Introduction to JDBC

Objectives of This Session

- State what is Java Database Connectivity
- State different types of drivers supported by JDBC
- Describe the steps to be followed for writing a simple JDBC application
- Describe the use of Resultset interface

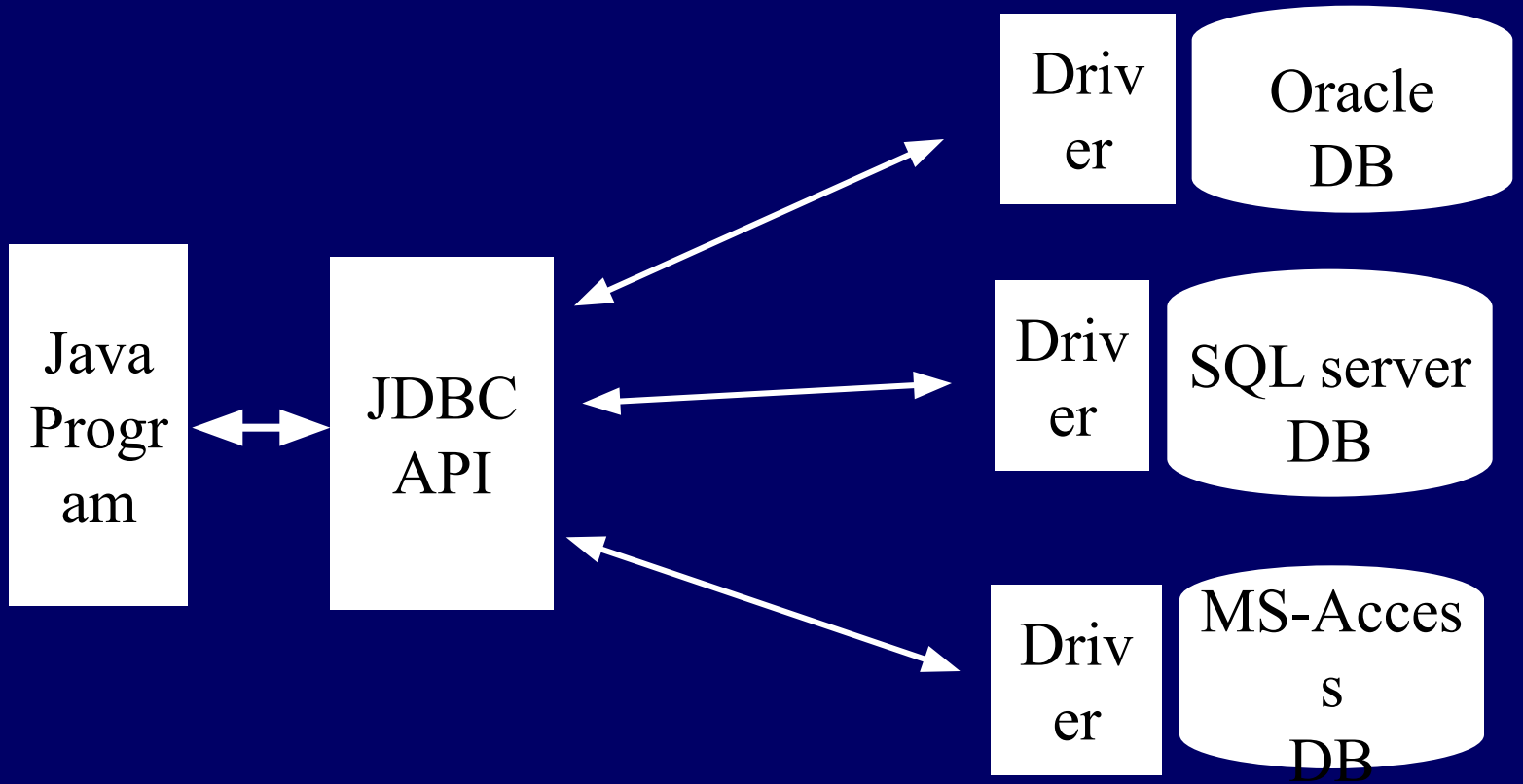
JDBC

- Lets programmers connect to a database, query it or update through a Java application.
- Programs developed with Java & JDBC are platform & DB vendor independent.
- JDBC library is implemented in java.sql package.

