**1. DB2 - Instance**

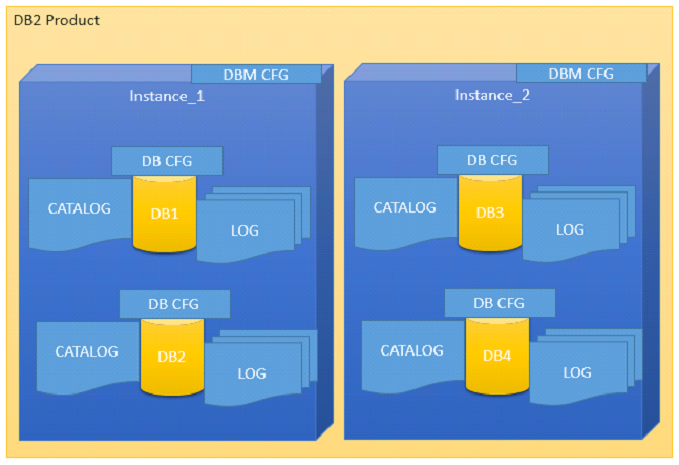
## Introduction

An Instance is a logical environment for DB2 Database Manager. Using instance, you can manage databases. Depending on our requirements, you can create multiple instances on one physical machine. The contents of Instance directory are:

* Database Manager Configuration file
* System Database Directory
* Node Directory
* Node Configuration File [db2nodes.cfg]
* Debugging files, dump files

For DB2 Database Server, the default instance is “DB2”. It is not possible to change the location of Instance directory after its creation. An instance can manage multiple databases. In an instance, each database has a unique name, its own set of catalog tables, configurations files, authorities and privileges.

## Architecture of instance in DB2 product



## Multiple instances

You can create multiple instances in one DB2Server on Linux, UNIX and Windows. It is possible to install multiple DB2Servers on a physical machine.

## Creating instance on Linux

You can create multiple instances on Linux and UNIX if DB2 Server is installed as root user. An instance can run simultaneously on Linux and UNIX independently. You can work within a single instance of the database manager at a time.

An Instance folder contains database configuration files and folders. The Instance directory is stored at different locations on Windows depending on the operating system versions.

## Listing instances

The following command is used to list instances:

### db2ilist

This command lists all the instances that are available on a system.

**Syntax:**

db2ilist

**Example:**[To see how many instances are created in DB2 copy]

db2ilist

**Output:**

db2inst1

db2inst2

db2inst3

## Instance environment commands

These commands are useful to work with arrangement of instance in the DB2 CLI.

### Get instance

This command shows details of the currently running instance.

**Syntax:**

db2 get instance

**Example:**[To see the current instance which activated the current user]

db2 get instance

**Output:**

The current database manager instance is : db2inst1

### Set instance

To start or stop the database manager of an instance on DB2 UDB, the following command is executed for the current instance.

**Syntax:**

set db2instance=<instance\_name>

**Example:**[ To arrange the “db2inst1” environment to current user]

set db2instance=db2inst1

### db2start

Using this command, you can start an instance. Before this, you need to run “set instance”.

**Syntax:**

db2start

**Example:**[To start an instance]

db2start

**Output:**

SQL1063N DB2START processing was successful

### db2stop

Using this command you can stop a running instance.

**Syntax:**

db2stop

**Output:**

SQL1064N DB2STOP processing was successful.

## Creating an instance

Let us see how to create a new instance.

### db2icrt

If you want to create a new instance, you need to log in with root. Instance id is not a root id or a root name.

Here are the steps to create a new instance:

**Step1**: Create an operating system user for instance.

**Syntax:**

useradd -u <ID> -g <group name> -m -d <user location> <user name>

-p <password>

**Example**: [To create a user for instance with name ‘db2inst2’ in group ‘db2iadm1’ and password ‘db2inst2’]

useradd -u 1000 -g db2iadm1 -m -d /home/db2inst2 db2inst2 -p db2inst2

**Step2**: Go to the DB2 instance directory in root user for create new instance.

**Location:**

cd /opt/ibm/db2/v10.1/instance

**Step3**: Create instance using the syntax below:

**Syntax:**

./db2icrt -s ese -u <inst id> <instance name>

**Example**: [To create a new instance ‘db2inst2’ in user ‘db2inst2’ with the features of ‘ESE’ (Enterprise Server Edition)]

./db2icrt -s ese -u db2inst2 db2inst2

**Output:**

DBI1446I The db2icrt command is running, please wait.

