools
-ALTER GROUPBUFFERPOOL	Alters attributes for the group buffer pools
-ALTER UTILITY	Alters parameter values of the REORG utility
-ARCHIVE LOG	Enables a site to close a current active log and open the next available log data set
-CANCEL THREAD	Cancels processing for specific local or distributed threads
-DISPLAY ARCHIVE	Displays information about archive-log processing
-DISPLAY BUFFERPOOL	Displays information about the buffer pools
-DISPLAY DATABASE	Displays status information about DB2 databases
-DISPLAY DDF	Displays information regarding the status and configuration of the DDF
-DISPLAY FUNCTION SPECIFIC	Displays statistics about external user-defined functions
-DISPLAY GROUP	Displays information about the data sharing group to which a DB2 subsystem belongs and which mode DB2 is operating in
-DISPLAY GROUPBUFFERPOOL	Displays status information about DB2 group buffer pools
-DISPLAY LOCATION	Displays status information about distributed threads
-DISPLAY LOG	Displays log information and status of the offload task
-DISPLAYPROCEDURE	Displays status information about stored procedures
-DISPLAY RLIMIT	Displays status information about the resource limit facility (governor)
-DISPLAY THREAD	Displays information about DB2 threads
-DISPLAY TRACE	Displays information about DB2 traces
-DISPLAY UTILITY	Displays status information about a DB2 utility
-MODIFY TRACE	Changes the trace events associated with a particular active trace
-RECOVER BSDS	Reestablishes dual-bootstrap data sets
-RECOVER INDOUBT	Recovers threads left in doubt
-RECOVER POSTPONED	Completes backout processing for units of recovery left incomplete during an earlier restart
-RESET GENERICLU	Purges information stored by VTAM in the coupling facility

Table 2-3 DB2 Commands (Continued)

Command	Function
-RESET INDOUBT	Purges information displayed in the in-doubt thread report generated by the -DISPLAY THREAD command
-SET ARCHIVE	Controls the allocation of tape units and the deallocation time of the tape units for archive-log processing
-SET LOG	Modifies the checkpoint frequency
-SET SYSPARM	Loads the subsystem parameters specified in the command
-START DATABASE	Makes the specified database available for use
-START DB2	Initializes the DB2 subsystem; can be issued only from an MVS console
-START DDF	Starts the Distributed Data Facility
-START FUNCTION SPECIFIC	Activates a stopped external function
-START PROCEDURE	Activates the definition of stopped or cached stored procedures
-START RLIMIT	Starts the resource limit facility (governor)
-START TRACE	Initiates DB2 trace activity 300
-STOP DATABASE	Makes specified databases unavailable for applications
-STOP DB2	Stops the DB2 subsystem
-STOP DDF	Stops the Distributed Data Facility
-STOP FUNCTION SPECIFIC	Stops the acceptance of SQL statements for specified functions
-STOP PROCEDURE	Stops the acceptance of SQL CALL statements for stored procedures
-STOP RLIMIT	Stops the resource limit facility (governor)
-STOP TRACE	Stops trace activity
-TERM UTILITY	Terminates execution of a utility

A DSN9022I message indicates the normal end of DB2 command processing. A DSN9023I message indicates the abnormal end of DB2 command processing.

UTILITIES

DB2 utilities either execute within the subsystem (online) or standalone and run outside the subsystem (offline). The activities performed by the utilities—loading and reorganizing data, recovering data and indexes, rebuilding indexes, gathering statistics, quiescing data, and repairing data—are discussed in detail in Chapters 7 and 8.

Executing Utilities

The most common way to execute the DB2 utilities is to create JCL with the appropriate control cards. The utilities must be executed on the subsystem where the objects reside. (Chapter 9 provides more information on how to do so in a data sharing environment.) However, other options for utility execution are available, as discussed next.

DB2I

The DB2I has a panel that can be used to generated and run utilities. This option does not require a great deal of JCL knowledge, and the jobs can be saved for future execution and editing. Figure 2-8 gives an example of the DB2I panel for DB2 utilities.

```
DB2 UTILITIES                    SSID: DSN
===>

Select from the following:

 1 FUNCTION ===> EDITJCL            (SUBMIT job, EDITJCL, DISPLAY, TERMINATE)
 2 JOB ID   ===> TEMP               (A unique job identifier string)
 3 UTILITY  ===>                    (CHECK DATA, CHECK INDEX, CHECK LOB,
                                     COPY, DIAGNOSE, LOAD, MERGE, MODIFY,
                                     QUIESCE, REBUILD, RECOVER, REORG INDEX,
                                     REORG LOB, REORG TABLESPACE, REPORT,
                                     REPAIR, RUNSTATS, STOSPACE, UNLOAD)
 4 STATEMENT DATA SET ===> UTIL

Specify restart or preview option, otherwise enter NO.

 5 RESTART  ===> NO                (NO, CURRENT, PHASE or PREVIEW)

 6 LISTDEF? (YES|NO) ===>           TEMPLATE? (YES|NO) ===>

 * The data set names panel will be displayed when required by a utility.

PRESS:  ENTER to process     END to exit    HELP for more information
```

Figure 2-8 DB2 utilities panel

DSNU Command

A DB2 online utility can be executed by invoking the `DSNU CLIST` command under TSO. The `CLIST` command generates the JCL data set required to execute the DSNUPROC procedure and execute online utilities as batch jobs. When you use the `CLIST` command, you need not be concerned about details of the JCL data set.