JDBC

A driver is a program that converts the Java method calls to the corresponding method calls understandable by the database in use.

JDBC


Java Program
JDBC API
Driver
Oracle DB
Driver
SQL server DB
Driver
MS-Access DB

ODBC

- A driver manager for managing drivers for SQL based databases.
- Developed by Microsoft to allow generic access to disparate database systems on windows platform.
- Java SE comes with JDBC-to-ODBC bridge database driver to allow a java program to access any ODBC data source.

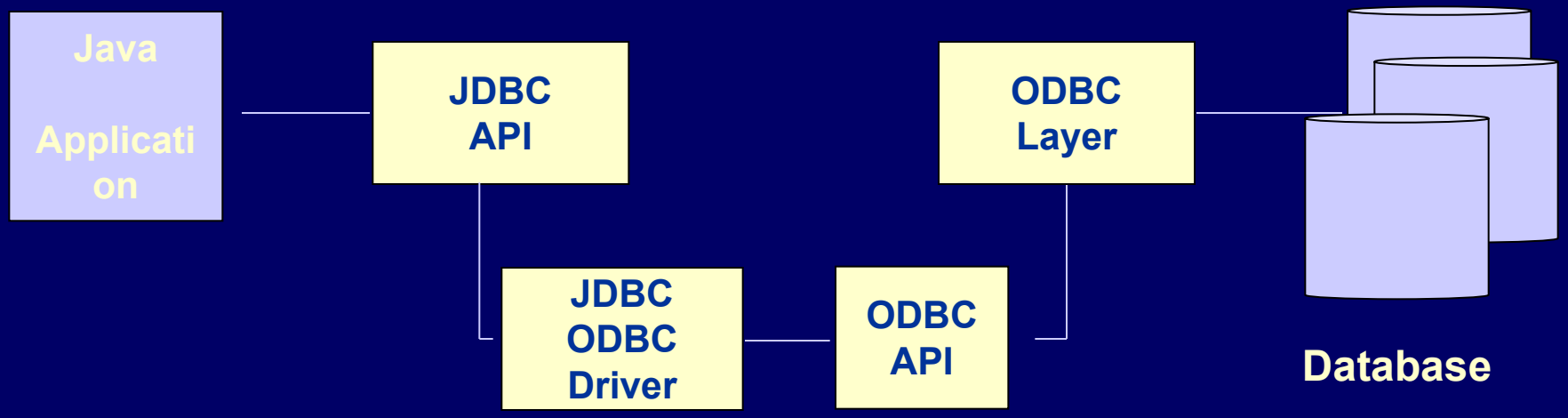
JDBC Vs ODBC

- ODBC is a ‘C’ API
- ODBC is hard to learn – because of low-level native ODBC.
- ODBC most suited for only Windows platform
- No platform independence

JDBC(Drivers)

- JDBC-ODBC Bridge (Type 1)
- Native-API partly Java Driver (Type 2)
- Net-Protocol All-Java Driver (Type 3)
- Native Protocol All-Java Driver (Type 4)

