edureka!

MODULE-7
JDBC

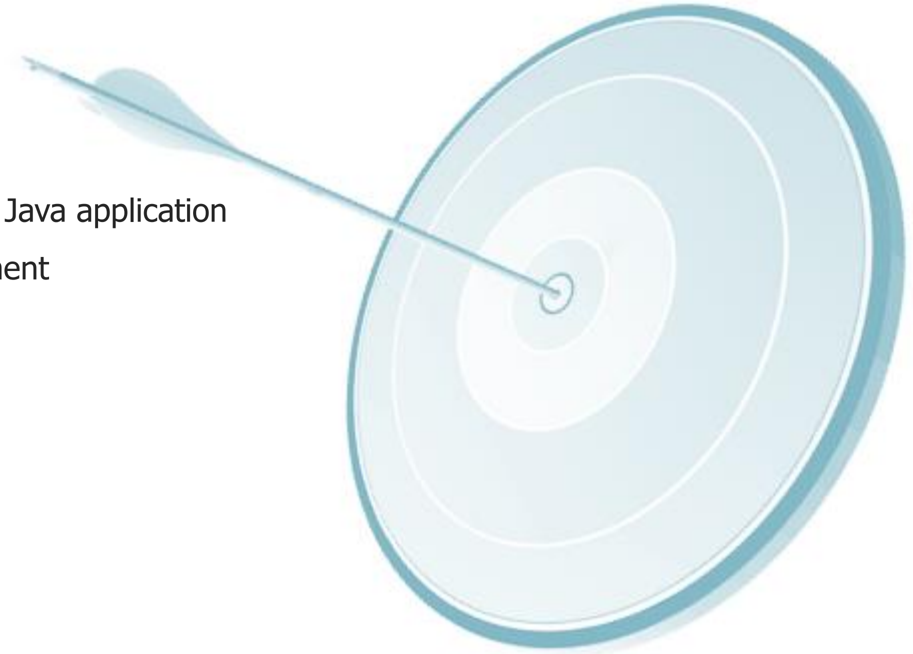
Course Topics

Objectives

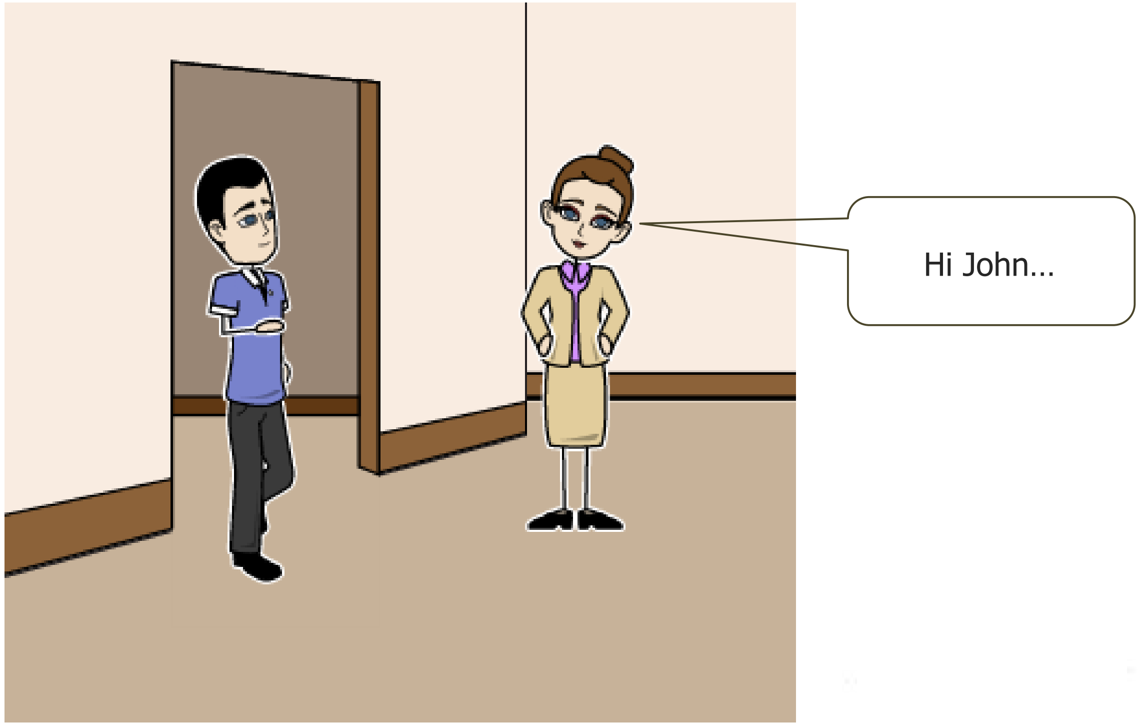
edureka!

