JDBC & Tools

Java Data Base Connectivity

Module 1

Agenda

1 Introduction to JDBC

2 Establishing Connection

3 Executing Query

4 Process Result

5 Callable Statement

Objectives

At the end of this module, you will be able to:

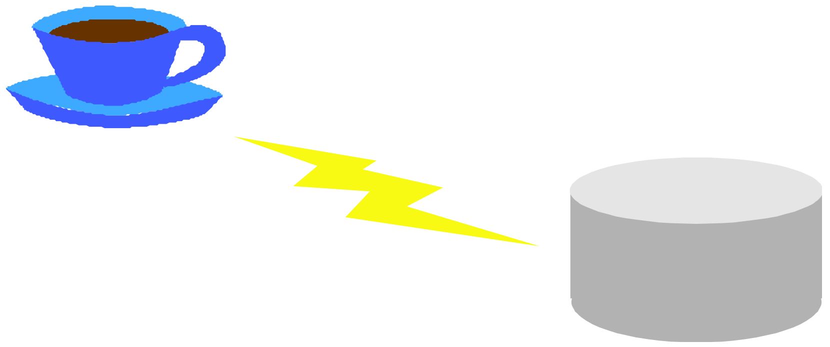
- Explain how to connect to a database using Java Database Connectivity (JDBC).
- Create and execute a query using JDBC API.
- Analyze how to use the Metadata objects to retrieve more information about the database or the result set.
- Know the function of commit and roll back in transactions.

Introduction to JDBC

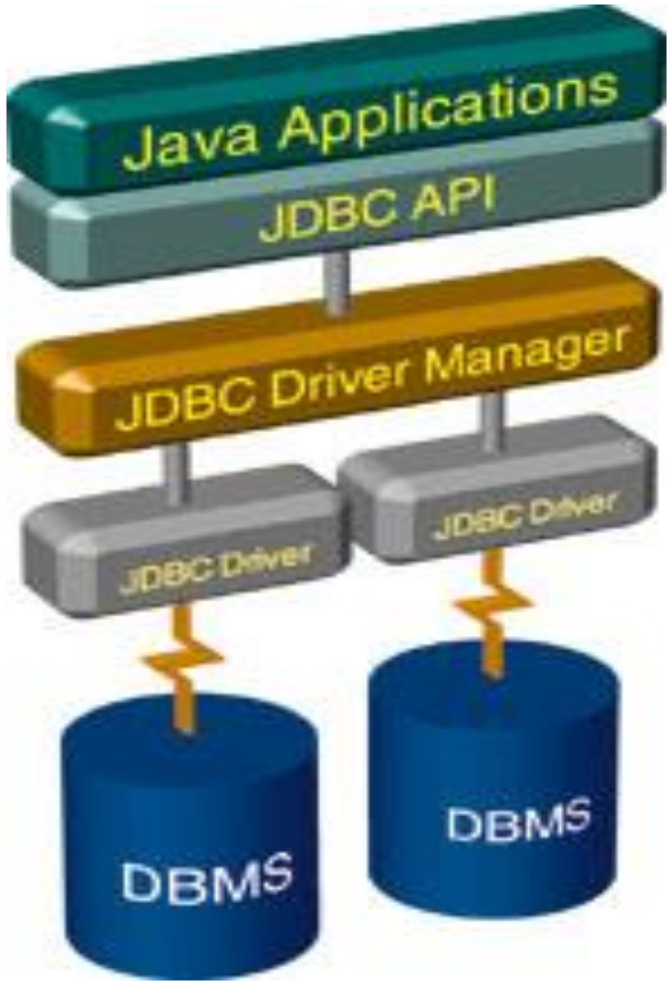
Introduction to JDBC

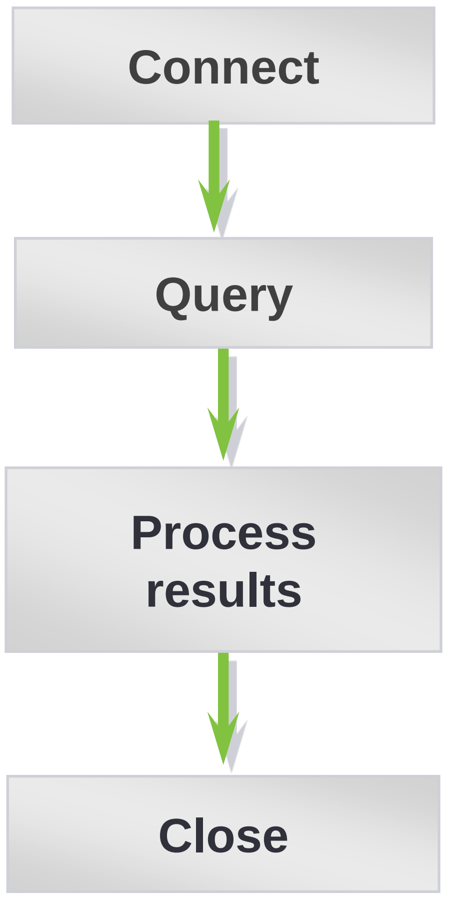
- JDBC is an API that helps a programmer to write java programs to connect to any database, retrieve the data from the database.

