UNIT-II JAVA-Database Programming

Prof.Chintan Dave

What is Database Programming ?

A **database program** is the heart of a business information system and provides
file creation, data entry, update, query and reporting functions.
The traditional term for **database** software is "**database**management system"

Context of Database-Apps

What is Java Database Programming??- JDBC

JDBC stands for **J**ava **D**ata**b**ase **C**onnectivity, which is a standard Java API for database- **independent connectivity between the Java programming language and a wide range of Database**.

What DBMS(S) You Can Work with JAVA-JDBC

ORACLE

MYSQL….

Common Tasks Involved in Java Database Programming

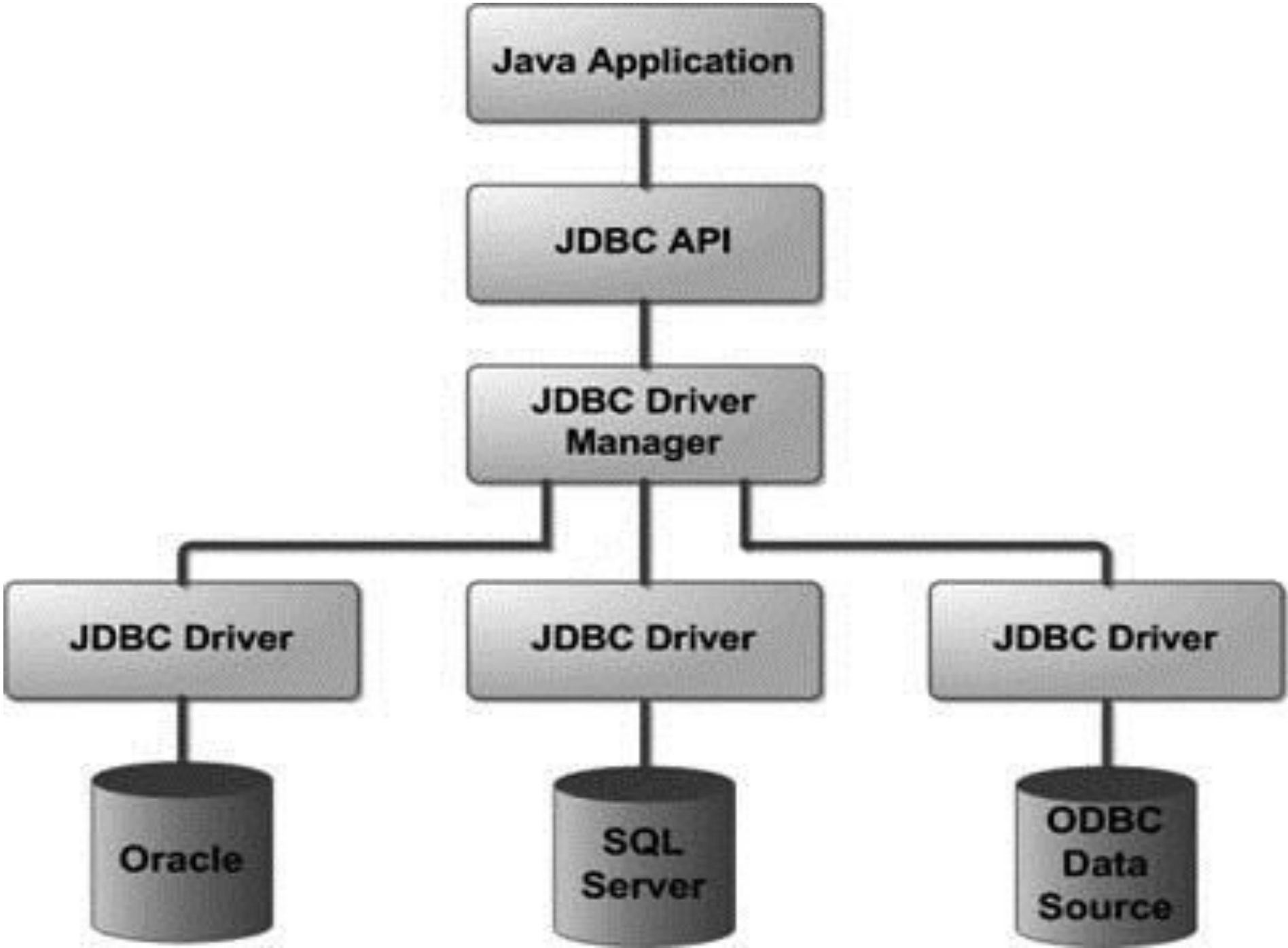
- Making a connection to a database
- Creating SQL or MySQL statements
- Executing that SQL or MySQL queries in the database
- Viewing & Modifying the resulting records

