Java Database Connectivity JDBC – Part 1

FSAD/SWW2 Week 10

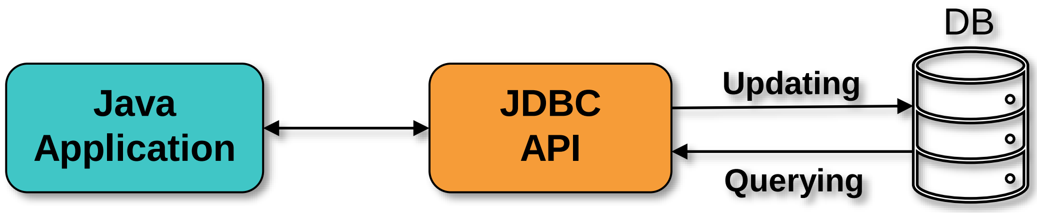
Ana Stroescu

Contents

UNIVERSITY OF BIRMINGHAM

Introduction

- JDBC is a Java Database Connectivity technology.
- This technology is an API for the Java programming language that defines how a client may access a database.
- It provides methods for querying and updating data in a database.
- JDBC is oriented towards relational databases.

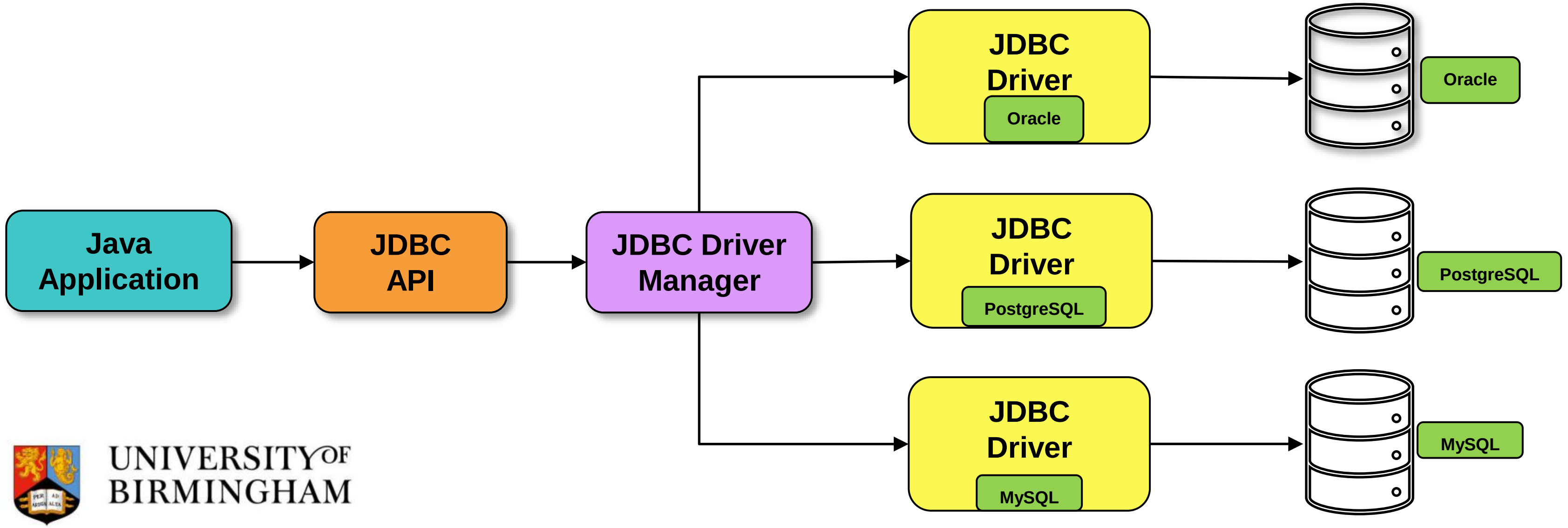


UNIVERSITY OF BIRMINGHAM

JDBC Basic Architecture

In general, JDBC architecture consists of two layers:

