

Getting Started with DB2 Express C v9.7

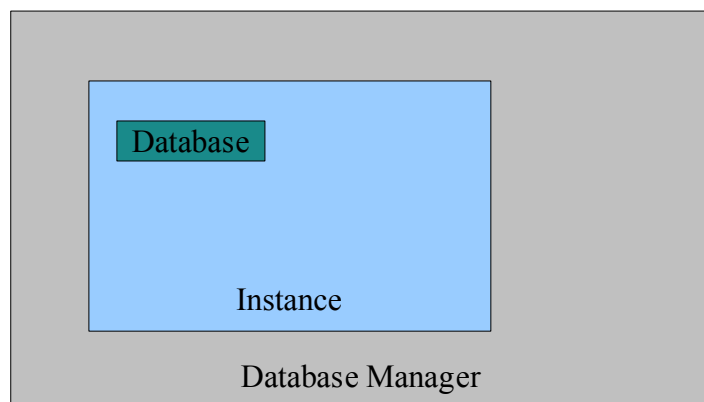
We are using DB2 Express C for the experiments and the project (SDBT).

You should use DB2 on the Amazon cloud infrastructure. See the web site for how to get started.

This document is not a full documentation of DB2. It is just an introduction to a quick start. I will introduce new functionalities as we need them. Documentation is available at the DB2 Information Center (Bookmark this!): <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>

1. Basic DB2 Concepts

You need to understand the concept of database manager, instance and databases to get started. Those concepts are in the DB2 environment section of the “Getting Started with DB2 Express C” available on the web site.



Briefly:

- The database manager is DB2-express C running on your EC2 instance
- A DB2 instance is an independent environment where databases can be created.
- A database is a collection of tables.

When db2 is running, a few processes are running in the background (db2sysc, db2acd, db2bp). No need to go into the details there for now.

Note that you can access a DB2 instance locally, on your EC2 instance. You can also access it remotely from your laptop (running Windows or Linux – or from a VM if you run Mac OS). For remote access, you need to install a client on your laptop (see Section 3).

2. AWS Setup

When you connect to your DB2 AMI for the first time (see movie on the web site and doc IBM DB2 AMI Start-up Guide), three DB2 users are created: a user in charge of the administration of the database manager (dsaur1 by default), a fenced user for adding code (UDF) to the server (db2fenc1 by default), and most importantly for our purpose a user that owns the instance created at setup time (db2inst1 by default and in the rest of this document). The DB2 AMI Start-up guide is very clear on

how to start working with your installed system: how to start/stop a db2 instance, how to setup parameters, and how to create a database and tablespaces. If you have not followed these steps, you should do it. From now on, I assume that you have created a database (that I have named tuning in this document) and that you can ssh to your AWS instance. Note that you must log in as root with ssh and then change user to db2inst1:

```
youraccount> ssh -i <KeyPair.pm> root@<AWSINSTANCE-IP>
aws-instance# su db2inst1
aws-instance#
```

You are now login as db2inst1 on your AWS instance, running a bash shell, with default location on your home directory. This is a linux server. DB2 on Linux has a couple of peculiarities: (1) there is no independent authentication mechanism – which means that the OS login/password is used as authentication for DB2, (2) a user must own a given instance -- the configuration and data files are located under that user's home directory (which is the case for db2inst1, but not for root for example).

For your convenience, you should:

1. Add the following path to your PATH environment variable:
`export PATH=$PATH:/home/db2inst1/sqllib/bin`
2. You need to source a config file that sets up all necessary environment variables:
`/home/db2inst1/sqllib/db2profile`

You should add those two steps to your bash configuration file: `/db2inst1/.bashrc` .

You can now launch the db2 client with the db2 command:

```
aws-instance# db2
```

You get a command prompt that you can use to enter commands to the database manager, as well as SQL commands once you are connected to a given database. Type quit to exit.

```
db2 => quit
DB20000I The QUIT command completed successfully.
aws-instance#
```

Before you can access a database (remember that I assume that it is called tuning), you must establish a connection with it. You can either type db2, enter the command editor, type commands and quit when you are finished, or give a command as parameter to the db2 program:

```
aws-instance# db2 connect to tuning
```

You can then enter SQL commands using the same db2 program. Use `db2 -t` to enter multiline SQL commands terminated by a semicolon.

```
aws-instance# db2 -t
```

```
db2 => create table r (a int);
DB20000I The SQL command completed successfully.
db2 => select *
db2 (cont.) => from r;
```

A

0 record(s) selected.

