Using Apache Derby:

With JDK 6, one of the biggest changes is the inclusion of a lightweight database known as Derby, which is from the Apache Database project.

Derby is installed (by default) in C:\Program Files\Java\ jdk1.8.0\db

If you are using JDK 9 or newer, you need to download Apache Derby JAR files.
As of JDK 9, it is no longer being planned to include it in the 'db' directory of Oracle JDK downloads. Developers looking ahead to JDK 9 should plan to get and bundle Apache Derby independently for the same purpose.

Apache Derby is an open source database written in Java. It is free and performs well.
Apache Derby is used in the JDK and is called Java DB. Apache Derby and Java DB are essentially the same

From https://docs.oracle.com/javadb/10.6.2.1/getstart/cgsjavadb.html :
Java DB is a relational database management system that is based on the Java programming language and SQL.
Java DB is the Oracle release of the Apache Derby project, the Apache Software Foundation's (ASF) open source relational database project.
The Java DB product includes Derby without any modification whatsoever to the underlying source code.
Because Java DB and Derby have the same functionality, the Java DB documentation refers to the core functionality as Derby.
https://docs.oracle.com/javadb/10.8.3.0/getstart/index.html

Java DB is installed in the db directory of the JDK installation.
This distribution contains scripts and libraries. The installation contains the following subdirectories:
The **bin** subdirectory contains the scripts for executing utilities and setting up the environment.
Eg : the ij (interactive Java DB) command is used to connect to the Java DB and run SQL commands
The **lib** subdirectory contains all JAR files that are used to work with Java DB.

Configuring Java DB:

Typically, you do not need to configure Java DB when you have installed JDK8.
If you come across any errors in starting up the database or running SQL commands from the command line, you need to set the following environment variables:

- Set the DERBY_HOME environment variable to the JDK_HOME\db directory.
- Set the JAVA_HOME environment variable to the JDK_HOME directory.
- Add the DERBY_HOME/bin directory to your PATH environment variable to run the ij tool.
- Include the JDK_HOME\bin directory in the PATH environment variable.

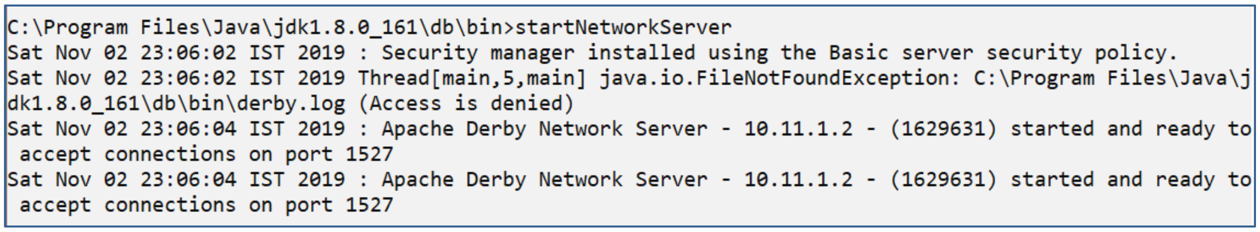
Server vs. embedded mode

Derby can be used in a ***server** mode* and in ***embedded** mode*.

In server mode, Java DB can be used by multiple users concurrently over the network. The Java DB runs in a separate JVM. Applications running in separate JVMs may connect to Java DB running in this mode

- If Derby runs in the *server mode* you start the Derby network server which will be responsible for handling the database requests.
- To start Java DB server: c:\java8\db\bin> startNetworkServer

DO NOT COPY

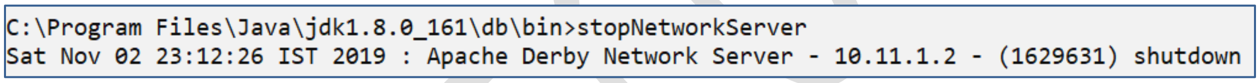


```
C:\Program Files\Java\jdk1.8.0_161\db\bin>startNetworkServer
Sat Nov 02 23:06:02 IST 2019 : Security manager installed using the Basic server security policy.
Sat Nov 02 23:06:02 IST 2019 Thread[main,5,main] java.io.FileNotFoundException: C:\Program Files\Java\j
dk1.8.0_161\db\bin\derby.log (Access is denied)
Sat Nov 02 23:06:04 IST 2019 : Apache Derby Network Server - 10.11.1.2 - (1629631) started and ready to
 accept connections on port 1527
Sat Nov 02 23:06:04 IST 2019 : Apache Derby Network Server - 10.11.1.2 - (1629631) started and ready to
 accept connections on port 1527
```

