Unlike the utilities and tools described in the previous chapter, which are also known as commands, all the commands in this chapter must be issued from the command line processor (or called from a programming interface). The programming interfaces are documented in the DB2 manual: Administrative API Reference.

Another difference is that while virtually every DB2 tool or utility is a one-word command that starts with “DB2,” you can tell the commands in this chapter apart because none of them begin with DB2. They also correspond to concepts familiar in almost every other relational database (backup, restore, load, import, export, describe, and so on).

Some examples in this chapter have the form “DB2 COMMAND etc.” Other examples are not preceded with “DB2”. These forms are interchangeable. The form with “DB2” is used with the DB2 Command window, or from a command prompt in UNIX or OS/2. You can use the commands with “DB2” preceding them in the Command Center or from the command line processor (CLP). Note that the “DB2” refers to the command line processor (versus the “DB2” as part of the name in the command, as in Chapter 29, for commands like DB2SET).

ACTIVATE DATABASE

This command wakes up the database and starts up all necessary database services, so the database is available for connection and use. Database administrators can use ACTIVATE DATABASE to start up selected databases. This eliminates any application time spent on database initialization.

If this command is not run, the first user connecting will experience a delay as DB2 creates all the buffer pools. You should run this command for production databases after they are started.

Authorization

SYSADM, SYSCTRL, or SYSMAINT.

Syntax

```
>>>ACTIVATE--++DATABASE++--database-alias----------------->
'--DB-----'

>-----+---------------------------------------+----------------><
'--USER--username--++----------------+--'
'--USING--password--'
```

Description

The following are the parameter descriptions for ACTIVATE DATABASE:

- **database-alias** Specifies the database to be started
USER username Specifies the user starting the database

USING password Specifies the password for the user

If a database is started by a CONNECT TO (or an implicit connect) command and subsequently an ACTIVATE DATABASE command is issued for that same database, then DEACTIVATE DATABASE command must be used to shut down that database. An application issuing ACTIVATE DATABASE cannot have an active database connection to any database.

ADD DATALINKS MANAGER

This command adds a DB2 Data Links Manager to the list of registered DB2 Data Links Managers for a specified database.

Authorization
None.

Syntax

```
>>-ADD DATALINKS MANAGER FOR-------+DATABASE---dbname---USING---->
----+'-DB-------'

>-----+-NODE--hostname---PORT--port-number---------------+-----><

'--CELL--DFS-cellname---DLMINSTANCE--instance-name--'
```

Description

The following are the parameter descriptions for ADD DATALINKS MANAGER:

- **DATABASE dbname** Specifies the database name.
- **USING NODE hostname** Specifies a fully qualified host name, or the IP address (but not both), of the DB2 Data Links Manager.
- **PORT port-number** Identifies the port number reserved for communications from the DB2 server to the DB2 Data Links Manager.
- **CELL DFS-cellname** Specifies the fully qualified name of the distributed file system (DFS) cell. For example, dln1.almaden.ibm.com. This parameter allows only one cell to be registered in a database.
- **DLMINSTANCE instance-name** Identifies the instance name that runs the Data Links Manager in the cell.

A Data Links Manager added by specifying USING NODE is a native type, whereas a Data Links Manager added by specifying USING CELL has the DFS type.
All Data Links Managers registered to a database must be of the same type, and only one Data Links Manager of type DFS can be registered to a database.

When registering one or more DB2 Data Links Managers for a database using this command, ensure that the DB2 Data Links Manager is not registered twice; otherwise, error SQL20056N with reason code “99” may be returned during data link processing. The db2diag.log file for the DB2 Data Links Manager server that is registered twice will have the following entry after such a failure:

```
dfm_xnstate_cache_insert : Duplicate txn entry.
dfmBeginTxn : Unable to insert ACTIVE transaction in cache, rc = 41.
DLFM501E : Transaction management service failed.
```

The CLP detects errors if duplicate Data Links Managers are added using the same name or address. However, duplicates are not detected if a Data Links Manager is added more than once using a different IP name or address. For example, if a Data Links Manager was added twice, once using the name dln1.almaden.ibm.com and again using the short name dln1, the preceding failure is possible. See also LIST DATALINKS MANAGERS.

## ADD NODE

ADD NODE adds a new node to a partitioned database. You can specify the source node for any system temporary table spaces to be created with the databases, or you can specify that no system temporary table spaces are to be created. The command must be issued from the node that is being added, and it can be issued only on a DB2 Enterprise—Extended Edition (EEE) server. This command affects only the node on which it is executed.

### Authorization

SYSADM or SYSCTRL.

### Syntax

```
>>-ADD NODE----------------------------------------------------------<
   +LIKE NODE--node-number++
   '-WITHOUT TABLESPACES------'
```

### Description

The following are the parameter descriptions for ADD NODE:
LIKE NODE node-number  The containers for the system temporary table spaces will be the same as the containers on the specified node-number for each database in the instance. The node specified must be a node that is already in the db2nodes.cfg file.

WITHOUT TABLESPACES  Containers for the system temporary table spaces are not created for any of the databases. The ALTER TABLESPACE statement must be used to add system temporary table space containers to each database before the database can be used.

If no option is specified, containers for the system temporary table spaces will be the same as the containers on the catalog node for each database. The catalog node may be a different node for each database in an EEE system. Before adding a new node, ensure that there is sufficient storage for the containers that must be created for all existing databases on the system. The add node operation creates an empty database partition on the new node for every database that exists in the instance. The configuration parameters for the new database partitions are set to the default value.

If an add node operation fails while creating a database partition locally, it enters a cleanup phase, in which it locally drops all databases that were created. This means that the database partitions are removed only from the node being added (that is, the local node). Existing database partitions remain unaffected on all other nodes. If this fails, no further cleanup is done, and an error is returned.

The database partitions on the new node cannot be used to contain user data until after the ALTER NODEGROUP statement has been used to add the node to a nodegroup. This command will fail if create database or drop database is in progress. The command can be reissued once the operation has completed.

If system temporary table spaces are created with the database partitions, ADD NODE may have to communicate with another node in the EEE system to retrieve the table space definitions. The start_stop_time database manager configuration parameter is used to specify the time in minutes by which the other node must respond with the table space definitions. If this time is exceeded, the command fails. Increase the value of start_stop_time and reissue the command.

ATTACH

ATTACH enables an application to direct instance-level commands such as CREATE DATABASE and FORCE APPLICATION to a specific instance. This can be the current instance, another instance on the same workstation, or a remote instance.

Authorization

None.
Syntax

```
>ATTACH----+---------------+---------------------------------->
  | 'TO--nodename--'
>-----+-----------------------------------------------------------------------------+>
  | 'USER--username--+--------------------------------------------------------+--'
  | 'USING--password--+-----------------------------------+-+
  | 'NEW--password--CONFIRM--password--'
  | 'CHANGE PASSWORD--'
```

Description

The following are the parameter descriptions for ATTACH:

- **TO nodename**  Specifies the instance to which the user wants to attach. This instance must have a matching entry in the local node directory. The only exception to this is the local instance (as specified by the DB2INSTANCE environment variable), which may be specified as the object of an attach, but which cannot be used as a node name in the node directory.

- **USER username**  Specifies the authentication identifier.

- **USING password**  Specifies the password for the username. If a user name is specified but a password is not specified, the user is prompted for the current password.

- **NEW password**  Specifies the new password if you decide to change it. The system on which the password will be changed depends on how user authentication has been set up.

- **CONFIRM password**  Specifies a string identical to the new password.

- **CHANGE PASSWORD**  If this is specified, the user is prompted for the current password, a new password, and for confirmation of the new password.

Examples

Catalog two remote nodes:

```
db2 catalog tcpip node node1 remote freedom server server1
db2 catalog tcpip node node2 remote flash server server1
```

Attach to the first node, end all user connections with FORCE APPLICATION ALL, and then detach:
db2 attach to node1
db2 force application all
db2 detach

Attach to the second node and see who is on:

db2 attach to node2
db2 list applications

**BACKUP DATABASE**

This command creates a backup copy of a database or table space. The command affects only the node on which it is executed. It must be run on all nodes to back up a partitioned database.

**Authorization**

SYSADM, SYSCtrl, or SYSMAINT.

**Syntax**

```sql
>>-BACKUP----+-DATABASE-+---database-alias---------------------->
     '-DB-------'
>-----+---------------------------------------+----------------->
     '-USER--username--+------------------+--'
     '-USING--password--'
>-----+--------------------------------------------+------------>
     |                .-,------------------.      |
     |                V                    |      |
>-----+---------+---+--------------------------+---------------->
     '-TABLESPACE--(-----tablespace-name---+---)--'
>-----+-------------------------------------------------------+->
     |-USE TSM--+-------------------------------+------------+
     |          '-OPEN--num-sessions--SESSIONS--'            |
     |     .-,--------.                                      |
     |     V          |                                      |
>-----+------------------------------------------------------+
```

**DB2 Command Line Processor Commands**

Complete Reference / DB2: TCR / Zikopoulos&Melnyk/ 3344-9 / Chapter 30
Description

BACKUP establishes a connection to the database that will be backed up. If a connection to the database already exists, it will be used for the backup and terminated at the end of the backup. These are the parameter descriptions for BACKUP:

- **DATABASE database-alias** Specifies the database to be backed up.
- **USER username** Specifies the user name under which to back up the database.
- **USING password** Specifies the password for the user name.
- **TABLESPACE tablespace-name** Specifies a list of table space names to be backed up. This is useful for a subset restore.
- **ONLINE** Allows the backup to take place while other users are connected to the database. The logs must be saved because they record all user activity during the backup. An online backup without accompanying logs cannot be restored and rolled forward to a consistent state.
- **INCREMENTAL** Specifies a cumulative (incremental) backup image. An incremental backup image is a copy of all database data that has changed since the most recent successful full backup operation. (See Figure 30-1.) Any INCREMENTAL CUMULATIVE backup image can be restored with the preceding full backup image. You can turn on tracking of incremental changes by using the database configuration parameter TRACKMOD.
- **DELTA** Specifies a noncumulative (delta) backup image. A delta backup image is a copy of all database data that has changed since the most recent successful backup operation of any type (full or incremental). Each DELTA image requires the preceding DELTA image for a RESTORE to be successful. (See Figure 30-1.)
- **USE TSM** Specifies that the backup will use Tivoli Storage Manager (formerly ADSM, or Adstar Storage Manager).
- **OPEN num-sessions SESSIONS** Allows faster backup by running parallel output over multiple sessions to storage software. Num-sessions is an integer.
The target has to be TSM, Legato, Veritas, or another storage product. This parameter has no effect when backing up to tape or disk.

- **TO dir/dev** Specifies a list of directory or tape device names. The full path of the directory must be specified, and the target must reside on the database server. This parameter may be repeated to specify the target directories and devices that the backup image will span. If more than one target is specified (target1, target2, and target3, for example), target1 will be opened first. The media header and special files (including the configuration file, table space table, and history file) are placed in target1. All remaining targets are opened, and they are then used in parallel during the backup operation. Because there is no general tape support on OS/2 or Windows, each type of tape device requires a unique device driver. The use of tapes may generate messages and prompts for user input. Valid response options are the following:

  - **Continue** Continues using the device that generated the warning message (For example, a new tape has been mounted.)
  - **Device terminate** Stops using only the device that generated the warning message (For example, there are no more tapes.)
  - **Terminate** Stops the backup

The database is quiesced before an online database backup operation with a user exit program starts. The backup utility waits until all transactions are committed or rolled back. While online backup is running, all new transactions wait until the backup operation completes. If the tape system does not support the ability to uniquely reference a backup image, do not keep multiple backup copies of the same database on the same tape.

- **LOAD library-name** Specifies the name of the shared library (DLL on OS/2 or Windows) containing the vendor backup and restore I/O functions. It can contain the full path. If the full path is not given, it will default to the path on which the user exit program resides.

- **WITH num-buffers BUFFERS** Specifies the number of buffers to use. The default is 2, but when backing up to multiple locations, more buffers can improve performance.

- **BUFFER buffer-size** Specifies the size, in 4KB pages, of the buffer used when building the backup image. The minimum is 8 pages and the default is 1024 pages. Specify 0 (zero) to use the value of database manager configuration parameter backbufsz.

- **PARALLELISM n** Determines the number of table spaces to be read in parallel by the backup utility. The default is 1.

- **WITHOUT PROMPTING** Specifies that the backup will run unattended. Any actions that require user intervention will return an error message.
Examples

The following code enables logretain and backs up the sample database on Windows to C:

```
db2 update database configuration for sample logretain on
db2 backup database sample to c:
```

Backup successful. The timestamp for this backup image is: 20010701205137. The backup taken with the preceding command will be in the following location (205137.001 is the file name of the backup image):

```
c:\SAMPLE.0\DB2\NODE0000\CATN0000\20010701\205137.001
```

Back up the sample database to Tivoli Storage Manager (TSM) using two sessions and four buffers:

```
db2 backup database sample use tsm open 2 sessions with 4 buffers
```

Back up the table space for the catalogs and table space USERSPACE1 for the payroll database with eight buffers and no user prompting:

```
db2 backup database payroll tablespace syscatspace, userspace1 to /dev/rmt0, /dev/rmt1 with 8 buffers without prompting
```
Turn on trackmod so that DB2 tracks incremental changes. Then make a full database backup on Sunday, cumulative incremental backups on Wednesday and Saturday, and incremental delta backups on the remaining days. All backups are made to TSM:

```bash
db2 update db cfg for sample using trackmod on
(Sun) db2 backup db kdr use tsm
(Mon) db2 backup db kdr online incremental delta use tsm
(Tue) db2 backup db kdr online incremental delta use tsm
(Wed) db2 backup db kdr online incremental use tsm
(Thu) db2 backup db kdr online incremental delta use tsm
(Fri) db2 backup db kdr online incremental delta use tsm
(Sat) db2 backup db kdr online incremental use tsm
```

## BIND

The BIND command binds SQL statements stored in the bind file generated by the precompiler and creates a package that is stored in the database.

### Authorization

One of the following:

- SYSADM
- DBADM
- BINDADD privilege if a package does not exist, and one of the following:
  - IMPLICIT_SCHEMA authority on the database if the schema name of the package does not exist
  - CREATEIN privilege on the schema if the schema name of the package exists
  - ALTERIN privilege on the schema if the package exists
- BIND privilege on the package if it exists

The user also needs all privileges required to compile any static SQL statements in the application. Privileges granted to groups are not used for authorization checking of static statements. If the user has SYSADM authority but not explicit privileges to complete the bind, the database manager grants explicit DBADM authority automatically.

### Syntax

```bash
>>-BIND--filename---------------------------------------------->
          |                   ---UNAMBIG--.  |
          |                          -BLOCKING--ALL------>
          |                     -NO------
```
Description

The following are the parameter descriptions for BIND:

- **filename** Specifies the name of the bind file that was generated when the application program was precompiled, or a list file containing the names of several bind files. Bind files have the extension .bnd, and the full path name can be specified. If a list file is specified, precede the list file name with the at (@) character—by convention, the extension is first. The list file can contain several lines of bind file names. Bind files listed on the same line must be separated by plus (+) characters, but a plus cannot appear in front of the first file listed on each line or after the last bind file listed. The plus can appear at the end of any line but the last. For example, /u/smith/sqlib/bnd/@all.lst is a list file that contains the following bind files:

  mybind1.bnd+mybind.bnd2+mybind3.bnd+
  mybind4.bnd+mybind5.bnd+
  mybind6.bnd+
  mybind7.bnd

- **BLOCKING** Allows three values: ALL, NO, and UNAMBIG:
  - **ALL** Blocks for read-only cursors and cursors not specified as FOR UPDATE OF. Ambiguous cursors are treated as read-only.
  - **NO** Does not block any cursors. Ambiguous cursors are treated as updateable.
UNAMBIG  Blocks for read-only cursors and cursors not specified as FOR UPDATE OF. Ambiguous cursors are treated as updateable.

CLIPKG cli-packages  Specifies an integer between 3 and 30 indicating the number of CLI large packages to be created when binding CLI bind files against a database.

CNULREQD  Specifies C/C++ precompilation level:

NO  Specifies that the application was coded on the basis of the langlevel SAA1 precompile option with respect to the null terminator in C string host variables.

YES  Specifies that the application was coded on the basis of the langlevel MIA precompile option with respect to the null terminator in C string host variables.

COLLECTION schema-name  Defines a 30-character collection identifier for the package. If this is not specified, the authorization identifier for the user processing the package is used.

DATETIME  Specifies the date and time format to use for date and time values in the database.

DEF  Uses a date and time format associated with the country code of the database.

EUR  Date is dd.mm.yyyy. Time is hh.mm.ss.

ISO  Date is yyyy-mm-dd. Time is hh.mm.ss.

JIS  Japanese Industrial Standard. Date is yyyy-mm-dd. Time is hh:mm:ss.

LOC  Uses the date and time format in local form associated with the country code of the database.

USA  Date is mm/dd/yyyy. Time is hh:mm AM or PM.

DEGREE  Specifies the degree of parallelism for execution of static SQL statements in an SMP system. This option does not affect CREATE INDEX parallelism.

1  Specifies no parallelism.

degree-of-parallelism (2 to 32,767)  Attempts to use this many agents (potentially threads, processes, and even CPUs) for SQL statements.

ANY  Lets DB2 determine the degree of parallelism.

DYNAMICRULES  Defines the rules for dynamic SQL at run time for the initial settings of authorization ID and the implicit qualification of unqualified object references.

RUN  The authorization ID of the user executing the package is used. This is the default.

BIND  All rules that apply to static SQL for authorization and qualification are used at run time. That is, the authorization ID of the package owner is used for
authorization checking of dynamic SQL statements, and the default package qualifier is used for implicit qualification of unqualified object references within dynamic SQL statements. When binding a package with this option, the binder of the package should not have any authorities that the user of the package should not inherit because dynamic SQL statements will be using the authorization ID of the package owner. The following dynamically prepared SQL statements cannot be used within a package that has been bound with this option: GRANT, REVOKE, ALTER, CREATE, DROP, COMMENT ON, RENAME, SET CONSTRAINTS, and SET EVENT MONITOR STATE.

- **EXPLAIN** Stores information in the Explain tables about the access plans chosen for each SQL statement in the package.
  - **NO** Explain information is not captured.
  - **YES** Explain tables are populated with information about the chosen access plan at prep/bind time for static statements and at run time for incremental bind statements.
  - **ALL** Explain information for each eligible static SQL statement is placed in the Explain tables at prep/bind time. Explain information for each eligible incremental bind SQL statement is placed in the Explain tables at run time. In addition, Explain information will be gathered for eligible dynamic SQL statements at run time, even if the CURRENT EXPLAIN SNAPSHOT register is set to NO.

- **EXPLSNAP** Stores Explain Snapshot information in the Explain tables.
  - **NO** An Explain Snapshot is not captured.
  - **YES** An Explain Snapshot for each eligible static SQL statement is placed in the Explain tables at prep/bind time for static statements and at run time for incremental bind statements.
  - **ALL** An Explain Snapshot for each eligible static SQL statement is placed in the Explain tables at prep/bind time. Explain Snapshot information for each eligible incremental bind SQL statement is placed in the Explain tables at run time. In addition, Explain Snapshot information is gathered for eligible dynamic SQL statements at run time, even if the CURRENT EXPLAIN SNAPSHOT register is set to NO.

- **FEDERATED** Specifies whether a static SQL statement in a package references a nickname or a federated view. If this option is specified and a nickname or federated view is not referenced, an error is returned and the package will not be created. If this option is not specified and a nickname or federated view is referenced by a static statement in the package, an error is returned and the package will not be created.
  - **NO** Binds against a DB2 database. This is the default.
  - **YES** Binds against a DB2 federated system.
**FUNCPATH** Specifies the function path to be used in resolving user-defined distinct types and functions in static SQL. If this option is not specified, the default function path is SYSIBM, SYSFUN, USER where `USER` is the value of the `USER` special register. The schema name can be an SQL identifier, either ordinary or delimited, which identifies a schema that exists at the application server. No validation that the schema exists is made at precompile time or at bind time. The same schema cannot appear more than once in the function path. The number of schemas that can be specified is limited by the length of the resulting function path, which cannot exceed 254 bytes. The schema SYSIBM does not need to be explicitly specified; it is implicitly assumed to be the first schema if it is not included in the function path.

**GRANT:** Who you are granting to
- **Authid** Grants EXECUTE and BIND privileges to a specified user name or group ID.
- **PUBLIC** Grants EXECUTE and BIND privileges to PUBLIC.
- **GRANT_GROUP** group-name Grants EXECUTE and BIND privileges to a specified group ID.
- **GRANT_USER** user-name Grants EXECUTE and BIND privileges to a specified user name.

**INSERT** Allows a program being precompiled or bound against a DB2 Enterprise—Extended Edition server to request that data inserts are buffered to increase performance.
- **BUF** Inserts from an application are buffered.
- **DEF** Inserts from an application are not buffered.

**ISOLATION** Determines how far a program bound to this package can be isolated from the effect of other executing programs:
- **CS** Cursor Stability is the isolation level.
- **RR** Repeatable Read is the isolation level.
- **RS** Read Stability is the isolation level.
- **UR** Uncommitted Read is the isolation level. This is also known as dirty read.

**MESSAGES** message-file Specifies the destination for warning, error, and completion status messages. A message file is created whether the bind is successful or not. If a message filename is not specified, messages are written to standard output. If the complete path to the file is not specified, the current directory is used. If the name of an existing file is specified, the contents of the file are overwritten.

**OWNER** authorization-id Specifies a 30-character authorization identifier for the package owner. The owner must have the privileges required to execute the SQL statements contained in the package. Only a user with SYSADM or
DBADM authority can specify an authorization identifier other than the user ID. The default is the primary authorization ID of the precompile/bind process. SYSIBM, SYSCAT, and SYSSTAT are not valid values for this option.

- **QUALIFIER qualifier-name** Specifies a 30-character implicit qualifier for unqualified objects contained in the package. The default is the owner’s authorization ID, whether or not owner is explicitly specified.

- **QUERYOPT optimization-level** Specifies the level of optimization for all static SQL statements contained in the package. The default is 5, and the choices are as follows:
  - 0 Perform minimal optimization. Query rewrite not active. This is most suitable for simple dynamic SQL access to well-indexed tables.
  - 1 Optimization roughly comparable to DB2/UNIX version 1. No query rewrite.
  - 2 A level of optimization higher than that of DB2 version 1, but at significantly less optimization cost than levels 3 and above, especially for very complex queries.
  - 3 A moderate amount of optimization to generate an access plan.
  - 5 A significant amount of optimization to generate an access plan. For complex dynamic SQL queries, heuristic rules are used to limit the amount of time spent selecting an access plan. Where possible, queries will use summary tables instead of the underlying base tables.
  - 7 A significant amount of optimization to generate an access plan. Similar to 5 but without the heuristic rules.
  - 9 A maximal amount of optimization to generate an access plan. This can greatly expand the number of possible access plans that are evaluated. Use 9 to determine if a better access plan can be generated for very complex and very long-running queries using large tables. Explain and performance measurements can be used to verify that a better plan has been generated.

- **SQLERROR** Indicates whether a package or a bind file is created if an error is encountered.

- **CHECK** The target system performs all syntax and semantic checks on the SQL statements being bound. A package will not be created as part of this process. If, while binding, an existing package with the same name and version is encountered, the existing package is neither dropped nor replaced even if action replace was specified.

- **CONTINUE** Creates a package, even if errors occur when binding SQL statements. Those statements that failed to bind for authorization or existence reasons can be incrementally bound at execution time if VALIDATE RUN is also specified. Any attempt to execute them at run time generates an error (SQLCODE -525, SQLSTATE 51015).
DB2: The Complete Reference

- **NOPACKAGE** No package or bind file is created if an error occurs.

- **SQLWARN** Indicates whether warnings are returned from the compilation of dynamic SQL statements (via PREPARE or EXECUTE IMMEDIATE), or from describe processing (via PREPARE...INTO or DESCRIBE).
  - **NO** Warnings are not returned from the SQL compiler.
  - **YES** Warnings are returned from the SQL compiler.

- **SQLCODE +238** An exception. It is returned regardless of the sqlwarn option value.

- **TRANSFORM GROUP** Specifies the transform group name used by static SQL statements for exchanging user-defined structured type values with host programs. This transform group is not used for dynamic SQL statements or for the exchange of parameters and results with external functions or methods. The groupname is an SQL identifier 1–18 characters long. A group name cannot include a qualifier prefix and cannot begin with the prefix SYS because this is reserved for database use. In a static SQL statement that interacts with host variables, the name of the transform group used for exchanging values of a structured type is the first of the following that applies:
  - The group name in the TRANSFORM GROUP bind option, if any.
  - The group name in the TRANSFORM GROUP prep option as specified at the original precompilation time, if any.
  - The DB2_PROGRAM group, if a transform exists for the given type whose group name is DB2_PROGRAM. No transform group is used if none of these conditions exist. The following errors are possible during the bind of a static SQL statement:
    - **SQLCODE yyy, SQLSTATE xxxxx** A transform is needed, but no static transform group has been selected.
    - **SQLCODE yyy, SQLSTATE xxxxx** The selected transform group does not include a necessary transform (TO SQL for input variables, FROM SQL for output variables) for the data type that needs to be exchanged.
    - **SQLCODE yyy, SQLSTATE xxxxx** The result type of the FROM SQL transform is not compatible with the type of the output variable, or the parameter type of the TO SQL transform is not compatible with the type of the input variable.

- **VALIDATE** Determines when the database manager checks for authorization errors and object not found errors. The package owner authorization ID is used for validity checking.

- **BIND** Validation is performed at precompile/bind time. If all objects do not exist or all authority is not held, error messages are produced. If sqlerror continue is specified, a package/bind file is produced despite the error message, but the statements in error are not executable.
Validation is attempted at bind time. If all objects exist and all authority is held, no further checking is performed at execution time. If all objects do not exist or all authority is not held at precompile/bind time, warning messages are produced and the package is successfully bound, regardless of the sqlerror continue option setting. However, authority checking and existence checking for SQL statements that failed these checks during the precompile/bind process may be redone at execution time.

Binding can be done as part of the precompile process for an application source file, or as a separate step later. Use BIND to perform this as a separate step. The name used to create the package is stored in the bind file, and it is based on the source filename from which it was generated (existing paths or extensions are discarded). For example, a precompiled source file called myapp.sql generates a default bind file called myapp.bnd and a default package name of MYAPP. However, the bind filename and the package name can be overridden at precompile time by using the bindfile and package options.

Binding a package with a schema name that does not already exist creates that schema. The schema owner is SYSIBM. The CREATEIN privilege on the schema is granted to PUBLIC.

BIND executes under the transaction that was started. After performing the bind, BIND issues a COMMIT or a ROLLBACK to terminate the current transaction and start another one.

Binding stops if a fatal error or more than 100 warning errors occur. If a fatal error occurs, binding stops, bind tries to close all files and discards the package.

If a package is bound with dynamicrules bind, the implicit or explicit value of the bind option owner is used for authorization checking of dynamic SQL statements, and the implicit or explicit value of the bind option qualifier is used as the implicit qualifier of unqualified objects within dynamic SQL statements. If multiple packages are referenced during a single connection, dynamic SQL statements prepared by a specific package behave according to the bind options for that package. The value of the special register CURRENT SCHEMA has no effect on qualification in a package bound with dynamicrules bind.

If an SQL statement is found to be in error and the BIND option SQLERROR CONTINUE was specified, the statement will be marked as invalid. To change the state of the SQL statement, issue another BIND. Implicit and explicit rebind will not change the state of an invalid statement. In a package bound with VALIDATE RUN, a statement can change from static to incremental bind or incremental bind to static across implicit and explicit rebinds depending on whether object existence or authority problems exist during the rebind.

Example

The following example binds myapp.bnd (the bind file generated when the myapp.sqc program was precompiled) to the database to which a connection has been established:

```
  db2 bind myapp.bnd
```

Any messages resulting from the bind process are sent to standard output.
CALL

CALL invokes a stored procedure defined in the catalog.

Authorization

The privileges held by the authorization ID of the CALL statement at run time must include at least one of the following:

- Execute privilege for the package associated with the stored procedure.
- Control privilege for the package associated with the stored procedure.
- SYSADM or DBADM authority.

Syntax

```
V

>>-CALL--proc-name-------------------------><
'

argument--'
```

Description

The following are the parameter descriptions for CALL:

- **proc-name** Identifies the stored procedure to be called. This can be the name of the procedure to run or the fully qualified schema name.
- **argument** Provides one or more arguments for the stored procedure.

Example

User newton creates a stored procedure named getMedian and issues

```
CALL getMedian
```

Then user dbuser1 creates a stored procedure that is also called getMedian. If newton reissues CALL getMedian, it will fail because the procedure name exists in multiple schemas (NEWTON and DBUSER1). For the command to work, Newton must issue

```
CALL NEWTON.getMedian
```
CATALOG DATABASE

This command stores database location information in the system database directory. The database can be located either on the local workstation or on a remote node. For EEE, when cataloging a local database into the system database directory, this command must be issued from a node on the server where the database resides.

Authorization

SYSADM or SYSCTRL.

Syntax

```
>>-CATALOG----+---DATABASE-+--database-name----+------------+----->
'-'DB------'      'AS--alias--'

>-----+--------------------+------------------------------------>
'+-ON--+-path--+------+
|     '-drive-'      |
| 'AT NODE--nodename--'

>-----+----------------------------------------------------------------+>
|                 .-SERVER------------------------------------.  |
| '-AUTHENTICATION--+-CLIENT------------------------------------+--'

+-DCS---------------------------------------+
+-SERVER_ENCRYPT----------------------------+
+-DCS_ENCRYPT-------------------------------+
+-KERBEROS TARGET PRINCIPAL--principalname--+
'--DCE SERVER PRINCIPAL--principalname------'

>-----+-------------------------+------------------------------><
| 'WITH--"comment-string"--'
```

Description

Use CATALOG DATABASE to identify databases located on local or remote nodes, recatalog databases that were uncataloged previously, or maintain multiple aliases for one database (regardless of location). DB2 automatically catalogs databases when they are created. It catalogs an entry for the database in the local database directory and another entry in the system database directory. If the database is created from a remote client (or a client that is executing from a different instance on the same machine), an entry is also made in the system database directory at the client instance.
If neither path nor node name is specified, the database is assumed to be local, and the location of the database is assumed to be that specified in the database manager configuration parameter dftdbpath. Databases on the same node as the database manager instance are cataloged as indirect entries. Databases on other nodes are cataloged as remote entries.

CATALOG DATABASE automatically creates a system database directory if one does not exist. The system database directory is stored on the path containing the database manager instance being used, and it is maintained outside of the database. List the contents of the system database directory using LIST DATABASE DIRECTORY.

The following are the parameter descriptions for CATALOG DATABASE:

- **DATABASE database-name**  Specifies the name of the database to catalog.
- **AS alias**  Specifies an alias to use as an alternate name for the database being cataloged. If an alias is not specified, the database manager uses database-name as the alias.
- **ON path/drive**  On UNIX  Specifies the path on which the cataloged database resides. On OS/2 or Windows, this identifies the letter of the drive containing the cataloged database.
- **AT NODE nodename**  Specifies the name of the node where the cataloged database resides. (See CATALOG TCP/IP NODE.) This name should match the name of an entry in the node directory. If the node name specified does not exist in the node directory, a warning is returned, but the database is cataloged in the system database directory. Catalog the node before trying to connect.
- **AUTHENTICATION**  Required only for DB2 version 1.

**Examples**

Catalog the sample database and add the comment “Sample Database”:

```
catalog database sample on /databases/sample with "Sample Database"
```

Catalog a TCP/IP node, catalog the database statsdb at this node, reset the connection, and connect to the database with the user ID dbtstv and the password 69chev:

```
catalog tcpip node statsdb remote cheetah server xdbtstv
catalog database stats at node statsdb
connect reset
connect to stats user dbtstv using 69chev
```

**CATALOG DCS DATABASE**

This command stores information about remote databases in the Database Connection Services (DCS) directory. These databases are accessed through an Application Requester (AR), such as DB2 Connect.
Authorization
SYSADM or SYSCTRL.

Syntax

```
>>>-CATALOG DCS----+DATABASE+-+database-name+-----------> 
              '-DB------'

              +---------------------------------------------+----------->

              '-AS--target-database-name-'  '-AR--library-name--'

              +---------------------------------------------+----------->

              '-PARMS--"parameter-string"--'

              +---------------------------------------------+----------->

              '-WITH--"comment-string"--'
```

Description
The following are the parameter descriptions for CATALOG DCS DATABASE:

- **DATABASE database-name**: Specifies the alias of the target database to catalog. This alias must match the database name entered during the remote cataloging of this database in the system database directory.
- **AS target-database-name**: Specifies the name of the target host database to catalog.
- **AR library-name**: This is not required for DB2 Connect. It specifies the name of the Application Requester library that is loaded and used to access a remote database listed in the DCS directory.
- **PARMS “parameter-string”**: Specifies a parameter string passed to the AR when invoked. The parameter string must be enclosed by double quotation marks.
- **WITH “comment-string”**: Describes the DCS directory entry. Any comment that helps to describe the database cataloged can be entered in this directory.

Example
The following example catalogs information about the DB1 database, which is a DB2 for OS/390 host database, in the DCS directory:

```
db2 catalog dcs database db1 as dsn_db_1 with "DB2/MVS location name DSN_DB_1"
```
CATALOG GLOBAL DATABASE

CATALOG GLOBAL DATABASE creates a system database directory that defines a local alias of the fully qualified DCE directory object name of the target database. The information about that database is stored centrally in the DCE directory.

Authorization

SYSADM or SYSCTRL.

Syntax

```
>>-CATALOG GLOBAL--+-DATABASE-+---database-global-name-------->
   '-DB-------'

>----AS--alias--USING DIRECTORY--DCE---------------------------->

>-----+-------------------------+------------------------------><

'-WITH--"comment-string"--'
```

Description

The following are the parameter descriptions for CATALOG GLOBAL DATABASE:

- **DATABASE database-global-name** Specifies the fully qualified name that uniquely identifies the database in the DCE name space.
- **AS alias** Identifies an alternate name for the database being cataloged.
- **USING DIRECTORY DCE** Specifies the global directory service being used.
- **WITH “comment-string”** Describes the DCE type entry in the system database directory. Any comment that helps to describe the database cataloged can be entered in this directory.

Example

Catalog a global database using DCE:

```
db2 catalog global database /.../cell1/subsys/database/DB3 as
dbtest using directory dce
```

CATALOG LDAP DATABASE

This command is used to register the database in Lightweight Directory Access Protocol (LDAP). This command is available only on Windows, AIX, and Solaris.
Authorization

None.

Syntax

```bash
>>-CATALOG LDAP----+DATABASE++database-name----------------->
   '-DB-------'

>------------------------------------------------------------------->
   '-AS--alias--'   '-AT NODE--nodename--'

>------------------------------------------------------------------->
   '-GWNODE--gateway-node--'

>------------------------------------------------------------------->
   '-PARMS--"parameter-string"--'   '-AR--library-name--'

>------------------------------------------------------------------->
   '-AUTHENTICATION--+-CLIENT------------------------------------+
      +SERVER-------------------------------+
      +SERVER_ENCRYPT----------------------+
      +DCS_ENCRYPT--------------------------+
      +DCS-----------------------------------+
      +KERBEROS TARGET PRINCIPAL--principalname--+
      '-DCE SERVER PRINCIPAL--principalname------'

>------------------------------------------------------------------->
   '-WITH--"comments"--'

>------------------------------------------------------------------->
   '-USER--username--+-------------------------------+
      'PASSWORD--password--'
```

Description

The following are the parameter descriptions for CATALOG LDAP:

- **DATABASE:DATABASE database-name** Specifies the name of the database to catalog.
- **AS alias** Provides an alias to use as an alternate name for the database. If an alias is not specified, the database name is used as the alias.
- **AT NODE nodename** Identifies the LDAP node name for the database server. Mandatory for a remote database.
- **GWNODE gateway-node** Specifies the LDAP node name for the gateway server (such as DB2 Connect Enterprise Edition).

- **PARMS “parameter-string”** Specifies a parameter string passed to the Application Requester (AR) when accessing DCS databases.

- **AR library-name** Required only for Distributed Relational Database Architecture (DRDA), and then not required with DB2 Connect. AR library name must be used when the database is cataloged from a DRDA requestor, such as an application on OS/390 accessing a DB2 database on UNIX.

- **AUTHENTICATION** Specifies the authentication level. Valid values are as follows:
  - **CLIENT** Authentication takes place on the node from which the application is invoked, such as the client workstation. This is suitable only if all clients are secure and controllable. It is not recommended for Windows 9x.
  - **SERVER** Authentication takes place on the node containing the target database. This is the most common choice.
  - **SERVER_ENCRYPT** Authentication takes place on the node containing the target database, and passwords are encrypted at the source. Passwords are decrypted at the target, as specified by the authentication type cataloged at the source.
  - **DCS_ENCRYPT** Authentication takes place on the node containing the target database, except when using DB2 Connect, when authentication takes place at the DRDA Application Server (AS). Passwords are encrypted at the source and decrypted at the target, as specified by the authentication type cataloged at the source.
  - **DCS** Authentication takes place on the node containing the target database, except when using DB2 Connect, when authentication takes place at the DRDA Application Server (AS).
  - **KERBEROS** Authentication takes place using Kerberos Security Mechanism. The TARGET PRINCIPAL principalname is valid only on Windows 2000 clients and is a fully qualified Kerberos principal name for the target server; that is, the logon account of the DB2 server service in this form:
    \[
    \text{userid@xxx.xxx.xxx.com or domain\userid}
    \]
  - **DCE** Authentication takes place using DCE Security Services. The SERVER PRINCIPAL principalname is a fully qualified DCE principal name for the target server. This value is also recorded in the keytab file at the target server.
  - **WITH “comments”** Specifies 1–30 characters describing the DB2 server. Any comment that helps to describe the server registered in the network directory can be entered.
  - **USER username** Provides the user’s LDAP distinguished name (DN). The LDAP user DN must have sufficient authority to create the object in the LDAP directory. If the user’s LDAP DN is not specified, the credentials of the current logon user will be used. If the user’s LDAP DN and password have been
specified using db2ldcfg, the user name and password do not have to be
specified here. For more information about this command, see the
DB2LDCFG command (it configures the LDAP environment) in Chapter 29.

**PASSWORD password** Specifies an account password. If the user’s LDAP DN
and password have been specified by using DB2LDCFG, the user name and
password do not have to be specified here. For more information about this command,
see Chapter 29. If the node name is not specified, DB2 uses the first node in LDAP
that represents the DB2 server on the current machine. It may be necessary to
manually register (catalog) the database in LDAP if one of these is true:

- The database server does not support LDAP. In this case, the administrator
  must manually register each database in LDAP to allow clients that support
  LDAP to access the database without having to catalog the database locally
  on each client machine.
- The application wants to use a different name to connect to the database. In this
  case, the administrator can catalog the database by using a different alias name.
- The database resides at the host database server (for example, DB2/390,
  DB2/400, and so on). In this case, the administrator can register the database
  in LDAP and specify the gateway node through the GWNODE parameter.

During CREATE DATABASE IN LDAP, the database name may already exist in
LDAP. The database is still created on the local machine (and can be accessed by local
applications), but the existing entry in LDAP will not be modified to reflect the new
database. In this case, the administrator can do either of the following:

- Remove the existing database entry in LDAP and manually register the new
database in LDAP.
- Register the new database in LDAP using a different alias name.

---

**CATALOG LDAP NODE**

This command catalogs a new node entry in LDAP. It is available only on Windows,
AIX, and Solaris.

**Authorization**

None.

**Syntax**

```
>>-CATALOG LDAP---NODE---nodename---AS---nodealias---------------->
>-----+------------------------------------------+------------>
```

---
'--USER--username--+'--password--'

Description
The following are the parameter descriptions for CATALOG LDAP NODE:

- **NODE nodename** Specifies the LDAP node name of the DB2 server.
- **AS nodealias** Provides a new alias name for the LDAP node entry.
- **USER username** Identifies the user’s LDAP distinguished name (DN). The LDAP user DN must have sufficient authority to create the object in the LDAP directory. If the user’s LDAP DN is not specified, the credentials of the current logon user will be used.
- **PASSWORD password** Specifies the account password.

The CATALOG LDAP NODE command is used to specify a different alias name for the node that represents the DB2 server.

### CATALOG LOCAL NODE
This command creates a local alias for an instance that resides on the same machine. A local node should be cataloged when there is more than one instance on the same workstation to be accessed from the user’s client. Interprocess Communications (IPC) is used to access the local node.

**Authorization**
SYSADM or SYSCTRL.

**Syntax**

```plaintext
>>-CATALOG--+-LOCAL NODE--nodename--+-ADMIN--'

>-----+-------------------------+---+----------------------+---->

'-INSTANCE--instancename--'  '-SYSTEM--system-name--'

>-----+--------------------------------+------------------------>

'-OSTYPE--operating-system-type--'

>-----+-------------------------+------------------------------><

'-WITH--"comment-string"--'
```
Description

The following are the parameter descriptions for CATALOG LOCAL NODE:

- **ADMIN** Catalogs a local administration server node.
- **NODE nodename** Specifies a local alias for the cataloged node. This is an arbitrary name on the user’s workstation used to identify the node. Make it a meaningful name that is easy to remember.
- **INSTANCE instancename** Identifies the local instance to be accessed.
- **SYSTEM system-name** Specifies the DB2 system name used to identify the server machine.
- **OSTYPE operating-system-type** Identifies the operating system. This can be AIX, Windows 95, NT, OS2, HPUX, SUN, MVS, OS400, VM, VSE, SNI, SCO, SGI, LINUX, or DYNIX.
- **WITH “comment-string”** Describes the node entry in the node directory. Any comment that helps to describe the node can be entered.

Example

Workstation A has two server instances, inst1 and inst2. To create databases at both instances from a single CLP session, issue the following commands (assume the DB2INSTANCE environment variable is set to inst1):

- Create a local database at inst1:
  
  ```
  db2 create database mydb1
  ```

- Catalog another server instance on this workstation:
  
  ```
  db2 catalog local node mynode2 instance inst2
  ```

- Create a database at mynode2:
  
  ```
  db2 attach to mynode2
  db2 create database mydb2
  ```

**CATALOG NAMED PIPE NODE**

This command adds a named pipe node entry to the node directory. The named pipe is used to access the remote node. This command is available only on Windows NT and Windows 2000.
Authorization

SYSADM or SYSCTRL.

Syntax

```
>>-CATALOG--+-NPIPE
NODE--nodename--REMOTE--computername-->'-ADMIN-'

>-----INSTANCE--instancename----+------------------------->
   '-SYSTEM--system-name--'

>-----+-------------------------+------------------------------><
   '-OSTYPE--operating-system-type--'

>-----+--------------------------------+------------------------->
   '-WITH--"comment-string"--'
```

Description

The database manager creates the node directory when the first node is cataloged (that is, when the first CATALOG...NODE command is issued). You can list the contents of the local node directory by using LIST NODE DIRECTORY.

