0301502 ADVANCED JAVA

UNIT	MODULES	WEIGHTAGE
1	File Handling	20 %
2	Java Collection Framework	20 %
3	Event Handling, Swing and GUI Components	20 %
4	Swing, GUI Components and Layout Manager	20 %
5	Database Connectivity (JDBC)	20 %

UNIT - 5 Database connectivity (JDBC)

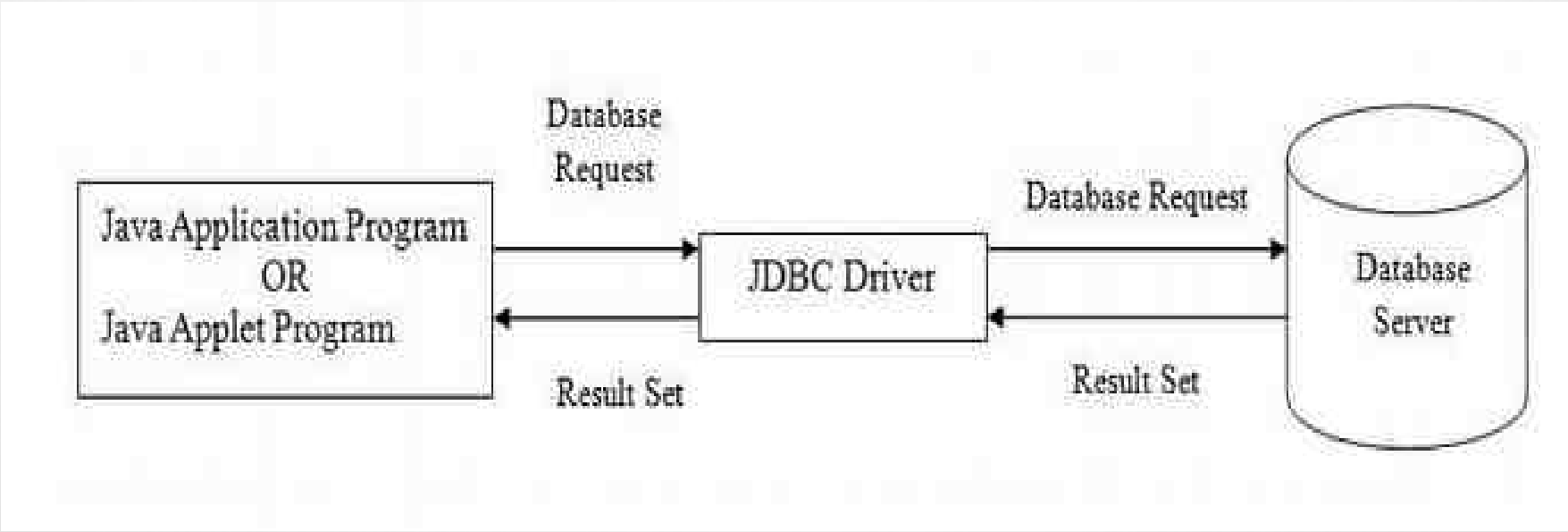
- Database Connectivity (JDBC)
- JDBC and ODBC
- Using a JDBC
- Driver Manager – Creating Connection
- Connection Interface – Creating Statement
- Types of Statement
- Statement Interface – Executing Statements
- Result Set Interface

UNIT – 5 Database Connectivity (JDBC)

- The terms JDBC is taken as an acronym for **Java Database Connectivity.**
- The **Java Application Program Interface** (API) makes a connection between **java application or applet and a database management system.**
- The different vender of DBMS or RDBMS has its own structure to organize its data. **Any application written to access a DBMS of one vender cannot be used to access the DBMS of another vendor.**
  - To solve this problem, **Microsoft developed a standard called "Open Database Connectivity (ODBC)".** Whic is free from any vendor specific DBMS structure.