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

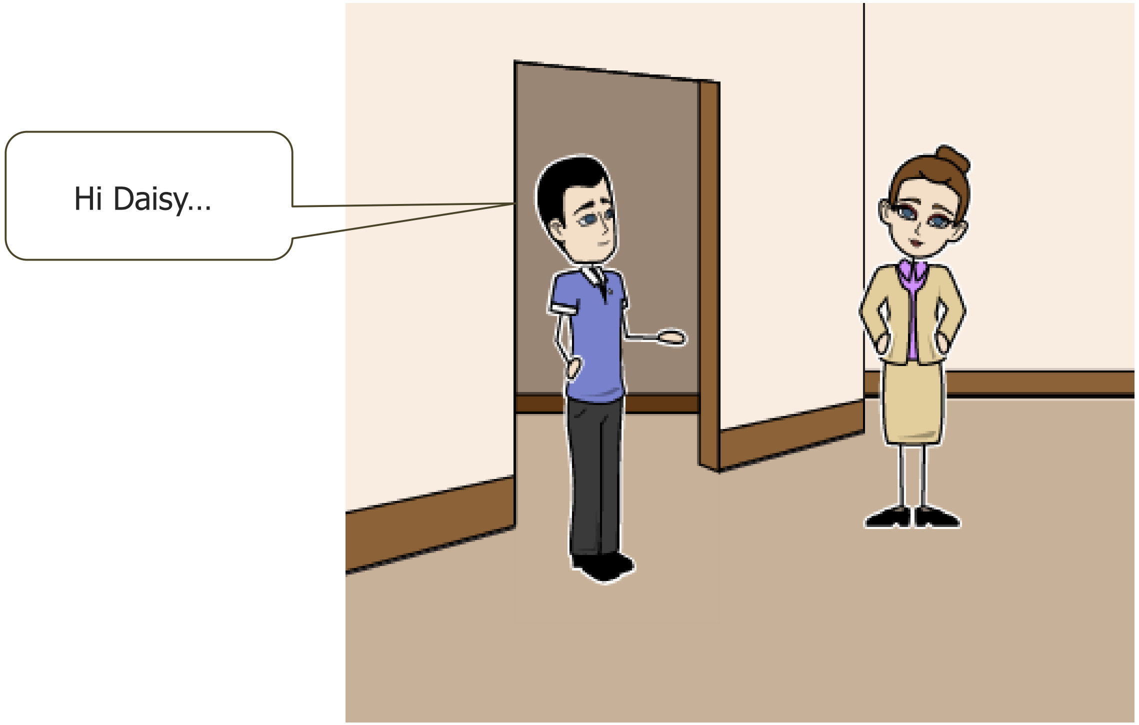
- → Understand database
- → Understand RDBMS and its advantages
- → Execute SQL Queries
- → Create JDBC connection to the database
- → Perform CRUD operation on the database from your Java application
- → Perform Batch processing and Transaction management



John Meets Daisy..



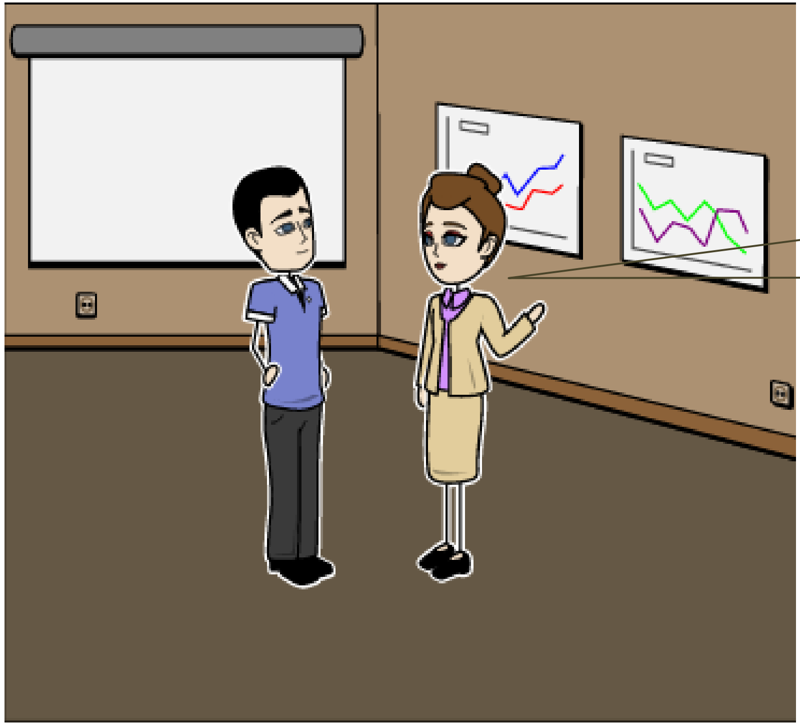
John Meets Daisy..



