JDBC (Java DataBase Connectivity)

JDBC = similar to ODBC (Open DataBase Connectivity)

It acts like a wrapper to let you feed SQL requests to the server.

∴ It is an Java's API to deal with structured data

What does JDBC do

1) It establishes a connection
2) Sends SQL statements
3) Process the results

eg:
```
Connection con = DriverManager.getConnection(
        "jdbc:odbc:wombat", "login", "password");
Statement s = con.createStatement();
ResultSet rs = s.executeQuery("SELECT a, b, c
                               FROM Table1");
while (rs.next())
{
    int x = getInt("a");
    String s = getString("b");
    float f = getFloat("c");
```

JDBC versus ODBC

→ ODBC is not appropriate for direct use with JAVA since it's build in C.

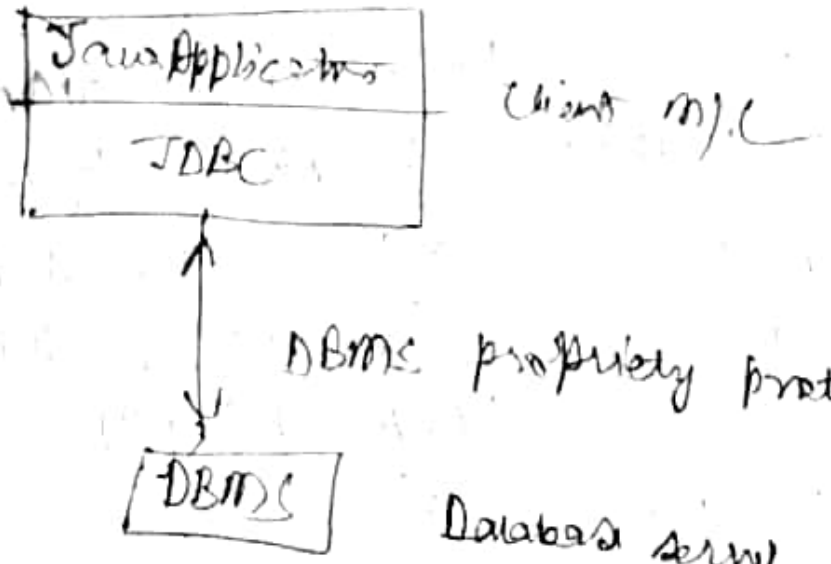
→ It's not feasible to translate ODBC API to Java API

→ ODBC is difficult to learn.

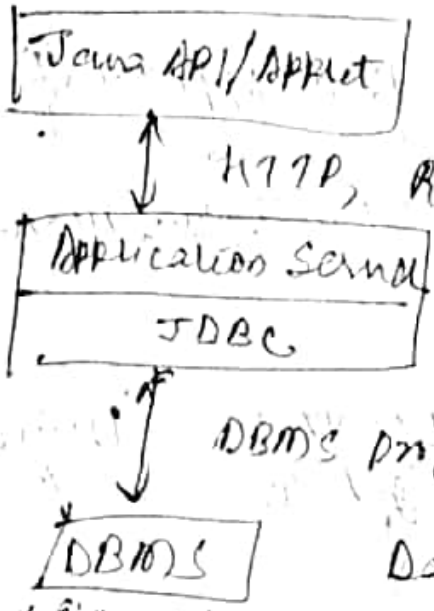
Hence one must use JDBC-ODBC bridge to connect to database, instead of ODBC, while working with JAVA API

Connectivity Model

→ 2-Tier Model



3-Tier Architecture.



JDBC drivers (driver: translates the API calls into operations for a specific data source)

