
JDBC
Lesson 1 : Introduction
Capgemini

Lesson Objectives

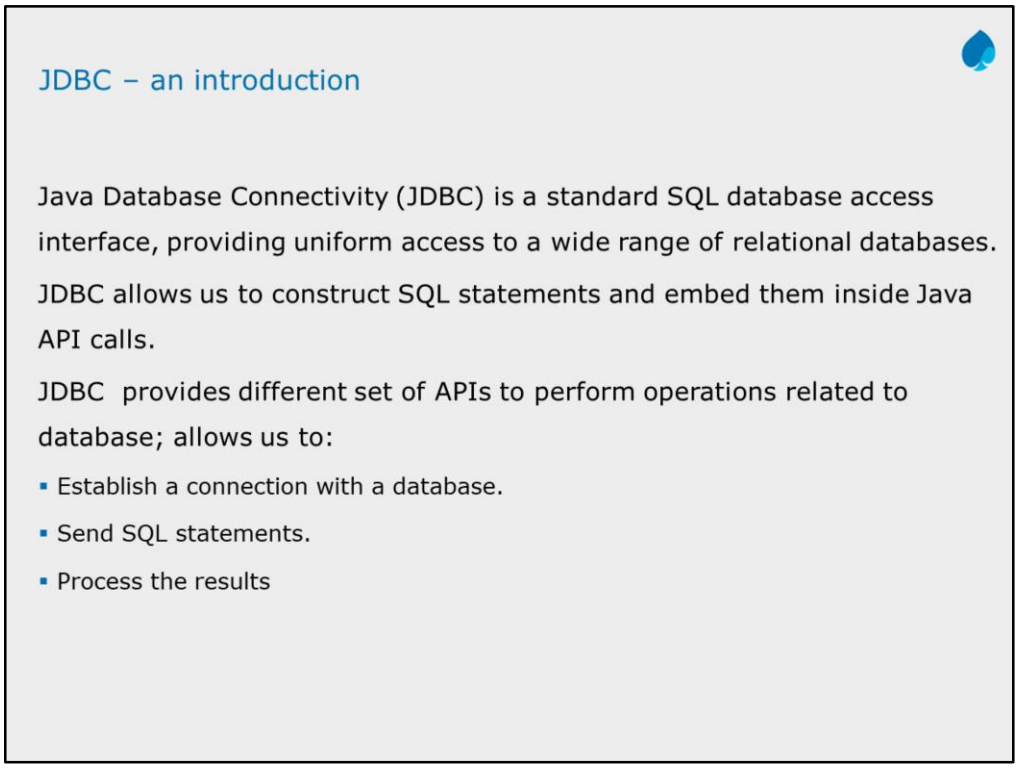
After completing this lesson, participants will be able to

- Understand concept database connectivity architecture
- Work with JDBC API 4.0
- Access database through Java programs
- Understand advance features of JDBC API