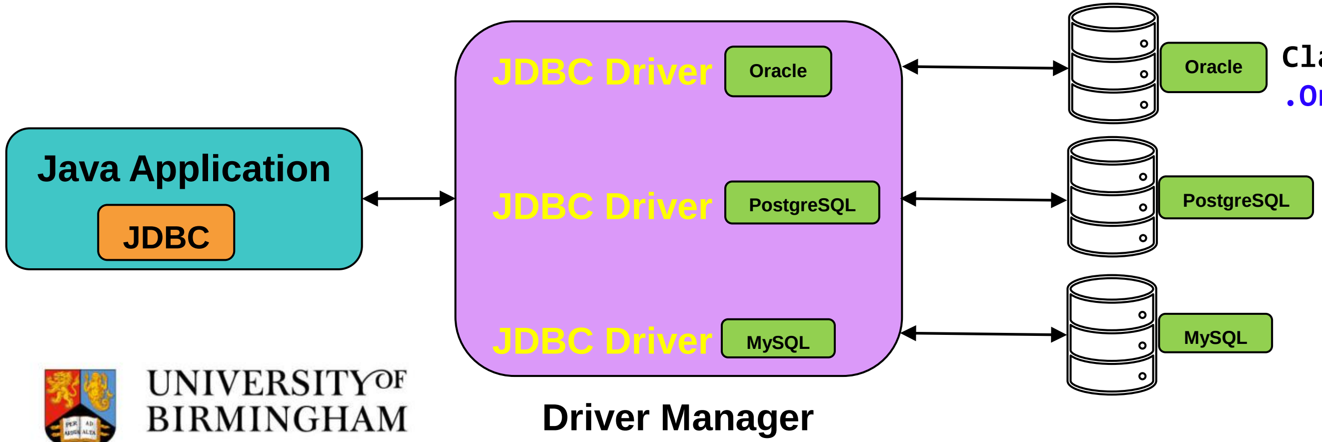
1. JDBC API – provides an application-to-JDBC Manager Connection;
2. JDBC Driver API – supports the JDBC Manager-to-Driver Connection.



UNIVERSITY OF BIRMINGHAM

Driver Manager

- It is the basic service for managing a set of JDBC drivers;
- It is used to match the connection request from a Java application with a proper database driver using the communication sub-protocol;
- The first driver that recognises a certain sub-protocol under JDBC will be used to establish a database connection.



```
Class.forName(“oracle.jdbc.driver
.OracleDriver");
```

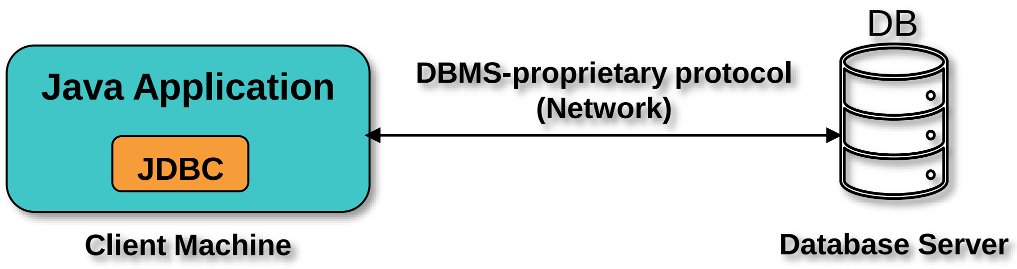
```
Class.forName("org.postgresql
.Driver");
```

```
Class.forName(“com.mysql.jdbc
.Driver");
```

UNIVERSITY OF BIRMINGHAM

JDBC Architecture – 2 tier

- JDBC supports *Two Tier* and *Three Tier* processing models for database access.