The `CLIST` command will create a job to perform one utility operation. The command can be issued for each utility operation necessary; then the issuer can edit and merge the outputs into one job or step.

Control Center

Utility execution is also supported in the DB2 Control Center, which also supports utility wildcarding: the ability to execute utilities against a list of objects matching a specified pattern of matching characters. A utility procedure could be created that would permit running, with one command, a mixture of several utilities against several objects, making maintenance much easier for the DBA. The Control Center also supports restarting utilities from the last committed phase or the last committed point. This support is available only for utilities started in the Control Center. The restart is accessible through the Display Utility dialog.

Support is also available for utility IDs. The Tools Settings notebook has an option to create a utility ID template using a variety of variables, such as `USERID` and `UTILNAME`. Before execution of the utility, the utility ID can be edited to make it more meaningful. The execution of utilities from the Control Center requires the DSNUTILS stored procedure described below.

DSNUTILS and DSNUTILU

DSNUTILS is a DB2-supplied stored procedure for executing utilities from a local or remote application via an SQL `CALL` statement. The client application calls in DSNUTILS with appropriate parameters. DSNUTILS then analyzes them to create a SYSIN stream and allocate all necessary data sets. After the data sets are allocated, DSNUTILS calls DSNUTILB, which then executes the appropriate utility. The utility statements are then processed, and DSNUTILS retrieves the data (execution results) in the SYSPRINT file, puts the data in the `SYSIBM.SYSPRINT` temporary table, and then opens a cursor on the table and passes control back to the client application. The client application then fetches all rows from the result set. Figure 2-9 shows this flow.

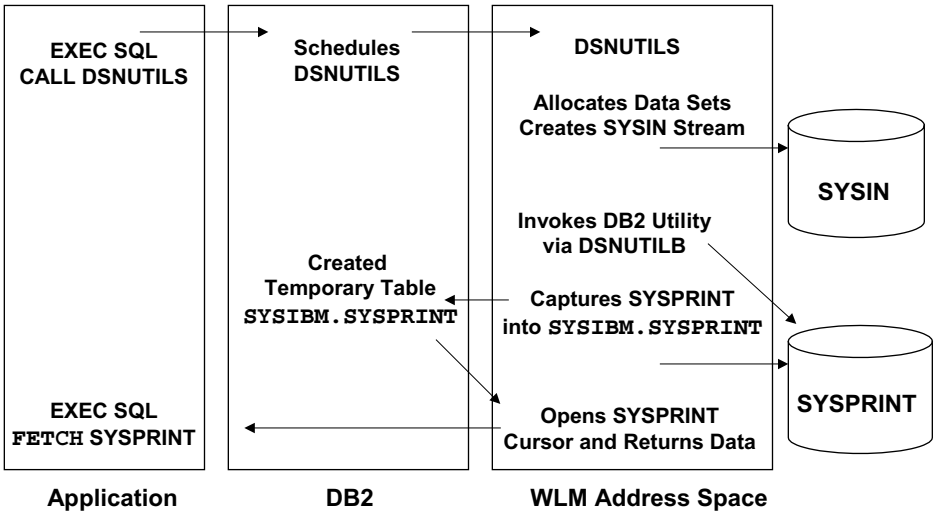

Figure 2-9 Execution of DSNUTILS

DB2 for z/OS Version 8 also includes the new DSNUTILU stored procedure which provides the same functions as DSNUTILS, but allows the control cards to be specified in Unicode (UTF-8).

Chapter 13 has more information on stored procedures.

Utility Templates

Some DB2 utilities produce data sets as a by-product or as an end result of utility execution. These data sets are referenced on utility control statements by a set of data definition (DD) name keywords and are specified in detail on the corresponding JCL DD cards.

These DD cards must be coded for each utility, as required, and maintained over time as the structure of data changes. Database administrators establish data set policies, refer to them on utility control statements, and let DB2 utilities administer those policies at execution time. Many DB2 utilities will accept a *template* construct instead of DD cards to dynamically allocate utility data sets.

Templates contain

- The data set naming convention
- DFSMS parameters
- DASD or TAPE allocation parameters

A TEMPLATE can be specified in the SYSIN data set, preceding the utility control statement that references it, or in one or more TEMPLATE data sets. The TEMPLATE data set DD name is specified on the OPTIONS utility control statement by TEMPLATEDD(*ddname*) and applies to all subsequent utility control statements until the end of input or until DB2 encounters a new OPTIONS TEMPLATEDD(*ddname*) specification. The default TEMPLATE data set DD name is SYSTEMPL.

TEMPLATE data sets may contain only TEMPLATE utility control statements. Any TEMPLATE defined within SYSIN overrides another TEMPLATE definition of the same name found in a TEMPLATE data set.