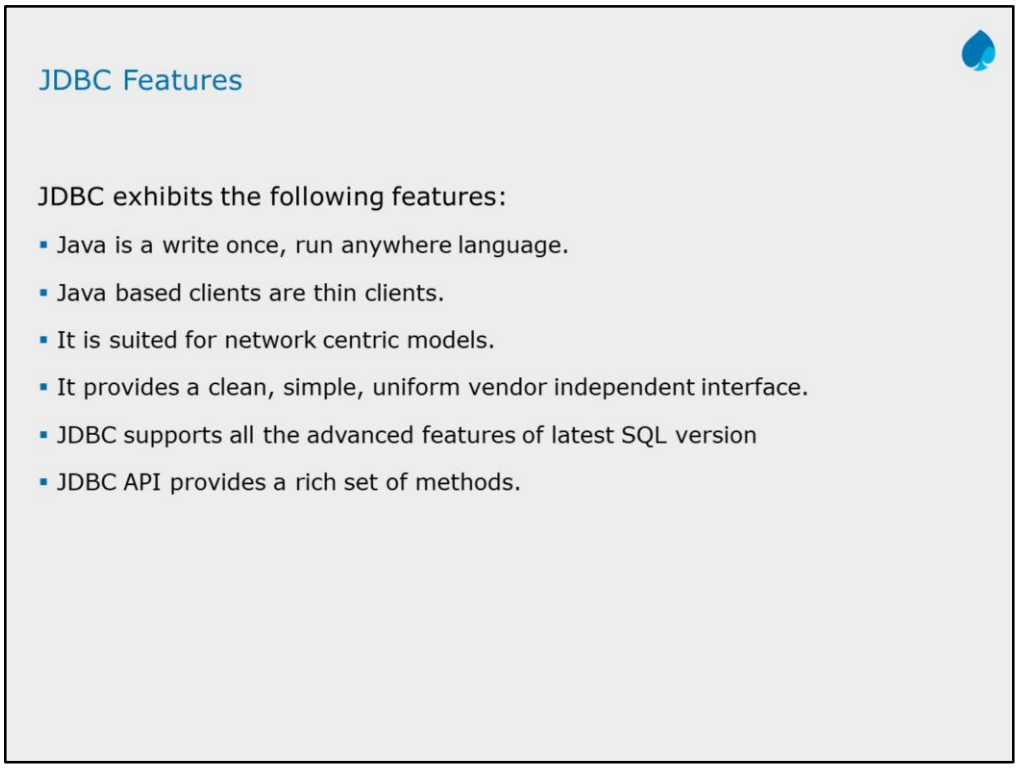
This lesson covers JDBC API, used to work with database.

Lesson outline:

1.1: Java Database Connectivity - Introduction
1.2: Database Connectivity Architecture
1.3: JDBC APIs
1.4: Database Access Steps
1.5: Calling database procedures/functions
1.6: Using Transaction
1.7: Best Practices



What is JDBC?
JDBC is used to allow Java applications to connect to the database and perform different data manipulation operations such as insertion, modification, deletion, and so on.



Why JDBC?

JDBC Features:

With JDBC technology, businesses are not locked in any proprietary architecture, and can continue to use their installed databases and access information easily – even if it is stored on different database management systems.

The combination of the Java API and the JDBC API makes application development easy and economical.

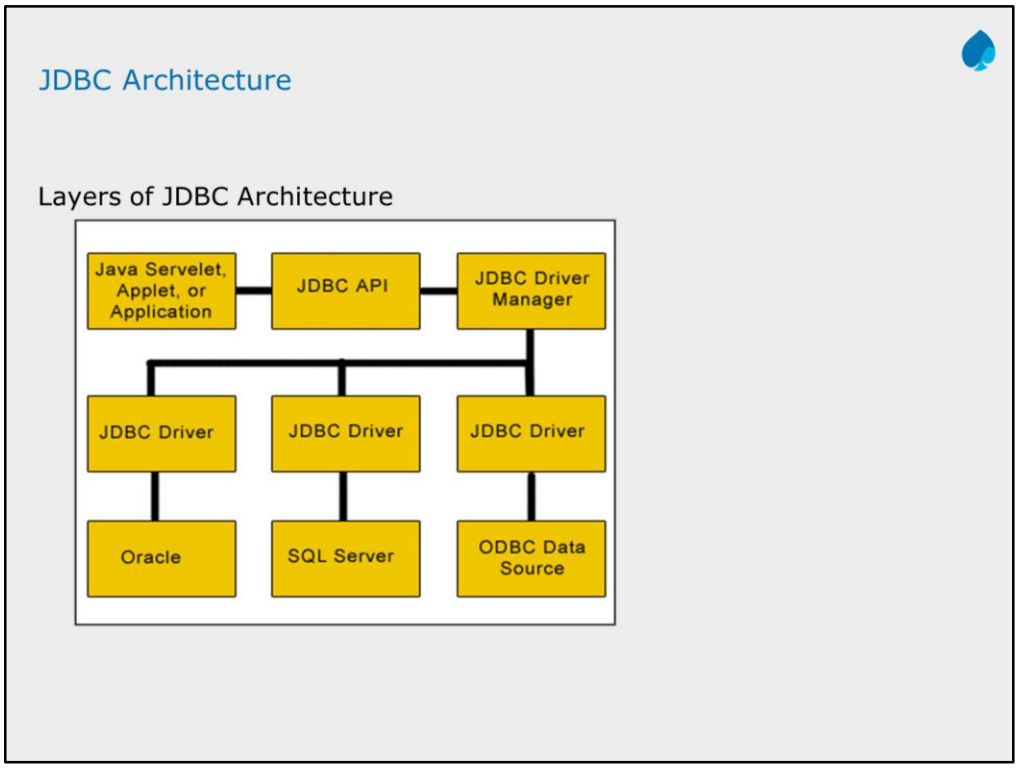
JDBC hides the complexity of many data access tasks, doing most of the “heavy lifting” for the programmer behind the scenes.