UNIT – 5 Database Connectivity (JDBC)

- A **JDBC client does note make a direct link to a DBMS server. A JDBC make used of ODBC to connect to DBMS server.**
- The bridge between a **Java program and a database is a JDBC-ODBC driver.**
- JDBC **driver acts as the interface between a database and java application or applet.**



UNIT – 5 JDBC – ODBC – Types of Drivers

- **Types of Drivers**
  - **The drivers** supporting Java language are **classifed into four types.**
  - They are **classified based on the technique used to access a DBMS.**

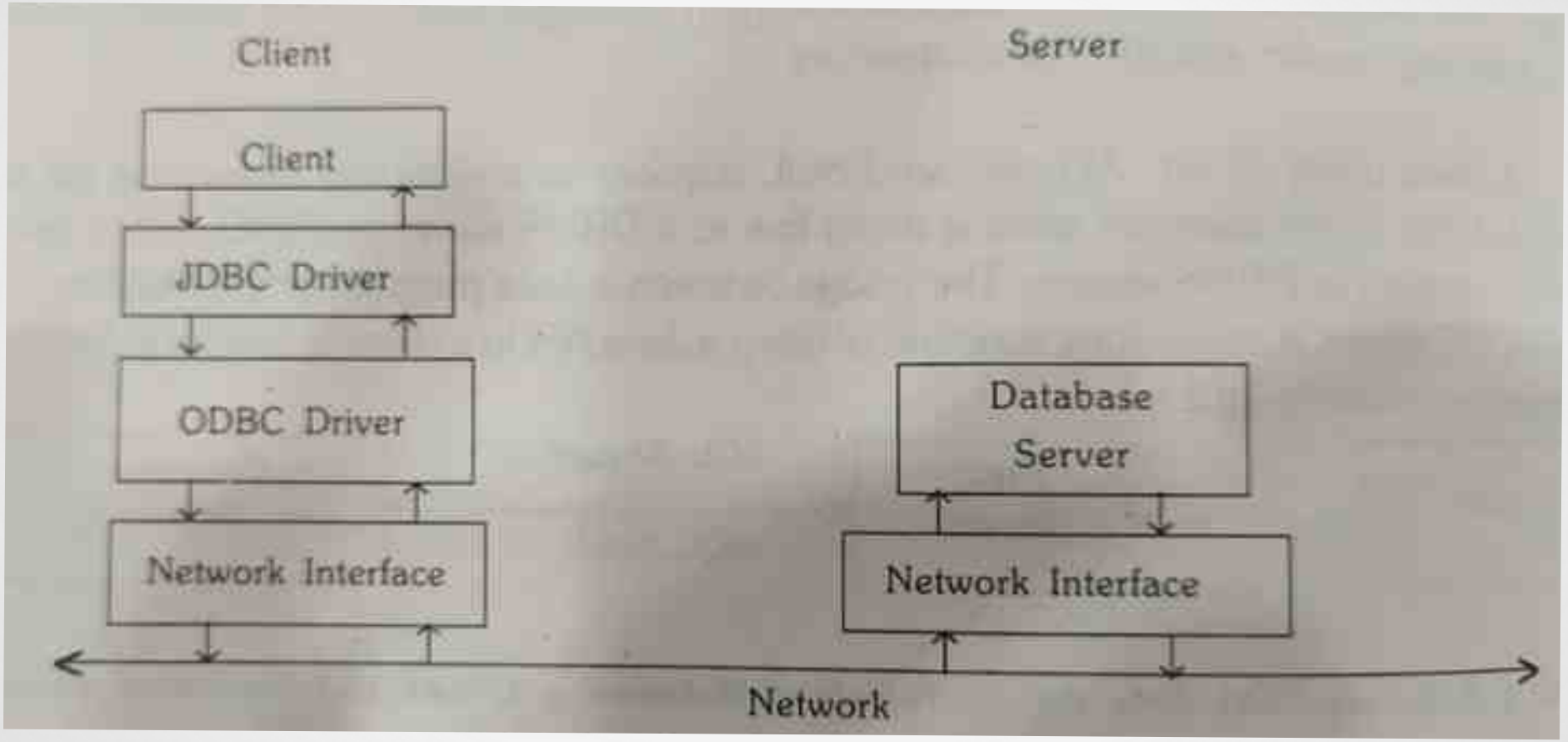
UNIT – 5 JDBC – ODBC – Types of Drivers

- **Types of Drivers**
  - **Type 1 : JDBC-ODBC Bridge Driver**
  - **Type 2 : Native API – Partly Java Driver**
  - **Type 3 : JDBC Net -All - Java Driver**
  - **Type 4 : Native-Protocol-All-Java Driver**