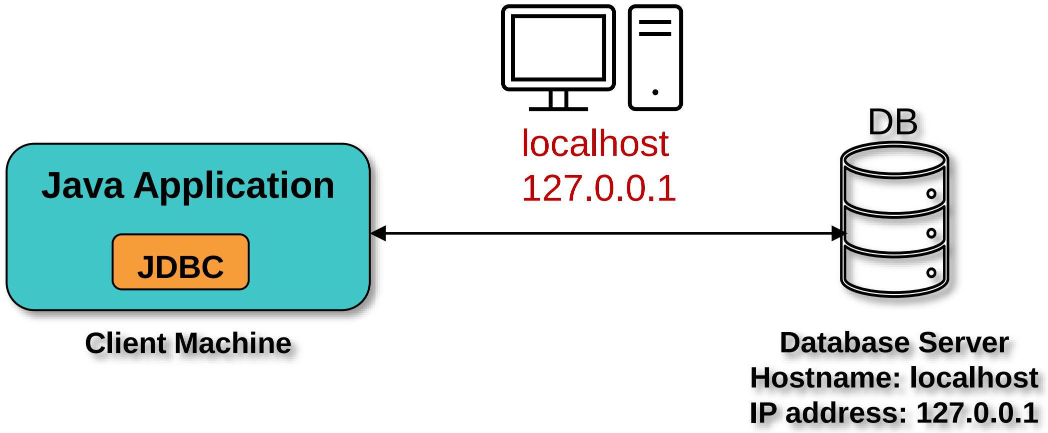
2-tier architecture

- A Java application talks directly to the data source.
- The database may be located on another machine to which the user is connected via network (client-server configuration).
- A JDBC driver is deployed on the client machine.

UNIVERSITY OF BIRMINGHAM

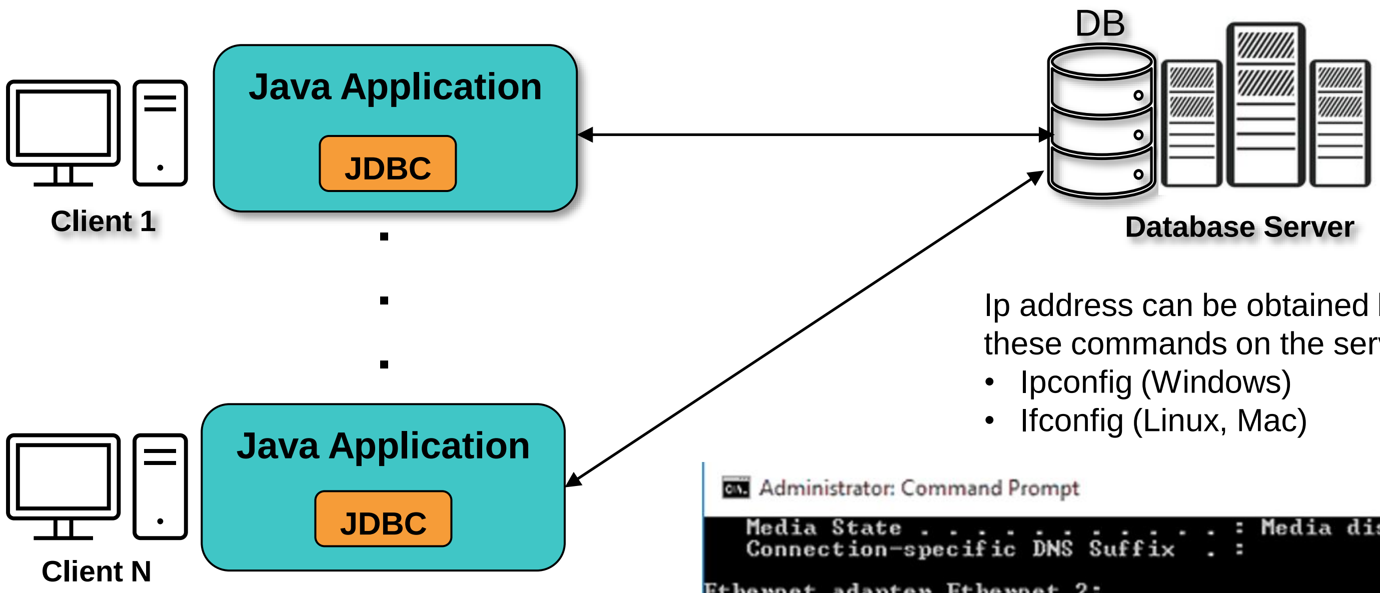
Traditional client/server application

- Scenario 1: Database server is running on a local machine.