JDBC Driver Type 1

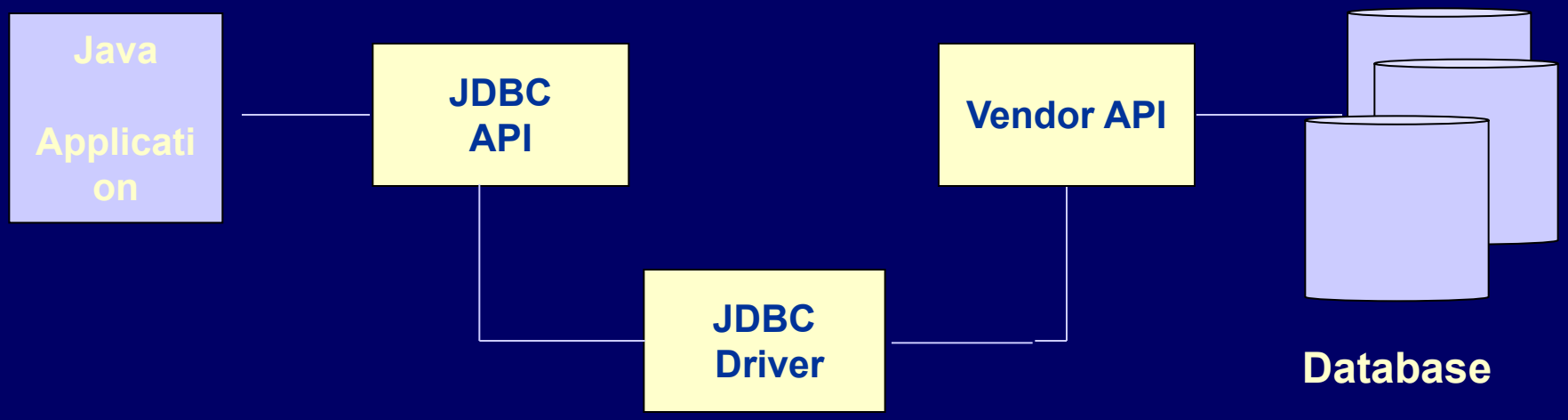


Type 1 Driver (JDBC-ODBC Bridge driver)

- Translates all JDBC API calls to ODBC API calls.
- Relies on an ODBC driver to communicate with the database.

- Disadvantages
  - ODBC required hence all problems regarding ODBC follow.
  - Slow

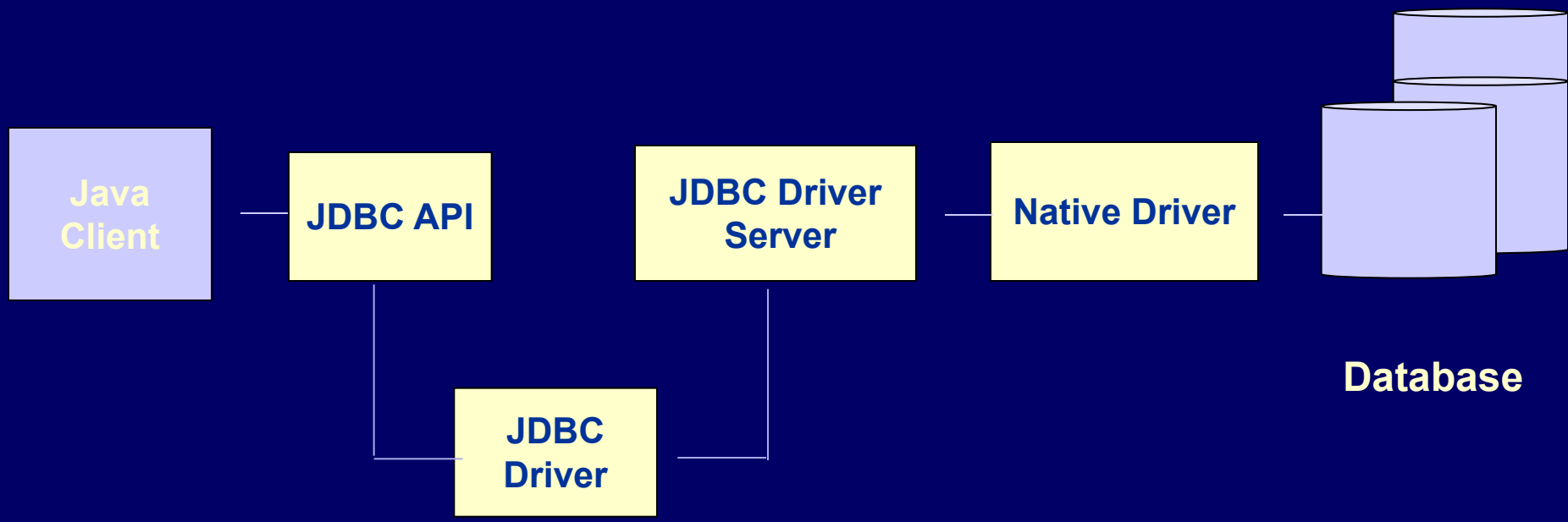
JDBC Driver Type 2



Type 2 (Native lib to Java implementation)

- Written partly in Java & partly in native code, that communicates with the client API of a database.
- Therefore, should install some platform-specific code in addition to Java library.
- The driver uses native ‘C’ lang lib calls for conversion.