UNIT – 5 Type -I JDBC-ODBC Bridge Driver

- In this type, a **JDBC – ODBC bridge acts as an interface between a client and a database server.**
- An application in a **client makes use of the JAVA API to send the requests to a database to the JDBC-ODBC.**
- The **JDBC-ODBC bridge converts theJDBC API to ODBC API and sends its to the databse server.**
- The **reply obtained from the database server is sent to the client via JDBC-ODBC driver.**
- In this type, the **JDBC-ODBC driver has to be installed in the client side.**

UNIT – 5 Type -I JDBC-ODBC Bridge Driver



UNIT – 5 Type -II Native API – Partly Java Driver

- In this type of driver, the **JDBC requests are translated to the Call Level Interface (CLI) of the database installed in the client machine to communicate with a database.**
- When database receive the request, they are processed and send back.
- **This result in the native format of the database is converted to JDBC format and presented to the application running in a client.**
- This types of **driver offers a faster response than Type I** Drivers.

UNIT – 5 Type -II Native API – Partly Java Driver

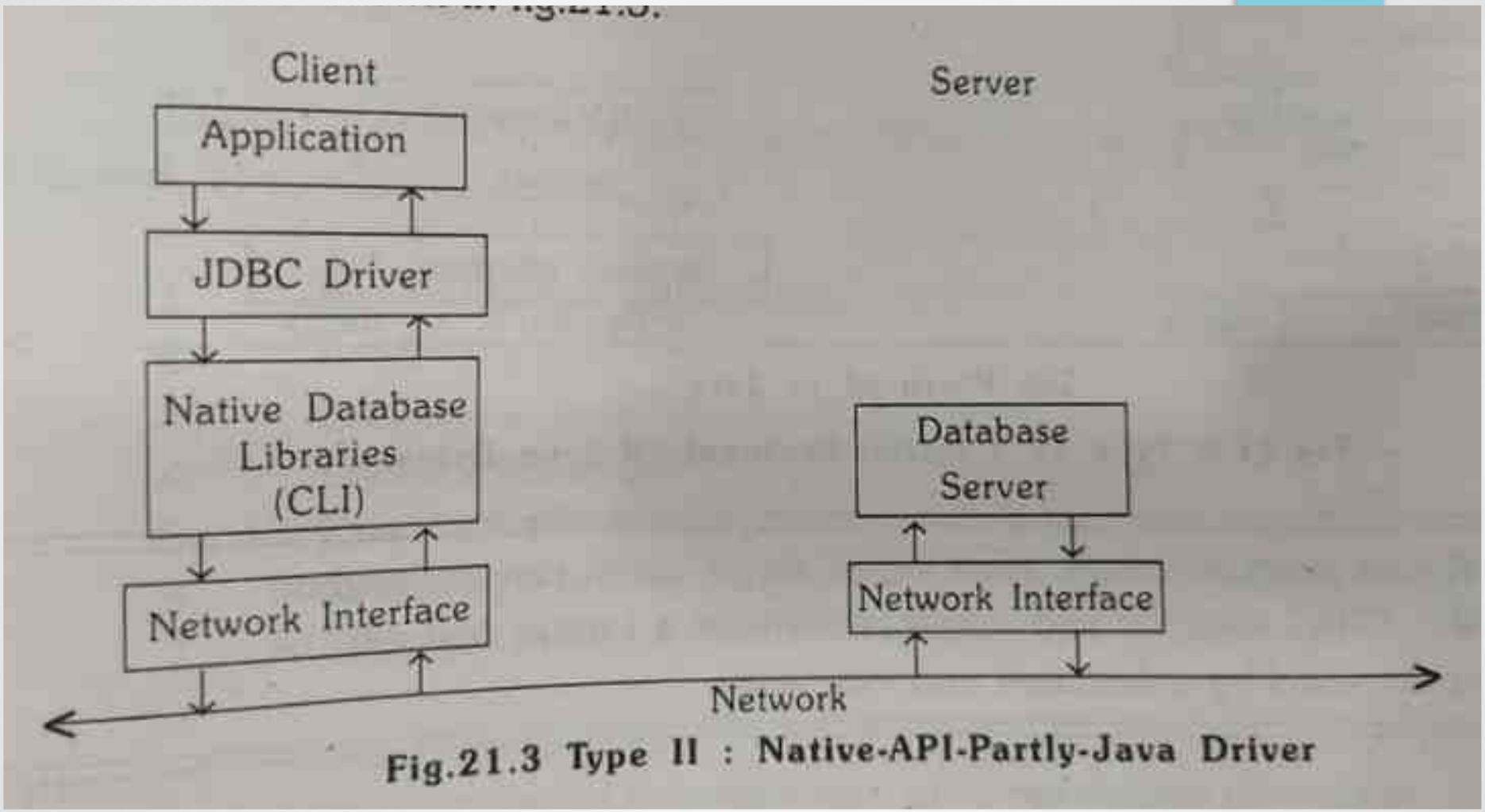


Fig.21.3 Type II : Native-API-Partly-Java Driver

UNIT – 5 Type -III JDBC Net ALL Java Driver

- It is similar to Type II Driver.
- The **only difference is that JDBC for server and Native Database Libraries (CLI) is stored in the remote server.**
- Communication between the client and the server takes place using a JDBC driver network protocol.