….

…..

DBI1070I Program db2icrt completed successfully.

## Arranging communication port and host for an instance

Edit the /etc/services file and add the port number. In the syntax given below, ‘inst\_name’ indicates the Instance name and ‘inst\_port’ indicates port number of instance.

**Syntax:**

db2c\_<inst name> <inst\_port>/tcp

**Example**: [Adding ‘50001/tcp’ port number for instance ‘db2inst2’ with variable ‘db2c\_db2inst2’ in ‘services’ file]

db2c\_db2inst2 50001/tcp

**Syntax 1**: [Update Database Manager Configuration with service name. The following syntax ‘svcename’ indicates the instance service name and ‘inst\_name’ indicates the instance name]

db2 update database manager configuration using svcename db2c\_&<inst\_name>

**Example 1**: [Updating DBM Configuration with variable svcename with value ‘db2c\_db2inst2’ for instance ‘db2inst2’

db2 update database manager configuration using svcename db2c\_db2inst2

**Output**

DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed successfully.

**Syntax 2**: set the “tcpip” communication protocol for the current instance

db2set DB2COMM=tcpip

**Syntax 3**: [Stopping and starting current instance to get updated values from database manager configuration]

db2stop

db2start

## Updating an instance

You can update an instance using following command:

### db2iupdt

This command is used to update the instance within the same version release. Before executing this command, you need to stop the instance database manager using “db2stop” command. The syntax below “inst\_name” indicates the previous released or installed db2 server instance name, which you want to update to newer release or installed db2 server version.

**Syntax 1**: To update an instance in normal mode

db2iupdt <inst\_name>

**Example1:**

./db2iupdt db2inst2

**Syntax 2**: To update an instance in debugging mode

db2iupdt -D <inst\_name>

**Example**

db2iupdt -D db2inst2

## Upgrading an instance

You can upgrade an instance from previous version of DB2 copy to current newly installed version of DB2 copy.

### db2iupgrade

On Linux or UNIX system, this command is located in DB2DIR/instance directory. In the following syntaxes, “inst\_name” indicates the previous version DB2 instance and “inst\_username” indicates the current installed version DB2 copy instance user.

**Syntax 2**:

db2iupgrade -d -k -u <inst\_username> <inst\_name>

**Example**:

db2iupgrade -d -k -u db2inst2 db2inst2

**Command Parameters:**

**-d** : Turns debugging mode on.

**-k** : Keeps the pre-upgrade instance type if it is supported in the DB2 copy, from where you are running this command.

If you are using the Super User (su) on Linux for db2iupgrade command, you must issue the “su” command with the “-” option.

## Dropping an instance

You can drop or delete the instance, which was created by “db2icrt” command.

### db2idrop

On Linux and UNIX operating system, this command is located in the DB2\_installation\_folder/instance directory.

**Syntax**: [in the following syntax, ‘inst\_username’ indicates username of instance and ‘inst\_name’ indicates instance name]

db2idrop -u <inst\_username> <inst\_name>

**Example**: [To drop db2inst2]

./db2idrop -u db2inst2 db2inst2

## Using other commands with instance

Command to find out which DB2 instance we are working on now.

**Syntax 1**: [to check the current instance activated by database manager]

db2 get instance

**Output:**

The current database manager instance is: db2inst1

**Syntax 2**: [To see the current instance with operating bits and release version]

db2pd -inst | head -2

**Example:**

db2pd -inst | head -2

**Output:**

Instance db2inst1 uses 64 bits and DB2 code release SQL10010

**Syntax 3**: [To check the name of currently working instance]

db2 select inst\_name from sysibmadm.env\_inst\_info

**Example:**

db2 select inst\_name from sysibmadm.env\_inst\_info

**Output:**

INST\_NAME --------------------------------------

db2inst1

1 record(s) selected.