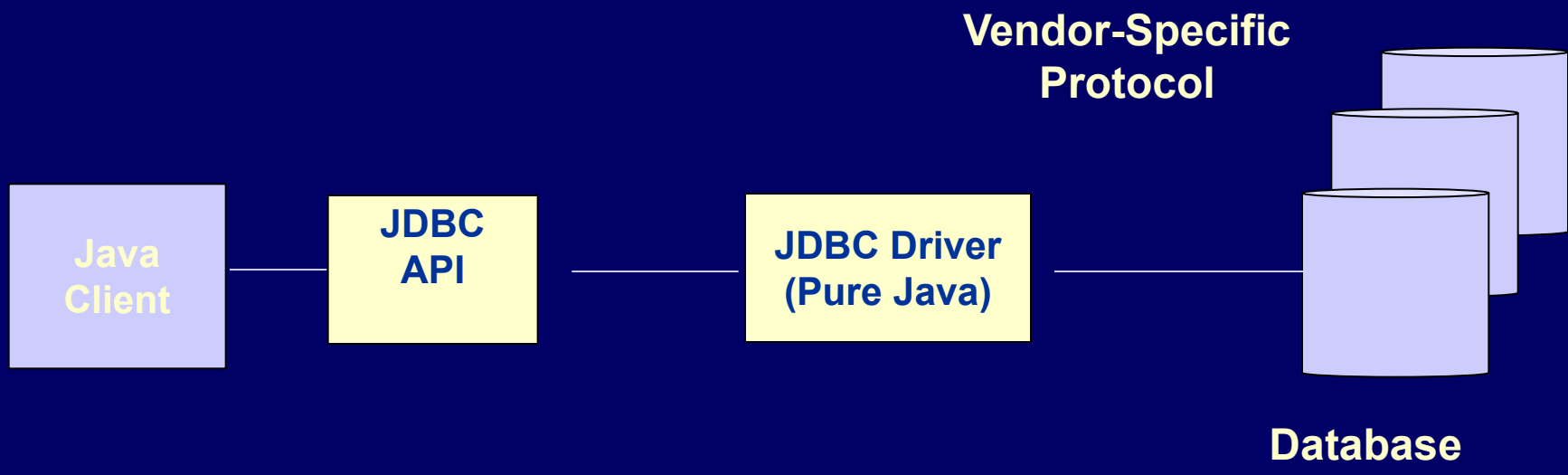
JDBC Driver Type 3



Type 3 (pure network protocol Java driver)

- Uses DB independent protocol to communicate DB-requests to a server component.
- This then translates requests into a DB-specific protocol.
- Since client is independent of the actual DB, deployment is simpler & more flexible.