UNIT – 5 Type -III JDBC Net ALL Java Driver

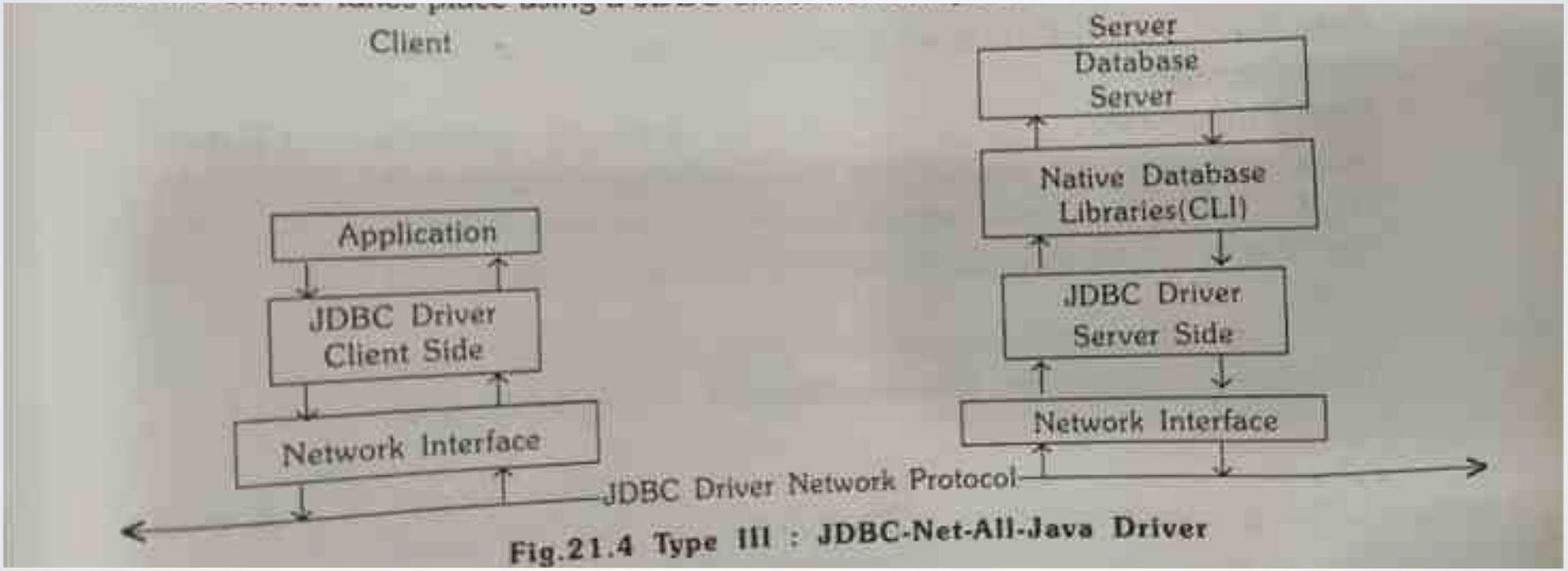
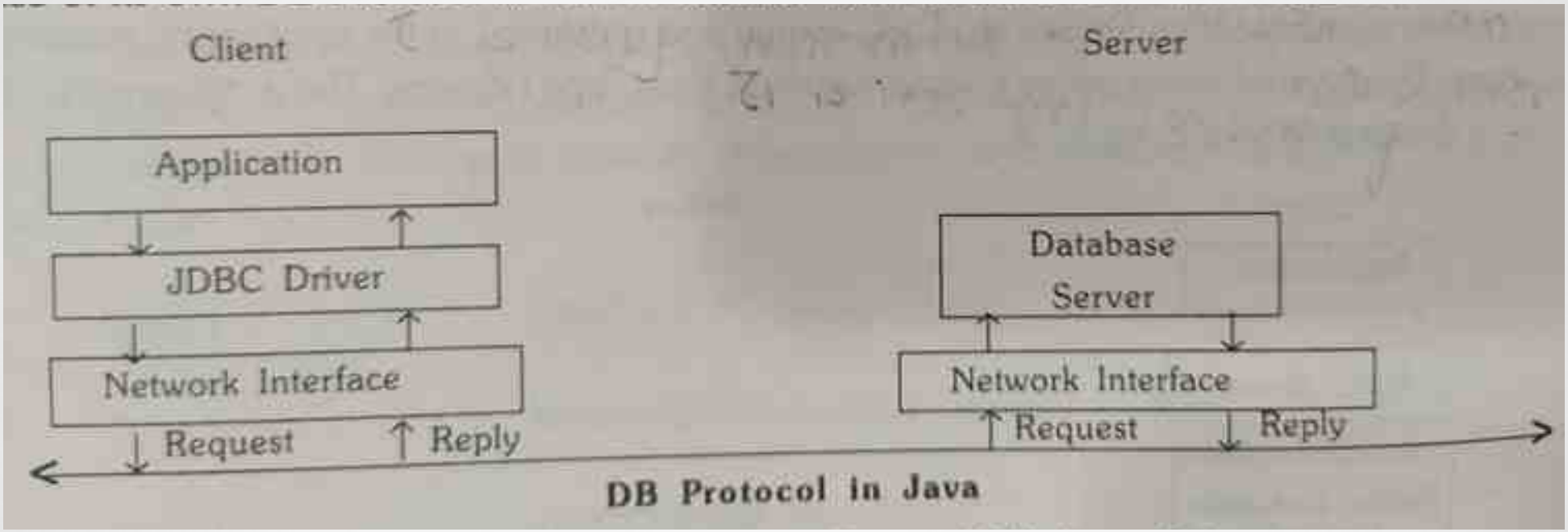


Fig.21.4 Type III : JDBC-Net-All-Java Driver

UNIT – 5 Type-IV Native-Protocol-All-Java Driver

- This type of a **driver is 100% java** and **does not make use of any CLI native libraries.**
- In this type, no translation takes place.
- A **JDBC makes a call directly to the database.**
- It makes use of its own DB protocol written in Java for Network Communication.

UNIT – 5 Type-IV Native-Protocol-All-Java Driver



UNIT – 5 JDBC-ODBC -Java SQL Package

- The classes required to handle a database are contained in a package:
  - ***Java.sql.****
- This package contains the follwoing **classes:**
  - *Date*
  - *DriverManager*
  - *DriverPropertyInfo*
  - *SQLPermission*
  - *Time*
  - *TimeStamp*
  - *Type*

UNIT – 5 JDBC-ODBC -Java SQL Package

- This package contains the follwoing **Interface**:

Array	Blob	ResultSet	ResultSetMetaData
CallableStatement	Clob	Savepoint	SQLData
Connection	DatabaseMetaData	SQLInput	SQLOutput
Driver	ParameterMetaData	Statement	Struct
PreparedStatement	Ref		