- java.sql package contains a set of interfaces that specify the JDBC API

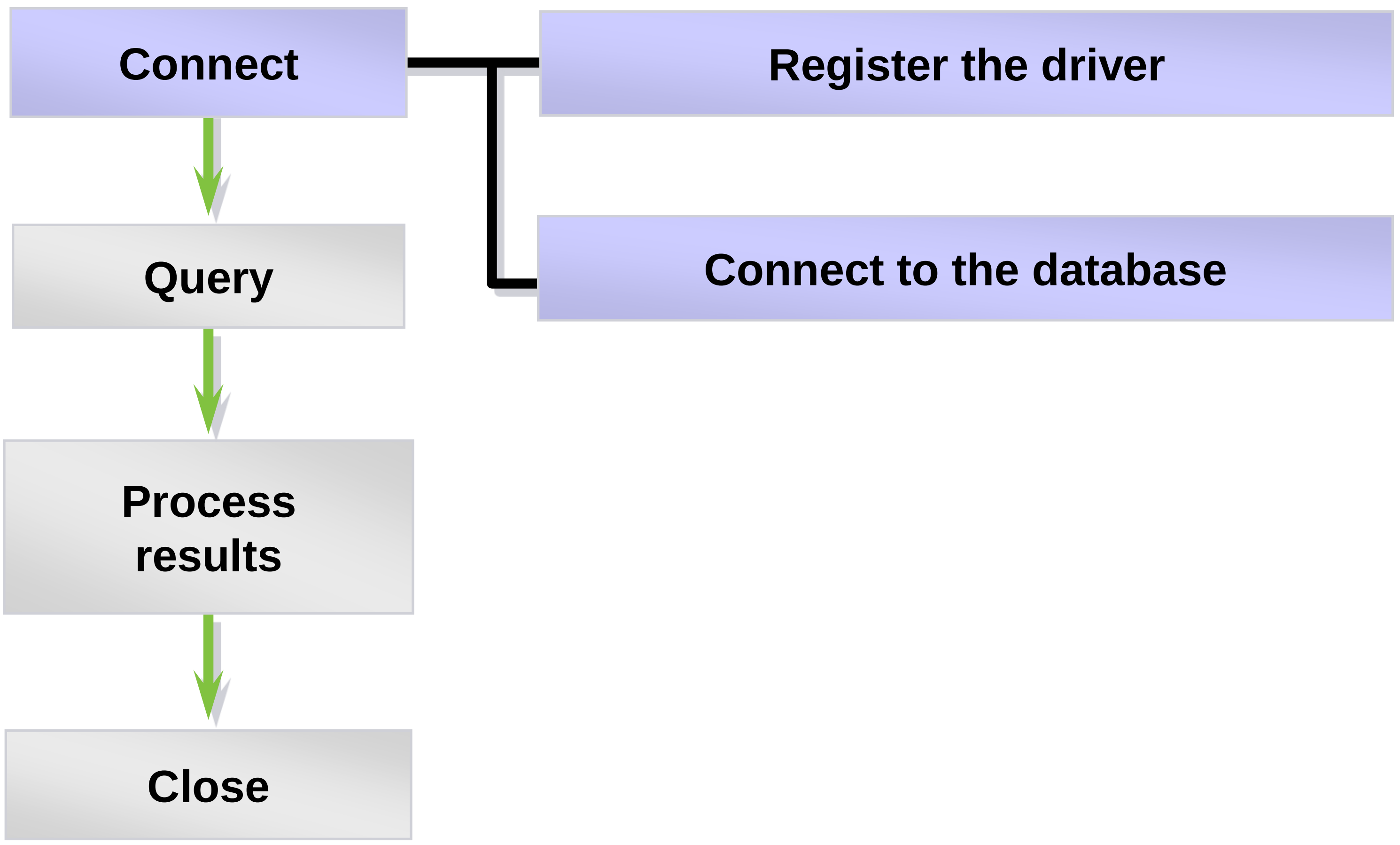


Architecture and Querying with JDBC



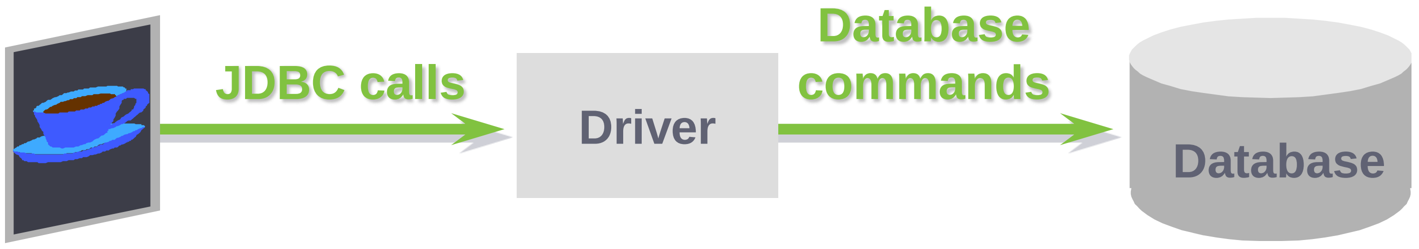


Stage 1: Connect



Connect: A JDBC Driver

- Is a set of classes and interfaces, written according to JDBC API to communicate with a database.

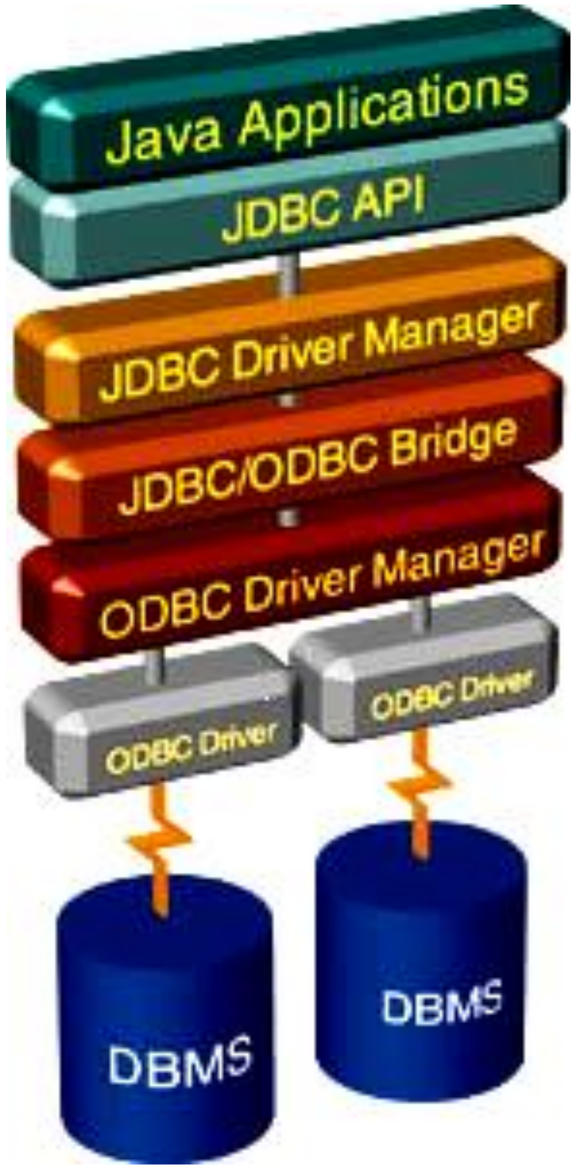


- Can also provide a vendor’s extensions to the JDBC standard

JDBC Driver (Contd.).