Daisy has a Doubt

I need a little help from you. I need to clarify some doubts..

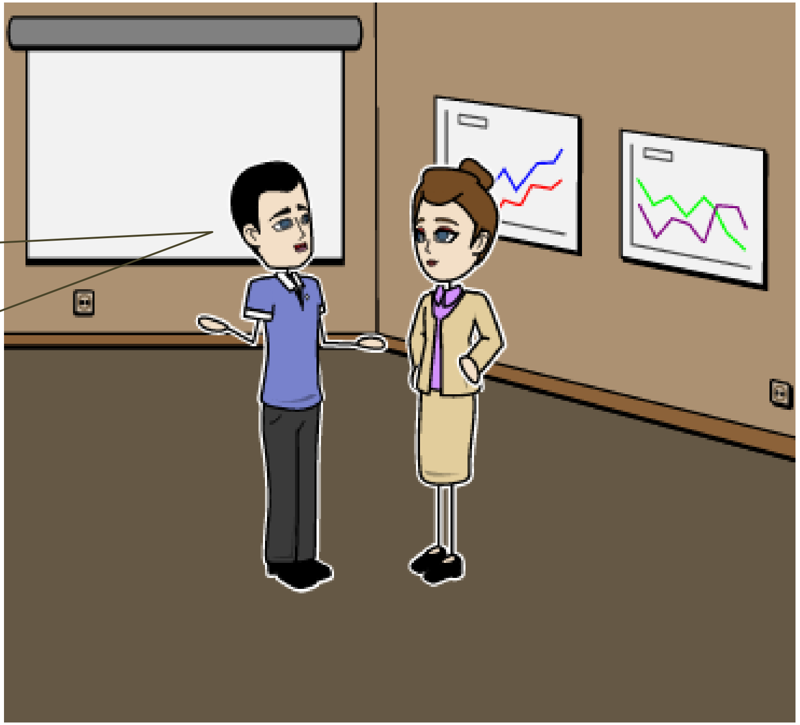
Daisy has a Doubt



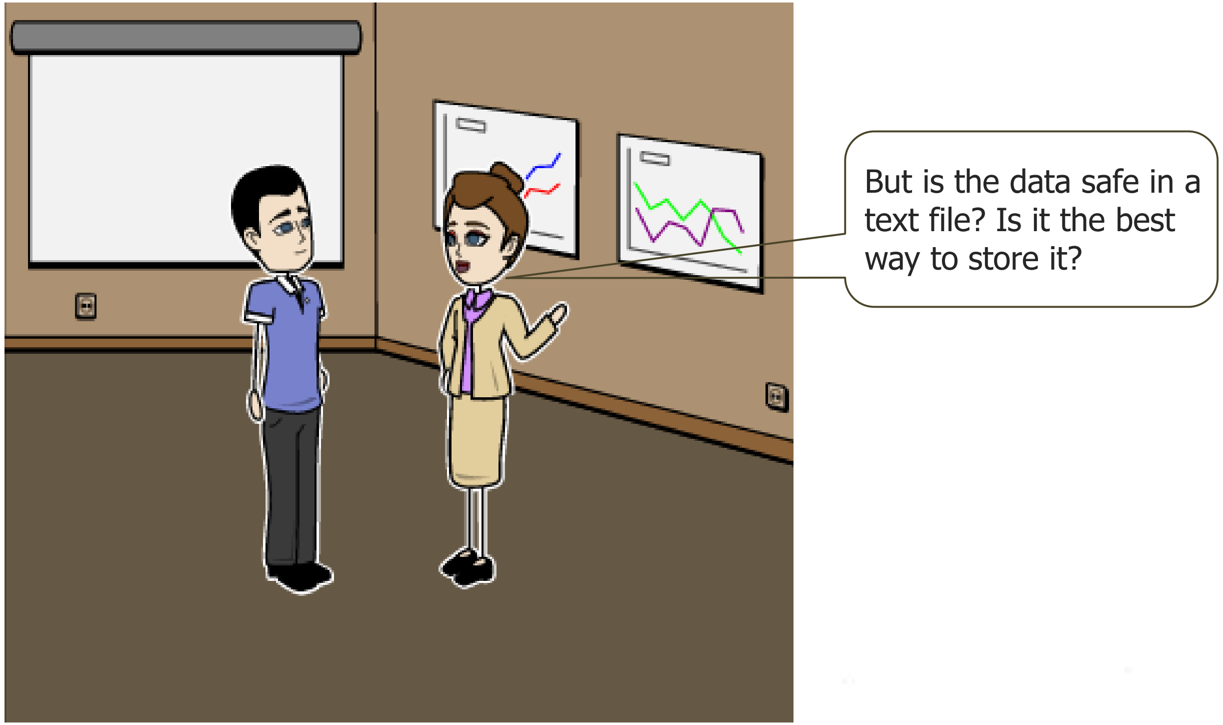
Why do we need to store data in the text file when I can have it in the program?