Java Application(s) that can Use JDBC

- Core Java Applications(Console Apps+ GUI Apps)
- Applet Application
- Servlet Applications
- Java Server Pages
- Enterprise Java Applications

JDBC Architecture

- **JDBC API:** This provides the application-to-JDBC Manager connection.
- **JDBC Driver API:** This supports the JDBC Manager-to-Driver Connection.


Java Application
JDBC API
JDBC Driver Manager
JDBC Driver
JDBC Driver
JDBC Driver
Oracle
SQL Server
ODBC Data Source

Components of JDBC

- **DriverManager:** This class manages a list of database drivers.
- **Driver:** This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects
- **Connection :** This interface with all methods for contacting a database. **Statement :** You use objects created from this interface to submit the SQL statements to the database.
- **ResultSet:** retrieved Data from a database
- **SQLException:** handles any errors that occur in a database application.

Please Refer SQL Syntax

- FOR CRUD Operations
- Get Some Supplementary Reading on SQL and Stored Procedures

What is JDBC Driver??

JDBC drivers **implement the defined interfaces in the JDBC API for interacting with your database server**.

Type 1: JDBC-ODBC Bridge Driver

Type 2: JDBC-Native API

Type 3: JDBC-Net pure Java

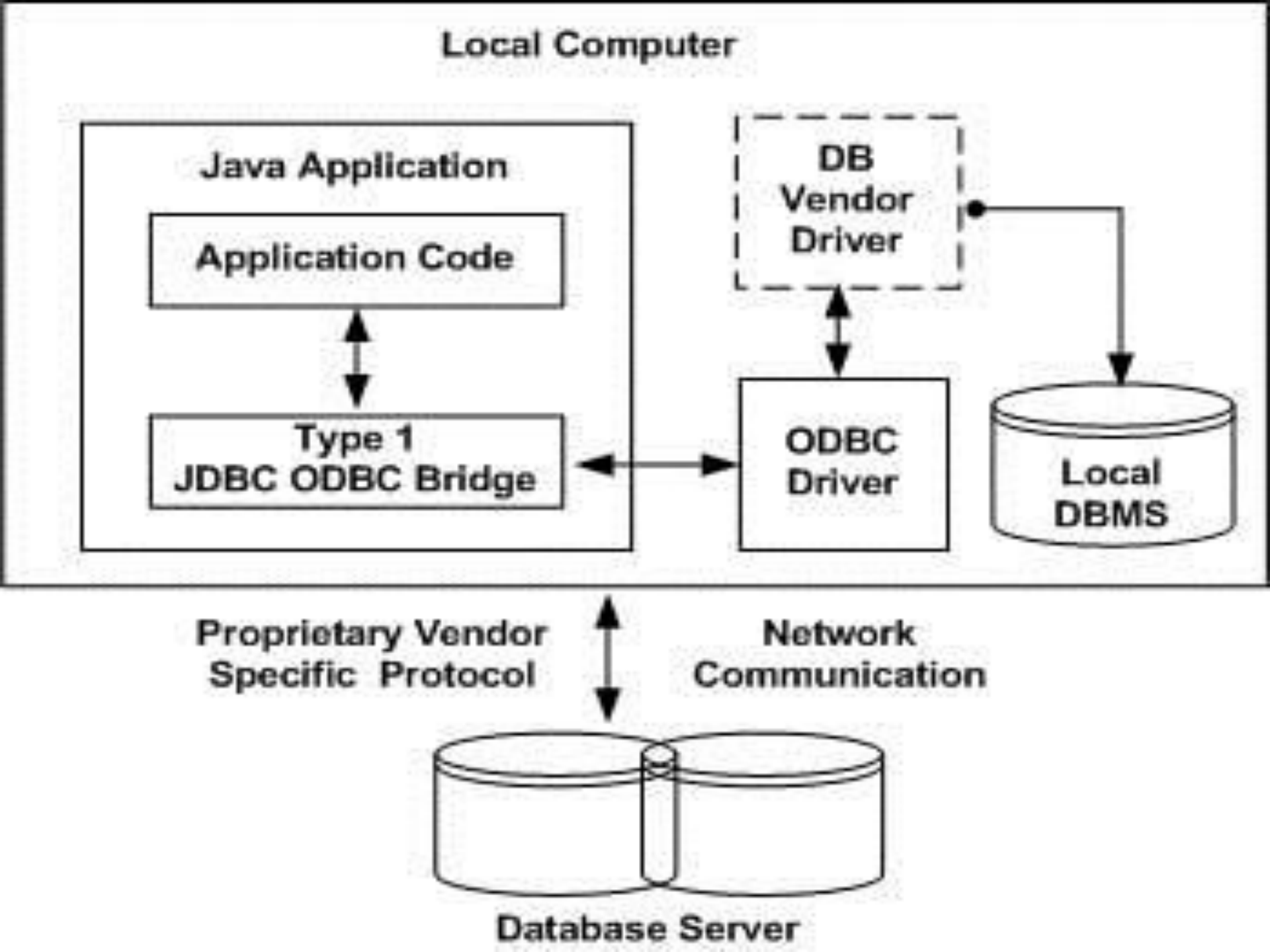
Type 4: 100% pure Java

Type 1: JDBC-ODBC Bridge Driver

- In a Type 1 driver, a JDBC bridge is used to access ODBC drivers installed on each client machine. Using ODBC requires configuring on your system a Data Source Name (DSN) that represents the target database.

- In a Type 1 driver, a JDBC bridge is used to access ODBC drivers **installed on each client machine**.

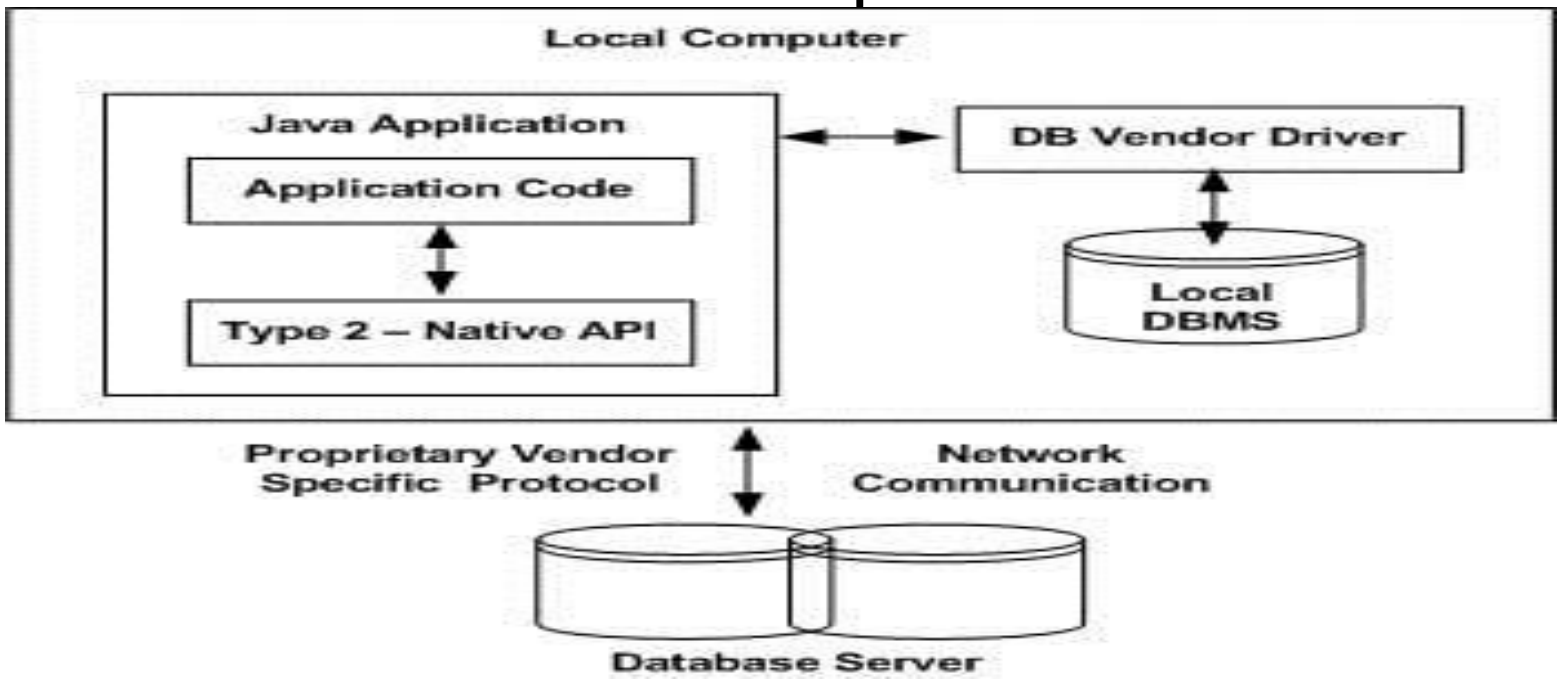
- Used for General Purpose.