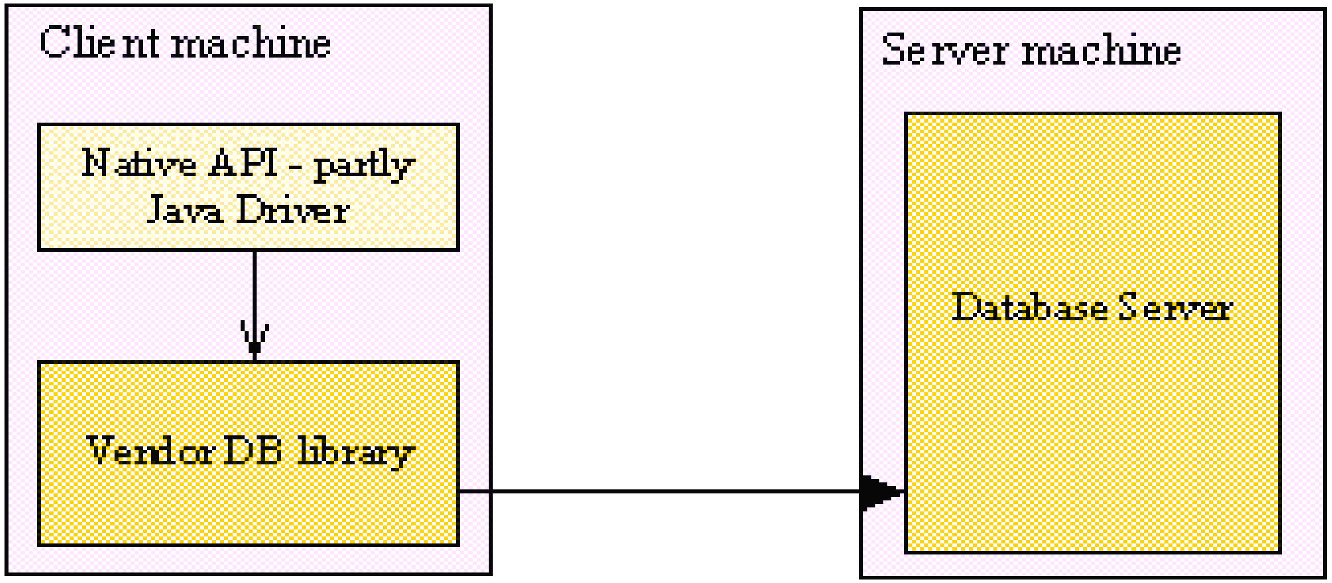
JDBC-ODBC Bridge Driver

(Type I Driver)



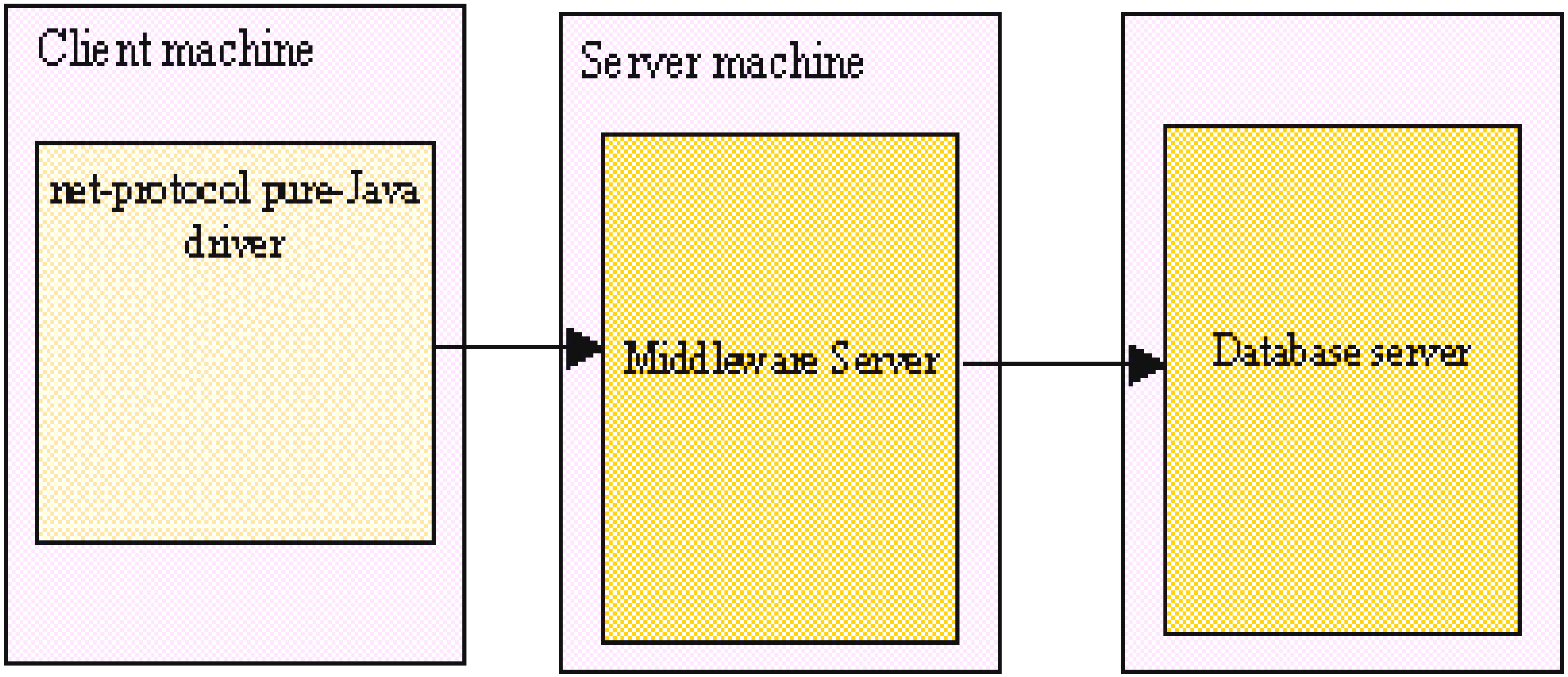
JDBC Driver (Contd.).

Native JDBC Driver (Type II Driver)



