Java Platform: Working with Databases using JDBC

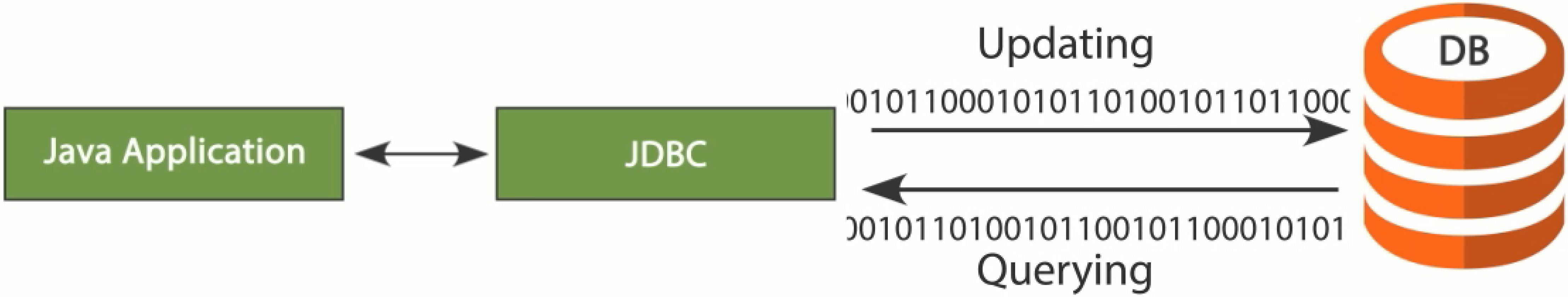
Sekhar Srinivasan
@sekharonline4u | sekhartheguru.net

Java Platform: Working with Databases using JDBC

- Introduction
- Getting Started With JDBC
- CRUD Operations Using JDBC
- Working with Stored Procedures
- Managing Transactions
- Working with BLOB and CLOB
- Working with Metadata
- Pooling Database Connections

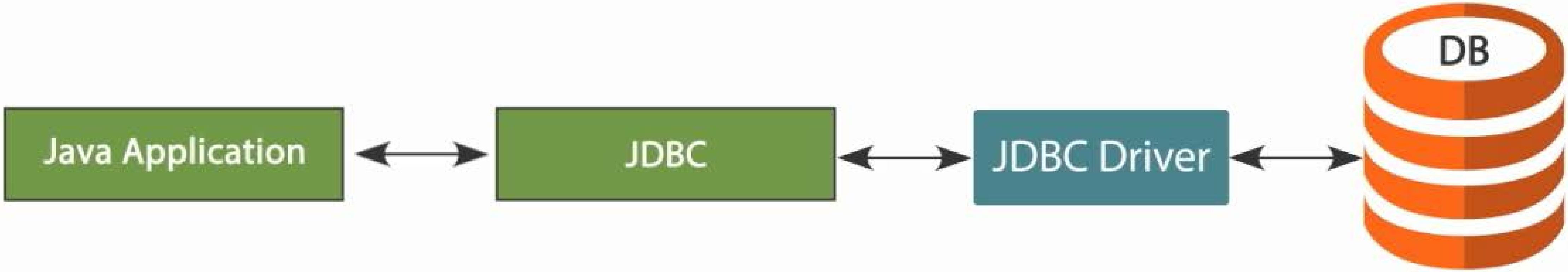
Introduction to JDBC

JDBC is an API for the Java programming language that defines how a client may access a Database

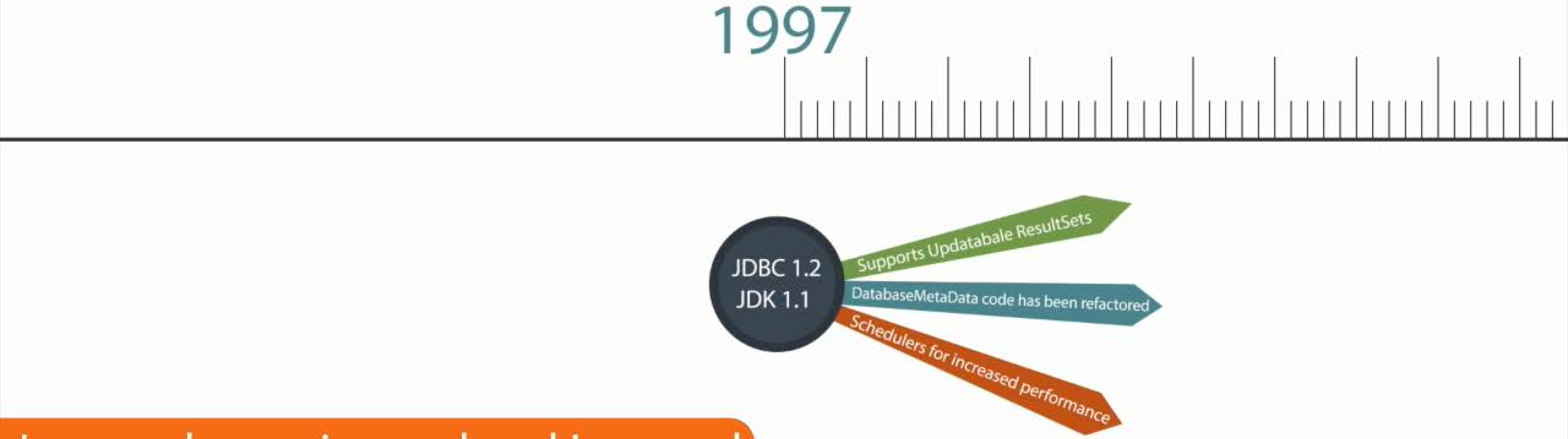


Introduction to JDBC

JDBC is an API for the Java programming language that defines how a client may access a Database



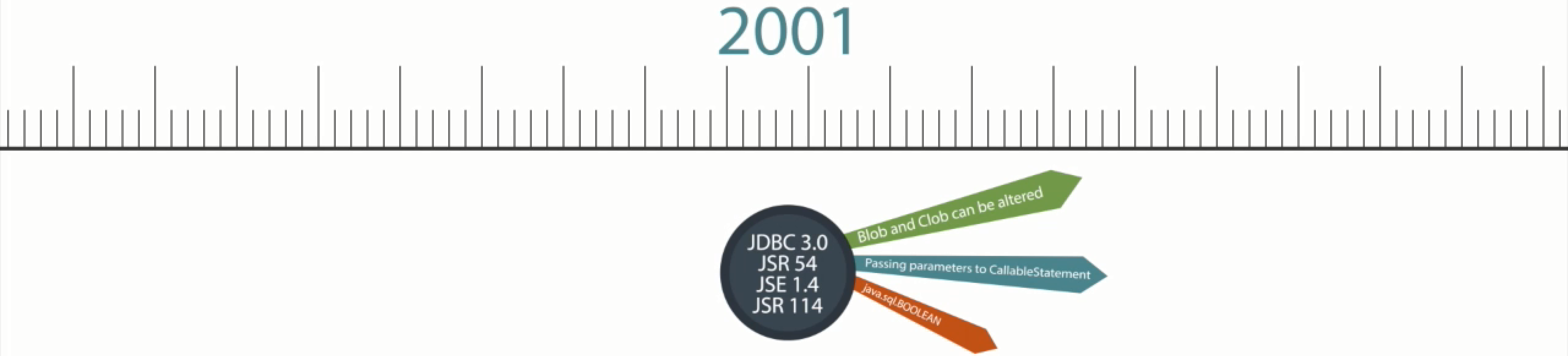
History



1997
JDBC 1.2
JDK 1.1
Supports Updatabale ResultSets
DatabaseMetaData code has been refactored
Schedulers for increased performance