The JDBC API is simple to learn, easy to deploy, and inexpensive to maintain.

With the JDBC API, no configuration is required on the client side.

With a driver written in the Java programming language, all the information needed to make a connection is completely defined by the JDBC URL or by a DataSource object registered with a Java Naming and Directory Interface (JNDI) naming service.

Zero configuration for clients supports the network computing paradigm and centralizes software maintenance.



JDBC Architecture:

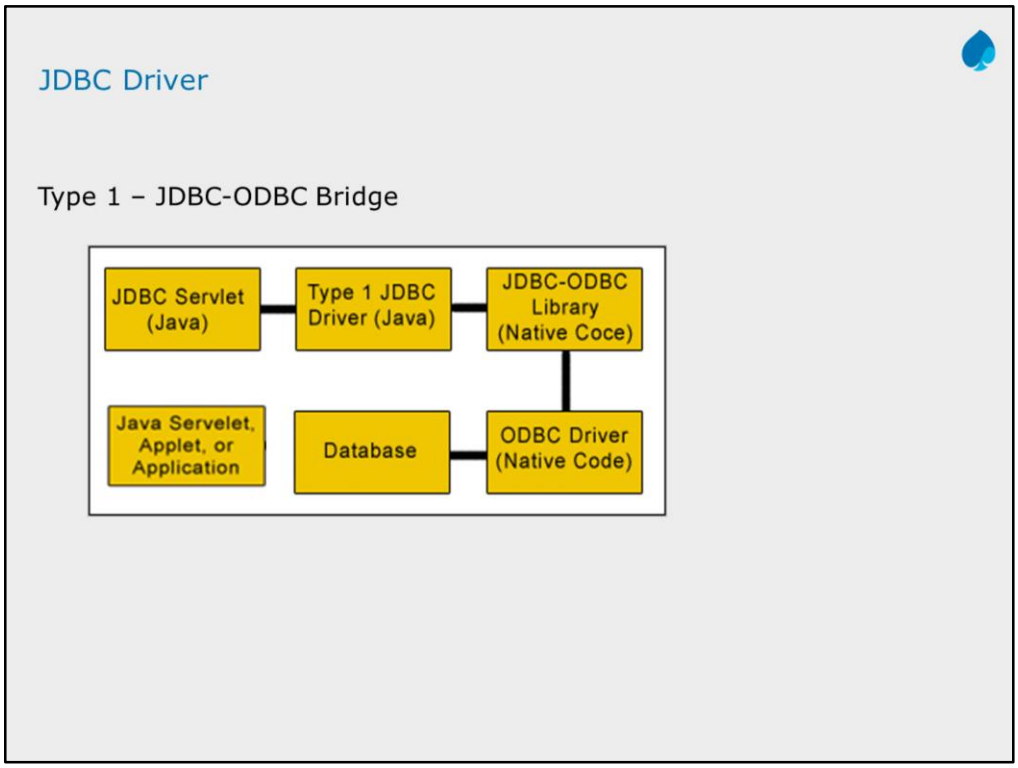
The JDBC Architecture can be classified as follows:

1. Type 1 – JDBC-ODBC Bridge
2. Type 2 – Java Native API
3. Type 3 – Java to Network Protocol
4. Type 4 – Java to Database Protocol

A JDBC driver translates standard JDBC calls into a network or database protocol or into a database library API call that facilitates communication with the database.