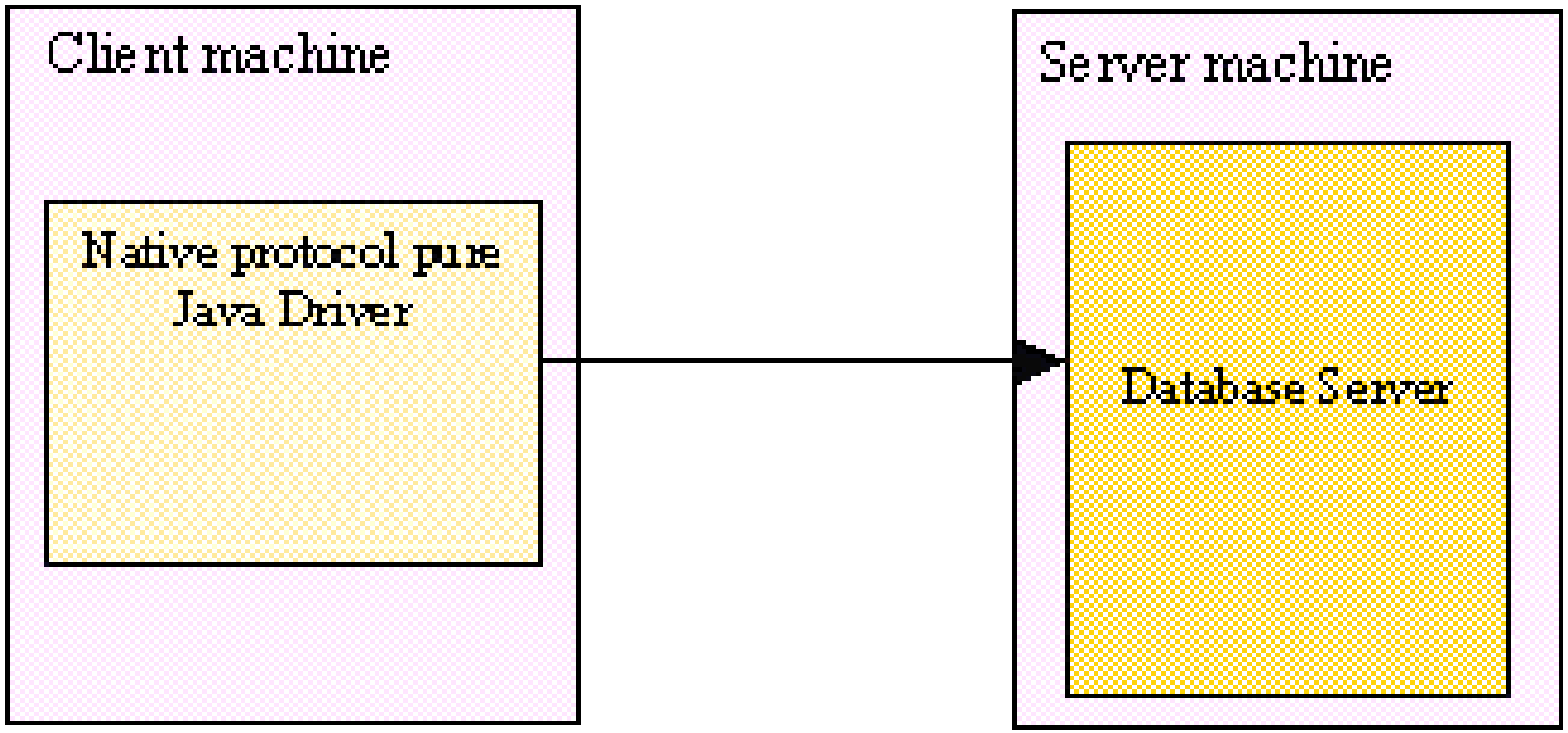
JDBC Drivers (Contd.).

All Java JDBC Net Drivers (Type III Driver)



JDBC Drivers (Contd.).

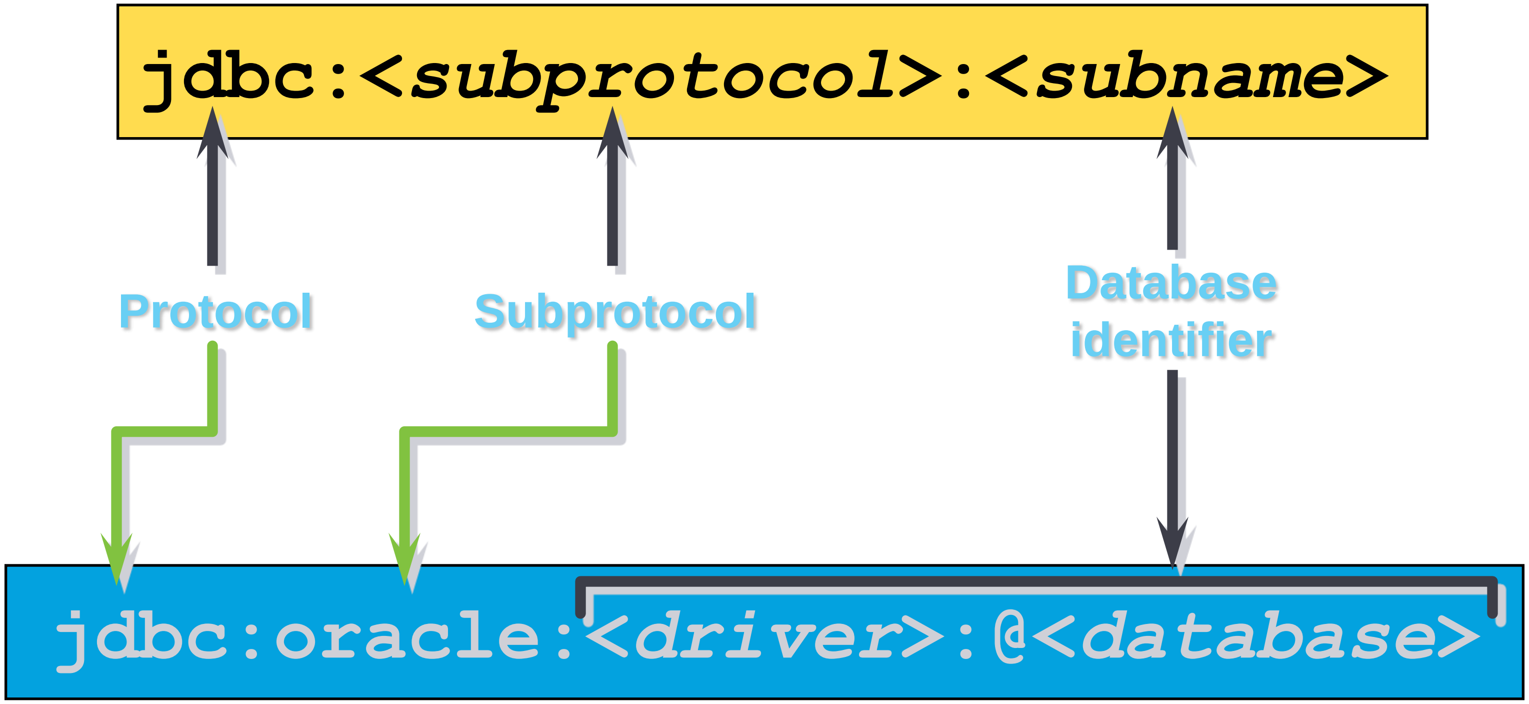
Native Protocol All Java Drivers (Type IV Driver)



Establishing Connection

Connect: About JDBC URL

URL represents a protocol to connect to the database


jdbc:<subprotocol>:<subname>
Protocol
Subprotocol
Database identifier
jdbc:oracle:<driver>:@<database>

JDBC URLs: Examples

- To connect to database using Sun jdbc-odbc driver

```
jdbc:odbc:jdbcoodbcDriverDsn
```

- To connect to oracle using thin driver provided by Oracle

```
jdbc:oracle:thin:@<TNSNAMES entry>
```

How to make the Connection?