Java package - java.sql and javax.sql

History



2001

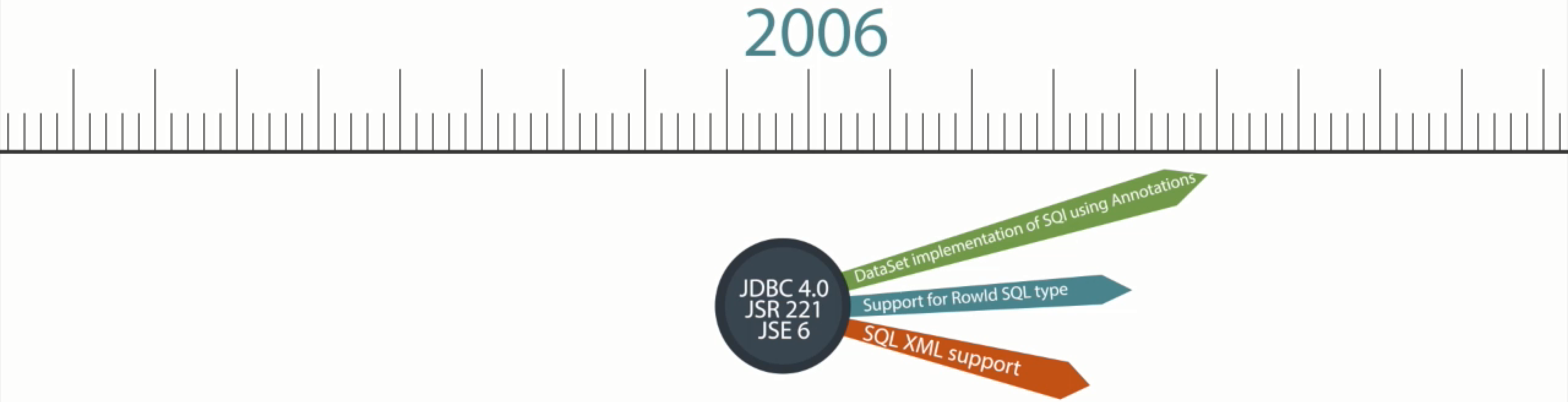
JDBC 3.0
JSR 54
JSE 1.4
JSR 114

Blob and Clob can be altered

Passing parameters to CallableStatement

java.sql.BOOLEAN

History



2006

JDBC 4.0
JSR 221
JSE 6

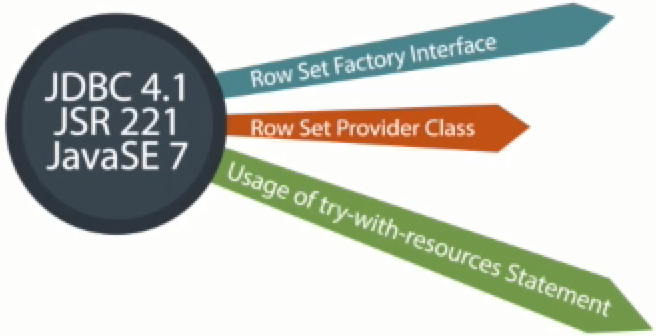
DataSet implementation of SQl using Annotations