This translation layer provides JDBC applications with database independence.

If the back-end database changes, then only the JDBC driver needs to be replaced with few code modifications required. There are four distinct types of JDBC drivers.



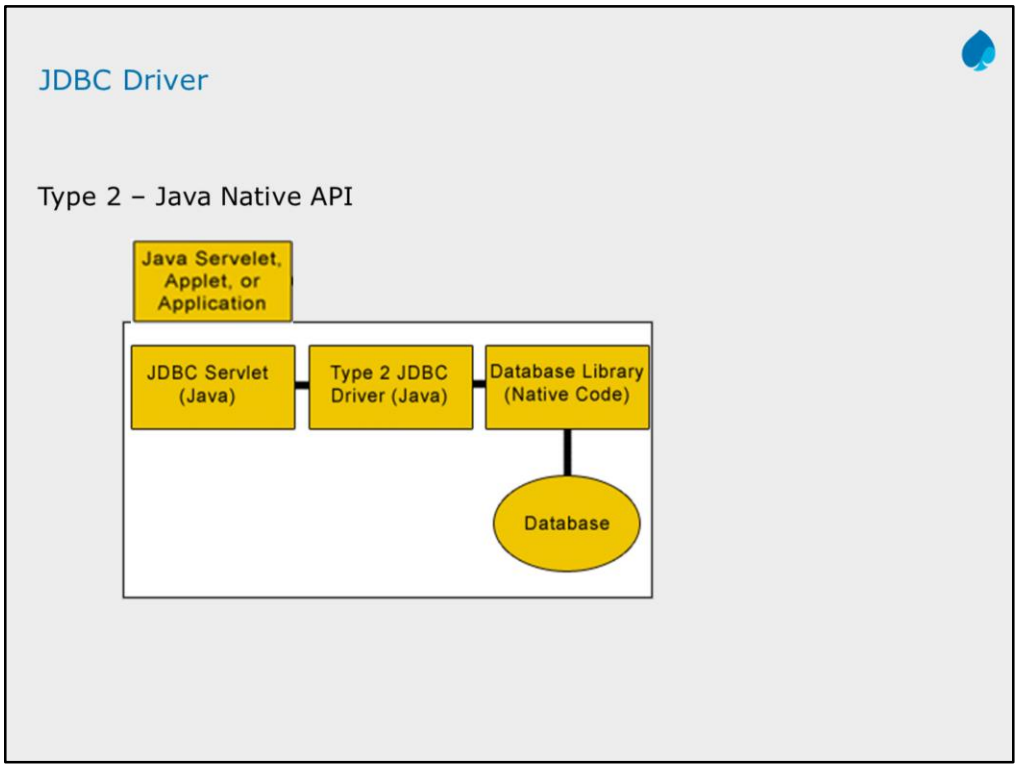
Type 1 JDBC-ODBC Bridge:

Type 1 drivers act as a "bridge" between JDBC and another database connectivity mechanism such as ODBC.

The JDBC- ODBC bridge provides JDBC access using most standard ODBC drivers.

This driver is included in the Java 2 SDK within the sun.jdbc.odbc package. In this driver the Java statements are converted to a JDBC statements.

JDBC statements call the ODBC by using the JDBC-ODBC Bridge. And finally the query is executed by the database. This driver has serious limitation for many applications.