However it is very cumbersome as there is no advanced editing feature. An alternative is to write your SQL code into a file which you execute with db2 as follows:

```
aws-instance# db2 -tf yourfile.sql
```

You can create your own yourfile.sql. We will give an example in a second. When you are done working you should terminate the connection:

```
aws-instance# db2 connect reset
```

NOTE: This is a new setup. This is the first time we are using DB2 on AWS in class. You are likely to face some problems that I have not foreseen. Please do not panic. Send mail to me (phbo@itu.dk). I will do my best to fix your problems as fast as I can.

Now to a slightly more interesting exercise. Let us transfer some files from your laptop to your AWS instance and run some more complex SQL statements there. You can start by quitting the ssh session you had opened on your aws instance (type exit from the prompt). Note that the DB2 server processes will be running in the background, even if you are not connected.

You can download airline-schema.sql and insert-airline.sql from the web site. You now need to transfer them to your aws instance. You should use scp for this purpose:

```
youraccount> scp -i -i <KeyPair.pm> airline-schema.sql root@<AWSINSTANCE-IP>:/root/
```

This will place airline-schema.sql under the /root/ directory. Feel free to create a directory that you will use for uploading file to your AWS instance and keep your /root/ account tidy. Now, connect via ssh to your aws instance and change user to db2inst1

```
youraccount> ssh -i <KeyPair.pm> root@<AWSINSTANCE-IP>  
aws-instance# su db2inst1  
aws-instance#
```

Good time to check that you have added the appropriate path extension to your /root/.bashrc

```
aws-instance# db2 connect to tuning
```

You can now load the SQL commands contained in the file airline-schema.sql

```
aws-instance# db2 -tf airline-schema.sql
```

This is an efficient alternative to typing in commands in an interactive way.

11 tables have been created. Use the list tables command to display their names:

aws-instance# db2 list tables

<i>Table/View</i>	<i>Schema</i>	<i>Type</i>	<i>Creation time</i>
<i>AIRCRAFT</i>	<i>DB2INST1</i>	<i>T</i>	<i>2010-01-27-11.19.16.576619</i>
<i>ASSIGNED_TO</i>	<i>DB2INST1</i>	<i>T</i>	<i>2010-01-27-11.19.17.395250</i>
<i>BOOKED_ON</i>	<i>DB2INST1</i>	<i>T</i>	<i>2010-01-27-11.19.17.137540</i>
<i>CAN_FLY</i>	<i>DB2INST1</i>	<i>T</i>	<i>2010-01-27-11.19.17.717832</i>
<i>DEPARTURE</i>	<i>DB2INST1</i>	<i>T</i>	<i>2010-01-27-11.19.16.904541</i>
<i>EMPLOYEE</i>	<i>DB2INST1</i>	<i>T</i>	<i>2010-01-27-11.19.15.979429</i>
<i>EQUIPMENT</i>	<i>DB2INST1</i>	<i>T</i>	<i>2010-01-27-11.19.18.017700</i>
<i>FLIGHT</i>	<i>DB2INST1</i>	<i>T</i>	<i>2010-01-27-11.19.16.759633</i>
<i>PERSON</i>	<i>DB2INST1</i>	<i>T</i>	<i>2010-01-27-11.19.15.537831</i>
<i>PILOT</i>	<i>DB2INST1</i>	<i>T</i>	<i>2010-01-27-11.19.16.329816</i>
<i>PLANE</i>	<i>DB2INST1</i>	<i>T</i>	<i>2010-01-27-11.19.16.447932</i>
<i>R</i>	<i>DB2INST1</i>	<i>T</i>	<i>2010-01-27-10.02.44.354480</i>

12 record(s) selected.