The following are the parameter descriptions for CATALOG NPIPE NODE:

- **ADMIN** Catalogs an NPIPE administration server node.
- **NODE nodename** Specifies a local alias for the node. Use something meaningful that is easy to remember.
- **REMOTE computername** Specifies the computer name of the node on which the target database resides. Maximum length is 15 characters.
- **INSTANCE instancename** Provides the name of the server instance on which the target database resides. This is identical to the name of the remote named pipe, which is used to communicate with the remote node.
- **SYSTEM system-name** Specifies the DB2 system name used to identify the server machine.
- **OSTYPE operating-system-type** Specifies the operating system. This can be AIX, Windows 95, NT, OS2, HPUX, SUN, MVS, OS400, VM, VSE, SNI, SCO, SGI, Linux, or DYNIX.
- **WITH “comment-string”** Describes the node entry in the node directory.
Example

Catalog a node with a named pipe and provide a comment:

```
db2 catalog npipe node db2np1 remote nphost instance db2inst1 with
"A remote named pipe node."
```

**CATALOG NETBIOS NODE**

CATALOG NETBIOS NODE adds a NetBIOS node entry to the node directory. The NetBIOS communications protocol is used to access the remote node. This command is available only on Windows and OS/2.

**Authorization**

SYSADM or SYSCTRL.

**Syntax**

```
>>-CATALOG--+-NETBIOS
      NODE--nodename--REMOTE--server-nname-->'-ADMIN-

      >---ADAPTER--adapter-number------------------------------------>
      >-----+---------------------------------+----------------------->
      >-----+----------------------+---------------------------------->
      >-----+--------------------------------+------------------------>
      >-----+-------------------------+------------------------------>

      '-REMOTE_INSTANCE--instance-name--'

      '-SYSTEM--system-name--'

      '-OSTYPE--operating-system-type--'

      '-WITH--"comment-string"--'
```

**Description**

The database manager creates the node directory when the first node is cataloged (that is, when the first CATALOG...NODE command is issued). You can list the contents of the local node directory by using LIST NODE DIRECTORY.
The following are the parameter descriptions for CATALOG NETBIOS NODE:

- **ADMIN** Specifies an administration server node.
- **NODE nodename** Specifies a local alias for the node. Use a meaningful name that is easy to remember.
- **REMOTE server-name** Specifies the name of the remote workstation where the target database resides.
- **ADAPTER adapter-number** Specifies the local, logical outgoing LAN adapter number. The default is 0, but it is often set to 1 on Windows.
- **REMOTE_INSTANCE instance-name** Specifies the real name of the instance to which an attachment is being made on the remote server machine.
- **SYSTEM system-name** Provides a name used to identify the server machine.
- **OSTYPE operating-system-type** Identifies the operating system. This can be AIX, Windows 95, NT, OS2, HPUX, SUN, MVS, OS400, VM, VSE, SNI, SCO, SGI, Linux, or DYNIX.
- **WITH “comment-string** Describes the node entry in the node directory.

**Example**

db2 catalog netbios node db2netb1 remote db2inst1 adapter 0 with "A remote NetBIOS node"

**CATALOG ODBC DATA SOURCE**

This command catalogs a user or system Open Database Connectivity (ODBC) data source. An ODBC data source is a user-defined name for a specific database. This is used to access the database (or file system) through ODBC interfaces. Either user or system data sources can be cataloged. A user data source is visible only to the user who cataloged it.

This command is available only on Windows.

**Authorization**

None.

**Syntax**

```
USER---.
>>-CATALOG--------ODBC DATA SOURCE--data-source-name-->
    '-SYSTEM-
```
Description
The following are the parameter descriptions for CATALOG ODBC DATA SOURCE:

- **USER** Catalogs a user data source. This is the default if no keyword is specified.
- **SYSTEM** Catalogs a system data source.
- **ODBC DATA SOURCE** **data-source-name** Specifies the name of the data source to be cataloged.

Example
Catalog the ODBC data source called pmrdb (in this case, a DB2 database called PMRDB):
```
catalog odbc data source pmrdb
```

**CATALOG TCP/IP NODE**
This command tells a DB2 client where it can find a database server through Transmission Control Protocol/Internet Protocol (TCP/IP).

Authorization
SYSADM or SYSCTRL.

Syntax
```
>>-CATALOG--------TCPIP NODE--nodename--REMOTE--hostname--> 
'--ADMIN--'

>-------------SERVER--service-name--------------+------------------>
'--SECURITY SOCKS-'

>----------------------------------->
'--REMOTE_INSTANCE--instance-name--'

>----------------------------------->
'--SYSTEM--system-name--'

>----------------------------------->
'--OSTYPE--operating-system-type--'

>----------------------------------><
'--WITH--"comment-string"--'
```
**Description**

The database manager creates the node directory when the first node is cataloged (that is, when the first CATALOG...NODE command is issued). If directory caching is enabled (see the configuration parameter `dir_cache` in the “GET DATABASE MANAGER CONFIGURATION” section later in this chapter), directory files are cached in memory.

An application’s directory cache is created during its first directory lookup. Because the cache is refreshed only when the application modifies any of the directory files, directory changes made by other applications may not be effective until the application has restarted. To refresh the directory cache, use TERMINATE. To refresh DB2’s shared cache, stop (db2stop) and then restart (db2start). To refresh the directory cache for another application, stop and then restart that application.

The following are the parameter descriptions for CATALOG TCPIP NODE:

- **ADMIN** Catalogs an administration node with TCP/IP.
- **NODE nodename** Specifies an alias for the node. This is an arbitrary name on the user’s workstation, used to identify the node. Use something meaningful that is easy to remember.
- **REMOTE hostname** Identifies the hostname of the node where the target database resides. The hostname is the name of the node that is known to the TCP/IP network. Type `hostname` at the server to verify the hostname. You can also type `ping hostname` from the client to verify that the server and client are in communication with TCP/IP.
- **SERVER service-name** Specifies the service name or the port number of the server database manager instance. The CATALOG TCPIP NODE command is run on a client. The use of SERVER for this parameter indicates the location of the database on another machine, indicated by the service-name. If a service-name is specified, the services file on the client is used to map the service name to a port number. For example,

  ```
  C:\WINNT\system32\drivers\etc\services
  ```

  A service name is specified in the database manager configuration file, and the services file on the server is used to map this service name to a port number. The port number on the client and the server must match. While you can specify a port number instead of a service name, don’t do this.

  If a port number is specified, it must match the port number associated with the service name specified in the server’s database manager configuration file. No service name needs to be specified in the local TCP/IP services file.

  The value of service-name is used as a key to search the local services file for the associated port number. If a matching entry is not found and the service-name is numeric, the value is interpreted as the port number. This parameter must not be specified for ADMIN nodes. The value on an ADMIN node is always 523.
SECURITY SOCKS  Specifies that the node will be SOCKS-enabled. The following environment variables are mandatory and must be set to enable SOCKS:

- **SOCKS_NS**  Identifies the Domain Name Server for resolving the host address of the SOCKS server. This should be an IP address.
- **SOCKS_SERVER**  Identifies the fully qualified host name or the IP address of the SOCKS server. If the SOCKSified DB2 client is unable to resolve the fully qualified host name, it assumes that an IP address has been entered.

One of the following conditions should be true to use SOCKS_SERVER:

- The SOCKS server should be reachable via the domain name server.
- The SOCKS server should be listed in the hosts file. The location of this file is described in the TCP/IP documentation.
- The SOCKS server should be in an IP address format.

If these environment variables are set after a db2start has been issued, it is necessary to issue a TERMINATE command and then CONNECT again.

- **REMOTE_INSTANCE instance-name**  Specifies the name of the server instance you’re attaching to.
- **SYSTEM system-name**  Provides the DB2 system name used to identify the server machine.
- **OSTYPE operating-system-type**  Specifies the operating system. This can be AIX, Windows 95, NT, OS2, HPUX, SUN, MVS, OS400, VM, VSE, SNI, SCO, SGI, Linux, or DYNIX.
- **WITH “comment-string”**  Describes the node entry in the node directory.

Examples

Catalog a TCPIP node with and without comments:

```
catalog tcpip node db2tcp1 remote tcphost server db2inst1 with "A remote TCP/IP node"
catalog tcpip node statsdb remote cheetah server xdbtsvt
catalog tcpip node db2tcp2 remote 9.21.15.235 server db2inst2 with "TCP/IP node using IP address"
```

**CHANGE DATABASE COMMENT**

CHANGE DATABASE COMMENT changes a database comment in the system database directory or the local database directory. New comment text replaces existing text. To append information, enter the old comment text, followed by the new text. This command affects only the node where it is run.
Authorization
SYSADM or SYSCTRL.

Syntax

```
>>-CHANGE----+-DATABASE-+--database-alias--COMMENT-------------->
     '-DB------'

>-----+----------------+---WITH--"comment-string"--------------><
     '-ON+++path+++--'
     '-drive-'
```

Description

Only the comment for an entry associated with the database alias is modified. Other entries with different aliases are not affected. If the path is specified, the database alias must be cataloged in the local database directory. If the path is not specified, the database alias must be cataloged in the system database directory.

The following are the parameters for CHANGE DATABASE COMMENT:

- **DATABASE database-alias** Specifies the database alias whose comment will be changed.
- **ON path/drive on UNIX** Specifies the path on which the database resides, and changes the comment in the local database directory. If a path is not specified, the database comment for the entry in the system database directory is changed. On Windows or OS/2, this specifies the letter of the drive on which the database resides.
- **WITH “comment-string”** Describes the entry in the system database directory or the local database directory. Enter any comment that helps to describe the database.

Example

The following example changes the text in the system database directory comment for the SAMPLE database from “Test 2 - Holding” to “Test 2 - Add employee inf rows”:

```
db2 change database sample comment with "Test 2 - Add employee inf rows"
```

**CHANGE ISOLATION LEVEL**

This command changes the way DB2 isolates data from other processes while a database is being accessed.
Authorization

None.

Syntax

```sql
CHANGE ISOLATION TO
```

Description

SQLISL is a synonym for ISOLATION; DB2 uses isolation levels to maintain data integrity in a database. The isolation level defines the degree to which an application’s reading and writing of data is isolated from changes made by other concurrent access to the same table.

- ISOLATION TO specifies the isolation level for the application to one of the following:
  - CS  Cursor Stability is the isolation level.
  - RR  Repeatable Read is the isolation level.
  - RS  Read Stability is the isolation level.
  - UR  Uncommitted Read is the isolation level. Also known as dirty read.

Isolation level changes are not permitted while connected to a database with a type 1 connection. (See SET CLIENT and CONNECT in Chapter 27.) Changes are permitted using a type 2 connection but should be made with caution, because the changes will apply to every connection made from the same command line processor process. You have to remember which isolation level applies to which connected database.

Examples

Terminate a connection, change the isolation level to Uncommitted Read, and reconnect to the SAMPLE database:

```sql
db2 terminate
db2 change isolation to ur
db2 connect to sample
```

In the following example, a user is in DB2 interactive mode following the creation of the SAMPLE database:

```sql
update command options using c off
```
catalog db sample as sample2

set client connect 2
connect to sample
connect to sample2

change isolation to cs
set connection sample
declare c1 cursor for select * from org
open c1
fetch c1 for 3 rows

change isolation to rr
fetch c1 for 2 rows

An SQL0514N error occurs in the preceding example because c1 is not in a prepared state for this isolation level.

change isolation to cs
set connection sample2
fetch c1 for 2 rows

An SQL0514N error occurs in the preceding example because c1 is not in a prepared state for this database. A DB21029E error also occurs because cursor c1 has already been declared and opened.

set connection sample
fetch c1 for 2 rows

This example works because the original database (SAMPLE) was used with the original isolation level (CS).

CREATE DATABASE

CREATE DATABASE initializes a new database with an optional user-defined collating sequence, creates the three initial table spaces, creates catalogs, and allocates the recovery log. In EEE, this command affects all nodes that are listed in the db2nodes.cfg file. The node from which this command is issued becomes the catalog node for the new database. To create a database at a remote node, attach to that node.

A database connection is temporarily established by using the CREATE DATABASE command.
Authorization
SYSADM and SYSCTRL.

Syntax

```
CREATE DATABASE database-name

AT NODE

CREATE Database options

ON path

ALIAS database-alias

drive

USING CODESET codeset

TERRITORY territory

SYSTEM

COLLATE USING compatibility

IDENTITY

NUMSEGS numsegs

DFT_EXTENT_SZ dft_extentsize

CATALOG TABLESPACE tblspace-defn

USER TABLESPACE tblspace-defn

TEMPORARY TABLESPACE tblspace-defn

WITH "comment-string"

tblspace-defn

MANAGED BY
```
CREATE DATABASE creates a database in the specified subdirectory. In an EEE system, it creates the database on all nodes listed in db2nodes.cfg, and it creates a $DB2INSTANCE/NODExxxx directory under the specified subdirectory at each node. In a non-EEE system, it creates a $DB2INSTANCE/NODE0000 directory under the specified subdirectory.

CREATE DATABASE also does the following:

- Creates the system catalog tables and recovery log.
- Catalogs the database in the following database directories. (If neither of these directories exists, it creates the directory. If specified, the comment and code set values are placed in both directories.)
  - Server’s local database directory on the path indicated by path or, if the path is not specified, on the default database path defined in the database manager system configuration file. A local database directory resides on each file system that contains a database.
Server’s system database directory for the attached instance. The resulting directory entry contains the database name and a database alias. If the command was issued from a remote client, the client’s system database directory is also updated with the database name and an alias.

Stores the specified code set, territory, and collating sequence. A flag is set in the database configuration file if the collating sequence consists of unique weights or if it is the identity sequence.

Creates the schemata called SYSCAT, SYSFUN, SYSIBM, and SYSSTAT with SYSIBM as the owner. The server node on which this command is issued becomes the catalog node for the new database. Two nodegroups are created automatically: IBMDEFAULTGROUP and IBMCATGROUP.

Binds the previously defined database manager bind files to the database. (These are listed in the utilities bind file list, db2ubind.lst.) If one or more of these files do not bind successfully, CREATE DATABASE returns a warning in the SQLCA, and it provides information about the binds that failed. If a bind fails, the user can take corrective action and manually bind the failing file. The database is created in any case. A schema called NULLID is implicitly created when performing the binds with the CREATEIN privilege granted to PUBLIC.

Creates SYSCATSPACE, TEMPSPACE1, and USERSPACE1 table spaces. The SYSCATSPACE table space is created only on the catalog node.

Grants the following privileges and authorities:

- DBADM authority, and CONNECT, CREATETAB, BINDADD, CREATE_NOT_FENCED, and IMPLICIT_SCHEMA
- LOAD privileges to the database creator
- CONNECT, CREATETAB, BINDADD, and IMPLICIT_SCHEMA privileges to PUBLIC
- USE privilege on the USERSPACE1 table space to PUBLIC
- SELECT privilege on each system catalog to PUBLIC
- BIND and EXECUTE privileges to PUBLIC for each successfully bound utility

With DBADM authority, one can grant these privileges to (and revoke them from) other users or PUBLIC. If another administrator with SYSADM or DBADM authority over the database revokes these privileges, the database creator nevertheless retains them.

In an EEE environment, the database manager creates a subdirectory, $DB2INSTANCE/NODExxxx, under the specified or default path on all nodes. The xxxx is the node number as defined in the db2nodes.cfg file (that is, node 0 becomes NODE0000). Subdirectories SQL00001 through SQLnnnnn will reside on this path. This ensures that the database objects associated with different nodes are stored in different directories (even if the subdirectory $DB2INSTANCE under the specified or default path is shared by all nodes).
If Lightweight Directory Access Protocol (LDAP) support is enabled on the current machine, the database will be automatically registered in the LDAP directory. If a database object of the same name already exists in the LDAP directory, the database is still created on the local machine, but a warning message is returned, indicating that there is a naming conflict. In this case, the user can manually catalog an LDAP database entry by using CATALOG LDAP DATABASE.

CREATE DATABASE fails if the application is already connected to a database. Use CATALOG DATABASE to define different alias names for the new database.

For details on the tblspace-defn parameters, see CREATE TABLESPACE in Chapter 27. The table space definitions specified on CREATE DATABASE apply to all nodes on which the database is being created. If the table space definitions are to be created differently on particular nodes, use CREATE TABLESPACE.

When defining containers for table spaces, $N can be used. $N is replaced by the node number when the container is actually created. Use this to specify containers in a multiple logical node database.

The following are the parameters for CREATE DATABASE:

- **DATABASE database-name** Specifies a name for the new database. This must be a unique name that differentiates the database from any other database in either the local database directory or the system database directory.

- **AT NODE** Specifies that the database is to be created only on the node that issues the command. This parameter is not intended for general use; improper use causes inconsistencies in the system, so use it with caution. It could be used with RESTORE DATABASE if the database partition at a node was damaged and must be re-created. If used to re-create a database partition that was dropped (because it was damaged), the database at this node will be in the restore-pending state. After re-creating the database partition, the database must immediately be restored on this node.

- **ON path/drive On UNIX** Specifies the path on which the database should be created. If a path is not specified, the database is created on the default database path specified in the database manager configuration file (dftdbpath parameter). On Windows or OS/2, this specifies the letter of the drive on which the database should be created.

- **ALIAS database-alias** Specifies an alias for the database in the system database directory. If not provided, the specified database name is used.

- **USING CODESET codeset** Specifies the code set to be used for data entered into this database. This is dependent on National Language requirements.

- **TERRITORY territory** Specifies the territory used for data entered into this database.

- **COLLATE USING** Identifies the collating sequence used for the database. Once the database has been created, the collating sequence cannot be changed. Possible values are:

  - **COMPATIBILITY** for version 2 compatibility only
IDENTITY Identity collating sequence, in which strings are compared byte for byte

SYSTEM Collating sequence based on the current territory

NUMSEGS numsegs Specifies the number of segment directories to be created and used to store DAT, IDX, LF, LB, and LBA files for default system-managed space (SMS) table spaces. This parameter does not affect database-managed space (DMS) table spaces, any SMS table spaces with explicit creation characteristics (created when the database is created), or any SMS table spaces explicitly created after the database is created.

DFT_EXTENT_SZ dft_extentsize Specifies the default extent size of table spaces in the database.

CATALOG TABLESPACE tblspace-defn Specifies the definition of the catalog table space: SYSCATSPACE. If not specified, SYSCATSPACE is created as an SMS table space, which we recommend, with numsegs number of directories as containers, and with an extent size of dft_extentsize. For example, the following containers would be created if numsegs were specified as 5:

/u/smith/smith/NODE0000/SQL00001/SQLT0000.0
/u/smith/smith/NODE0000/SQL00001/SQLT0000.1
/u/smith/smith/NODE0000/SQL00001/SQLT0000.2
/u/smith/smith/NODE0000/SQL00001/SQLT0000.3
/u/smith/smith/NODE0000/SQL00001/SQLT0000.4

In an EEE system, the catalog table space is created only on the catalog node (the node on which CREATE DATABASE is issued).

USER TABLESPACE tblspace-defn Specifies the definition for the initial user table space, USERSPACE1. If not specified, USERSPACE1 is created as an SMS table space with numsegs number of directories as containers, and with an extent size of dft_extentsize. For example, the following containers would be created if numsegs were specified to be 5:

/u/smith/smith/NODE0000/SQL00001/SQLT0001.0
/u/smith/smith/NODE0000/SQL00001/SQLT0001.1
/u/smith/smith/NODE0000/SQL00001/SQLT0001.2
/u/smith/smith/NODE0000/SQL00001/SQLT0001.3
/u/smith/smith/NODE0000/SQL00001/SQLT0001.4

TEMPORARY TABLESPACE tblspace-defn Specifies the definition of the initial system temporary table space, TEMPSPACE1. If not specified, TEMPSPACE1 will be created as an SMS table space (which we recommend), with numsegs number of directories as containers, and with an extent size
of dftextsiz. For example, the following containers would be created if numseg were specified to be 5:

```
/u/smith/smith/NODE0000/SQL00001/SQLT0002.0
/u/smith/smith/NODE0000/SQL00001/SQLT0002.1
/u/smith/smith/NODE0000/SQL00001/SQLT0002.2
/u/smith/smith/NODE0000/SQL00001/SQLT0002.3
/u/smith/smith/NODE0000/SQL00001/SQLT0002.4
```

- WITH “comment-string” Describes the database entry in the database directory. Enter a comment that helps to describe the database.

---

**DEACTIVATE DATABASE**

DEACTIVATE DATABASE stops the specified database. In an EEE system, this command deactivates the specified database on all nodes in the system. If one or more of these nodes encounters an error, a warning is returned. The database will be successfully deactivated on some nodes, but may remain activated on the nodes encountering the error.

**Authorization**

SYSADM, SYSCRTL, or SYSMAINT.

**Syntax**

```
>>-DEACTIVATE-+DATABASE-+database-alias----->
  '-DB------'

>-----+---------------------------------------+----------------><
  '-USER--username--+------------------+--'
  '-USING--password--'
```

**Description**

The following are the parameters for DEACTIVATE DATABASE:

- **DATABASE database-alias** Specifies the alias of the database to be stopped
- **USER username** Identifies the user stopping the database
- **USING password** Specifies the password for the user ID

Databases initialized by ACTIVATE DATABASE can be shut down by DEACTIVATE DATABASE or by DB2STOP. If a database was initialized by ACTIVATE DATABASE, the
last application disconnecting from the database will not shut down the database, and DEACTIVATE DATABASE must be used. (In this case, DB2STOP will also shut down the database.) The application issuing the DEACTIVATE DATABASE command cannot have an active database connection to any database.

## DEREGISTER

This command deregisters the DB2 server from the network directory server.

### Authorization

None.

### Syntax

```bash
>DEREGISTER------------------------------------------------------------------------------------------------------------------------------------------>
    '-DB2 SERVER--' '-IN--'
>-----+-NWBINDERY--USER--username-------------------------------------------------------------------------------------------------------------------->
    |    '-PASSWORD--password--'
    |    '-LDAP--NODE--nodename------------------------------------------------------------------------------------------------------------------------'
    |    '-USER--username--------------------------------------------------------------------------------------------------------------------------------
    |    '-PASSWORD--password--'
```

### Description

The following are the parameters for DEREGISTER:

- **IN** Specifies the network directory server from which to deregister the DB2 server. Values are NWBINDERY for a NetWare bindery, and LDAP for an LDAP directory server.

- **USER username** For NWBINDERY Specifies the user ID to log on to the network server. For LDAP, it specifies the user’s LDAP distinguished name (DN). The LDAP user DN must have sufficient authority to delete the object from the LDAP directory. The user name is optional when deregistering in LDAP. If the user’s LDAP DN is not specified, the credentials of the current logon user are used.

- **PASSWORD password** Specifies the account password.

- **NODE nodename** Gives the value specified when the DB2 server was registered in LDAP.
DESCRIBE

This command displays the SQL Data Area (SQLDA) information about a SELECT statement, columns of a table or a view, and indexes of a table.

Authorization

To display the SQLDA information about a SELECT statement, one of the following privileges or authorities is required for each table or view referenced in the SELECT statement:

- SELECT privilege
- CONTROL privilege
- SYSADM or DBADM

These same privileges or authorities are required to display the columns or indexes of a table or a view for the system catalogs SYSCAT.COLUMNS (DESCRIBE TABLE) and SYSCAT.INDEXES (DESCRIBE INDEXES FOR TABLE).

Syntax

```sql
DESCRIBE

select-statement

TABLE table-name

INDEXES FOR TABLE table-name

SHOW DETAIL
```

- **select-statement** Specifies the statement for which you want to see the SQLDA. The SELECT statement is prepared automatically by the CLP or Command Center.
- **TABLE table-name** Identifies the table or view to be described. Use the fully qualified name or alias in the form schema.table-name. The schema is the user name under which the table or view was created. This option is not supported for declared temporary tables.
- **INDEXES FOR TABLE table-name** Identifies the table for which indexes are to be described. Use the fully qualified name or alias in the form schema.table-name. The schema is the user name under which the table was created.
- **SHOW DETAIL** Specifies that extra detail information (as explained in Chapter 27) will be shown for the DESCRIBE TABLE or DESCRIBE INDEXES FOR TABLE command.

Description

The following are the parameters for DESCRIBE:

- Column name
The DESCRIBE INDEXES FOR TABLE command lists the following information about each index of the table:

- Index schema
- Index name
- Unique rule
- Column count

The SHOW DETAIL option for the DESCRIBE TABLE command specifies that output include the following information:

- Whether a CHARACTER, VARCHAR, or LONG VARCHAR column was defined as FOR BIT DATA
- Column number
- Partitioning key sequence
- Code page
- Default

For the DESCRIBE INDEXES FOR TABLE command, SHOW DETAIL specifies that the output include column names.

Examples

Showing the SQLDA for SELECT:

```sql
describe select * from staff
```

SQLDA Information

```text
sqlaid : SQLDA   sqldabc: 896  sqln: 20  sqld: 7  
Column Information
sqtype   sqlen  sqlname.data  sqlname.length
-------------------  -----  ----------------------
```
Describing a table:

```
describe table user1.department
Table: USER1.DEPARTMENT
Column       Type       Type
name         schema     name
nulls
--------------- ----------- -------- -----
AREA         SYSIBM     SMALLINT  2
DEPT         SYSIBM     CHARACTER 3
DEPTNAME     SYSIBM     CHARACTER 20
```

Describing an index:

```
describe indexes for table user1.department
Table: USER1.DEPARTMENT
Index schema    Index name Unique rule Number of columns
--------------- ----------- -------- -------- ------------
USER1           IDX1        U         U           2
```
DETACH

DETACH removes the logical instance attachment and terminates the physical communication connection if there are no other logical connections.

Authorization

None.

Syntax

```
>>-DETACH----------------------------------------><
```

DROP DATABASE

DROP DATABASE deletes the database contents, uncatalogs the database, deletes the database subdirectory, and deletes all log files. If log files are needed for a roll-forward recovery after a restore operation, the files should be saved prior to issuing this command.

Authorization

SYSADM or SYSCTRL.

Syntax

```
>>-DROP----+-DATABASE-+--database-alias----+----------+--------><

'-DB-------'                    '-AT NODE--'
```

Description

The following are the parameters for DROP DATABASE:

- **DATABASE database-alias** Specifies the alias or name of the database to drop. The database must be cataloged in the system database directory.

- **AT NODE** Specifies that the database should be deleted only on the node that issued the DROP DATABASE command. This parameter is used for EEE only by utilities. It is not intended for general use. Improper use of this parameter causes inconsistencies in the system.

The database must not be in use when DROP DATABASE is issued; all users must be disconnected from the database before it can be dropped.
Only the specified database alias is removed from the system database directory. If other aliases with the same database name exist, their entries remain. If the database being dropped is the last entry in the local database directory, the local database directory is deleted automatically.

If DROP DATABASE is issued from a remote client (or from a different instance on the same machine), the specified alias is removed from the client’s system database directory. The corresponding database name is removed from the server’s system database directory.

This command unlinks all files that are linked through any DATALINK columns. Because the unlink operation is performed asynchronously on the DB2 Data Links Manager, its effects may not be seen immediately on the DB2 Data Links Manager, and the unlinked files may not be immediately available for other operations. When the command is issued, all the DB2 Data Links Managers configured to that database must be available; otherwise, the drop database operation will fail.

Example
Delete the SAMPLE database:

```
drop database sample
```

**DROP NODE VERIFY**

This command verifies whether a node exists in the nodegroups of any databases, and whether an event monitor is defined on the node. This command should be used prior to dropping a node from an EEE system. This command affects only the node on which it is issued.

**Authorization**

SYSADM.

**Syntax**

```
>>-DROP NODE VERIFY--------------------------------------------><
```

**Description**

If a message is returned indicating that the node is not in use, use STOP DATABASE MANAGER with DROP NODENUM to remove the entry for the node from the db2nodes.cfg file, which removes the node from the database system. If a message is returned indicating that the node is in use, follow these steps:
1. If the node contains data, redistribute the data to remove it from the node by using REDISTRIBUTE NODEGROUP. Use either the DROP NODE option on the REDISTRIBUTE NODEGROUP command or the ALTER NODEGROUP statement to remove the node from any nodegroups for the database. This must be done for each database that contains the node in a nodegroup.

2. Drop any event monitors defined on the node.

3. Rerun DROP NODE VERIFY to ensure that the database is no longer in use.

---

**ECHO**

ECHO permits the user to write character strings to standard output, like the DOS echo command.

**Authorization**

None.

**Syntax**

```bash
>>>-ECHO--+-+------------------+----------------------------------><
'-character-string-'  
```

**Description**

For this command, the character-string can be any character string.

If an input file is used as standard input or comments are to be printed without being interpreted by the command shell, the ECHO command prints character strings directly to standard output. One line is printed each time ECHO is issued.

The ECHO command is not affected by the verbose (-v) option of the CLP.

---

**EXPORT**

This command exports data from a database to one of several external file formats. The user specifies the data to be exported by supplying an SQL SELECT statement or by providing hierarchical information for typed tables.

**Authorization**

SYSADM, DBADM, CONTROL, or SELECT on each participating table or view.
Syntax

```plaintext
>>>-EXPORT TO--filename--OF--filetype-------------------------->
>>-+-+---------------------------+-----------------------------+
| .-,-----------. |
| V 
| '-LOBS TO-----lob-path-----'
>>-+---------------------------+-----------------------------+
| .-,-----------. |
| V 
| '-LOBFILE-----filename----'
>>-+-----------------------------------+---------------------+
| .-----------------. |
| V 
| '-MODIFIED BY-----filetype-mod-----'
>>-+--------------------------------------+------------------+
| .-,--------------. |
| V 
| '-METHOD N--(--column-name--++)--'
>>-+-----------------------------------+
| .-----------------. |
| V 
| '-MESSAGES--message-file--'

-----select-statement--------------------------------------------+
| '-HIERARCHY---STARTING--subtable-name--------++-------------'
| ' |-- traversal-order-list | ' '-where-clause--'
```

description

The following are the parameters for EXPORT:

- **HIERARCHY traversal-order-list**Exports a subhierarchy using the specified traverse order. All subtables must be listed in PRE-ORDER fashion. The first subtable name is used as the target table name for the SELECT statement.
DB2 Command Line Processor Commands

- **HIERARCHY STARTING subtable-name** Exports a subhierarchy starting from subtable-name using the default traverse order (OUTER order for ASC, DEL, or WSF files, or the order stored in PC/IXF data files).

- **LOBFILE filename** Specifies one or more file names for LOB files. When name space is exhausted for the first name, the second name is used, and so on.

  **Note**
  
  When creating LOB files during an export, file names are constructed by appending the current name from this list to the current path (from lob-path), and then adding a three-digit sequence number. If the current LOB path is the directory /u/foo/lob/path, and the current LOB filename is bar, the LOB files created will be /u/foo/lob/path/bar.001, /u/foo/lob/path/bar.002, and so on.

- **LOBS TO lob-path** Specifies one or more paths to directories where LOB files are stored. When file space is exhausted on the first path, the second path will be used, and so on.

- **MESSAGES message-file** Specifies the destination for warning and error messages that occur during export. If the file already exists, export appends the information. If message-file is omitted, the messages are written to standard output.

- **METHOD N column-name** Specifies one or more column names to be used in the output file. If this parameter is not specified, the column names in the table are used. This parameter is valid only for WSF and IXF files, and it is not valid when exporting hierarchical data.

- **MODIFIED BY filetype-mod** Specifies additional options described in Table 30-1. These can apply to all file types, WSF, or DEL (delimited ASCII).

- **OF filetype** Specifies the format of the data in the output file:
  - **DEL (delimited ASCII format)** Used by a variety of database manager and file manager programs.
  - **WSF (work sheet format)** Used by programs such as Lotus 1-2-3.

  **Note**
  
  When exporting BIGINT or DECIMAL data, only values that fall within the range of type DOUBLE can be exported accurately. Although values that do not fall within this range are also exported, importing or loading these values back may result in inconsistent data, depending on the operating system.

- **IXF (integrated exchange format, PC version)** Most of the table attributes, as well as any existing indexes, are saved in the IXF file, except when columns are specified in the SELECT statement. With this format, the table can be re-created, while with the other file formats, the table must already exist before data can be imported into it.
**select-statement** Specifies the SELECT statement that chooses the data to be exported. If the SELECT statement causes an error, a message is written to the message file (or to standard output). If the error code is SQL0012W, SQL0347W, SQL0360W, SQL0437W, or SQL1824W, export continues; otherwise, it stops.

**TO filename** Specifies the file to which data is exported. If the complete path to the file is not specified, export uses the current directory and the default drive as the destination. If the name of a file that already exists is specified, export overwrites the contents of the file; it does not append the information.

### Delimiter Restrictions

Ensure that the chosen delimiter character is not part of the data to be moved. If it is, unexpected errors may occur. The following restrictions apply to column, string, DATALINK, and decimal point delimiters when moving data:

<table>
<thead>
<tr>
<th>File Type Modifiers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>All File Formats</strong></td>
<td></td>
</tr>
<tr>
<td>Lobsinfile</td>
<td>lob-path specifies the path to the files containing LOB values. Used when the table has CLOB or LOB values, and you want these written to files in the lob-path on the hard disk.</td>
</tr>
<tr>
<td><strong>DEL</strong></td>
<td>Delimited ASCII</td>
</tr>
<tr>
<td>chardelx</td>
<td>(x) is a single character string delimiter that is used in place of the default double quotation mark (&quot;&quot;') to enclose a character string. The single quotation mark (') can be specified as a character string delimiter as follows: modified by chardel''</td>
</tr>
<tr>
<td>coldelx</td>
<td>(x) is a single character column delimiter that is used in place of the default, a comma (,), to signal the end of a column. In the following example, coldel causes export to interpret any semicolon (;) it encounters as a column delimiter: export to temp of del modified by coldel; select * from staff where dept = 20</td>
</tr>
<tr>
<td>dateiso</td>
<td>Date format. Causes all date data values to be exported in ISO format (YYYY-MM-DD).</td>
</tr>
</tbody>
</table>

**Table 30-1.** Valid File Type Modifiers (Export)
Delimiters are mutually exclusive.

A delimiter cannot be binary zero, a linefeed character, a carriage return, or a blank space.

The default decimal point (.) cannot be a string delimiter.

The following characters are specified differently by an ASCII-family code page and an EBCDIC-family code page:

- The Shift-In (0x0F) and the Shift-Out (0x0E) character cannot be delimiters for an EBCDIC MBCS data file.

### Table 30-1. Valid File Type Modifiers (Export) (continued)

<table>
<thead>
<tr>
<th>File Type Modifiers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>decplusbank</td>
<td>Plus sign (+) character. Causes positive decimal values to be prefixed with a blank space instead of the default plus sign.</td>
</tr>
<tr>
<td>Decpt( \text{x} )</td>
<td>( \text{x} ) specifies a single character substitute for the default decimal point character, the period (.).</td>
</tr>
<tr>
<td>Dldel( \text{x} )</td>
<td>( \text{x} ) is a single character DATALINK delimiter. The specified character is used in place of the default, a semicolon (;), as the interfield separator for a DATALINK value. It is needed because a DATALINK value may have more than one subvalue. ( \text{x} ) must not be the same character specified as the row, column, or character string delimiter.</td>
</tr>
<tr>
<td>nodoubledel</td>
<td>Suppresses recognition of double character delimiters. For more information, see the following “Delimiter Restrictions” section.</td>
</tr>
</tbody>
</table>

### WSF

- **Workstation File Format (Lotus 1-2-3)**
  - 1 Creates a WSF file that is compatible with Lotus 1-2-3 release 1. This is the default setting.
  - 2 Creates a WSF file that is compatible with Lotus Symphony release 1.0.
  - 3 Creates a WSF file that is compatible with Lotus 1-2-3 version 2, or Lotus Symphony release 1.1.
  - 4 Creates a WSF file containing double byte (DBCS) characters.
Delimiters for MBCS (Unicode), EUC, or DBCS code pages cannot be greater than 0x40, except the default decimal point for EBCDIC MBCS data, which is 0x4b.

Default delimiters for data files in ASCII code pages or EBCDIC MBCS code pages are as follows:
- " (0x22, double quotation mark; string delimiter)
- , (0x2c, comma; column delimiter)

Default delimiters for data files in EBCDIC SBCS code pages are as follows:
- " (0x7F, double quotation mark; string delimiter)
- , (0x6B, comma; column delimiter)

The default decimal point for ASCII data files is 0x2e (period).

The default decimal point for EBCDIC data files is 0x4B (period).

If the code page of the server is different from the code page of the client, specify the hex representation of nondefault delimiters. For example,

```
load from ... modified by chardel0x0C coldelX1e ...
```

The following information about support for double character delimiter recognition in DEL files applies to the export, import, and load utilities: Character delimiters are permitted within the character-based fields of a DEL file. This applies to fields of type CHAR, VARCHAR, LONG VARCHAR, or CLOB (except when lobsinfile is specified). Any pair of character delimiters found between the enclosing character delimiters is imported or loaded into the database.

**Examples**

The following example shows how to export information from the STAFF table in the SAMPLE database (to which the user must be connected) to myfile.ixf, with the output in IXF format:

```
export to myfile.ixf of ixf messages msgs.txt select * from staff
```

The next example shows how to export the information about employees in Department 20 from the STAFF table in the SAMPLE database (to which the user must be connected) to awards.ixf, with the output in IXF format:

```
export to awards.ixf of ixf messages msgs.txt select * from staff where dept = 20
```
The following example exports the contents of the ADDRESS table, including its primary key. Rows are ordered on output by column NAME, which improves but does not guarantee the chances of a similar order on disk when the IXF file is imported:

```
export to C:\IXF\ADDRESS.IXF of ixf select * from ADDRESS order by name;
```

Next is an example that shows how to export LOBs to a DEL file:

```
export to myfile.del of del lobs to mylobs lobfile lobs1, lobs2 modified by lobsinfile select * from emp_photo
```

The next example shows how to export LOBs to a DEL file, specifying a second directory for files that may not fit into the first directory:

```
export to myfile.del of del lobs to /db2exp1, /db2exp2 modified by lobsinfile select * from emp_photo
```

The following example shows how to export data to a DEL file, using a single quotation mark as the string delimiter, a semicolon as the column delimiter, and a comma as the decimal point. The same convention should be used when importing data back into the database:

```
export to myfile.del of del modified by chardel'' coldel; decpt, select * from staff
```

### FORCE APPLICATION

This command forces local users, remote users, or applications off the system.

**Authorization**

SYSADM or SYSCTRL.