With this functionality, database administrators can standardize data set allocation and the utility control statements that refer to those data sets. This functionality reduces the need to customize and alter utility job streams.

NOTE Templates cannot be used with the REPAIR utility.

The TEMPLATE specification can be used for both DASD and TAPE data set allocation, including support for data set stacking on tape and Generation Data Group (GDG) base definition. TEMPLATE syntax also allows user-specified DASD SPACE parameters. If the SPACE keyword

is not specified, the size of the data set will be estimated on the basis of formulas that vary by utility and by data set.

The TEMPLATE statement required for a COPY example might look something like this:

```
TEMPLATE TMP1
DSNAME(DB2.&TS..D&JDATE..COPY&ICTYPE.&LOCREM.&PRIBAK.)
VOLUMES(Vol1,Vol2,Vol3)
TEMPLATE TMP2
DSNAME(DB2.&TS..D&JDATE..COPY&ICTYPE.&LOCREM.&PRIBAK.)
VOLUMES(Vol4,Vol5,Vol6)
LISTDEF PAYROLL INCLUDE TABLESPACE CERTTS.*
INCLUDE INDEXSPACE CERTTS.*IX
EXCLUDE TABLESPACE CERTTS.TEMP*
EXCLUDE INDEXSPACE CERTTS.TMPIX*
COPY LIST PAYROLL ...COPYDDN(TMP1,TMP1)RECOVERYDDN(TMP2,TMP2)
```

Database administrators can check their utility control statements without execution, using the PREVIEW function. In PREVIEW mode, DB2 expands all TEMPLATE data set names appearing in the SYSIN DD, as well as any from the TEMPLATE DD that are referenced on a utility control statement. DB2 then prints the information to the SYSPRINT data set to halt execution. You can specify PREVIEW either as a JCL PARM or on the OPTIONS PREVIEW utility control statement.

For more information on templates and the DB2 utilities, refer to the *IBM DB2 UDB for z/OS Version 8 Utility Guide and Reference*.

Displaying Utilities

Utilities can be displayed in order to see important runtime information about the jobs. This includes jobs running in a data sharing group. The output has information about the

- Type of utility
- How much of the processing the utility has completed
- Status of the utility
- Member on which the utility is executing
- Phase the utility is in

CATALOG AND DIRECTORY

The DB2 catalog and directory act as central repositories for all information about the support and operations of DB2 objects, authorizations, and communications. The catalog comprises several DB2 tables and can be accessed via SQL. The catalog contains details about DB2 objects obtained from the DDL (Data Definition Language) when an object is created or altered or from

DCL (Data Control Language) when an authorization is granted on an object or a group of objects. The DB2 catalog also contains information about communications with other DB2 and non-DB2 databases through the use of the communications database (CDB), which contains information about VTAM and TCP/IP addresses.

Table 2-4 lists the DB2 catalog tables and the types of information in each. (Descriptions of all the columns in the DB2 catalog tables can be found in detail in Appendix D of the *IBM DB2 for z/OS Version 8 SQL Reference Manual*.)

Table 2-4 DB2 Catalog Tables

Table Name (SYSIBM.*table*)	Information Contents
IPLIST	Allows multiple IP addresses to be specified for a given LOCATION. Insert rows into this table when you want to define a remote DB2 data sharing group. Rows can be inserted, updated, and deleted.
IPNAMES	Defines the remote DRDA servers DB2 can access using TCP/IP. Rows in this table can be inserted, updated, and deleted.
LOCATIONS	Contains a row for every accessible remote server. The row associates a LOCATION name with the TCP/IP or SNA network attributes for the remote server. Requesters are not defined in this table. Rows in this table can be inserted, updated, and deleted.
LULIST	Allows multiple LU (Logical Unit) names to be specified for a given LOCATION. Insert rows into this table when you want to define a remote DB2 data sharing group. The same value for the LUNAME column cannot appear in both the SYSIBM.LUNAMES table and the SYSIBM.LULIST table. Rows in this table can be inserted, updated, and deleted.
LUMODES	Each row of the table provides VTAM with conversation limits for a specific combination of LUNAME and MODENAME. The table is accessed only during the initial conversation-limit negotiation between DB2 and a remote LU. This negotiation is called *change-number-of-sessions* (CNOS) processing. Rows in this table can be inserted, updated, and deleted.
LUNAMES	The table must contain a row for each remote SNA client or server that communicates with DB2. Rows can be inserted, updated, or deleted.
MODESELECT	Associates a mode name with any conversation created to support an outgoing SQL request. Each row represents one or more combinations of LUNAME, authorization ID, and application plan name. Rows in this table can be inserted, updated, and deleted.
SYSAUXRELS	Contains one row for each auxiliary table created for a LOB column. A base table space that is partitioned must have one auxiliary table for each partition of each LOB column.

