IBM DB2® 9.7

Getting started with DB2
Hands-on Lab

Information Management Cloud Computing Center of Competence

IBM Canada Lab
1. **Introduction**

This module is designed to introduce you to instances, databases, and to practice with DB2 commands, including connecting to a DB2 server.

2. **Objectives**

By the end of this lab, you will be able to:
- Create a DB2 database
- Practice with DB2 commands
- Connect to a remote database

3. **Suggested reading**

**Getting started with DB2 Express-C eBook (Chapters 1, 3-7, 9)**
https://www.ibm.com/developerworks/wikis/display/DB2/FREE+Book+-+Getting+Started+with+DB2+Express-C

A free eBook that can quickly get you up to speed with DB2

4. **Getting Started**

4.1 **Environment Setup Requirements**

To complete this lab you will need the following:

- DB2 Academic Associate Bootcamp VMware image
- VMware Player 2.x or VMware Workstation 5.x or later

4.2 **Initial Steps**

1. Start the VMware image by clicking the **Power On** button in VMware.

2. At the login prompt, login with the following credentials:
Username: **db2inst1**  
Password: **password**

3. Open a terminal window by right-clicking on the **Desktop** and choosing the **Open Terminal** item.

![Desktop with Open Terminal option highlighted](image)

4. Ensure that the DB2 Database Manager has been started by issuing the following command at the prompt:

```bash
db2inst1@db2rules:~> db2start
```

**Note:** This command will only work if you logged in as the user **db2inst1**. If you accidentally logged in as another user, type `su – db2inst1` at the command prompt **password:** **password**.

5. Throughout the lab, the **SAMPLE** database will be used to explore the features of DB2. To create the **SAMPLE** database we need to first remove the existing **SAMPLE** database by issuing the following command.

```bash
db2inst1@db2rules:~> db2 drop db sample
```
5. Working with DB2 Databases

5.1 First Steps

First Steps is a graphical tool that helps get you started with DB2. As part of the DB2 installation process, the First Steps panel is displayed allowing the user to generate a number of sample databases to work with:

Most users will want to create the SAMPLE database and use that to explore the features of DB2. This panel can be invoked by issuing the command `db2fs` from a command-line prompt.

```
db2inst1@db2rules:~> db2fs
```

First Steps requires a browser and browser profile to run and function properly. Select Yes to create the browser profile and select OK to continue.
In addition, issuing the command `db2sampl` from a command-line prompt will also generate the SAMPLE database. Once the SAMPLE button has been selected, an additional panel is displayed to determine where the SAMPLE database will be created.
When creating the SAMPLE database, it is recommended that you select the XML and SQL objects and data option. This option will generate the database in UTF-8 (Unicode) format that will allow you to manipulate XML objects. If you do not select the XML option, you will not be able to add XML objects to your SAMPLE database.

Now let's move on to creating a DB2 database without a GUI.

### 5.2 Using the db2sampl command

If you created the SAMPLE database using the First Steps method describe above, drop the sample database so we can see how to create it using the command line.
As mentioned in the previous section, we can issue the command `db2sampl` from a command-line prompt in order to also generate the SAMPLE database.

Let's take a look at these tablespaces that DB2 created when we issued to create the SAMPLE database. Connect to the sample database (discussed in further details later) and list the tablespaces for the database by issuing the following commands:

You should observe an output similar to the following:

<table>
<thead>
<tr>
<th>Tablespace ID</th>
<th>Name</th>
<th>Type</th>
<th>Contents</th>
<th>State</th>
<th>Detailed explanation:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SYSCATSPACE</td>
<td>Database managed space</td>
<td>All permanent data. Regular table space.</td>
<td>0x0000</td>
<td>Normal</td>
</tr>
<tr>
<td></td>
<td>TEMPSPACE1</td>
<td>System managed space</td>
<td>System Temporary data</td>
<td>0x0000</td>
<td>Normal</td>
</tr>
<tr>
<td></td>
<td>USERSPACE1</td>
<td>Database managed space</td>
<td>All permanent data. Large table space.</td>
<td>0x0000</td>
<td>Normal</td>
</tr>
</tbody>
</table>

5.3 Try it: Practice DB2 commands

In this section, practice DB2 commands following the instructions provided. Solutions for this part can be found at the end of the lab, but we encourage you to only look at them after you try this on your own!

1.
Open a Command Line Processor (CLP) either on Windows or Linux. Your prompt should be "db2 =>". If this is not the case, you are not in the CLP.