1. To register the driver is to send the driver class name as parameter for Class.forName() method

```
Class c = Class.forName("oracle.jdbc.driver.OracleDriver");
Class c = Class.forName(”sun.jdbc.odbc.JdbcOdbcDriver");
```

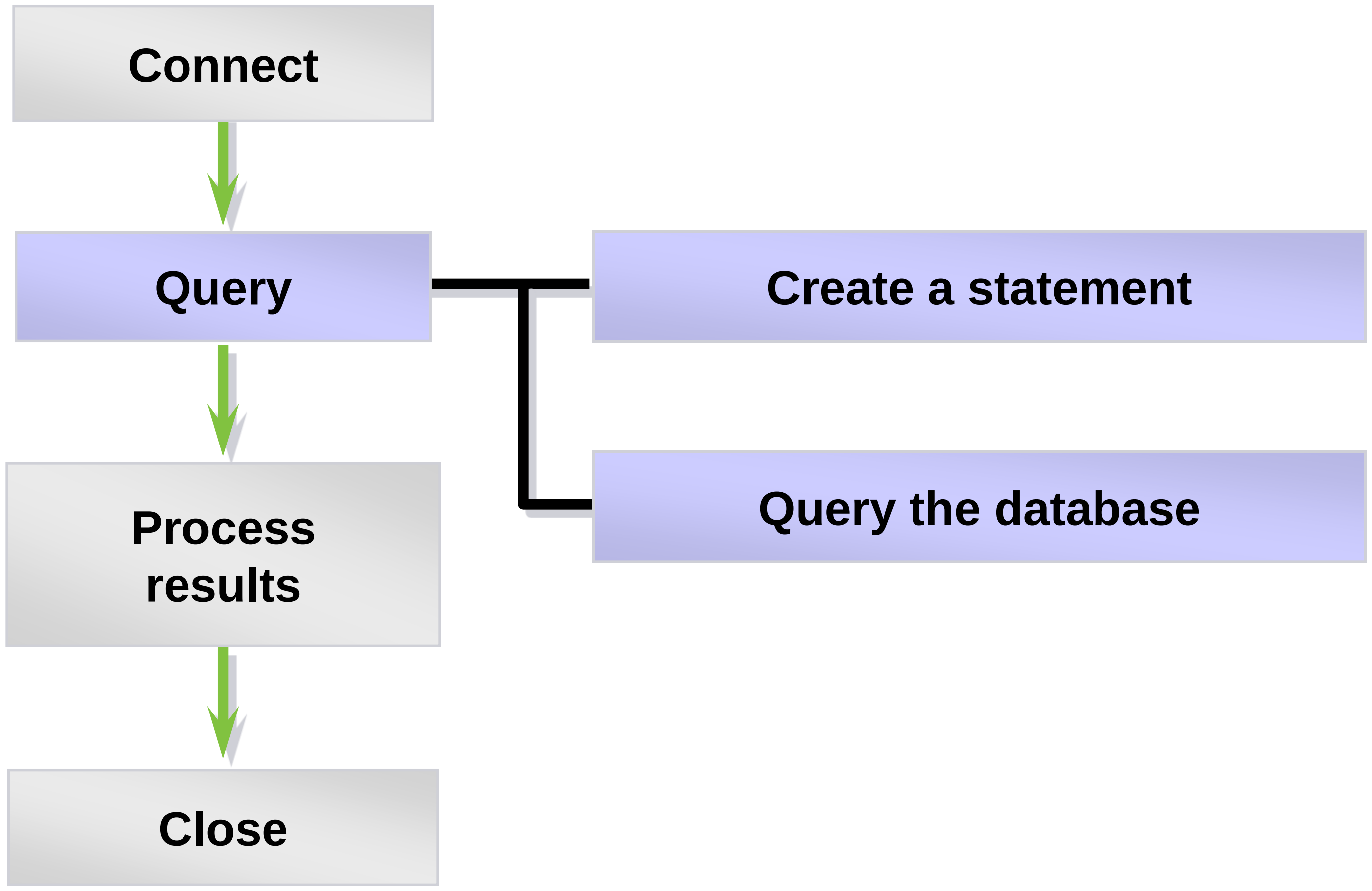
2. To connect to a database use getConnection() method

Connection
conn=DriverManager.getConnection(*URL*,*userid*,*password*);

```
Connection conn = DriverManager.getConnection
("jdbc:oracle:thin:@myhost:1521:orcl", "scott", "tiger");
```

Executing Query

Stage 2: Query



Query: The Statement Object

- To execute SQL statements use Statement Object.
- You need an active connection to create a JDBC statement
- Statement object has three methods to execute a SQL statements:
  - executeQuery() for SELECT statements
  - executeUpdate()for INSERT, UPDATE, DELETE, or DDL statements
  - execute() for either type of statement

How to Query the Database?

1. To execute SQL statement , we should first create Statement object, as:

```
Statement stmt = conn.createStatement();
```

2. To execute the query on the database

```
ResultSet rset = stmt.executeQuery(statement);
int count = stmt.executeUpdate(statement);
boolean isquery = stmt.execute(statement);
```

Querying the Database: Examples

- Following Statements are used to execute Select statement:

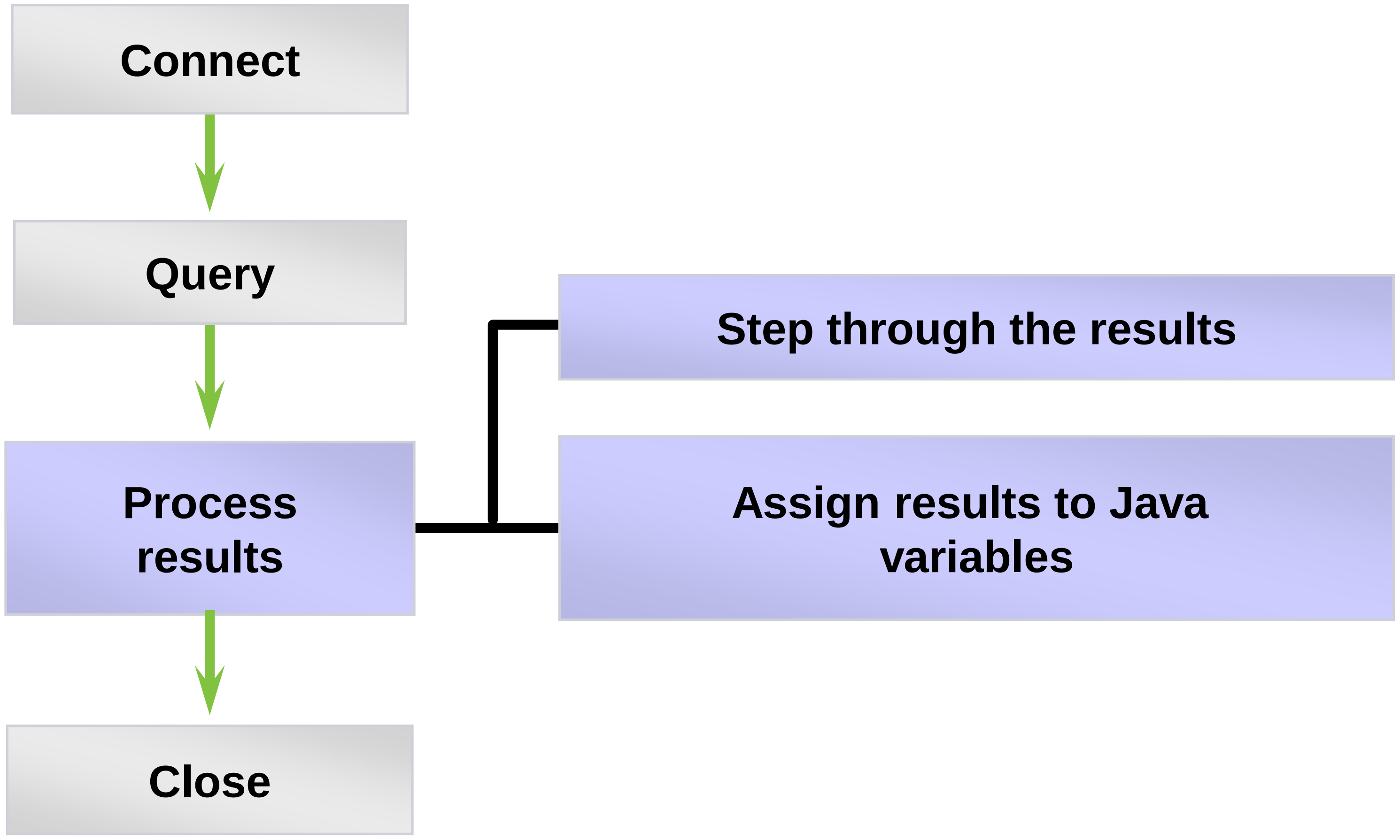
```
Statement stmt = conn.createStatement();
ResultSet rset = stmt.executeQuery
  ("select NAME, VERTICAL from STUDENT");
```

