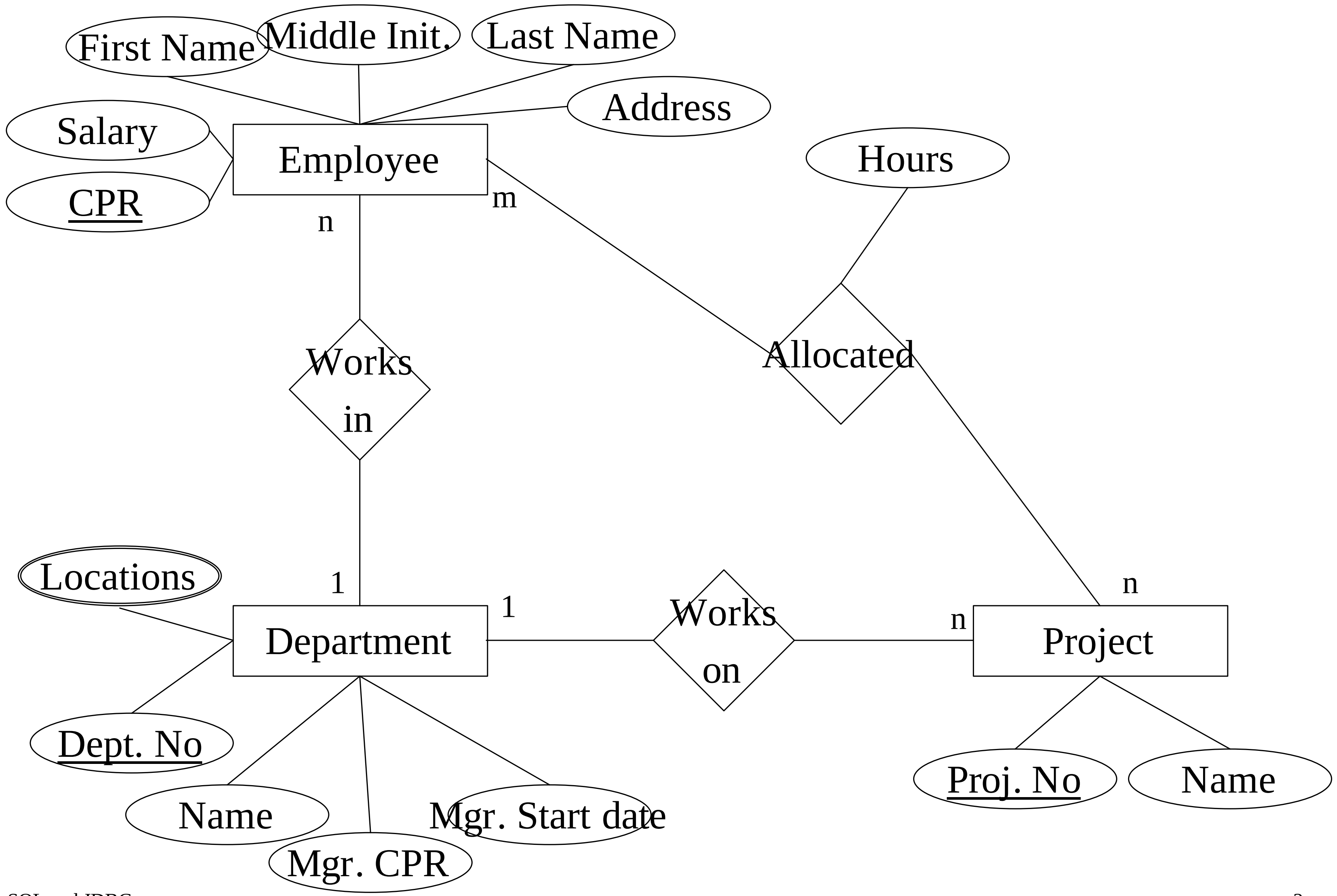
SQL and JDBC

- SQL (a database query language)
  - CREATE/DROP
  - INSERT/DELETE/UPDATE
  - SELECT
- JDBC (Java Database Connectivity)
  - The standard way to access databases from Java.

SQL

- Standard query language for accessing relational databases.
- Persistency of data across program invocations.