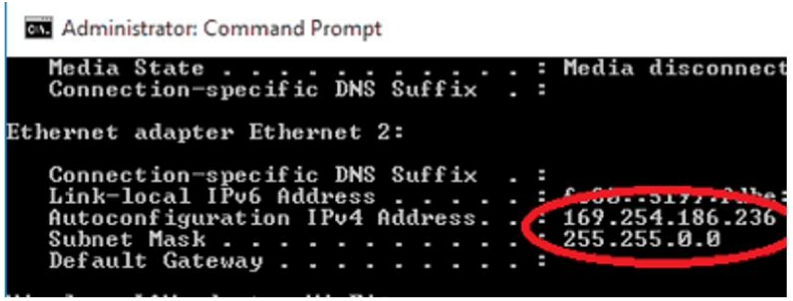
UNIVERSITY OF BIRMINGHAM

- Scenario 2: Database server is running on an external server.



Ip address can be obtained by running these commands on the server machine:

- Ipconfig (Windows)
- Ifconfig (Linux, Mac)

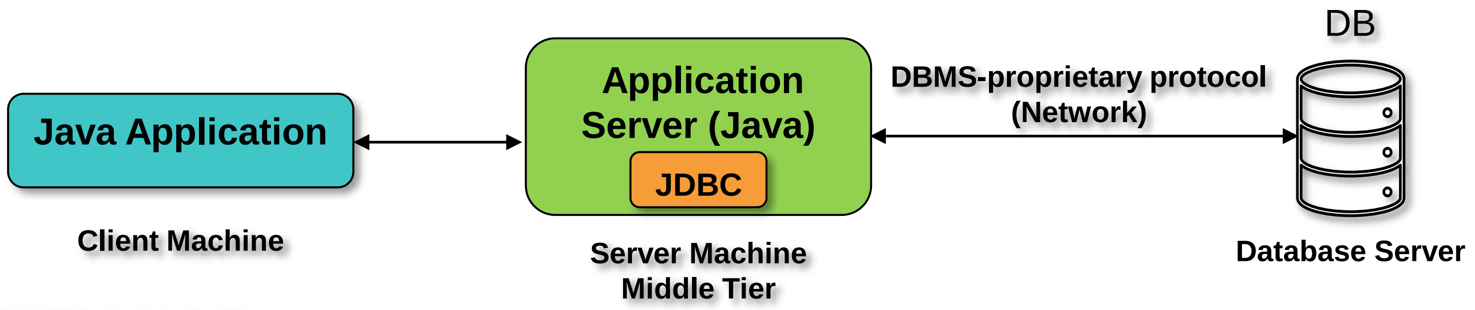


```
   Media State . . . . . . . . . . . : Media disconnect
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : [illegible]
   Autoconfiguration IPv4 Address. . : 169.254.186.236
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . :
```

UNIVERSITY OF BIRMINGHAM

JDBC Architecture – 3 tier

- Commands are sent to a middle tier of services, which then sends them to the database server.
- The database server processes the commands and sends the result back to the middle tier, which then sends them to the user.
- Many organisations find the 3-tier very attractive because it does not give direct access to the database for the client machine and it facilitates separation of concerns.

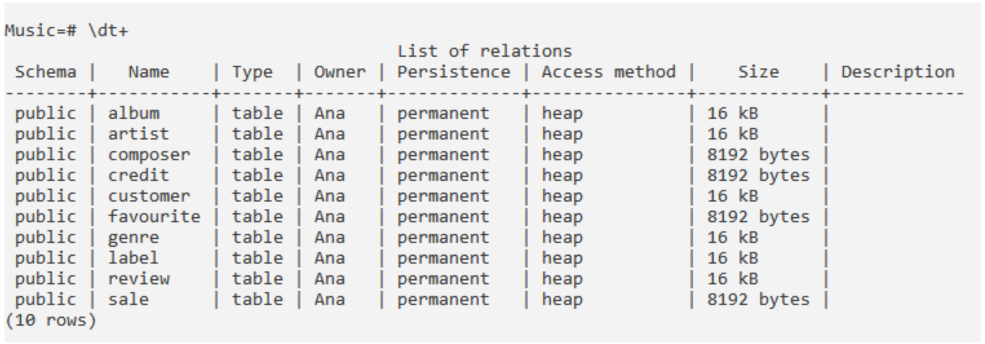


3-tier architecture

UNIVERSITY OF BIRMINGHAM

Prerequisites

- PostgreSQL
- The ‘Music’ database
- Java IDE
- JDBC driver for PostgreSQL (Jar file)