**Syntax**

```
>>-FORCE APPLICATION----+ALL-------------------------------+--->
   |    .-,---------------------.      |
   |    V                       |      |
   '-(-----application-handle---+---)--'

>----+------------+--------------------------------------------><
'-MODE ASYNC-'```
Description

To preserve database integrity, only users who are idling or executing interruptible database operations can be terminated. Users creating a database cannot be forced.

After a FORCE has been issued, the database will still accept requests to connect. Additional forces may be required to completely force all users off (or you can connect to the database in EXCLUSIVE mode).

The database manager remains active so that subsequent database manager operations can be handled without the need for DB2START. DB2STOP cannot be executed during a force.

The following are the parameter descriptions for FORCE APPLICATION:

- **APPLICATION ALL** Causes all applications to be disconnected from the database.
- **APPLICATION application-handle** Lists the agent or agents to terminate. List the values using LIST APPLICATIONS.
- **MODE ASYNC** Does not wait for all specified users to terminate before returning; it returns when the function has been issued or an error (such as invalid syntax) is discovered.

Examples

The following example forces two users, with application-handle values of 41408 and 55458, to disconnect from the database:

```
force application (41408, 55458)
```

This next example shows how you can list applications, force them off, and then connect in EXCLUSIVE mode (for example, to perform an offline backup):

```
db2 list applications

Auth Id  Application    Appl.      Application Id
DB       # of           ---------- -------------- ----------
        Name           Handle
-------- -------------- ---------- ------------------------------
-------- -----
ADAMACHE db2bp.exe      2          *LOCAL.DB2.010625124601
SAMPLE   1
ADAMACHE javaw.exe      4          *LOCAL.DB2.010625124611
SAMPLE   1

db2 force application(2,4)

DB20000I  The FORCE APPLICATION command completed successfully.
DB21024I  This command is asynchronous and may not be effective
```
**GET ADMIN CONFIGURATION**

This command returns the values of individual entries in the database manager configuration file that are relevant to the DB2 Administration Server. The DB2 Administration Server is a special DB2 instance that enables remote administration of DB2 servers. The following database manager configuration parameters are displayed:

- **AGENT_STACK_SZ**
- **AUTHENTICATION**
- **DIAGLEVEL**
- **DIAGPATH**
- **DISCOVER**
- **DISCOVER_COMM**
- **FILESERVER**
- **IPX_SOCKET**
- **NNAME**
- **OBJECTNAME**
- **QUERY_HEAP_SZ**
- **SVCTCBNAME**
- **SYSADM_GROUP**
- **SYSCTRL_GROUP**
- **SYSSMFT_GROUP**
- **TPNAME**
- **TRUST_ALLCLNTS**
- **TRUST_CLNTAUTH**

**Note**

The **SVCTCBNAME** parameter, set by the installation program, cannot be modified by the user. The Administration Server service name is set to use the DB2 registered TCP/IP port (523). For more information about these parameters, see the GET DATABASE MANAGER CONFIGURATION command description later in this chapter.

**Authorization**

None.

**Syntax**

```
>>-GET ADMIN------+----- CONFIGURATION---------------------<
     +----- CONFIG-------+
     '----- CFG--------'
```

**Example**

The following is sample output from GET ADMIN CONFIGURATION.
Admin Server Configuration

Node type = Database Server with local and remote clients
Database manager configuration release level = 0x0900
Diagnostic error capture level (DIAGLEVEL) = 3
Notify Level (NOTIFYLEVEL) = 2
 Diagnostic data directory path (DIAGPATH) =
SYSADM group name (SYSADM_GROUP) =
SYSCTRL group name (SYSCTRL_GROUP) =
SYSMAINT group name (SYSMAINT_GROUP) =
Database manager authentication (AUTHENTICATION) = SERVER
Trust all clients (TRUST_ALLCLNTS) = YES
Trusted client authentication (TRUST_CLNTAUTH) = CLIENT
Agent stack size (AGENT_STACK_SZ) = 16
Query heap size (4KB) (QUERY_HEAP_SZ) = 250
NetBIOS Workstation name (NNAME) =
TCP/IP Service name (SVCENAME) = 523
APPC Transaction program name (TPNAME) =
IPX/SPX File server name (FILESERVER) =
IPX/SPX DB2 server object name (OBJECTNAME) =
IPX/SPX Socket number (IPX_SOCKET) = 87A2
Discovery mode (DISCOVER) = SEARCH
Discovery communication protocols (DISCOVER_COMM) = TCPIP

GET AUTHORIZATIONS

This command reports the authorities of the current user from values found in the database configuration file and the authorization system catalog view SYSCAT.DBAUTH. Direct authorities are acquired by explicit commands that grant the authorities to a user ID. Indirect authorities are based on authorities acquired by the groups to which a user belongs.

Authorization

None.

Syntax

```bash
>>-GET AUTHORIZATIONS------------------------------------------><
```

Example

The following is sample output from GET AUTHORIZATIONS:

```
Administrative Authorizations for Current User
Direct SYSADM authority = NO
Direct SYSCTRL authority = NO
```
Direct SYSMAINT authority = NO
Direct DBADM authority = YES
Direct CREATETAB authority = YES
Direct BINDADD authority = YES
Direct CONNECT authority = YES
Direct CREATE_NOT_FENC authority = YES
Direct IMPLICIT_SCHEMA authority = YES
Direct LOAD authority = YES
Indirect SYSADM authority = YES
Indirect SYSCTRL authority = NO
Indirect SYSMAINT authority = NO
Indirect DBADM authority = NO
Indirect CREATETAB authority = YES
Indirect BINDADD authority = YES
Indirect CONNECT authority = YES
Indirect CREATE_NOT_FENC authority = NO
Indirect IMPLICIT_SCHEMA authority = YES
Indirect LOAD authority = NO

**GET CLI CONFIGURATION**

This command lists the entire file, or a specified section of the db2cli.ini file. The db2cli.ini file is used as the DB2 Call Level Interface (CLI) configuration file.

The CLI also supports ODBC. The file keywords and values are used to modify the behavior of the DB2 CLI and the applications using it. The file is divided into sections, each section corresponding to a database alias name.

**Authorization**

None.

**Syntax**

```
>>> GET CLI-++CONFIGURATION++--------------------------------------+------>
                +-CONFIG--------- 'AT GLOBAL LEVEL--'
                '-CFG----------'

>>>>-------------------------------+--------------------------><
                'FOR SECTION--section-name--'
```
Description
The following are the parameter descriptions for GET CLI CONFIGURATION:

- **AT GLOBAL LEVEL**  Displays the default CLI configuration parameters in the LDAP directory. (This is valid only on Windows.)
- **FOR SECTION section-name**  Specifies the section for which you want to view keywords. The default is to show all sections.

Example
Get the CLI configuration for the section called sample:

```
get cli configuration for section sample
```

### GET CONNECTION STATE
This command displays the connection state. Possible states are the following:

- Connectable and connected
- Connectable and unconnected
- Unconnectable and connected
- Implicitly connectable (if implicit connect is available)

This command also returns information about the database connection mode (SHARE or EXCLUSIVE), and the alias and name of the database to which a connection exists.

Authorization
None.

Syntax
```
>> GET CONNECTION STATE
```

### GET DATABASE CONFIGURATION
GET DATABASE CONFIGURATION returns the values of individual entries in a specific database configuration file.

Authorization
None.
Syntax

```plaintext
>>-GET----+-DATABASE-+---+-CONFIGURATION--+-FOR----------------->
  '-DB-------'   +-CONFIG--------+
  '-CFG-----------'

>----database-alias--------------------------------------------><
```

Description

The default output for this command varies by operating system. In the following list, which explains some of the parameters, the parameters whose name appears in lowercase are maintained by the database manager and cannot be updated.

- **APP_CTL_HEAP_SZ** Determines the maximum size, in 4KB pages, for the application control heap. The heap shares information among agents working on behalf of the same application at a node in an EEE or an SMP system. If complex applications are being run, or the EEE configuration has a large number of nodes, the size of this heap should be increased.

- **APPLHEAPSZ** Specifies the size, in pages, of the application heap that is available for each individual agent.

- **AUDIT_BUF_SZ** Specifies the size, in 4KB units, of the buffer used when auditing the database.

- **AUTORESTART** Indicates whether the database manager will attempt to restart itself if the database connection is disrupted, or if the database is terminated abnormally.

- **AVG_APPLS** Indicates the average number of active applications. This parameter is not as important as MAXAPPLS.

- **backup_pending (Backup pending)** Specifies whether a backup is pending:
  - **NO** The database can be accessed.
  - **YES** The database must be backed up offline before it can be used.

- **BUFFPAGE** Specifies the default size, in pages, of any buffer pool you create. The buffer pool is used to store and manipulate data read in from the database. This parameter is used only when a buffer pool’s size has been explicitly set to –1 through CREATE BUFFERPOOL, ALTER BUFFERPOOL, or MIGRATE DATABASE. The size of the buffer pool is normally controlled through these SQL statements.

- **CATALOGCACHE_SZ** Controls the size, in pages, of the internal catalog cache (allocated from the dbheap), used by the SQL compiler to hold the
packed descriptors for commonly referenced objects such as tables and constraints.

- **CHNGPGS_THRESH** Specifies the percentage of changed pages above which the asynchronous page cleaners will be started, if they are not currently active.

- **codepage (Database code page)** Specifies the National Language code page of the database. For example, 0437 for USA.

- **codeset (Database code set)** Identifies the code set of the database.

- **COPYPROTECT (OS/2 only)** Enables the copy-protect attribute on database backup images.

- **country (Database country code)** Specifies the country code of the database.

- **database_consistent (Database is consistent)** Indicates whether the database data is consistent:
  - **NO** A transaction or some other task is pending on the database, and the data on disk is not consistent with data in the buffer pool at this point.
  - **YES** All transactions have been committed or rolled back, and all data is consistent.

- **database_level (database release level)** Identifies the release level of the database manager that can use the database.

- **DBHEAP** Specifies the size, in pages, of the database heap that is used to hold control information on all open cursors accessing the database. Both logbufsz and catalogcache_sz are allocated from the dbheap.

- **DFT_DEGREE** Specifies the degree of query parallelism (for example, 4 on a machine with four processors); also specifies the default for the CURRENT DEGREE special register and the DEGREE bind option.

- **DFT_EXTENT_SZ** Specifies the default extent size of table spaces (in pages).

- **DFT_LOADREC_SES** Specifies the default number of load recovery sessions. This is applicable only if roll-forward recovery is enabled.

- **DFT_PREFETCH_SZ** Specifies the default prefetch size of table spaces (in pages).

- **DFT_QUERYOPT** Specifies the query optimization class that is used to direct the optimizer when compiling SQL queries.

- **DFT_REFRESH_AGE** Specifies the default for the REFRESH AGE if the CURRENT REFRESH AGE special register is not specified. This parameter is used to determine whether summary tables are eligible to help optimize processing of a dynamic SQL query.

- **DFT_SQLMATHWARN** Specifies whether arithmetic errors and retrieval conversion errors are handled as warnings during SQL statement compilation (the value YES specifies that they are). This is similar to friendly arithmetic on DB2 for OS/390, which protects you from divide-by-zero overflow errors.
- **DISCOVER_DB** If set to DISABLE, prevents information about a database from being returned to a client when a discovery request is issued by the client against the server.

- **DL_EXPINT** Specifies the number of seconds for which a generated file access token is valid.

- **DL_NUM_COPIES** Specifies the number of additional copies of a file to be made in the archive server when a file is linked to the database. (This parameter is only for DB2 Data Links Manager.)

- **DL_TIME_DROP** Specifies the number of days files are retained on an archive server after a DROP TABLE, DROP DATABASE, or DROP TABLESPACE command. (This parameter is only for DB2 Data Links Manager.)

- **DL_TOKEN** Specifies the algorithm used in the generation of DATALINK file access control tokens. (This parameter is only for DB2 Data Links Manager.)

- **DL_UPPER** Specifies whether the file access control tokens use only uppercase letters or mixed case. (This parameter is only for DB2 Data Links Manager.)

- **DLCHKTIME** Specifies the time interval in milliseconds deadlock detection.

- **DYN_QUERY_MGMT** If set to ENABLE and the cost of the dynamic query exceeds the trap threshold for the user or group (as specified in the Query Patroller user profile table), this query will be trapped by DB2 Query Patroller.

- **ESTORE_SEG_SZ** Specifies the number of pages in each of the extended memory segments in the database. Extended memory supports read-only swapping from the buffer pool on AIX, Solaris, and Windows. Not required if 64-bit memory is supported (such as Windows AWE).

- **INDEXREC** Specifies when invalid indexes will be re-created. The default setting is SYSTEM, which uses the value of the database manager configuration parameter indexrec. Choices are the following:
  - SYSTEM(ACCESS)
  - SYSTEM(RESTART)
  - ACCESS
  - RESTART

- **INDEXSORT** Indicates whether sorting of index keys will occur during index creation.

- **LOCKLIST** Specifies the maximum storage, in pages, allocated to the lock list. When the lock list is exhausted, row locks will escalate to table locks.

- **LOCKTIMEOUT** Specifies the number of seconds that an application will wait to obtain a lock.

- **LOGBUFSZ** Specifies the number of pages used to buffer log records prior to writing them to disk. These are allocated from dbheap.
DB2: The Complete Reference

- **LOGFILSIZ** Specifies the amount of disk storage, in pages, allocated to log files used for data recovery. This parameter defines the size of each primary and secondary log file.

- **loghead (first active log file)** Specifies the name of the log file containing the head of the active log. The next log record that is written will start at the head of the active log.

- **logpath (path to log files)** Specifies the location of log files. Contains the current path being used for logging purposes.

- **LOGPRIMARY** Specifies the number of primary log files used for database recovery.

- **LOGRETAIN** Indicates whether the active log files are to be kept for roll-forward recovery or for use by the data replication Capture program.

- **log_retain_status (log retain for recovery status)** Indicates whether log files are retained for use in roll-forward recovery.

- **LOGSECOND** Specifies the number of secondary log files that can be used for database recovery.

- **MAXAPPLS** Specifies the maximum number of applications (both local and remote) that can connect to the database at once.

- **MAXFILOP** Specifies the maximum number of database files that an application can have open at one time.

- **MAXLOCKS** Specifies the maximum percentage of the lock list that any single application can use.

- **MINCOMMIT** Specifies the number of SQL commits that can be grouped for a given database. Grouping SQL commits permits faster writing to the log.

- **multipage_alloc** Specifies that multipage file allocation is used to improve insert performance. It applies only to SMS table spaces.

- **NEWLOGPATH** Specifies an alternate path to the recovery log files for a database. Because the newlogpath directory accepts only fully qualified directories, the absolute path must be specified.

- **NUM_DB_BACKUPS** Specifies the number of database backups to retain for a database. After the specified number of backups is reached, old backups are marked as expired in the recovery history file. When a backup is marked as expired, the physical backups can be removed. The next database backup will prune the expired entries from the history file.

- **NUM_ESTORE_SEGS** Specifies the number of extended storage memory segments available for use by the database.

- **NUM_FREQVALUES** Specifies the number of “most frequent values” that will be collected when the WITH DISTRIBUTION option is specified in RUNSTATS.
DB2 Command Line Processor Commands

- **NUM_IOCLEANERS**: Specifies the number of asynchronous page cleaners for a database.
- **NUM_IOSERVERS**: Specifies the number of I/O servers for a database. I/O servers are used on behalf of the database agents to perform prefetch I/O and asynchronous I/O by utilities such as backup and restore.
- **NUM_QUANTILES**: Controls the number of quantiles that will be collected for columns when the WITH DISTRIBUTION option is specified in RUNSTATS.
- **numsegs (default number of containers)**: Specifies the number of containers that will be created within the default SMS table spaces.
- **PCKCACHESZ**: Specifies the amount of memory to use for caching packages and dynamic SQL statements. The label (calculated) is displayed in the output for GET DATABASE CONFIGURATION if either of these is true:
  - The internal value is –1
  - MAXAPPLS*8 is less than 32. In this case, 32 is displayed with the label (calculated).
- **REC_HIS_RETENTN**: Specifies the number of days to retain historical information on backups.
- **release (database configuration release level)**: Indicates the release level of the database configuration file.
- **restore_pending (restore pending)**: Indicates whether a RESTORE PENDING status exists in the database.
- **rollfwd_pending (rollforward pending)**: Indicates whether a roll-forward recovery procedure is required for the database. The possible values are these:
  - **NO**: Neither the database nor any table space is in roll-forward pending state.
  - **DATABASE**: The database first needs to be rolled forward.
  - **TABLESPACES**: One or more table spaces in the database requires roll-forward recovery.
- **SEQDETECT**: Indicates whether sequential detection for a database is to be enabled or disabled.
- **SORTMAX**: Specifies the frequency with which soft checkpoints are taken, which is the same as the number of logs that are recovered after a crash.
- **SORTHEAP**: Specifies the number of private memory pages available for each sort in the application.
- **STAT_HEAP_SZ**: Specifies the statistics heap size (in pages), which is the maximum size of the heap used in creating and collecting all table statistics when distribution statistics are gathered.
- **STMTHEAP**: Specifies the heap size, in pages, that can be used for compiling SQL statements.
territory (database territory) Specifies the territory of the database.

TSM_MGMTCLASS Specifies how the server should manage the backup versions or archive copies of the objects being backed up. The management class is assigned from the TSM administrator. Once assigned, this parameter should be set to the management class name. When performing any TSM backup, the database manager uses this parameter to pass the management class to TSM.

TSM_NODENAME Overrides the default setting for the node name associated with TSM. The node name is needed when restoring a database that was backed up to TSM from another node.

TSM_OWNER Overrides the default for the owner associated with the TSM. The owner name is needed when restoring a database that was backed up to TSM from another node.

TSM_PASSWORD Overrides the default setting for the password associated with TSM. The password is needed when restoring a database that was backed up to TSM from another node.

USEREXIT Indicates whether a user exit function for archiving or retrieving log files can be called the next time the database is opened:

OFF Specifies that a user exit function cannot be called.

ON Specifies that a user exit function can be called.

user_exit_status (user exit for logging status) Specifies whether the user exit function can be called to store archive log files with OFF or ON:

OFF Specifies that the user exit function cannot be called to store archive log files.

ON Specifies that the user exit function can be called to store archive log files.

UTIL_HEAP_SZ Specifies the maximum amount of shared memory that can be used simultaneously by the utility heap (backup, restore, and load).

GET DATABASE MANAGER CONFIGURATION
This command returns the values of individual entries in the database manager configuration file.

Authorization
None.

Syntax

```plaintext
>>>GET--++DATABASE MANAGER---+---+CONFIGURATION+-----------><
  +--DB MANAGER-------+  +--CONFIG---------+
    'DBM------------'    'CFG-'----------'
```
Description

The following list explains some of the parameters:

- **AGENTPRI** Specifies the priority assigned to database manager processes and threads on a particular machine.
- **ASLHEAPSZ** Specifies the size (in pages) of the memory shared between a local client application and a database manager agent.
- **AUDIT_BUF_SZ** Specifies the size (in pages) of the buffer used when auditing the database.
- **AUTHENTICATION** Specifies how and where authentication of a user takes place. A value of CLIENT indicates that all authentication takes place at the client. If the value is SERVER, the user ID and password are sent from the client to the server so that authentication can take place at the server.
- **BACKBUFSZ** Specifies the size (in pages) of the buffer used when backing up the database, if the buffer size is not specified when calling the backup utility.
- **CATALOG_NOAUTH** Specifies whether users are able to catalog and uncatalog databases and nodes, or DCS and ODBC directories, without SYSADM authority. The default (0) for this parameter indicates that SYSADM authority is required. If this parameter is set to 1 (Yes), SYSADM authority is not required.
- **COMM_BANDWIDTH (EEE only)** Specifies the calculated value, in megabytes per second, used by the SQL optimizer to estimate the cost of performing certain operations between the database partition servers of an EEE database.
- **CONN_ELAPSE (EEE only)** Specifies the maximum wait time for a TCP/IP connection to be established between two nodes. If the attempt completes within the time specified by this parameter, communications are established. If it fails, another attempt is made to establish communications. If the connection is attempted the number of times specified by the `max_connretries` parameter and always times out, an error is returned.
- **CPUSPEED** Specifies a CPU speed (in milliseconds per instruction) that is used by the SQL optimizer to estimate the cost of performing certain operations. This value is set automatically when the database manager is installed, but it can be modified to model a production environment on a test system, or to assess the impact of upgrading hardware.
- **DATALINKS** Specifies whether Data Links support is enabled.
- **DFT_ACCOUNT_STR** Specifies the default accounting string.
- **DFT_MON_BUFPOOL** Specifies the default value of the snapshot monitor’s buffer pool switch.
- **DFT_MON_LOCK** Specifies the default value of the snapshot monitor’s lock switch.
DB2: The Complete Reference

- **DFT_MON_SORT** Specifies the default value of the snapshot monitor’s sort switch.
- **DFT_MON_STMT** Specifies the default value of the snapshot monitor’s statement switch.
- **DFT_MON_TABLE** Specifies the default value of the snapshot monitor’s table switch.
- **DFT_MON_UOW** Specifies the default value of the snapshot monitor’s unit of work (UOW) switch.
- **DFTDBPATH** Specifies the default database path. If no path is specified when a database is created, the database is created on the path specified by this parameter.
- **DIAGLEVEL** Specifies the diagnostic error capture level, which determines the severity of diagnostic errors recorded in the error log file (db2diag.log). The default is 3. A setting of 4 is appropriate for problem reproduction scenarios.
- **DIAGPATH** Specifies the fully qualified path for DB2 diagnostic information, where the db2diag.log file is written.
- **DIR_CACHE** Specifies whether directory cache support is used. If set to Yes, then database, node, and DCS directory files are cached in memory. This reduces connect costs by eliminating directory file I/O and minimizing the directory searches required to retrieve directory information.
- **DISCOVER** Defines the type of discovery request supported on a client or server. Discovery requests can be issued from the client configuration assistant or from control center tools. Specify SEARCH to support the DB2 client searching the network for DB2 databases. Specify KNOWN to support known discovery, in which the discovery request is issued against the administration server specified by the user. Specify DISABLE to disable the client or server from supporting any type of discovery request.
- **DISCOVER_COMM** Specifies the communications protocols that clients use to issue search discovery requests, and that servers use to listen for search discovery requests. If more than one protocol is required, separate them by commas:
  
  **TCPIP, NETBIOS**

- **DISCOVER_INST** Enables or disables client discovery of an instance.
- **DRDA_HEAP_SZ** Specifies the size, in pages, of the DRDA heap. This is used by the DRDA Application Server and by DB2 Connect.
**DB2 Command Line Processor Commands**

- **FCM_NUM_BUFFERS (EEE only)** Specifies the number of 4KB buffers used for internal communications (messages) among the nodes in an instance. These nodes communicate by using the Fast Communication Manager (FCM).

- **FCM_NUM_CONNECT** Specifies the number of FCM connection entries. Agents use connection entries to pass data among themselves.

- **FCM_NUM_RQB** Specifies the number of FCM request blocks. Request blocks are the media through which information passes between the FCM daemon and an agent.

- **FEDERATED** Specifies whether the instance can use nicknames to access data managed by other DB2 instances. When set to Yes, nicknames can be used. This parameter is used with DB2 Relational Connect to manage data on Oracle, Sybase, and SQL Server.

- **INDEXREC** Specifies when invalid database indexes should be re-created. This parameter is used if the database configuration parameter `indexrec` is set to SYSTEM. Possible values are ACCESS and RESTART.

- **INITDARI_JVM** Indicates whether each fenced stored procedure process will load the Java Virtual Machine (JVM) when starting. This parameter can reduce the initial startup time for fenced Java stored procedures, especially when used in conjunction with the `num_initdaris` parameter. This parameter could increase the initial load time for non-Java fenced stored procedures, however, because they do not need the JVM.

- **INTRA_PARALLEL** Specifies whether the database manager can use intrapartition parallelism. In an SMP environment, the default for this parameter is YES. In a non-SMP environment, the default for this parameter is NO. This parameter can be used on both partitioned and nonpartitioned database systems. Operations that can take advantage of parallel performance improvements when the value of this parameter is YES include database queries and index creation.

- **JAVA_HEAP_SZ** Specifies the maximum size of the heap that is used by the Java interpreter. For non-EEE database systems, one heap is allocated for the instance; for EEE database systems, one heap is allocated for each database partition server.

- **KEEPDARI KEEPDARI=YES** Indicates whether a stored procedure process should be kept loaded in memory after use. Doing so can improve performance for stored procedures that are called repeatedly, but this will waste memory for stored procedures that are seldom called.

- **MAX_CONNRETRIES (EEE only)** Specifies the number of connection retries that can be made to a node if an attempt to establish communication between two nodes fails because the value specified by the `conn_elapse` parameter is
reached (for example, the attempt to establish TCP/IP communication times out). If the value specified for this parameter is exceeded, an error is returned.

- **MAXCOORDAGENTS**: Specifies the maximum number of coordinating agents that can exist at one time on a node.

- **MAXLOGICAGENTS**: Specifies the maximum number of applications that can be connected to the instance. When the parameter is set to a value greater than the default (`max_coordagents`), the concentrator feature is activated.

- **MAX_QUERYDEGREE**: Specifies the maximum degree of parallelism used for any SQL statement executing on this instance of the database manager. An SQL statement will not use more than this number of parallel operations when the statement is executed. For an EEE system, this parameter applies to the degree of parallelism within a single node.

- **MAX_TIME_DIFF (EEE only)**: Specifies the maximum time difference, in minutes, that is permitted among the nodes listed in the `db2nodes.cfg` file. (Each node has its own system clock.)

- **MAXAGENTS**: Specifies the maximum number of database manager agents that can exist simultaneously on a node, regardless of which database is used.

- **MAXCAGENTS**: Specifies the maximum number of database manager agents that can be concurrently executing a database manager transaction. This cannot exceed the value of `maxagents`.

- **MAXDARI**: Specifies the maximum number of stored procedure processes that can reside at the database server. This cannot exceed the value of `maxagents`.

- **MON_HEAP_SZ**: Specifies the amount (in 4KB pages) of memory to allocate for database system monitor data in the database system monitor heap.

- **NUM_POOLAGENTS**: Specifies the maximum size of the agent pool.

- **NUMDB**: Specifies the maximum number of local databases that can be concurrently active.

- **QUERY_HEAP_SZ**: Specifies the maximum amount of memory (in pages) that can be allocated for the query heap. A query heap is used to store each query in the agent’s private memory.

- **release (database manager configuration release level)**: Specifies the release level of the configuration file.

- **RESTBUFSZ**: Specifies the size (in 4KB pages) of the buffer used when restoring the database, if the buffer size is not specified when calling the restore utility.

- **RESYNC_INTERVAL**: Specifies the time interval (in seconds) after which a Transaction Manager (TM) or a Resource Manager (RM) retries the recovery of any outstanding indoubt transactions found in the TM or the RM. This is
applicable when transactions are running with distributed unit of work (DUOW).

- **RQRIOLBK** Specifies the size (in bytes) of the client I/O block—the communication buffer between remote applications and their database agents on the database server.

- **SHEAPTHRES** Specifies the limit on the total amount of memory (in pages) available for sorting across the entire instance.

- **SPM_LOG_FILE_SZ** Specifies the sync point manager (SPM) log file size in 4KB pages.

- **SPM_LOG_PATH** Specifies the directory where the SPM logs are written. By default, the logs are written to the sqllib directory, which, in a high-volume transaction environment, can cause an I/O bottleneck. Use this parameter to have the SPM log files placed on a faster disk than the current sqllib directory. This allows for better concurrency among the SPM agents.

- **SPM_MAX_RESYNC** Specifies the number of simultaneous agents that can perform resync operations.

- **SPM_NAME** Specifies the name of the SPM. The *spm_name* must be defined in the system database directory and, if remote, in the node directory.

- **START_STOP_TIME (EEE only)** Specifies the number of minutes for all nodes to respond to START DATABASE MANAGER, STOP DATABASE MANAGER, or ADD NODE.

- **SVCENAME** Specifies the name used to update the database manager configuration file at the server. This value must be the same as the Connection Service name specified in the services file: for example, WINNT\system32\drivers\etc\services.

- **SYSADM_GROUP** Specifies the group name with System Administrative (SYSADM) authority for the database manager instance. This is the highest level of authority within the database manager, and it controls all database objects.

- **SYSCTRL_GROUP** Specifies the group name with System Controller (SYSCTRL) authority for the database manager instance. This level has privileges allowing operations affecting system resources, but not allowing direct access to data.

- **SYSMAINT_GROUP** Specifies the group name with System Maintenance (SYSMAINT) authority for the database manager instance. This level has authority allowing maintenance operations on all databases associated with an instance, but not allowing direct access to data.

- **TM_DATABASE** Specifies the TM database for each DB2 instance.

- **TP_MON_NAME** Specifies the transaction processing (TP) monitor product being used.
TRUST_ALLCLNTS Determines, with the trust_clntauth parameter, where users are validated for the database environment. By accepting the default for this parameter, all clients are treated as trusted clients. This means a level of security is available at the client, and that users can be validated at the client. Other options may be used to protect the server against certain clients based on their platform or database protocol.

TRUST_CLNTAUTH Determines, with the trust_allclnts parameter, where users are validated to the database environment. By accepting the default for this parameter, all users of trusted clients are validated at the client.

UDF_MEM_SZ for a fenced user-defined function (UDF) Specifies the default allocation for memory to be shared between the database process and the UDF. For an unfenced process, this parameter specifies the size of the private memory set. In both cases, this memory is used to pass data to a UDF and back to a database.

GET DATABASE MANAGER MONITOR SWITCHES
This command displays the status of the database system monitor switches. Monitor switches instruct the database system manager to collect database activity information. Each application using the database system monitor interface has its own set of monitor switches. (See the GET MONITOR SWITCHES command later in this chapter.) A database manager-level switch is on when any of the monitoring applications has turned it on. This command determines whether the database system monitor is currently collecting data for any monitoring application.

Authorization
SYSADM, SYSCRTL, or SYSMAINT.

Syntax

```
>>>GET--+-DATABASE MANAGER--+-MONITOR SWITCHES----------------->
  +--DB MANAGER--------+
    '--DBM-------------'

                           +-------------------------------<
                             +--AT NODE--nodenum--+    
    '--GLOBAL-------------'
```

Description
The following are the parameter descriptions for GET DATABASE MANAGER MONITOR SWITCHES:
AT NODE nodenum  Specifies the node for which the status of the database manager monitor switches are to be displayed.
GLOBAL  Returns an aggregate result for all nodes in an EEE database system.

GET INSTANCE
GET INSTANCE returns the value of the DB2INSTANCE environment variable.

Authorization
None.

Syntax

```
>>-GET INSTANCE------------------------------------------------><
```

GET MONITOR SWITCHES
This command displays the status of the database system monitor switches for the current session. Each application using the database system monitor interface has its own set of monitor switches. To display the database manager-level switches, use GET DATABASE MANAGER MONITOR SWITCHES.

Authorization
SYSADM, SYSCRTL, or SYSMAINT.

Syntax

```
>>-GET MONITOR SWITCHES-----------------------------------------><
  +-AT NODE--nodenum--+
  '-GLOBAL------------'
```

Description
The following are the parameter descriptions for GET MONITOR SWITCHES:

- AT NODE nodenum  Specifies the node for which the status of the monitor switches are to be displayed.
- GLOBAL  Returns an aggregate result for all nodes in an EEE database system.
GET SNAPSHOT

This command collects status information and formats the output for the user. The information returned represents a snapshot of the database manager when the command is issued.

In a partitioned database environment, this command can be invoked from any node in the db2nodes.cfg file. It acts only on that node or partition.

Authorization

SYSADM, SYSCRTL, or SYSMAINT.

Syntax

```
>>-GET SNAPSHOT FOR--------------------------------------------->

>-----+-------------------+------------------------------------><
```
Description
The following are the parameter descriptions for GET SNAPSHOT:

- DATABASE MANAGER Provides statistics for the active database manager instance.
- ALL DATABASES Provides general statistics for all active databases on the current instance.
- ALL APPLICATIONS Provides statistics for all active applications connected to a database on the current instance.
- ALL BUFFERPOOLS Provides statistics on buffer pool activity for all active databases.
- APPLICATION APPLID appl-id Provides information about only the application whose ID is specified. To get a specific application ID, use LIST APPLICATIONS.
- APPLICATION AGENTID appl-handle Provides information about only the application whose application handle is specified. The application handle is a 32-bit number uniquely identifying the application currently running. Use LIST APPLICATIONS to get a specific application handle.
- FCM FOR ALL NODES (EEE only) Provides FCM statistics for all nodes.
- LOCKS FOR APPLICATION APPLID appl-id Provides information about all locks held by the specified application, identified by application ID.
- LOCKS FOR APPLICATION AGENTID appl-handle Provides information about all locks held by the specified application, identified by application handle.
- ALL REMOTE_DATABASES Provides general statistics about all active remote databases on the current node.
- ALL REMOTE_APPLICATIONS Provides information about all active remote applications connected to the current node.
- ALL ON database-alias Provides general statistics and information about all applications, tables, table spaces, buffer pools, and locks for a specified database.
- DATABASE ON database-alias Provides general statistics for a specified database.
APPLICATIONS ON database-alias Provides information about all applications connected to a specified database.

TABLES ON database-alias Provides information about tables in a specified database. This includes only those tables that have been accessed since the TABLE recording switch was turned on.

TABLESPACES ON database-alias Provides information about table spaces for a specified database.

LOCKS ON database-alias Provides information about every lock held by each application connected to a specified database.

BUFFERPOOLS ON database-alias Provides information about buffer pool activity for the specified database.

REMOTE_DATABASES ON database-alias Provides general statistics about all active remote databases for a specified database.

REMOTE_APPLICATIONS ON database-alias Provides information about remote applications for a specified database.

DYNAMIC SQL ON database-alias Returns a point-in-time picture of the contents of the SQL statement cache for the database. This is useful if dynamic SQL is experiencing performance problems, you expect the SQL statement cache to help more than it does, and/or you want to capture samples of the dynamic SQL run by users.

WRITE TO FILE Stores snapshot results in a file at the server, as well as passing them back to the client. This command is valid only over a database connection. The snapshot data can then be queried through the table function SYSFUN.SQLCACHE_SNAPSHOT over the same connection on which the call was made.

DCS Obtains statistics for database connections made through DB2 Connect. The keyword following DCS will request statistics on one of the following:

- A specific DRDA application currently running on DB2 Connect
- All DRDA applications
- All DRDA applications currently connected to a specific DRDA server
- A specific DRDA server database
- All DRADA server databases

AT NODE nodenum Returns results for the node specified.

GLOBAL Returns an aggregate result for all nodes in a partitioned database system.

HELP Invokes help from the Information Center. This command is not available on UNIX.
Authorization

None.

Syntax

```
>>>HELP+----------------------------------+
     '-character-string-'<
```

Description

The HELP character string can be any SQL or DB2 command, or any other item listed in the Information Center.

**IMPORT**

IMPORT inserts data from an external file with a supported file format into a table or view. LOAD is faster, but has these drawbacks:

- LOAD does not add data at the hierarchy (subtable) level.
- LOAD is not logged, so log-based operations like replication will not work on LOADed rows.
- LOAD locks the entire table space.
- LOAD will not fire triggers.

IMPORT inserts rows sequentially, so INSERT triggers can fire based on IMPORTed data.

Authorization

To use the IMPORT command using the INSERT option requires one of the following:

- SYSADM or DBADM
- CONTROL privilege on each participating table or view
- INSERT and SELECT privileges on each participating table or view

To use the IMPORT command to import to an existing table by using the INSERT_UPDATE, REPLACE, or the REPLACE_CREATE option requires one of the following:

- SYSADM or DBADM
- CONTROL privilege on the table or view
Syntax

```
> IMPORT FROM <filename> OF <filetype>  
|  |   
| V |   
| "LOBS FROM <lob-path>"  
|  |   
| V |   
| "MODIFIED BY <filetype-mod>"  
|  |   
| V |   

"METHOD="L(<column-start column-end>)"  
|  |   
| V |   
| "NULL INDICATORS= (<null-indicator-list>="")"  
|  |   
| V |   
| +N(<column-name>)  
|  |   
| V |   
| "P=<column-position>

"COMMITCOUNT=n"  
|  |   
| V |   

"RESTARTCOUNT=n"

"MESSAGES=<message-file>"

"INSERT"  
|  |   
| V |   
| "INSERT_UPDATE="  
|  |   
| V |   
| "REPLACE="  
|  |   
| V |   
| "REPLACE_CREATE="  
|  |   
| V |   
| "CREATE="  
|  |   
| V |   
| "IND=<table-name>"  
|  |   
| V |   

"<insert-column-hierarchy> | tbspace-specs |"
```
DB2 Command Line Processor Commands

```plaintext
V
| "{(---insert-column---)}" |
| "hierarchy description ---AS ROOT TABLE--" |
| "UNDER---sub-table-name--" |
>
"DATALINK SPECIFICATION--- datalink-spec ---" |
hierarchy description