Sample Application



CREATE TABLE

```
CREATE TABLE Employees (
      FNAME        VARCHAR (30),
      MINIT        VARCHAR (1),
      LNAME        VARCHAR (30),
      CPR          VARCHAR (11),
      …
      SALARY       NUMERIC (8,0),
      DNO          NUMERIC (2,0));


CREATE TABLE Departments (
      DNAME              VARCHAR (20),
      DNUMBER            NUMERIC (3,0),
      MGRCPR             VARCHAR (11),
      MGRSTARTDATE       DATE);
```

DROP TABLE

```
DROP TABLE Employees;
DROP TABLE Departments;
DROP TABLE Projects;
DROP TABLE Locations;
DROP TABLE Allocations;
```

- Drops both the table definition and the data.

INSERT

```
INSERT INTO Employees VALUES
('Lars', NULL, 'Andersen', '123', '1955-12-10',
 'Klarup', 'M', '15000',  '12');


INSERT INTO Employees VALUES
('Charlotte', 'F', 'Kierkegaard', '789', '1975-08-06',
 'Vejgaard', 'F', '14000', '11');
COMMIT;
```

- The ordering of the attributes is important
- If no value is available use the special NULL value.

Update

```
-- Update a single employees salary
UPDATE Employees SET
      minit = ’M’,
      salary = 23400
WHERE fname = ’Lars’ AND lname = ’Andersen’;

-- Update all the salaries
UPDATE Employees SET
  salary = salary * 1.1
```

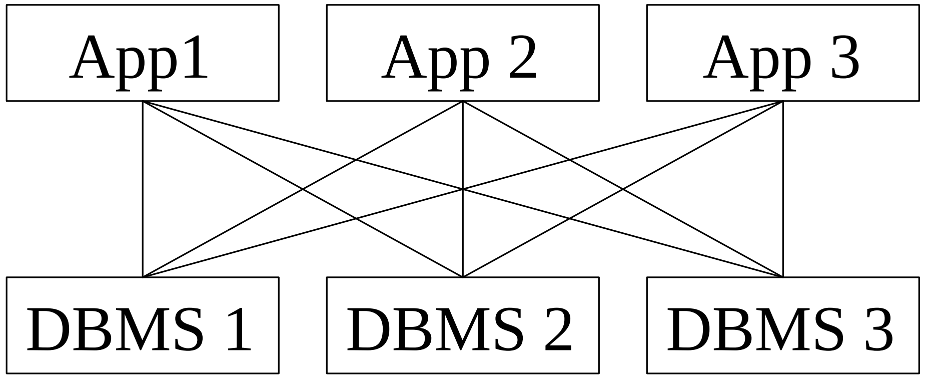
DELETE

```
-- Delete a single employee
DELETE FROM Employees
WHERE fname = ’Lars’ AND lname = ’Andersen’;


-- Delete all employees
DELETE FROM Employees;
```

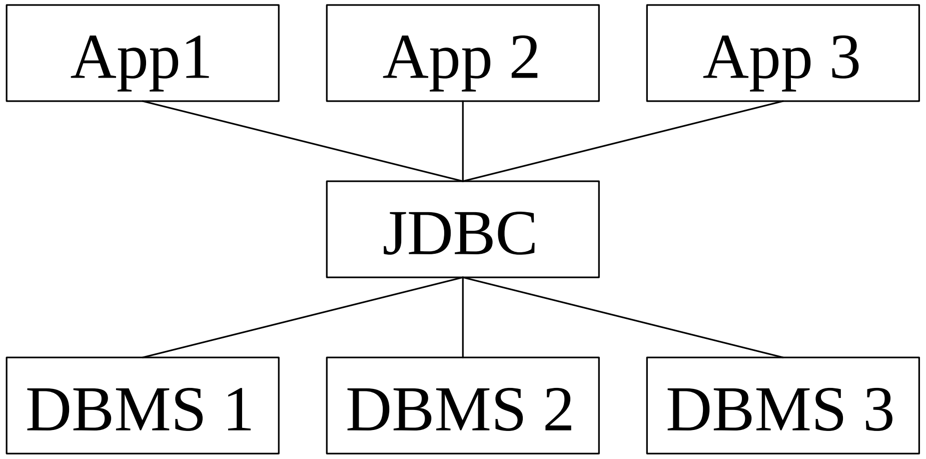
SELECT

```
-- Get all the contents from the Employees table
SELECT *
FROM   Employees;

-- Find the first names of female employees
SELECT FName
FROM   Employees
WHERE  sex = 'F';

-- Find info on employees in specific department
SELECT employees.fname, employees.cpr
FROM   employees, department
WHERE  employees.dno = department.dnumber
AND    department.dname = ’Interactive TV’;
```

The Problem Solved by JDBC



n interfaces each app



1 interface each app

The Need and the Approach

- The need for accessing data from heterogeneous databases, within an application not targeted towards ad-hoc queries.