UNIT – 5 STEPS FOR JDBC CONNECTION

- Step 1 : Import the java.sql package
- Step 2 : Load Driver
- Step 3 : Establish Database Connection
- Step 4 : Create Statement
- Step 5 : Execute the statement
- Step 6 : Retrieve the results
- Step 7 : Close the connection and statement

UNIT – 5 STEPS FOR JDBC CONNECTION

- Step 1 : Import the sql package
  - The first step of importing java.sql is the routine procedure.
    - ***import java.sql.*;***
- Step 2 : Load the Driver
  - Loading of the driver is done using the following method:
    - ***Class.forName(“com.mysql.jdbc.Driver”);***

UNIT – 5 STEPS FOR JDBC CONNECTION

- Step 3 :Establish connection
  - Connection is established wiht the database using the following method defined in the DriverManager class:
    - ***Connection con = DriverManager.getConnection("jdbc:odbc:database");***
    - Where "*jdbc:odbc:database*" is the database URL object specifiying the protocol, subprotocol and the databasename.

UNIT – 5 STEPS FOR JDBC CONNECTION

- Step 4 :prepare statement
  - In this step, staatement object that is required to send a query to the database is prepared. This statement is prepared using the following method defined in Connection interface:
    - ***Statement stmnt = con.createStatement();***
- Step 5 :Execute query
  - The SQL query that is to be sent to the database is executed by calling the following method on statement object, which returns a ResultSet Object:
    - ***ResultSet reset = stmnt.executeQuery(“select * from database”);***

UNIT – 5 STEPS FOR JDBC CONNECTION

- Step 6 :Retrieve the result
  - The result returned by the database to the query executed are extracted from ResultSet using the get method defined in ResultSet interface:
    - ***while(reset.next())***

      System.out.println(reset.getString("name");
- Step 7 :Close the statement and connection
  - Using the close method, the Statement and Connection are close:
    - ***con.close();***
    - ***Stmnt.close();***

UNIT – 5 JDBC CONNECTION

- Examples :
  - Database_con.java

UNIT – 5 Driver Manager – Creating Connection

- The ***DriverManager*** **is class** contains **methods to manage the JDBC drivers loaded in program.**
- The JDBC drivers are loaded using the ***forName()*** method.
- This class has **no constructor, but has only static methods**.

UNIT – 5 Driver Manager – Methods

Method	Description
Static Connection getConnection(String url)	Create a connection to the specified database URL; throws SQLExecption
Static Connection getConnection(String url, Properties Prop)	Creates a connection to the specified database URL using the properties specfied; throw SQLException
Static Connection getConnection(String url, String user, String pswd)	Creates a connection to the specified database URL using the user name and password: throws SQLException
Static Driver getDriver(String url)	Selects a driver from the specified database url: throws SQLException

UNIT – 5 Driver Manager – Creating Connection

- **JDBC URLs**
  - **Database URLs have three components :**
    - ProtocolName
    - Sub-protocol
    - Subname
  - The syntax of JDBC URL is
    - ***<protocol>:<subprotocol>:<subname>***
  - Example
    - ***jdbc:mysql://localhost/sonoo***

UNIT – 5 Connection Interface – Creating Statement

- The java.sql package contains an **interface Connection.**
- **Connection object represent an SQL sessiona with database.**
- This **interface contains methods which can be used to prepare statements.**
- There are basically three types of statements
  - ***Statement***
  - ***PreparedStatement***
  - ***CallableStatement***

UNIT – 5 Connection Interface – Creating Statement

- **Statement**
  - A statement is **used to execute static SQL statements. There are no IN or OUT parameters.**
  - When an SQL sttatement is executed, only one result is returned.
- **PreparedStatement**
  - A PreparedStatement object is **used to execute dynamic SQL statement with IN paremeter.**
  - A PreparedStatement is **compiled once by the database. This used generally when large size of data retrive.**

UNIT – 5 Connection Interface – Creating Statement

- **CallableStatement**
  - A CallableStatement **objeect is used for executting a stored procedure** that can be used in an application.
  - A CallableStatement **contains contains an OUT parameter. It can also include IN parameter.**