"ALL TABLES---" |
| sub-table-list |
| "IN-" |
> "HIERARCHY---STARTING---sub-table-name---" |
sub-table-list
V
| (---sub-table-name---) |
| "-,---" |
| V |
| "{(---insert-column---)}" |
traversal-order-list
V
| (---sub-table-name---) |
subspace-specs
| "IN---subspace-name---" |
| "INDEX IN---subspace-name---" |
| "LONG IN---subspace-name---" |
datalink-spec
V
| (---) |
| "DL_LINKTYPE URL--- DL_URL_REPLACE_PREFIX---"prefix"--- |
| "DL_URL_DEFAULT_PREFIX---"prefix"--- |
```
Description

IMPORT adds rows to the target table by using the SQL INSERT statement; it issues one INSERT statement for each row of data in the input file. If an INSERT statement fails, one of two actions results:

- If subsequent INSERT statements are likely to succeed, a warning message is written to the message file and processing resumes.
- If subsequent INSERT statements are likely to fail and there is potential for database damage, an error message is written to the message file, and the import stops.

IMPORT automatically COMMITs after the old rows are deleted during a REPLACE or a REPLACE_CREATE operation. Therefore, if the IMPORT command fails, or if the application interrupts the database manager after the table object is truncated, all of the old data is lost. Ensure that the old data is no longer needed before using these options.

Be sure to complete all table operations and release all locks before starting an import. Do this by issuing a COMMIT after closing all cursors opened WITH HOLD, or by issuing a ROLLBACK.

If the log becomes full during a CREATE, REPLACE, or REPLACE_CREATE operation, IMPORT performs an automatic COMMIT on inserted records. If the system fails, or the application interrupts the database manager after an automatic COMMIT, a table with partial data remains in the database. Use the REPLACE or the REPLACE_CREATE option to rerun the whole import, or use INSERT with the RESTARTCOUNT parameter set to the number of rows successfully imported.

By default, automatic COMMITs are not performed for the INSERT or the INSERT_UPDATE option. They are, however, performed if the COMMITCOUNT parameter is not zero. A full log results in a ROLLBACK. Whenever import COMMITs, two messages are written to the message file: one indicates the number of records to be committed, and the other is written after a successful COMMIT. When restarting the import after a failure, specify the number of records to skip, as determined from the last successful COMMIT.

IMPORT accepts input data with minor incompatibility problems (for example, character data can be imported by using padding or truncation, and numeric data can be imported by using a different numeric data type), but data with major incompatibility problems is not accepted.
Referential constraints and foreign-key definitions are not preserved when creating tables from PC/IXF files. (Primary key definitions are preserved if the data was previously exported by using SELECT * to select all rows.)

Importing to a remote database requires enough disk space on the server for a copy of the input data file, the output message file, and potential growth in the size of the database. If an import is run against a remote database, and the output message file is very long (more than 60KB), the message file returned to the user on the client may be missing messages from the middle of the import. The first 30KB of message information and the last 30KB of message information are always retained.

The data in ASC (ASCII) and DEL (delimited ASCII) files is assumed to be in the code page of the client application performing the import. PC/IXF files, which allow for different code pages, are recommended when importing data in different code pages. If the PC/IXF file and import use the same code page, processing occurs like for a regular application. If the two differ, and the FORCEIN option is specified, the IMPORT command assumes that data in the PC/IXF file has the same code page as the application performing the import. This occurs even if there is a conversion table for the two code pages. If the two differ, the FORCEIN option is not specified, and there is a conversion table, then all data in the PC/IXF file will be converted from the file code page to the application code page.

For table objects on an 8KB page close to the limit of 1012 columns, import of PC/IXF data files may cause DB2 to return an error if the maximum size of an SQL statement is exceeded. This situation can occur only if the columns are of type CHAR, VARCHAR, or CLOB. The restriction does not apply to the import of DEL or ASC files. If PC/IXF files are being used to create a new table, an alternative is to dump the source table's DDL by using DB2LOOK, and then run the statement through the CLP.

DB2 Connect can be used to import data to DRDA servers such as DB2 for OS/390, DB2 for VM and VSE, and DB2 for OS/400. Only the PC/IXF import (INSERT option) is supported.

IMPORT can recover a table previously exported to a PC/IXF file. The table returns to the state it was in when exported. Data cannot be imported to a system table, a declared temporary table, or a summary table. Views cannot be created through IIMPORT.

The following are the parameters for the IMPORT command:

- **ALL TABLES** This is the default and is relevant only for hierarchy subtables. It indicates to IMPORT into all subtables of a hierarchy.
- **AS ROOT TABLE** Creates one or more subtables as a standalone table hierarchy.
- **COMMITCOUNT n** COMMITs after n rows are imported.
- **CREATE** Creates the table definition and inserts the rows. If rows were exported from a DB2 table, subtable, or hierarchy, indexes are created. If this option operates on a hierarchy, and data was exported from DB2, a type hierarchy will also be created. This option can be used only with IXF files.
DATALINK SPECIFICATION Defines column specifications for DATALINK columns. For each DATALINK column, there can be one column specification enclosed by parentheses—each column specification consists of one or more DL_LINKTYPE, prefix, and a DL_URL_SUFFIX specification. The prefix specification can be either DL_URL_REPLACE_PREFIX or DL_URL_DEFAULT_PREFIX. This is the DATALINK column specification. There can be as many DATALINK column specifications as the number of DATALINK columns defined in the table. The order of specifications follows the order of DATALINK columns found within the insert-column list, or within the table definition (if an insert-column list is not specified).

DL_LINKTYPE Matches the LINKTYPE of the column definition. Thus, DL_LINKTYPE URL is acceptable if the column definition specifies LINKTYPE URL.

DL_URL_DEFAULT_PREFIX “prefix” Acts as the default prefix for all DATALINK values within the same column. In this context, prefix refers to the scheme host port part of the URL specification. (For DFS, prefix refers to the scheme cellname filespace-junction part of the URL specification.) Typical prefixes are the following:

- "http://server"
- "file://server"
- "file:"
- "http://server:80"
- "dfs://.../cellname/fs"

If no prefix is found in a column’s data, and a default prefix is specified with DL_URL_DEFAULT_PREFIX, the default prefix is prefixed to the column value (if not NULL). For example, if DL_URL_DEFAULT_PREFIX specifies the default prefix “http://toronto”, the column input value “/x/y/z” is stored as “http://toronto/x/y/z”. The column input value “http://coyote/a/b/c” is stored as “http://coyote/a/b/c”. The column input value NULL is stored as NULL.

DL_URL_REPLACE_PREFIX “prefix” Specifies the prefix for all non-NULL column values. If this is specified, it is used for loading or importing data previously generated by export, when you want to globally replace the hostname in the data with another hostname. If a column value has a prefix, this will replace it. If a column value has no prefix, the prefix specified by DL_URL_REPLACE_PREFIX is prefixed to the column value. For DFS, the prefix refers to the scheme cellname filespace-junction part of the URL specification. For example, if DL_URL_REPLACE_PREFIX specifies the prefix “http://toronto”, the column input value “/x/y/z” is stored as “http://toronto/x/y/z”. The column input value “http://coyote/a/b/c” is stored as “http://toronto/a/b/c”. Note that “toronto” replaces “coyote”. The column input value NULL is stored as NULL.
DB2 Command Line Processor Commands

- **DL_URL_SUFFIX “suffix”** Specifies a suffix to be appended to every non-NULL column value for the column.
- **FROM filename** Specifies the file containing the data to be imported. If the path is omitted, the current working directory is used.
- **HIERARCHY** Specifies that hierarchical data will be imported.
- **IN tablespace-name** Specifies the table space in which the table will be created. It must exist as a REGULAR table space. If no other table space is specified, all table parts are stored in this table space. If this clause is not specified, the table is created in a table space created by the authorization ID. If none is found, the table is placed into the default table space USERSPACE1. If USERSPACE1 has been dropped, table creation fails.
- **INDEX IN tablespace-name** Specifies the table space in which any indexes on the table will be created. This option is allowed only when the primary table space specified in the IN clause is a DMS table space. The specified table space must exist, and must be a REGULAR DMS table space. Specifying which table space will contain an index can be done only when the table is created.
- **insert-column** Specifies the name of a column in the table or the view into which data will be inserted.
- **INSERT** Adds the imported data to the table without changing the existing table data.
- **INSERT_UPDATE** Adds rows of imported data to the target table, or updates existing rows (of the target table) with matching primary keys.
- **INTO table-name** Specifies the table into which the data is to be imported. This table cannot be a system table, a declared temporary table, or a summary table. One can use an alias for INSERT, INSERT_UPDATE, or REPLACE. A qualified table name is in the form: schema.tablename. The schema is the user name under which the table was created.
- **LOBS FROM lob-path** Specifies one or more paths that store LOB files. The names of the LOB data files are stored in the main data file (ASC, DEL, or IXF), in the column that will be loaded into the LOB column. This option is ignored if the lobsinfile modifier is not specified.
- **LONG IN tablespace-name** Specifies the table space in which the values of any long columns (LONG VARCHAR, LONG VARGRAPHIC, LOB data types, or distinct types with any of these as source types) are stored. This option is allowed only if the primary table space specified in the IN clause is a DMS table space. The table space must exist, and it must be a LONG DMS table space.
- **MESSAGES message-file** Defines the destination for warning and error messages that occur during an import. If the file already exists, IMPORT appends the information. If the complete path to the file is not specified,
IMPORT uses the current directory and the default drive as the destination. If message-file is omitted, the messages are written to standard output.

- **METHOD** Explains how to specify columns for IMPORT.
  - **L** Identifies the start and end column numbers from which to import data. A column number is a byte offset from the beginning of a row of data. It is numbered starting from 1. L can be used only with ASC files, and it is the only valid option for that file type.
  - **N** Names the columns to import. Used with IXF files.
  - **P** Identifies the indexes (numbered from 1) of the input data fields to be imported. This method can be used only with IXF or DEL files. This is the only valid option for the DEL file type.

- **MODIFIED BY filetype-mod** Specifies additional options. (See the “Valid File Type Modifiers (Import)” section.)

- **NULL INDICATORS null-indicator-list** Provides a comma-separated list of positive integers specifying the column number of each null indicator field. This parameter can be used only when METHOD L is specified and the input file is an ASC file. The column number is the byte offset of the null indicator field from the beginning of a row of data. There must be one entry in the null indicator list for each data field defined in the METHOD L parameter. A column number of zero indicates that the corresponding data field always contains data. A value of Y in the nullindicator column specifies that the column data is null, and any other character specifies that the column data is not null, and that column data specified by the METHOD L option will be imported. The null indicator character can be changed by using the MODIFIED BY option. (See the description of the nullindchar modifier in the “Valid File Type Modifiers (Import)” section later in this chapter.)

- **OF filetype** Specifies the format of the data in the input file:
  - **ASC** (nondelimited ASCII format)
  - **DEL** (delimited ASCII format), which is used by a variety of database manager and file manager programs.
  - **WSF** (work sheet format), which is used by programs such as Lotus 1-2-3.
  - **IXF** (integrated exchange format, PC version), which means it was exported from the same or another DB2 table. An IXF file also contains the table definition and definitions of any existing indexes, except when columns are specified in the SELECT statement.

- **REPLACE** Deletes all existing data from the table by truncating the data object, and inserts the imported data. The table index definitions are not changed. This parameter can be used only if the table exists. It is not valid for tables with DATALINK columns. If this parameter is used when moving data between
hierarchies, only the data for an entire hierarchy, not individual subtables, can be replaced.

IMPORT with REPLACE can also be used to delete all rows from a table without logging. Do this by importing an IXF file that contains no rows. LOAD can also do this, but LOAD locks the entire table space and IMPORT does not. LOAD must be used if the target table has referential integrity dependencies or summary tables defined on it. If you plan to use LOAD a lot, put each table that gets LOADed (or uses LOAD to be emptied) in its own table space, or use IMPORT with REPLACE.

**REPLACE_CREATE** Replaces or creates a table, depending on whether the table already exists. This parameter can be used only with IXF files. It is not valid for tables with DATALINK columns. If this option is used when moving data between hierarchies, only the data for an entire hierarchy, not individual subtables, can be replaced. REPLACE_CREATE allows you to replace an existing table or create a new table, depending on whether the table previously existed:

- If the table exists, REPLACE_CREATE deletes all existing data from the table by truncating the data object, and inserts the imported data without changing the table definition or the index definitions.
- If the table does not exist, REPLACE_CREATE creates the table and index definitions, as well as the row contents.

**RESTARTCOUNT n RESTARTCOUNT** Allows you to resume an interrupted IMPORT at a specific row count within the table. It starts an import at record n + 1. The first n records are skipped.

**STARTING subtable-name** Requests hierarchy tables in the default order, starting from subtable-name. For PC/IXF files, the default order is the order stored in the input file. The default order is the only valid order for the PC/IXF file format.

**subtable-list** Specifies a list of subtable names for typed tables with the INSERT or the INSERT_UPDATE option. Used to indicate the subtables that are targets of the import. For example, RETIREES and PART_TIMERS could be subtypes of the EMPLOYEE table.

**traversal-order-list** Specifies a list of subtable names for typed tables with the INSERT, INSERT_UPDATE, or the REPLACE option. Indicates the traversal order of the importing subtables in the hierarchy.

**UNDER subtable-name** Identifies a parent table for creating one or more subtables.

Valid File Type Modifiers (Import)
File type modifiers vary depending on whether the input file is IXF, WSF, ASCII, or delimited ASCII (DEL).
Import File Type Modifiers for All File Formats

The following are the valid import file type modifiers for all file formats:

- **compound=x Compound** Allows INSERT statements run by IMPORT to be grouped together. \( x \) is a number between 1 and 100 inclusive. This modifier uses nonatomic compound SQL to insert the data, and \( x \) statements will be attempted each time. If specified, and the transaction log is not sufficiently large, the IMPORT will fail. The transaction log must be large enough to accommodate either the number of rows specified by COMMITCOUNT, or the number of rows in the data file if COMMITCOUNT is not specified. Use COMMITCOUNT to avoid transaction log overflow. This modifier is incompatible with INSERT_UPDATE mode, hierarchical tables, and the following modifiers:
  - usedefaults
  - identitymissing
  - identityignore
  - generatedmissing
  - generatedignore

- **generatedignore** Makes import ignore all data-generated columns in the data file. This results in all values for the generated columns being generated by import. generatedignore cannot be used with generatedmissing.

- **generatedmissing** Indicates to the IMPORT command that the input data file contains no data and no NULLs for generated columns. Import generates a value for each row. generatedmissing cannot be used with generatedignore.

- **identityignore** Specifies that IMPORT should ignore identity column data in the data files. Import generates all identity values for both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. Therefore, for GENERATED ALWAYS columns, no rows will be rejected. identityignore cannot be used with identitymissing.

- **identitymissing** Specifies that IMPORT should assume that the input data file contains no data for the identitycolumn (not even NULLs), and should generate a value for each row with both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. identitymissing cannot be used with identityignore.

- **lobsinfile lobsinfile** Lets you tell IMPORT that CLOB and BLOB data is found in files on the hard disk. It specifies the path to the files containing LOB values.

- **no_type_id** Used only when importing into a single subtable. It is typically used to export data from a regular table, and then to invoke an import operation (using no_type_id) to convert the data into a single subtable.
**nodefaults** Specifies that defaults not be imported into target table columns unless they are identified in source columns. Without this option, if a source column for one of the target table columns is not specified, one of the following occurs:

- If a default can be specified for a column, import assigns the default.
- If the column is nullable and a default is not specified, a NULL is assigned.
- If the column is not nullable and a default cannot be specified, an error is returned, and import stops.

**Usedefaults** Specifies that defaults will be assigned when a source column for a target table column is specified, but contains no data for some rows. Examples of missing data are the following:

- For DEL files, two commas (,,) are specified for the column.
- For ASC files, the NULL indicator is set to yes for the column.
- For DEL/ASC/WSF files, a row that does not have enough columns, or is not long enough for the original specification has missing data.

Without Usedefaults, if a source column contains no data for a row instance, one of the following occurs:

- If the column is nullable, a NULL is loaded.
- If the column is not nullable, import rejects the row.

**Import File Type Modifiers for the ASCII File Format** The following are the valid import file type modifiers for ASCII file formats (ASC/DEL):

- **dateformat=“x”** x is the date format in the source file. Valid date elements are as follows:

  - **YYYY Year** Four digits ranging from 0000–9999
  - **M Month** One or two digits ranging from 1–12
  - **MM Month** Two digits ranging from 1–12; mutually exclusive with M
  - **D Day** One or two digits ranging from 1–31
  - **DD Day** Two digits ranging from 1–31; mutually exclusive with D
  - **DDD Day of the year** Three digits ranging from 001–366; mutually exclusive with D, DD, M and MM

  A default of 1 is assigned for each element that is not specified. Examples of date formats are D-M-YYYY, MM.DD.YYYY, and YYYYDDD.

- **implieddecimal** Specifies that the location of an implied decimal point will be determined by the column definition. For example, the value 12345 is loaded into a DECIMAL(8,2) column as 123.45, not 12345.00.
noeofchar Specifies that the optional end-of-file character x’1A’ will not be recognized as the end of file. Processing continues as if it were a normal character.

timeformat="x" Specifies the format of the time in the source file. Valid time elements are as follows:

- **H Hour** One or two digits ranging from 0–12 for a 12-hour system, and 0–24 for a 24-hour system
- **HH Hour** Two digits ranging from 0–12 for a 12-hour system, and 0–24 for a 24-hour system; mutually exclusive with H
- **M Minute** One or two digits ranging from 0–59
- **MM Minute** Two digits ranging from 0–59; mutually exclusive with M
- **S Second** One or two digits ranging from 0–59
- **SS Second** Two digits ranging from 0–59; mutually exclusive with S
- **SSSS Second of the day after midnight** Five digits ranging from 00000–86399; mutually exclusive with other time elements
- **TT Meridian indicator** (A.M. or P.M.)

A default of 0 is assigned for each element that is not specified. Examples of time formats are HH:MM:SS, HH.MM TT, and SSSSS.

timestampformat="x" Specifies the format of the timestamp in the source file. Valid timestamp elements are as follows:

- **YYYY Year** Four digits ranging from 0000–9999
- **M Month** One or two digits ranging from 1–12
- **MM Month** Two digits ranging from 1–12; mutually exclusive with M, month
- **D Day** One or two digits ranging from 1–31
- **DD Day** Two digits ranging from 1–31; mutually exclusive with D
- **DDD Day of the year** Three digits ranging from 001–366; mutually exclusive with other day or month elements
- **H Hour** One or two digits ranging from 0–12 for a 12-hour system, and 0–24 for a 24-hour system
- **HH Hour** Two digits ranging from 0–12 for a 12-hour system, and 0–24 for a 24-hour system; mutually exclusive with H
- **M Minute** One or two digits ranging from 0–59
- **MM Minute** Two digits ranging from 0–59; mutually exclusive with M, minute
- **S Second** One or two digits ranging from 0–59
- **SS Second** Two digits ranging from 0–59; mutually exclusive with S
SSSSS Second of the day after midnight  Five digits ranging from 00000–86399; mutually exclusive with other time elements

UUUUUU Microsecond  Six digits ranging from 000000–999999

TT Meridian indicator  (A.M. or P.M.)

A default of 1 is assigned for unspecified YYYY, M, MM, D, DD, or DDD elements. A default of 0 is assigned for all other unspecified elements. Following is an example of a timestamp format:

"YYYY/MM/DD HH:MM:SS.UUUUUU"

The following example illustrates how to import data containing user-defined date and time formats into a table called SCHEDULE:

import from delfile2 of del
modified by timestampformat="yyyy.mm.dd hh:mm tt" insert into schedule

Import File Type Modifiers for the ASC File Format  The following are the valid import file type modifiers for the ASC (nondelimited ASCII) file format:

- nochecklengths  Specifies that import will try to add rows even when source data is wider than target columns. Such rows can be successfully imported if code-page conversion causes the source data to shrink; for example, four-byte EUC data in the source could shrink to two-byte DBCS data in the target and require half the space. This option is particularly useful when you know that the source data will fit in all cases despite mismatched column definitions.

- nullindchar=x  Changes the character denoting a NULL value to x (x is a single character). The default value of x is Y.

- reclen=x  Specifies how many characters are to be read for each row. x is an integer with a maximum value of 32 767. x characters are read for each row, and a newline character is not used to indicate the end of the row.

- striptblanks  Truncates trailing blank spaces when importing data into variable-length fields. If striptblanks is not specified, blank spaces are kept. In the following example, striptblanks causes import to truncate trailing blank spaces:

import from myfile.asc of asc modified by striptblanks method l (1 10, 12 15) messages msgs.txt insert into staff

This option cannot be specified together with striptnulls. These are mutually exclusive options.
striptnulls  Truncates trailing NULLs (0x00 characters) when loading data into variable-length fields. If striptnulls is not specified, NULLs are kept. This option cannot be specified together with striptblanks.

Import File Type Modifiers for the DEL File Format The following are the valid import file type modifiers for the DEL (Delimited ASCII) file format:

- **chardelx** Specifies a character to be used in place of double quotation marks to enclose a character string. *x* is a single character string delimiter. The default is a double quotation mark (".").
  
The single quotation mark (') can also be specified as a character string delimiter. In the following example, chardel' causes import to interpret any single quotation mark (') it encounters as a character string delimiter:

  ```
  import from myfile.del of del modified by chardel''
  method p (1, 4) insert into staff (id, years)
  ```

- **coldelx** Specifies a character to be used instead of a comma to signal the end of a column. *x* is a single character column delimiter, and the default is a comma (,). In the following example, coldel; causes import to interpret any semicolon (;) it encounters as a column delimiter:

  ```
  import from myfile.del of del modified by coldel;
  messages msgs.txt insert into staff
  ```

- **datesiso** Causes all date data values to be imported in ISO date format.

- **decplusblank** Causes positive decimal values to be prefixed with a blank space instead of a plus sign (+). The default is to prefix positive decimal values with a plus sign.

- **decptx** Specifies a character to be used in place of a period as a decimal point character. *x* is a single character substitute for the default, a period (.), as a decimal point character. In the following example, decpt; causes import to interpret any semicolon (;) it encounters as a decimal point:

  ```
  import from myfile.del of del modified by chardel'
  decpt; messages msgs.txt insert into staff''
  ```

- **delprioritychar** Specifies the default priority for delimiters:
  - record delimiter
  - character delimiter
  - column delimiter

  This modifier is for applications using the following order of priority: character delimiter, record delimiter, column delimiter.
The following is the syntax:

```
import ... modified by delprioritychar ...
```

For example, suppose you are given the following DEL data file:

```
"Smith, Joshua",4000,34.98<row delimiter>
"Vincent,<row delimiter>, is a manager", ...
... 4005,44.37<row delimiter>
```

With the delprioritychar modifier specified, there will be only two rows in this data file. The second `<row delimiter>` will be interpreted as part of the first data column of the second row, while the first and the third `<row delimiter>` are interpreted as actual record delimiters. If this modifier is not specified, there will be three rows in this data file, each delimited by a `<row delimiter>`.

- **ddlex**: Specifies a character to be used as the interfield separator for a DATALINK value in place of the default, a semicolon(`;`). `x` is a single character DATALINK delimiter, and it must not be the same character specified as the row, column, or character string delimiter. The interfield separator is needed because a DATALINK value may have more than one subvalue.

- **keepblanks**: Preserves leading and trailing blanks in each field of type CHAR, VARCHAR, LONG VARCHAR, or CLOB. Without this option, all leading and trailing blanks that are not inside character delimiters are removed, and a NULL is inserted into the table for all blank fields.

- **nodoubledel**: Suppresses recognition of double-character delimiters. For more information, see Delimiter Restrictions in the description of the EXPORT command.

### Import File Type Modifiers for the IXF File Format

The following are the valid import file type modifiers for the IXF file format:

- **forcein**: Causes import to accept data with code-page mismatches, and to suppress translation between code pages. It also causes fixed-length target fields to be checked to verify that they are large enough for the data. If nochecklengths is specified, no checking is done, and an attempt is made to import each row.

- **indexixf**: Causes import to drop all indexes currently defined on the existing table, and to create new ones from the index definitions in the PC/IXF file. This option can be used only when the contents of a table are replaced. It cannot be used with a view, or when an insert-column is specified.

- **indexschema=schema**: Uses the identified schema for the index name during index creation. If schema is not specified (but indexschema is specified), import
uses the connection user ID. If the keyword is not specified, import uses the schema in the IXF file.

- **nochecklengths** Causes import to attempt importing each row, even if the source data has a column definition that exceeds the size of the target table column. Such rows can be successfully imported if code-page conversion causes the source data to shrink; for example, four-byte EUC (Extended UniCode) data in the source could shrink to two-byte DBCS data in the target and require half the space. This option is particularly useful if you know that the source data will fit in all cases, despite mismatched column definitions.

### Examples

The following example imports all rows in myfile.ixf into the STAFF table:

```sql
import from myfile.ixf of ixf messages msg.txt insert into staff
```

The next example imports the columns in T3.IXF in IXF format into column c1 in table t3:

```sql
import from C:\TMP\T3.IXF OF IXF METHOD n (c1) insert into t3
```

The next example deletes the contents of the employee table without logging. This takes two steps. The first step is to create an IXF file with a SELECT that retrieves no rows:

```sql
export to c:\tmp\testvar.ixf of ixf
select * from testvar where col1 in (0) and col1 not in (0)
```

The next step is to import the empty IXF file:

```sql
import from c:\ixf\testvar.ixf of ixf replace into testvar
```

The following example shows how to import the MOVIE TABLE table from the input file delfile1, which has data in the DEL format. MOVIE TABLE has four columns: actorname VARCHAR(n), description VARCHAR(m), url_making_of DATALINK (with LINKTYPE URL), and url_movie DATALINK (with LINKTYPE URL).

```sql
import from delfile1 of del modified by dldel |
insert into movietable
(actorname, description, url_making_of, url_movie)
data linkage specification (dl_url_default_prefix "http://narang"),
```
The following example imports into a table with an identity column:

**TABLE1** has 4 columns:
- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

**TABLE2** is the same as **TABLE1**, except that C2 is a GENERATED ALWAYS identity column:

**TABLE2**:
- 1 VARCHAR(30)
- C2 INT GENERATED ALWAYS AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

The data records in **DATAFILE1** (DEL format) follow:

"Liszt"
"Hummel",187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I

The data records in **DATAFILE2** (DEL format) follow:

"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I

The following command generates identity values for rows 1 and 2 because no identity values are supplied in **DATAFILE1** for those rows. Rows 3 and 4, however, are assigned the user-supplied identity values of 100 and 101, respectively.

```
import from datafile1.del of del replace into table1
```

To import **DATAFILE1** into **TABLE1** so that identity values are generated for all rows, issue this command:

```
import from datafile1.del of del method P(1, 3, 4) replace into table1 (c1, c3, c4)
```
To import DATAFILE2 into TABLE1 so that identity values are generated for each row, issue this command:

```
import from datafile2.del of del replace into table1 (c1, c3, c4)
```

### INITIALIZE TAPE

DB2 for Windows supports backup and restore to streaming tape devices. Use this command for tape initialization. This command is available only on Windows NT and Windows 2000.

#### Authorization

None.

#### Syntax

```
>>-INITIALIZE TAPE-------------------------<
   '-ON--device--'
   '-USING--blksize--'
```

#### Description

The following are the parameter descriptions for INITIALIZE TAPE:

- **ON device** Specifies a valid tape device name. The default is `\\.\TAPE0`.
- **USING blksize** Specifies the block size for the device, in bytes. The device is initialized to use the block size specified if the value is within the supported range of block sizes for the device. The buffer size specified in backing up and restoring the database must be divisible by the block size specified here. If a value for this parameter is not specified, the device is initialized to use its default block size. If a value of 0 is specified, the block size will be variable. If the device does not support variable-length block mode, an error is returned.

### INVOKE STORED PROCEDURE

See the CALL command earlier in this chapter.

### LIST ACTIVE DATABASES

This command displays a subset of the information listed by the GET SNAPSHOT FOR ALL DATABASES command. (See the GET SNAPSHOT command earlier in this chapter). An active database is available for connection and use by any application. For each active database, this command displays the following:
**DB2 Command Line Processor Commands**

- Database name
- Number of applications currently connected to the database
- Database path

**Authorization**
SYSADM, SYSCTRL, or SYSMAINT.

**Syntax**

```plaintext
>>-LIST ACTIVE DATABASES------------------------------------------><
  ++AT NODE--nodenum++
  '-GLOBAL-------------'
```

**Description**

The following are the parameter descriptions for LIST ACTIVE DATABASES:

- **AT NODE nodenum** Identifies the node in an EEE database for which the status of databases is displayed.
- **GLOBAL** Returns an aggregate result for all nodes in an EEE database.

**LIST APPLICATIONS**

LIST APPLICATIONS displays the application program name, authorization ID (user name), application handle, application ID, and database name of all active database applications. This command can also optionally display an application’s sequence number, status, status change time, and database path. On EEE systems, this command returns information only for the node on which it is issued.

You can use this command to help in problem determination. LIST APPLICATIONS shows you the application handle, which you need to know to use GET SNAPSHOT or FORCE APPLICATION.

**Authorization**
SYSADM, SYSCTRL, or SYSMAINT.

**Syntax**

```plaintext
>>-LIST APPLICATIONS------------------------------------------><
  '-FOR---DATABASE----database-alias--'
  '-DB-----'
```
Description

The following are the parameter descriptions for LIST APPLICATIONS:

- **FOR DATABASE database-alias** Displays information for each application connected to the specified database. Database name information is not displayed. If this option is not specified, the command displays the information for each application that is currently connected to any database at the node to which the user is currently attached. The default application information is comprised of the following:
  - Authorization ID
  - Application program name
  - Application handle
  - Application ID
  - Database name

- **AT NODE nodenum** Specifies the node for which the status of the monitor switches is displayed.

- **GLOBAL** Returns an aggregate result for all nodes in an EEE database system.

- **SHOW DETAIL** Specifies that the output includes the following additional information:
  - Sequence #
  - Application status
  - Status change time
  - Database path

If this option is specified, direct the output to a file and view the report with an editor.

LIST COMMAND OPTIONS

This command lists the current settings for these environment variables, which control the CLP:

- **DB2BQTIME** Specifies the backend process wait time. The backend is the way the CLP maintains a connection to the database.
DB2 Command Line Processor Commands

- **DB2DQTRY** Specifies the number of retries to connect to the backend process.
- **DB2RQTIME** Specifies the request queue wait time in seconds.
- **DB2IQTIME** Specifies the input queue wait time in seconds.
- **DB2OPTIONS** Specifies settings for the following options:
  - `-a` Displays SQLCA
  - `-c` Shows the setting of autocommit
  - `-e` Displays SQLCODE/SQLSTATE
  - `-f` Reads from input file
  - `-l` Logs commands in history file
  - `-n` Removes newline character
  - `-o` Displays output
  - `-p` Displays interactive input prompt
  - `-r` Saves output to report file
  - `-s` Stops execution command error
  - `-t` Sets statement termination character
  - `-v` Echoes current command
  - `-w` Displays FETCH/SELECT warning messages
  - `-z` Saves all output to output file

Authorization
None.

Syntax

```
>>>LIST COMMAND OPTIONS----------------------------------------><
```

LIST DATABASE DIRECTORY
This command lists the contents of the system database directory. If a path is specified, the contents of the local database directory are listed.

Authorization
None.
Syntax

```
>>>-LIST----+-DATABASE-+--DIRECTORY----+----------------+-------><
 '-DB-------'               '-ON--+-path--+--'
     '-drive-
```

Description

The ON path/drive parameter specifies the local database directory from which to list information. If it is not specified, the contents of the system database directory are listed, and this information is the same at all nodes.

If the ON path parameter is specified, the local database directory on that path is returned. This information is not the same at all nodes.

The fields returned include the following:

- **Database alias**
  Identifies the alias used when the database was created or cataloged. If an alias was not entered when the database was cataloged, the database manager uses the database name.

- **Database global name**
  Returns the fully qualified name that uniquely identifies the database in the DCE name space. This is applicable only to DCE.

- **Database name**
  Specifies the name with which the database was created.

- **Local database directory**
  Specifies the path on which the database resides.

- **Database directory/Database drive**
  Specifies the name of the directory or drive where the database resides.

- **Node name**
  Specifies the name of a remote node. This name corresponds to the value entered for the nodename parameter when the database and node were cataloged.

- **Database release level**
  Specifies the release level of the database manager that can operate on the database.

- **Comment**
  Specifies comments entered when the database was cataloged.

- **Directory entry type**
  Specifies the location of the database:

  - Remote describes a database that resides on another node.
  - Indirect describes a database that is local.
  - A home entry indicates that the database directory is on the same path as the local database directory.
  - An LDAP entry indicates that the database location information is stored in an LDAP directory.

All entries in the system database directory are either remote or indirect. All entries in local database directories are identified in the system database directory as indirect entries.
Authentication  Specifies the authentication type cataloged at the client. This field is used to determine whether the client is authenticated with operating system security, Kerberos, or DCE.

Principal name  Specifies the fully qualified Kerberos or DCE principal name.

Catalog node number  Specifies the catalog node (EEE systems only). This is the node on which the CREATE DATABASE command was issued.

Node number  Specifies the number that is assigned in db2nodes.cfg to the node where the command was issued (EEE systems only). In non-EEE systems where there is no db2nodes.cfg file, the node number will always be zero.

### LIST DATALINKS MANAGERS

LIST DATALINKS MANAGERS lists the DB2 Data Links Managers that are registered to a specified database.

**Authorization**

None.

**Syntax**

```
>>> LIST DATALINKS MANAGERS FOR----+-DATABASE-+--dbname---------><
   '-DB------'
```

### LIST DCS APPLICATIONS

This command displays information about applications that are connected to databases through DB2 Connect Enterprise Edition using DRDA.

**Authorization**

SYSADM, SYSCtrl, or SYSMAINT.

**Syntax**

```
>>> LIST DCS APPLICATIONS--------------------------><
       +-SHOW DETAIL+-
       '-EXTENDED----'
```

**Description**

The default application information displayed includes the following.
DB2: The Complete Reference

- Host authorization ID (user name)
- Application program name
- Application handle
- Outbound application ID (luwid)

SHOW DETAIL output includes the following additional information:

- Client application ID
- Client sequence number
- Client database alias
- Client node name (nname)
- Client release level
- Client code page
- Outbound sequence number
- Host database name
- Host release level

EXTENDED output generates an extended report. This report includes all of the fields that are listed when the SHOW DETAIL option is specified, plus the following additional fields:

- DCS application status
- Status change time
- Client platform
- Client protocol
- Client code page
- Process ID of the client application
- Host coded character set ID (CCSID)

The application status field contains one of the following values:

- **connect pending—outbound** The request to connect to a host database has been issued, and DB2 Connect is waiting for the connection to be established.
- **waiting for request** The connection to the host database has been established, and DB2 Connect is waiting for an SQL statement from the client application.
- **waiting for reply** The SQL statement has been sent to the host database.
LIST DCS DIRECTORY
This command lists the contents of the DCS directory. This lists DRDA databases (which are the same as DCS databases).

Authorization
None.

Syntax

```sql
>>-LIST DCS DIRECTORY------------------------------------------><
```

Description
The fields returned are as follows:

- **Local database name** Specifies the alias of the target host database. This corresponds to the database-name parameter entered when the host database was cataloged in the DCS directory.
- **Target database name** Specifies the name of the host database that can be accessed. This corresponds to the target-database-name parameter entered when the host database was cataloged in the DCS directory.
- **Application requester name** Specifies the name of the program residing on the application requester or server.
- **DCS parameters** Contains the connection and operating environment parameters to use with the application requester. This corresponds to the parameter string entered when the host database was cataloged. The string must be enclosed by double quotation marks and the parameters must be separated by commas.
- **Comment** Describes the database entry.

The DCS directory is created the first time CATALOG DCS DATABASE is invoked. It is maintained on the path/drive where DB2 is installed, and provides information about host databases that the workstation can access if the DB2 Connect program has been installed. The host databases can be DRDA servers like DB2 for OS/390.

LIST DRDA INDOUBT TRANSACTIONS
This command provides a list of transactions that are indoubt between DRDA requesters and DRDA servers. If DRDA commit protocols are being used, the command lists indoubt transactions between DRDA syncpoint managers.
Authorization

SYSADM.

Syntax

```
>>-LIST DRDA INDOUBT TRANSACTIONS--+-+-----------+--------+
   '-WITH PROMPTING-'  
```

Description

When the WITH PROMPTING parameter is specified, an interactive dialog mode is initiated, permitting you to commit or roll back indoubt transactions. If this parameter is not specified, indoubt transactions are displayed, and the interactive dialog mode is not initiated.

A forget option is not supported. Once the indoubt transaction is committed or rolled back, the transaction can no longer be displayed with LIST DRDA INDOUBT TRANSACTIONS.

The interactive dialog mode of this command permits you to use the following options:

- **L** Lists all indoubt transactions
- **L x** Lists indoubt transaction number x
- **q** Quits
- **c x** Commits transaction number x
- **r x** Rolls back transaction number x

A blank space must separate the command letter from its argument. Before a transaction is committed or rolled back, the transaction data is displayed, and the user is asked to confirm the action.

DRDA indoubt transactions occur when communication is lost between coordinators and participants in distributed units of work, which let a user or application read and update data at multiple locations within a single unit of work. Such work requires a two-phase commit. The first phase requests all the participants to prepare for a commit. The second phase commits or rolls back the transactions. If a coordinator or participant becomes unavailable after the first phase, the distributed transactions are indoubt.

Before issuing the LIST DRDA INDOUBT TRANSACTIONS command, the application process must be connected to the DB2 Syncpoint Manager (SPM) instance. Use the `spm_name` database manager configuration parameter as the dbalias on the CONNECT statement. TCP/IP connections, using the SPM to coordinate commits, use DRDA two-phase commit protocols. APPC connections use LU6.2 two-phase commit protocols.
LIST HISTORY

LIST HISTORY lists entries in the history file, which contains a record of recovery and administrative events. Recovery events include full database- and table space–level backup, incremental backup, restore, and roll-forward operations. Additional logged events include create, alter, and rename table space, run statistics, reorganize table, drop table, and load.

The report generated by LIST HISTORY contains the following operation symbols:

- A Creates table space
- B Backs up
- C Loads copy
- D Drops table
- F Rolls forward
- G Reorganizes table
- L Loads
- N Renames table space
- Q Quiesces
- R Restores
- S Runs statistics
- T Alters table space
- U Unloads

The report also contains a variety of types:

- Backup types:
  - F Offline
  - N Online
  - I Incremental offline
  - O Incremental online
  - D Delta offline
  - E Delta online

- Roll-forward types:
  - E End of logs
  - P Point in time
DB2: The Complete Reference

Load types:
- I Insert
- R Replace

Alter table space types:
- C Add containers
- R Rebalance

Quiesce types:
- S Quiesce share
- U Quiesce update
- X Quiesce exclusive
- Z Quiesce reset

If a roll forward has been carried out to the end of all the logs, the backup ID represents the end of time; that is, the backup ID value is 99991231235959. The backup ID is displayed with LIST HISTORY.

Authorization
None.

Syntax

>>>-LIST HISTORY------------------------+------------------------->
   +---BACKUP----------------------+
   +---ROLLFORWARD------+
   +---DROPPED TABLE------+
   +---LOAD----------------+
   +---CREATE TABLESPACE--
   +---ALTER TABLESPACE---
   '---RENAME TABLESPACE-'

>-----+---ALL--------------------------------+-------------------->
   +---SINCE--timestamp-------------------+
   '---CONTAINING---+schema.object_name+-'
   '---object_name-------'

>-----+FOR-----------------------+database-alias------------------------><
   +---DATABASE++
   '---DB------'
DB2 Command Line Processor Commands

Description

The following parameters can be used with the LIST HISTORY command:

- **HISTORY** Lists all events that are currently logged in the history file.
- **BACKUP** Lists backup and restore operations.
- **ROLLFORWARD** Lists roll-forward operations.
- **DROPPED TABLE** Lists all drop table commands.
- **LOAD** Lists loads.
- **CREATE TABLESPACE** Lists CREATE TABLESPACE and DROP TABLESPACE operations.
- **RENAME TABLESPACE** Lists RENAME TABLESPACE operations.
- **ALTER TABLESPACE** Lists ALTER TABLESPACE operations.
- **ALL** Lists all entries of the specified type in the history file.
- **SINCE timestamp** Gives a complete timestamp (format yyyymmddhhmnnss), or an initial prefix (minimum yyyy) can be specified. This parameter lists all entries with timestamps equal to or greater than the timestamp provided.
- **CONTAINING schema.object_name** Specifies a qualified name uniquely identifying a table.
- **CONTAINING object_name** Specifies an unqualified name uniquely identifying a table space.
- **FOR DATABASE database-alias** Identifies the database whose recovery history file is to be listed.

Example

List backup history for the sample database:

```
db2 list history since 19980201 for sample
```

List backup history for the table space userspace 1 for the sample database:

```
db2 list history backup containing userspace1 for sample
```

List all dropped tables for the sample database:

```
db2 list history dropped table all for db sample
```

The following example shows a LIST HISTORY command with its output.
db2 list history backup all for sample

List History File for sample

Number of matching file entries = 1

<table>
<thead>
<tr>
<th>Op</th>
<th>Obj</th>
<th>Timestamp+Sequence</th>
<th>Type</th>
<th>Dev</th>
<th>Earliest Log</th>
<th>Current Log</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>D</td>
<td>20010701205137001</td>
<td>F</td>
<td>D</td>
<td>S0000000.LOG</td>
<td>S0000000.LOG</td>
</tr>
</tbody>
</table>

Contains 2 tablespace(s):

00001 SYSCATSPACE
00002 USERSPACE1

Comment: DB2 BACKUP SAMPLE OFFLINE

Start Time: 20010701205137
End Time: 20010701205233

Location: c:\SAMPLE.0\DB2\NODE0000\CATN0000\20010701

### LIST INDOUBT TRANSACTIONS

This command returns a list of indoubt transactions on the executed node. The user can interactively commit, roll back, or forget the indoubt transactions.

The two-phase commit protocol is comprised of the following:

- The PREPARE phase, in which the resource manager writes the log pages to disk so that it can respond to either a COMMIT or a ROLLBACK primitive
- The COMMIT (or ROLLBACK) phase, in which the transaction is actually committed or rolled back

#### Authorization

DBADM.

#### Syntax

```bash
> > LIST INDOUBT TRANSACTIONS--+-+----------+-<
'--WITH PROMPTING--'
```
Description

An indoubt transaction is a global transaction that is neither committed nor rolled back. This occurs when either the Transaction Manager (TM) or at least one Resource Manager (RM) becomes unavailable after successfully completing the first phase (that is, the PREPARE phase) of the two-phase commit protocol. The RMs do not know whether to commit or to roll back their branch of the transaction until the TM can consolidate its own log with the indoubt status information. This happens when the RMs again become available.

The WITH PROMPTING parameter specifies that indoubt transactions are to be processed. If this parameter is specified, an interactive dialog mode is initiated, permitting you to commit, roll back, or forget indoubt transactions. If this parameter is not specified, indoubt transactions are displayed and the interactive dialog mode is not initiated.

The LIST INDOUBT TRANSACTIONS command shows the role of the database in each indoubt transaction:

- TM The indoubt transaction is using the database as a Transaction Manager database.
- RM The indoubt transaction is using the database as a Resource Manager, meaning that it is one of the databases participating in the transaction, but it is not the Transaction Manager database.

If LIST INDOUBT TRANSACTIONS is issued against the currently connected database, the command returns the information on indoubt transactions in that database. Only transactions whose status is indoubt (i), or missing commit acknowledgement (m) can be committed. Only transactions whose status is indoubt (i) or ended (e) can be rolled back. Only transactions whose status is committed (c) or rolled back (r) can be forgotten.

Indoubt transaction information is valid only at the time that the command is issued. Once in interactive dialog mode, transaction status may change because of external activities. If this happens and you attempt to process an indoubt transaction that is no longer in an appropriate state, an error message is displayed. After this type of error occurs, quit (q) the interactive dialog mode and reissue the LIST INDOUBT TRANSACTIONS WITH PROMPTING command to refresh the information shown.

The interactive dialog mode permits the user to do the following:

- L Lists all indoubt transactions
- L x Lists indoubt transaction number x
- Q Quits
- c x Commits transaction number x
- r x Rolls back transaction number x
- f x Forgets transaction number x
A blank space must separate the command letter from its argument. Before a transaction is committed, rolled back, or forgotten, the transaction data is displayed, and the user is asked to confirm the action.

**Example**

The following is sample dialog generated by LIST INDOUBT TRANSACTIONS:

In-doubt Transactions for Database SAMPLE

1. originator: XA
   appl_id: *LOCAL.DB2.95051815165159 sequence_no: 0001
   status: i
   xid: 53514C2000000017 00000000544D4442 00000000002F93DD
   A92F8C4FF3000000 0000BD

2. originator: XA
   appl_id: *LOCAL.DATABASE.950407161043 sequence_no: 0002
   status: i
   xid: 53514C2000000017 00000000544D4442 00000000002F95FE
   B62F8C4FF3000000 0000C1

Enter in-doubt transaction command or 'q' to quit.
e.g. 'c 1' heuristically commits transaction 1.
c/r/f/l/q: c 1

1. originator: XA
   appl_id: *LOCAL.DB2.95051815165159 sequence_no: 0001
   status: i
   xid: 53514C2000000017 00000000544D4442 00000000002F93DD
   A92F8C4FF3000000 0000BD
Do you want to heuristically commit this in-doubt transaction?
(y/n) y

DB20000I "COMMIT INDOUBT TRANSACTION" completed successfully

c/r/f/l/q: c 5

DB20030E "5" is not a valid in-doubt transaction number.

c/r/f/l/q: l

In-doubt Transactions for Database SAMPLE

1. originator: XA
   appl_id: *LOCAL.DB2.95051815165159 sequence_no: 0001
   status: c
   xid: 53514C2000000017 00000000544D4442 00000000002F93DD A92F8C4FF3000000
   0000BD

2. originator: XA
   appl_id: *LOCAL.DATABASE.950407161043 sequence_no: 0002
   status: i
   xid: 53514C2000000017 00000000544D4442 00000000002F95FE B62F8C4FF3000000
   0000C1
   .
   .
   c/r/f/l/q: r 2

2. originator: XA
   appl_id: *LOCAL.DATABASE.950407161043 sequence_no: 0002
   status: i
   xid: 53514C2000000017 00000000544D4442 00000000002F95FE B62F8C4FF3000000
   0000C1
Do you want to heuristically rollback this in-doubt transaction? (y/n) y

DB20000I "ROLLBACK INDOUBT TRANSACTION" completed successfully

c/r/f/l/q: l 2

2. originator: XA
   appl_id: *LOCAL.DATABASE.950407161043 sequence_no: 0002
   status: r
   xid: 53514C20000000017 00000000544D4442 00000000002F95FE
   B62F8C4FF3000000
   0000C1

c/r/f/l/q: f 2

2. originator: XA
   appl_id: *LOCAL.DATABASE.950407161043 sequence_no: 0002
   status: r
   xid: 53514C20000000017 00000000544D4442 00000000002F95FE
   B62F8C4FF3000000
   0000C1

Do you want to forget this in-doubt transaction? (y/n) y

DB20000I "FORGET INDOUBT TRANSACTION" completed successfully

c/r/f/l/q: l 2

2. originator: XA
   appl_id: *LOCAL.DATABASE.950407161043 sequence_no: 0002
   status: f
   xid: 53514C20000000017 00000000544D4442 00000000002F95FE
   B62F8C4FF3000000
   0000C1

c/r/f/l/q: q
LIST NODE DIRECTORY

This command lists the contents of the node directory.

You'll need to create a node directory on each database client. It contains an entry for each remote workstation having databases that the client can access. The DB2 client uses the communication endpoint information in the node directory whenever a database connection or instance attachment is requested.

The database manager creates a node entry and adds it to the node directory each time it processes a CATALOG...NODE command. The entries vary, depending on the communications protocol used by the node.

The node directory can contain entries for the following types of nodes:

- APPC
- APPCLU
- APPN
- IPX/SPX
- Local
- Named pipe
- NetBIOS
- TCP/IP

Authorization

None.

Syntax

```
>>-LIST--+-NODE DIRECTORY+-+<
   '-ADMIN-
   '-SHOW DETAIL-
```

Description

The following are the parameter descriptions for LIST NODE DIRECTORY:

- **ADMIN**  Specifies administration server nodes
- **SHOW DETAIL**  Includes the following information:
  - Remote instance name
  - System
  - Operating system type

The common fields shown by this command are as follows.
**Node name**  Gives the name of the remote node. This corresponds to the name entered for the nodename parameter when the node was cataloged.

**Comment**  Displays comments entered when the node was cataloged. To change a comment in the node directory, uncatalog the node, and then catalog it again with the new comment.

**Protocol**  Displays the communications protocol cataloged for the node.

### LIST NODEGROUPS

This command lists all nodegroups associated with the current database.

**Authorization**

For the system catalogs SYSCAT.NODEGROUPS and SYSCAT.NODEGROUPDEF, one of the following is required:

- SYSADM or DBADM
- CONTROL privilege
- SELECT privilege

**Syntax**

```
>>>>LIST NODEGROUPS-----------------<
               'SHOW DETAIL--'
```

**Description**

With the SHOW DETAIL parameter, the output includes the following information:

- Partitioning map ID
- Node number
- In-use flag

The fields displayed have the following meanings:

- **NODEGROUP NAME**  Specifies the name of the nodegroup. The name is repeated for each node in the nodegroup.
- **PMAP_ID**  Specifies the ID of the partitioning map. The ID is repeated for each node in the nodegroup.
- **NODE NUMBER**  Specifies the number of the node.
**IN_USE** Specifies one of four values:

- **Y** The node is being used by the nodegroup.
- **D** The node is going to be dropped from the nodegroup as a result of a REDISTRIBUTE NODEGROUP operation. When the operation completes, the node will not be included in reports from LIST NODEGROUPS.
- **A** The node has been added to the nodegroup but is not yet added to the partitioning map. The containers for the table spaces in the nodegroup have been added on this node. The value is changed to **Y** when the REDISTRIBUTE NODEGROUP operation finishes.
- **T** The node has been added to the nodegroup, but is not yet added to the partitioning map. The containers for the table spaces in the nodegroup have not been added on this node. Table space containers must be added on the new node for each table space in the node group. The value is changed to **A** when containers have been successfully added.

**LIST NODES**

LIST NODES lists all nodes associated with the current database. This command can be issued from any node that is listed in $HOME/sqlib/db2nodes.cfg. It returns the same information from any of these nodes.

**Authorization**

None.

**Syntax**

```sql
>>>LIST NODES
```
Authorization
None.

Syntax
```
.-USER--.
>>-LIST------------------ODBC DATA SOURCES----------------------->
'SYSTEM-
```

Description
The following are the parameter descriptions for LIST ODBC DATA SOURCES:

- **USER** Lists only user ODBC data sources. This is the default if no keyword is specified.
- **SYSTEM** Lists only system ODBC data sources.

LIST PACKAGES/TABLES
Lists packages or tables associated with the current database.

Authorization
For the system catalog SYSCAT.PACKAGES (LIST PACKAGES) and SYSCAT.TABLES (LIST TABLES), one of the following is required:

- SYSADM or DBADM
- CONTROL privilege
- SELECT privilege

Syntax
```
>>-LIST--++PACKAGES+------------------------------------+
  '-TABLES--'
   '-FOR---ALL---------------------'
  '+SCHEMA--schema-name++
  '-SYSTEM----------'
```
Description

The following are the parameter descriptions for the LIST command:

- **FOR** Specifies which packages or tables are to be listed. If this is not specified, the packages or tables for USER are listed.
- **ALL** Lists all packages or tables in the database.
- **SCHEMA** Lists all packages or tables in the database only for the specified schema.
- **SYSTEM** Lists all system packages or tables in the database.
- **USER** Lists all user packages or tables in the database for the current user.
- **SHOW DETAIL** Displays the full table name (valid only for the LIST TABLES command). If this option is not specified, the name is truncated to 30 characters, and the left angle (>) symbol in the thirty-first column represents the truncated portion of the table name.

Example

LIST PACKAGES and LIST TABLES commands provide a quick interface to the system tables. The following SELECT statements return information found in the system tables. They can be expanded to select the additional information that the system tables provide.