- Conventional Solutions:
  - Embedded SQL + use of a precompiler
  - Application Level Interface (API) “Just Another Library“ idea.

JDBC

- Java API
- Newest version is JDBC 2.x

- Based on Open Database Connectivity (ODBC), but there are important differences.
- No software needs to be installed on the client it can run directly over the internet.
- JDBC is multiplatform by nature due to the nature of Java

A Simple JDBC Application

```
import java.sql.*;
// Load the driver
Class.forName (“myDriver.ClassName”);
// <protocol>[:<sub protocol>]:@<host>:port:SID
String url = “jdbc:oracle:thin:@blob.cs.auc.dk:1521:blob1”;
// Make a connection
Connection con =
   DriverManager.getConnection (url, “myLogin”, “myPassword”);
// Create a statement
Statement stmt = con.createStatement();
// Query and result set
ResultSet rs = stmt.executeQuery (“SELECT * FROM Emp”);
while (rs.next()){/* print the result set */ }
// Clean up
stmt.close();
con.close();
```

Get a Connection

```
public Connection connector (String user_name,
                             String password)
     throws SQLException {
 Connection conn = null;
   try {
     // Load the Oracle JDBC driver
     DriverManager.registerDriver(
           new oracle.jdbc.driver.OracleDriver());
     String url =
        "jdbc:oracle:thin:@blob.cs.auc.dk:1521:blob2";
     conn = DriverManager.getConnection (url,
                                         user_name,
                                         password);
  }
  catch (SQLException e) { System.err.println (e); }
  return conn;}
```

JDBC CREATE TABLE

```
public void create_table
      (Connection conn,
       String table_stmt) throws SQLException  {
  try {
    Statement stmt = conn.createStatement();
    int res = stmt.executeUpdate (table_stmt);
    if (res == 0) {
      System.out.println ("Table created");
    }
    stmt.close();
  }
  catch (SQLException e) {
    System.err.println (e) ;
  }
}
```

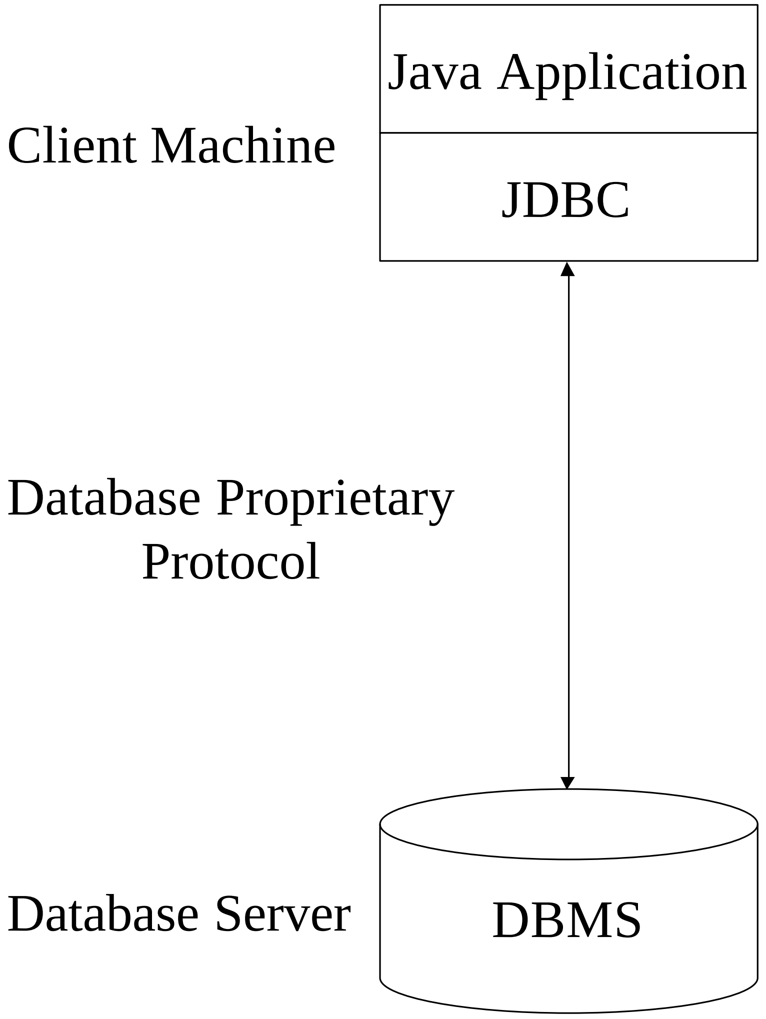
JDBC Query

```
public void query_1 (Connection conn){
      String query = "SELECT * FROM Employees";
  try {
      Statement stmt = conn.createStatement();
      ResultSet rs = stmt.executeQuery(query);
      while (rs.next()){
            String fname   = rs.getString ("FNAME");
            String minit   = rs.getString (2);
            String lname   = rs.getString (3);
            …
            String address = rs.getString (6);
            String sex     = rs.getString (7);
            System.out.println (fname + minit + lname);
      }
      stmt.close();
}
```