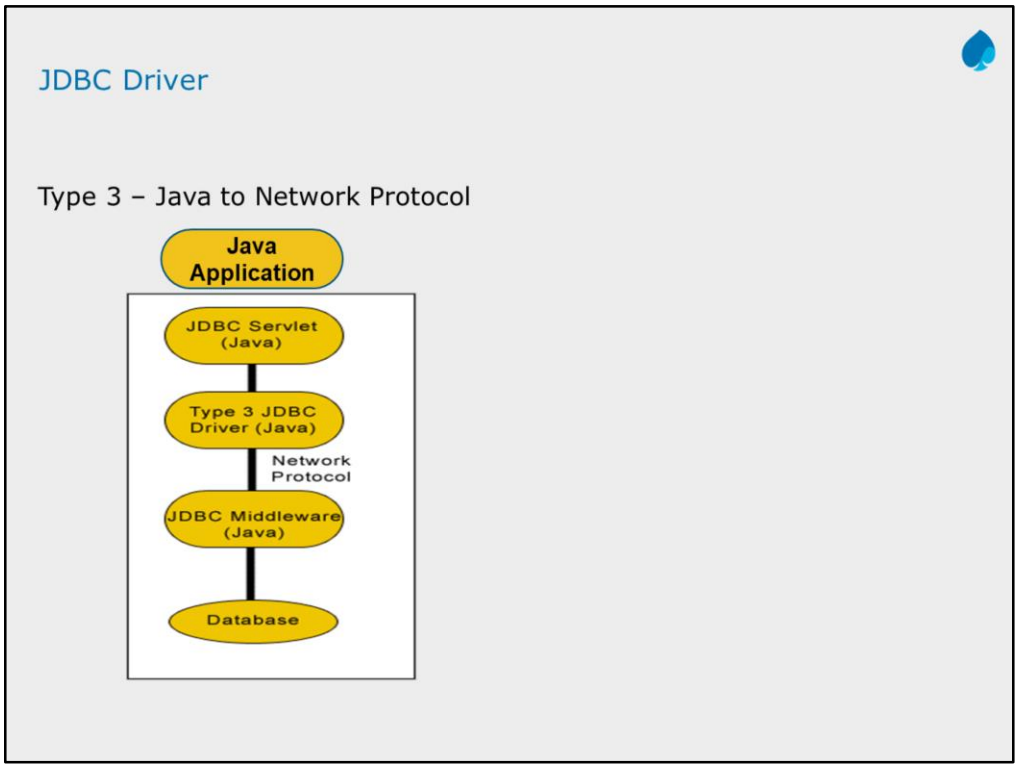
Type 2 Java to Native API:

Type 2 drivers use the Java Native Interface (JNI) to make calls to a local database library API.

This driver converts the JDBC calls into a database specific call for databases such as SQL, ORACLE, and so on. This driver communicates directly with the database server.

It requires some native code to connect to the database. Type 2 drivers are usually faster than Type 1 drivers.

Like Type 1 drivers, Type 2 drivers require native database client libraries to be installed and configured on the client machine.



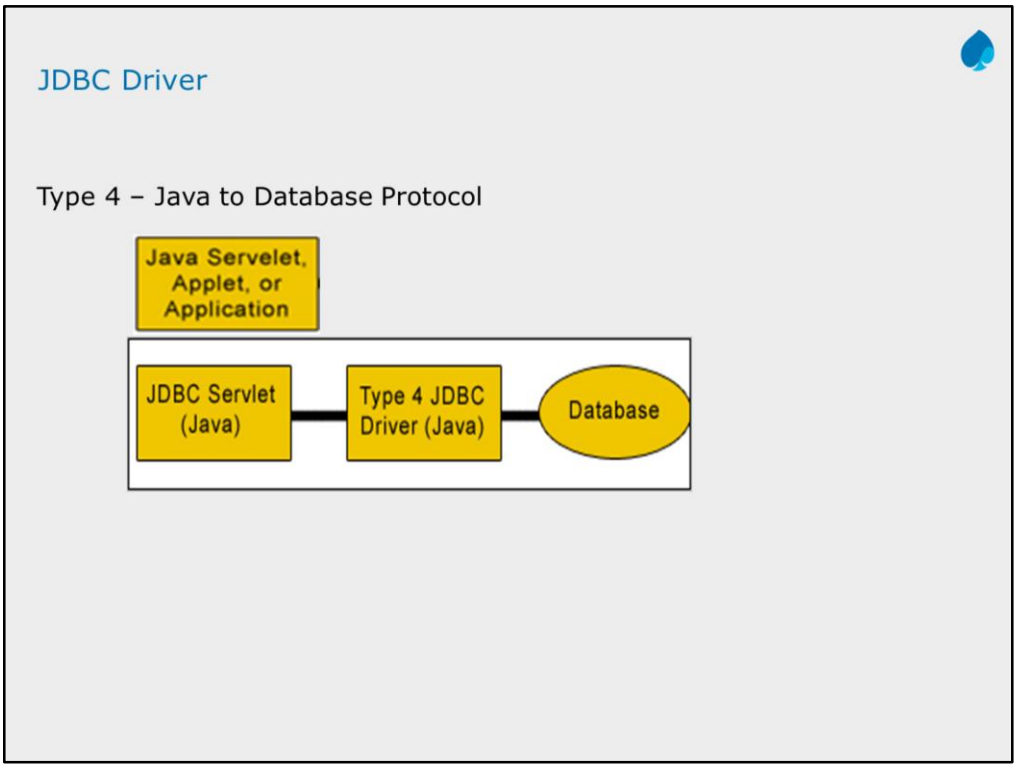
Type 3 Java to Network Protocol Or All- Java Driver:

Type 3 drivers are pure Java drivers that use a proprietary network protocol to communicate with JDBC middleware on the server.

The middleware then translates the network protocol to database-specific function calls.

Type 3 drivers are the most flexible JDBC solution because they do not require native database libraries on the client and can connect to many different databases on the back end.

Type 3 drivers can be deployed over the Internet without client installation.



Type 4 Java to Database Protocol:

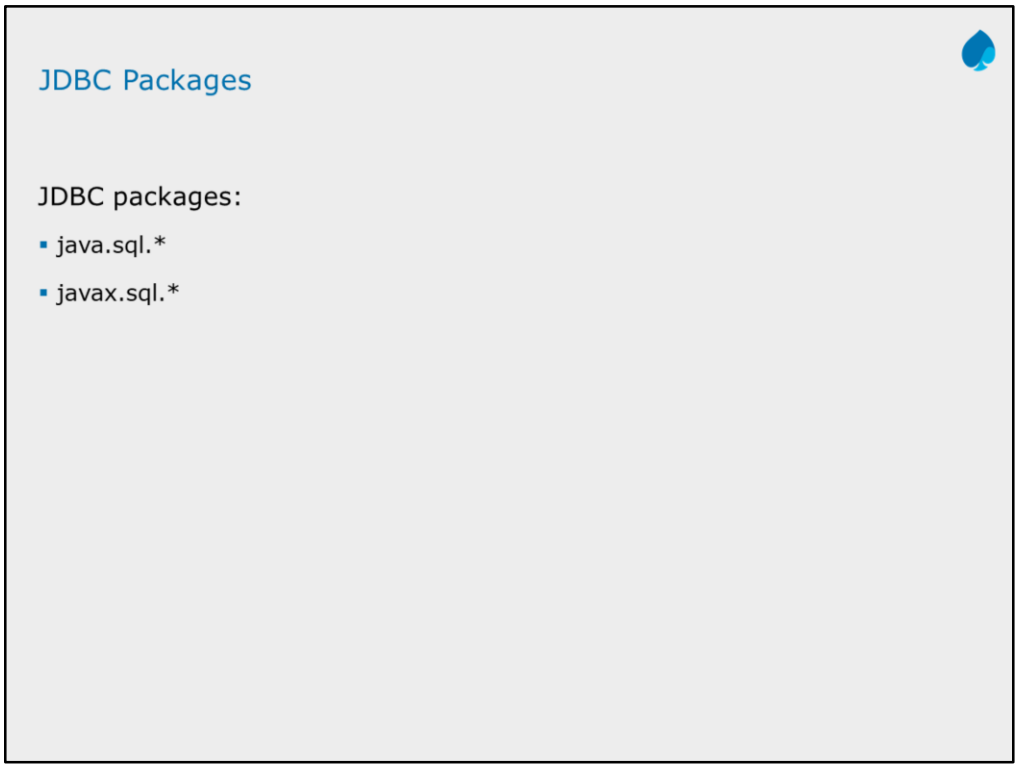
Type 4 drivers are pure Java drivers that implement a proprietary database protocol to communicate directly with the database.

Like Type 3 drivers, they do not require native database libraries and can be deployed over the Internet without client installation.