Support for RowId SQL type

SQL XML support

History

2011



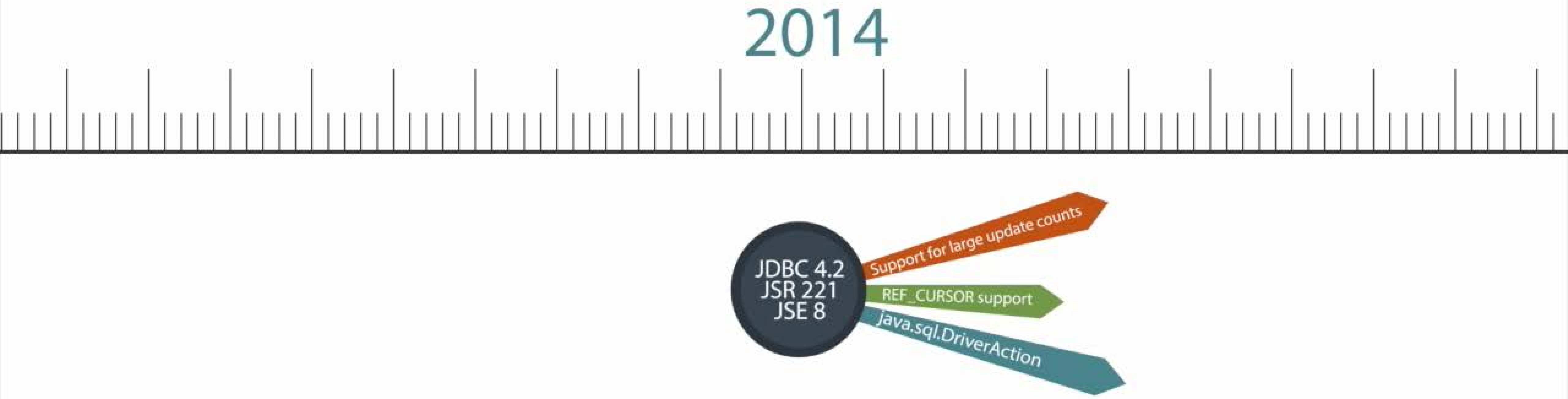
JDBC 4.1
JSR 221
JavaSE 7

Row Set Factory Interface

Row Set Provider Class

Usage of try-with-resources Statement

History



2014

JDBC 4.2
JSR 221
JSE 8

Support for large update counts

REF_CURSOR support

java.sql.DriverAction

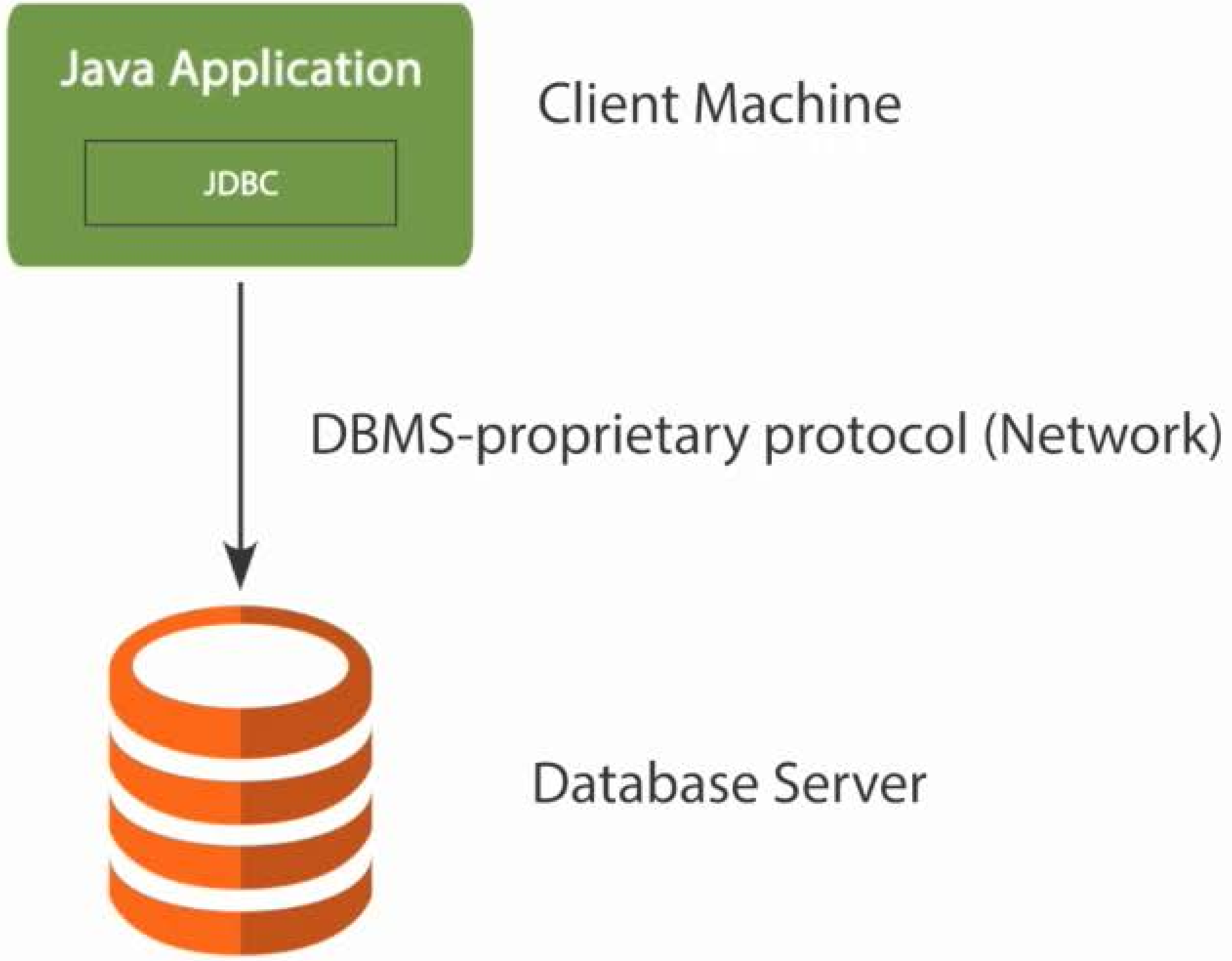
JSR - Java Specification Request

Architecture of JDBC

JDBC Supports

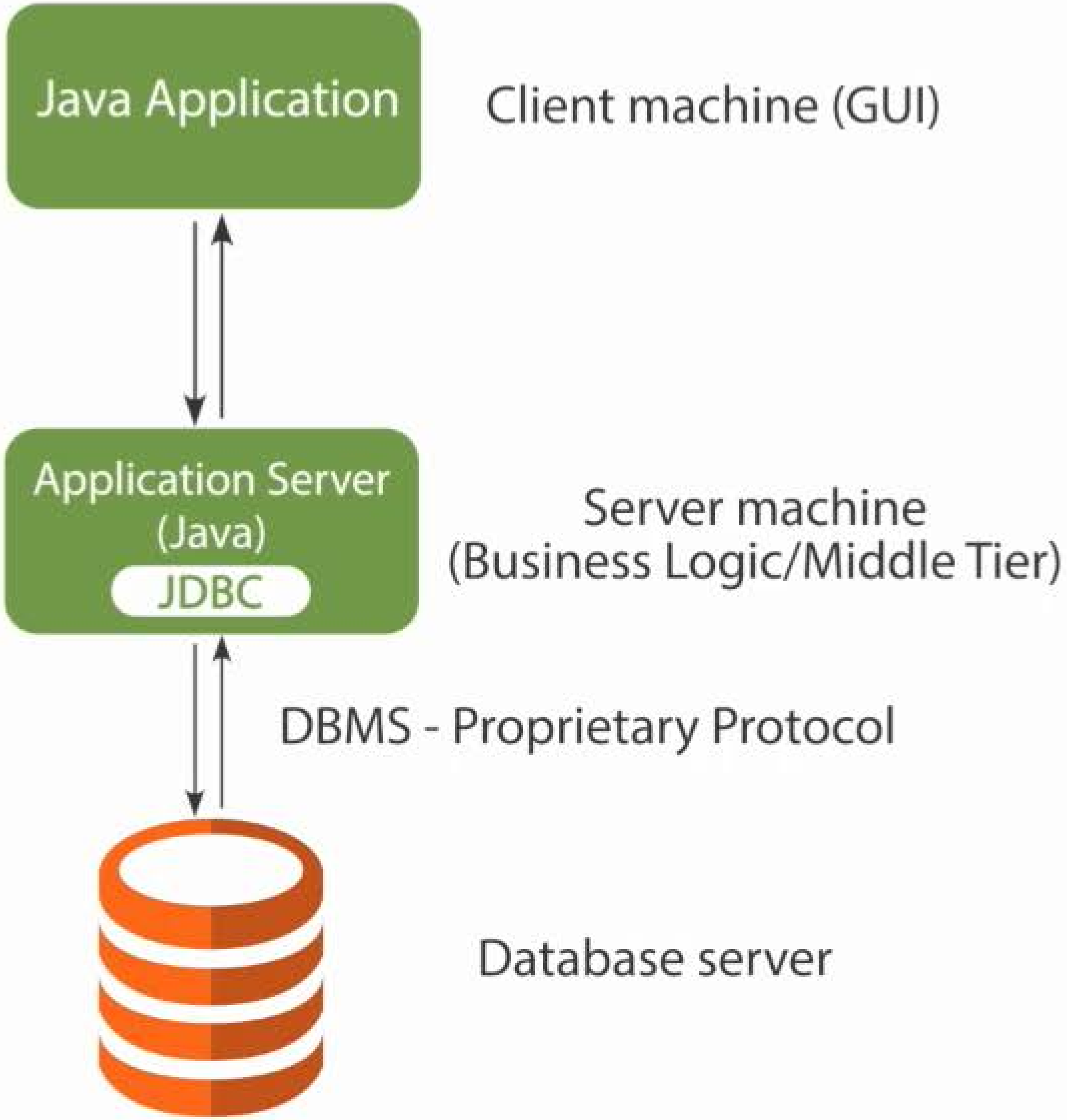
- Two Tier Architecture
- Three Tier Architecture

Architecture of JDBC



Two Tier Architecture

Architecture of JDBC



Java Application

Client machine (GUI)

Application Server (Java)

JDBC

Server machine (Business Logic/Middle Tier)

