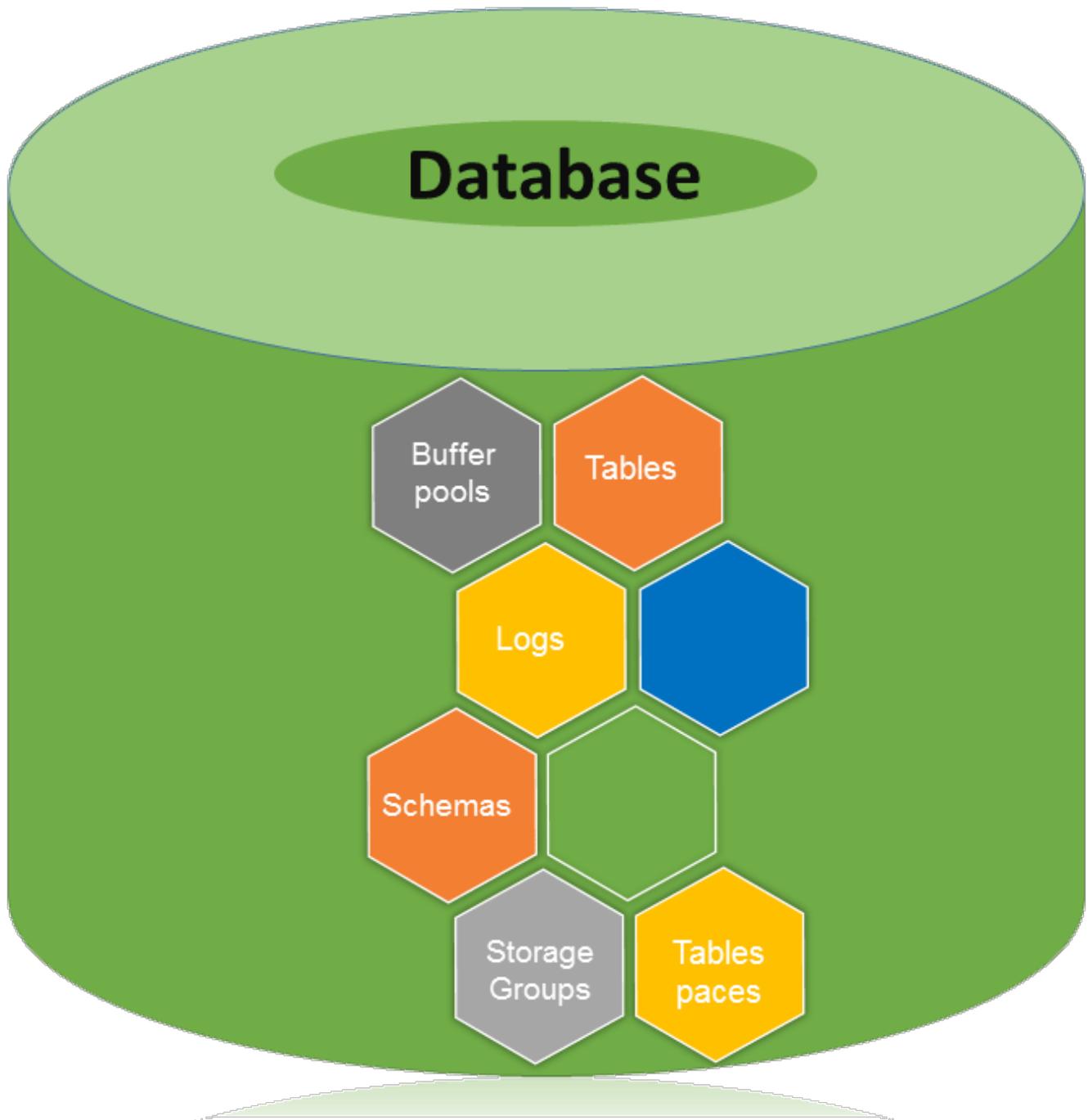


This chapter describes creating, activating and deactivating the databases with the associated syntax.

Database architecture



A database is a collection of Tables, Schemas, Bufferpools, Logs, Storage groups and Tablespace working together to handle database operations efficiently.

Database directory

Database directory is an organized repository of databases. When you create a database, all the details about database are stored in a database directory, such as details of default storage devices, configuration files, and temporary tables list etc.

Partition global directory is created in the instance folder. This directory contains all global information related to the database. This partition global directory is named as NODExxxx/SQLyyy, where xxxx is the data partition number and yyy is the database token.

In the partition-global directory, a member-specific directory is created. This directory contains local database information. The member-specific directory is named as MEMBERxxxx where xxxx is a member number. DB2 Enterprise Server Edition environment runs on a single member and has only one member specific directory. This member specific directory is uniquely named as MEMBER0000.

Partitioned global directory

Directory Location : <instance>/NODExxx/SQLxxx

The partition-global directory contains database related files as listed below.