- Following Statements are used to execute Select statement:

```
Statement stmt = conn.createStatement();
int rowcount = stmt.executeUpdate
  ("delete from STUDENT  where ID = 1000");
```

Process Result

Process the Results



Process the Results: The ResultSet Object

- ResultSet is an object that contains the results of executing a SQL statement.
- A ResultSet maintains a cursor pointing to its current row of data
- Use next() to step through the result set row by row
- To retrieve the data from the columns, we can use getXXX() method.

How to Process the Result?

1. Step through the result set

```
while (rset.next()) { … }
```

2. Use get*XXX*() to get each column value

```
String val =
rset.getString(colname);
```

```
String val =
rset.getString(colIndex);
```

```
while (rset.next()) {
  String name = rset.getString(“NAME");
  String supervisor = rset.getString(“SUPERVISOR");
  … // Process or display the data
}
```

Example (Contd.).

```
class TestConnection{
         public  static void   main(String args[] ) {
                      new MakeDatabaseConnection();
             }
 }
```

Quiz

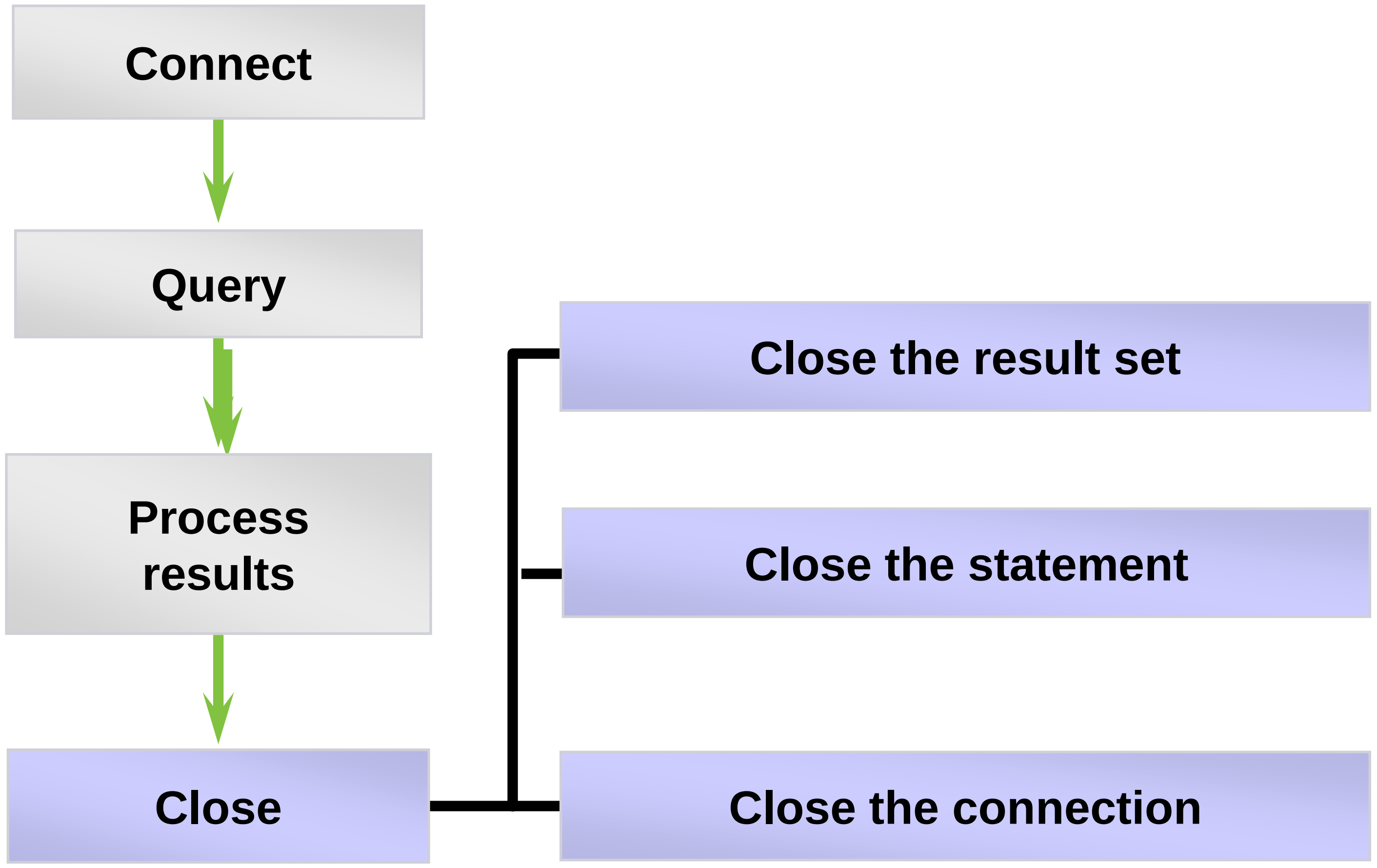
1. To load a driver into the memory _____________ method is used.
2. To make a connection ___________ method is used.
3. ________ method is used to create a Statement Object.
4. _____________ method is used to retrieve a String from ResultSet Object.

How to handle SQL Null values?