```sql
select tabname, tabschema, type, create_time
from syscat.tables
order by tabschema, tabname;

select pkgname, pkgschema, boundby,
total_sect, valid, format, isolation, blocking
from syscat.packages
order by pkgschema, pkgname;

select tabname, tabschema, type, create_time
from syscat.tables
where tabschema = 'SYSCAT'
order by tabschema, tabname;
```
SELECT pkgname, pkgschema, boundby, total_sect, valid, format, isolation, blocking
FROM syscat.packages
WHERE pkgschema = 'nullID'
ORDER BY pkgschema, pkgname;

SELECT tabname, tabschema, type, create_time
FROM syscat.tables
WHERE tabschema = USER
ORDER BY tabschema, tabname;

SELECT pkgname, pkgschema, boundby, total_sect, valid, format, isolation, blocking
FROM syscat.packages
WHERE pkgschema = USER
ORDER BY pkgschema, pkgname;

LIST TABLESPACE CONTAINERS
This command lists containers for the specified table space. It returns information only for the node on which it is executed.

Authorization
SYSADM, SYSCTRL, SYSMAINT, or DBADM.

Syntax
>>LIST TABLESPACE CONTAINERS FOR--tablespace-id--+-SHOW DETAIL-'

Description
The following are the parameter descriptions for LIST TABLESPACE CONTAINERS:

- **FOR tablespace-id** Specifies an integer uniquely identifying the table space used by the current database. To get a list of all the table spaces used by the current database, use LIST TABLESPACES.

- **SHOW DETAIL** Specifies that detailed information be provided. If this is not specified, only the following basic information about each container is provided:
LIST TABLESPACES

LIST TABLESPACES lists table spaces for the current database.

Authorization

SYSADM, SYSCTRL, SYMAINT, DBADM, or LOAD.

Syntax

```bash
>>-LIST TABLESPACES-+------------------------------------+
    "-SHOW DETAIL-"
```

Description

If the SHOW DETAIL parameter is not specified, only the following basic information about each table space is provided:

- Table space ID
- Name
- Type (system-managed space or database-managed space)
- Contents (any data, only long data, or temporary data)
- State, a hexadecimal value indicating the current table space state. The externally visible state of a table space is composed of the hexadecimal sum of certain state values. For example, if the state is quiesced: EXCLUSIVE and load pending, the value is 0x0004 + 0x0008, which is 0x000c. See db2tbst in Chapter 29 to obtain the table space state associated with a given hexadecimal value.

If SHOW DETAIL is specified, the following additional information about each table space is provided:

- Total number of pages
- Number of usable pages
- Accessible (yes or no)
Example

In a multinode environment, the LIST TABLESPACES command does not return all the table spaces in the database. To obtain a list of all the table spaces, run the following:

```
SELECT * FROM STSCAT.TABLESPACES
```

During a table space rebalance, the number of usable pages will include pages for the newly added container, but these new pages will not be reflected in the number of free pages until the rebalance is complete. Without a table space rebalance, the number of used pages plus the number of free pages will equal the number of useable pages.

LOAD

This command loads data into a DB2 table by writing raw data directly to disk. Data residing on the server may be in the form of a file, tape, or named pipe. Data residing on a remotely connected client may be in the form of a fully qualified file or named pipe.

In EEE, this command affects only the partition to which a direct connection exists; LOAD operates only on a single database partition. Use the AutoLoader (db2atld) to run LOAD against multiple database partitions in parallel.
Authorization

You need to have one of these authorities to run LOAD:

- SYSADM or DBADM
- LOAD authority on the database and INSERT privilege on the table when LOAD is invoked in INSERT mode, TERMINATE mode (to terminate a previous load insert operation), or RESTART mode (to restart a previous load insert operation).
- INSERT and DELETE privilege on the table when load is invoked in REPLACE mode, TERMINATE mode (to terminate a previous load replace operation), or RESTART mode (to restart a previous load replace operation).
- INSERT privilege on the exception table, if such a table is used as part of the load. Because all LOAD processes (and all DB2 server processes, in general) are owned by the instance owner, and all of these processes use the identification of the instance owner to access needed files, the instance owner must have read access to input data files. These input data files must be readable by the instance owner, regardless of who invokes the command.

Syntax

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD ------------</td>
<td>FROM ------------</td>
<td>filename</td>
</tr>
<tr>
<td>CLIENT:        +</td>
<td>pipename:       +</td>
<td></td>
</tr>
<tr>
<td>device:        +</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD:           +</td>
<td>LOBS FROM------</td>
<td>lob-path--------</td>
</tr>
<tr>
<td>file-type:      +</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD:           +</td>
<td>MODIFIED BY-----</td>
</tr>
<tr>
<td>file-type:      +</td>
<td>+</td>
</tr>
</tbody>
</table>
```


```
$\text{METHOD}\rightarrow\text{L-}\{\text{-column-start-}\mid\text{column-end}\}\rightarrow\text{column-name}\rightarrow\text{column-position}\rightarrow\text{null INDICATORS}\rightarrow\text{-null-indicator-list-}\rightarrow\text{-null-}\rightarrow\text{-RESTART-}\rightarrow\text{-TERMINATE-}\rightarrow\text{-INTO-}\rightarrow\text{-DATALINK SPECIFICATION-}\rightarrow\text{datalink-spec}
```

Description

The following are the parameter descriptions for LOAD.
CLIENT Specifies that the data to be loaded resides on a remotely connected client. This option is ignored if the load is not invoked from a remote client. Loading data that resides on a remotely connected client is not supported under the following conditions:
- The database the client is connected to is DB2 Enterprise—Extended Edition.
- The database the client is connected to is cataloged against an already cataloged database.

The DUMPFILE and LOBSINFILE modifiers refer to files on the server even when the CLIENT keyword is specified. Code page conversion is not performed during a remote load. If the code page of the data is different from that of the server, the data code page should be specified using the CODEPAGE modifier.

When loading remote client data from files (but not named pipes), there is an upper limit of 2GB per file.

COPY NO Specifies that the table space in which the table resides will be placed in backup pending state if forward recovery is enabled (either logretain or userexit is on). The data in any table in the table space cannot be updated or deleted until a table space backup or a full database backup is made. However, it is possible to access the data in any table by using the SELECT statement.

COPY YES Specifies that a copy of the loaded data will be saved. This option is invalid if forward recovery is disabled (both logretain and userexit are off). The option is not supported for tables with DATALINK columns. COPY YES has its own set of parameters for specifying how to save the loaded data in another location:
- USE TSM Specifies that the copy will be stored using Tivoli Storage Manager (TSM).
- OPEN num-sess SESSIONS Specifies the number of I/O sessions used with TSM or the vendor product. The default is 1.
- TO device/directory Specifies the device or directory on which the copy image will be created.
- LOAD lib-name Specifies the name of the shared library (DLL on OS/2 or Windows) containing the vendor backup and restore I/O functions used. It may contain the full path. If the full path is not given, it will default to the path where the user exit programs reside.
- CPU_PARALLELISM n Specifies the number of processes or threads that LOAD will spawn for parsing, converting, and formatting records when building table objects. This parameter is designed to exploit intrapartition parallelism. It is particularly useful when loading presorted data, because record order in the source data is preserved. If the value of this parameter is 0
or has not been specified, LOAD uses an intelligent default (usually based on the number of CPUs available) at run time.

There is no LOAD parallelism with long fields. If this parameter is used with tables containing either LOB or LONG VARCHAR fields, its value becomes 1, regardless of the number of system CPUs or the value specified by the user.

Specifying a small value for the SAVECOUNT parameter causes LOAD to perform many more I/O operations to flush both data and table metadata. When CPU_PARALLELISM is greater than 1, the flushing operations are asynchronous, permitting load to exploit all processors (CPUs). When CPU_PARALLELISM is set to 1, load waits on I/O during consistency points. A load with CPU_PARALLELISM set to 2 and SAVECOUNT set to 10,000, completes faster than the same operation with CPU_PARALLELISM set to 1, even though there is only one CPU. This is known as overparallelization, and is effective for any workload that is I/O bound, and not CPU-bound.

- **DATA BUFFER buffer-size** Specifies the number of 4KB pages (regardless of the degree of parallelism) to use as buffered space for transferring data within RESTORE. If the value specified is less than the algorithmic minimum, minimum resource is used, and no warning is returned. This memory is allocated directly from the utility heap, whose size can be modified through the util_heap_sz database configuration parameter. If a value is not specified, an intelligent default is calculated by RESTORE at run time. The default is based on a percentage of the free space available in the utility heap at the instantiation time of load, as well as some characteristics of the table.

- **DATALINK SPECIFICATION** Specifies how DATALINK columns are handled. For each DATALINK column, there can be one column specification enclosed by parentheses. Each column specification consists of one or more of DL_LINKTYPE, prefix, and a DL_URL_SUFFIX specification. The prefix specification can be either DL_URL_REPLACE_PREFIX or DL_URL_DEFAULT_PREFIX. There can be as many DATALINK column specifications as the number of DATALINK columns defined in the table. The order of specifications follows the order of DATALINK columns found within the insert-column list, or within the table definition (if an insert-column list is not specified).

- **DISK_PARALLELISM n** Specifies the number of processes or threads spawned by load for writing data to the table space. If not specified, RESTORE selects a default based on the number of table space containers and the characteristics of the table.

- **DL_LINKTYPE** Specifies the type of DATALINK column. This must match the LINKTYPE of the column definition, so DL_LINKTYPE URL is acceptable if the column definition specifies LINKTYPE URL.
DL_URL_DEFAULT_PREFIX “prefix”  Acts as a default prefix for all DATALINK values within the same column. In this context, prefix refers to the “scheme host port” part of the URL specification. (For DFS, prefix refers to the “scheme cellname filesystem-junction” part of the URL specification.) Typical prefixes include the following:

- “http://server”
- “file://server”
- “file:”
- “http://server:80”
- “dfs://.../cellname/fs”

If no prefix is found in the column data, and a default prefix is specified with DL_URL_DEFAULT_PREFIX, the default prefix is prefixed to the column value (if not null). For example, if DL_URL_DEFAULT_PREFIX specifies the default prefix “http://toronto”:

- The column input value “/x/y/z” is stored as “http://toronto/x/y/z”.
- The column input value “http://coyote/a/b/c” is stored as “http://coyote/a/b/c”.
- The column input value null is stored as null.

DL_URL_REPLACE_PREFIX “prefix”  Specifies a prefix for all non-null column values. This is optional but can be useful when loading or importing data previously generated by export, if the user wants to globally replace the hostname in the data with another hostname. If a column value has a prefix, this will replace it. If a column value has no prefix, the prefix specified by DL_URL_REPLACE_PREFIX is prefixed to the column value. For DFS, the prefix refers to the “scheme cellname filesystem-junction” part of the URL specification.

DL_URL_SUFFIX “suffix”  Specifies a suffix to be appended to every non-null column value for the column. DL_URL_SUFFIX is appended to the “path” component of the data location part of the DATALINK value.

FOR EXCEPTION table-name  Specifies the exception table into which rows in error will be copied. Any row in violation of a unique index or a primary key index is copied. DATALINK exceptions are also captured in the exception table. If an unqualified table name is specified, the table is qualified with the CURRENT SCHEMA. Information that is written to the exception table is not written to the dump file. In an EEE database, an exception table must be defined for those nodes on which the loading table is defined. The dump file, on the other hand, contains rows that cannot be loaded because they are invalid or have syntax errors.
FROM filename/pipename/device  Specifies the file, pipe, or device containing the data being loaded. This file, pipe, or device must reside on the node where the database resides, unless the CLIENT option is specified. If several names are specified, they will be processed in sequence. If the last item specified is a tape device, the user is prompted for another tape. Valid response options include the following:

- **c** Continue using the device that generated the warning message (for example, when a new tape has been mounted).
- **d** Device terminate. Stop using the device that generated the warning message (for example, when there are no more tapes).
- **t** Terminate all devices.

Use the fully qualified file name for DATALINK columns. If the database resides on the same node as the caller, relative paths may be used. If loading data that resides on a client machine, the data must be in the form of either a fully qualified file or a named pipe. Loading data from multiple IXF files is supported if the files are physically separate, but logically one file. It is not supported if the files are both logically and physically separate.

**HOLD QUIESCE** Specifies that the table stays in quiesced exclusive state after the load. To remove quiesce state for the table spaces, issue this command:

```
db2 quiesce tablespaces for table <tablename> reset
```

This ensures that no phantom quiesces are created. (See the QUIESCE TABLESPACES FOR TABLE command later in this chapter.)

**INDEXING MODE** Indicates whether load is to re-create indexes or extend them incrementally. Values are the following:

- **AUTOSELECT** Load automatically decides between REBUILD or INCREMENTAL mode.
- **REBUILD** All indexes are rebuilt. Load needs sufficient resources to sort all index key parts for both old and appended table data.
- **INCREMENTAL** Indexes are extended with new data. This approach consumes index-free space. It requires only enough sort space to append index keys for the inserted records. This method is supported only in cases where the index object is valid and accessible at the start of a load. (This method cannot be used immediately after a load in which the DEFERRED mode was used). If this mode is specified but not supported due to the state of the index, a warning is returned, and the load continues in REBUILD.
mode. Similarly, if a load restart operation is begun in the load build phase, INCREMENTAL mode is not supported.

Incremental indexing is not supported when all of the following conditions are true:

- The LOAD COPY option is specified (logretain or userexit is enabled).
- The table resides in a DMS table space.
- The index object resides in a table space that is shared by other table objects belonging to the loaded table. Bypass this restriction by placing indexes in a separate table space.

- **DEFERRED** Indicates that load will not create indexes. Indexes will be marked as needing a refresh. The first access to such indexes during a load may force a rebuild, or indexes may be rebuilt when the database is restarted. This approach requires enough sort space for all key parts for the largest index. The total time subsequently taken for index construction is longer than that required in REBUILD mode. Therefore, when performing multiple loads with deferred indexing, have the last load in the sequence perform an index rebuild, rather than allowing indexes to be rebuilt when they are first accessed by SQL requests from applications or users.

Deferred indexing is supported only for tables with nonunique indexes so that duplicate keys inserted during the load phase are not persistent after the load. Deferred indexing is not supported for tables that have DATALINK columns.

- **INSERT** Adds the loaded data to the table without changing the existing data.
- **insert-column** Specifies the table column into which the data is to be inserted. Load cannot parse columns whose names contain one or more spaces. For example, the following command will fail because of the Int 4 column:

```sql
db2 load from delfile1 of del modified by noeofchar noheader
method P (1, 2, 3, 4, 5, 6, 7, 8, 9)
insert into table1 (BLOB1, S2, I3, Int 4, I5, I6, DT7, I8, TM9)
```

Instead, enclose such column names with double quotation marks, as follows:

```sql
...insert into table1 (BLOB1, S2, I3, "Int 4", ...)
```

- **INTO table-name** Specifies the database table into which the data is loaded. This table cannot be a system table or a declared temporary table. An alias or the fully qualified or unqualified table name can be specified. A qualified table name is in the form schematablename. If an unqualified table name is specified, the table will be qualified with the CURRENT SCHEMA.
LOBS FROM lob-path  Specifies the path to the data files containing LOB values to be loaded. The path must end with a slash. If the CLIENT option is specified, the path must be fully qualified. The names of the LOB data files are stored in the main data file (ASC, DEL, or IXF), in the column that is loaded into the LOB column. This option is ignored if lobsinfile is not specified within the filetype-mod string.

MESSAGES message-file  Specifies the destination for warning and error messages that occur during the load. If a message file is not specified, messages are written to standard output. If the complete path to the file is not specified, load uses the current directory and the default drive as the destination. If an existing file is specified, LOAD appends the messages. The message file is usually empty until load finishes. It cannot be used to monitor load. For real-time monitoring of a load, use LOAD QUERY.

METHOD  Specifies how columns are handled during a LOAD.

L  Specifies the start and end column numbers from which to load data. A column number is a byte offset from the beginning of a row of data. It is numbered starting from 1. This method can be used only with ASC files, and it is the only valid option for that file type.

N  Specifies the names of the columns in the data file to be loaded. The case of these column names must match the case of the corresponding names in the system catalogs (uppercase by default). Each table column that is not nullable should have a corresponding entry in the METHOD N list. For example, given data fields F1, F2, F3, F4, F5, and F6, and table columns C1 INT, C2 INT NOT null, C3 INT NOT null, and C4 INT, method N (F2, F1, F4, F3) is a valid request, while method N (F2, F1) is not valid. This method can be used only with IXF files.

P  Specifies the indexes (numbered from 1) of the input data fields to be loaded. Each table column that is not nullable should have a corresponding entry in the METHOD P list. For example, given data fields F1, F2, F3, F4, F5, and F6, and table columns C1 INT, C2 INT NOT null, C3 INT NOT null, and C4 INT, method P (2, 1, 4, 3) is a valid request, while method P (2, 1) is not valid. This method can be used only with IXF or DEL files, and is the only valid option for DEL.

MODIFIED BY filetype-mod  See the “Valid File Type Modifiers (LOAD)” section later in this chapter for a description of additional file modification options.

NONRECOVERABLE  Marks the load transaction as nonrecoverable and it will not be possible to recover it with a subsequent roll-forward action. The roll-forward utility will skip the transaction and will mark the table that was being loaded as “invalid.” Load will ignore any subsequent transactions against that table. After the roll forward ends, such a table can be dropped or restored only from a backup (full or table space) taken after a commit point following the completion of the nonrecoverable load. Table spaces are not put in backup pending state following the load, and a copy of the loaded data does not have
to be made during the load. This option should not be used when DATALINK columns with the FILE LINK CONTROL attribute are present in or being added to the table.

- **null INDICATORS null-indicator-list** Specifies how nulls are handled. It is used only with method L. Nulls are indicated with a comma-separated list of positive integers specifying the column number of each null indicator field. The column number is the byte offset of the null indicator field from the beginning of a row of data. There must be one entry in the null indicator list for each data field defined in the METHOD L parameter: a column number of zero indicates that the corresponding data field always contains data; Y in the null indicator column specifies that the column data is null; and any character other than Y in the null indicator column specifies that the column data is not null, and that column data specified by the METHOD L option will be loaded. The null indicator character can be changed by using the MODIFIED BY option. (See the description of the nullindchar modifier in the “Valid File Type Modifiers (LOAD)” section later in this chapter.)

- **OF filetype** Specifies the format of the data in the input file:
  - ASC (nondelimited ASCII format)
  - DEL (delimited ASCII format)
  - IXF (integrated exchange format, PC version), exported from a DB2 table

- **REPLACE** Deletes existing data from the table and inserts the loaded data. The table definition and index definitions are not changed. If this option is used when moving data between hierarchies, only the data for an entire hierarchy can be replaced, not individual subtables. REPLACE is not supported for tables with DATALINK columns.

This option can also be used to delete all rows in a table with no logging. There are two ways to do this in DB2. The fastest is IMPORT REPLACE with an IXF file that contains no rows. IMPORT is the best choice because it does not lock the table space. LOAD must be used if the target table has referential integrity dependencies or summary tables defined on it. If you plan to use LOAD a lot, put each table that gets LOADed (or uses LOAD to be emptied) in its own table space, or use IMPORT REPLACE.

- **RESTART** Restarts a previously interrupted load. The load automatically continues from the last consistency point in the load, build, or delete phase.

- **RESTARTCOUNT** Reserved.

- **ROWCOUNT n** Specifies the number of physical records in the file to load. Allows a user to load only the first n rows in a file.

- **SAVECOUNT n** Causes load to establish consistency points after every n rows. This value is converted to a page count and rounded up to intervals of
the extent size. Because a message is issued at each consistency point, this option should be selected if the load will be monitored by using LOAD QUERY. If the value of \( n \) is not sufficiently high, the cost of creating the savecount will make the load slower.

- **SORT BUFFER buffer-size**  Reserved.
- **STATISTICS NO**  Specifies that no statistics are collected, and statistics in the catalogs are not altered. This is the default.
- **STATISTICS YES**  Specifies that statistics are collected for the table and any existing indexes. This option is supported only if the load is in REPLACE mode. The following choices are supported for gathering statistics:
  - **WITH DISTRIBUTION**  Distribution statistics are collected.
  - **AND INDEXES ALL**  Both table and index statistics are collected.
  - **FOR INDEXES ALL**  Only index statistics are collected.
  - **DETAILED**  Extended index statistics are collected.
- **TEMPFILES PATH temp-pathname**  Specifies the pathname used when creating temporary files during a load. It must be fully qualified to the server node. Temporary files take up file system space, and sometimes this space requirement is large.
- **TERMINATE**  Terminates an interrupted load and rolls back the operation to where it started, even if consistency points were passed. The states of any table spaces involved in the operation return to normal, and all table objects are made consistent. (Index objects may be marked as invalid, in which case, index rebuilds take place at the next access.) If the load terminated is a load REPLACE, the table is truncated to an empty table after the load TERMINATE operation. If the load terminated is a load INSERT, the table will retain all of its original records after the load TERMINATE operation. The load terminate option will not remove a backup pending state from table spaces.
- **USING directory**  Reserved.
- **WARNINGCOUNT n**  Stops the load after \( n \) warnings. Set this parameter if no warnings are expected, but verification that the correct file and table are used is desired. If \( n \) is zero or this option is not specified, the load will continue regardless of the number of warning messages. If the load is stopped because the threshold of warnings was encountered, another load can be started in RESTART mode. The load automatically continues from the last consistency point. Or another load starts in REPLACE mode, starting at the beginning of the input file.
- **WITHOUT PROMPTING**  Specifies that the list of data files contains all the files that are to be loaded, and that the devices or directories listed are sufficient for the entire load. If a continuation input file is not found or the copy targets
are filled before the load finishes, the load will fail, and the table will remain in load pending state. If WITHOUT PROMPTING is not specified and a tape device encounters an end of tape for the copy image, or the last item listed is a tape device, you are prompted for a new tape on that device.

Valid File Type Modifiers (LOAD)
File type modifiers vary depending on whether the input file is IXF, ASCII, or delimited ASCII (DEL).

Load File Type Modifiers for All File Formats
The following are the valid load file type modifiers for all file formats:

- **anyorder** Allows greater SMP parallelism by not enforcing load to preserve the order of data in the source files. It is used with the cpu_parallelism parameter; if cpu_parallelism = 1, anyorder is ignored. This option is not supported if SAVECOUNT > 0 because crash recovery after a consistency point requires that data be loaded in sequence.

- **fastparse** Speeds up load by reducing syntax checking on user-supplied column values. Tables loaded with fastparse are guaranteed to be architecturally correct, and load performs sufficient data checking to prevent segmentation violations and traps. Data that is in the correct form will be loaded correctly. For example, if 123qwr4 were supplied for an integer in an ASC file, load would ordinarily flag a syntax error because qwr cannot be in an integer. With fastparse, a syntax error is not detected, and an arbitrary number is loaded into the integer column. Use fastparse only with clean data. Performance improvements from fastparse can be large with ASCII files. The binary format of IXF prevents fastparse from speeding up IXF loads.

- **generatedignore** Informs load that data for all generated columns is present in the data file but should be ignored. For nullable generated columns, this results in nulls being loaded into the column; for non-null generated columns, this loads the default for the generated column’s data type. At the end of the load, run SET INTEGRITY to force the replacement of loaded values with values computed according to the generated column definition. generatedignore cannot be used with either generatedmissing or generatedoverride.

- **generatedmissing** Makes load assume that the input data file contains no data for the generated column (not even nulls), and load will therefore load nulls into the column. At the end of the load operation, run SET INTEGRITY to replace the nulls with values computed according to the generated column definition. generatedmissing cannot be used with either generatedignore or generatedoverride.

- **generatedoverride** Instructs load to accept explicit, non-null data for all generated columns in the table (contrary to the normal rules for these types of columns). This is useful when migrating data from another database system, or
when loading a table from data that was recovered using the DROPPED TABLE RECOVERY option on the ROLLFORWARD DATABASE command. When generatedoverride is used, any rows with no data or null data for non-null generated columns will be rejected (SQLCODE SQL3116W). Load will not attempt to validate generated column values when generatedoverride is used. generatedoverride cannot be used with either generatedmissing or generatedignore.

- **identityignore** Tells load to ignore data for the identity column in the data file. This results in all identity values being generated by load, for both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. For GENERATED ALWAYS columns, no rows will be rejected. identityignore cannot be used with identitymissing or identityoverride.

- **identitymissing** Makes load assume that the input data file contains no data for the identity column (not even nulls), and load will therefore generate a value for each row. This happens for both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. identitymissing cannot be used with either identityignore or identityoverride.

- **identityoverride** Causes Load to accept explicit, non-null data for identity columns (contrary to the normal rules for these types of identity columns). Use this only when an identity column defined as GENERATED ALWAYS is in the table to be loaded. This is useful when migrating data from another database system when the table must be defined as GENERATED ALWAYS, or when loading a table from data that was recovered using the DROPPED TABLE RECOVERY option on the ROLLFORWARD DATABASE command. With identityoverride, any rows with no data or null data for the identity column will be rejected (SQLCODE SQL3116W). identityoverride cannot be used with either identitymissing or identityignore. Load will not attempt to maintain or verify the uniqueness of values in the table’s identity column when using identityoverride.

- **inexfreespace=x** Specifies an integer from 0 to 99 that will define the percentage of each index page left as free space when loading the index. The first entry in a page is added without restriction; subsequent entries are added if the percent-free space threshold can be maintained. The default for indexfreespace is that chosen when the index was created. The indexfreespace chosen on a load takes precedence over the PCTFREE value from CREATE INDEX statement, and affects only index leaf pages.

- **lobsinfile lob-path** Specifies the path to the files containing LOB values. This allows each LOB to be in its own file before being LOADed. The ASC, DEL, or IXF load input files contain the names of the files having LOB data in the LOB column.

- **noheader** Skips the header verification code (applies to only EEE, and only to loads into tables that reside in a single-node nodegroup). AutoLoader writes a header to each file contributing data to a table in a multinode nodegroup, and the header includes the node number, partitioning map, and partitioning key specification. Load requires these to verify that data is loaded at the correct
node. When loading files into a table on a single-node nodegroup, the headers do not exist. Use noheader to skip the header verification code.

- **norowwarnings** Suppressed warnings about rejected rows.
- **pagefreespace=x** Specifies an integer from 0 to 100 that defines the percentage of each data page left as free space. If the percentage is invalid because of the minimum row size (for example, a row that is at least 3K long being loaded on a 4K page with pagefreespace=50), the row will be placed on a new page. If a value of 100 is specified, each row is on its own page. The PCTFREE value of a table determines the amount of free space designated per page. If a pagefreespace value on the load or a PCTFREE value on a table have not been set, load will fill up as much space as possible on each page. The value set by pagefreespace overrides the PCTFREE value specified for the table.
- **totalfreespace=x** Specifies an integer from 0 to 100 inclusive that defines the percentage of the total pages in the table that is to be appended to the end of the table as free space. For example, if \( x = 20 \) and the table has 100 data pages, 20 additional empty pages are appended. The total number of data pages for the table will be 120.
- **usedefaults** Specifies that defaults be loaded if a source column for a target table column has been specified, but contains no data for one or more row instances. Examples of missing data are the following:
  - **DEL files** A double comma (,,) is specified for the column.
  - **DEL/ASC/WSF files** A row that does not have enough columns or is not long enough for the original specification.

Without this option, if a source column contains no data for a row instance, one of the following occurs:

- If the column is nullable, a null is loaded.
- If the column is not nullable, load rejects the row.

**Load File Type Modifiers for ASCII File Formats (ASC/DEL)** The following are the valid load file type modifiers for the ASCII file format:

- **codepage=x** Specifies an ASCII character string, \( x \), whose value is interpreted as the code page of the data in the input data set. It converts character data (and numeric data specified in characters) from this code page to the database code page during the load.
- **dateformat=“x”** Specifies that \( x \) is the date format in the source file. Valid date elements are as follows:
  - **YYYY Year** Four digits ranging from 0000–9999
  - **M Month** One or two digits ranging from 1–12
DB2 Command Line Processor Commands

- **MM Month** Two digits ranging from 1–12; mutually exclusive with M
- **D Day** One or two digits ranging from 1–31
- **DD Day** Two digits ranging from 1–31; mutually exclusive with D
- **DDD Day of the year** Three digits ranging from 001–366; mutually exclusive with other day or month elements

A default of 1 is assigned for each element that is not specified. These are some examples of date formats: D-M-YYYY, MM.DD.YYYY, and YYYYDDD.

**dumpfile = x** Specifies x, the fully qualified name (according to the server node) of an exception file for writing rejected rows. A maximum of 32KB of data is written per record. The following example shows how to specify a dump file:

```
db2 load from data of del
modified by dumpfile = /u/user/filename
insert into table_name
```

In an EEE database, the path should be local to the loading node so that concurrently running loads do not attempt to write to the same file. The file contents are written to disk in an asynchronous buffered mode. If load fails or is interrupted, the number of records committed to disk cannot be known with certainty, and consistency cannot be guaranteed after a LOAD RESTART. The file can be assumed to be complete only for a load that starts and completes in a single pass.

*dumpfile* cannot be a filename with multiple file extensions. For example,

```
dumpfile = /home/svtdbm6/DUMP.FILE
```

is valid, but the following is not:

```
dumpfile = /home/svtdbm6/DUMP.LOAD.FILE.
```

- **implieddecimal** Specifies that the location of an implied decimal point be determined by the column definition. Loading 12345 into DECIMAL(8,2) will result in 123.45, not 12345.00.
- **timeformat=x** Specifies that x is the format of the time in the source file. Valid time elements are the following:
  - **H Hour** One or two digits ranging from 0–12 for a 12-hour system, and 0–24 for a 24-hour system
  - **HH Hour** Two digits ranging from 0–12 for a 12-hour system, and 0–24 for a 24-hour system; mutually exclusive with H
  - **M Minute** One or two digits ranging from 0–59
**MM Minute**  Two digits ranging from 0–59; mutually exclusive with M

**S Second**  One or two digits ranging from 0–59

**SS Second**  Two digits ranging from 0–59; mutually exclusive with S

**SSSSS Second of the day after midnight**  Five digits ranging from 00000–86399; mutually exclusive with other time elements

**TT Meridian indicator**  A.M. or P.M.

A default of 0 is assigned for each element that is not specified. Some examples of time formats are the following: HH:MM:SS, HH.MM TT, and SSSSS.

**timestampformat=“x”**  Specifies that x is the format of the timestamp in the source file. Valid timestamp elements are as follows:

- **YYYY Year**  Four digits ranging from 0000–9999
- **M Month**  One or two digits ranging from 1–12
- **MM Month**  Two digits ranging from 1–12; mutually exclusive with M, month
- **D Day**  One or two digits ranging from 1–31
- **DD Day**  Two digits ranging from 1–31; mutually exclusive with D
- **DDD Day of the year**  Three digits ranging from 001–366; mutually exclusive with other day or month elements
- **H Hour**  One or two digits ranging from 0–12 for a 12-hour system, and 0–24 for a 24-hour system
- **HH Hour**  Two digits ranging from 0–12 for a 12-hour system, and 0–24 for a 24-hour system; mutually exclusive with H
- **M Minute**  One or two digits ranging from 0–59
- **MM Minute**  Two digits ranging from 0–59; mutually exclusive with M, minute
- **S Second**  One or two digits ranging from 0–59
- **SS Second**  Two digits ranging from 0–59; mutually exclusive with S
- **SSSSS Second of the day after midnight**  Five digits ranging from 00000–86399; mutually exclusive with other time elements
- **UUU Microsecond**  Six digits ranging from 000000–999999
- **TT Meridian indicator**  A.M. or P.M.

A default of 1 is assigned for unspecified YYYY, MM, D, DD, or DDD elements. A default of 0 is assigned for all other unspecified elements. Following is an example of a timestamp format:

"YYYY/MM/DD HH:MM:SS.UUUUUU"
The following example illustrates how to import data containing user-defined date and time formats into a table called schedule:

```
   db2 import from delfile2 of del
   modified by timestampformat="yyyy.mm.dd hh:mm tt"
   insert into schedule
```

- **noeofchar** Specifies that the optional end-of-file character x’1A’ is not recognized as the end of file. Processing continues as if it were a normal character.

**Load File Type Modifiers for the ASC (Nondelimited ASCII) File Format**

The following are the valid load file type modifiers for the ASC file format:

- **binarynumerics** Specifies that numeric (but not DECIMAL) data must be in binary form, not the character representation. This avoids costly conversions. This option is supported only with positional ASC, using fixed-length records specified by the reclen option. The noeofchar option is assumed. The following rules apply:
  - No conversion between data types is performed, with the exception of BIGINT, INTEGER, and SMALLINT.
  - Data lengths must match their target column definitions.
  - FLOATs must be in IEEE floating-point format.
  - Binary data in the load source file is assumed to be big endian, regardless of the platform on which the load is running.
  - nulls cannot be present in the data for columns affected by this modifier. Blanks (normally interpreted as null) are interpreted as a binary value when this modifier is used.

- **nochecklengths** Specifies that an attempt will be made to load each row, even if the source data has a column definition that exceeds the size of the target table column. Such rows can be successfully loaded if code page conversion causes the source data to shrink; for example, four-byte EUC data in the source could shrink to two-byte DBCS data in the target, and require half the space. This option is particularly useful if you know that the source data will fit in all cases despite mismatched column definitions.

- **nullindchar=x** Changes the character denoting a null value to x, which must be a single character. The default value of x is Y. This modifier is case-sensitive for EBCDIC data files, except when the character is an English letter. For example, if the null indicator character is specified to be the letter N, then n is also recognized as a null indicator.

- **packeddecimal** Loads packed-decimal data directly because the binarynumerics modifier does not include the DECIMAL field type. This option is supported only...
with positional ASC, using fixed-length records specified by the reclen option. The noeofchar option is assumed. Supported values for the sign nibble are the following:

- + 0xC 0xA 0xE 0xF
- - 0xD 0xB

Nulls cannot be present in the data for columns affected by this modifier. Blanks (normally interpreted as null) are interpreted as a binary value when this modifier is used. Regardless of the server platform, the byte order of binary data in the load source file is assumed to be big endian; that is, when using this modifier on Linux, Intel, OS/2, or Windows, the byte order must not be reversed.

- **reclen=x** Specifies an integer with a maximum value of 32,767. x characters are read for each row, and a newline character is not used to indicate the end of the row.
- **striptblanks** Truncates any trailing blank spaces when loading data into a variable-length field. If not specified, blank spaces are kept. striptblanks cannot be specified together with striptnulls.
- **striptnulls** Truncates any trailing nulls (0x00) when loading data into a variable-length field. If not specified, nulls are kept. striptnulls cannot be specified together with striptblanks.
- **zoneddecimal** Loads zoned decimal data because the BINARYNUMERICS modifier does not include the DECIMAL field type. This option is supported only with positional ASC, using fixed-length records specified by the RECLLEN option. The NOEOFCHAR option is assumed. Half-byte sign values can be one of the following:
  - + 0xC 0xA 0xE 0xF
  - - 0xD 0xB

Supported values for digits are 0x0 to 0x9. Supported values for zones are 0x3 and 0xF.

**Load File Type Modifiers for the DEL (Delimited ASCII) File Format**

The following are the valid load file type modifiers for the DEL file format:

- **chardelx** Specifies that x is a single character string delimiter. The default is a double quotation mark ("). The specified character is used in place of double quotation marks to enclose a character string. The single quotation mark (‘) can also be specified as a character string delimiter, as follows:

  modified by chardel''

- **coldelx** Specifies that x is a single character column delimiter. The default is a comma (,). The specified character is used in place of a comma to signal the end of a column.

- **datesiso** Causes all date data values to be loaded in ISO format.
**DB2 Command Line Processor Commands**

- **decplusblank** Causes positive decimal values to be prefixed with a blank space instead of a plus sign (+). The default action is to prefix positive decimal values with a plus sign.

- **decptx** Specifies that x is a single character substitute for the period (.) as a decimal point character.

- **delprioritychar** Specifies a priority for delimiters. The default priority for delimiters is as follows:
  - record delimiter
  - character delimiter
  - column delimiter

  Delprioritychar is for applications using the priority of character delimiter, record delimiter, column delimiter.

```
db2 load ... modified by delprioritychar ...
```

- **dldelx** Specifies that x will be used in place of a semicolon (;) as the interfield separator for a DATALINK value. It is needed because a DATALINK value may have more than one subvalue. x must not be the same character specified as the row, column, or character string delimiter.

- **keepblanks** Preserves leading and trailing blanks in each field of type CHAR, VARCHAR, LONG VARCHAR, or CLOB. Without this option, all leading and trailing blanks not inside character delimiters are removed, and a null is inserted into the table for all blank fields. The following example illustrates how to load data into a table called TABLE1, while preserving all leading and trailing spaces in the data file:

```
db2 load from delfile3 of del
modified by keepblanks
insert into table1
```

- **nodoubledel** Suppresses recognition of double-character delimiters. For more information, see Delimiter Restrictions in EXPORT in this chapter.

**Load File Type Modifiers for the IXF File Format** The following are the valid load file type modifiers for the IXF file format:

- **forcein** Directs load to accept data despite code-page mismatches, and to suppress translation between code pages. Fixed-length target fields are checked to verify that they are large enough for the data. If nochecklengths is specified, no checking is done, and an attempt is made to load each row.
nochecklengths Specifies that an attempt be made to load each row, even if the source data has a column definition that exceeds the size of the target table column. Such rows can be successfully loaded if code-page conversion causes the source data to shrink; for example, four-byte EUC data in the source could shrink to two-byte DBCS data in the target, and require half the space. This option is particularly useful if you know that the source data will fit in all cases despite mismatched column definitions.

Load does not issue a warning if an attempt is made to use unsupported file types with the MODIFIED BY option. If this is attempted, the load fails and an error code is returned.

Double quotation marks around the date format string are mandatory. Field separators cannot contain any of the following: a–z, A–Z, and 0–9. The field separator should not be the same as the character delimiter or field delimiter in the DEL file format. A field separator is optional if the start and end positions of an element are unambiguous. Ambiguity can exist if (depending on the modifier) elements such as D, H, M, or S are used, because of the variable length of the entries.

For timestamp formats, take care to avoid ambiguity between the month and minute descriptors because they both use the letter M. A month field must be adjacent to other date fields. A minute field must be adjacent to other time fields. In ambiguous cases, load will report an error message, and the operation will fail. The following are some unambiguous timestamp formats:

- "M:YYYY" Month
- "S:M" Minute
- "M:YYYY:S:M" Month....Minute
- "M:H:YYYY:M:D" Minute....Month

Example

TABLE1 has five columns, named COL1 through COL5:

- COL1 VARCHAR 20 NOT NULL WITH DEFAULT
- COL2 SMALLINT
- COL3 CHAR 4
- COL4 CHAR 2 NOT NULL WITH DEFAULT
- COL5 CHAR 2 NOT null

ASCFILE1 has seven elements:

- ELE1 positions 01 to 20
- ELE2 positions 21 to 22
- ELE5 positions 23 to 23
DB2 Command Line Processor Commands

- ELE3 positions 24 to 27
- ELE4 positions 28 to 31
- ELE6 positions 32 to 32
- ELE7 positions 33 to 40

The file has the data laid out as follows:

```
1...5....10...15...20...25...30...35...40
Test data 1         XXN 123abcdN
Test data 2 and 3   QQY    wxyzN
Test data 4,5 and 6 WWN6789    Y
```

The following command loads the table from the file:

```
db2 load from ascfile1 of asc modified by striptblanks reclen=40
    method L (1 20, 21 22, 24 27, 28 31)
    null indicators (0,0,23,32)
    insert into table1 (col1, col5, col2, col3)
```

In the preceding example, the parameters function as follows:

- striptblanks in the MODIFIED BY parameter forces the truncation of blanks in VARCHAR columns (COL1, for example, which is 11, 17, and 19 bytes long, in rows 1, 2, and 3, respectively).
- reclen=40 indicates that there is no newline character at the end of each input record, and that each record is 40 bytes long. The last 8 bytes are not used to load the table.
- COL4 is not provided in the input file; it will be inserted into TABLE1 with its default value. (It is defined NOT NULLWITH DEFAULT.)
- COL2 and COL3 use positions 23 and 32 to indicate whether TABLE1 will be loaded null for a given row. If there is a Y in the column’s null indicator position for a given record, the column will be null. If there is an N, the data values in the column’s data positions of the input record (as defined in L(........)) are used as the source of column data for the row. In this example, neither column in row 1 is null; COL2 in row 2 is null; and COL3 in row 3 is null.

COL1 and COL5 have null INDICATORS specified as 0 (zero), indicating that the data is not nullable. The null INDICATOR for a given column can be anywhere in the input record, but the position and Y or N values must be supplied.

Example: Loading LOBs from Files

TABLE1 has three columns.
DB2: The Complete Reference

- COL1 CHAR 4 NOT NULL WITH DEFAULT
- LOB1 LOB
- LOB2 LOB

ASCFILE1 has three elements:
- ELE1 positions 01 to 04
- ELE2 positions 06 to 13
- ELE3 positions 15 to 22

The following files reside in either /u/user1 or /u/user1/bin:
- ASCFILE2 has LOB data
- ASCFILE3 has LOB data
- ASCFILE4 has LOB data
- ASCFILE5 has LOB data
- ASCFILE6 has LOB data
- ASCFILE7 has LOB data

Data records in ASCFILE1 are like this (but there would normally be thousands or more records like this in a typical load):

```
1...5....10...15...20...25...30.
REC1 ASCFILE2 ASCFILE3
REC2 ASCFILE4 ASCFILE5
REC3 ASCFILE6 ASCFILE7
```

The following command loads the table:

```
db2 load from ascfile1 of asc
    lobs from /u/user1, /u/user1/bin
        modified by lobsinfile reclen=22
            method L (1 4, 6 13, 15 22)
        insert into table1
```

In the preceding example, the parameters function as follows:
- lobsinfile tells load that all LOB data is loaded from files.
- reclen=22 indicates that there is no newline character at the end of each input record, and that each record is 22 bytes long.
LOB data is contained in six files, ASCFILE2 through ASCFILE7. Each file contains the data that will be used to load a LOB column for a specific row. The relationship between LOBs and other data is specified in ASCFILE1. The first record of this file tells load to place REC1 in COL1 of row 1. The contents of ASCFILE2 are used to load LOB1 of row 1, and the contents of ASCFILE3 are used to load LOB2 of row 1. Similarly, ASCFILE4 and ASCFILE5 are used to load LOB1 and LOB2 of row 2, and ASCFILE6 and ASCFILE7 are used to load the LOBs of row 3.

The LOBS FROM parameter contains two paths that will be searched for the named LOB files when those files are required by load. To load LOBs directly from ASCFILE1 (a nondelimited ASCII file) without the lobsinfile modifier, the following rules must be observed:

- The total length of any record, including LOBs, cannot exceed 32KB.
- LOB fields in the input records must be of fixed length, and LOB data padded with blanks as necessary.
- The striptblanks modifier must be specified so that the trailing blanks used to pad LOBs can be removed as the LOBs are inserted into the database.

**Example: Using Dump Files**

Table FRIENDS has three columns:

```sql
(table friends "( c1 INT NOT null, c2 INT, c3 CHAR(8) )")
```

An attempt is made to load the following data records into this table:

```
23, 24, bobby
, 45, john
4,, mary
```

The second row is rejected because the first INT is null, and the column definition specifies NOT null. Columns that contain initial characters inconsistent with the DEL format will generate an error, and the record will be rejected. Such records can be written to a dump file. (See the “Valid File Type Modifiers (LOAD)” section earlier in this chapter.)