2. Perform these operations:

   a) Create the database "DB101" with default values. This may take a few minutes to complete while DB2 creates database objects discussed in the lesson.

   b) Connect to the database

   c) Turn off "Self tuning memory" for this database. Hint: Look in the db cfg

   d) Show the db cfg confirming that "Self tuning memory" is now off.

   e) Stop the instance. You may receive an error indicating there are connections to databases. If this is the case, list which connections you have by running the command:

```
db2=> list applications
```

   To force all applications (connections) off, run the command:

```
db2=> force applications all
```

   Now try again to stop the instance.

   f) Start the instance

   g) List the contents of the DB2 Profile Registry. You need to do this from the Linux shell or DB2 Command Window

**5.4 Working with scripts**

**5.4.1 Creating a SQL Script**

1) Create a SQL script with these characteristics:

   a) Script name:   myscript1.db2

   b) SQL the script runs:
```sql
select * from department
create table tbl1 (name varchar(30), phone varchar(20))
insert into tbl1 values
    ('Tom','123456789'),  ('Mary','987654321')
select * from tbl1
drop table tbl1
```

c) Use "#" as the statement terminator

2)

Run the script from the DB2 Command Window or Linux Shell. You first need to connect to the SAMPLE database.

### 5.4.2 Creating an operating system script

1) Create an operating system script with these characteristics:

   a) Script name: `myscript2`

   b) The script should invoke `myscript1.db2` created in the previous section.

   c) The script should be invoked with 3 parameters as follows:

```
myscript2 <dbname> <userID> <password>,
```

where:

- `<dbname>` is the database name

- `<userID>` and `<password>` are parameters for connecting to the database

2) Run `myscript2` using the SAMPLE database as follows:

   a) Grant execute permission:

```
chmod +x myscript2
```

   b) Execute using the SAMPLE database:
If running on Windows, there’s no need to use the chmod command; instead, add the “.bat” extension, and execute as follows:

```
C:\> myscrip2.bat sample <userID> <password>
```

### 5.5 Connecting to a DB2 Database

Before working with a database, a user or application program must establish a connection with that database. You connect to databases using the CONNECT statement.

#### 5.5.1 Using the CONNECT statement

Before you can issue a SQL statement, you have to connect to a database.

To connect to our sample database, enter the command:

```
db2inst1@db2rules:~> db2 CONNECT TO sample USER db2inst1 USING password
```

You can also connect to a database and have DB2 prompt you for the password by issuing the command:

```
db2inst1@db2rules:~> db2 CONNECT TO sample USER db2inst1
```

Or if the database you want to connect to is local, and you simply want to connect to it using the default user ID, issue the command:

```
db2inst1@db2rules:~> db2 CONNECT TO sample
```

Anytime that you need to terminate the connection to the database, you can issue the TERMINATE command:

```
db2inst1@db2rules:~> db2 terminate
```

#### 5.5.2 Cataloging a local DB2 Database

So, why does a database have to be cataloged? Without this information, an application can't connect to a database!

When you create a database, the database is automatically cataloged in the local and system database directories. Therefore, a local connection will work as we showed in the previous section.
For a **remote connection**, that is, when the DB2 client and DB2 servers don’t reside on the same system, then you must run catalog commands at the DB2 client. This is described in the next section.

DB2 has multiple directories that are used to access databases. These directories allow DB2 to find databases known to it whether they are on the local system or a remote system. The system database directory contains a list and pointer indication where each of the known databases can be found.

To put an entry into any of these directories, a `CATALOG` command is used. To remove an entry, the `UNCATALOG` command is used.

To view the entries in the system databases directory, execute the command:

```
db2inst1@db2rules:~> db2 list database directory
```

The output should be similar to the following:

```
System Database Directory

Number of entries in the directory = 1

Database 1 entry:

  Database alias                         = SAMPLE
  Database name                          = SAMPLE
  Local database directory               = /home/db2inst1
  Database release level                 = d.00
  Comment                                =
  Directory entry type                   = Indirect
  Catalog database partition number      = 0
  Alternate server hostname              =
  Alternate server port number           =
```

Here we can see the sample database cataloged on our system. This information is used to connect to the database.

**However,** if this information was **not** set here (ie, the database was **not** cataloged upon creation) we would not be able to connect to the database.

Let’s take a look at how it would affect us if the database was not cataloged. Issue the `UNCATALOG` command on the SAMPLE database:

```
db2inst1@db2rules:~> db2 uncatalog database sample
```

Then try connecting to the SAMPLE database.

```
db2inst1@db2rules:~> db2 connect to sample
```
Notice that it is not possible. You will most likely receive a SQL1013N message:

SQL1013N  The database alias name or database name "SAMPLE"
could not be found.  SQLSTATE=42705

The fact is that the sample database and the files associated with it still exist within our
system, however the information in the database directory does not exist for the DB2
client to establish a connection. You can verify this by checking the system database
directory as before:

db2inst1@db2rules:~> db2 list database directory

SQL1057W  The system database directory is empty.  SQLSTATE=01606

Catalog the database by entering the following commands in the command line
processor:

db2 catalog database database_name as database_alias on path/drive

where:

- database_name represents the name of the database you want to catalog.
- database_alias represents a local nickname for the database you want to
catalog.
- path/drive specifies the path on which the database being cataloged resides.