- Java primitive types cannot have null values
- Do not use a primitive type when your query might return a SQL null
- Use ResultSet.wasNull() to determine whether a column has a null value

```
while (rset.next()) {
   String year = rset.getString("YEAR");
   if (rset.wasNull() {
      … // Handle null value}
…}
```

Close Connection



How to Close the Connection?

1. Close the ResultSet object

```
rset.close();
```

2. Close the `Statement` object

```
stmt.close();
```

3. Close the connection (not necessary for server-side driver)

```
conn.close();
```

The DatabaseMetaData Object

- DatabaseMetaData is an interface to get comprehensive information about the database as a whole.

- The `Connection` object can be used to get a `DatabaseMetaData` object

- This object provides more than 100 methods to obtain information about the database

How to obtain Database Metadata?

1. To get the DatabaseMetaData Object

```
DatabaseMetaData dbmd = conn.getMetaData();
```

2. Use the object’s methods to get the metadata

```
DatabaseMetaData dbmd = conn.getMetaData();
String  s1 = dbmd getURL();
String  s2 = dbmd.getSQLKeywords();
boolean b1 = dbmd.supportsTransactions();
boolean b2 = dbmd.supportsSelectForUpdate();
```

The ResultSetMetaData Object

- ResultSetMetaData is an interface which contains methods to get information about the types and properties of the columns in the ResultSet object.

- ResultSetMetaData object provides metadata, including:
  - Number of columns in the result set
  - Column type
  - Column name

How to obtain ResultSetMetadata?

1. To get the `ResultSetMetaData` object

```
ResultSetMetaData rsmd = rset.getMetaData();
```

2. Use the object's methods to get the metadata

```
ResultSetMetaData rsmd = rset.getMetaData();
for (int i = 1; i <= rsmd.getColumnCount(); i++) {
  String colname = rsmd.getColumnName(i);
  int coltype = rsmd.getColumnType(i);
  …
}
```

Example

```
import java.sql.*;
public class MetaDataEx
{
public static void main(String s[])
{    try{ Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connectioncon=DriverManager.getConnection("jdbc:odbc:mdsn"
  ,"scott","tiger");
DatabaseMetaData dbmd = con.getMetaData();
String  s1 = dbmd.getURL();
System.out.println(s1);
String  s2 = dbmd.getSQLKeywords();
System.out.println(s2);
boolean b1 = dbmd.supportsTransactions();
System.out.println(b1);
boolean b2 = dbmd.supportsSelectForUpdate();
System.out.println(b2);
Statement st=con.createStatement();
```

Example (Contd.).

```
ResultSet rset=st.executeQuery("select
  ename,empno,sal,comm from emp");
ResultSetMetaData rsmd = rset.getMetaData();
System.out.println(rsmd.getColumnCount());
for (int i = 1; i <= rsmd.getColumnCount(); i++) {
  String colname = rsmd.getColumnName(i);
System.out.println(colname);
String coltype = rsmd.getColumnTypeName(i);
System.out.println(coltype);
}
con.close();
}
catch(Exception e1){System.out.println(e1);
}
}
}
```

Mapping Database Types to Java Types

ResultSet maps database types to Java types.

```
ResultSet rset = stmt.executeQuery
  ("select ID, DATE_OF_JOIN, SUPERVISOR
  from STUDENT");

int id = rset.getInt(1);
Date rentaldate = rset.getDate(2);
String status = rset.getString(3);
```

Col Name	Type
ID	NUMBER
DATE_OF_JOIN	DATE
SUPERVISOR	VARCHAR2

The PreparedStatement Object

- Using PreparedStatement in place of Statement interface will improve the performance of a JDBC program.
- A PreparedStatement object holds precompiled SQL statements
- Use this object for statements you want to execute more than once
- A prepared statement can contain variables that you supply each time you execute the statement

How to Create a PreparedStatement?

1. Register the driver and create the database connection

2. Create the prepared statement, identifying variables with a question mark (?)

```
PreparedStatement pstmt =
  conn.prepareStatement("update STUDENT
  set SUPERVISOR = ? where ID = ?");
```

```
PreparedStatement pstmt =
  conn.prepareStatement("select SUPERVISOR from
  STUDENT where ID = ?");
```

How to execute PreparedStatement?

1. Supply values for the variables

```
pstmt.setXXX(index, value);
```

2. Execute the statement

```
pstmt.executeQuery();
pstmt.executeUpdate();
```

```
PreparedStatement pstmt =
  conn.prepareStatement("update STUDENT
  set SUPERVISOR = ? Where ID = ?");
pstmt.setString(1, "OUT");
pstmt.setInt(2, id);
pstmt.executeUpdate();
```

Example

```
import java.sql.*;
public class PreparedStEx
{
private Connection con;
private PreparedStatement pstmt;
public PreparedStEx()
{    try{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    con=DriverManager.getConnection("jdbc:odbc:krishna");
     st=con.createStatement();
     st.executeUpdate("create table test (name char(25), id
  int)");
String
  data[][]={{"Ford","100"},{"Arthur","110"},{"Trillian","120"},
  {"Zaphod","130"}};
pstmt=con.prepareStatement("insert into test(name,id)
  values(?,?)");
      for(int i=0;i<data.length;i++){
```

Example (Contd.).

```
pstmt.setString(1,data[i][0]);
pstmt.setInt(2,Integer.parseInt(data[i][1]));
pstmt.executeUpdate();
}
pstmt.close();
con.close();
}catch(Exception e)
{
e.printStackTrace();
}
}
public static void main(String[]a )
{
PreparedStEx  t=new PreparedStEx();
}
}
```

Callable Statement

The CallableStatement Object

- A CallableStatement object is used for calling the stored procedure from JDBC program.

- A callable statement can contain variables that you supply each time you execute the call

- When the stored procedure returns, computed values (if any) are retrieved through the CallableStatement object

How to Create a CallableStatement?

- Register the driver and create the database connection
- On connection object prepareCall() method is used to call the stored procedure.
- Create the callable statement, identifying variables with a question mark (`?`)

```
CallableStatement cstmt =
    conn.prepareCall("{call " +
    ADDITEM +
    "(?,?,?)}");
  cstmt.registerOutParameter(2,Types.INTEGER);
  cStmt.registerOutParameter(3,Types.DOUBLE);
```

How to execute a CallableStatement?