DBMS - Proprietary Protocol

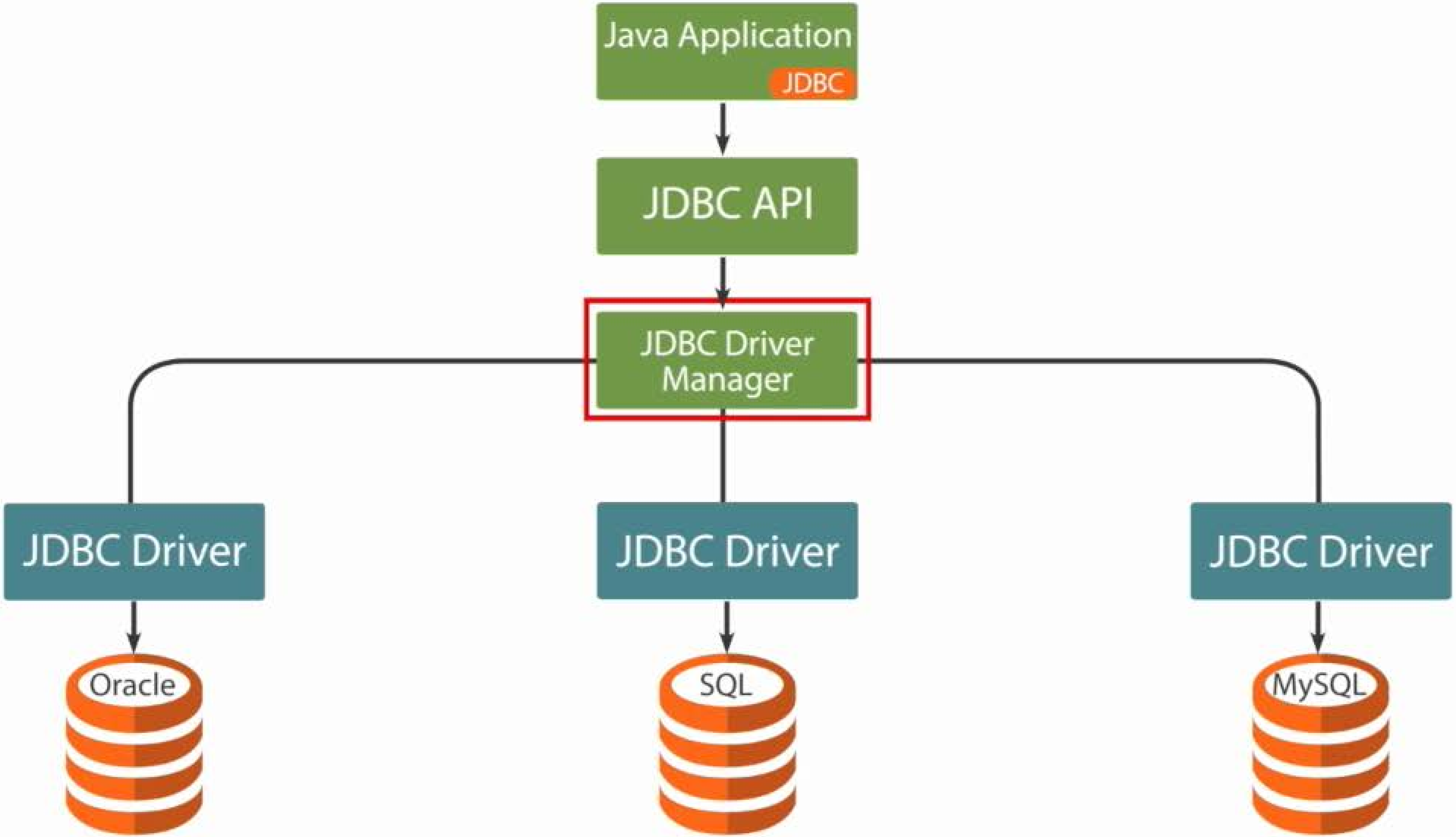
Database server

Three Tier Architecture

Architecture of JDBC

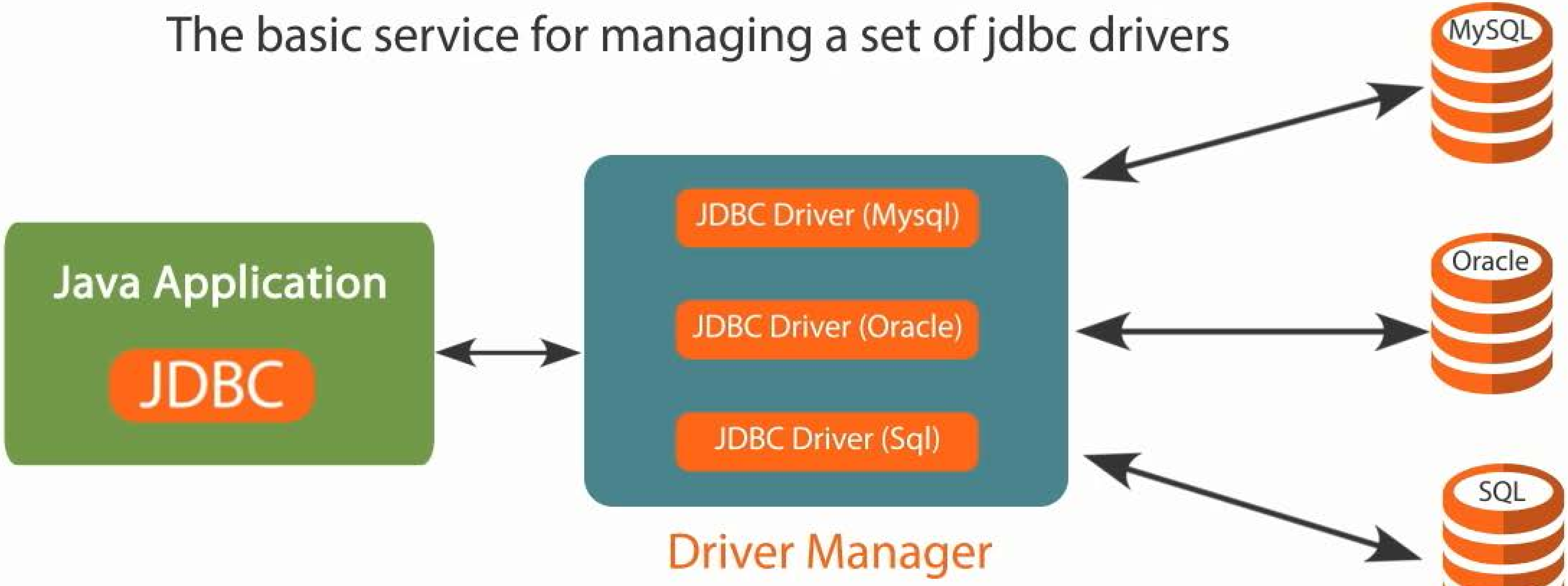
- JDBC API - Application-to-JDBC Manager Connection
- JDBC Driver API - JDBC Manager-to-Driver Connection