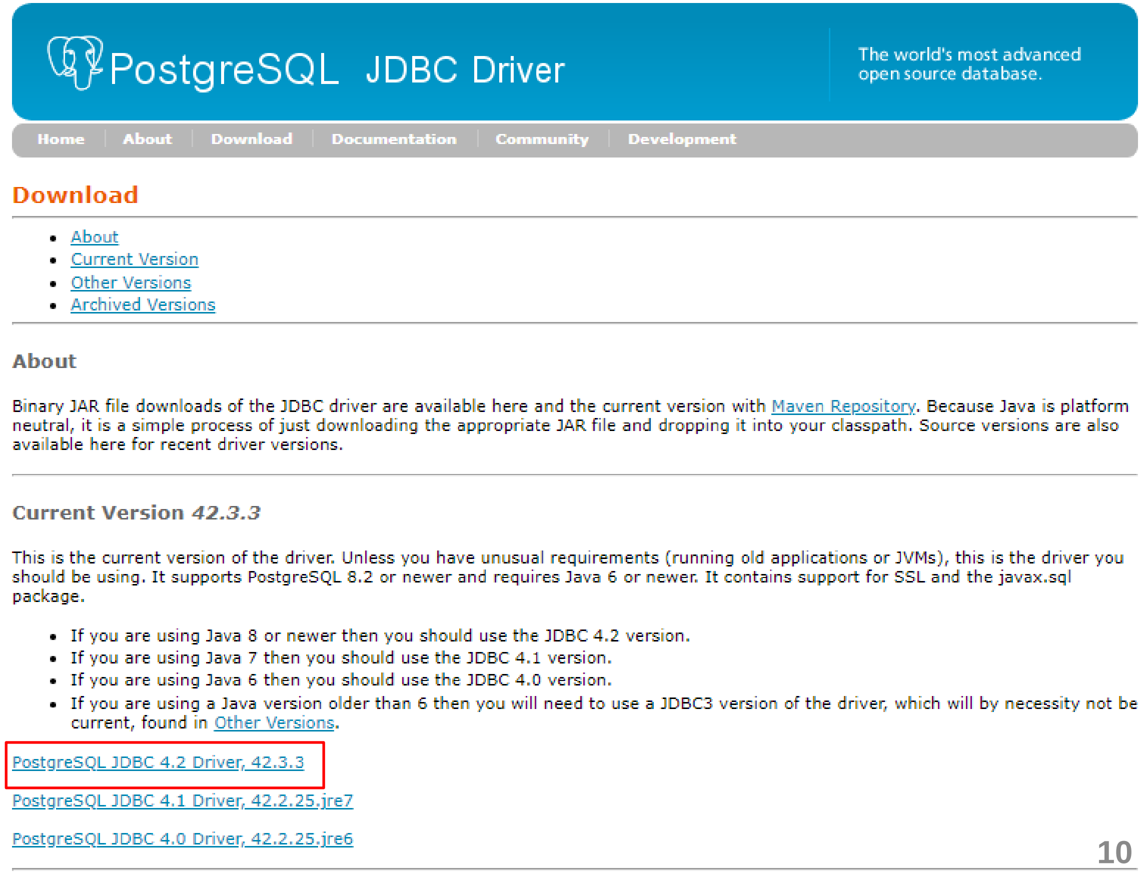
```
Music=# \dt+
                                        List of relations
 Schema |    Name    | Type  | Owner | Persistence | Access method |    Size    | Description
--------+------------+-------+-------+-------------+---------------+------------+-------------
 public | album      | table | Ana   | permanent   | heap          | 16 kB      |
 public | artist     | table | Ana   | permanent   | heap          | 16 kB      |
 public | composer   | table | Ana   | permanent   | heap          | 8192 bytes |
 public | credit     | table | Ana   | permanent   | heap          | 8192 bytes |
 public | customer   | table | Ana   | permanent   | heap          | 16 kB      |
 public | favourite  | table | Ana   | permanent   | heap          | 8192 bytes |
 public | genre      | table | Ana   | permanent   | heap          | 16 kB      |
 public | label      | table | Ana   | permanent   | heap          | 16 kB      |
 public | review     | table | Ana   | permanent   | heap          | 16 kB      |
 public | sale       | table | Ana   | permanent   | heap          | 8192 bytes |
(10 rows)
```