Daisy explains the Concept..

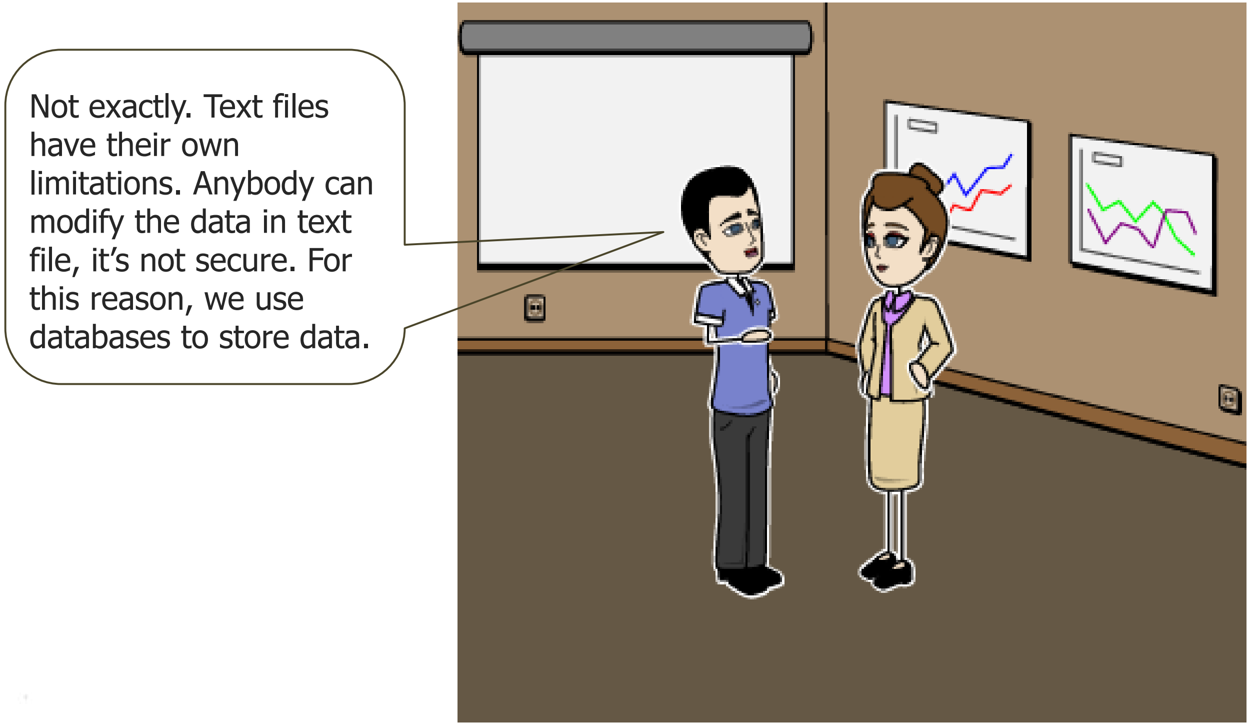
We can store the data in text file permanently. If it is in a program then once the system is off, all the data is lost. The data in a program is volatile.



Daisy explains the Concept..


But is the data safe in a text file? Is it the best way to store it?

Daisy explains the Concept..


Not exactly. Text files have their own limitations. Anybody can modify the data in text file, it's not secure. For this reason, we use databases to store data.