- This will start the network server which can serve an unlimited number of databases. By default the server will be listening on port 1527 but this can be changed via the -p option.
- startNetworkServer -p 3301
- By default, in server mode, Java DB starts at localhost and at port 1527

You may get an AccessControlException in starting the server. The error message may read as follows:
java.security.AccessControlException: access denied ("java.net.SocketPermission" "localhost:1527“ "listen,resolve")
To resolve the AccessControlException, you can start the server with no security manager installed as follows:
c:\java8\db\bin> startNetworkServer -noSecurityManager

- Use the following command to stop the Java DB running in server mode.
  Note that you will need to run this command using a separate command prompt.
  c:\java8\db\bin>stopNetworkServer



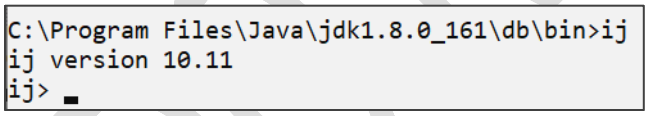
```
C:\Program Files\Java\jdk1.8.0_161\db\bin>stopNetworkServer
Sat Nov 02 23:12:26 IST 2019 : Apache Derby Network Server - 10.11.1.2 - (1629631) shutdown
```

In *embedded mode* Derby runs within the JVM of the application. In this mode only the application can access the database, e.g. another user / application will not be able to access the database.

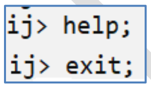
ij tool:

Derby provides a command-line tool called ij, which is an abbreviation for interactive JDBC scripting tool. This tool provides a way to connect to and manipulate Derby databases.

Start the ij tool : c:\java8\db\bin>ij



```
C:\Program Files\Java\jdk1.8.0_161\db\bin>ij
ij version 10.11
ij> _
```



```
ij> help;
ij> exit;
```

Connecting to a new database:

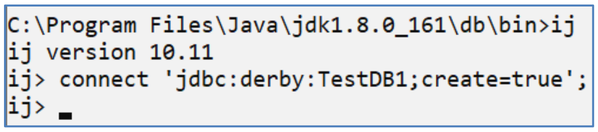
To create a database, specify the create=true attribute in the connection URL. For example, the command below creates a new database called MyDbTest:

ij> connect 'jdbc:derby:MyDbTest;create=true';

In embedded mode:

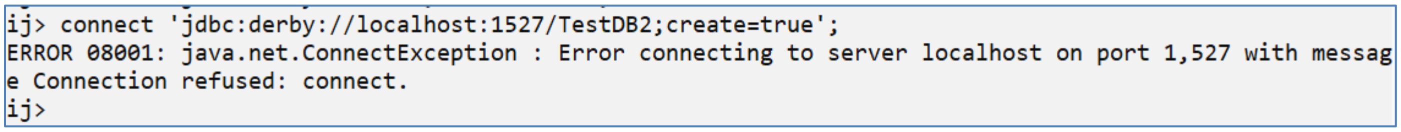
Establish connection to new database as : connect 'jdbc:derby:SampleDB;create=true';

DO NOT COPY



```
C:\Program Files\Java\jdk1.8.0_161\db\bin>ij
ij version 10.11
ij> connect 'jdbc:derby:TestDB1;create=true';
ij> _
```

In server mode:

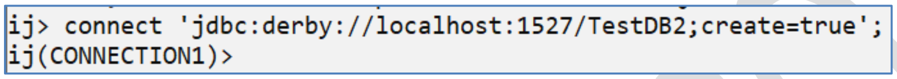


```
ij> connect 'jdbc:derby://localhost:1527/TestDB2;create=true';
ERROR 08001: java.net.ConnectException : Error connecting to server localhost on port 1,527 with messag
e Connection refused: connect.
ij>
```

Connection here is refused since I haven't started in Server mode using startNetworkServer command!

Make sure you run command prompt in Admin mode, else nothing really works

After starting server, run command: connect 'jdbc:derby://localhost:1527/TestDB2;create=true';



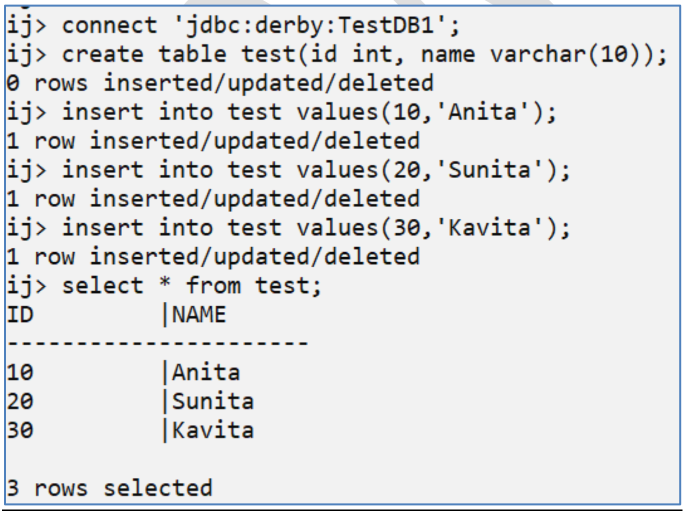
```
ij> connect 'jdbc:derby://localhost:1527/TestDB2;create=true';
ij(CONNECTION1)>
```