Architecture of JDBC

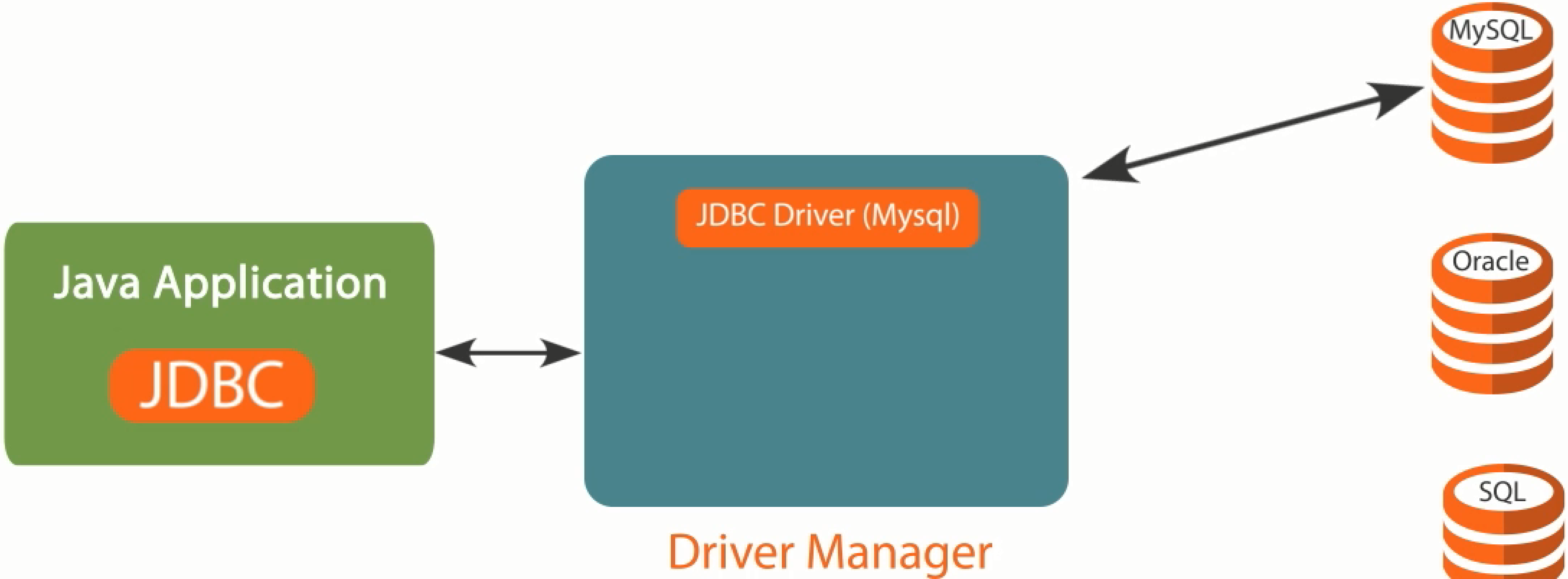


Role of Driver Manager

The basic service for managing a set of jdbc drivers

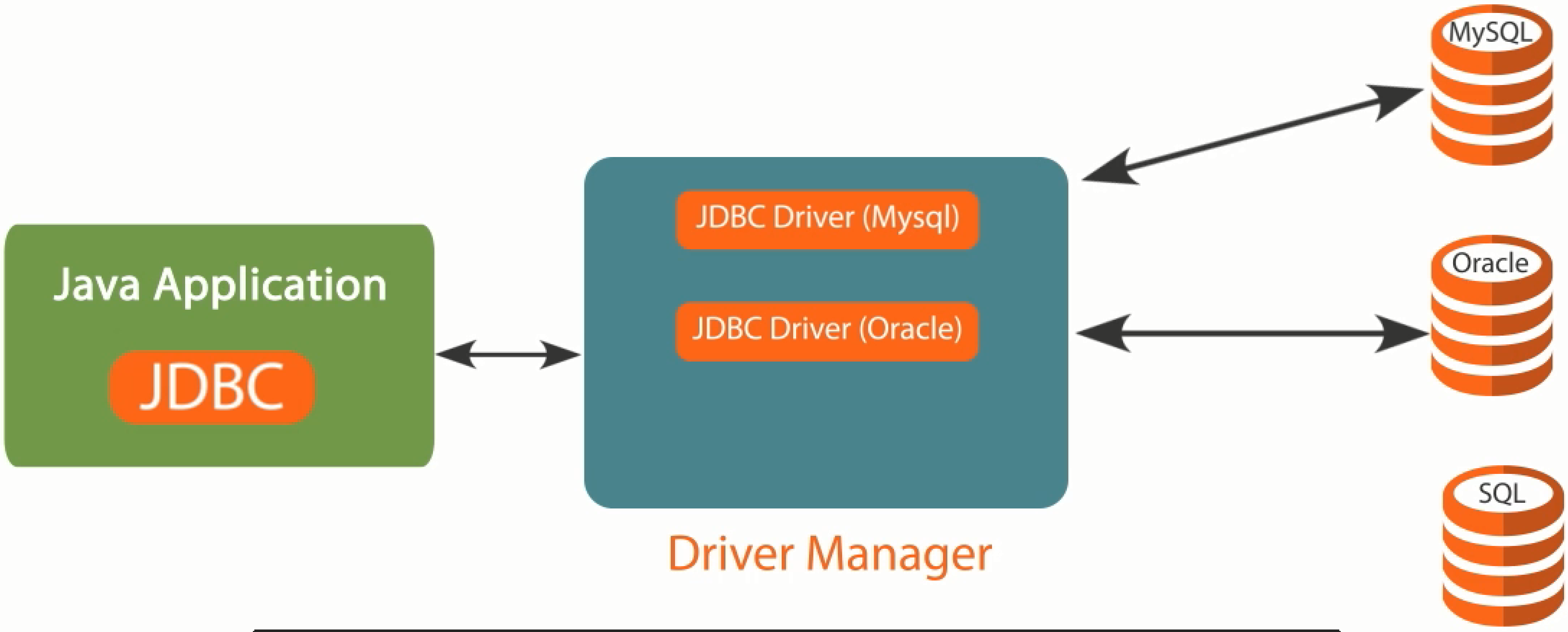


Role of Driver Manager



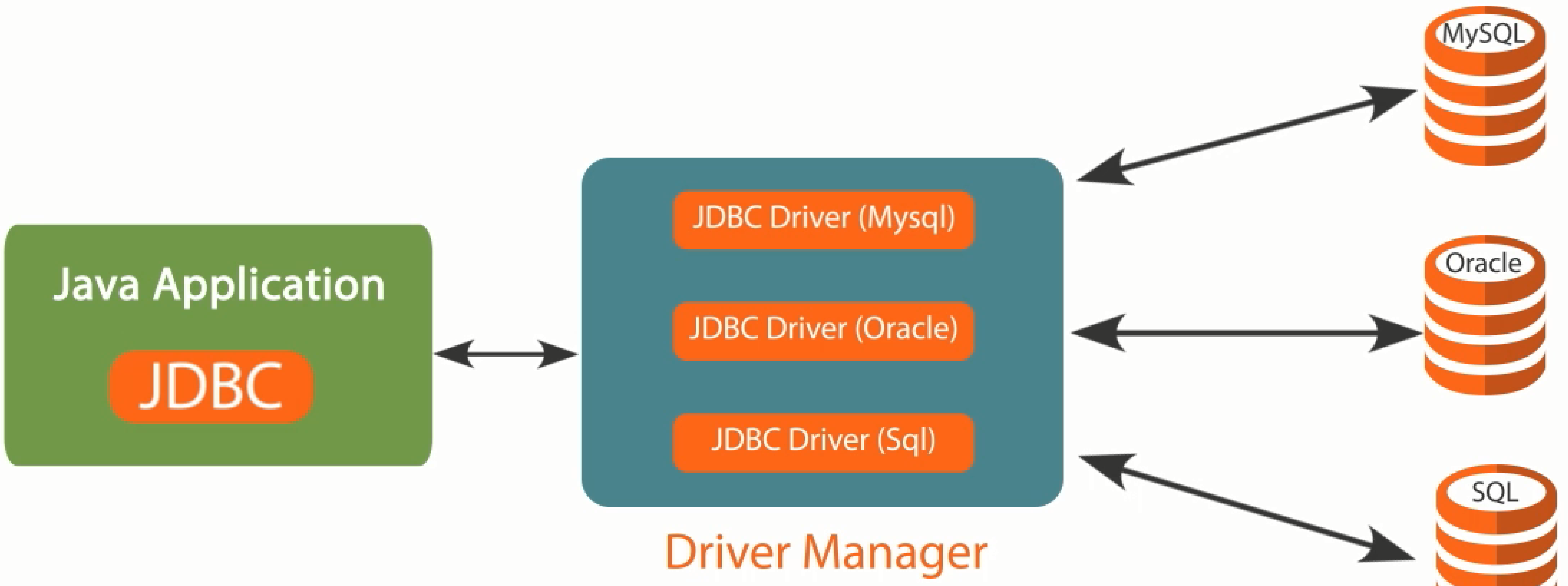
```
Class.forName("com.mysql.jdbc.Driver");
```

Role of Driver Manager



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

Role of Driver Manager



```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver")
```

Driver Manager Updates from JDBC 4.0

- getConnection and getDrivers methods has been enhanced
- No need to load JDBC Drivers explicitly
- Application using Class.forName() will work without modification
- getConnection method of DriverManager will locate suitable Driver

Understanding JDBC Driver Types

A JDBC driver is a set of Java classes that implement the JDBC interfaces, targeting a specific Database

- The JDBC interfaces comes with standard Java

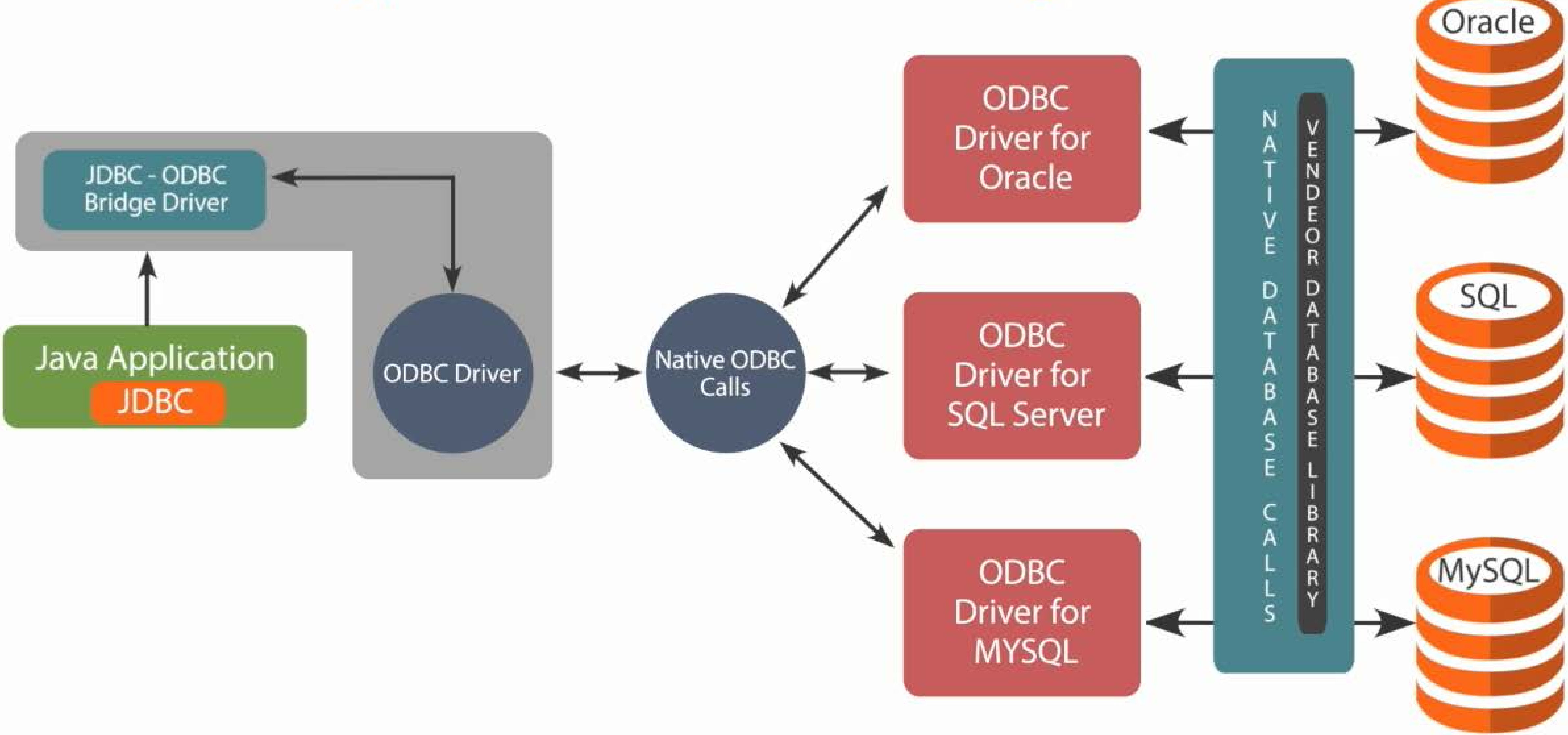
- Implementation of these interfaces is specific to the Database