Table 2-4 DB2 Catalog Tables (Continued)

Table Name (SYSIBM.*table*)	Information Contents
SYSCHECKDEP	Contains one row for each reference to a column in a table check constraint.
SYSCHECKS	Contains one row for each table-check constraint.
SYSCHECKS2	Contains one row for each table-check constraint created in or after version 7.
SYSCOLAUTH	Records the UPDATE or REFERENCES privileges that are held by users on individual columns of a table or view.
SYSCOLDIST	Contains one or more rows for the first key column of an index key. Rows in this table can be inserted, updated, and deleted.
SYSCOLDIST_HIST	Contains rows from SYSCOLDIST. Whenever rows are added or changed in SYSCOLDIST, the rows are also written to the new history table. Rows in this table can be inserted, updated, and deleted.
SYSCOLDISTSTATS	Contains zero or more rows per partition for the first key column of a partitioning index or DPSI (Data Partitioned Secondary Index). Rows are inserted when RUNSTATS scans index partitions of the partitioning index. No row is inserted if the index is nonpartitioning. Rows in this table can be inserted, updated, and deleted.
SYSCOLSTATS	Contains partition statistics for selected columns. For each column, a row exists for each partition in the table. Rows are inserted when RUNSTATS collects either indexed column statistics or nonindexed column statistics for a partitioned table space. No row is inserted if the table space is nonpartitioned. Rows in this table can be inserted, updated, and deleted.
SYSCOLUMNS	Contains one row for every column of each table and view.
SYSCOLUMNS_HIST	Contains rows from SYSCOLUMNS. Whenever rows are added or changed in SYSCOLUMNS, the rows are also written to the new history table. Rows in this table can be inserted, updated, and deleted.
SYSCONSTDEP	Records dependencies on check constraints or user-defined defaults for a column.
SYSCOPY	Contains information needed for recovery.
SYSDATABASE	Contains one row for each database, except for database DSNDB01.
SYSDATATYPES	Contains one row for each distinct type defined to the system.
SYSDBAUTH	Records the privileges held by users over databases.
SYSDBRM	Contains one row for each DBRM of each application plan.
SYSDUMMY1	Contains one row. The table is used for SQL statements in which a table reference is required, but the contents of the table are not important.

continues

Table 2-4 DB2 Catalog Tables (Continued)

Table Name (SYSIBM.*table*)	Information Contents
SYSFIELDS	Contains one row for every column that has a field procedure.
SYSFOREIGNKEYS	Contains one row for every column of every foreign key.
SYSINDEXES	Contains one row for every index.
SYSINDEXES_HIST	Contains rows from SYSINDEXES. Whenever rows are added or changed in SYSINDEXES, they are also written to the new history table. Rows in this table can be inserted, updated, and deleted.
SYSINDEXPART	Contains one row for each nonpartitioning index and one row for each partition of a partitioning index or a DPSI.
SYSINDEXPART_HIST	Contains rows from SYSINDEXPART. Whenever rows are added or changed in SYSINDEXPART, they are also written to the new history table. Rows in this table can be inserted, updated, and deleted.
SYSINDEXSTATS	Contains one row for each partition of a partitioning index. Rows in this table can be inserted, updated, and deleted.
SYSINDEXSTATS_HIST	Contains rows from SYSINDEXSTATS. Whenever rows are added or changed in SYSINDEXSTATS, they are also written to the new history table. Rows in this table can be inserted, updated, and deleted.
SYSJARCLASS_SOURCE	Auxiliary table for SYSIBMSYSCONTENTS.
SYSJARCONTENTS	Contains Java class source for installed JAR (Java Archive).
SYSJARDATA	Auxiliary table for SYSIBMSYSOBJECTS.
SYSJAROBJECTS	Contains binary large object representing the installed JAR.
SYSJAVAOPTS	Contains build options used during INSTALL_JAR.
SYSKEYCOLUSE	Contains a row for every column in a unique constraint—primary key or unique key—from the SYSIBM.SYSTABCONST table.
SYSKEYS	Contains one row for each column of an index key.
SYSLOBSTATS	Contains one row for each LOB table space.
SYSLOBSTATS_HIST	Contains rows from SYSLOBSTATS. Whenever rows are added or changed in SYSLOBSTATS, they are also written to the new history table. Rows in this table can be inserted, updated, and deleted.
SYSPACKAGE	Contains a row for every package.
SYSPACKAUTH	Records the privileges that users hold over packages.
SYSPACKDEP	Records the dependencies of packages on local tables, views, synonyms, table spaces, indexes and aliases, functions, and stored procedures.