**Syntax**: [To set a new instance as default]

db2set db2instdef=<inst\_name> -g

**Example**: [To array newly created instance as a default instance]

db2set db2instdef=db2inst2 -g

# DB2 - Tablespaces

## Introduction

A table space is a storage structure, it contains tables, indexes, large objects, and long data. It can be used to organize data in a database into logical storage group which is related with where data stored on a system. This tablespaces are stored in database partition groups

## Benefits of tablespaces in database

The table spaces are beneficial in database in various ways given as follows:

**Recoverability**: Tablespaces make backup and restore operations more convenient. Using a single command, you can make backup or restore all the database objects in tablespaces.

**Automatic storage Management**: Database manager creates and extends containers depending on the needs.

**Memory utilization**: A single bufferpool can manage multiple tablespaces. You can assign temporary tablespaces to their own bufferpool to increase the performance of activities such as sorts or joins.

## Container

Tablespaces contains one or more containers. A container can be a directory name, a device name, or a filename. In a database, a single tablespace can have several containers on the same physical storage device. If the tablespace is created with automatic storage tablespace option, the creation and management of containers is handled automatically by the database manager. If it is not created with automatic storage tablespace option, you need to define and manage the containers yourself.

## Default tablespaces

When you create a new database, the database manager creates some default tablespaces for database. These tablespace is used as a storage for user and temporary data. Each database must contain at least three tablespaces as given here:

1. Catalog tablespace
2. User tablespace
3. Temporary tablespace

**Catalog tablespace**: It contains system catalog tables for the database. It is named as SYSCATSPACE and it cannot be dropped.

**User tablespace**: This tablespace contains user-defined tables. In a database, we have one default user tablespace, named as USERSPACE1. If you do not specify user-defined tablespace for a table at the time you create it, then the database manager chooses default user tablespace for you.

**Temporary tablespace**: A temporary tablespace contains temporary table data. This tablespace contains system temporary tablespaces or user temporary tablespace.

System temporary tablespace holds temporary data required by the database manager while performing operation such as sorts or joins. A database must have at least one system temporary tablespace and it is named as TEMPSPACE1. It is created at the time of creating the database. User temporary tablespace holds temporary data from tables. It is created with DECLARE GLOBAL TEMPORARY TABLE or CREATE GLOBAL TEMPORARY TABLE statement. This temporary tablespace is not created by default at the time of database creation.

**Tablespaces and storage management:**

Tablespaces can be setup in different ways, depending on how you want to use them. You can setup the operating system to manage tablespace allocation, you can let the database manager allocate space or you can choose automatic allocation of tablespace for your data.

The following three types of managed spaces are available:

**System Managed Space (SMS)**: The operating system’s file system manager allocates and manages the space where the table is stored. Storage space is allocated on demand. This model consists of files representing database objects. This tablespace type has been deprecated in Version 10.1 for user-defined tablespaces, and it is not deprecated for catalog and temporary tablespaces.

**Database Managed Space (DMS)**: The Database Server controls the storage space. Storage space is pre- allocated on the file system based on container definition that you specify when you create the DMS table space. It is deprecated from version 10.1 fix pack 1 for user-defined tablespaces, but it is not deprecated for system tablespace and temporary tablespace.

**Automatic Storage Tablespace**: Database server can be managed automatically. Database server creates and extends containers depend on data on database. With automatic storage management, it is not required to provide container definitions. The database server looks after creating and extending containers to make use of the storage allocated to the database. If you add storage space to a storage group, new containers are automatically created when the existing container reach their maximum capacity. If you want to use the newly-added storage immediately, you can rebalance the tablespace.

**Page, table and tablespace size:**

Temporary DMS and automatic storage tablespaces, the page size you choose for your database determines the maximum limit for the tablespace size. For table SMS and temporary automatic storage tablespaces, the page size constrains the size of table itself. The page sizes can be 4kb, 8kb, 16kb or 32kb.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tablespace type** | **4K page size limit** | **8K page size limit** | **16K page size limit** | **32K page size limit** |
| DMS, non-temporary automatic storage tablespace regular | 64G | 128G | 256G | 512G |
| DMS, temporary DMS and non- temporary automatic storage table space large | 1892G | 16384G | 32768G | 65536G |