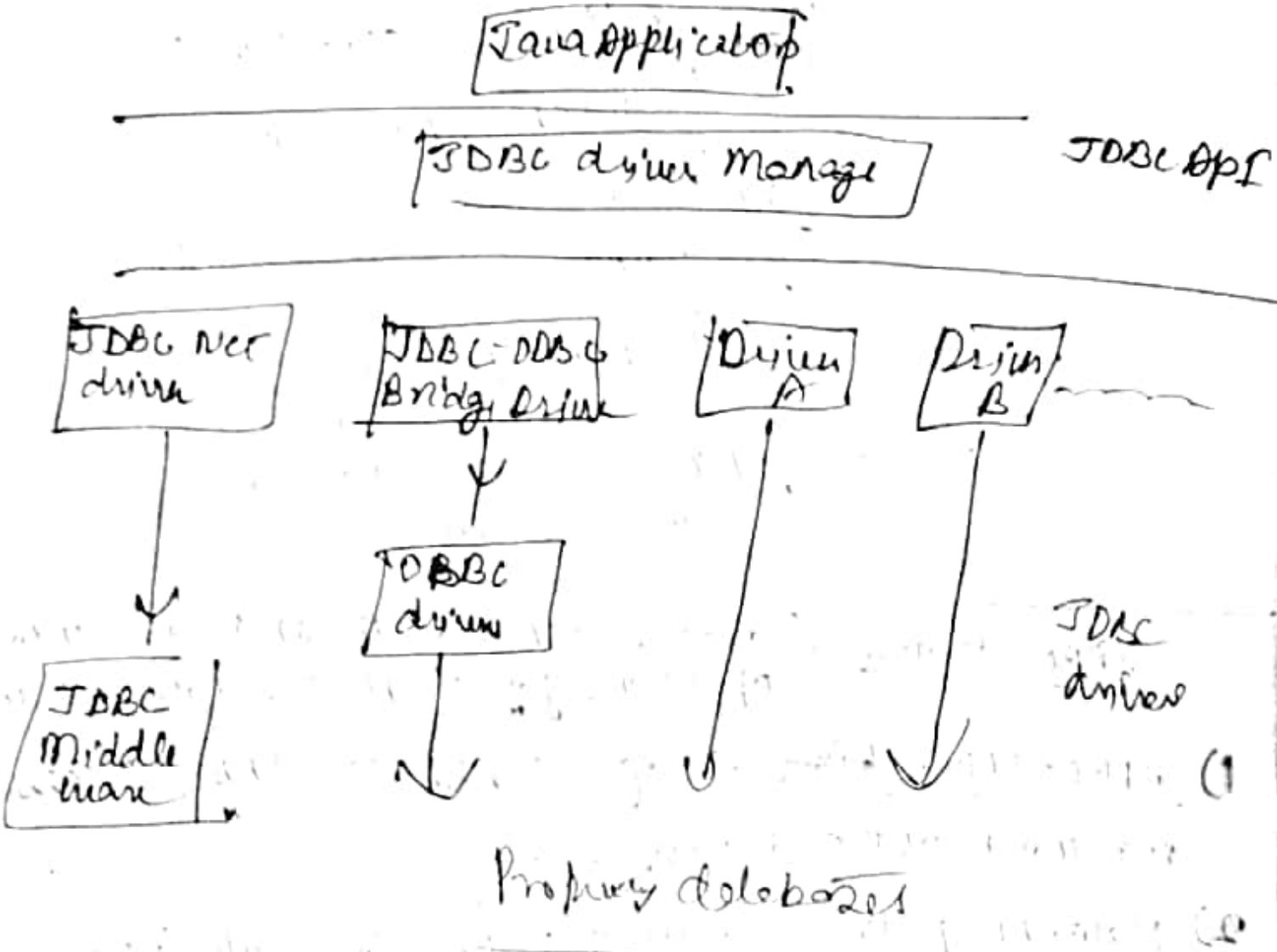
1) JDBC-ODBC bridge: provides JDBC access via most ODBC drivers.

2) Native API partly-Java driver: used for Oracle, Sybase, Informix, DB2 or other DBMS

3) Net protocol all Java driver: translates JDBC calls into a DBMS-independent net protocols, which is translated to a DBMS protocol by a server.

4) Native-protocol all Java driver: converts JDBC calls into the network protocol used by DBMS's directly.

JDBC Architecture



INTEGRAL UNIVERSITY, LUCKNOW

Ist / IInd Mid Semester Test-200.....

ANSWER SHEET

B

Name .... Roll No ....

Subject .... Branch .... Semester ....

Section .... Room No. .... Date .... Student's Sign ....

Invig. Sign.

(2)

Essential JDBC prg (Steps Imp)

Steps to be followed while building a JDBC application

1. java.sql.* pkg is imported
2. load and register the driver
3. connection is established to a database
4. Create a statement (SQL query)
5. If the statement is simple one (eg: CREATE TABLE) then simply exe it, otherwise receive the result of the query in a ResultSet & process it one tuple at a time.
6. Close the statement
7. Commit the transaction in the connection & close the connection

How to load a JDBC driver

complete pkg name of the driver

```
try {
    Class.forName("sun.jdbc.odbc.
                   JdbcOdbcDriver");
}
catch (Exception e)
{
    System.err.println("Unable to load driver");
}
```

Establishing a Connection

→ You establish connection by calling DriverManager.getConnection()

```
Try {
    Connection c;
    String db = "jdbc:odbc:DemoDataSource";
    String username = "";
    String pswd = "";
    c = DriverManager.getConnection(db, username, pswd);
    S.o.p("connected to database");
}
catch (SQLException e)
{
    S.o.p(e.getMessage());
    S.o.p(e.getSQLState());
    S.o.p(e.getErrorCode());
```

Create A Statement

your query is executed by object of Statement class

eg: Statement S = C.CreateStatement();

Execute the Statement

Execution is done by calling Statement.executeQuery() method. The executeQuery() method takes SQL query as an argument and returns the result of the query in a ResultSet object.

eg: ResultSet obj = S.executeQuery("SELECT e.Name, e.Salary" + " " + "FROM Emp e" + " " + "WHERE e.Dept = 'Toys'");

Process the ResultSet

The ResultSet consists of tuples and returns one tuple at a time when the next() fun" is applied.

```
eg: int count=0; double payroll=0.0;
    String name; double salary=0.0;
    while (obj.next()) ← till 'true'
    {
      name = obj.getString("Name");
      salary = obj.getInt("Salary");
      S.o.p (name + salary);
```

count = count + 1;
payroll = payroll + salary;
}

Close the statement

s.close();

Close the Connection

c.commit(); → to commit all transaction done in this connection.
c.close();