Limitations of a text File

→ A regular text file is not secured.

→ Concurrency is not provided.

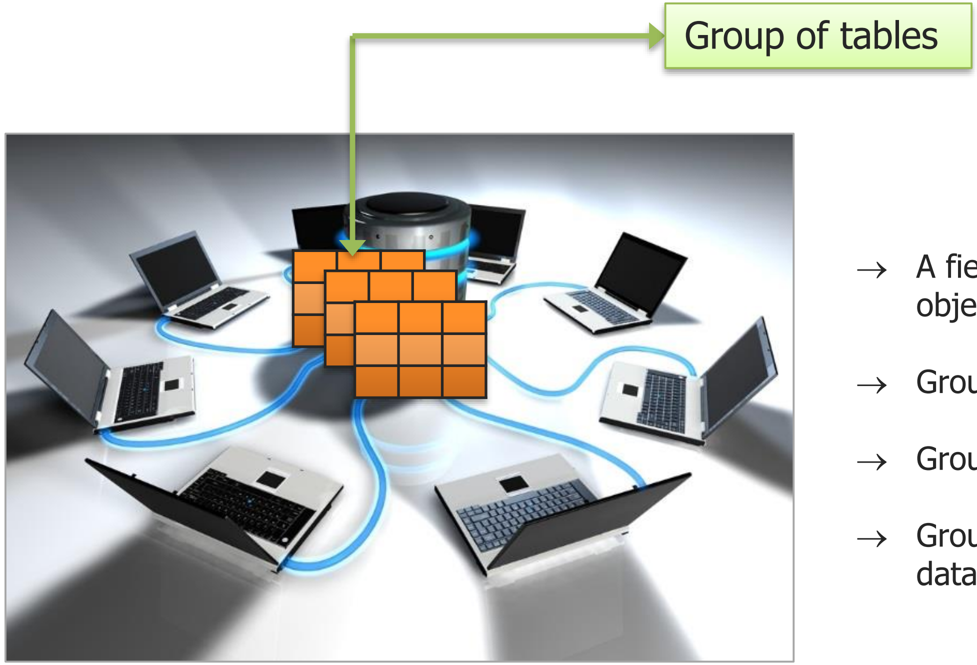
→ There is no proper structure or guidelines when to create a new file or to add a new field in the existing data file.

→ Anybody can modify/delete the data by just opening the file.

→ Everybody can maintain their own data file and the user may not update their copied data file when the data is updated.

→ Multiple copies of the text file is maintained and the user may not be able to distinguish the file with relevant and complete data.

What is a Database?



→ A field is an information or attribute of an object.

→ Group of fields will form a record.

→ Group of records are in a table.

→ Group of tables are in a database or relational database or RDBMS.

Database - Example

edureka!

Example: fields to students on fields, records, table and db.