Table 2-4 DB2 Catalog Tables (Continued)

Table Name (SYSIBM.*table*)	Information Contents
SYSPACKLIST	Contains one or more rows for every local application plan bound with a package list. Each row represents a unique entry in the plan's package list.
SYSPACKSTMT	Contains one or more rows for each statement in a package.
SYSPARMS	Contains one row for each parameter of a routine or multiple rows for table parameters—one for each column of the table.
SYSPKSYSTEM	Contains zero or more rows for every package. Each row for a given package represents one or more connections to an environment in which the package could be executed.
SYSPLAN	Contains one row for each application plan.
SYSPLANAUTH	Records the privileges that users hold over application plans.
SYSPLANDEP	Records the dependencies of plans on tables, views, aliases, synonyms, table spaces, indexes, functions, and stored procedures.
SYSPLSYSTEM	Contains zero or more rows for every plan. Each row for a given plan represents one or more connections to an environment in which the plan could be used.
SYSRELS	Contains one row for every referential constraint.
SYSRESAUTH	Records `CREATE IN` and `PACKADM ON` privileges for collections, `USAGE` privileges for distinct types, and `USE` privileges for buffer pools, storage groups, and table spaces.
SYSROUTINEAUTH	Records the privileges that users hold on routines. (A routine can be a user-defined function, a cast function, or a stored procedure.)
SYSROUTINES	Contains one row for every routine. (A routine can be a user-defined function, a cast function, or a stored procedure.)
SYSROUTINES_OPTS	Contains one row for each generated routine, such as one created by the DB2 Development Center tool, that records the build options for the routine. Rows in this table can be inserted, updated, and deleted.
SYSSCHEMAAUTH	Contains one or more rows for each user granted a privilege on a particular schema in the database.
SYSSEQUENCEAUTH	Records the privileges that users hold over sequences.
SYSSEQUENCES	Contains one row for each identity column.
SYSSEQUENCESDEP	Records the dependencies of identity columns on tables.
SYSSTMT	Contains one or more rows for each SQL statement of each DBRM.

continues

Table 2-4 DB2 Catalog Tables (Continued)

Table Name (SYSIBM.*table*)	Information Contents
SYSSTOGROUP	Contains one row for each storage group.
SYSSTRINGS	Contains information about character conversion. Each row describes a conversion from one coded character set to another.
SYSSYNONYMS	Contains one row for each synonym of a table or a view.
SYSTABAUTH	Records the privileges that users hold on tables and views.
SYSTABCONST	Contains one row for each unique constraint—primary key or unique key—created in DB2 for OS/390 version 7 or later.
SYSTABLEPART	Contains one row for each nonpartitioned table space and one row for each partition of a partitioned table space.
SYSTABLEPART_HIST	Contains rows from SYSTABLEPART. Rows are added or changed when RUNSTATS collects history statistics. Rows in this table can be inserted, updated, and deleted.
SYSTABLES	Contains one row for each table, view, or alias.
SYSTABLES_HIST	Contains rows from SYSTABLES. Rows are added or changed when RUNSTATS collects history statistics. Rows in this table can be inserted, updated, and deleted.
SYSTABLESPACE	Contains one row for each table space.
SYSTABSTATS	Contains one row for each partition of a partitioned table space. Rows in this table can be inserted, updated, and deleted.
SYSTABSTATS_HIST	Contains rows from SYSTABSTATS. Rows are added or changed when RUNSTATS collects history statistics. Rows in this table can be inserted, updated, and deleted.
SYSTRIGGERS	Contains one row for each trigger.
SYSUSERAUTH	Records the system privileges that users hold.
SYSVIEWDEP	Records the dependencies of views on tables, functions, and other views.
SYSVIEWS	Contains one or more rows for each view.
SYSVOLUMES	Contains one row for each volume of each storage group.
USERNAMES	Uses each row in the table to carry out one of the following operations: • Outbound ID translation • Inbound ID translation and “come from” checking Rows in this table can be inserted, updated, and deleted.

Catalog Consistency Queries

Consistency queries can be executed as part of the migration process to ensure that the data in the catalog is correct. These queries are found in the data set *prefix*.SDSNSAMP(DSNTESQ). These queries test such logical relationships as ensuring that all indexes are created on tables that exist.

The SQL statements can be executed from SPUFI or from a dynamic SQL program, such as DSNTEP2. The queries can be executed on the catalog tables or on copies of the catalog. RUNSTATS should be run on the catalog or the copies to ensure the best performance. In some cases, the queries will perform better when executed on the copies, using the extra indexes as defined for some of the tables.

Following is an example of a catalog consistency query that checks whether all table spaces belong to a defined database. This query will find all the SYSTABLESPACE databases that do not have corresponding rows in SYSDATABASE. The desired—expected—result is to have no rows returned.

```
SELECT DBNAME, NAME
   FROM SYSIBM.SYSTABLESPACE TS
   WHERE NOT EXISTS
     (SELECT *
        FROM SYSIBM.SYSDATABASE DB
        WHERE DB.NAME = TS.DBNAME);
```

DB2 Directory

The DB2 directory is used to store information about the operation and housekeeping of the DB2 environment. This directory, unlike the DB2 catalog, cannot be accessed by using SQL. The DB2 directory contains information required to start DB2; activities and utilities in the DB2 environment do the updating and deleting of table entries in the DB2 directory. The DB2 directory contains five tables: a description of each is given in Table 2-5.

DISTRIBUTED DATA

Using DB2's Distributed Data Facility (DDF) provides access to data held by other data management systems or to make your DB2 data accessible to other systems. A DB2 application program can use SQL to access data at other database management systems (DBMSs) other than the DB2 at which the application's plan is bound. This DB2 is known as the *local DB2*. The local DB2 and the other DBMSs are called *application server*s. Any application server other than the local DB2 is considered a *remote server*, and access to its data is a distributed operation.