1. To pass the input parameters

```
cstmt.setXXX(index, value);
```

2. CallableStatement should be executed, as:

```
cstmt.execute();
```

3. To get the output parameters

```
var = cstmt.getXXX(index);
```

Example

```
import java.sql.*;
public class ProcedureCall
{
public static void main(String args[])
{    try{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection
  con=DriverManager.getConnection(“jdbc:odbc:mdsn","s
  cott","tiger");
  CallableStatement cstmt = con.prepareCall("{call "
  +"addnumbers" + "(?,?,?)}");
    cstmt.registerOutParameter(3,Types.INTEGER);
  cstmt.setInt(1,Integer.parseInt(args[0]));
  cstmt.setInt(2,Integer.parseInt(args[1]));
  cstmt.execute();
```

Example (Contd.).

```
System.out.println(cstmt.getInt(3));
con.close();
  }catch(Exception e)
  {
    System.out.println(e);
  }
}
}
```

Using Transactions

- With JDBC drivers:
  - New connections are in autocommit mode
  - Use conn.setAutoCommit(false) to turn autocommit off

- To control transactions when you are not in autocommit mode:
  - conn.commit(): Commit a transaction
  - conn.rollback(): Roll back a transaction

Example for creating a table

```
import java.sql.*;
class MakeConnection {
  Connection con;
   Statement stmt;
   ResultSet  rs;
   MakeConnection() {
     try{
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  con=DriverManager.getConnection("Jdbc:Odbc:emp","","");
          stmt = con.createStatement();
          int i=stmt.executeUpdate("create table pradeep(empno
  integer,ename varchar(20),deptno integer)");
     }
     catch(Exception e) {
          System.out.println(e);
     }
  }
}
```

Example for Creating a table (Contd.).

```
class TestConnection1{
            public  static void main(String args[] ) {
                           new MakeConnection();
             }
  }
```

Example for inserting values into table

```
import java.sql.*;
class MakeConnection {
  Connection con;
  Statement stmt;
  ResultSet  rs;
  int i1, i2, i3;
  MakeConnection() {
    try{
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      con=DriverManager.getConnection("Jdbc:Odbc:emp","","");
        stmt = con.createStatement();
         i1=stmt.executeUpdate("insert into pradeep
  values(1,'sakre',23)");
         i2=stmt.executeUpdate(“insert into pradeep
  values(1,'pradeep',223)");
         i3=stmt.executeUpdate(" insert into pradeep values
  (001,'vivek',243)");
```

Example for inserting values into table(Contd.).

```
          }
          catch(Exception e) {
             System.out.println(e);
          }
     }
}
class TestConnection2{
           public  static void   main(String args[] ) {
                      new MakeConnection();
           }
}
```

Quiz

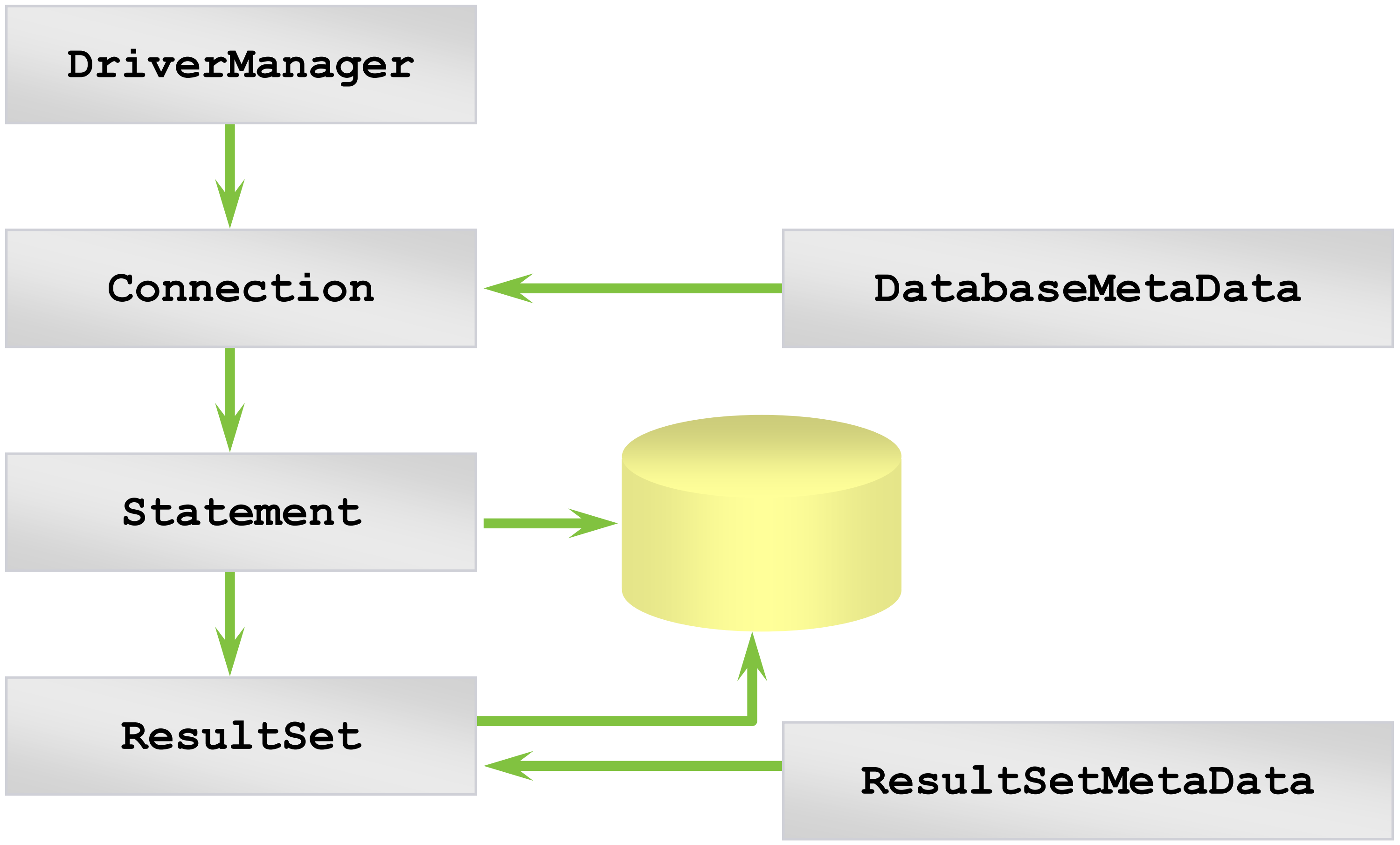
1. **_____ method is used for PreparedStatement Object.**

2. **_________ method is used changed for auto commit mode.**

3. **_______ method is used for call a stored procedure from JDBC.**

4.

Summary of JDBC Classes



Summary

- In this module, you were able to:
  - Explain how to connect to a database using Java Database Connectivity (JDBC).
  - Create and execute a query using JDBC API.
  - Analyze how to use the Metadata objects to retrieve more information about the database or the result set.
  - Know the function of commit and roll back in transactions.

References

1. Armstrong, E., Ball and others (2005). *The J2EE^{TM}1.4 tutorial.* Retrieved March 9, 2012, from, http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

2. Oracle (2012). *JDBC Basics.* Retrieved March 9, 2012, from http://java.sun.com/docs/books/tutorial/jdbc/basics/index.html

Thank You