- Global deadlock write-to-file event monitoring files
- Table space information files [SQLSPCS.1, SQLSPCS.2]
- Storage group control files [SQLSGF.1, SQLSGF.2]
- Temporary table space container files. [/storage path//T0000011/C000000.TMP/SQL00002.MEMBER0001.TDA]
- Global Configuration file [SQLDBCONF]
- History files [DB2RHIST.ASC, DB2RHIST.BAK, DB2TSCHG.HIS, DB2TSCHG.HIS]
- Logging-related files [SQLOGCTL.GLFH.1, SQLOGCTL.GLFH.2]
- Locking files [SQLINSLK, SQLTMPLK]
- Automatic Storage containers

Member specific directory

Directory location : /NODExxxx/SQLxxxx/MEMBER0000

This directory contains:

- Objects associated with databases
- Buffer pool information files [SQLBP.1, SQLBP.2]
- Local event monitoring files
- Logging-related files [SQLOGCTL.LFH.1, SQLOGCTL.LFH.2, SQLOGMIR.LFH].
- Local configuration files
- Deadlocks event monitor file. The detailed deadlock events monitor files are stored in the database directory of the catalog node in case of ESE and partitioned database environment.

Creating database

You can create a database in instance using the “CREATE DATABASE” command. All databases are created with the default storage group “IBMSTOGROUP”, which is created at the time of creating an instance. In DB2, all the database tables are stored in “tablespace”, which use their respective storage groups.

The privileges for database are automatically set as PUBLIC [CREATETAB, BINDADD, CONNECT, IMPLICIT_SCHEMA, and SELECT], however, if the RESTRICTIVE option is present, the privileges are not granted as PUBLIC.

Creating non-restrictive database

This command is used to create a non-restrictive database.

Syntax: [To create a new Database. 'database_name' indicates a new database name, which you want to create.]

```
db2 create database <database name>
```

Example: [To create a new non-restrictive database with name 'one']

```
db2 create database one
```

Output:

```
DB20000I The CREATE DATABASE command completed successfully.
```

Creating restrictive database

Restrictive database is created on invoking this command.

Syntax: [In the syntax below, "db_name" indicates the database name.]

```
db2 create database <db_name> restrictive
```

Example: [To create a new restrictive database with the name 'two']

```
db2 create database two restrictive
```

Creating database with different user defined location

Create a database with default storage group "IBMSTOGROUP" on different path. Earlier, you invoked the command "create database" without any user-defined location to store or create database at a particular location. To create the database using user-defined database location, the following procedure is followed:

Syntax: [In the syntax below, 'db_name' indicates the 'database name' and 'data_location' indicates where have to store data in folders and 'db_path_location' indicates driver location of 'data_location'.]

```
db2 create database '<db_name>' on '<data location>' dbpath on '<db_path_location>'
```

Example: [To create database named 'four', where data is stored in 'data1' and this folder is stored in 'dbpath1']

```
db2 create database four on '/data1' dbpath on '/dbpath1'
```

Viewing local or system database directory files

You execute this command to see the list of directories available in the current instance.

Syntax:

```
db2 list database directory
```

Example:

```
db2 list database directory
```

Output:

```
System Database Directory
Number of entries in the directory = 6
Database 1 entry:
Database alias           = FOUR
Database name           = FOUR
Local database directory =
/home/db2inst4/Desktop/dbpath
```

```
Database release level      = f.00
Comment                    =
Directory entry type       = Indirect
Catalog database partition number = 0
Alternate server hostname   =
Alternate server port number =
Database 2 entry:
Database alias             = SIX
Database name              = SIX
Local database directory   = /home/db2inst4
Database release level     = f.00
Comment                    =
Directory entry type       = Indirect
Catalog database partition number = 0
Alternate server hostname   =
Alternate server port number =
```

Activating database

This command starts up all necessary services for a particular database so that the database is available for application.

Syntax:['db_name' indicates database name]

```
db2 activate db <db_name>
```

Example: [Activating the database 'one']

```
db2 activate db one
```

Deactivating database

Using this command, you can stop the database services.

Syntax:

```
db2 deactivate db <db_name>
```

Example: [To Deactivate database 'one']

```
db2 deactivate db one
```

Connecting to database

After creating a database, to put it into use, you need to connect or start database.

Syntax:

```
db2 connect to <database name>
```

Example: [To Connect Database one to current CLI]

```
db2 connect to one
```

Output:

```
Database Connection Information
Database server      = DB2/LINUX8664 10.1.0
SQL authorization ID = DB2INST4
Local database alias = ONE
```

Verifying if database is restrictive

To check if this database is restrictive or not, here is the syntax:

Syntax: [In the following syntax, 'db' indicates Database, 'cfg' indicates configuration, 'db_name' indicates database name]

```
db2 get db cfg for <db_name> | grep -i restrict
```

Example: [To check if 'one' database is restricted or not]

```
db2 get db cfg for one | grep -i restrict
```

Output:

```
Restrict access = NO
```

Configuring the database manager and the database

Instance configuration *Databasemanagerconfiguration* is stored in a file named 'db2system' and the database related configuration is stored in a file named 'SQLDBCON'. These files cannot be edited directly. You can edit these files using tools which call API. Using the command line processor, you can use these commands.