Table 2-5 DB2 Directory Tables

Directory Table	Information Contents
SPT01	Referred to as the skeleton package table (SKPT), this table contains information about the access paths and the internal form of the SQL for a package at bind time. Entries are made into this table during bind time (`BIND PACKAGE`), and entries are deleted when a package is freed (`FREE PACKAGE`). This table is loaded into memory at execution time, along with the `SCT02` table described next.
SCT02	Referred to as the skeleton cursor table (SKCT), this table contains information about access paths and the internal form of the SQL for an application plan. Entries in this table are made when a plan is bound (`BIND PLAN`) and deleted when a plan is freed (`FREE PLAN`). This table is also loaded into memory at execution time.
DBD01	Information about DBDs (database descriptors), which are internal control blocks, is kept in this table. Each DB2 database has one DBD for its objects: table spaces, indexes, tables, referential integrity constraints, and check constraints. Updates to this table are made when a database is created or updated. This information is accessed by DB2 in place of continually using the DB2 catalog, permitting faster, more efficient access to this information. The information in the `DBD01` directory table is also contained in the DB2 catalog.
SYSLGRNX	Referred to as the log range table, this table contains information from the DB2 logs about the RBA (relative byte address) range for updates. This allows DB2 to efficiently find the RBAs needed from the DB2 logs for recovery purposes. A row is inserted every time a table space or a partition is opened or updated and is updated when the object is closed.
SYSUTILX	This system utilities table stores information about the execution of DB2 utilities, including the status and the steps during execution. This information is used when a utility needs to be restarted. Information in this table is added when a utility is started, and the entry is removed when the execution has ended.

DB2 provides two methods of accessing data at remote application servers: DRDA and DB2 private protocol access. For application servers that support the two-phase commit process, both methods allow for updating data at several remote locations within the same unit of work.

The location name of the DB2 subsystem is defined during DB2 installation. The CDB records the location name and the network address of a remote DBMS. The tables in the CDB are part of the DB2 catalog.

Distributed Relational Database Architecture

With DRDA, the recommended method, 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, if bound with `DBPROTOCOL(DRDA)`, an alias to access the server.

Queries can originate from any system or application that issues SQL statements as an *application requester* in the formats required by DRDA. DRDA access supports the execution of dynamic SQL statements and SQL statements that satisfy all the following conditions.

- The static statements appear in a package bound to an accessible server.
- The statements are executed using that package.
- The objects involved in the execution of the statements are at the server where the package is bound. If the server is a DB2 subsystem, three-part names and aliases can be used to refer to another DB2 server.

DRDA access can be used in application programs by coding explicit `CONNECT` statements or by coding three-part names and specifying the `DBPROTOCOL(DRDA)` bind option. For more on bind options, refer to Chapter 11.

DRDA access is based on a set of DRDA protocols. (These protocols are documented by the Open Group Technical Standard in *DRDA Volume 1: Distributed Relational Database Architecture (DRDA)*.) DRDA communication conventions are invisible to DB2 applications and allow a DB2 to bind and rebind packages at other servers and to execute the statements in those packages.

For two-phase commit using SNA connections, DB2 supports both presumed-abort and presumed-nothing protocols that are defined by DRDA. If you are usingTCP/IP, DB2 uses the sync point manager defined in the documentation for DRDA Level 3.

DB2 Private Protocol

With private protocol, the application must use an alias or a 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 for z/OS subsystems.

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)`. The *current server* is the DBMS to which an application is actively connected. DB2 private protocol access uses *DB2 private connections.* The statements that can be executed are `SQL INSERT`, `UPDATE`, and `DELETE` and `SELECT` statements with their associated `SQL OPEN`, `FETCH`, and `CLOSE` statements.

In a program running under DB2, a *three-part name* or an *alias* can refer to a table or a view at another DB2. The location name identifies the other DB2 to the DB2 application server. A three-part name consists of a location, an authorization ID, and an object name. For example, the name `NYSERVER.DB2USER1.TEST` refers to a table named `DB2USER1.TEST` at the server whose location name is `NYSERVER`.

Alias names have the same allowable forms as table or view names. The name can refer to a table or a view at the current server or to a table or a view elsewhere. For more on aliases, refer to Chapter 4.

> **NOTE** Private protocol does not support many distributed functions, such as TCP/IP or stored procedures. The newer data types, such as LOB or user-defined types, are also not supported by private protocol. It is not the recommended method to use and is no longer being enhanced or supported from version 8 forward.

Communications Protocols

DDF uses TCP/IP or SNA to communicate with other systems. Setting up a network for use by database management systems requires knowledge of both database management and communications. Thus, you must put together a team of people with those skills to plan and implement the network.

TCP/IP

Transmission Control Protocol/Internet Protocol (TCP/IP) is a standard communication protocol for network communications. Previous versions of DB2 supported TCP/IP requesters, although additional software and configuration were required. Native TCP/IP eliminates these requirements, allowing gatewayless connectivity to DB2 for systems running UNIX System Services.