DEL data appearing in a column outside of character delimiters is ignored, but it does generate a warning. For example,

```
22,34,"bob"
24,55,"sam" sdf
```

“sam” is loaded in the third column of the table, and the characters sdf are flagged in a warning. The record is not rejected. Consider
22, 34, "bob"

22, 34, and “bob” are loaded, and load warns you that data in column 1 following the 22 was ignored. The record is not rejected.

**Example: Loading DATALINK Data**

The following command loads the table MOVIELABLE from the input file delfile1, which has data in DEL format:

```sql
db2 load from delfile1 of del
    modified by dldel|
    insert into movietable
    (actorname, description, url_making_of, url_movie)
    datalink specification (dl_url_default_prefix
    "http://narang"),
    (dl_url_replace_prefix "http://bomdel" dl_url_suffix
    ".mpeg")
    for exception excptab
```

The table has four columns:

- **actorname**: VARCHAR(n)
- **description**: VARCHAR(m)
- **url_making_of**: DATALINK (with LINKTYPE URL)
- **url_movie**: DATALINK (with LINKTYPE URL)

The DATALINK data in the input file has the vertical bar (|) character as the subfield delimiter. If any column value for url_making_of does not have the prefix character sequence, “http://narang” is used.

Each non-null column value for url_movie will get “http://bomdel” as its prefix. Existing values are replaced. Each non-null column value for url_movie will get “.mpeg” appended to the path. For example, if a column value of url_movie is “http://server1/x/y/z”, it is stored as “http://bomdel/x/y/z.mpeg”; if the value is “/x/y/z”, it is stored as “http://bomdel/x/y/z.mpeg”.

If any unique index or DATALINK exception occurs while loading the table, the affected records are deleted from the table and put into the exception table excptab.

**Example: Loading a Table with an Identity Column**

TABLE1 has four columns:

- **C1**: VARCHAR(30)
C2 INT GENERATED BY DEFAULT AS IDENTITY  
C3 DECIMAL(7,2)  
C4 CHAR(1)  

TABLE2 is the same as TABLE1, except that C2 is a GENERATED ALWAYS identity column.  
The following data records are in DATAFILE1 (DEL format):

"Liszt"
"Hummel", 187.43, H
"Grieg", 100, 66.34, G
"Satie", 101, 818.23, I  
The following data records are in DATAFILE2 (DEL format):

"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I  
The following command generates identity values for rows 1 and 2, because no identity values are supplied in DATAFILE1 for those rows. Rows 3 and 4, however, are assigned the user-supplied identity values of 100 and 101, respectively:

```
db2 load from datafile1.del of del replace into table1
```

To load DATAFILE1 into TABLE1 so that identity values are generated for all rows, issue one of the following commands:

```
db2 load from datafile1.del of del method P(1, 3, 4) replace into table1 (c1, c3, c4)
db2 load from datafile1.del of del modified by identityignore replace into table1
```

To load DATAFILE2 into TABLE1 so that identity values are generated for each row, issue one of the following commands:

```
db2 load from datafile2.del of del replace into table1 (c1, c3, c4)
db2 load from datafile2.del of del modified by identitymissing replace into table1
```
To load DATAFILE1 into TABLE2 so that the identity values of 100 and 101 are assigned to rows 3 and 4, issue the following command:

```
db2 load from datafile1.del of del modified by identityoverride replace into table2
```

In this case, rows 1 and 2 are rejected because load has been instructed to override system-generated identity values in favor of user-supplied values. If user-supplied values are not present, however, the row must be rejected because identity columns are implicitly not null.

If DATAFILE1 is loaded into TABLE2 without using any of the identity-related file type modifiers, rows 1 and 2 are loaded, but rows 3 and 4 are rejected because they supply their own non-null values, and the identity column is GENERATED ALWAYS.

Data is loaded in the sequence that appears in the input file. If a particular sequence is desired, the data should be sorted before the load. If clustering is required, the data should be sorted on the clustering index prior to loading. Load builds indexes based on existing definitions. Exception tables handle duplicates on unique keys.

Load does not enforce referential integrity, perform constraints checking, or update summary tables that depend on the tables being loaded. Tables that include referential or check constraints are placed in check pending state.

Summary tables that are defined with REFRESH IMMEDIATE and that depend on tables being loaded are also placed in check pending state. Issue SET INTEGRITY to take the tables out of check pending state. Loads cannot be carried out on replicated summary tables.

**Example: Remote Load**

The data file (/u/user/data.del) resides on a remotely connected client. The following command loads data.del into MYTABLE on the server database:

```
load client from /u/user/data.del of del modified by codepage=850 insert into mytable
```

**Example: Remove All Table Rows Without Logging with LOAD**

Empty IXF files used with IMPORT REPLACE and LOAD REPLACE can be used to delete all rows in a table as unlogged operations. IMPORT is usually a better choice than load because it does not lock the table space. LOAD must be used if the target table has referential integrity dependencies or summary tables defined on it. For example, the employee table in the sample data has RI constraints.

First, generate the empty IXF file with a SELECT that retrieves no rows:
export to c:\tmp\employee.ixf of ixf
select * from employee where salary in (0) and salary not in (0)

Then load the empty file back into the table with REPLACE:

load from c:\tmp\employee.ixf of ixf replace into employee

LOAD QUERY

LOAD QUERY checks the status of a load during processing. A connection to the same database and a separate CLP session are also required to successfully query a load.

Authorization
None.

Syntax

```
>>-LOAD QUERY---TABLE--table-name------------------------------->
>-----+-------------------------+---+-------------+------------->
'-TO--local-message-file--'   +-NOSUMMARY---+
|'-SUMMARYONLY-'               
>-----+------------+-------------------------------------------><
'-SHOWDELTA--'
```

Description

The following are the parameter descriptions for LOAD QUERY:

- **NOSUMMARY** Specifies that no load summary information (rows read, rows skipped, rows loaded, rows rejected, rows deleted, rows committed, and number of warnings) is reported.
- **SHOWDELTA** Specifies that only new information (pertaining to load events that have occurred since the last invocation of the LOAD QUERY command) is reported.
- **SUMMARYONLY** Specifies that only load summary information is reported.
- **TABLE table-name** Specifies the name of the table being loaded. If an unqualified name is specified, the table will be qualified with the CURRENT SCHEMA.
TO local-message-file  Specifies the destination for warning and error messages. This file cannot be the message-file specified for the LOAD command. If the file already exists, all messages that load has generated are appended to it.

Example
LOAD is issued against the STAFF table in the SAMPLE database. To query the load, use these commands:

db2 connect to sample
db2 load query table staff to /u/mydir/staff.tempmsg

The output file /u/mydir/staff.tempmsg looks like this:

```
SQL3500W The utility is beginning the "LOAD" phase at time "06-13-2001 19:40:29.645353".
SQL3519W Begin Load Consistency Point. Input record count = "0".
SQL3520W Load Consistency Point was successful.
SQL3109N The utility is beginning to load data from file "/u/mydir/data/staffbig.ixf".
SQL3150N The H record in the PC/IXF file has product "DB2 01.00", date "20010111", and time "194554".
SQL3153N The T record in the PC/IXF file has name "data/staffbig.ixf", qualifier " ", and source " ".
SQL3519W Begin Load Consistency Point. Input record count = "111152".
SQL3520W Load Consistency Point was successful.
SQL3519W Begin Load Consistency Point. Input record count = "222304".
SQL3520W Load Consistency Point was successful.
```

MIGRATE DATABASE
This command converts version 5 and version 6 DB2 databases to version 7.

Authorization
SYSADM.

Syntax
```
>>-MIGRATE----+-DATABASE-+--database-alias-------------------------->
   '-DB------'
```
Description

The following are the parameter descriptions for MIGRATE DATABASE:

- **DATABASE database-alias** Specifies the alias of the database to migrate to the currently installed version of the database manager.
- **USER username** Specifies the user name under which the database is migrated.
- **USING password** Specifies the password used to authenticate the user name. If the password is omitted but a user name is specified, you are prompted for it.

You can migrate a database only to a newer version, not backwards to a previous version. If an error occurs during migration, it may be necessary to issue TERMINATE before attempting the suggested user response. For example, if a log full error occurs during migration (SQL1704: Database migration failed. Reason code “3”), it will be necessary to issue the TERMINATE command before increasing the values of the database configuration parameters LOGPRIMARY and LOGFILSZ.

The CLP must refresh its database directory cache if the migration failure occurs after the database has already been relocated (which is likely to be the case when a log full error returns).

Example

The following example migrates the database cataloged under the database alias sales:

```
db2 migrate database sales
```

**PRECOMPILE PROGRAM**

The PRECOMPILE PROGRAM command processes an application program source file containing embedded SQL statements. A modified source file is produced, containing host language calls for the SQL statements and, by default, a package is created in the database. In EEE, PRECOMPILE updates the database catalogs on the catalog node. Its effects are visible to all nodes, and it can be run from any node.
Authorization

The following authorities are required for PRECOMPILE PROGRAM:

- SYSADM or DBADM
- BINDADD privilege if a package does not exist, and one of the following:
  - IMPLICIT_SCHEMA authority on the database if the schema name of the package does not exist
  - CREATEIN privilege on the schema if the schema name of the package exists
  - ALTERIN privilege on the schema if the package exists
  - BIND privilege on the package if it exists

The user also needs all privileges required to compile any static SQL statements in the application. Privileges granted to groups are not used for authorization checking of static statements. If the user has SYSADM authority but not explicit privileges to complete the bind, the database manager grants explicit DBADM authority automatically.

Syntax

```bash
>>>+-PRECOMPILE-++filename------------------------------------->
    '-PREP-------'
>-----+----------------------------------+-------------------->
    '-BINDFILE--+-------------------+--'
    '-USING--bind-file--'
>-----+-------------------------+------------------------------->
|           .-UNAMBIG--.  |
    '-BLOCKING--+-ALL------+--'
    '-NO-------'
>-----+---------------------+---+----------------------------+--> 
|           .-DEF--.  |   |                   .-NO--.  |
    '-DATETIME--+-EUR--+--'   '-DEFERRED_PREPARE--+-ALL-+--'
```
DB2 Command Line Processor Commands

+-ISO++
+jis++
+loc++
+usa+-

>-------------------------- +--------------------------+
| -degree-of-parallelism+-
| -any-------------------|

>-------------------------- +--------------------------+
| .-explicit---- |
| -disconnect+-automatic---|
| -conditional- |

>-------------------------- +--------------------------+
| .-run-- | .-no-- |
| -dynamicrules+-bind+++ | -explain++all+++ |
| +yes- |

>-------------------------- +--------------------------+
| .-no-- | .-no-- |
| -explanet++all+++ | -federated++yes+++ |
| +yes- |

>-------------------------- +--------------------------+
| .-,-------------- | .-def-- |
| v | -insert++buf+++ |
| -functpath-----schema-name---- | |

>-------------------------- +--------------------------+
| .-cs-- | .-saa1--- |
| -isolation++rr+++ | -langlevel++mia++++ |
| +rs++ | -sql92e- |
| -ur- |

>-------------------------- +--------------------------+
| .-no-- | .-no-- |
| -longerror++yes+++ |

>-------------------------- +--------------------------+
| -messages--message-file-- | -nolinemacro- |
>-----+-------------------+---+-------------------+------------->
   |           .-0--.  |   '-OUTPUT--filename--'
   | '-OPTLEVEL--+1--+--'
>-----+--------------------------+------------------------------>
   |-OWNER--authorization-id--'
>-----+------------------------------------+-------------------->
   |-PACKAGE--+----------------------+--'
   | '-USING--package-name--'
>-----+-------------------------------------------+------------->
   |-PREPROCESSOR--+"preprocessor-command"++--'
   | '-'preprocessor-command'-'
>-----+-------------------------------------------+------------->
   |-QUALIFIER--qualifier-name--'
>-----+-------------------------------------------+------------->
   |-QUERYOPT--optimization-level--'   | .-SAA--.   |
   | '-SQLCA---NONE++--'
>-----+-------------------------------------------+------------->
   |'-NOPACKAGE--.  |
   |'--------SQLERROR--+-CHECK------+--'
   |'-CONTINUE--'
>-----+-------------------------------------------+------------->
   |-SQLFLAG--+-SQL92E----+---SYNTAX--'
   | '+MVSD3B2V23++
   | '+MVSD3B2V31++
   | '-MVSD3B2V41-'
>-----+-------------------------------------------------------->
   | '.DB2--.   | '.-YES--.   |
   | '-SQLRULES--+-STD+-++'   '| '-SQLWARN--+-NO---;++'
>-----+-------------------------------------------------------->
   | '.ONEPHASE--.   | 'SYNTAX--'
   | '-SYNCPOINT--+-NONE---;++'
   | '-TWOPHASE--'
Description

PRECOMPILE produces a modified source file that contains language equivalents to the SQL statements. By default, a package is created in the database to which a connection has been established. The name of the package is the same as the filename (minus the extension and folded to uppercase), up to a maximum of eight characters. Creating a package with a schema name that does not already exist results in the implicit creation of that schema. The schema owner will be SYSIBM, and the CREATEIN privilege on the schema is granted to PUBLIC.

During precompilation, an Explain Snapshot is not taken unless a package is created and explsnap has been specified. The snapshot is put into the Explain tables of the user creating the package. Similarly, Explain table information is captured only when explain is specified and a package is created.

Precompiling stops if a fatal error or more than 100 errors occur. If a fatal error occurs, precompiling stops, attempts to close all files, and discards the package. If a package is bound with dynamicrules bind, the implicit or explicit value of the bind option owner is used for authorization checking of dynamic SQL statements, and the implicit or explicit value of the bind option qualifier is used as the implicit qualifier of unqualified objects within dynamic SQL statements. If multiple packages are referenced during a single connection, dynamic SQL statements prepared by a specific package behave according to the bind options for that package. The value of the special register CURRENT SCHEMA has no effect on qualification in a package bound with dynamicrules bind.

- **filename** Specifies the source file to be precompiled. This can be one of the following:
- **.sqc** for C applications Generates a .c file
.sqx (OS/2 or Windows), or .sQC (UNIX) for C++ applications  Generates a .cxx file on OS/2 or Windows, or a .C file on UNIX

.sqb for COBOL applications  Generates a .cbl file

.sqf for Fortran applications  Generates a .for file on OS/2 or Windows or a .f file on UNIX

The preferred extension for C++ applications containing embedded SQL on UNIX is sqC; however, the sqx convention, invented for systems that are not case-sensitive, is tolerated by UNIX.

BINDFILE  Results in the creation of a bind file. A package is not created unless the package option is also specified. If a bind file is requested but no package is to be created, as in the following example:

db2 prep sample.sqc bindfile yes

The object existence and authentication SQLCODEs will be treated as warnings instead of errors. This allows a bind file to be successfully created, even if the database being used for precompilation does not have all of the objects referred to in static SQL statements within the application. The bind file can be successfully bound, creating a package, once the required objects have been created.

USING bind-file  Specifies the name of the bind file generated by the precompiler. The filename must have an extension of .bnd. If a filename is not entered, the precompiler uses the name of the program (entered as the filename parameter) and adds the .bnd extension. If a path is not provided, the bind file is created in the current directory.

BLOCKING  Specifies blocking for cursors:

ALL block  For read-only cursors and cursors not specified as FOR UPDATE OF. Ambiguous cursors are treated as read-only.

NO  Do not block any cursors. Ambiguous cursors are treated as updateable.

UNAMBIG block  For read-only cursors and cursors not specified as FOR UPDATE OF. Ambiguous cursors are treated as updateable.

COLLECTION schema-name  Specifies a 30-character collection identifier for the package. If not specified, the authorization identifier for the user processing the package is used.

CONNECT  Specifies one of two possible CONNECT types:
The CONNECT is a type 1 CONNECT.

The CONNECT is a type 2 CONNECT.

**DATETIME** Specifies the date and time format to be used. Choose one of the following to identify the date and time format:

- **DEF** Use a date and time format associated with the country code of the database.
- **EUR** Date is dd.mm.yyyy. Time is hh:mm:ss.
- **ISO** Date is yyyy-mm-dd. Time is hh:mm:ss.
- **JIS Japanese Industrial Standard** Date is yyyy-mm-dd. Time is hh:mm:ss.
- **LOC** Use the date and time format in local form associated with the country code of the database.
- **USA** Date is mm/dd/yyyy. Time is hh:mm A.M. or P.M.

**DEFERRED PREPARE** Provides a performance enhancement by combining the SQL PREPARE statement flow with the associated OPEN, DESCRIBE, or EXECUTE statements to minimize client/server communication.

- **NO** The PREPARE statement is executed when it is issued.
- **YES** Execution of the PREPARE statement is deferred until the corresponding OPEN, DESCRIBE, or EXECUTE statement is issued. The PREPARE statement will not be deferred if it uses the INTO clause, which requires an SQLDA to be returned immediately. However, if the PREPARE INTO statement is issued for a cursor that does not use any parameter markers, the processing will be optimized by pre-OPENing the cursor when the PREPARE is executed.
- **ALL** Same as YES, except that a PREPARE INTO statement is also deferred. If the PREPARE statement uses the INTO clause to return an SQLDA, the application must not reference the content of this SQLDA until either the OPEN, DESCRIBE, or EXECUTE statement is issued and returned.

**DEGREE** Specifies the degree of parallelism for the execution of static SQL statements in an SMP system. This option does not affect CREATE INDEX parallelism.

- **1** The statement will not use parallelism.
- **degree-of-parallelism** The degree of parallelism with which the statement can be executed, a value from 2 to 32,767.
- **ANY** Determines the parallelism that execution of the statement can use.

**DISCONNECT** Specifies when database connections are disconnected.
AUTOMATIC  All database connections are terminated after a commit.

CONDITIONAL  Database connections marked RELEASE or with no open WITH HOLD cursors are disconnected at commit.

EXPLICIT  Only database connections explicitly marked for release by RELEASE statements are disconnected at commit.

DYNAMICRULES  Defines rules for dynamic SQL at run time for initial settings for authorization ID and implicit qualification of unqualified object references.

RUN  The authorization ID of the user executing the package is used. This is the default.

BIND  All rules that apply to static SQL for authorization and qualification are used at run time. The authorization ID of the package owner is used for authorization checking of dynamic SQL statements, and the default package qualifier is used for implicit qualification of unqualified object references within dynamic SQL statements. When binding a package with this option, the binder of the package should not have any authorities that the user of the package should not receive, because dynamic SQL statements will be using the authorization ID of the package owner. The following dynamically prepared SQL statements cannot be used within a package that has been bound with this option: GRANT, REVOKE, ALTER, CREATE, DROP, COMMENT ON, RENAME, SET CONSTRAINTS, and SET EVENT MONITOR STATE.

EXPLAIN  Stores information in the Explain tables about the access plans chosen for each SQL statement in the package.

NO  Explain information is not captured.

YES  Explain tables are populated with information about the chosen access plan at prep/bind time for static statements and at run time for incremental bind statements.

ALL  Explain information for each eligible static SQL statement is placed in the Explain tables at prep/bind time. Explain information for each eligible incremental bind SQL statement is placed in the Explain tables at run time. In addition, Explain information is gathered for eligible dynamic SQL statements at run time, even if the CURRENT EXPLAIN SNAPSHOT register is set to NO. This option is not supported by DRDA databases (like DB2 on OS/390).

EXPLSNAP  Stores Explain Snapshot information in the Explain tables. This DB2 precompile/bind option is not supported by DRDA databases.

NO  An Explain Snapshot is not captured.

YES  An Explain Snapshot for each eligible static SQL statement is placed in the Explain tables at prep/bind time for static statements and at run time for incremental bind statements.

ALL  An Explain Snapshot for each eligible static SQL statement is placed in the Explain tables at prep/bind time. Explain Snapshot information for each eligible incremental bind SQL statement is placed in the Explain tables at
run time. In addition, Explain Snapshot information is gathered for eligible dynamic SQL statements at run time, even if the CURRENT EXPLAIN SNAPSHOT register is set to NO.

- **FEDERATED** Specifies whether a static SQL statement in a package references a nickname or a federated view. If this option is specified and a nickname or federated view is not referenced, an error is returned and the package is not created. If this option is not specified and a nickname or federated view is referenced by a static statement in the package, an error is returned and the package is not created.

- **NO precompile** Connects to a DB2 Universal Database. This is the default.
- **YES precompile** Accesses a DB2 federated system.

- **FUNCPATH** Specifies the function path used to resolve user-defined distinct types and functions in static SQL. If not specified, the default function path is “SYSIBM”, “SYSFUN”, USER, where USER is the value of the USER special register. FUNCPATH is not supported by DRDA databases.

- **schema-name** An SQL identifier, ordinary or delimited, identifying a schema that exists at the application server. No validation that the schema exists is made at precompile or bind time. The same schema cannot appear more than once in the function path. The number of schemas specified is limited by the length of the resulting function path, which cannot exceed 254 bytes. Don’t specify the SYSIBM schema—it is implicitly assumed to be the first schema.

- **INSERT** Specifies that data inserts be buffered to improve performance for a program precompiled or bound against DB2 EEE.
- **BUF** Inserts from an application are buffered.
- **DEF** Inserts from an application are not buffered.

- **ISOLATION** Trades off concurrency for faster processing against more exclusive access to data.
- **CS** Cursor Stability is the isolation level.
- **RR** Repeatable Read is the isolation level.
- **RS** Read Stability is the isolation level.
- **UR** Uncommitted Read is the isolation level. Also known as dirty read.

- **LANGLEVEL** Specifies the SQL rules that apply for the syntax and the semantics for both static and dynamic SQL in the application. This option is not supported by DRDA databases (connections through DB2 Connect).

You declare an SQLCA in the application for SQLCODE or SQLSTATE checking.
C null-terminated strings are padded with blanks and always include a null-terminating character, even if truncation occurs.

The FOR UPDATE clause is optional for all columns updated in a positioned UPDATE.

A searched UPDATE or DELETE requires the SELECT privilege on the object table of the UPDATE or DELETE statement if a column of the object table is referenced in the search condition or on the righthand side of the assignment clause.

A column function that can be resolved using an index (for example, MIN or MAX) will also check for nulls and return warning SQLSTATE 01003 if there are any nulls.

An error is returned when no privilege is granted and the grantor has no privileges on the object.

- **SAA1:**

Declare an SQLCA in the application to support error SQLCODE or SQLSTATE checking. C null-terminated strings are not terminated with a null character if truncation occurs.

The FOR UPDATE clause is required for all columns updated in a positioned UPDATE.

A searched UPDATE or DELETE will not require the SELECT privilege on the object table of the UPDATE or DELETE statement unless a fullselect in the statement references the object table.

A column function that can be resolved using an index (for example, MIN or MAX) will not check for nulls.

- **SQL92E** defines the ISO/ANS SQL92 rules as follows:

To support checking of SQLCODE or SQLSTATE values, variables may be declared in the host variable declare section (if neither is declared, SQLCODE is assumed during precompilation).

C null-terminated strings are padded with blanks and always include a null-terminating character, even if truncation occurs.

The FOR UPDATE clause is optional for all columns updated in a positioned UPDATE.

A searched UPDATE or DELETE requires SELECT privilege on the object table of the UPDATE or DELETE statement if a column of the object table is referenced in the search condition or on the righthand side of the assignment clause.

A column function that can be resolved using an index (for example MIN or MAX) will also check for nulls and return warning SQLSTATE 01003 if there are nulls.

An error is returned when no privilege is granted and the grantor has no privileges on the object (otherwise a warning is returned).

- **LONGERROR** Indicates whether long host variable declarations are treated as errors. For portability, sqlint32 can be used as a declaration for an INTEGER column in precompiled C and C++.

- **NO** Does not generate errors for long host variable declarations. This is the default for 32 bit systems and for 64 bit NT systems where long host variables...
may be used as declarations for INTEGER columns. The use of this option on 64 bit UNIX will allow long host variables to be used as declarations for BIGINT columns.

- **YES** Generates errors for the use of long host variable declarations. This is the default for 64 bit UNIX.

- **MESSAGES message-file** Specifies the destination for warning, error, and completion status messages. A message file is created whether the bind is successful or not. If a message file name is not specified, the messages are written to standard output. If the complete path to the file is not specified, the current directory is used. If using the name of an existing file, the contents of the file are overwritten.

- **NOLINEMACRO** Suppresses the generation of the #line macros in the output .c file (this is for C/C++ only). This is useful when the file is used with development tools that require source line information, such as profiles, cross-reference utilities, and debuggers.

- **OPTLEVEL** Indicates whether C/C++ precompilation optimizes initialization of internal SQLDA s when host variables are used in SQL statements. Such optimization increases performance when a single SQL statement (such as FETCH) is used inside a tight loop.
  - **0** The precompiler does not optimize SQLDA initialization.
  - **1** The precompiler optimizes SQLDA initialization. Do not choose 1 (optimization) if the application uses pointer host variables, as in the following example:

```sql
exec sql begin declare section;
char (*name)[20];
short *id;
exec sql end declare section;
```

Do not choose 1 (optimization) if the application uses C++ data members directly in SQL statements.

- **OUTPUT filename** Overrides the default name of the modified source file produced by the compiler. It can include a path.

- **OWNER authorization-id** Designates a 30-character authorization identifier for the package owner. The owner must have the privileges required to execute the SQL statements contained in the package. Only a user with SYSADM or DBADM authority can specify an authorization identifier other than the user ID. The default is the primary authorization ID of the precompile/bind process. SYSIBM, SYSCAT, and SYSSTAT cannot be used as OWNERs.

- **PACKAGE** Creates a package.
- **USING package-name** Specifies the name of the package generated by the precompiler. If a name is not entered, the name of the application program source file (minus extension and folded to uppercase) is used. Maximum length is 8 characters.

- **PREPROCESSOR “preprocessor-command”** Specifies the preprocessor command that can be executed by the precompiler before it processes embedded SQL statements. The preprocessor command string (maximum length 1024 bytes) must be enclosed either by double or by single quotation marks. This option enables the use of macros within the declare section. A valid preprocessor command is one that can be issued from the command line to invoke the preprocessor without specifying a source file. For example:

  ```
xlc -P -DMYMACRO=0
  ```

- **QUALIFIER qualifier-name** Specifies a 30-character implicit qualifier for unqualified objects contained in the package. The default is the owner’s authorization ID, whether or not the owner is explicitly specified.

- **QUERYOPT optimization-level** Specifies the level of optimization for all static SQL statements contained in the package. The default is 5. This is not supported by DRDA.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Perform minimal optimization. Query rewrite not active. This is most suitable for simple dynamic SQL access to well-indexed tables.</td>
</tr>
<tr>
<td>1</td>
<td>Optimization roughly comparable to DB2/UNIX Version 1. No query rewrite.</td>
</tr>
<tr>
<td>2</td>
<td>A level of optimization higher than that of DB2 Version 1, but at significantly less optimization cost than levels 3 and above, especially for very complex queries.</td>
</tr>
<tr>
<td>3</td>
<td>A moderate amount of optimization to generate an access plan.</td>
</tr>
<tr>
<td>5</td>
<td>A significant amount of optimization to generate an access plan. For complex dynamic SQL queries, heuristic rules are used to limit the amount of time spent selecting an access plan. Where possible, queries will use summary tables instead of underlying base tables.</td>
</tr>
<tr>
<td>7</td>
<td>A significant amount of optimization to generate an access plan. Similar to 5 but without the heuristic rules.</td>
</tr>
<tr>
<td>9</td>
<td>A maximal amount of optimization to generate an access plan. This can greatly expand the number of possible access plans that are evaluated. Use 9 to determine whether a better access plan can be generated for very complex and very long-running queries using large tables. Explain that performance measurements can be used to verify that a better plan has been generated.</td>
</tr>
</tbody>
</table>
SQLCA  Specifies, only for Fortran, whether the modified source code is consistent with the SAA definition.
- NONE  The modified source code is not consistent with the SAA definition.
- SAA   The modified source code is consistent with the SAA definition.

SQLERROR Indicates whether DB2 will create a package or a bind file if an error is encountered. SQLERROR CONTINUE is useful for applications that are bound against one database, but use tables in multiple databases (such as Federated or two-phase commit applications).

CHECK  The target system performs all syntax and semantic checks on the SQL statements being bound. A package will not be created as part of this process. If, while binding, an existing package with the same name and version is encountered, the existing package is neither dropped nor replaced even if the replace action was specified.

CONTINUE Creates a package, even if errors occur when binding SQL statements. Statements that fail to bind for authorization or existence reasons can be incrementally bound at execution time if VALIDATE RUN is also specified. Any attempt to execute them at run time generates an error (SQLCODE -525, SQLSTATE 51015).

NOPACKAGE A package or a bind file is not created if an error is encountered.

SQLFLAG Reports on deviations from the SQL language syntax specified with SQLFLAG. A bind file or a package is created only if the bind file or the package option is specified, in addition to the sqlflag option. Deviations are reported in the precompiler listing. Local syntax checking is performed only with these options:
- bind file
- package
- sqlerror check
- syntax

If sqlflag is not specified, the flagger function is not invoked, and the bind file or the package is not affected.

SQL92E SYNTAX SQL statements are checked against ANSI/ISO SQL92 entry-level SQL language format and syntax except for syntax rules that require access to the database catalog.

MVSD2V23 SYNTAX SQL statements are checked against MVS DB2 version 2.3 SQL language syntax.

MVSD2V31 SYNTAX SQL statements are checked against MVS DB2 version 3.1 SQL language syntax.

MVSD2V41 SYNTAX SQL statements are checked against MVS DB2 version 4.1 SQL language syntax.
**SQLRULES** Specifies whether type 2 CONNECTs are processed according to the DB2 rules or the Standard (STD) rules based on ISO/ANS SQL92. Also specifies how a user or application can specify the format of LOB answer set columns.

- DB2 permits the SQL CONNECT statement to switch the current connection to another established (dormant) connection. The application can specify the format of a LOB column only during the first fetch request.
- STD permits the SQL CONNECT statement to establish only a new connection. The SQL SET CONNECTION statement must be used to switch to a dormant connection. The user or application can change the format of a LOB column with each fetch request.

**SQLWARN** Indicates whether warnings are returned from the compilation of dynamic SQL statements (via PREPARE or EXECUTE IMMEDIATE), or from describe processing (via PREPARE...INTO or DESCRIBE). This is not supported by DRDA.

- NO Warnings are not returned from the SQL compiler, except SQLCODE +238, which will be returned.
- YES Warnings are returned from the SQL compiler.

**SYNCPOINT** Specifies how commits or rollbacks are coordinated among multiple database connections.

- NONE No TM is used to perform a two-phase commit. A COMMIT is sent to each participating database. The application is responsible for recovery if any of the commits fail.
- ONEPHASE No TM is to be used to perform a two-phase commit. A one-phase commit is used to commit the work done by each database in multiple database transactions.
- TWOPHASE A TM coordinates two-phase commits among the databases that support two-phase commit.

**SYNTAX** Specifies that no package or bind file is created during precompilation. Use this to check the validity of the source file without modifying or altering existing packages or bind files. Syntax is a synonym for SQLERROR check. If SYNTAX is used together with the PACKAGE option, no package is created.

**TARGET** Specifies that the precompiler should produce modified code tailored to one of the supported compilers on the current platform.

- IBMCOB On AIX, code is generated for the IBM COBOL Set for AIX compiler. On OS/2, code is generated for the IBM VisualAge for COBOL compiler.
- MFCOB Code is generated for the Micro Focus COBOL compiler. This is the default.
- **ANSI_COBOL** Code compatible with the ANS X3.23-1985 standard is generated.
- **C** Code compatible with the C compilers supported by DB2 on the current platform is generated.
- **CPLUSPLUS** Code compatible with the C++ compilers supported by DB2 on the current platform is generated.
- **Fortran** Code compatible with the Fortran compilers supported by DB2 on the current platform is generated.

**TRANSFORM GROUP** Specifies the transform group name to be used by static SQL statements for exchanging user-defined structured type values with host programs. This transform group is not used for dynamic SQL statements or for the exchange of parameters and results with external functions or methods.

**groupname** Specifies an SQL identifier up to 18 characters long. A group name cannot include a qualifier prefix and cannot begin with the prefix SYS because this is reserved for database use. In a static SQL statement that interacts with host variables, the name of the transform group to be used for exchanging values of a structured type is the first of these that applies:

- The group name in the TRANSFORM GROUP bind option, if any
- The group name in the TRANSFORM GROUP prep option as specified at the original precompilation time, if any
- The DB2_PROGRAM group, if a transform exists for the given type whose group name is DB2_PROGRAM

No transform group is used if none of the previous conditions exist.

The following errors are possible during the bind of a static SQL statement:

- With the SQLCODE yyy, SQLSTATE xxxx error, a transform is needed, but no static transform group has been selected.
- With the SQLCODE yyy, SQLSTATE xxxx error, the selected transform group does not include a necessary transform (TO SQL for input variables, FROM SQL for output variables) for the data type that needs to be exchanged.
- With the SQLCODE yyy, SQLSTATE xxxx error, the result type of the FROM SQL transform is not compatible with the type of the output variable, or the parameter type of the TO SQL transform is not compatible with the type of the input variable.

**VALIDATE** Determines when the database manager checks for authorization errors and object not found errors. The package owner authorization ID is used for validity checking. VALIDATE RUN is useful for applications bound against one database involving tables in other databases.

**BIND** Validation is performed at precompile/bind time. If all objects do not exist or all authority is not held, error messages are produced. If sqlerror
continue is specified, a package/bind file is produced despite the error message, but the statements in error are not executable.

- **RUN** Validation is attempted at bind time. If all objects exist and all authority is held, no further checking is performed at execution time. If all objects do not exist or all authority is not held at precompile/bind time, warning messages are produced, and the package is successfully bound, regardless of the sqlerror continue option setting. However, authority checking and existence checking for SQL statements that failed these checks during the precompile/bind process is retried at execution time.

- **WCHARTYPE** Allows application data in wide-character format to be converted to/from multibyte format when it is exchanged with the database engine. This is used with Unicode and Double Byte character sets (such as Japanese, Chinese, and Korean).

- **CONVERT** Host variables declared using the wchar_t base type are treated as containing data in wchar_t format. Because this format is not directly compatible with the format of graphic data stored in the database (DBCS format), input data in wchar_t host variables is implicitly converted to DBCS format on behalf of the application, using the ANSI C wcstombs() function. Similarly, output DBCS data is implicitly converted to wchar_t format, using mbstowcs(), before being stored in host variables.

- **NOCONVERT** Host variables declared using the wchar_t base type are treated as containing data in DBCS format. This is the format used within the database for graphic data; it is, however, different from the native wchar_t format implemented in C. Using noconvert means that graphic data will not undergo conversion between the application and the database, improving efficiency. However, the application is responsible for ensuring that data in wchar_t format is not passed to the database manager. When this option is used, wchar_t host variables should not be manipulated with the C-wide character string functions, and should not be initialized with wide-character literals (L-literals).

### PRUNE HISTORY/LOGFILE

This command deletes entries from the recovery history file and log files from the active log file path. Deleting entries from the recovery history file may be necessary if the file becomes excessively large and the retention period is high. Deleting log files from the active log file path may be necessary if logs are archived manually (rather than through a user exit program).

The history file is used not only by the DBA responsible for the backup and disaster recovery plan for the databases, but also by DB2 itself. Incremental Restore uses the history file to establish the required chain of backup images and determine the starting point (the youngest backup image containing the entire table space(s) being restored).
Pruning backup entries from the history file causes related file backups on DB2 Data Links Manager servers to be deleted as well.

**Authorization**

One of the following authorities is required: SYSADM, SYSCTRL, SYSMAINT, or DBADM.

**Syntax**

```
>>-PRUNE----+-HISTORY--timestamp----+--------------------+-+---><
|                       '-WITH FORCE OPTION--' |
| '-LOGFILE PRIOR TO--log-file-name--------------'
```

**Description**

The following are the parameter descriptions for PRUNE:

- **HISTORY timestamp** Identifies a range of entries in the recovery history file to delete. A complete timestamp (in the form yyyyymmddhhmss) or an initial prefix (minimum yyyy) can be specified. All entries with timestamps equal to or less than the timestamp provided are deleted from the recovery history file.

- **WITH FORCE OPTION** Specifies that entries be pruned according to the timestamp specified, even if some entries from the most recent restore set are deleted from the file. A restore set is the most recent full database backup including any restores of that backup image. If this is not specified, all entries from the backup image forward will be maintained in the history file.

- **LOGFILE PRIOR TO log-file-name** Specifies a string for a log file name, for example, S0000100.LOG. All log files prior to (but not including) the specified log file get deleted. The LOGRETAIN database configuration parameter must be set to RECOVERY or CAPTURE.

**Example**

To remove from the recovery history file the entries for all restores, loads, table space backups, and full database backups taken before and including December 1, 2000, enter the following:

```
db2 prune history 200012
```

200012 is interpreted as 20001201000000.
QUERY CLIENT

QUERY CLIENT returns current connection settings for an application process.

Authorization

None.

Syntax

>>>QUERY CLIENT

QUIESCE TABLESPACES FOR TABLE

This command quiesces table spaces for a table to enable LOAD. There are three valid quiesce modes: share, intent to update, and exclusive. Three possible states can result from running quiesce: quiesced share, quiesced update, and quiesced exclusive.

In a single-partition environment, this command quiesces all table spaces involved in a load in exclusive mode for the duration of the load. In an EEE database, quiesce acts locally on a node. Only the portion of table spaces belonging to the node where the load is performed are quiesced.

Authorization

One of the following:

- SYSADM
- SYSCTRL
- SYSMAN
- DBADM
- LOAD

Syntax

>>>QUIESCE TABLESPACES FOR TABLEtablename

'+SHARE'

'-INTENT TO UPDATE+'

'-EXCLUSIVE+'

'-RESET'
Description

A quiesce is a persistent lock. Unlike other locks, it persists across transaction failures, connection failures, and even across system failures (such as power failure or reboot). QUIESCE is not supported for declared temporary tables.

The database connection owns the quiesce. If the connection is lost, the quiesce remains, but it has no owner, and it is called a phantom quiesce. A phantom quiesce becomes “owned” by the next connection that issues the QUIESCE TABLESPACES FOR TABLE command against the same table spaces or table. For example, if a power outage caused a load to be interrupted during the delete phase, the table spaces for the loaded table would be left in delete pending, quiesce exclusive state. Upon database restart, this quiesce would be an unowned (or phantom) quiesce.

To remove a phantom quiesce perform these steps:

1. Connect to the database.
2. Use LIST TABLESPACES to determine which table space is quiesced.
3. Requiesce the table space using the current quiesce state. For example,

   db2 quiesce tablespaces for table mytable exclusive

Once this is completed, the new connection owns the quiesce, and the load can be restarted.

There is a limit of five quiescers on a table space at any given time. A quiescer can upgrade the state of a table space from a less restrictive state to a more restrictive one (for example, S to U, or U to X). If a user requests a state lower than one that is already held, the original state is returned. States are not downgraded.

The parameters for the QUIESCE TABLESPACES FOR TABLE are as follows:

- **TABLE tablename** Specifies the unqualified table name. The table cannot be a system catalog table.
- **schema.tablename** Specifies the qualified table name. If schema is not provided, the CURRENT SCHEMA will be used.
- **SHARE** Specifies that the quiesce is to be in share mode. When a “quiesce share” request is made, the transaction requests intent share locks for the table spaces and a share lock for the table. When the transaction obtains the locks, the state of the table spaces is changed to quiesced share. The state is granted to the quiescer only if there is no conflicting state held by other users. The state of the table spaces, along with the authorization ID and the database agent ID of the quiescer, are recorded in the table space table so that the state is persistent. The table cannot be changed while the table spaces for the table are in quiesced share state. Other share mode requests to the table and table spaces are allowed. When the transaction
commits or rolls back, the locks are released, but the table spaces for the table remain in quiesced share state until the state is explicitly reset.

- **INTENT TO UPDATE** Specifies that the quiesce is to be in INTENT TO UPDATE mode. When a “quiesce intent to update” request is made, the table spaces are locked in INTENT EXCLUSIVE (IX) mode, and the table is locked in UPDATE (U) mode. The state of the table spaces is recorded in the table space table.

- **EXCLUSIVE** Specifies that the quiesce is to be in EXCLUSIVE mode. When a “quiesce exclusive” request is made, the transaction requests super exclusive locks on the table spaces and a super exclusive lock on the table. When obtained, the state of the table spaces changes to quiesced exclusive. The state of the table spaces, along with the authorization ID and the database agent ID of the quiescer, are recorded in the table space table. Because the table spaces are held in SUPER EXCLUSIVE mode, no other access to the table spaces is allowed. The user who invokes the quiesce function (the quiescer) has exclusive access to the table and the table spaces.

- **RESET** Releases quiesce locks from the table space.

### Examples

Here’s how to quiesce table spaces:

```
db2 quiesce tablespaces for table staff share
db2 quiesce tablespaces for table boss.org intent to update
```

The following example removes a phantom quiesce. First connect to the database

```
db2 connect to sample
```

Then use the LIST TABLESPACES command to determine which table space is quiesced:

```
db2 list tablespaces
```

Requiesce the table space using the current quiesce state. For example,

```
db2 quiesce tablespaces for table mytable exclusive
```

Once completed, the new connection owns the quiesce and the load can be restarted.
QUIT exits the command line processor interactive input mode and returns to the
operating system command prompt. If a batch file is being used to send commands to
the command line processor, commands are processed until QUIT, TERMINATE, or
the end-of-file is encountered.

**Authorization**
None.

**Syntax**

```sql
>>>QUIT
```

**Description**
QUIT does not terminate the command line processor back-end process or break a
database connection. CONNECT RESET breaks a connection, but it does not terminate
the back-end process. TERMINATE does both.

---

**REBIND**
REBIND allows the user to re-create a package stored in the database without the need
for a bind file.

**Authorization**
One of the following:

- SYSADM or DBADM
- ALTERIN privilege on the schema
- BIND privilege on the package

The authorization ID logged in the BOUNDBY column of the SYSCAT.PACKAGES
system catalog table, which is the ID of the most recent binder of the package, is used as
the binder authorization ID for the rebind, and for the default schema for table references
in the package. This default qualifier may be different from the authorization ID of the
user executing the rebind request. REBIND will use the same bind options that were
specified when the package was created.
Syntax

```bash
>>-REBIND-+--------+--package-name--+-CONSERVATIVE-+--->
   '-PACKAGE-'
   '-ANY---
   >-/>RESOLVE--+-CONSERVATIVE-+--->
```

Description

REBIND provides a quick way to re-create a package. This enables you to take advantage of a change in the system without a need for the original bind file. For example, if it is likely that a particular SQL statement can take advantage of a newly created index, the REBIND command can be used to re-create the package. REBIND can also be used to re-create packages after RUNSTATS has been executed to take advantage of new statistics.

REBIND can also re-create inoperative packages. Inoperative packages must be explicitly rebound with bind or rebind. A package is marked inoperative (the VALID column of the SYSCAT.PACKAGES system catalog will be set to X) if a function on which the package depends is dropped.