Connect to an existing database

Start up ij again and connect to the database you just created **(don't use create=true anymore)**

ij> connect 'jdbc:derby:TestDB1';

Using the database:



```
ij> connect 'jdbc:derby:TestDB1';
ij> create table test(id int, name varchar(10));
0 rows inserted/updated/deleted
ij> insert into test values(10,'Anita');
1 row inserted/updated/deleted
ij> insert into test values(20,'Sunita');
1 row inserted/updated/deleted
ij> insert into test values(30,'Kavita');
1 row inserted/updated/deleted
ij> select * from test;
ID         |NAME
----------------------
10         |Anita
20         |Sunita
30         |Kavita

3 rows selected
```

To use Derby in Java application:

DO NOT COPY

You have to place appropriate jar file to the classpath:

- `derby.jar`: for embedded driver.
- `derbyclient.jar`: for network client driver.

Derby differentiates two types of JDBC driver:

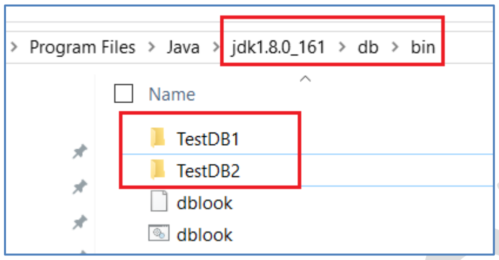
Type of driver	JDBC Driver Class Name
Embedded driver	org.apache.derby.jdbc.EmbeddedDriver
Network client driver	org.apache.derby.jdbc.ClientDriver

To use : Eg:

- DriverManager.registerDriver (new org.apache.derby.jdbc.EmbeddedDriver());
- Class.forName("org.apache.derby.jdbc.EmbeddedDriver");

If I use Embedded database, in my Java application I will have to say create=true and then actually create database, table etc…

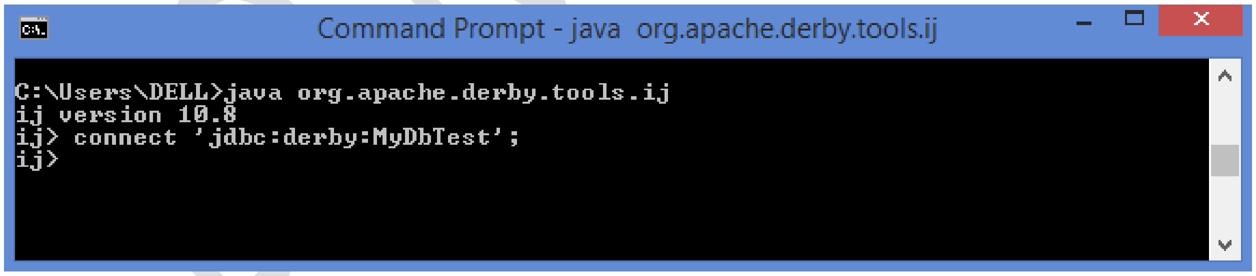
List the contents of your directory using dir command. The database is created in same path where your commands began



Connect to a database

Start up ij again and connect to the database you just created **(don't use create=true anymore)**

Ij> connect 'jdbc:derby:MyDbTest';



```
C:\Users\DELL>java org.apache.derby.tools.ij
ij version 10.8
ij> connect 'jdbc:derby:MyDbTest';
ij>
```

Internally, ij determines by default which driver to load from the protocol ("jdbc:derby:"). In this case, it knows to load the embedded JDBC driver. We could also have specified the protocol with a property as shown below:

```
java -Dij.protocol=jdbc:derby: org.apache.derby.tools.ij

ij> connect 'MyDbTest';
```

DO NOT COPY

Database

Connecting to the MyDbTest database in the connection URL above works because the MyDbTest database directory is in the current working directory; *i.e.*, the directory where you started up ij.