UNIVERSITY OF BIRMINGHAM

JDBC driver for PostgreSQL

Download the driver from

https://jdbc.postgresql.org/download.html



Download

- About
- Current Version
- Other Versions
- Archived Versions

About

Binary JAR file downloads of the JDBC driver are available here and the current version with Maven Repository. Because Java is platform neutral, it is a simple process of just downloading the appropriate JAR file and dropping it into your classpath. Source versions are also available here for recent driver versions.

Current Version *42.3.3*

This is the current version of the driver. Unless you have unusual requirements (running old applications or JVMs), this is the driver you should be using. It supports PostgreSQL 8.2 or newer and requires Java 6 or newer. It contains support for SSL and the javax.sql package.

- If you are using Java 8 or newer then you should use the JDBC 4.2 version.
- If you are using Java 7 then you should use the JDBC 4.1 version.
- If you are using Java 6 then you should use the JDBC 4.0 version.
- If you are using a Java version older than 6 then you will need to use a JDBC3 version of the driver, which will by necessity not be current, found in Other Versions.

PostgreSQL JDBC 4.2 Driver, 42.3.3

PostgreSQL JDBC 4.1 Driver, 42.2.25.jre7

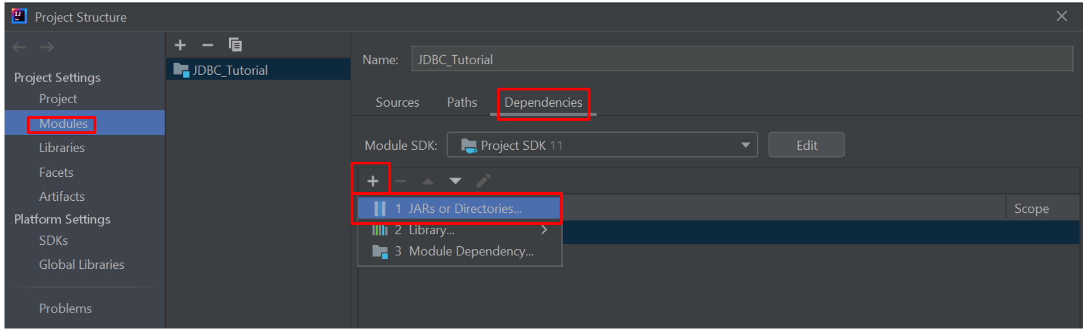
PostgreSQL JDBC 4.0 Driver, 42.2.25.jre6

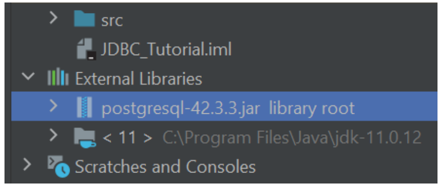
UNIVERSITY OF BIRMINGHAM

Configure JDBC driver in IntelliJ (Community Edition)

1. Open IntelliJ and create a new Java project.
2. Go to `File |Project Structure | Modules | Dependencies`
3. Click on `+ | JARs or Directories…`
4. Select the downloaded JDBC driver jar file `postgresql-42.3.3.jar`

- Alternatively, watch the video on Canvas.




src
JDBC_Tutorial.iml
External Libraries
postgresql-42.3.3.jar library root
< 11 > C:\Program Files\Java\jdk-11.0.12
Scratches and Consoles

Newly added JDBC driver is visible in the workspace

UNIVERSITY OF BIRMINGHAM

If you use IntelliJ Ultimate edition, follow the instructions on this link: https://www.jetbrains.com/help/idea/postgresql.html

Steps in Database connectivity

- The fundamental steps involved in the process of connecting to a database and executing a query consists of the following:

1. Import JDBC packages;
2. Load and register the JDBC driver;
3. Open a connection to the database;
4. Querying the Database;
5. Process the results of a query;
6. Close the `ResultSet` and `Statement` objects;
7. Close the connection.