Local Computer
Java Application
Application Code
Type 1
JDBC ODBC Bridge
DB
Vendor
Driver
ODBC
Driver
Local
DBMS
Proprietary Vendor
Specific Protocol
Network
Communication
Database Server

Type 2: JDBC-Native API

JDBC API calls are converted into native C/C++ API calls which are unique to the database.

If we change the Database we have to change the native API as it is specific to a database



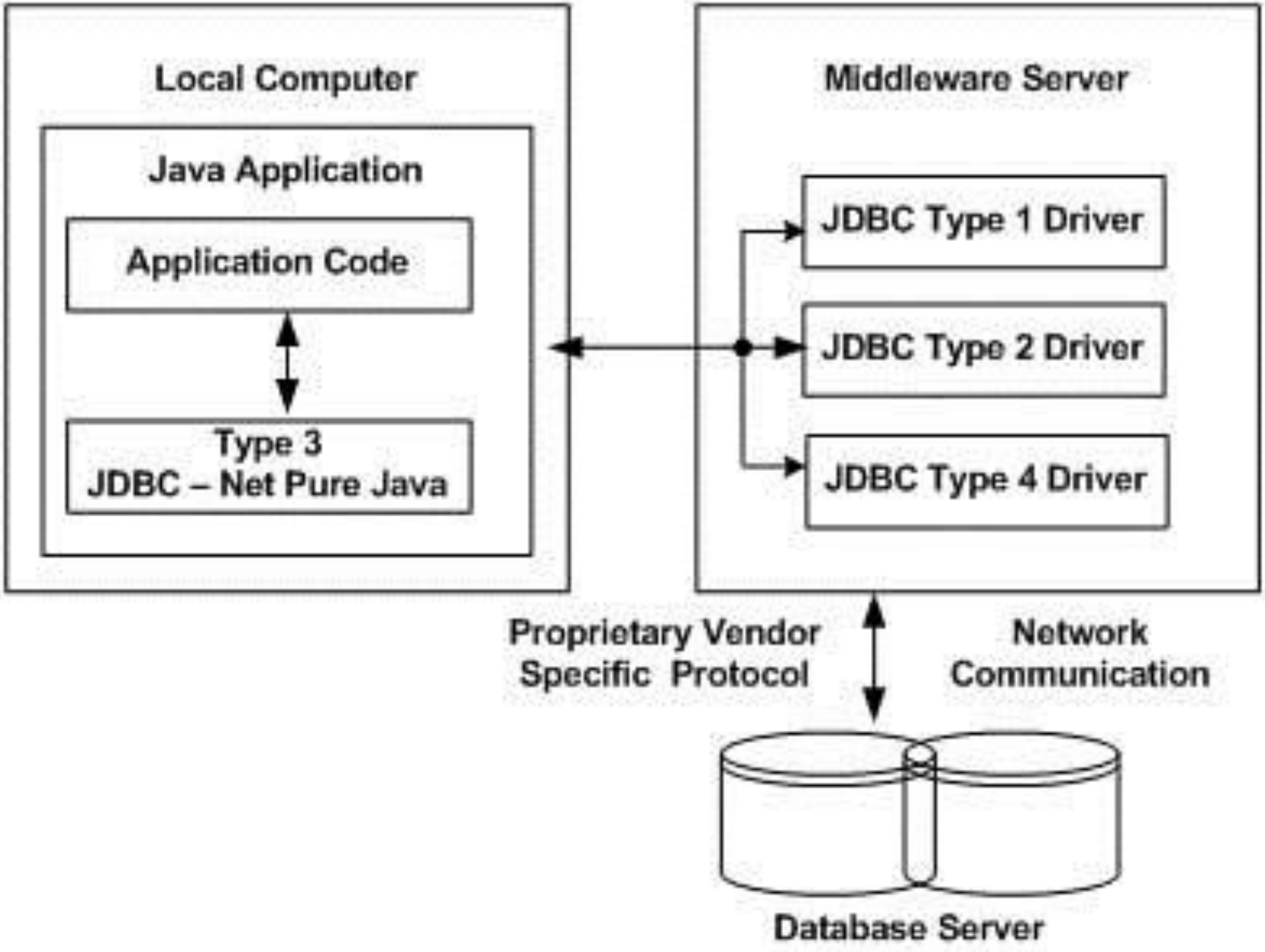
Type 3: JDBC-Net pure Java:

a three-tier approach is used to accessing databases.

1.The JDBC clients use standard network sockets to communicate with an middleware application server.

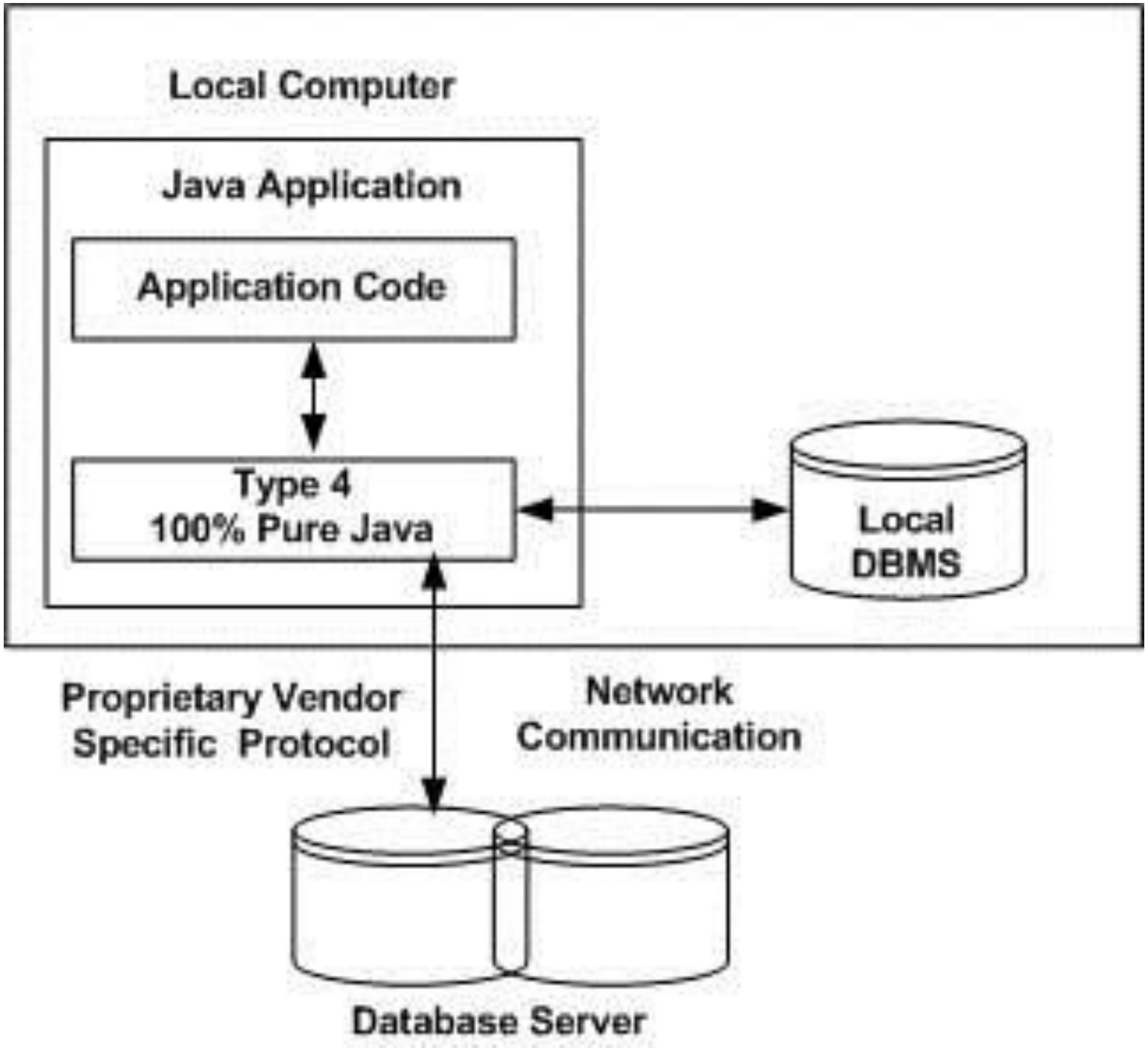
2. The socket information is then translated by the middleware application server into the call format required by the DBMS.