Understanding JDBC Driver Types

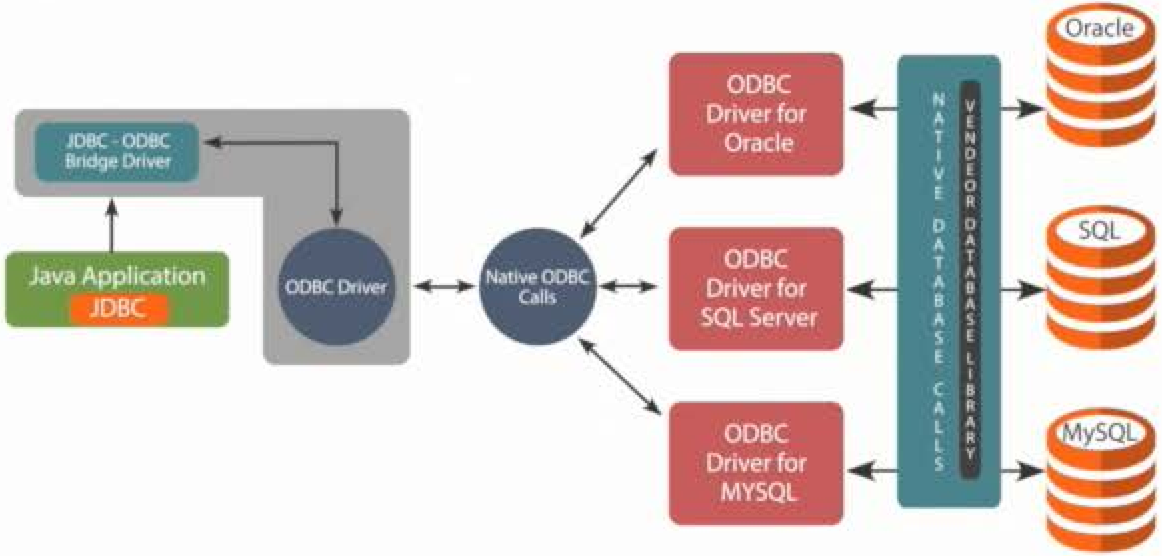
Types of JDBC drivers

- Type 1 : JDBC-ODBC bridge
- Type 2 : Native-API driver
- Type 3 : Network-Protocol driver (Middleware driver)
- Type 4 : Database-Protocol driver (Pure Java driver)