UNIT – 5 Connection Interface – Creating Statement

Method	Description
Void close()	**Releases the Connection** object's database and JDBC resources.
Void commit()	**Makes all the changes made since the last commit or rollback** permanent; throws SQLException
Statement createStatement()	**Creates a Statement object for sending SQL statement to the databases**; throws SQLException
Boolean isClosed()	**Checks whether the connection is closed**
CallableStatement prepareCall(String sql)	Creates a **CallableStatement object for calling stored procedure**; throws SQLException

UNIT – 5 Connection Interface – Creating Statement

Method	Description
PreparedStatement prepareStatement(String sql)	Create a **PreparedStatement object for sending SQL statement with or without IN** parameter; throws SQLException
Void rollback()	**Undoes all changes made in the current transaction**

UNIT – 5 Statement Interface – Executing Statements

- The Statement object created is used for executing static SQL statement .
- Statement is the simplest one to execute an SQL statement.
- The Statement interface has several concrete methods to execute SQL statements.

UNIT – 5 Statement Interface – Executing Statements

Method	Description
Void close()	**Releases** the Statement object's database and JDBC resources.
Bollean execute(String sql)	**Executes the specified sql statement,** the result obtained is to be retrieved using getResultSet()
ResultSet executeQuery(String sql)	**Executes the given sql statement and returns one ResultSet**
Int executeUpdate(String sql)	**Executes the specified sql,** which may be INSERT, DELETE or UPDATE
Int getMaxRows()	**Returns the maximum number of rows** that the result set contains.
ResultSet getResultSet()	**Retrieves the ResultSet generated by the execute() method.**

UNIT – 5 Statement Interface

- Examples :
  - MysqlCon.java

UNIT – 5 PreparedStatement Interface

- A **PreparedStatement object can be used to execute a dynamic SQL** statement with IN parameter.
- A PreparedStatement **can be precompiled and used repeatedly**.
- **PreparedStatement object is created using PreparedStatement() method in Connection class.**

UNIT – 5 PreparedStatement Interface - Methods

Method	Description
Boolean execute()	**Execute the SQL statement** in this object, one must use getResult() method to retrieve the result.
ResultSet executeQuery()	Execute the SQL statement in this object and returns ResultSet object.
Int executeUpdate()	Execute the SQL statement, it must be insert,update and delete statement
ResultSetMetaData getMetaData()	Retrieves a resultsetmetadata object that contains information about the columns.
Void setBigDecimal(int index, BigDeimal x)	Sets the parameter specified by the index to the BigDecimal value.

UNIT – 5 PreparedStatement Interface - Methods

Method	Description
Void setBlob(int index, Blob x)	Sets the specifed parameter to the given Blob object.
Void setBoolean(int index, boolean x)	Sets the specifiecd parameter to the boolean value.
Void setByte(int index, byte x)	Sets thespecified parameter to the byte value.
Void setClob(int index, Clob x)	Sets the specified parameter to the Clob object.
Void setDate(int index, Date x)	Sets the specified parameter to the Date value.

UNIT – 5 PreparedStatement Interface - Methods

Method	Description
Void setDouble(int index, double x)	Sets the specified parameter to the double value.
Void setFloat(int index, float x)	Sets the specified parameter to the float value.
Void setInt(int index, int x)	Sets the specified parameter to the int value.
Void setLong(int index, long x)	Sets the specified parameter to the value.

UNIT – 5 PreparedStatement Interface - Methods

Method	Description
Void setObject(int index, Object x)	Sets the specified parameter to the object.
Void setShort(int index, short x)	Sets the specified parameter to the short value.
Void setString (int index, String x)	Sets the specified parameter to the String value.

UNIT – 5 CallableStatement Interface

- A CallableStatement object is **used to execute SQL stored procedures defined in the RDBMS.**
- A procedure with OUT parameter can be executed only in this CallableStatement.
- An OUT parameter in the stored procedure is **represented by the ?.**
- An **OUT parameter is registered using the registerOutParameter()** method. This method declares what the type of the OUT parameter is
- A CallableStatement can also contai**n IN parameter**

UNIT – 5 CallableStatement Interface

- A **CallableStatement interface has several methods inherited from Statement and PreparedStatement interfaces and some of its own.**

UNIT – 5 CallableStatement Interface - Methods

Method	Description
BigDecimal getBigDecimal(int index)	Retrieves the OUT parameter of JDBC NUMERIC type at the specified index
Byte getByte(int index)	Retrieves the OUT parameter of JDBC NUMERIC type at the specified index location as a byte.
Date getDate(int index)	Retrieves the OUT parameter of JDBC DATE type at the specified index location as a date.
Double getDouble(int index)	Retrieves the OUT parameter of JDBC DOUBLE at the specified index location as a double
Float getFloat(int index)	Retrieves the OUT parameter of JDBC FLOAT at the specified index location as a float.