3. forwarded to the database server.


Local Computer
Java Application
Application Code
Type 3
JDBC – Net Pure Java
Middleware Server
JDBC Type 1 Driver
JDBC Type 2 Driver
JDBC Type 4 Driver
Proprietary Vendor
Specific Protocol
Network
Communication
Database Server

Type 4: 100% pure Java:

a pure Java-based driver that communicates directly with vendor's database through socket connection. This is the highest performance driver available for the database and is usually provided by the vendor itself.


Local Computer
Java Application
Application Code
Type 4
100% Pure Java
Local
DBMS
Proprietary Vendor
Specific Protocol
Network
Communication
Database Server

JDBC Connection

- **Import JDBC Packages:** Add **import** statements to your Java program to import required classes in your Java code.
- **Register JDBC Driver:** This step causes the JVM to load the desired driver implementation into memory so it can fulfill your JDBC requests.
- **Database URL Formulation:** This is to create a properly formatted address that points to the database to which you wish to connect.
- **Create Connection Object:** Finally, code a call to the *DriverManager* object's *getConnection( )*method to establish actual database connection.

1. import java.sql.*
2. Class.forName("oracle.jdbc.driver.OracleDriver”);

RDBMS	JDBC driver name	URL format
MySQL	com.mysql.jdbc.Driver	**jdbc:mysql:**//hostname/ databaseName
ORACLE	oracle.jdbc.driver.OracleDriver	**jdbc:oracle:thin:@**hostname:port Number:databaseName
DB2	COM.ibm.db2.jdbc.net.DB2Driver	**jdbc:db2:**hostname:port Number/databaseName
Sybase	com.sybase.jdbc.SybDriver	**jdbc:sybase:Tds:**hostname: port Number/databaseName

3.

String URL =
"jdbc:oracle:thin:@amrood:1521:EMP";

String USER = "username";

String PASS = "password“;

4. Connection conn =

DriverManager.getConnection(URL, USER, PASS);

Statement in JDBC

JDBC *Statement,CallableStatement,* and *PreparedStatement* interfaces define the methods And properties that enable you to send SQL or PL/SQL commands and receive data from your database.

1.Statement-Use for general-purpose access to your Database.

2. PreparedStatement -Use when you plan to use the SQL statements many times. The PreparedStatement interface accepts input parameters at runtime.

3.CallableStatement –Used with Stored Procedure

Statement Object

You can use a Statement object to execute a SQL statement, you need to create one using the Connection object's createStatement( ) method

Statement stmt = conn.createStatement( )

Methods

- **boolean execute(String SQL)** : DDL Statements
- **int executeUpdate(String SQL)** : Returns the numbers of rows affected for example, an INSERT, UPDATE, or DELETE statement.
- **ResultSet executeQuery(String SQL)** : Returns a ResultSet object. as you would with a SELECT statement.

PreparedStatement

PreparedStatement interface - statement gives you the flexibility of supplying arguments dynamically.

```
PreparedStatement pstmt = null;
String SQL = "Update Employees SET age = ?
    WHERE id = ?";
pstmt = conn.prepareStatement(SQL);
```

CallableStatement

execute a call to a database stored procedure.

```
CREATE OR REPLACE PROCEDURE getEmpName
   (EMP_ID IN NUMBER, EMP_FIRST OUT
   VARCHAR) AS BEGIN SELECT first INTO
   EMP_FIRST FROM Employees WHERE ID =
   EMP_ID; END;
CallableStatement cstmt = null;
String SQL = "{call getEmpName (?, ?)}";
 cstmt = conn.prepareCall (SQL);
```