UNIVERSITY OF BIRMINGHAM

Step 1. Import JDBC packages

- This is for making the JDBC API classes immediately available to the application program.
- The following import statement should be included in the program irrespective of the JDBC driver being used:

```
import java.sql.*;
```

Step 2. Load and register the JDBC driver

- This is for establishing a communication between the JDBC program and the database.
- `forName()` method of the `java.lang.Class` class can be used to load and register the JDBC driver:

```
Class.forName("org.postgresql.Driver");
```

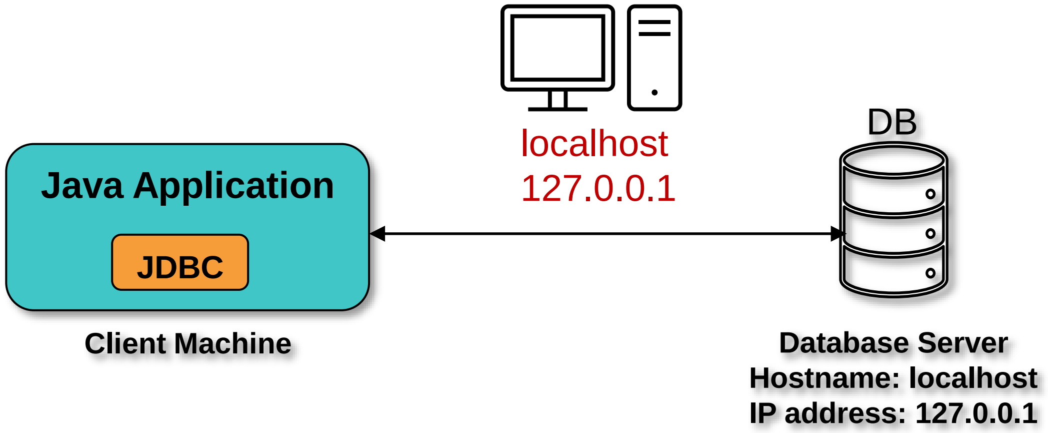
Note: This step can be skipped, as from JDBC 4 onwards, applications no longer need to explicitly load JDBC drivers using `Class.forName()`. Existing programs which currently load JDBC drivers using `Class.forName()` will continue to work without modifications.

Step 3. Open a connection to database

- Once the required packages have been imported and the JDBC driver has been loaded and registered, a database connection must be established.
- This is done by using the `getConnection()` method of the `DriverManager` class.
- The `getConnection()` method takes three parameters of type String: URL, username and password.

```
Connection con = DriverManager.getConnection(URL, username, password);
```

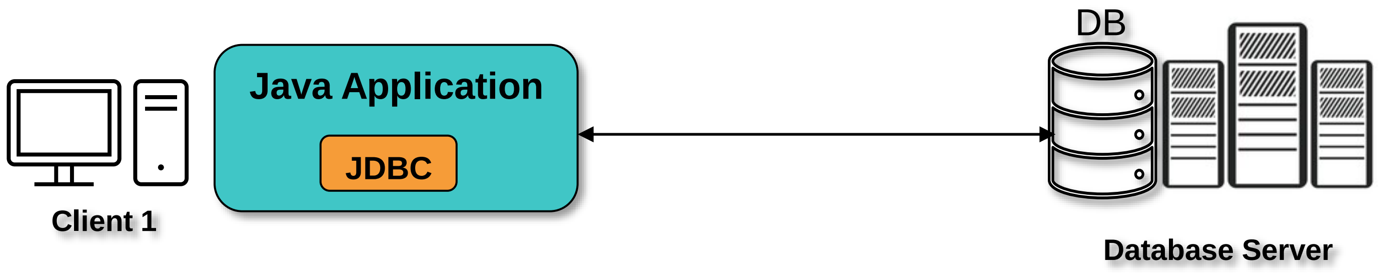
- Scenario 1: Database server is running on a local machine.