Type 1 : JDBC ODBC Bridge Driver



Type 1 : JDBC ODBC Bridge Driver



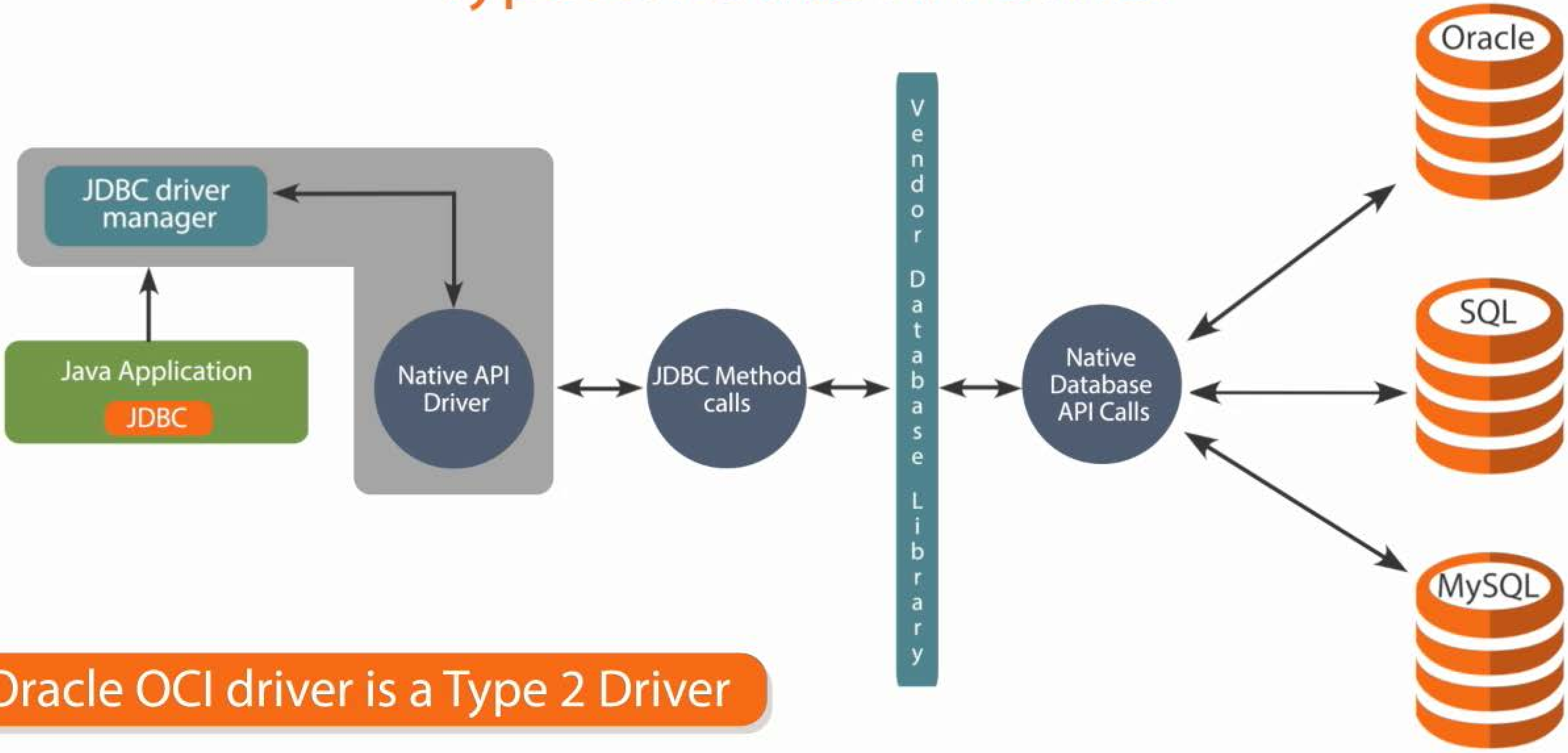
Advantages

- It is very easy to use
- Almost any database is supported

Limitations

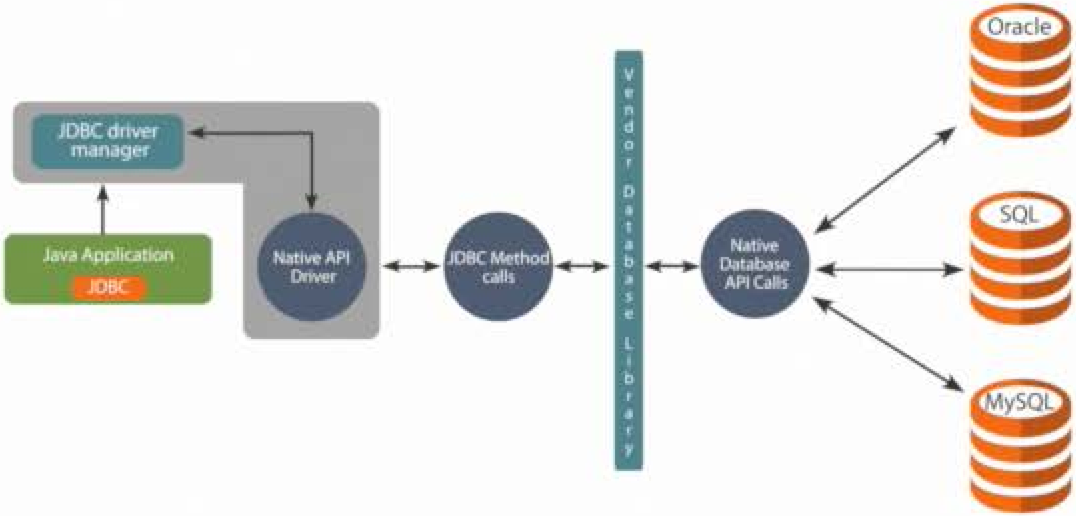
- Performance will not be efficient
- ODBC Driver needs to be installed
- Type 1 drivers are not protable
- Not suitable for Applets

Type 2 : Native-API Driver



Oracle OCI driver is a Type 2 Driver

Type 2 : Native-API Driver



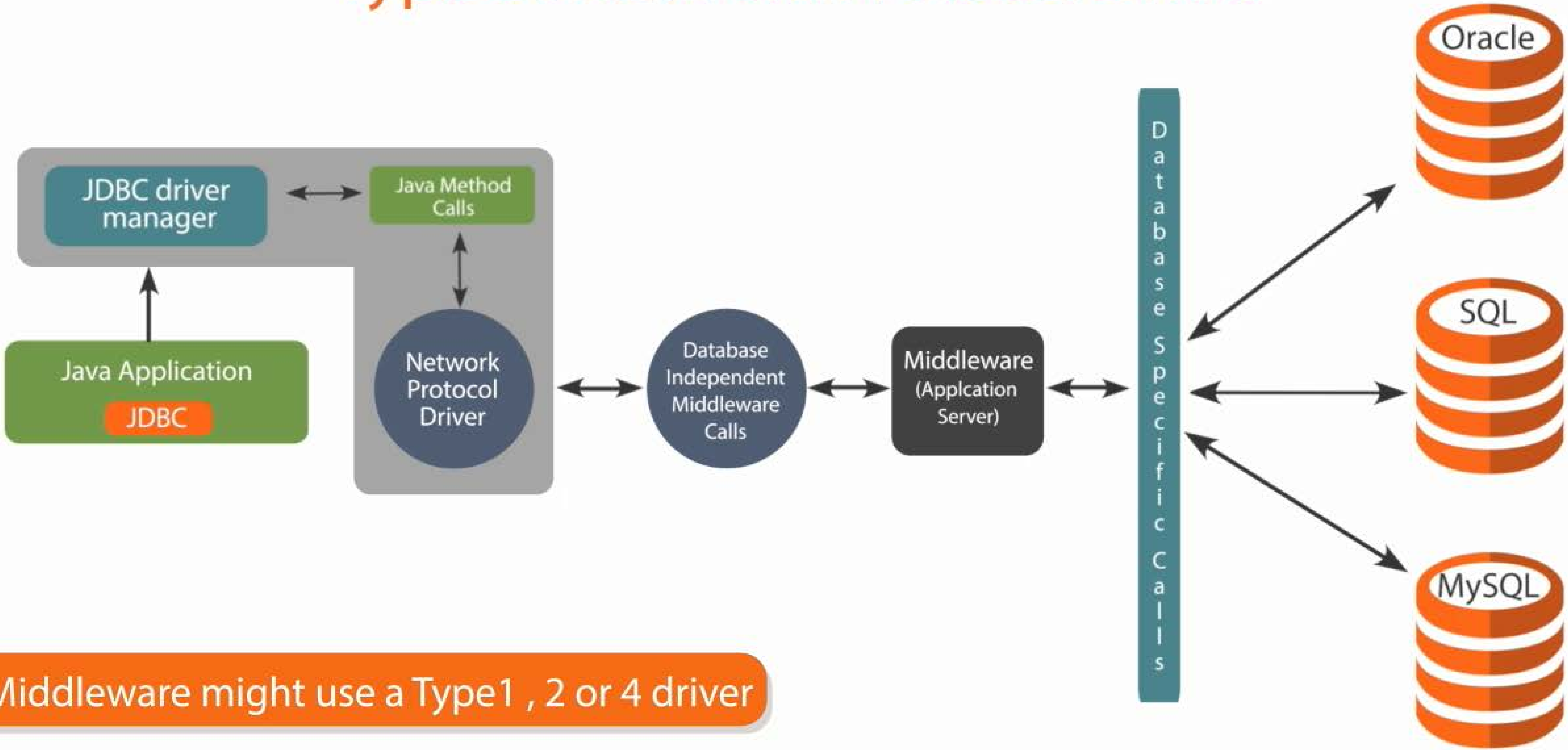
Advantages

- Faster than Type 1 Driver

Limitations

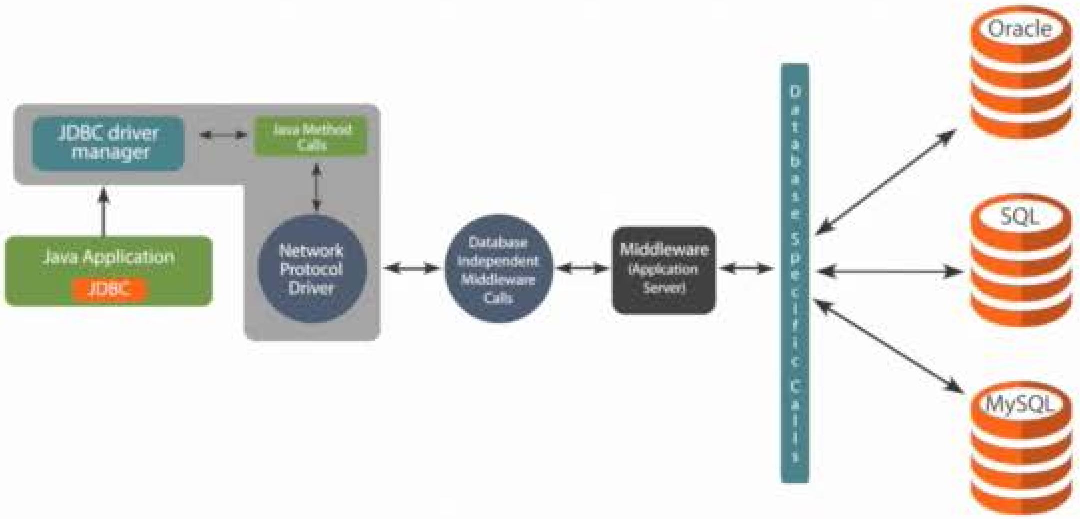
- Client Side Library is not avaiable for all databases
- Vendor Client Library needs to be installed
- It is a Platform Dependent
- Not Thread Safe