SNA

System Network Architecture (SNA) is the description of the logical structure, formats, protocols, and operational sequences for transmitting information through and controlling the configuration and operation of the networks. It is one of the two main network architectures used for network communications to the enterprise servers.

VTAM

DB2 also uses Virtual Telecommunications Access Method (VTAM) for communicating with remote databases. This is done be assigning two names for the local DB2 subsystem: a location name and a logical unit (LU) name. A *location name* distinguishes a specific database management system in a network, so applications use this name to direct requests to the local DB2 subsystem. Other systems use different terms for a location name. For example, DB2 Connect calls this the *target database nam*e. DB2 uses the DRDA term, *RDBNAM,* to refer to non-DB2 relational database names.

Communications Database

The DB2 catalog includes the communications database (CDB), which contains several tables that hold information about connections with remote systems. These tables are

- SYSIBM.LOCATIONS
- SYSIBM.LUNAMES
- SYSIBM.IPNAMES
- SYSIBM.MODESELECT
- SYSIBM.USERNAMES
- SYSIBM.LULIST
- SYSIBM.LUMODES

Some of these tables must be populated before data can be requested from remote systems. If this DB2 system services only data requests, the CDB does not have to be populated; the default values can be used.

When sending a request, DB2 uses the LINKNAME column of the SYSIBM.LOCATIONS catalog table to determine which protocol to use.

- To receive VTAM requests, a LUNAME must be selected in installation panel DSNTIPR.
- To receive TCP/IP requests, a DRDA port and a resynchronization port must be selected in installation panel DSNTIP5. TCP/IP uses the server's port number to pass network requests to the correct DB2 subsystem. If the value in the LINKNAME column is found in the SYSIBM.IPNAMES table, TCP/IP is used for DRDA connections. If the value is found in the SYSIBM.LUNAMES table, SNA is used.
- If the same name is in both SYSIBM.LUNAMES and SYSIBM.IPNAMES, TCP/IP is used to connect to the location.

NOTE A requester cannot use both SNA and TCP/IP to connect to a given location. For example, if SYSIBM.LOCATIONS specifies a LINKNAME of LU1, and if LU1 is defined in both the SYSIBM.IPNAMES and SYSIBM.LUNAMES tables, TCP/IP is the only protocol used to connect to LU1 from this requester for DRDA connections. For private protocol connections, the SNA protocols are used. If private protocol connections are being used, the SYSIBM.LUNAMES table must be defined for the remote location's LUNAME.

SUBSYSTEM POOLS

In the DBM1 address space are many objects critical to the operation of DB2. Here, we take a brief look at buffer pools and the RID, EDM, and Sort pools. For tuning information about the performance or use of these pools, refer to Chapter 17.

Buffer Pools

Buffer pools are database objects used to cache database data pages in memory. If an object's data page is placed in a buffer pool, physical input/output (I/O) access to disks will be avoided. Buffer pools can be assigned to cache only a particular table space's data, that is, within the table space definition.

Buffer pools are areas of virtual storage that temporarily store pages of table spaces or indexes. When a program accesses a row of a table, DB2 places the page containing that row in a buffer. When a program changes a row of a table, DB2 must eventually write the data in the buffer back to disk, normally at either a checkpoint or a write threshold. The write thresholds are either vertical—at the page set level—or horizontal—at the buffer pool level. Storage for buffer pools is backed by memory in the DBM1 address space. Prior to version 8, this was limited to 2GB and dataspaces and hiperpools were used to provide more storage. Version 8 has a large area of real addressable memory provided by 64-bit addressability. With 64-bit support DB2 can address up to 16 exabytes (less some operating system overhead). This allows buffer pools to mainly exist in real memory with up to a total of 1TB for all buffer pools in the susbystem.

Up to 80 virtual buffer pools are available. This allows for

- Fifty 4K-page buffer pools (BP0–BP49)
- Ten 32K-page buffer pools (BP32K–BP32K9)
- Ten 8K-page buffer pools
- Ten 16K-page buffer pools

Buffer pools take their space in the DBM1 address space. A single buffer pool can be up to 1TB, but the total of all buffer pools in the subsystem is also 1TB. As mentioned previously, up to 80 buffer pools can be defined. Creating a buffer pool is easily done via an `-ALTER BUFFERPOOL` command. The following example shows how to create a 3,000-page buffer pool with a sequential threshold of 50 percent:

```
-ALTER BUFFERPOOL (BP3) VPSIZE(3000) VPSEQT(50)
```

(Fifty percent of the buffer pool can be used for sequentially processed pages and 50 percent for randomly processed pages.) For more information on tuning these parameters, refer to Chapter 17.

Each buffer pool can have a different size and different parameter settings. These parameters control such activities as writing changed pages to disk. Changes to the buffer pool sizes and thresholds are also done via the -ALTER BUFFERPOOL command.