Example URL: String *URL* = "jdbc:postgresql://localhost:5432/Music“;

Local machine | Default port for PostgreSQL | Name of databse

- Scenario 2: Database server is running on an external server



Example URL: String *URL* = "jdbc:postgresql://<Insert Server IP here>:5432/Music“;

Ip address can be obtained by running these commands on the server machine:
- Ipconfig (Windows)
- Ifconfig (Linux, Mac)

Default port for PostgreSQL

Name of databse

UNIVERSITY OF BIRMINGHAM

Step 4. Querying the database

- Querying the database involves two steps:

1. Creating a `Statement` object to perform a query:

```
Statement stmt = con.createStatement();
```

There are other interfaces for executing SQL statements and returning the results, such as `PreparedStatement`, discussed later.

2. Executing the query and returning a `ResultSet`:

```
String sql = "SELECT * FROM album";
ResultSet rs = stmt.executeQuery(sql);
```

A `ResultSet` object is a table of data representing a database result set, which is usually generated by executing a `Statement` that queries the database.

`stmt.executeQuery()` has a String argument containing the text of an SQL SELECT query.

UNIVERSITY OF BIRMINGHAM

! For SQL DML (Data Manipulation Language) statements, such as INSERT, UPDATE or DELETE, `stmt.executeUpdate()` is used:

```
String sql = “INSERT INTO label VALUES ('Warner', 'New York', 'USA')";
int count = stmt.executeUpdate(sql);
```

This returns and integer `count` representing the number of rows updated. The next step “Step 5. Process the results of a query” can be skipped.

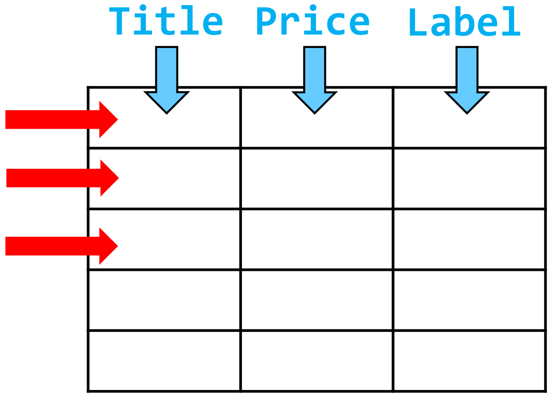
- `stmt.execute()` can also be used to execute an arbitrary SQL statement which may be of any type or a stored procedure. This may return multiple results, so extracting the results is less convenient.

UNIVERSITY OF BIRMINGHAM

Step 5. Process the results of a query

- Once the query has been executed, there are two steps to be carried out:
  1. Processing the output `ResultSet` to fetch the rows;
  2. Retrieving the column values of the current row.

```
while (rs.next()){
      System.out.print("Title: " + rs.getObject("title")
         + " Price: " + rs.getObject("price")
         + " Label: " + rs.getObject("label") + "\n“);}
```



The `getObject` method takes a String argument containing the column name.

!

- There are also type specific methods, such as `getInt, getString`, etc. However, the major disadvantage is that if a field is null in the database, the value returned by these methods will not be null.
- For `getObject` method, if the field is null then the object value returned will be null.

```
while (rs.next()){
        System.out.print("Title: " + rs.getString("title")
            + " Price: " + rs.getInt("price")
            + " Label: " + rs.getString("label") + "\n“);}
```

If a price field is null in the database, then the value returned by `getInt` will be 0.

UNIVERSITY OF BIRMINGHAM

Check out the `java.sql` documentation here:
https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html

Step 6. Close the `ResultSet` and `Statement` objects

- Once the `ResultSet` and `Statement` objects have been used, they must be closed explicitly;
- This is done by calls to the `close()` method of `ResultSet` and `Statement` classes.

```
rs.close();

stmt.close();
```

UNIVERSITY OF BIRMINGHAM

Step 7. Close the connection

- The last step is to close the database connection:

```
con.close();
```

UNIVERSITY OF BIRMINGHAM

References

- Oracle Tutorial: Lesson: JDBC Basics
- PostgreSQL JDBC Driver Documentation
- PostgreSQL JDBC Tutorial
- Package java.sql