One drawback to Type 4 drivers is that they are database specific. Unlike Type 3 drivers, if your back-end database changes, you may save to purchase and deploy a new Type 4 driver (some Type 4 drivers are available free of charge from the database manufacturer).

However, since Type 4 drivers communicate directly with the database engine rather than through middleware or a native library, they are usually the fastest JDBC drivers available.

This driver directly converts the Java statements to SQL statements.



JDBC APIs:

The JDBC API provides universal data access from the Java programming language.

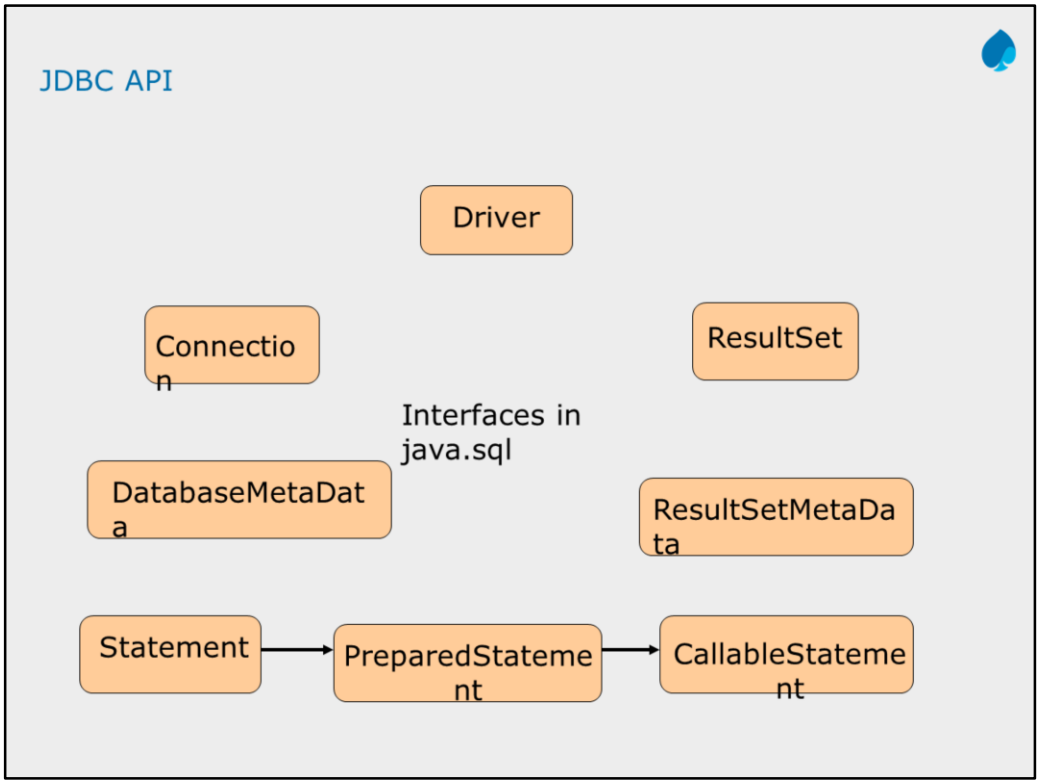
Using the JDBC 3.0 API, you can access virtually any data source, from relational databases to spreadsheets and flat files.

JDBC technology also provides a common base on which tools and alternate interfaces can be built.

The JDBC 3.0 API comprises of two packages:

1. java.sql package
2. javax.sql package

You automatically get both packages when you download the Java 2 Platform Standard Edition 5.0.



Java.sql package:

The **java.sql package** provides the API for accessing and processing data stored in a data source (usually a relational database) using the Java programming language.
This API includes a framework whereby different drivers can be installed dynamically to access different data sources.
Although the JDBC API is mainly geared to passing SQL statements to a database, it provides for reading and writing data from any data source with a tabular format.
The **reader/writer** facility, available through the **javax.sql.RowSet** group of interfaces, can be customized to use and update data from a spread sheet, flat file, or any other tabular data source.

Please note: Here we are not discussing about javax.sql package.

Summary

In this lesson, you have learnt:

JDBC architecture

Role of Driver Manager

Types of JDBC Drivers

Review Question