Let's say that your current directory location is /home/bill/databases and that you decide to change your directory to a different place entirely. You can connect to the MyDbTest database by specifying the complete directory path, like this:

java org.apache.derby.tools.ij ij> connect 'jdbc:derby:/home/bill/databases/MyDbTest';

You could also specify the Derby system home for the database like this:

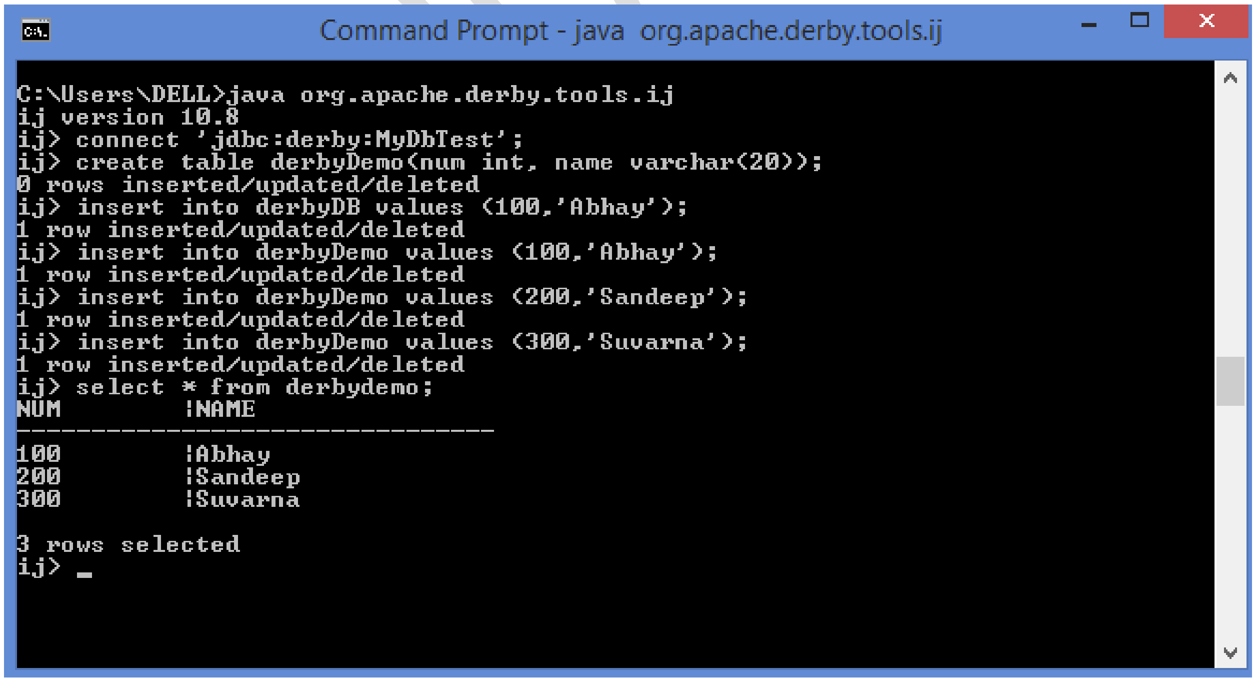
java org.apache.derby.tools.ij -Dderby.system.home=/home/bill/databases ij> connect 'jdbc:derby:MyDbTest';

COPY

Execute SQL statements

Once you connect to a database, you can execute SQL statements. ij expects each statement to be terminated with a semicolon (;); for example:

```
ij> create table derbyDB(num int, addr varchar(40));
ij> insert into derbyDB values (1956,'Webster St.');
ij> insert into derbyDB values (1910,'Union St.');
ij> update derbyDB set num=180, addr='Grand Ave.' where num=1956;
ij> select * from derbyDb;
```



```
C:\Users\DELL>java org.apache.derby.tools.ij
ij version 10.8
ij> connect 'jdbc:derby:MyDbTest';
ij> create table derbyDemo(num int, name varchar(20));
0 rows inserted/updated/deleted
ij> insert into derbyDB values (100,'Abhay');
1 row inserted/updated/deleted
ij> insert into derbyDemo values (100,'Abhay');
1 row inserted/updated/deleted
ij> insert into derbyDemo values (200,'Sandeep');
1 row inserted/updated/deleted
ij> insert into derbyDemo values (300,'Suvarna');
1 row inserted/updated/deleted
ij> select * from derbydemo;
NUM        |NAME
--------------------------------
100        |Abhay
200        |Sandeep
300        |Suvarna

3 rows selected
ij> _
```

Disconnect from a database

The disconnect command disconnects from the current database:

```
ij> disconnect;
```

Run SQL Scripts

You can execute SQL scripts in ij as shown below:

```
ij> run 'my_file.sql';
```

You can also run SQL scripts from the command line:

```
java org.apache.derby.tools.ij my_file.sql
```

DO NOT COPY