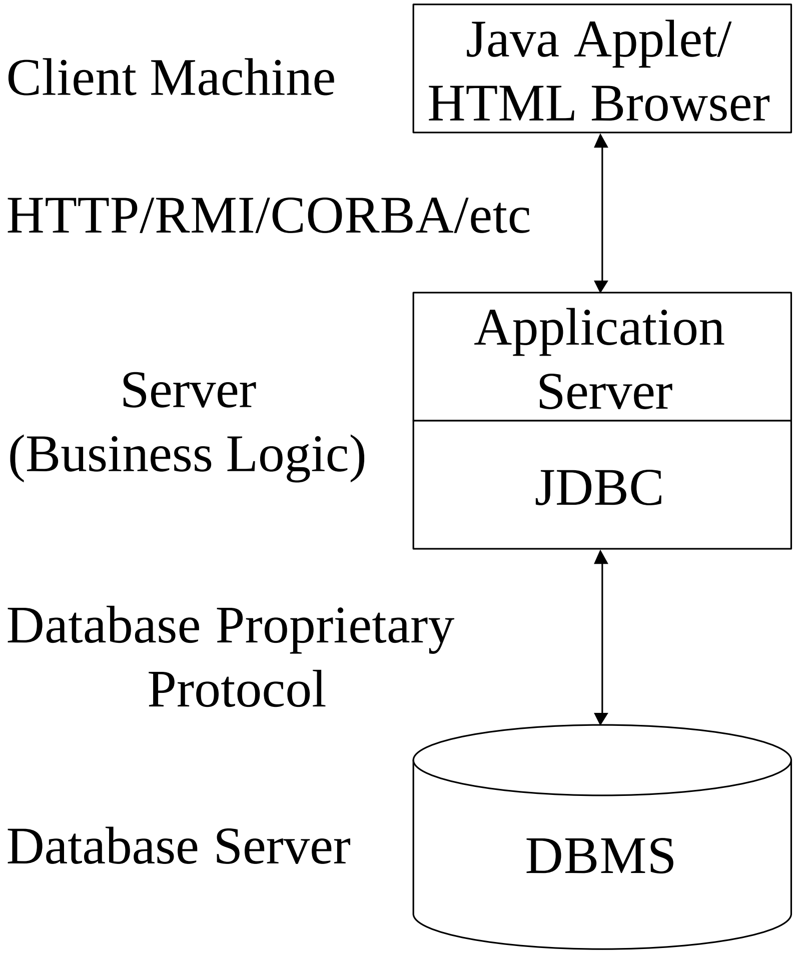
`java.sql` Overview

- **`Driver`**. Supports the creation of a data connection.
- **`Connection`**. Represents the connection between a Java client and an SQL database.
- **`DatabaseMetaData`**. Contains information about the SQL database.
- **`Statement`**. Includes methods for execution queries.
- **`PreparedStatement`**. Represents precompiled and stored queries.
- **`ResultSet`**. Contains the results of the execution of a query.
- **`ResultSetMetaData`**. Contains information about a **`ResultSet`**, e.g., attribute names and types.