UNIT – 5 CallableStatement Interface - Methods

Method	Description
Int getInt(int index)	Retrieves the OUT parameter of JDBC INTEGER at the specified index location as an int.
Long getLong(int index)	Retrieves the OUT parameter of JDBC LONG at the specified index location as an int
String getString(int index)	Retrieves the OUT parameter of JDBC CHAR, VARCHAR or LONGVARCHAR at the specified index location as a string.

UNIT – 5 ResultSet Interface

- The **executeQuery() and getResultSet() when called on Statement, PreparedStatement and CallabesStatement return objects of type ResultSet.**
- The ResultSet objects **contain results after the execution** of SQL statements.
- The ResultSet object **maintains a cursor pointing to the current row of results.**

UNIT – 5 ResultSet Interface - Methods

Method	Description
Boolean absolute(int row)	Moves the cursor to the specified row number in this result set
Void afterLast()	Moves the cursor to the end of the result set just after the last row
Void close()	Releases the object's database
Void deleteRow()	Deletes the current row of this result set
Boolean first()	Moves the cursor to the first row

UNIT – 5 ResultSet Interface - Methods

Method	Description
BigDecimal getBigDecimal(int columnIndex)	Retrieves the value of the psecified column as BigDecimal
Boolean getBoolean(int columnIndex)	Retrieves the value of the specified column name as boolean
Boolean getBoolean(String columnName)	Retrieves the value of the specified column name as boolean
Byte getByte(int columnIndex)	Retrieves the value of the specified column as a byte
Byte getByte(String columnName)	Retrieves the values of the specified column name as a byte.

UNIT – 5 ResultSet Interface - Methods

Method	Description
Date getDate(int columnIndex)	Retrieves the value of the specifed column as a Date
Date getDate(String columnName)	Retrieves the vaue of the specified column name as a Date
Double getDouble(int columnIndex)	Retrieves the vaue of the specified column name as a double
Double getDouble(string columnName)	Retrieves the vaue of the specified column name as a double
RealtSetMetaData getMetaData()	Returns the properties of the ResultSet object

UNIT – 5 ResultSet Interface - Methods

Method	Description
Double getFloat(int columnIndex)	Retrieves the vaue of the specified column name as a Float
Double getFloat(string columnName)	Retrieves the vaue of the specified column name as a Float
Double getInt(int columnIndex)	Retrieves the vaue of the specified column name as a Int
Double getInt(string columnName)	Retrieves the vaue of the specified column name as a Int
Int getRow()	Returns the current row number

UNIT – 5 ResultSet Interface - Methods

Method	Description
Double getLong(int columnIndex)	Retrieves the vaue of the specified column name as a Long
Double getLong(string columnName)	Retrieves the vaue of the specified column name as a Long
Statement getStatement()	Returns the statement object which produced the ResultSet
String getString(int ColumnIndex)	Retrieves the vaue of the specified column name as a Stirng
String getString(String ColumnIndex)	Retrieves the vaue of the specified column name as a String

UNIT – 5 ResultSet Interface - Methods

Method	Description
Boolean isFirst()	Checks whether the cursor is in first row.
Boolean isLast()	Checks whether the cursor is in Lasat row.
Boolean Last()	Moves the cursor to the last row.
Boolean next()	Moves the cursor to the next row.
Boolean previous()	Moves the cursor to the previous row.

UNIT – 5 JDBC Example

- CUI Examples :
  - Data_Insert.java
  - Data_Insert_2.java
  - Data_Insert_3.java
  - Data_Delete.java
  - Data_Update.java

UNIT – 5 JDBC Example

- GUI Examples :
  - InsertExample .java
  - DisplayExample.java
  - DeleteExample.java