REBIND also gives users control over the rebinding of invalid packages. Invalid packages will be automatically (or implicitly) rebound by DB2 when executed. This may result in a noticeable delay in the execution of the first SQL request for the invalid package. If you explicitly rebind invalid packages, rather than allow DB2 to rebind them at run time, you eliminate initial delay and can find unexpected SQL error messages before the application is run.

The choice of whether to use BIND or REBIND to explicitly rebind a package depends on the circumstances, but REBIND is usually a better choice. BIND must be used if any of the following apply:

- The program has changed.
- The package does not match the executable for the program.
- You want to modify bind options. For example, BIND has a grant option.
- The package does not currently exist in the database.
- You want to detect all bind errors. REBIND returns only the first error it detects. BIND returns the first 100.

The following parameters can be used with REBIND:

- **PACKAGE package-name** Specifies the qualified or unqualified name that designates the package to rebind. An unqualified package name is implicitly qualified by the current authorization ID.
RESOLVE  Specifies whether rebinding of the package is performed with or without conservative binding semantics. This affects whether new functions and data types are considered during function resolution and type resolution on static DML statements in the package. RESOLVE is not supported by DRDA databases like DB2 for OS/390. The choices are as follows:

- ANY  All of the functions and types in the SQL path are considered for function and type resolution. Conservative binding semantics are not used. This is the default.
- CONSERVATIVE  Only functions and types in the SQL path defined before the last explicit bind timestamp are considered for function and type resolution. Conservative binding semantics are used.

REBIND does not automatically commit the transaction following a successful rebind. The user must explicitly commit the transaction. This enables “what if” analysis, in which the user updates certain statistics and then tries to rebind the package to see what changes. It also permits multiple rebinds within a unit of work.

The REBIND command will commit the transaction if autocommit is enabled. If REBIND is executed on a package that is in use by another user, the rebind will not occur until the other user’s logical unit of work ends, because an exclusive lock is held on the package’s record in the SYSCAT.PACKAGES system catalog table during the rebind.

When REBIND is executed, the database manager re-creates the package from the SQL statements stored in the SYSCAT.STATEMENTS system catalog table. If REBIND encounters an error, processing stops, and an error message is returned.

REBIND will reexplain packages that were created with the explsnap bind option set to YES or ALL (indicated in the EXPLAIN_SNAPSHOT column in the SYSCAT.PACKAGES catalog table entry for the package) or with the explain bind option set to YES or ALL (indicated in the EXPLAIN_MODE column in the SYSCAT.PACKAGES catalog table entry for the package). The Explain tables used are those of the REBIND requester, not the original binder.

If an SQL statement was found to be in error and the BIND option SQLERROR CONTINUE was specified, the statement will be marked as invalid even if the problem has been corrected. REBIND will not change the state of an invalid statement. In a package bound with VALIDATE RUN, a statement can change from static to incremental bind or incremental bind to static across a REBIND depending on whether object existence or authority problems exist during the REBIND.

RECONCILE  RECONCILE validates references to files for the DATALINK data of a table. The rows for which the references to files cannot be established are copied to an exception table (if specified), and modified in the input table.
RECONCILE produces a message file (reconcil.msg) in the instance path on UNIX, and in the install path on Windows NT and OS/2. This contains warning and error messages generated during validation of the exception table.

Authorization

One of the following:

- SYSADM
- SYSCTRL
- SYSMAINT
- DBADM
- CONTROL privilege on the table

Syntax

```
>>>-RECONCILE--table-name---DLREPORT--filename------------------->
>-----+----------------------------+---------------------------><
'-FOR EXCEPTION--table-name--'
```

Description

The following are the parameter descriptions for RECONCILE:

- **RECONCILE table-name** Specifies the table to run reconciliation on. An alias or the fully qualified or unqualified table name can be specified. A qualified table name is in the form schema.tablename. If an unqualified table name is specified, the table will be qualified with the current authorization ID.

- **DLREPORT filename** Specifies the list of files that are unlinked during reconciliation. The name must be fully qualified (for example, /u/johnh/ report). Reconcile appends a .ulk extension to the filename (for example, report.ulk).

- **FOR EXCEPTION table-name** Specifies the exception table for rows that encounter link failures for DATALINK values.

Example

The following command reconciles the table DEPT, and writes exceptions to the exception table EXCPTAB, which was created by the user. Information about files that were unlinked during reconciliation is written to report.ulk.
db2 reconcile dept dlreport /u/johnh/report for exception excptab

**REDISTRIBUTE NODEGROUP**

This command redistributes EEE data across nodes in a nodegroup. You can specify the current data distribution, whether uniform or skewed. The redistribution algorithm selects the partitions to move based on the current data distribution.

`REDISTRIBUTE` must be issued from the catalog node. Use `LIST DATABASE DIRECTORY` to determine which node is the catalog node for a database.

**Authorization**

SYSADM, SYSCTRL, or DBADM.

**Syntax**

```bash
>>-REDISTRIBUTE NODEGROUP--nodegroup--+-UNIFORM-------------------+-+------------------------><
| '-USING DISTFILE--distfile--' |
+-'USING TARGETMAP--targetmap----+' |
+ CONTINUE----------------------|
+ ROLLBACK----------------------'
```

**Description**

You use `ALTER NODEGROUP` to add nodes to a nodegroup. This statement permits you to define the containers for the table spaces associated with the nodegroup. `REDISTRIBUTE NODEGROUP` moves data to the new nodes.

`REDISTRIBUTE` performs intermittent COMMITs during processing. The following are the parameter descriptions for `REDISTRIBUTE NODEGROUP`:

- **NODEGROUP nodegroup** Specifies the name of the nodegroup, from the SYSNODEGROUPS catalog table. One redistribution can run against a nodegroup at a time. Tables in IBMCATGROUP and IBMTEMPGROUP cannot be redistributed.

- **UNIFORM** Specifies that data be uniformly distributed across hash partitions (every hash partition is assumed to have the same number of rows), but the same number of hash partitions do not map to each node. After redistribution, all nodes in the nodegroup have approximately the same number of hash partitions.
USING DISTFILE distfile Helps ensure uniform distribution. If the distribution of partitioning key values is skewed, use this option to achieve a uniform redistribution of data across the nodes. The distfile indicates the current distribution of data across the 4096 hash partitions. Use row counts, byte volumes, or any other measure to indicate the amount of data represented by each hash partition. REDISTRIBUTE reads the integer value associated with a partition as the weight of that partition. When a distfile is specified, REDISTRIBUTE generates a target partition map that it uses to redistribute the data across the nodes in the nodegroup as uniformly as possible. After the redistribution, the weight of each node in the nodegroup is approximately the same. (The weight of a node is the sum of the weights of all partitions that map to that node.)

For example, the input distribution file may contain entries as follows:

```
10223
1345
112000
0
100
...
```

In the example, hash partition 2 has a weight of 112,000, and partition 3 (with a weight of 0) has no data mapping to it at all. The distfile should contain 4096 positive integer values in character format. The sum of the values should be less than or equal to 4,294,967,295 (4^32). If the path for distfile is not specified, the current directory is used.

USING TARGETMAP targetmap Specifies a file to be used as the target partition map. Data redistribution is done according to this file. If the path is not specified, the current directory is used.

If a node included in the target map is not in the nodegroup, an error is returned. Issue ALTER NODEGROUP ADD NODE before running REDISTRIBUTE NODEGROUP.

If a node excluded from the target map is in the nodegroup, that node will not be included in the partitioning. Such a node can be dropped by using ALTER NODEGROUP DROP NODE either before or after REDISTRIBUTE NODEGROUP.

CONTINUE continues a previously failed REDISTRIBUTE NODEGROUP operation. If none occurred, an error is returned.

ROLLBACK rolls back a previously failed REDISTRIBUTE NODEGROUP operation. If none occurred, an error is returned.
After the Redistribute Is Complete

When a redistribution operation is done, a message file is written. In UNIX, the file is written to

```
/sqllib/redist
```

using this format:

```
database-name.nodegroup-name.timestamp
```

In Windows, the file is written to

```
\sqllib\redist\ directory
```

using this format:

```
database-name\first-eight-characters-of-the-nodegroup-name\date\time
```

The timestamp is when the command was issued.

All packages with a dependency on a redistributed table are invalid. You should explicitly rebind such packages after redistribute nodegroup has finished. Explicit rebinding eliminates the initial delay in the execution of the first SQL request for the invalid package. The redistribute message file contains a list of all the tables that have undergone redistribution.

Issue RUNSTATS after the redistribute nodegroup operation has completed. Otherwise, the statistics will not be accurate.

Nodegroups containing replicated summary tables or tables that are replication sources for Data Propagator (defined with DATA CAPTURE CHANGES) cannot be redistributed. Redistribution is not allowed if there are user temporary table spaces with existing declared temporary tables in the nodegroup.

---

**REFRESH LDAP**

This command refreshes the cache on a local machine with updated information when the information in LDAP has changed.

**Authorization**

None.
**Syntax**

```
>>-REFRESH LDAP----+CLI_CFG--+--------------------------------<<
   +DB DIR---+
   '-NODE DIR-'  
```

**Description**

The following are the parameter descriptions for REFRESH LDAP:

- **CLI_CFG** Specifies the CLI (DB2 CLI/ODBC) configuration to refresh (Windows only)
- **DB DIR** Refreshes the database directory
- **NODE DIR** Refreshes the node directory

If the object in LDAP is removed or changed during refresh, so is the corresponding LDAP entry on the local machine. If the db2cli.ini file is manually updated, the REFRESH LDAP CLI CFG command must be run to update the cache for the current user.

The REFRESH LDAP DB DIR and REFRESH LDAP NODE DIR commands remove the LDAP database or node entries found in the local database or node directories. The database or node entries will be added to the local database or node directories again when the user connects to a database or attaches to an instance found in LDAP, and DB2LDAPCACHE is not set to NO.

**REGISTER**

This command registers the DB2 server in the network directory server.

**Authorization**

None.

**Syntax**

```
>>-REGISTER------------------------+ NWBINDERY path |-----><<
   '-DB2 SERVER--IN--'
   '-LDAP path |------'

NWBINDERY path

|---NWBINDERY--USER--username--+---------------------+----------|
   '-PASSWORD--password--'
```

```
DB2 Command Line Processor Commands

LDAP path

|--- LDAP AS--nodename-------------------------------------------|

>-----PROTOCOL--TCP/IP--------------------------------------|
  | 'HOSTNAME--hostname' 'SVCENAME--svce_name' 'SECURITY--SOCKS--' |
  | 'NAME--name' |
  |-| APPN path |---------------------------------------------|
  |-| TP/IPX----------------------------------|
  | 'IPX_ADDRESS--ipxaddr--' |
  | 'IPX_ADDRESS--ipxaddr--' |

>-----REMOTE--computer--'INSTANCE--instance--'

>-------------------------------|-------------------------------|
  | 'REMOTE--computer--' 'INSTANCE--instance--' |

>-----APPN---NETWORK--net_id---PARTNERLU--partner_lu-------->

>-----MODE--mode----+------------------+-------------------------|
  | 'TPNAME--tp_name--' |

>-----SAME----|
  | 'TPNAME--tp_name--' |

>-----|------------------------+---+--------------------------+--------|
  | 'CHANGEPWDLU--change_password_lu--' |

APPN path

|--| APPN---NETWORK--net_id---PARTNERLU--partner_lu--------

>-----NODE--node------------------------------------------------|
  | 'TPNAME--tp_name--' |

>-----TPNAME--tp_name--'

>-----SECURITY--NONE------+ 'LANADDRESS--lan_address--' |
  | 'SAME' |

>-----PROGRAM--'

>------CHANGEPWDLU--change_password_lu--'
Description

The following are the parameter descriptions for REGISTER:

- **IN** Specifies the network directory server on which to register the DB2 server. Valid values are NWBINDERY for a NetWare bindery, and LDAP for an LDAP directory server.

- **USER username for NWBINDERY** Specifies the user ID to log on to the network server. For LDAP, this is the user’s LDAP distinguished name (DN). The LDAP user DN must have sufficient authority to create and update the object in the LDAP directory. The user name is optional when registering in LDAP. If the user’s LDAP DN is not specified, the credentials of the current logon user will be used.

- **PASSWORD password** Specifies the account password.

- **AS nodename** Indicates a short name to represent the DB2 server in LDAP. A node entry is cataloged in LDAP using this node name. The client can attach to the server using this node name.

- **PROTOCOL** Specifies the protocol used by LDAP clients. The DB2 server must be registered once per protocol, which can be either TCPIP, NetBIOS, APPN, IPXSPX or NPIPE.

- **HOSTNAME hostname** Specifies the TCP/IP hostname (or IP address).

- **SVCENAME svcename** Specifies the TCP/IP service name or port number.

- **SECURITY SOCKS** Uses TCP/IP socket security.

- **NNAME nname** Specifies the NetBIOS workstation name.

- **NETWORK net_id** Specifies the APPN network ID.

- **PARTNERLU partner_lu** Specifies the APPN partner LU name for the DB2 server machine.

- **MODE mode** Specifies the APPN mode.

- **TPNAME tpname** Specifies the APPN transaction program name. This defaults to DB2DRDA.

- **SECURITY** Specifies the APPN security level. This can be one of NONE, PROGRAM, or SAME.

- **LANADDRESS lan_address** Specifies the APPN network adaptor address.

- **CHGPWDLU change_password_lu** Specifies the partner LU when using APPC to change the password for a host database server.

- **IPX_ADDRESS ipxaddr** Specifies the complete IPX address.

- **REMOTE computer** Indicates the computer name of the machine on which the DB2 server resides. Specify this parameter only if you are registering a remote DB2 server in LDAP.
DB2 Command Line Processor Commands

- **INSTANCE instance** Specifies the instance name of the DB2 server. The instance name must be specified for a remote instance (that is, when a value for the REMOTE parameter has been specified).
- **NODETYPE** Specifies the node type for the database server. This can be one of the following:
  - SERVER DB2 Enterprise Edition (This is the default.)
  - EEE DB2 Enterprise—Extended Edition
  - DCS DB2 Connect
- **OSTYPE ostype** Specifies one of AIX, Win95, NT, OS/2, HPUX, Sun, MVS, OS/400, VM, VSE, SNI, SCO, SGI, Linux, or PTX.
- **WITH “comments”** Allows you to enter any comment that helps to describe the server registered in the network directory.

**Example**

A DB2 server supports both NetBIOS and TCP/IP. In this case, you will need to run REGISTER twice:

```
  db2 register db2 server in ldap as tcpnode protocol tcpip
  db2 register db2 server in ldap as nbnode protocol netbios
```

**REORGANIZE TABLE**

This command reorganizes a table by reconstructing the rows to eliminate fragmented data, use empty space, and order data on disk. This affects all nodes in the nodegroup on EEE.

**Authorization**

One of the following:

- SYSADM
- DBADM
- SYSCTRL
- SYSMAINT
- CONTROL privilege on the table
Syntax

```sql
>>>-REORG TABLE--table-name--+-INDEX--index-name--

'-USE--tablespace-name--'
```

Description

Tables that have been modified so many times that data is fragmented and access performance is noticeably slow are candidates for reorganization. Use REORGCHK to determine whether a table needs reorganizing. Be sure to complete all database operations and release all locks before reorganizing a table. After reorganizing a table, use RUNSTATS to update the table statistics, and REBIND the packages that use the table.

If the table is partitioned onto several nodes, and the table reorganization fails on any of the affected nodes, only the failing nodes have the table reorganization rolled back. If the reorganization is not successful, do not delete temporary files. The database manager uses these files to recover the database.

Specify the most frequently used index in the REORG. If the name of an index is not specified, and if a clustering index exists, the data will be ordered according to the clustering index.

The PCTFREE value of a table determines the amount of free space reserved on each page. If the value has not been set, REORG uses as much space as possible on each page. REORGANIZE TABLE cannot be used on views, declared temporary tables, or a DMS table while an online backup is running of a table space in which the table resides.

The following parameters can be used with REORGANIZE TABLE:

- **TABLE table-name** Indicates the table to reorganize. The table can be in a local or remote database. The fully qualified name or alias in the form schema.table-name must be used. The schema is the user name under which the table was created. For typed tables, use the hierarchy’s root table.

- **INDEX index-name** Specifies the index to use when reorganizing the table. The fully qualified name in the form schema.index-name must be used. The schema is the user name under which the index was created. The database manager uses the index to physically reorder the records in the table, and if an index is not provided, rows are reorganized without regard to order, or they are ordered based on any clustered index on the table.

- **USE tablespace-name** Specifies the name of a system temporary table space where the database manager can temporarily store the table being reconstructed. If a table space name is not entered, the database manager stores a working copy of the table in the same table space(s) in which the reorganized
The page size of any system temporary table space explicitly specified by the user must match the page size of the table space(s) in which the table data (including any LONG or LOB column data) resides.

**Example**

To reorganize the EMPLOYEE table using the system temporary table space TEMPSPACE1 as a work area, enter the following command:

```sql
db2 reorg table homer.employee using temppacel
```

This will reclaim space, but because no index is specified, the order will be arbitrary unless EMPLOYEE has a clustering index.

**REORGCHK**

REORGCHK calculates statistics on the database to determine whether tables need to be reorganized. REORGCHK can be run from any node in an EEE system. It can also be used to update table and index statistics in the catalogs.

**Authorization**

One of the following:

- SYSADM
- DBADM
- CONTROL privilege on the table

**Syntax**

```
>>-REORGCHK--------------------------------------------->
| .-UPDATE--. | '-++CURRENT+++STATISTICS--'

>---------------------------------------------<
| .-USER------. | '-ON TABLE+++SYSTEM------'
  +--ALL-------+
  '|-table-name-'```
Description

REORGCHK calculates statistics obtained from six different formulas to determine whether performance has deteriorated or can be improved by reorganizing a table. The formulas are displayed when you run reorgchk (formulas 1 through 6 below):

The terms for the table statistics formulas (Formulas 1 through 3) have the following meaning:

- **CARD** (cardinality) Number of rows in base table.
- **OV** Number of overflow rows.
- **NP** Number of pages that contain data.
- **FP (FPAGES)** Total number of pages.
- **TSIZE** Table size in bytes. This is calculated as the product of the number of rows in the table (CARD) and the average row length. The average row length is computed as the sum of the average column lengths (AVGCOLLEN in SYSCOLUMNS) plus ten bytes of row overhead. For long fields and LOBs, only the approximate length of the descriptor is used. The actual long field or LOB data is not counted in TSIZE.
- **TABLEPAGESIZE** Page size of the table space in which the table data resides.
- **REORG** Each hyphen (-) displayed in this column in the output for REORGCHK indicates that the calculated results were within the set bounds of the corresponding formula, and each asterisk (*) indicates that the calculated results exceeded the set bounds of its corresponding formula.

The left side of the column corresponds to F1 (Formula 1).

The middle of the column corresponds to F2 (Formula 2).

The right side of the column corresponds to F3 (Formula 3).

Table reorganization is suggested when the results of the calculations exceed the bounds set by the formula.

The terms for the index statistics (F4 through F6) mean the following:

- **CARD** Number of rows in base table.
- **LEAF** Total number of index leafs (pages).
- **LVLS (LEVELS)** Number of index levels.
- **ISIZE** Index size, calculated from the average column length of all columns participating in the index.
- **KEYS (FULLKEYCARD)** Number of unique index entries.
- **INDEXPAGESIZE** Page size of the table space in which the table indexes reside.
The percentage of each index page to leave as free space, assigned when defining the index. Values can range from 0 to 99. The default is 10.

REORGCHK uses formulas F1 through F3 to analyze the physical location of rows and the size of the table:

Formula F1:

\[ 100 \times \frac{\text{OVERFLOW}}{\text{CARD}} < 5 \]

The total number of overflow rows in the table should be less than 5 percent of the total number of rows. Overflow rows can be created when rows are updated and the new rows contain more bytes than the old ones (VARCHAR fields), or when columns are added to existing tables.

Formula F2:

\[ 100 \times \frac{\text{TSIZE}}{\left( (\text{FPAGES}-1) \times (\text{TABLEPAGESIZE}-76) \right)} > 70 \]

The table size in bytes (TSIZE) should be more than 70 percent of the total space allocated for the table. (There should be less than 30 percent free space.) The total space allocated for the table depends on the page size of the table space in which the table resides (minus an overhead of 76 bytes). Because the last page allocated is not usually filled, 1 is subtracted from FPAGES.

Formula F3:

\[ 100 \times \frac{\text{NPAGES}}{\text{FPAGES}} > 80 \]

The number of pages that contain no rows at all should be less than 20 percent of the total number of pages. (Pages can become empty after rows are deleted.)

REORGCHK uses formulas F4 through F6 to analyze the relationship of the indexes to the table data:

Formula F4:

\[ \text{CLUSTERRATIO} \text{ or normalized CLUSTERFACTOR} > 80 \]

The clustering ratio of an index should be greater than 80 percent. When multiple indexes are defined on one table, some of these indexes have a low cluster ratio. (The index sequence is not the same as the table sequence.) This cannot be avoided. Be sure to specify the most important index when reorganizing the table. The cluster ratio is rarely optimal for indexes that contain many duplicate keys and many entries.
**DB2: The Complete Reference**

Formula F5:

\[
100 \times (\text{KEYS} \times (\text{ISIZE}+8) + (\text{CARD-KEYS}) \times 4) \div (\text{NLEAF} \times \text{INDEXPAGESIZE}) > 50
\]

Less than 50 percent of the space reserved for index entries should be empty (only checked when NLEAF>1).

Formula F6:

\[
\frac{(100-\text{PCTFREE}) \times (\text{INDEXPAGESIZE}-96) / (\text{ISIZE}+12)^n \times (\text{INDEXPAGESIZE}-96)}{(\text{KEYS} \times (\text{ISIZE}+8) + (\text{CARD-KEYS}) \times 4)} < 100
\]

The actual number of index entries should be more than 90 percent (or 100-PCTFREE) of the number of entries an NLEVELS-1 index tree can handle (checked only if NLEVELS>1).

The parameters for REORGCHK are as follows:

- **UPDATE STATISTICS** Calls RUNSTATS to update table statistics, and then uses the updated statistics to determine whether table reorganization is required.
- **CURRENT STATISTICS** Uses current table statistics to determine if table reorganization is required.
- **ON TABLE** Identifies which tables you want to check:
  - USER checks tables that are owned by the connected user.
  - SYSTEM checks the catalog tables.
  - ALL checks all user and system tables.
  - table-name checks this particular table. The fully qualified name or alias in the form schema.table-name must be used. Use SYSIBM for system catalog tables. For typed tables, use the hierarchy’s root table.

**Example**

db2 reorgchk update statistics on table system

**RESET ADMIN CONFIGURATION**

This command resets the parameters for the database manager configuration for the DB2 Administration Server to the system defaults. The following configuration parameters are reset:

- **AGENT_STACK_SZ**
RESET DATABASE CONFIGURATION

This command resets the configuration of a specific database to system defaults. In EEE, it affects the node where you run it.

Authorization

SYSADM, SYSCTRL, or SYSMAINT.
Syntax

```
GR-RESET------+-DATABASE---+---+-CONFIGURATION-+---------------->
  '-DB------'  '+CONFIG-----'
  '-CFG------'
>-----FOR--database-alias--------------------------------------><
```

Description

The FOR database-alias parameter specifies the alias of the database. Its configuration is reset to system defaults.

To see the database configuration parameters, use `GET DATABASE CONFIGURATION`. To change a parameter, use `UPDATE DATABASE CONFIGURATION`.

RESET DATABASE MANAGER CONFIGURATION

This command resets the parameters in the database manager configuration to system defaults. The values are reset by node type, which is always a server with remote clients.

Authorization

SYSADM.

Syntax

```
GR-RESET------+-DATABASE MANAGER---+-CONFIGURATION-+---------------->
  '+DB MANAGER-----'  '+CONFIG-----'
  '-DBMANAGER-----'  '-CFG------'
```

Description

To see the database manager configuration, use `GET DATABASE MANAGER CONFIGURATION`. To change a parameter, use `UPDATE DATABASE MANAGER CONFIGURATION`.

RESET MONITOR

RESET MONITOR resets the internal database system monitor data areas of a specified database, or of all active databases, to zero. The internal database system monitor data
areas include the data areas for all applications connected to the database, as well as the data areas for the database itself.

**Authorization**
SYSADM, SYSCtrl, or SYMAINT.

**Syntax**

```
>>-RESET MONITOR------------------------------------------------>
>-----+-ALL--+-----+---------------------------------+-------->
     | '-DCS-'                                 |
'FOR--+-----+--+-DATABASE-+---database-alias--'
  '-DCS-'  '-DB------'
>-----+-------------------+------------------------------------><
+AT NODE--nodenum--+
'GLOBAL----------'
```

**Description**
The following are the parameter descriptions for RESET MONITOR:
- **ALL** Resets internal counters for all databases.
- **FOR DATABASE database-alias** Resets counters for this database.
- **DCS** Resets counters for databases monitored by DB2 Connect. This can be either:
  - All DCS databases
  - A specific DCS database
- **AT NODE nodenum** Specifies, in EEE, the node for which switches are reset.
- **GLOBAL** Resets counters for all nodes in an EEE database.

**RESTART DATABASE**
RESTART DATABASE restarts a database that has stopped in an inconsistent state and runs crash recovery (indicated in the db2diag.log file). In EEE, this affects only the node where you run RESTART DATABASE.

**Authorization**
None.
Syntax

```
>>>-RESTART-----+--DATABASE-+--database-alias------------------>
   '-DB-------'

>>>-+----------------------------------------------------------+>
   '-USER--username--+------------------+--'
   '-USING--password--'

>>>-+----------------------------------------------------------+>
   '-DROP PENDING TABLESPACES--(-----tablespace-name---++)--'
```

Description

Run RESTART DATABASE if connecting to a database returns an error message saying that the database must be restarted. This occurs only if the previous session with this database terminated abnormally (due to a power failure, for example).

When RESTART DATABASE finishes, an SQL warning is issued if any indoubt transactions exist. In this case, the database is still usable, but if the indoubt transactions are not resolved before the last connection to the database is dropped, another RESTART DATABASE must be issued to restart the database. Use LIST INDOUBT TRANSACTIONS to see any indoubt transactions.

If the database is restarted only on a single node in EEE, a message may be returned on a subsequent database query indicating that the database needs to be restarted. This occurs because the database partition on a node on which the query depends must also be restarted. Restarting the database on all nodes solves the problem.

These are the parameters for RESTART DATABASE:

- **DATABASE database-alias** Specifies the database to be restarted.
- **USER username** Specifies the user name under which the database is restarted.
- **USING password** Specifies the password used to authenticate username. If omitted, you are prompted.
- **DROP PENDING TABLESPACES tablespace-name** Specifies that DB2 will restart even if table space container problems are encountered. If a problem occurs with a container for a specified table space during the restart process, the corresponding table space will not be available (it will be in drop pending state) after the restart operation. In the case of circular logging, this could cause restart to fail. Problem table spaces are listed in db2diag.log if a restart database
operation fails because of container problems. If there is only one system temporary table space in the database, and it is in drop pending state, a new system temporary table space is created immediately after the restart.

### RESTORE DATABASE

This command rebuilds a damaged or corrupted database that has been saved using DB2’s BACKUP command. The restored database is in the same state it was in when the backup copy was made. This utility can also move a database (in addition to being able to restore to a new database).

Backups made with DB2 version 5 and version 6 can be restored on DB2 version 7. Database migration happens automatically as part of such a restore.

#### Authorization

To restore an existing database: SYSADM, SYSCTRL, or SYSMAINT
To use RESTORE to create a new database: SYSADM or SYSCTRL

#### Syntax

```
>>-RESTORE----+-DATABASE-+--source-database-alias--------------->
   '-DB-------'
>------+-| restore-options |+--------------------------><
   '+CONTINUE-------------+
   '-ABORT-------------'

restore-options

|+---------------------------------------+------------------->
   '-USER--username--+------------------+--'
   '-USING--password--'
>

>-----+--------------------------------------------------------+>

+-TABLESPACE--+-ONLINE--

+-TABLESPACE--(-----tablespace-name---+---)--+---------+-+
|                                            '-ONLINE--' |

'-HISTORY FILE--+-ONLINE--'
```
'-'ONLINE--'

>'-INCREMENTAL--'
  +AUTO------
  +AUTOMATIC--
  '-ABORT-----'

>'-USE TSM--'
  | '-OPEN--num-sessions--SESSIONS--'
  | '.-,-----------------.                        
  | V                  |                              
  +FROM----+-directory-+--
      | '-device----'                              
      | '-LOAD--shared-library--'
      | '-OPEN--num-sessions--SESSIONS--'

>'-TAKEN AT--date-time--'

>'-TO--target-directory--'

>'-INTO--target-database-alias--'

>'-NEWLOGPATH--directory--'

>'-WITH--num-buffers--BUFFERS--'

>'-BUFFER--buffer-size--'

>'-DLREPORT--filename--'

>'-REPLACE EXISTING--'

>'-REDIRECT--'

>'-PARALLELISM--n--'

>'-WITHOUT ROLLING FORWARD--'

>'-WITHOUT DATALINK--'

>'-WITHOUT PROMPTING--'
Description

Any RESTORE DATABASE command of the form db2 restore db <name> will perform a full database restore, regardless of whether the image restored is a database image or a table space image. Any RESTORE DATABASE command of the form db2 restore db <name> tablespace will perform a table space restore of the table spaces found in the image. Any RESTORE DATABASE command with a list of table spaces will perform a restore of the table spaces explicitly listed.

If, at the time of the backup operation, the database was enabled for roll-forward recovery, the database can be brought to any state prior to the occurrence of damage or corruption by invoking the roll-forward utility after successful completion of a restore operation. Roll forward requires the logs of all database activity since the backup image was taken. See BACKUP for an explanation of how incremental and delta backup images must be restored. A table space backup can also be restored.

Restore can move DB2 Solaris backup images to HP-UX, and DB2 HP-UX images to Solaris. To move DB2 data to other platforms, see DB2MOVE.

In EEE, a backup must be restored on each node independently.

To restore to a new remote database, attach to the instance where the new database will reside. Then create the new database, specifying the code page and the territory of the server.

The following parameters can be used with the RESTORE DATABASE command:

- **DATABASE source-database-alias** Specifies an alias of the source database for the backup image.
- **CONTINUE** Indicates that the containers have been redefined and that the final step in the redirected restore should be performed.
- **ABORT** Stops a redirected restore. This is useful when an error has occurred that would require repetition of steps. After ABORT is issued, each step of a redirected restore must be repeated, including RESTORE DATABASE with the REDIRECT option. ABORT also terminates an incremental restore operation before completion.
- **USER username** Specifies the user name under which the database is restored.
- **USING password** Specifies the password that authenticates the user name. If omitted, the user is prompted to enter a password.
- **TABLESPACE tablespace-name** Provides a list of names specifying table spaces to restore.
- **ONLINE** Allows the backup to be restored while the database is active—this is applicable only when doing a table space–level restore. Other agents can connect to the database while the backup is being restored, because data in other table spaces will be available while the list of table spaces specified with ONLINE is restored.
HISTORY FILE  Restores only the history file from the backup image.

INCREMENTAL  Results in a manual cumulative restore of the database.
The user will issue each of the restore commands.

AUTOMATIC/AUTO  Specifies that the database will examine the history
file and generate an automatic incremental restore.

USE TSM  Specifies that the database will be restored from TSM-
managed output.

OPEN num-sessions SESSIONS  Specifies the number of I/O sessions to be
used with TSM or the vendor product. More sessions can make the restore
faster through parallelism.

FROM directory/device  Specifies the directory or device on which the backup
images reside. If USE TSM, FROM, and LOAD are omitted, the default is the
current directory. On OS/2 or Windows, the specified directory must not be a
DB2-generated directory. For example, given the following commands, DB2
generates subdirectories under the c:\backup directory that should be ignored:

db2 backup database sample to c:\backup
db2 restore database sample from c:\backup

To specify precisely which backup image to restore, use the TAKEN AT parameter.
Because there can be many backup images in the same path, TAKEN AT uses a
timestamp to uniquely identify each one.

If several items are specified and the last item is a tape device, you are prompted
for another tape. Valid response options are as follows:

- c  Continue using the device that generated the warning message (for
  example, when a new tape has been mounted).

- d  Device terminate. Stop using only the device that generated the warning
  message (for example, when there are no more tapes).

- t  Terminate the restore.

LOAD shared-library  Specifies the name of the shared library (DLL on OS/2
or Windows) containing the vendor backup and restore I/O functions—used
with non-IBM backup software products like Veritas and Legato. It may contain
the full path or, if the full path is not given, it defaults to the path where the
user exit programs reside.

TAKEN AT date-time  Specifies the timestamp of the database backup image
in the form yyyymmddhhmms. The timestamp is displayed after successful
completion of a backup operation, and it is part of the pathname for the backup
image. A partial timestamp can also be specified. For example, if two different
backup images with timestamps 19971001010101 and 19971002010101 exist,
specifying 19971002 will select the image with timestamp 19971002010101. If this parameter is not specified, there must be only one backup image on the source media.

- **TO target-directory**  Specifies the target database directory. “TO”  is ignored if an existing database is restored. Specify only the drive letter on Windows and OS/2.

- **INTO target-database-alias**  Specifies an alias of the target database. If the target database does not exist, RESTORE creates it.

- **NEWLOGPATH directory**  Specifies a fully qualified directory used by the database after restore for active log files. This parameter has the same function as the database configuration parameter newlogpath, except that its effect is limited to the RESTORE command in which it is specified. This parameter can be used when the log path in the backup image is not suitable to be used after the restore, such as when the path is no longer valid or is currently used by a different database. If possible, put the logs on a different device than the data to increase performance.

- **WITH num-buffers BUFFERS**  Specifies the number of buffers used; this defaults to 2. A larger number of buffers may be used to improve performance when multiple sources are read or when the parallelism parameter is increased.

- **BUFFER buffer-size**  Specifies the size, in pages, of the buffer used for the restore operation. The minimum value for this parameter is 8 pages; the default is 1024 pages. If a buffer size of zero is specified, the value of the database manager configuration parameter Restbufsz will be used as the buffer allocation size. The restore buffer size must be a positive integer multiple of the backup buffer size specified during the backup operation. If an incorrect buffer size is specified, the buffers allocated will be the smallest acceptable size.

- **DLREPORT filename**  Reports files unlinked during restore due to fast reconcile. This is used only for tables with DATALINKs columns.

- **REPLACE EXISTING**  Specifies that restore will replace the existing database when the target exists under the same alias. Use this in scripts containing the RESTORE DATABASE command, because the CLP will not prompt to verify deletion of the existing database. If the WITHOUT PROMPTING parameter is specified, it is not necessary to specify REPLACE EXISTING, but if there is no prompting, the command will fail if events occur that normally require user intervention.

- **REDIRECT**  Redirects the restore. To complete a redirected restore, follow REDIRECT with one or more SET TABLESPACE CONTAINERS commands, and then issue RESTORE DATABASE with the CONTINUE option. All commands associated with a single redirected restore must be executed from the same window or CLP session.
DB2: The Complete Reference

- **WITHOUT ROLLING FORWARD** Does not place the database in roll-forward pending state after it has been successfully restored. If, following a successful restore, the database is in roll-forward pending state, ROLLFORWARD DATABASE must be executed before the database can be used.

- **WITHOUT DATALINK** Specifies that tables with DATALINK columns are placed in DataLink_Reconcile_Pending (DRP) state, and no reconciliation of linked files is performed.

- **PARALLELISM** Specifies the number of buffer manipulators spawned during the restore process. The default is 1.

- **WITHOUT PROMPTING** Specifies that restore will run unattended, and any actions that normally require user intervention instead return error messages. When using a tape, you are prompted at the end of the device even if this option is specified.

**Example**

The following is a backup strategy and commands for a recoverable database. It includes a weekly full database backup operation, a daily noncumulative (delta) backup operation, and a mid-week cumulative (incremental) backup operation:

```
(Sun) backup db kdr use tsm
(Mon) backup db kdr online incremental delta use tsm
(Tue) backup db kdr online incremental delta use tsm
(Wed) backup db kdr online incremental use tsm
(Thu) backup db kdr online incremental delta use tsm
(Fri) backup db kdr online incremental delta use tsm
(Sat) backup db kdr online incremental use tsm
```

For an automatic database restore of the images created on Friday, issue this command:

```
restore db kdr incremental automatic taken at (Thu)
```

For a manual database restore of the images created on Friday, issue these commands:

```
restore db kdr incremental taken at (Thu)
restore db kdr incremental taken at (Sun)
restore db kdr incremental taken at (Wed)
restore db kdr incremental taken at (Thu)
```
**REWIND TAPE**

DB2 for Windows supports backup and restore to streaming tape devices. Use this command for tape rewinding. Operating system commands should be used for tape rewinding on UNIX and Linux.

**Authorization**
None.

**Syntax**

```
>>>REWIND TAPE------+-------------+------------------------------><
             '-ON--device--'
```

**Description**
The ON device parameter specifies a valid tape device name. The default is `\\TAPE0`.

---

**ROLLFORWARD DATABASE**

This command recovers a database by applying transactions recorded in the database log files.

**Authorization**
SYSADM, SYSCTRL, or SYMAINT.

**Syntax**

```
>>>ROLLFORWARD----+-DATABASE-+---database-alias----------------->
         '-DB-------'
         >-----+---------------------------------------+----------------->
                  '-USER--username--+------------------+--'
                  '-USING--password--'
         >-----+----------------------------------------------------------------+>
               |-TO--+-isotime--+--------------+-----------+---+--------------+-+
               |     |          '-ON ALL NODES-'           |   +-AND COMPLETE-+ |
On Node clause

| '---ON--+| Node List clause | -----+---------------------+
| 'ALL NODES--+---------------------+
| '---EXCEPT--+| Node List clause | ---+

Node List clause

| '---+-NODE---+--(-------node-number1--+-------------------+--+>
| 'NODES--'                        '-TO--node-number2--'

Log Overflow clause

.'---+-----------------------------------------------------+>

On Node clause

On Node clause

On Node clause

On Node clause

On Node clause

On Node clause
Description

Run roll forward after a database or a table space backup image has been restored, or if any table spaces have been taken offline by the database due to a media error. The database must be recoverable (either logretain, userexit, or both of these database configuration parameters must be enabled) before the database can be rolled forward. In an EEE database, run ROLLFORWARD from the catalog node. A database or table space roll forward to a point in time affects all nodes listed in the db2nodes.cfg file. A database or table space rolled forward to the end of logs affects the nodes specified. If no nodes are specified, all nodes listed in the db2nodes.cfg file roll forward.

If you are restoring from an image created during an online backup operation, the specified point in time for the roll forward must be later than the time at which the online backup operation completed. If roll forward is stopped before passing this point, the database is left in roll-forward pending state. If a table space was being rolled forward, it is left in roll-forward in progress state.

If one or more table spaces is rolled forward to a point in time, the roll forward must continue at least to the minimum recovery time, which is the last update to the system catalogs for this table space or its tables. The minimum recovery time (in Coordinated Universal Time) for a table space can be retrieved by using LIST TABLESPACES.

Rolling databases forward may require load recovery using tape devices. If you are prompted for another tape, respond by using these options:

- **c** Continue using the device that generated the warning message (for example, when a new tape has been mounted).
- **d** Device terminate. Stop using the device that generated the warning message (for example, when there are no more tapes).
- **t** Terminate all devices.

The parameters for the ROLLFORWARD DATABASE command are as follows:

- **DATABASE database-alias** Specifies the alias of the database to roll forward.
- **USER username** Specifies the user name under which the database will roll forward.
- **USING password** Specifies the password to authenticate the user name. If omitted, you are prompted for the password.
- **TO isotime** Specifies the point in time to which all committed transactions are rolled forward (including the transaction committed precisely at that time, as
well as all transactions committed previously). The value is a timestamp, a seven-part character string identifying a combined date and time:

```
yyyy-mm-dd-hh.mm.ss.nnnnnn
(year, month, day, hour, minutes, seconds, microseconds)
```

The time must be Coordinated Universal Time (UTC), formerly known as Greenwich Mean Time (GMT). UTC avoids having the same timestamp associated with different logs (a danger if a daylight saving time change occurs during the roll forward). Note that the timestamp of a backup image is based on the local time when the backup started. The CURRENT TIMEZONE special register specifies the difference between UTC and local time at the application server. For example, during daylight saving time at EST (Eastern Standard Time):

```
values current timezone
1
--------
-40000.
```

The difference is a time duration (a decimal number in which the first 1–2 digits represent the number of hours, the next two digits represent the number of minutes, and the last two digits represent the number of seconds). Subtracting CURRENT TIMEZONE from a local time converts that local time to UTC:

```
db2 values current time - current timezone
1
--------
23:25:52
```

■ **END OF LOGS** Specifies that all committed transactions from all online archive log files listed in the database configuration parameter logpath are applied.

■ **ALL NODES** Specifies that all transactions are rolled forward on all nodes in the db2nodes.cfg file. This is used only in EEE, and it is the default if a node clause is not specified.

■ **EXCEPT** Specifies that transactions are rolled forward on all nodes in the db2nodes.cfg file, except those specified with EXCEPT.

■ **ON NODE / ON NODES** Lists the node or nodes to roll forward.

■ **node-number1** Identifies a node number in the node list.
node-number2 Identifies the final node number so that all nodes from node-number1 up to and including node-number2 are included in the node list.

COMPLETE / STOP Stops the roll forward of logs, and completes roll forward by rolling back incomplete transactions and turning off the roll-forward pending state of the database. This allows access to the database or table spaces that are rolled forward. These keywords are equivalent. The keyword AND permits specification of multiple operations:

```
db2 rollforward db sample to end of logs and complete
```

When rolling table spaces forward to a point in time, the table spaces are placed in backup pending state.

CANCEL Cancels the roll forward and puts the database or one or more table spaces on all nodes on which forward recovery started in restore pending state:

- If a database roll forward is not in progress (when the database is in roll-forward pending state), CANCEL puts the database in restore pending state.
- If a table space roll forward is not in progress (the table spaces are in roll-forward pending state), specify a table space list. All table spaces in the list are put in restore pending state.
- If a table space roll forward is in progress (at least one table space is in roll-forward in progress state), all table spaces that are in roll-forward in progress state are put in restore pending state. If you specify a table space list, include all table spaces that are in roll-forward in progress state.
- If rolling forward to a point in time, any table space name that is passed in is ignored, and all table spaces that are in roll-forward in progress state are put in restore pending state.
- If rolling forward to the end of the logs with a table space list, only the table spaces listed are put in restore pending state.

CANCEL cannot be used to cancel a roll forward that is actually running. It can only be used to cancel a roll forward that is in progress but not actually running at the time. A roll forward can be in progress but not running if any of the following occurred:

- It terminated abnormally.
- STOP was not specified.
- An error caused it to fail. Some errors, such as rolling forward through a nonrecoverable load, can put a table space into restore pending state.
Use CANCEL only if the roll forward cannot finish because some table spaces are in roll-forward pending or restore pending state. When in doubt, use the LIST TABLESPACES command to identify the table spaces that are in roll-forward in progress state or roll-forward pending state.

- **QUERY STATUS** Lists the log files that the database manager has rolled forward, the next archive file required, and the timestamp (in UTC) of the last committed transaction since the roll forward started. In EEE, this status is returned for each node. QUERY STATUS is the default if the TO, STOP, COMPLETE, and CANCEL clauses are omitted. The following is an example of the information returned:

  ```
  db2 rollforward database sample query status
  Rollforward Status
  Input database alias                   = sample
  Number of nodes have returned status   = 1
  Node number                            = 0
  Rollforward status                     = not pending
  Next log file to be read               =
  Log files processed                    = -
  Last committed transaction             =
  2001-07-01-11.56.25.000000
  ```

The QUERY STATUS fields are as follows:

- Node number 0 (this is not an EEE database, so it has only one node)
- Roll forward status is one of:
  - database roll-forward pending
  - database roll-forward in progress
  - database rollforward processing STOP
  - table space roll-forward pending
  - table space roll-forward in progress
  - table space roll-forward processing STOP
  - not pending

  Next log file to be read identifies the next required log file. In EEE, use this information if the roll-forward utility fails with a return code indicating a missing log file or a log information mismatch.

  Log files processed identifies the names of processed log files no longer needed for recovery. They can be removed from the directory. If the oldest uncommitted
transaction starts in log file \( x \), the range of obsolete log files will not include \( x \); the range ends at the log file preceding \( x \).

Last committed transaction provides an ISO timestamp (yyyymm-dd-hh.mm.ss) marking the last transaction committed after the completion of roll-forward recovery. The timestamp applies to the database. For table space roll forward, it is the timestamp of the last transaction committed to the database.

If TO, STOP, or COMPLETE are specified, this information will be displayed if the command ran successfully. If individual table spaces are identified with the QUERY STATUS request, they are ignored; the status request will not be limited only to the table spaces.

- **TABLESPACE** Specifies options for table space–level roll forward.

  Tablespace-name is mandatory for table space–level roll forward to a point in time. It also allows a subset of table spaces to be specified for a roll forward to the end of logs. In EEE, each table space in the list does not have to exist at each node that is rolling forward. If it does exist at the node, it must be in the correct state.

- **ONLINE** Allows table space–level roll-forward recovery to be done online. Other agents are allowed to connect during the roll forward.

- **OVERFLOW LOG PATH log-directory** Specifies an alternate log path to search for archived logs during recovery. Use this parameter if log files were moved to a location other than that specified by the logpath database configuration parameter. In EEE, this is the fully qualified default overflow log path for all nodes. If the roll forward utility cannot find the next required log, the log name is returned in the SQLCA (SQL Communications Area), and roll-forward recovery stops. You can use the –a option from the CLP to display the SQLCA:

  ```
  db2 -a rollforward database sample...
  ```

  If no more logs are available, use the STOP option to terminate roll forward. Incomplete transactions are rolled back to ensure the consistency of the database or table space.

- **log-directory ON NODE** Allows a different log path to override the default overflow log path in EEE.

- **RECOVER DROPPED TABLE drop-table-id** Recovers a dropped table during the roll forward. The table ID can be obtained by using LIST HISTORY. This is the only data definition language (DDL) that can be manipulated granularly within a Rollforward Point in Time. DDL includes CREATE, ALTER, and DROP.
TO export-directory Specifies a directory for files containing the table data. The directory must be accessible to all nodes. These files are used by DB2 to recover the dropped table after the roll forward finishes.

Example: Roll Forward Sample Database
To roll forward to the end of logs and complete, run these two commands:

```
db2 rollforward db sample to end of logs
db2 rollforward db sample complete
```

They can be combined into one command using the AND keyword, but then you lose the opportunity to verify the roll forward before completing it.

By using two commands, you have a chance to find any logs reported by DB2 as missing after the first step. These logs might be archived on TSM or moved to an unexpected directory if multiple DBAs work on the same system and log storage procedures are ignored or incomplete. This also allows you to deal with a damaged log and replace it with a good copy (which might also be on TSM or in a duplicate directory). In such cases, an undamaged backup copy of that log can be used to continue the roll forward through more logs.

Example: Roll Forward to End of Logs
This example rolls forward to the end of the logs (two table spaces have been restored):

```
db2 rollforward db sample to end of logs and stop
```

Neither AND STOP nor AND COMPLETE is needed for table space roll-forward recovery. If not specified, all table spaces requiring roll-forward recovery will be included. If only a subset of these table spaces is to roll forward, their names must be specified.

Example: Roll Forward for Tablespace
After three table spaces have been restored, the following commands roll one forward to the end of the logs and the other two to a point in time, both being done online:

```
db2 rollforward db sample to end of logs tablespace(TBS1) online
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
tablespace(TBS2, TBS3) online
```
Example: Roll Forward with an Overflow Log Path

After restoring the database, the following commands roll forward to a point in time, using OVERFLOW LOG PATH to specify the directory where the user exit saves archived logs:

```
db2 rollforward db sample to 2001-04-03-14.21.56.245378 and stop
overflow log path (/logs)
```

Example: Roll Forward with Multinode (EEE) Databases

In this example, there are three nodes: 0, 1, and 2. Table space TBS1 is defined on all nodes, and table space TBS2 is defined on nodes 0 and 2. After restoring the database on node 1 and TBS1 on nodes 0 and 2, the following command rolls the database forward on node 1:

```
db2 rollforward db sample to end of logs and stop
```

This example returns warning SQL1271: “Database is recovered but one or more table spaces are off-line on node(s) 0 and 2.”