You can also use the describe command followed by an sql query to find out the attribute names of the query result, e.g. Execute db2 -t to enter interactive mode:

db2 => describe select * from aircraft;

Column Information

Number of columns: 2

<i>SQL type</i>	<i>Type length</i>	<i>Column name</i>	<i>Name length</i>
<i>500 SMALLINT</i>	<i>2</i>	<i>SERIAL_NO</i>	<i>9</i>
<i>448 VARCHAR</i>	<i>15</i>	<i>MODEL_NO</i>	<i>8</i>

What are the attribute names for the tables Flight and Plane?

Step 2

Download the file insert-airline.sql.

Same procedure using the insert-airline.sql file that inserts rows into the tables you have just created.

```
aws-instance# db2 -tf insert-airline.sql
```

You can select all the tuples of a table using the following SQL command:

```
aws-instance# db2 select * from Aircraft
```

How many rows are there in table Flight?

What is the earlier arrival time in table Flight?

Step 3

The airline database contains information about airplanes, personel (including pilots) as well as flight routes.

The following SQL query finds out the employee number and name of the pilots booked (as passengers) on a flight they are assigned to (as pilots). Ideally, this should not happen in a consistent database. Is it the case for the database you have created? The way to find out is to run the following SQLquery that finds the employee number, name of those employees who are booked and assigned to a same flight. The result of this query should be the empty set. Use db2 -t:

```
select distinct Employee.Emp_No, Employee.name
from Employee, Booked_On, Assigned_To, Pilot
where Employee.Name = Booked_On.Name
and Employee.Emp_No = Pilot.Emp_No
and Assigned_To.Emp_No = Employee.Emp_No
and Assigned_To.Dep_date = Booked_On.Dep_Date;
```

What result do you get? Why?

3. Remote Access

This step is very much optional.

You should download the data server client from

<https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=swg-idsc97>

You will need to register to the IBM academic initiative, which is gratis and has no downside that I can see.

Once you have downloaded the client, read the readmefirst.txt file and follow the instructions (they are different on windows and linux). A GUI shows up, you should simply follow the steps. The client comprises a set of tools both commands that you can run from a shell prompt and GUI for database management. These tools are clients that connect to the database instance running on your AWS instance. At the end of the process, a post installation message shows up that lets you know that you should run the db2profile script before using the tools. You can also follow some first steps (I skipped that so I cannot say much about it).

Now, you should have a working db2 client environment on your laptop. For example you can run the db2 command:

```
your laptop $ db2 list db directory
```

This command should not return anything if you have just installed the DB2 client and there are no local databases.

What we need to do is to register the remote instance and the remote database on our client. The way it works in DB2 is that this registration is static (it is a separate step that must take place before you can connect to the database). The concept is that of a local catalog that you update with information about the remote instance and database. Run the command:

```
your laptop$ db2 catalog tcpip node aws remote <YOUR AWS INSTANCE IP> server <YOUR INSTANCE PORT NUMBER> remote_instance db2inst1
```

Note that aws is the name I gave to the node. You can pick your own.

You get the instance port number on the server by running

```
aws-instance# db2 attach to db2inst1
```

```
aws-instance# db2 get dbm cfg show detail
```

This command displays a set of lines. One of them gives the port number (that DB2 mixes with the notion of service name, which I don't want to get into):

```
TCP/IP Service name          (SVCENAME) = 50001
```

Now, you need to configure your AWS Instance so that your client can connect to the DB2 instance via port 50001. You need to configure your security group on the AWS Management console (under the running instance – there is a security group link on the left hand side). You should create a new allowed connection of type custom allowing tcp connections on port 50001 (or whatever port you have) from your IP address (e.g., 10.0.2.15/0 – don't forget the /0).

Now back to the client on your laptop. To commit the change to the catalog you need to flush the cache through:

```
your laptop$ db2 terminate
```

Once you have registered the remote instance (node), you can register the remote database:

```
your laptop$ db2 catalog database tuning on node aws
```

```
your laptop$ db2 terminate
```

After you have registered remote instance and database, you can connect to the database:

```
your laptop$ db2 connect to tuning user db2inst1 using DB2INST1PASSWORD
```

Now that you are connected you can run any command on your remote database using db2:

```
your laptop$ db2 list tables
```