Fields

ID	NAME	AGE	ADDRESS	SALARY

Record

1	John	32	California	2000.00

Column

ADDRESS
California
Arizona
Texas
Washington
New Mexico
Georgia
Florida

Table

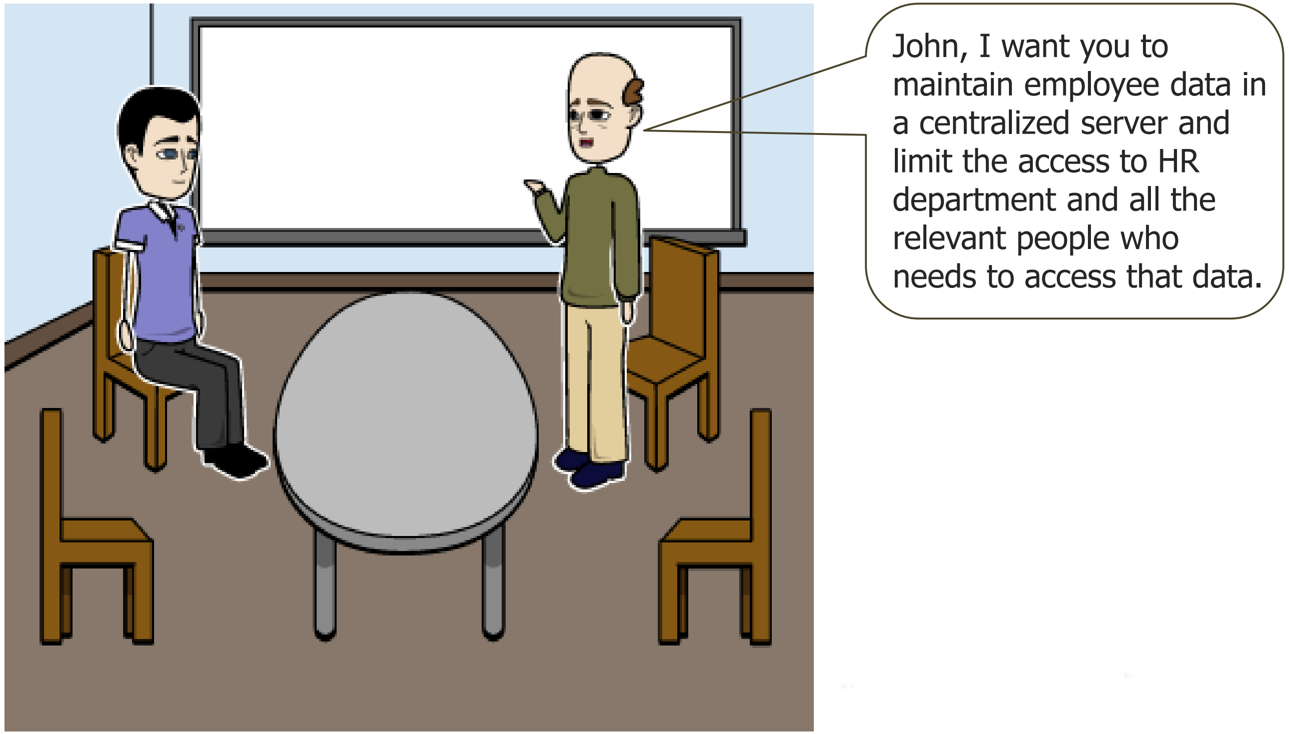
ID	NAME	AGE	ADDRESS	SALARY
1	John	32	California	2000.00
2	Jack	25	Arizona	1500.00
3	Nancy	23	Texas	2000.00
4	Linda	25	Washington	6500.00
5	Shane	27	New Mexico	8500.00
6	David	22	Georgia	4500.00
7	Mary	24	Florida	10000.00