```
db2 rollforward db sample to end of logs
```

RUNSTATS

RUNSTATS updates statistics about the physical characteristics of a table and its indexes. Statistics include the number of rows, number of pages, and average record length. The optimizer uses these statistics when determining access paths to the data. Use RUNSTATS when any of the following conditions apply:

- DB2 performance is slower than expected.
- The optimizer is choosing poor access paths.
- A table has had many updates.
- A table has had many inserts.
- A table has had many deletes.
- You have reorganized a table.
- A new index has been created.

After statistics have been updated, the optimizer can select new access paths to the table if you are rebinding packages using the table with BIND. After issuing this command, run a COMMIT to release locks.
Statistics are collected based on the table data that is located on the database partition where the command executes. Global table statistics for an entire partitioned table are derived by multiplying the values obtained at a database partition by the number of database partitions in the nodegroup over which the table is partitioned. All statistics are stored in the catalog tables.

**Authorization**

One of the following:

- SYSADM
- SYSCTRL
- SYSMAINT
- DBADM
- CONTROL privilege on the table

**Syntax**

```
> -RUNSTATS ON TABLE--table-name--|
    >-----+--------------------------------------------------------------------+-+>
    | '-WITH DISTRIBUTION--->-INDEXES ALL-----' |
    | '-AND--INDEX--index-name--' |
    | '-DETAILED--INDEX--index-name--' |
    | .-CHANGE----. |
    '|-SHRLEVEL--REFERENCE--' |
```

**Description**

Running RUNSTATS with only some options (such as for one index) can cause inconsistent statistics. If this happens, a warning message is returned. For example, if table distribution statistics are gathered on the first issue, and only index statistics are
gathered on the second issue, inconsistencies may be detected and the table distribution statistics are dropped. You should then issue RUNSTATS to gather table distribution statistics.

RUNSTATS cannot be used for declared temporary tables, as these are not recorded in the catalog tables. RUNSTATS can be used to gather up-to-date statistics for the catalog tables.

In EEE, RUNSTATS collects statistics for a table on the node from which it is invoked. If the table does not exist on that node, the first node in the nodegroup is selected.

The following parameters can be used for the RUNSTATS command:

- **TABLE table-name** Updates statistics for this table. Use the fully qualified name or alias in the form schema.table-name, such as ADAMACHE.ADDRESS. The schema is the user name under which the table was created. If no options are specified, only table statistics will be updated. Other users will have access to the table while statistics are gathered. For typed tables, specify the root table.

- **WITH DISTRIBUTION** Specifies that distribution statistics will be gathered. This is important for tables with skew (an uneven distribution of values, for example, in ADDRESS, more last names in Toronto start with A than Z). The number of most frequent values collected is defined by the num_freqvalues database configuration parameter. This might be “SMITH” or “CHAN” in a table of last names. The number of quantiles collected is defined by the num_quantiles database configuration parameter.

- **AND INDEXES ALL** Updates statistics on the table and all of its indexes.

- **AND INDEX index-name** Updates statistics on both the table and the specified index, where index-name is a fully qualified name in the form schema.index-name.

- **FOR INDEXES ALL** Updates statistics only on the table’s indexes. If statistics on the table have never been generated, DB2 calculates statistics on the table as well as on the indexes.

- **FOR INDEX index-name** Updates statistics only on the specified index. If table statistics have never been generated, DB2 gathers statistics on the table as well as on the index. The index name is a fully qualified name in the form schema.index-name.

- **DETAILED** Calculates extended index statistics.

- **SHRLEVEL** Specifies one of the following:
  - **CHANGE** Other users can read from and write to the table while statistics are calculated.
REFERENCE Other users can have read-only access to the table while statistics are calculated.

Example

The following command collects statistics only on the table, without distribution statistics:

```sql
db2 runstats on table smith.table1
```

The next example collects statistics only on the table, with distribution statistics:

```sql
db2 runstats on table smith.table1 with distribution
```

The following command collects basic statistics only on indexes:

```sql
db2 runstats on table smith.table1 for indexes all
```

The next example collects statistics on table and all indexes (basic level):

```sql
db2 runstats on table smith.table1 and indexes all
```

The following command collects statistics on the table, with distribution statistics and index statistics:

```sql
db2 runstats on table smith.table1 with distribution and indexes all
```

The next command collects all possible statistics (distribution and extended index):

```sql
db2 runstats on table smith.table1 with distribution and detailed index
```

Finally, the following example collects distribution statistics only on index INDEX1:

```sql
db2 runstats on table smith.table1 with distribution for index smith.index1
```

SET RUNTIME DEGREE

This command sets the maximum run-time degree of intrapartition parallelism for SQL statements for specified active applications. SET RUNTIME DEGREE can be used to override the value that was determined at SQL statement compilation time.
RUNTIME DEGREE should be the number of processors (CPUs) you want the application to exploit for every SELECT. If you have one concurrent user on a four-way SMP machine, a value of 4 gives you a good chance of using every processor for complex SELECT statements. If dozens of users share a four-way machine, use a lower run-time degree. If the processors are often idle because your workload drives a lot of disk activity, the workload is “I/O bound.” In this case, you can overparallelize the query and have more agents per query than processors. (A useful experiment would be to try a degree of 8 for a four-way machine that is I/O bound.) If the workload is “CPU-bound,” and the disks tend to be idle while all processors are 100 percent utilized, a higher run-time degree will not help.

Note that intrapartition parallelism specifies the amount of query parallelism within one DB2 instance. For EEE, DB2 can exploit both intra- and interpartition parallelism. Interpartition parallelism is the ability of DB2 to do work in parallel by dividing it among many DB2 instances. You increase interpartition parallelism by adding more DB2 instances as you add CPUs and machines.

The run-time degree of intrapartition parallelism specifies the maximum number of parallel operations that will be used when an SQL statement is executed. The degree of intrapartition parallelism for an SQL statement can be specified at statement compilation time by using the CURRENT DEGREE special register or the degree bind option. The maximum run-time degree of intrapartition parallelism for an active application can be chosen by using SET RUNTIME DEGREE. The max_querydegree database manager configuration parameter specifies the maximum run-time degree for any SQL statement executing on this instance of the database manager.

The actual run-time degree will be the lowest of these:

- max_querydegree configuration parameter
- Application run-time degree
- The SQL statement compilation degree

Authorization
SYSADM or SYSCTRL.

Syntax

```
>>-SET RUNTIME DEGREE FOR----------------------------->

--------+ALL-------------------------------+--TO--degree--------><

| .-,-----------------------------.
| V

'-(-----application-handle-------)----'
```
**Description**

The following are the parameter descriptions for SET RUNTIME DEGREE:

- **FOR: ALL** Specifies that the degree will apply to all applications.
- **application-handle** Specifies the agent for which the new degree applies. List application handles using LIST APPLICATIONS.
- **TO degree** Specifies the maximum run-time degree of intrapartition parallelism.

**Example**

The following example sets the maximum run-time degree of parallelism for two users, with application-handle values of 41408 and 55458, to 4:

```
DB2 SET RUNTIME DEGREE FOR (41408, 55458) TO 4
```

The following examples use LIST APPLICATIONS to identify an application, and then sets the degree of parallelism to 2:

```
db2 list applications
Auth Id  Application    Appl.      Application Id  DB
# of
Name    Agents
-------- -------------- ---------- ------------------------------
-------- ----- --------
ADAMACHE cheese.exe 2   *LOCAL.DB2.010702053513
SAMPLE   1
```

```
db2 SET RUNTIME DEGREE FOR (2) TO 2
DB20000I The SET RUNTIME DEGREE command completed successfully.
```

**SET TABLESPACE CONTAINERS**

SET TABLESPACE CONTAINERS is used with RESTORE DATABASE. A backup of a database, or one or more table spaces, keeps a record of all table space containers used by the table spaces being backed up. During a restore, all containers listed in the backup are checked to see if they currently exist and are accessible. If one or more of the containers is inaccessible for any reason, the restore fails. To allow a restore in such a case, redirection of table space containers is accomplished during the restore with SET TABLESPACE CONTAINERS. This includes adding, changing, or removing table space containers.
Use SET TABLESPACE CONTAINERS if one or more containers become inaccessible for any reason, and the restore must be directed to different containers. A redirected restore is not allowed in combination when a user exit program runs the restore.

**Authorization**

SYSADM or SYSCTRL.

**Syntax**

```
>>-SET TABLESPACE CONTAINERS FOR--tablespace-id----------------->
>-----+--------------------------------------------------+----->
|  .-REPLAY--.                                     |
'--+-IGNORE--+---ROLLFORWARD CONTAINER OPERATIONS--'
>----USING----------------------------------------------------->
>-----+-(-----PATH--"container-string"---+---)---------------------------+>
|    .-,----------------------------------------------------.      |
|    V                                                      |      |
| '-(------+-FILE---+---"container-string"--number-of-pages---+---)--' |
| '-DEVICE-'                                              |
>--------------------------------------------------------------><
```

**Description**

The following are the parameter descriptions for SET TABLESPACE CONTAINERS:

- **FOR tablespace-id** Specifies the ID of the table space used by the database being restored.
- **REPLAY ROLLFORWARD CONTAINER OPERATIONS** Specifies that any ALTER TABLESPACE operation issued against this table space since the database was backed up will be redone if the database is rolled forward after the restore.
- **IGNORE ROLLFORWARD CONTAINER OPERATIONS** Specifies that ALTER TABLESPACE operations in the log are ignored during a roll forward.
- **USING PATH “container-string”** Identifies the containers that will belong to an SMS table space and into which the table space data will be stored. It is an
absolute or relative directory name. If the directory name is not absolute, it is relative to the database directory. The PATH cannot exceed 240 bytes.

**USING FILE/DEVICE “container-string” number-of-pages** Identifies one or more containers that will belong to a DMS table space and into which the table space data will be stored. The container type (either FILE or DEVICE) and its size are specified, and a mixture of file and device containers is allowed. The string cannot exceed 254 bytes. For a file container, the string must be an absolute or relative file name. If the filename is not absolute, it is relative to the database directory. For a device container, the string must be a device name that already exists.

**Example**

The following four steps will run a redirected restore for database MYDB:

1. **RESTORE DATABASE** with the REDIRECT option.

   ```
   db2 restore db mydb replace existing redirect
   ```

   You can stop the restore before it finishes by using **ABORT**:

   ```
   db2 restore db mydb abort
   ```

2. **Run SET TABLESPACE CONTAINERS** for each table space whose containers require a new definition (this is a Windows example):

   ```
   db2 set tablespace containers for 5 using
   (file 'f:\ts3con1' 20000, file 'f:\ts3con2' 20000)
   ```

3. Verify that the containers of the restored database are the ones specified in this step, and issue **LIST TABLESPACE CONTAINERS**.

   After successful completion of steps 1 through 3 above, issue:

   ```
   db2 restore db mydb continue
   ```

   This is the final step of the redirected restore.

4. If step 3 fails, or if the restore is aborted, the redirected restore can be restarted, beginning at step 1.

**SET TAPE POSITION**

DB2 for Windows supports backup and restore to streaming tape devices. Use this command for tape positioning. This command is not required on UNIX, Linux, or OS/2.

**Authorization**

None.
DB2 Command Line Processor Commands

Syntax

```
>>-SET TAPE POSITION-----+------------+--TO--position----------><
 'ON--device--'
```

Description

The following are the parameter descriptions for SET TAPE POSITION:

- **ON device**  Specifies a valid tape device name. The default is \ \ TAPE0.
- **TO position**  Specifies the mark at which the tape is positioned. DB2 writes a tape mark after every backup. A value of 1 specifies the first position, 2 specifies the second position, and so on. If the tape is positioned at tape mark 1, archive 2 is positioned restored.

START DATABASE MANAGER

This command starts the current database manager instance on a single node or on all nodes defined in EEE.

Authorization

SYSADM, SYSCTRL, or SYSMAINT.

Syntax

```
>>>+-START---+DATABASE MANAGER--+-+-------------------------------+---->
 |        +-DB MANAGER-------+ |   '-PROFILE--profile--'
 |        '-DBM--------------' |
 | '-db2start--------------------'
-----------------------------<
  '-NODENUM--nodenum--| start options |--'
```

start options

```
| --+-------------------------------|
 | +-ADDNODE-- addnode options ++
 | +-STANDALONE------------------+
 | 'RESTART-- restart options |--'
```
**addnode options**

```
|---HOSTNAME--hostname---PORT--logical-port--------------------->
|---+--------------------------+---+-----------------+-------->
'|COMPUTER--computer-name--'  '|USER--username--'
|---+---------------------+---+-----------------+----------->
'|PASSWORD--password--'  '|NETNAME--netname--'
|---+-------------------------+-------------------------------|
`LIKE NODE--node-number--`
'-WITHOUT TABLESPACES-----'
```

**restart options**

```
|---+---------------------+---+---------------------+----------->
|---+--------------------------+-------------------------------------|
'|HOSTNAME--hostname--'  '|PORT--logical-port--'
|---+-----------------+-------------------------------------|
'|NETNAME--netname--'
```

**Description**

Once you start DB2, the database manager instance runs until you stop it, even if all applications have disconnected. Do not issue `START DATABASE MANAGER` on a node that has only clients.

If DB2 starts successfully, a successful completion message is displayed. If an error occurs, processing stops, and an error message is displayed. In EEE, messages are returned on the node that issued `START DATABASE MANAGER`. If no parameters are specified in EEE, DB2 is started on all nodes using the parameters in the node configuration file (`db2nodes.cfg`).

db2 start database manager is valid on all editions of DB2. However, starting DB2 with the parameters that follow is valid only with EEE:

- **PROFILE profile** Specifies the profile at each node that defines the DB2 environment. DB2 accesses the profile for the nodes that are started. The profile must reside in the sqllib directory of the instance owner. The environment variables in the profile file are not necessarily all defined in the user session.

- **NODENUM nodenum** Specifies the node to start. If no other options are specified, a normal startup is done at this node. nodenum can be from 0 to 999. If a node is not being added although `ADDNODE` is specified, the node must
exist in the db2nodes.cfg file of the instance owner. If no node number is specified, DB2 starts all nodes defined in the node configuration file.

- **ADDNODE** Adds the new node to the db2nodes.cfg file of the instance owner with the hostname and logical-port values (requires either SYSADM or SYSCTRL authority). Ensure that the combination of hostname and logical-port is unique. DB2 runs ADDNODE internally to create all existing databases on the node being added. The db2nodes.cfg file is updated when you run db2stop; the node is not part of the EEE system until the next db2start following the db2stop. When database partitions are created on the new node, their configuration parameters are set to the default.

The following are the ADDNODE options:

- **HOSTNAME hostname** Specifies the hostname added to the db2nodes.cfg file.
- **PORT logical-port** Specifies the logical port added to the db2nodes.cfg file. This is a number from 0 to 999.
- **COMPUTER computer-name** Specifies the computer name of the machine on which the new node is (used only on Windows).
- **USER username** Specifies the user name for the account on the new node (used only on Windows).
- **PASSWORD password** Specifies the password for the account on the new node (used only on Windows).
- **NETNAME netname** Specifies the netname added to the db2nodes.cfg file. If not specified, this defaults to hostname.
- **LIKE NODE node-number** Specifies that containers for the system temporary table spaces are the same as the containers on another node you name with LIKE NODE. The copied node must already be the db2nodes.cfg file.
- **WITHOUT TABLESPACES** Specifies that containers for the system temporary table spaces are not created for any of the databases. Use ALTER TABLESPACE to add system temporary table space containers to each database before using the database.
- **STANDALONE** Specifies that the node is started in standalone mode. DB2 will not attempt to establish a connection to any other node. Use STANDALONE when adding a node.
- **RESTART** Starts the database manager after a failure. Other nodes are still operating, and this node attempts to connect to the others. If neither the hostname nor the logical-port parameter is specified, the database manager restarts using the hostname and logical-port values specified in db2nodes.cfg. If either parameter is specified, the new values are sent to other nodes when a connection is established. The db2nodes.cfg file is updated with this information.
The following are the options for RESTART:

- **HOSTNAME hostname with RESTART, specifying hostname** Uses the hostname to override the HOSTNAME in the node configuration file.
- **PORT logical-port with RESTART** Specifies the logical port number to override the PORT in the node configuration file. If not specified, PORT defaults to the logical-port value that corresponds to the nodenum value in the db2nodes.cfg file. PORT is a number from 0 to 999.
- **NETNAME netname** Specifies the netname to override the one specified in the db2nodes.cfg file. If not specified, this parameter defaults to the netname value that corresponds to the nodenum value in the db2nodes.cfg file.

**Example**
The following is sample output from DB2START issued on a three-node system with nodes 10, 20, and 30:

```
04-07-1997 10:33:05    10   0   SQL1063N  DB2START processing was successful.
04-07-1997 10:33:07    20   0   SQL1063N  DB2START processing was successful.
04-07-1997 10:33:07    30   0   SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
```

**STOP DATABASE MANAGER**
This command stops the current database manager instance, but it does not stop DB2 if any application is connected to a database. If there are no database connections, but there are instance attachments, STOP DATABASE MANAGER detaches them and stops DB2. It also deactivates any outstanding database activations before stopping.

On EEE, STOP DATABASE MANAGER can stop DB2 on one or all nodes. It can also drop a node from the db2nodes.cfg file.

**Authorization**
SYSADM, SYSCTRL, or SYSMAINT.

**Syntax**
```bash
>>>-+STOP----DATABASE MANAGER-+---------------------------------------------+---+
    +DB MANAGER--------+    'PROFILE--profile--'
```
DB2 Command Line Processor Commands

The following are the parameter descriptions for STOP DATABASE MANAGER:

- **PROFILE profile** Specifies the name of the profile run at startup to set the DB2 environment (for EEE only). If a profile for START DATABASE MANAGER was specified, the same profile must be specified here. The profile resides in the sqllib directory of the instance owner.

- **NODENUM nodenum** Specifies the node to stop (for EEE only). This can be a value from 0 to 999 from db2nodes.cfg. If not specified, all nodes defined in the node configuration file are stopped.

- **DROP NODENUM nodenum** Specifies that the node will be dropped from the db2nodes.cfg file (for EEE only). Before using DROP NODENUM, run DROP NODE VERIFY to ensure that there is no user data on this node. When this parameter is specified, all nodes in the db2nodes.cfg file are stopped.

- **FORCE** Does a FORCE APPLICATION ALL to end-user and application connections when stopping the database manager.

- **NODENUM nodenum** Specifies the node to stop after all applications and users on the node have been forced off (for EEE only). If FORCE is used without this parameter, all applications on all nodes are forced before all the nodes are stopped.

Example

This example stops DB2 and forces all applications and users to disconnect:

```
db2 stop database manager force
```

The following is sample output from db2stop issued on a three-node system with nodes 10, 20, and 30:

```
04-07-1997 10:32:53 10 0 SQL1064N DB2STOP processing was successful.
```
**DB2: The Complete Reference**

04-07-1997 10:32:54    20   0   SQL1064N  DB2STOP processing was successful.
04-07-1997 10:32:55    30   0   SQL1064N  DB2STOP processing was successful.
SQL1064N  DB2STOP processing was successful.

**TERMINATE**

TERMINATE explicitly terminates the command line processor’s back-end process and breaks the database connection (like combining QUIT and CONNECT RESET).

**Authorization**

None.

**Syntax**

```
>>-TERMINATE---------------------------------------------------><
```

**Description**

If an application is connected to a database or a process is in the middle of a unit of work, TERMINATE causes the database connection to be lost. An internal commit is then performed.

Although TERMINATE and CONNECT RESET both break the connection to a database, only TERMINATE results in termination of the back-end process that maintains a connection to the database for the command line processor.

**Truncate Table**

DB2 does not have a truncate table command. There are two ways to remove the contents of a table without logging, but preserve the table definition in DB2.

The fastest is IMPORT REPLACE with an IXF file that contains no rows. IMPORT is the best choice because it does not lock the table space.

The alternative is LOAD, which must be used if the target table has referential integrity dependencies or summary tables defined on it. If you plan to use LOAD a lot, put each table that gets LOADED (or uses LOAD to be emptied) in its own table space, or use IMPORT REPLACE.

See the IMPORT and LOAD command descriptions in this chapter.
**UNCATALOG DATABASE**

This command deletes a database entry from the system database directory.

**Authorization**

SYSADM or SYSCTRL.

**Syntax**

```
>>-UNCATALOG----+-DATABASE-+--database-alias-------------------><
'-'DB--------'
```

**Description**

The DATABASE database-alias parameter specifies the database to remove from the directory of cataloged databases. The database will not be dropped unless you issue DROP DATABASE.

Only entries in the system database directory can be uncataloged. Entries in the local database directory are removed only when the data and database are deleted with DROP DATABASE.

To recatalog the database, use CATALOG DATABASE. To list the databases that are cataloged on a node, use LIST DATABASE DIRECTORY. To refresh the CLP’s directory cache, use TERMINATE. To refresh DB2’s shared cache, stop (db2stop) and then restart (db2start) the database. To refresh the directory cache for another application, stop and then restart that application.

---

**UNCATALOG DCS DATABASE**

This command deletes an entry from the DCS directory.

**Authorization**

SYSADM or SYSCTRL.

**Syntax**

```
>>-UNCATALOG DCS----+-DATABASE-+--database-alias---------------><
'-'DB--------'
```

---

**UNCATALOG DATABASE**

This command deletes a database entry from the system database directory.

**Authorization**

SYSADM or SYSCTRL.

**Syntax**

```
>>-UNCATALOG----+-DATABASE-+--database-alias-------------------><
'-'DB--------'
```

**Description**

The DATABASE database-alias parameter specifies the database to remove from the directory of cataloged databases. The database will not be dropped unless you issue DROP DATABASE.

Only entries in the system database directory can be uncataloged. Entries in the local database directory are removed only when the data and database are deleted with DROP DATABASE.

To recatalog the database, use CATALOG DATABASE. To list the databases that are cataloged on a node, use LIST DATABASE DIRECTORY. To refresh the CLP’s directory cache, use TERMINATE. To refresh DB2’s shared cache, stop (db2stop) and then restart (db2start) the database. To refresh the directory cache for another application, stop and then restart that application.

---

**UNCATALOG DCS DATABASE**

This command deletes an entry from the DCS directory.

**Authorization**

SYSADM or SYSCTRL.

**Syntax**

```
>>-UNCATALOG DCS----+-DATABASE-+--database-alias---------------><
'-'DB--------'
```

---
Description

The DATABASE database-alias parameter specifies the alias of the DCS database to uncatalog. DCS databases are also cataloged in the system database directory as remote databases that can be uncataloged using UNCATALOG DATABASE.

To recatalog a database in the DCS directory, use CATALOG DCS DATABASE. To list the DCS databases that are cataloged on a node, use LIST DCS DIRECTORY.

UNCATALOG LDAP DATABASE

This command is used to deregister the database from LDAP.

Authorization

None.

Syntax

```
>>-UNCATALOG LDAP----+DATABASE-+--dbalias-------------------->
                   '-DB-------'
>-----+------------------------------------------+-------------><
'-USER--username--+---------------------+--'
  '-PASSWORD--password--'
```

Description

When a database is dropped, the database object is removed from LDAP. The database is also automatically deregistered from LDAP when the database server that manages the database is deregistered from LDAP. It may, however, be necessary to manually uncatalog the database from LDAP in two situations:

- If the database server does not support LDAP. In this case, the administrator must manually uncatalog each database from LDAP after the database is dropped.
- During DROP DATABASE, the database object cannot be removed from LDAP (because LDAP cannot be accessed). In this case, the database is still removed from the local machine, but the existing entry in LDAP is not deleted.

The following parameters can be used with the UNCATALOG LDAP DATABASE command:
**UNCATALOG LDAP NODE**

This command removes a node entry in LDAP.

**Authorization**

None.

**Syntax**

```plaintext
>>>-UNCATALOG LDAP---NODE--nodename------------------------------>

>-----+------------------------------------------+------------->

'-USER--username--+---------------------+--'

'-PASSWORD--password--'
```

**Description**

The following are the parameter descriptions for UNCATALOG LDAP NODE:

- **NODE nodename** Specifies the name of the node to remove from the LDAP directory.
- **USER username** Specifies the user’s LDAP distinguished name (DN). The LDAP user DN must have sufficient authority to delete the object from the LDAP directory. If the user’s LDAP DN is not specified, the credentials of the current logon user are used.
- **PASSWORD password** Specifies the account password.

**UNCATALOG NODE**

UNCATALOG NODE deletes an entry from the node directory.
Authorization

SYSADM or SYSCTRL.

Syntax

```
>>-UNCATALOG NODE--nodename------------------------------------><
```

Description

In this command, the NODE nodename parameter specifies the node entry to remove from the node directory.

UNCATALOG NODE can be executed on any type of node, but only the local directory is affected, even if there is an attachment to a remote instance or a different local instance.

**UNCATALOG ODBC DATA SOURCE**

This command uncatalogs a user or system ODBC data source.

Authorization

None.

Syntax

```
.-USER----.
>>-UNCATALOG----+--------+--ODBC DATA SOURCE--data-source-name--><
'-SYSTEM-'
```

Description

The following are the parameter descriptions for UNCATALOG ODBC DATA SOURCE:

- **USER** Uncatalogs a user data source. This is the default if no keyword is specified.
- **SYSTEM** Uncatalogs a system data source.
- **ODBC DATA SOURCE** *data-source-name* Specifies the name of the data source to uncatalog
UPDATE ADMIN CONFIGURATION

This command modifies individual entries in the database manager configuration file that control the DB2 Administration Server. The following configuration parameters can be changed:

- AGENT_STACK_SZ
- AUTHENTICATION
- DIAGLEVEL
- DIAGPATH
- DISCOVER
- DISCOVER_COMM
- FILESERVER
- IPX_SOCKET
- NNAME
- OBJECTNAME
- QUERY_HEAP_SZ
- SYSDM_GROUP
- SYSCTRL_GROUP
- SYSMaint_GROUP
- TPNAME
- TRUST_ALLCLNTS
- TRUST_CLNTAUTH

Authorization

SYSADM.

Syntax

```bash
>>>UPDATE ADMIN-----+CONFIGURATION+-+---------------------------->
+-CONFIG--------+
'-'CFG---------'

-----------------------------
V

V

| >-----USING----config-keyword value+-+---------------------------->
```
**Description**

The USING config-keyword value specifies the configuration parameter to change.

To view or print a list of the admin configuration parameters, use GET ADMIN CONFIGURATION. To restore all admin configuration parameters to the defaults, use RESET ADMIN CONFIGURATION. Changes to the database manager configuration file become effective only at db2start.

**UPDATE CLI CONFIGURATION**

This command updates the contents of a specified section in the db2cli.ini file. This file is used as the DB2 CLI configuration file. It contains various keywords and values that can modify the behavior of the DB2 CLI and the applications using it. The file is divided into sections, each section corresponding to a database alias name.

**Authorization**

SYSADM.

**Syntax**

```
>>-UPDATE CLI----+-CONFIGURATION-+------------------------------>
+--CONFIG--------+
'-CFG-----------'

>-----+-------------------------+--FOR SECTION--section-name---->
'-AT--+-GLOBAL-+---LEVEL--'
'-USER---'

-.-----------------
 V                  |

>-----USING-----keyword value---+------------------------------><
```

**Description**

The following are the parameter descriptions for UPDATE CLI CONFIGURATION:

- **FOR SECTION section-name** Specifies the name of the section to update. If it does not exist, a new section is created.
- **AT GLOBAL LEVEL** Specifies that the CLI configuration parameter will be updated at the global level. This parameter is applicable only when LDAP support is enabled.
**AT USER LEVEL** Specifies that the CLI configuration parameter will be updated at the user level. This parameter is applicable only when LDAP support is enabled.

**USING keyword value** Specifies the CLI/ODBC parameter to change. The section name and the keywords specified on this command are not case-sensitive. However, the keyword values are case-sensitive. If a keyword value is a string containing single quotation marks or embedded blanks, the entire string must be delimited by double quotation marks. When the AT USERLEVEL keywords are specified, the CLI configuration parameters for the specified section are updated only for the current user; otherwise, they are updated for all users on the local machine. The CLI configuration at the user level is maintained in the LDAP directory and cached on the local machine. When reading the CLI configuration, DB2 always reads from the cache. The cache is refreshed when

- The user updates the CLI configuration.
- The user explicitly forces a refresh of the CLI configuration using the REFRESH LDAP command.

In an LDAP environment, users can configure a set of default CLI settings for a database cataloged in the LDAP directory. When an LDAP cataloged database is added as a DSN, any default CLI settings that exist in the LDAP directory are configured for that DSN on the local machine. The AT GLOBAL LEVEL clause must be specified to configure a CLI parameter as a default setting.

### Example

Here’s an example of how to update a CLI configuration:

```db2 update cli cfg for section tstcllx
  using TableType "'TABLE','VIEW','SYSTEM TABLE'"
```

## UPDATE COMMAND OPTIONS

This command sets one or more command options during an interactive session or from a batch input file. The settings revert to system defaults (or the system default overrides in DB2OPTIONS) when the interactive session or batch input file ends.

**Authorization**

None.
Syntax

```
>>-UPDATE COMMAND OPTIONS USING---------------------------------->
   .---------------------------------.
V
>--------option-letter--+-ON--value--+--+----------------------><
'-OFF--------'
```

Description

These settings override system defaults, settings in DB2OPTIONS, and options specified using the command line option flags. The file input option (-f) and the statement termination option (-t) cannot be updated using this command.

The parameters are as follows:

- **USING option-letter**  The following option letters can be set:
  - a Displays SQLCA
  - c Autocommits SQL statements
  - e Displays SQLCODE/SQLSTATE
  - l Logs commands in a history file
  - n Removes newline character
  - o Displays to standard output
  - p Displays DB2 interactive prompt
  - r Saves output report to a file
  - s Stops execution on command error
  - v Echoes current command
  - w Shows SQL statement warning messages
  - z Redirects all output to a file.

- **ON value**  Specifies that the e, l, r, and z options require a value if they are turned on. For the e option, the value can be c to display the SQLCODE, or s to display the SQLSTATE. For the l, r, and z options, the value represents the name to be used for the history file or the report file. No other options accept a value.

Example

This command displays the SQLCA (Communications Area):

```
db2 update command options using a on
```
UPDATE DATABASE CONFIGURATION

This command modifies individual entries in a specific database configuration file. A database configuration file resides on every node for EEE. This command affects only the node on which it is executed.

Authorization

SYSADM, SYSCTRL, or SYSMAINT.

Syntax

>>-UPDATE----+-DATABASE-+---+-CONFIGURATION-+------------------->
       '-DB-------'   +-CONFIG--------+
           '  '-CFG-----------'           
               V

>>----FOR--database-alias---USING-----config-keyword value---+--><

Description

The following are the parameter descriptions for UPDATE DATABASE CONFIGURATION:

- **FOR database-alias**  Specifies the alias of the database whose configuration is updated.
- **USING config-keyword value** Specifies the database configuration parameter to change or a brief description of configurable parameters. (See the GET DATABASE CONFIGURATION command.) To reset database configuration parameters to defaults, use RESET DATABASE CONFIGURATION. Not all parameters can be updated.
Most changes become effective only after they are loaded into memory. All applications must disconnect from the database before this can occur. If an error occurs, the database configuration file does not change.

**UPDATE DATABASE MANAGER CONFIGURATION**

This command modifies individual entries in the database manager configuration file.

**Authorization**

SYSADM.

**Syntax**

```plaintext
>>>UPDATE------DATABASE MANAGER-------CONFIGURATION-------
+DB MANAGER------+  +CONFIG------+
'DBM------'    'CFG------'

----------------------------------------

V

>>>USING-----config-keyword value------
```

**Description**

The USING config-keyword value specifies the database manager configuration parameter to change.

For a brief description of configurable parameters, see the GET DATABASE MANAGER CONFIGURATION command. To reset the database manager configuration parameters to the default values, use RESET DATABASE MANAGER CONFIGURATION.

Most changes to the database manager configuration file become effective only after they are loaded into memory. For a server configuration parameter, this occurs during execution of db2start. For a client configuration parameter, this occurs when the application is restarted. If the client is the command line processor, first use TERMINATE. If an error occurs, the database manager configuration file does not change.

**Example**

This example turns on circular logging for the sample database (effective the next time the database is started):

```plaintext
update database configuration for sample using logretain off
```
UPDATE LDAP NODE

This command updates the protocol information associated with a node entry for the DB2 server in LDAP.

Authorization

None.

Syntax

```
>>-UPDATE LDAP---NODE--nodename---------------------+-------
        '-HOSTNAME--hostname--'

>-----+---------------------+----+
       '-SVCENAME--svcename--'  '-NNAME--nname--'

>-----+---------------------+---+
       '-NETWORK--net_id--'  '-PARTNERLU--partner_lu--'

>-----+---------------------+---+
       '-TPNAME--tpname--'  '-MODE--mode--'

>-----+---------------------+---+
       '-SECURITY--NONE----+'  '-LANADDRESS--lan_address--'
        'SAME----
        '-PROGRAM-'

>-----+---------------------+---+
       '-CHGPWDLU--change_password_lu--'

>-----+---------------------+---+
       '-IPX_ADDRESS--ipxaddr--'  '-WITH--"comments"--'

>-----+---------------------+---+
       '-USER--username--'  '-PASSWORD--password--'
```

Description

The following are the parameter descriptions for UPDATE LDAP NODE:

- **NODE nodename** Specifies the node name when updating a remote DB2 server.
  The node name is the value specified when registering the DB2 server in LDAP.
DB2: The Complete Reference

- **HOSTNAME hostname** Specifies the TCP/IP hostname (or IP address).
- **SVCENAME svcename** Specifies the TCP/IP service name or port number.
- **NNAME nname** Specifies the NetBIOS workstation name.
- **NETWORK net_id** Specifies the APPN network ID.
- **PARTNERLU partner_lu** Specifies the APPN partner LU name for the DB2 server machine.
- **TPNAME tpname** Specifies the APPN transaction program name.
- **MODE mode** Specifies the APPN mode name.
- **SECURITY** Specifies the APPN security level: NONE, SAME, or PROGRAM.
- **LANADDRESS lan_address** Specifies the APPN network adapter address.
- **CHGPWDLU change_password_lu** Specifies the partner LU used when changing the password for a host database server.
- **IPX_ADDRESS ipxaddr** Specifies the complete IPX address.
- **WITH “comments”** Describes the DB2 server. You can enter a comment that describes the server registered in the network directory.
- **USER username** Identifies the user's LDAP distinguished name (DN). The LDAP user DN must have sufficient authority to create and update the object in the LDAP directory. If the user's LDAP DN is not specified, the credentials of the current logon user are used.
- **PASSWORD password** Specifies the account password.

**UPDATE MONITOR SWITCHES**

When the database manager starts, the settings of the six database monitor recording switches are determined by the dft_mon database manager configuration parameters. (See the GET DATABASE MANAGER CONFIGURATION command). The database monitor records a base set of information at all times. Users who require more than this basic information can turn on the appropriate switches, but each switch slows DB2 down slightly. This command turns one or more database monitor recording switches on or off.

The amount of information available in output from GET SNAPSHOT reflects the switches that are on. Information is collected by DB2 only after a switch is turned on. The switches remain set until db2stop is issued or the application that issued UPDATE MONITOR SWITCHES ends. To clear the information related to a particular switch, set the switch off, and then on. Updating switches in one application does not affect other applications.

To view the switch settings, use GET MONITOR SWITCHES.
Authorization

SYSADM, SYSCTRL, or SYSMAINT.

Syntax

```
V
>>-UPDATE MONITOR SWITCHES USING-------switch-name--+-ON--+--+-->

'-OFF-

>>-------------------------<

+-AT NODE--nodenum++++-

'--GLOBAL--------'
```

Description

The following are the parameter descriptions for UPDATE MONITOR SWITCHES:

- **USING switch-name** Identifies the switch. The following switch names are available:
  - BUFFERPOOL
  - LOCK
  - SORT
  - STATEMENT (SQL statement information)
  - TABLE
  - UOW (logical unit of work information)

- **AT NODE nodenum** Specifies the node for which the status of the monitor switches is displayed.

- **GLOBAL** Returns an aggregate result for all nodes in a partition database system.

---

**UPDATE HISTORY FILE**

The UPDATE HISTORY FILE command updates the location, device type, or comment in a history file entry. The history file is used by database administrators for record keeping. It is used internally by DB2 for the automatic recovery of incremental backups.
Authorization
SYSADM, SYSCTRL, SYSMAINT, or DBADM.

Syntax

```sql
>>>-UPDATE HISTORY FOR--object-part--WITH------------------------>

>-----+-LOCATION--new-location--DEVICE TYPE--new-device-type--+-><

'-COMMENT--new-comment----------------------------------'
```

Description
The following are the parameter descriptions for UPDATE HISTORY FILE:

- **FOR object-part** Specifies the identifier for the backup or copy image. It is a timestamp with an optional sequence number from 001 to 999.
- **LOCATION new-location** Specifies the new physical location of a backup. The interpretation of this parameter depends on the device type.
- **DEVICE TYPE new-device-type** Specifies a new device type for storing the backup. Valid device types are the following:
  - D Disk
  - K diskette
  - T Tape
  - A TSM (formerly ADSM)
  - U user exit
  - Other
- **COMMENT new-comment** Specifies a new comment to describe the entry.

Example
To update the history file entry for the full database backup taken on April 13, 2001, at 10:00 A.M., enter this command:

```sql
db2 update history for 20010413100000001 with
  location /backup/dbbackup.1 device type d
```