Type 3 : Network Protocol Driver



Middleware might use a Type1 , 2 or 4 driver

Type 3 : Network Protocol Driver



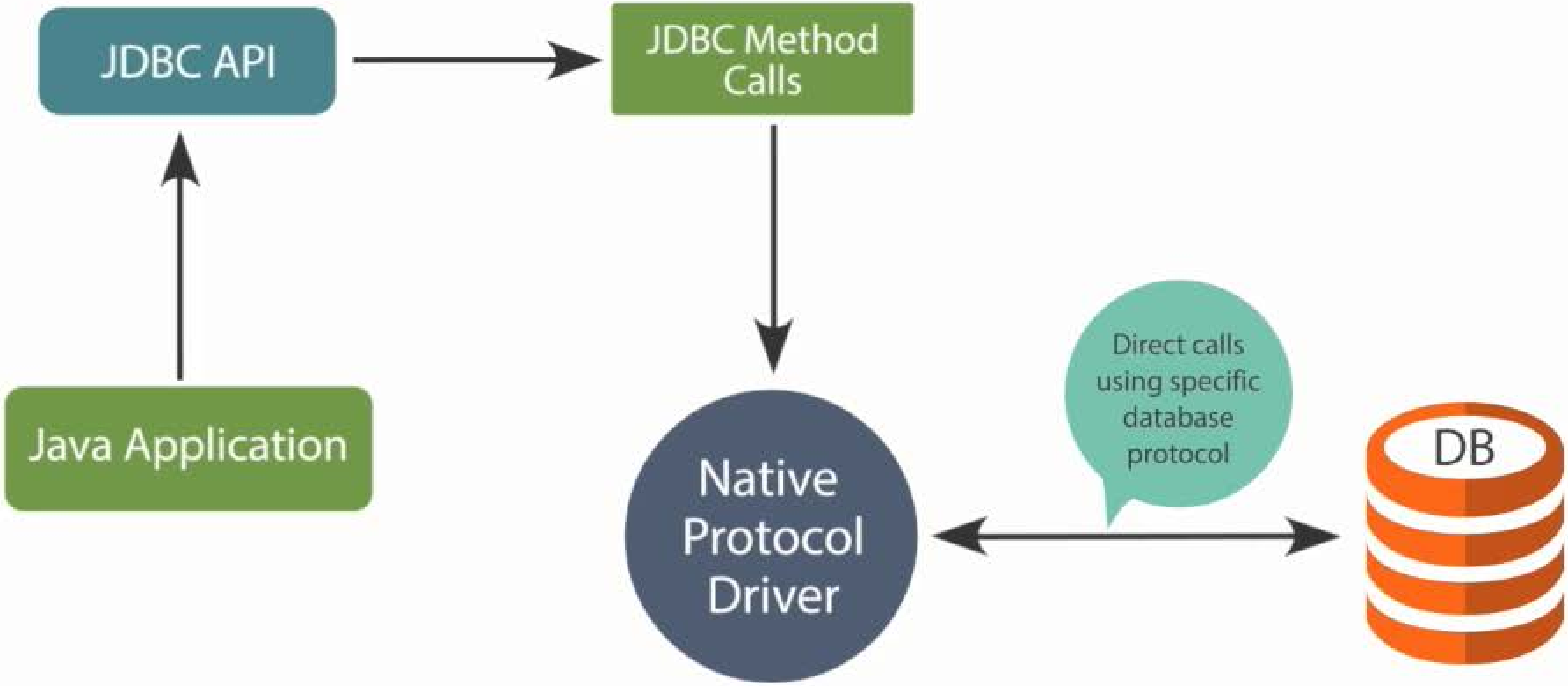
Advantages

- No additional library installation is required on client system
- No changes are required at client for any DB
- Supports Caching of Connection,Query Results, Load Balancing, Logging and Auditing etc.
- A Single Driver can handle any database provided the middleware supports it.

Limitations

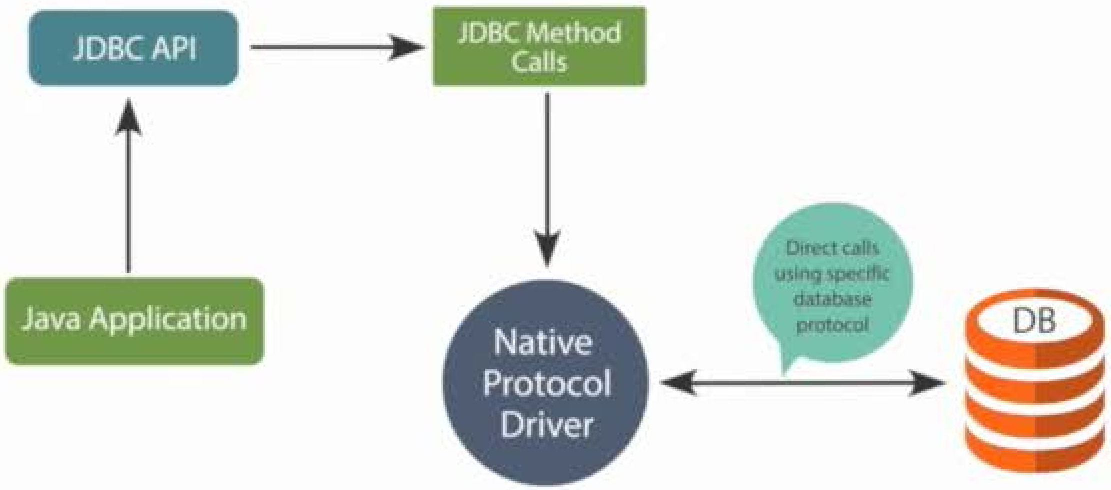
- Performance will be slow
- Requires Database-specific coding is required
- Maintainance of Network Protocol driver becomes costly

Type 4 : Database Protocol Driver



ex: Oracle Thin Driver

Type 4 : Database Protocol Driver



Advantages

- Platform Independenct
- No intermediate format is required
- Application connects directly to the database server
- Performance will be very fast
- JVM manage all aspects

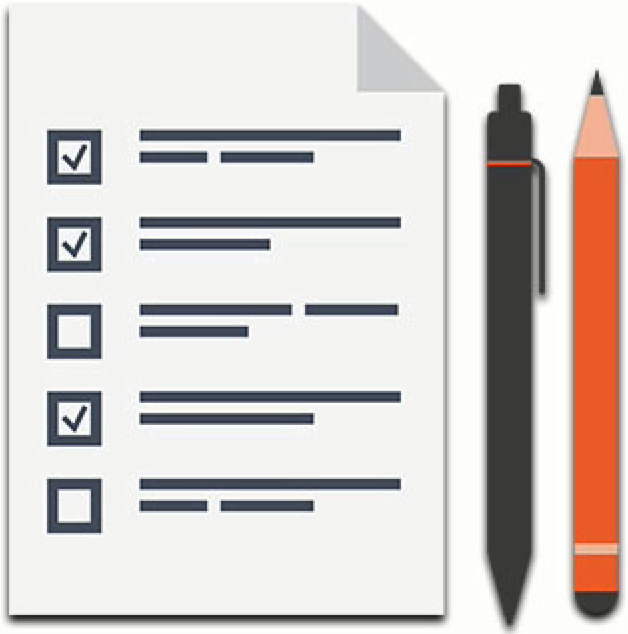
Limitations

- Drivers are database dependent

Overview

- If you are accessing one type of database such as Oracle, SQL Server, MYSQL etc. then the preferred driver type is 4
- If your java application is accessing multiple types of databases at the same time, type 3 is the preferred driver.
- Type 2 drivers are useful in situations where a type 3 or type 4 driver is not avaiable yet for your database
- The type 1 driver is not considered a deployment-level driver and it is typically used for development and testing purposes only

Summary



- Understanding Prerequisites
- Introduction to JDBC
- Architecture of JDBC
- Role of Driver Manager
- Understanding JDBC Driver Types