JDBC Driver Type 4

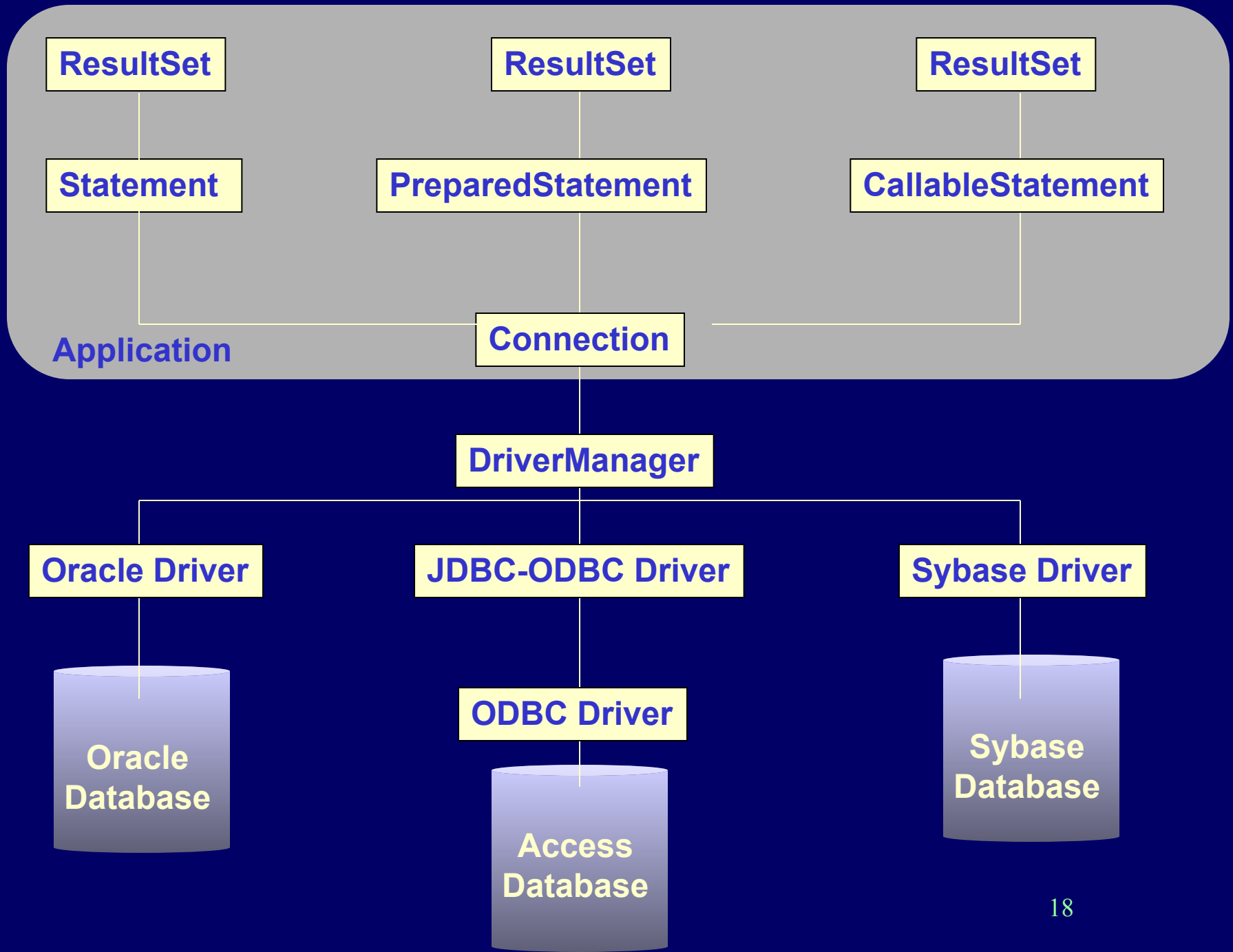


Type 4 Driver (Pure Java drivers)

- JDBC calls are directly converted to network protocol used by the DBMS server.
- Driver converts JDBC API calls to direct network calls using vendor-specific networking protocols by making direct socket connections with the DB.
- But driver usually comes only from DB-vendor.

JDBC API

- API layer has2 levels of interface.
  - **Application layer**: developer uses API to make calls to DB via SQL & retrieve results.
  - **Driver layer** : handles all communication with a specific Driver implementation.


ResultSet
ResultSet
ResultSet
Statement
PreparedStatement
CallableStatement
Application
Connection
DriverManager
Oracle Driver
JDBC-ODBC Driver
Sybase Driver
Oracle Database
ODBC Driver
Access Database
Sybase Database

JDBC(Getting Connection)

```
try{
Class.forName(“sun.jdbc.odbc.JdbcOdbcDriver”);
      Properties p = new Properties();
      p.put(“user”,”dba”);
      p.put(“password”,”sql”);
String url = “jdbc:odbc:mytable”;
Connection c = DriverManager.getConnection(url,p);
}
```

JDBC URL

- Needed by drivers to locate ,access and get other valid information about the databases.
- jdbc:driver:database-name
  - jdbc:Oracle:products
  - jdbc:odbc:mydb; uid = aaa; pwd = secret
  - jdbc:odbc:Sybase
  - jdbc:odbc://whitehouse.gov.5000/cats;