Database

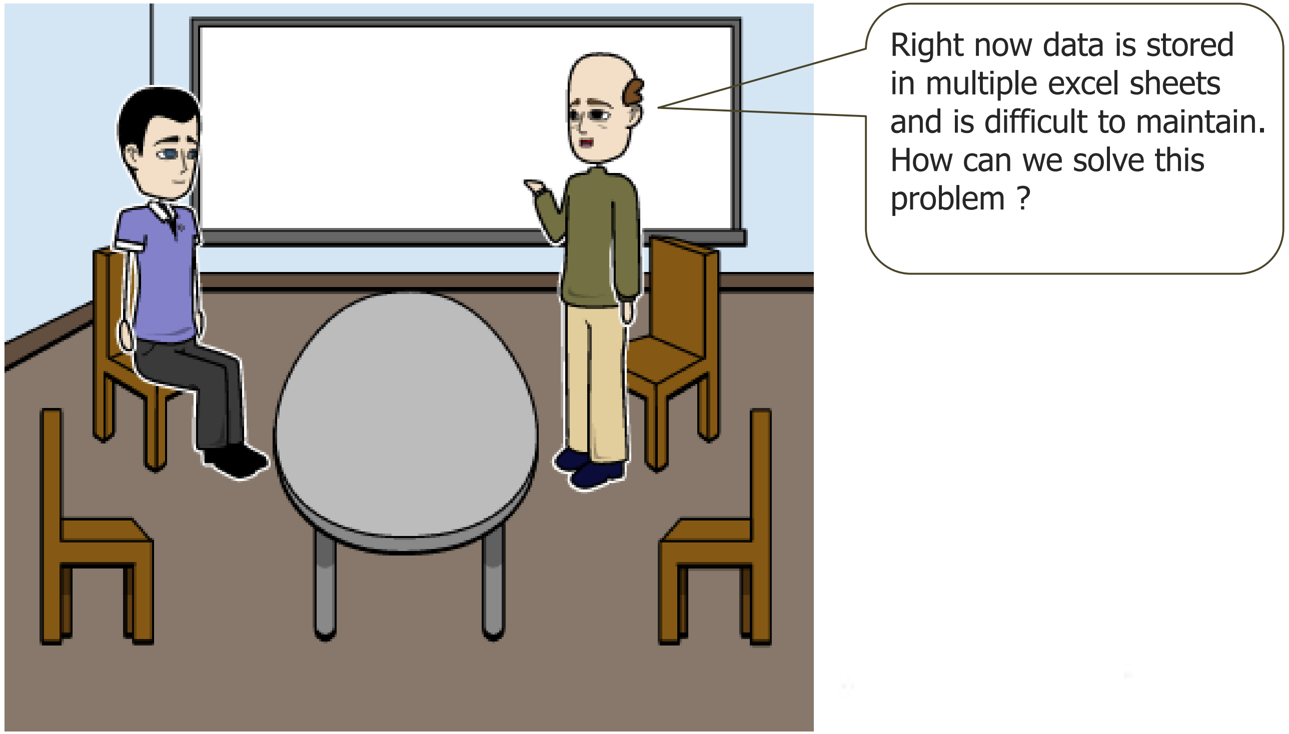
A Database is a collection of tables

RDBMS

A Project for John..

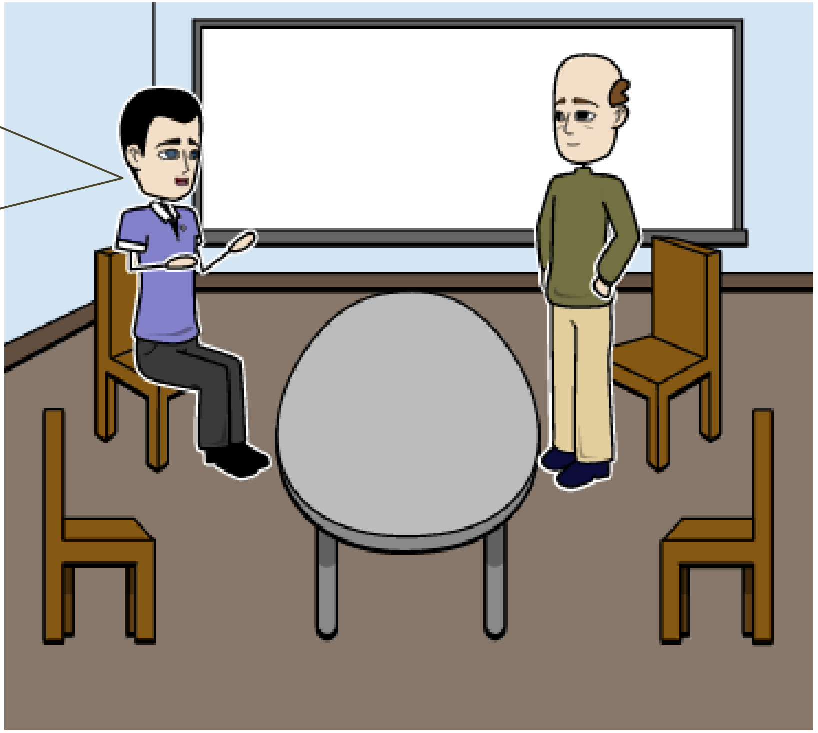

John, I want you to maintain employee data in a centralized server and limit the access to HR department and all the relevant people who needs to access that data.

A Project for John..