To catalog the database called sample so that it has the local database alias mysample,
enter the following command:

db2inst1@db2rules:~> db2 catalog database sample as mysample

Issue the following command to check the database directory for this new entry:

db2inst1@db2rules:~> db2 list database directory

System Database Directory

   Number of entries in the directory = 1

Database 1 entry:

      Database alias = MYSAMPLE
    Database name = SAMPLE
Local database directory = /home/db2inst1
   Database release level = d.00
    Comment =
  Directory entry type = Indirect
Now, since the database is cataloged and the information is back in the database directory, we can connect to it using the alias we have specified with the catalog statement above and issue our SQL statements.

```
db2inst1@db2rules:~> db2 connect to mysample
```

Issue a TERMINATE command to terminate the connection to the SAMPLE database.

```
db2inst1@db2rules:~> db2 terminate
```

### 5.5.3 Try it: Cataloging a remote DB2 Database

In this section, practice DB2 commands following the instructions provided. Solutions for this part can be found at the end of the lab, but we encourage you to only look at them after you try this on your own!

Cataloging a remote DB2 database implies that you have handy the following information:

- IP Address or hostname of the remote DB2 server
- Port number of the instance where the database resides
- Name of the database to connect to
- User ID / password as defined on the remote server.

Let’s simulate how you would connect to a remote database treating your own server as the remote server by using LOCALHOST. Use the following information for your remote server:

<table>
<thead>
<tr>
<th>IP Address or hostname</th>
<th>LOCALHOST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance Port Number</td>
<td>50001</td>
</tr>
<tr>
<td>Database Name</td>
<td>SAMPLE</td>
</tr>
<tr>
<td>User ID</td>
<td>db2inst1</td>
</tr>
<tr>
<td>Password</td>
<td>password</td>
</tr>
</tbody>
</table>

Use the ALIAS “SAMPLE2” to point to this database.
Solutions for section 5.3

a) Create the database "DB101" with default values.

   ```
   db2=> create database db101
   ```

b) Connect to the database

   ```
   db2=> connect to db101
   ```

c) Turn off "Self tuning memory" for this database.

   ```
   db2=> update db cfg using SELF_TUNING_MEM off
   ```

d) Show the db cfg confirming that "Self tuning memory" is now off.

   ```
   db2=> get db cfg
   ```

e) Stop the instance.

   ```
   db2=> list applications
   db2=> force applications all
   db2=> db2stop
   ```

f) Start the instance

   ```
   db2=> db2start
   ```

g) List the contents of the DB2 Profile Registry. You need to do this from the Linux shell or DB2 Command Window

   ```
   db2=> quit
   db2set -all
   ```

Solutions for section 5.5.3

a) Catalog the TCPIP node
b) Catalog the database ensuring to point to the TCPIP node in the previous step

```
db2=> catalog tcpip node mynode remote localhost server 50001
```

c) Test that you can connect. Note that you must input a userID/psw in the syntax since with this setup we are accessing the database using TCPIP which means DB2 thinks it’s a remote database.

```
db2=> catalog db sample as sample2 at node mynode
```

d) If you cannot connect, verify db2comm at the server (which in this case happens to be the same system as your client) is set to TCPIP:

```
db2inst1@db2rules:~> db2set -all
```

e) If not set, set it as follows:

```
db2inst1@db2rules:~> db2set db2comm=tcpip
db2inst1@db2rules:~> db2stop
db2inst1@db2rules:~> db2start
```

Note that changes to DB2 profile registry variables require a db2stop/db2start for the change to take effect.

f) If the connection still does not work, verify at your DB2 server if the port is set to 50001:

```
db2inst1@db2rules:~> db2 get dbm cfg | grep SVCENAME
```

g) If SVCENAME has a string instead of a numeric value like 50001, look for this string in /etc/services. For example, if SVCENAME had a value of db2c_DB2, look for it as follows:

```
db2inst1@db2rules:~> cat /etc/services | grep db2c_DB2
```

There should be an entry like: db2c_DB2 50001/tcp

If not there, add it.
7. Summary

This exercise introduced you to the objects that make up a DB2 database, and to the factors that affect how the database is created. You also learned how to review and change DB2 configuration parameters, and how to set up connectivity.