JDBC(Interfaces)

- Driver
- Connection
- Statement
- PreparedStatement
- CallableStatement
- DatabaseMetadata
- ResultSet
- ResultSetMetadata

JDBC(Classes)

- Date
- DriverManager
- DriverPropertyInfo
- Time
- TimeStamp
- Types

Driver Interface

- Connection connect(String URL, Properties info)
  - Checks to see if URL is valid.
  - Opens a TCP connection to host & port number specified.
  - Returns an instance of Connection object.
- Boolean acceptsURL(String URL)

Driver Manager class

- Connection getConnection(String URL)
- void registerDriver(Driver driver)
- void deregisterDriver()

Eg : Connection conn = null;

conn =
DriverManager.getConnection("jdbc:odbc:mydsn");

Connection

- Represents a session with the DB connection provided by driver.
- You use this object to execute queries & action statements & commit or rollback transactions.

JDBC(Connection)

- close()
- commit()
- void setAutoCommit(boolean b)
- rollback()
- Statement createStatement()
- CallableStatement prepareCall(String sql)
- PreparedStatement prepareStatement(String sql)

JDBC(Statement)

- Statement
  - PreparedStatement
  - CallableStatement

Statement Methods

- boolean execute(String sql)
- ResultSet executeQuery(String sql)
- int executeUpdate(String sql)

JDBC(Simple Query)

```
url = “jdbc:odbc:MyDataSource”
Connection con = DriverManager.getConnection( url);
Statement stmt = con.createStatement();
String  sql = “SELECT Last_Name FROM EMPLOYEES”
ResultSet rs = stmt.executeQuery(sql);

while(rs.next())
{       System.out.println(rs.getString(“Last_Name”));
}
```

JDBC(Simple Query)

```
public static void main(String args[]
{
try{
    Class x = Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
}
catch( Exception se){
    System.out.println("database connection error!");
}

}
```

JDBC(Parameterised SQL)

```
String SQL =
    "select * from Employees where First_Name=?"

PreparedStatement pstat = con.prepareStatement(sql);

pstat.setString(1, "John");

ResultSet rs = pstat.executeQuery();

pstat.clearParameters();
```

JDBC(stored procedure)

```
CREATE OR REPLACE PROCEDUE sp_interest
( id  IN  INTEGER
bal  IN  OUT  FLOAT ) AS
BEGIN
     SELECT balance INTO bal FROM    accounts
WHERE account_id = id ;
     bal := bal + bal * 0.03 ;
     UPDATE accounts
     SET balance = bal
     WHERE account_id = id ;
END;
```

JDBC(stored procedure)

```
CallableStatement cstmt =
con.prepareCall( "{ call sp_interest(?,?)} ");
cstmt.setInt(1,accountID);
cstmt.setFloat(2, 5888.86);
cstmt.registerOutParameter(2,Types.FLOAT);

cstmt.execute();

System.out.println(cstmt.getFloat(2));
```

Java 2 Resultset

```
Statement stmt =
        conn.createStatement(type, concurrency);
```

- TYPE_FORWARD_ONLY
- TYPE_SCROLL_INSENSITIVE
- TYPE_SCROLL_SENSITIVE
- CONCUR_READ_ONLY
- CONCUR_UPDATEABLE

JDBC(ResultSet)

- first()
- last()
- next()
- previous()
- beforeFirst()
- afterLast()
- absolute(int)
- relative(int)

JDBC

```
Statement stmt =
con.createStatement(ResultSet.TYPE_SCROLL_
SENSITIVE, ResultSet.CONCUR_UPDATABLE);

ResultSet rs = stmt.executeQuery("Select ….");

rs.first();
rs.updateInt(2, 75858 );
rs.updateRow();
```

JDBC

```
ResultSet rs = stmt.executeQuery("Select ….");

rs.moveToInsertRow();
rs.updateString(1, "fkjafla" );
rs.updateInt(2, 7686);
rs.insertRow();

rs.last();
rs.deleteRow();
```

ResultSetMetadata Interface

- Object that can be used to find out about the types and properties of the columns in a ResultSet
- Example
  - Number of columns
  - Column title
  - Column type