Environmental Descriptor Manager Pool

The EDM pool contains many items, including the following:

- EDM DBD cache
  - DBDs (database descriptors)
- EDM pool
  - SKCTs (skeleton cursor tables)
  - CTs (cursor tables, or copies of the SKCTs)
  - SKPTs (skeleton package tables)
  - PTs (package tables, or copies of the SKPTs)
  - Authorization cache block for each plan, except one with CACHESIZE set to 0
- EDM statement cache
  - Skeletons of dynamic SQL for CACHE DYNAMIC SQL

If the pool is too small, you will see increased I/O activity in the following DB2 table spaces, which support the DB2 directory: DSNDB01.DBD01, DSNDB01.SPT01, and DSNDB01.SCT02.

The impact of having too little memory for the statement cache for dynamic SQL is a small hit ratio, causing extra CPU for the full prepares and I/O on the catalog. The main goal for the EDM pool is to limit the I/O against the directory and catalog. If the pool is too small, response times will increase because of the loading of the SKCTs, SKPTs, and DBDs and repreparing the dynamic SQL statements because they could not remain cached.

By correctly sizing the EDM pool, you can avoid unnecessary I/Os from accumulating for a transaction. A SKCT, SKPT, or DBD that has to be reloaded into the EDM pool is additional I/O. This situation can occur if the pool pages are stolen because the EDM pool is too small. Pages in the pool are maintained on an LRU (least recently used) queue, and the least recently used pages are stolen, if required. In version 8, the DBD cache and dynamic statement cache sizes are separately configurable. EDM pool structures are mainly located above the 2GB bar.

Row Identifier Pool

The RID pool is used for storing and sorting RIDs for such operations as

- List prefetch
- Multiple index access
- Hybrid joins
- Enforcing unique keys while updating multiple rows

The optimizer looks at the size of the RID pool for prefetch and RID use. The full use of the RID pool is possible for any single user at runtime. Runtime can result in a table space scan if not enough space is available in the RID. For example, if you want to retrieve 10,000 rows from a 100,000,000-row table and no RID pool is available, a scan of 100,000,000 rows would occur, at any time and without external notification. The optimizer assumes that physical I/O will be less with a large pool.

The size is set with an installation parameter (128 KB–10,000 MB). The RID pool is created at start-up time, but no space is allocated until RID storage is needed. It is then allocated above the 16MB line in 16KB blocks as needed, until the maximum size you specified on installation panel DSNTIPC is reached. Some of the RID pool, 25 percent, is located below the 2GB bar and the remainder is located above the 2GB bar.

Sort Pool

At startup, DB2 allocates a sort pool in the private area of the DBM1 address space. DB2 uses a special sorting technique, called a tournament sort. During the sorting processes, this algorithm commonly produces logical work files called runs, which are intermediate sets of ordered data. If the sort pool is large enough, the sort completes in that area. More often than not, the sort cannot complete in the sort pool, and the runs are moved into the DSNDB07 work files that are used to perform sorts. These runs are later merged to complete the sort. When DSNDB07 is used for holding the pages that make up the sort runs, you could experience performance degradation if the pages get externalized to the physical work files, as they will have to be read back in later in order to complete the sort.

The sort pool size is also defined by a DSNZPARM and is currently 240KB to 128MB, with a 2MB default.

SUMMARY

This chapter examined most of the components that make up the DB2 environment. The operating systems for DB2 on a zSeries enterprise server are based on the core operating system of MVS. Within this structure, several address spaces make up the environment, and each address space is allocated certain functions. The main components of DB2 and all its managers are in the DBAS and SSAS. The IRLM communicates with DB2 and provides the internal lock manager.

Across the environment are security services, as a total umbrella to provide many levels of security. The highest level is the component called Security Server, which houses RACF.

DB2I and SPUFI are only two of several interfaces to DB2 but provide most of the direct interface to DB2 for DBAs and developers. SPUFI is used as a method of submitting batches of SQL statements, and DB2I as a real-time interface.

The installation process, performed from TSO or the msys for Setup DB2 Customization Center, is focused primarily on picking options for the many DSNZPARMs, the configuration parameters for DB2. More than 123 parameters can be modified online, without taking an outage.

The DB2 environment has six categories of commands and many utilities. The DSN commands and the DB2 commands are used the most.

The DB2 catalog and directory comprise the control repository for the environment. There are a large number of catalog tables, populated primarily by SQL DDL and DCL, for every object creation and modification.

We briefly examined the distributed networks architecture of SNA and TCP/IP and looked at how DRDA provides the services on top of those layers. Other subsystems objects, such as the buffer pools, the EDM pool, the RID pool, and the Sort pool, were discussed in terms of how DB2 uses them. These objects will be revisited in Chapter 17.

Additional Resources

IBM DB2 UDB for z/OS Version 8 Installation Guide (GC18-7418)

IBM DB2 UDB for z/OS Version 8 Utility Guide and Reference (SC18-7427)

IBM DB2 UDB for z/OS Version 8 Administration Guide (SC18-7413)

IBM DB2 UDB for z/OS Version 8 SQL Reference (SC18-7426)

Open Group Technical Standard in DRDA Volume 1: Distributed Relational Database Architecture (DRDA)

www.opengroup.com for information on DRDA

SNA LU 6.2 Peer Protocols Reference