Database Manager Configuration Parameters

Syntax: [To get the information of Instance Database manager]

```
db2 get database manager configuration
```

OR

```
db2 get dbm cfg
```

Syntax: [To update instance database manager]

```
db2 update database manager configuration
```

OR

```
db2 update dbm cfg
```

Syntax: [To reset previous configurations]

```
db2 reset database manager configuration
```

OR

```
db2 reset dbm cfg
```

Database Configuration Parameters

Syntax: [To get the information of Database]

```
db2 get database configuration
```

OR

```
db2 get db cfg
```

Syntax: [To update the database configuration]

```
db2 update database configuration
```

OR

```
db2 update db cfg
```

Syntax: [To reset the previously configured values in database configuration]

```
db2 reset database configuration
```

OR

```
db2 reset db cfg
```

Syntax: [To check the size of Current Active Database]

```
db2 "call get_dbsize_info(?,?,?, -1)"
```

Example: [To verify the size of Currently Activate Database]

```
db2 "call get_dbsize_info(?,?,?, -1)"
```

Output:

```
Value of output parameters
-----
Parameter Name   : SNAPSHOTTIMESTAMP
Parameter Value  : 2014-07-02-10.27.15.556775
Parameter Name   : DATABASESIZE
Parameter Value  : 105795584
Parameter Name   : DATABASECAPACITY
Parameter Value  : 396784705536
Return Status = 0
```

Estimating space required for database

To estimate the size of a database, the contribution of the following factors must be considered:

- System Catalog Tables
- User Table Data
- Long Field Data
- Large Object *LOB* Data
- Index Space
- Temporary Work Space
- XML data
- Log file space
- Local database directory
- System files

Checking database authorities

You can use the following syntax to check which database authorities are granted to PUBLIC on the non-restrictive database.

Step 1: connect to database with authentication user-id and password of instance.

Syntax: [To connect to database with username and password]

```
db2 connect to <db_name> user <userid> using <password>
```

Example: [To Connect "one" Database with the user id 'db2inst4' and password 'db2inst4']

```
db2 connect to one user db2inst4 using db2inst4
```

Output:

```
Database Connection Information
Database server      = DB2/LINUX8664 10.1.0
SQL authorization ID = DB2INST4
Local database alias = ONE
```

Step2: To verify the authorities of database.

Syntax: [The syntax below shows the result of authority services for current database]

```
db2 "select substr(authority,1,25) as authority, d_user, d_group,
d_public, role_user, role_group, role_public,d_role from table(
sysproc.auth_list_authorities_for_authid ('public','g'))as t
order by authority"
```

Example:

```
db2 "select substr(authority,1,25) as authority, d_user, d_group,
d_public, role_user, role_group, role_public,d_role from table(
sysproc.auth_list_authorities_for_authid ('PUBLIC','G'))as t
order by authority"
```

Output:

AUTHORITY D_ROLE	D_USER	D_GROUP	D_PUBLIC	ROLE_USER	ROLE_GROUP	ROLE_PUBLIC	

ACCESSCTRL	*	*	N	*	*	N	*
BINDADD	*	*	Y	*	*	N	*
CONNECT	*	*	Y	*	*	N	*
CREATETAB	*	*	Y	*	*	N	*
CREATE_EXTERNAL_ROUTINE	*	*	N	*	*	N	*
CREATE_NOT_FENCED_ROUTINE	*	*	N	*	*	N	*
CREATE_SECURE_OBJECT	*	*	N	*	*	N	*
DATAACCESS	*	*	N	*	*	N	*
DBADM	*	*	N	*	*	N	*
EXPLAIN	*	*	N	*	*	N	*
IMPLICIT_SCHEMA	*	*	Y	*	*	N	*
LOAD	*	*	N	*	*	N	*
QUIESCE_CONNECT	*	*	N	*	*	N	*
SECADM	*	*	N	*	*	N	*
SQLADM	*	*	N	*	*	N	*
SYSADM	*	*	*	*	*	*	*
SYCTRL	*	*	*	*	*	*	*
SYSMAINT	*	*	*	*	*	*	*
SYSMON	*	*	*	*	*	*	*
WLMADM	*	*	N	*	*	N	*

20 record(s) selected.

Dropping Database

Using the Drop command, you can remove our database from instance database directory. This command can delete all its objects, table, spaces, containers and associated files.

Syntax: [To drop any database from an instance]

```
db2 drop database <db_name>
```

Example: [To drop 'six' database from instance]

```
db2 drop database six
```

Output:

DB20000I The DROP DATABASE command completed successfully

Loading [Mathjax]/jax/output/HTML-CSS/jax.js