Two-Tier and Three-Tier Models

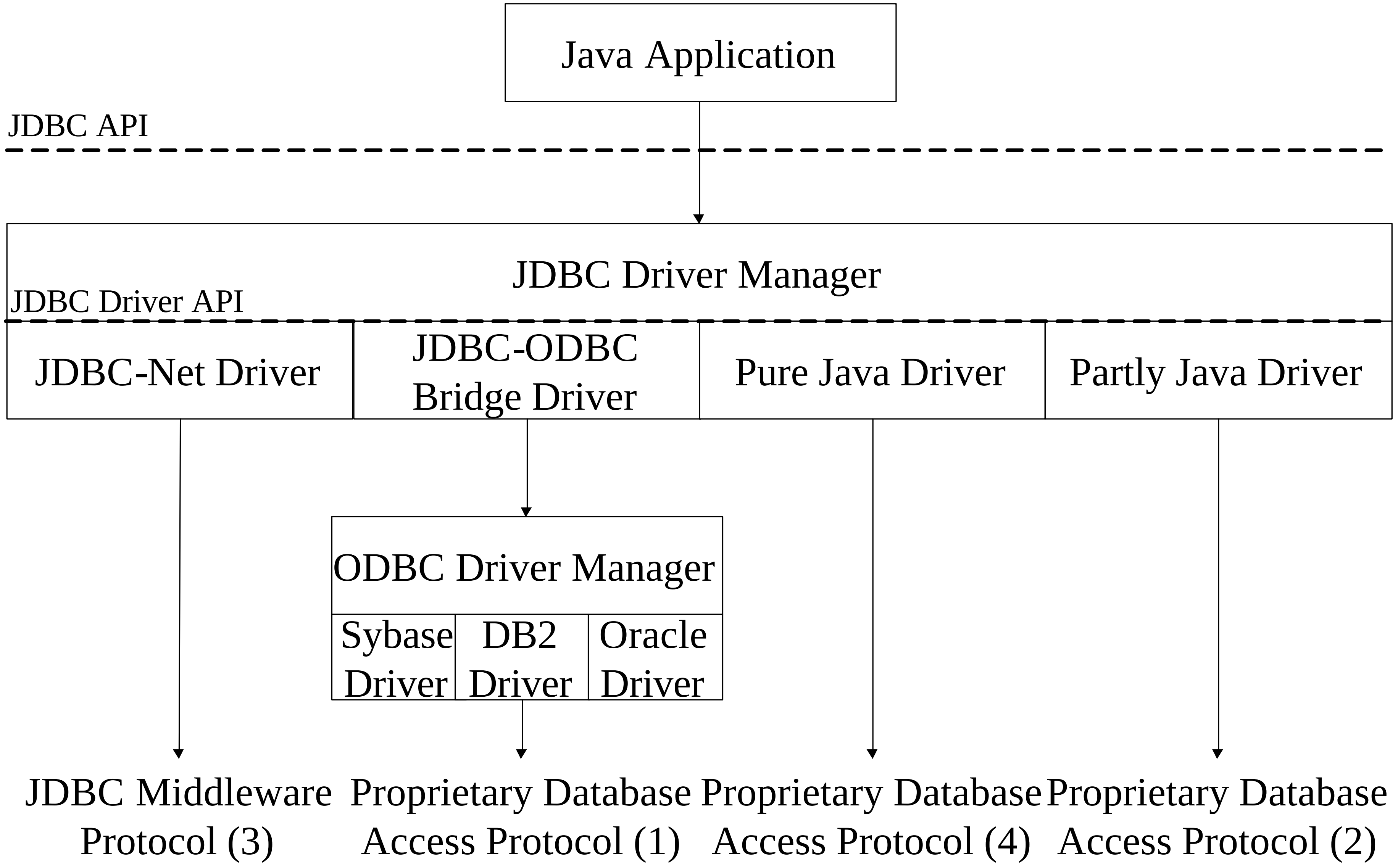


two-tier



three-tier

JDBC Driver Types



JDBC Summary

- Object-oriented API
- Very widely accepted and used in the Java world
- Can be used to access DBMSs from applets
- Both client platform and DBMS platform independence