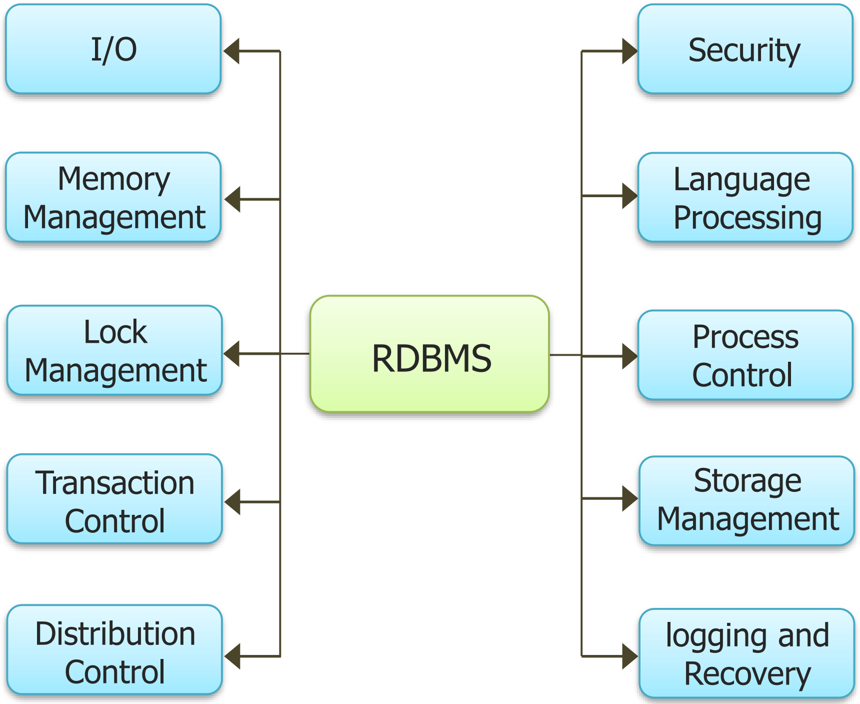

Right now data is stored in multiple excel sheets and is difficult to maintain. How can we solve this problem ?

John has the Solution!

Sir, we can use MySQL RDBMS to resolve this issue. We can store all the employee data in this database. I will create the users and give the required permissions that allows concurrent access.



What is RDBMS?



RDBMS stands for

Relational **D**atabase **M**anagement **S**ystem.

A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd.

Advantages of RDBMS

→ Related group of fields are formed as table and is easy to understand.

→ Many users can access the tables simultaneously.

→ Many users can be created for the database with access privileges. Few users are administrator. They can create/add/delete data and tables. Few users are given privileges just to access the data and few users are given access to add and modify with no delete option. This way data security can be provided.

→ The database can be placed on a centralized server and can be accessed by everyone who is on the network.

→ RDBMS gives the best performance in terms of speed.

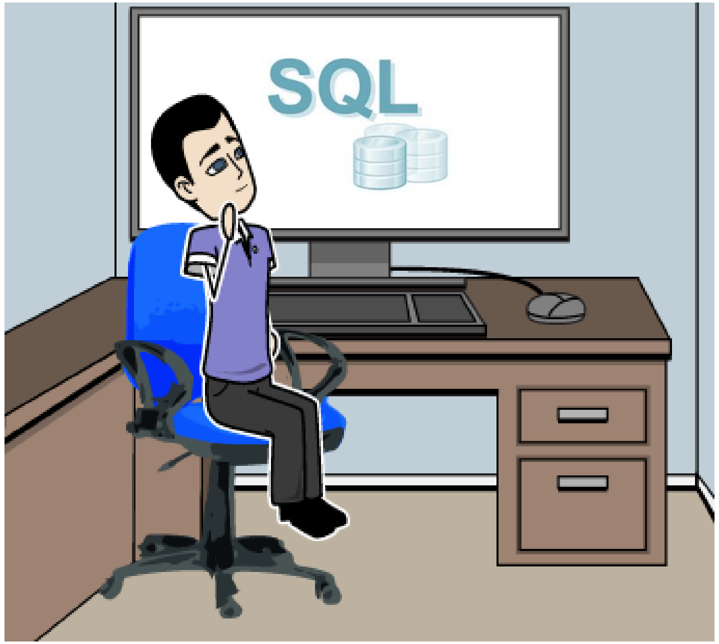
→ Oracle, MySQL are the famous RDBMS. Some of the versions of these software are free and available on net.

→ SQL (Structured Query Language) is used to perform operations in RDBMS.

SQL

Introduction

In continuation of the above case, John wants to create tables, insert data and wants to fetch the records based on the requirements in his program. For this John finds SQL easy to use. He uses SQL to create the tables manually on the database client. He uses SQL Insert, update and delete to perform the operations on database data and SQL select statements to fetch the data from the database.



SQL

edureka!

→ SQL Stands for Structured Query Language.

→ It is the language used in RDBMS to perform operations on DB.

→ It is easy to learn and understand.

→ User can perform create tables, insert, update, delete and select operations using SQL. When SQL command is fired on the database, database parser will parse the SQL command, understands it and performs the required operations.

SQL Commands

create table → To create a table.

insert into → To add the data in the table.

update → Update a record in the table.

delete from → To delete records in the table.

drop table → To delete a table.

alter table → To modify the table structure. A field can be added or deleted in the table.

select → Used to display records from the table.

views → View of the db/table can be created.

avg(), count(), min(), max() are some of the functions of SQL.

SQL Datatypes

Data type	SQLServer	Oracle	MySQL
boolean	Bit	Byte	N/A
integer	Int	Number	Int Integer
float	Float Real	Number	Float
currency	Money	N/A	N/A
string (fixed)	Char	Char	Char
string (variable)	Varchar	Varchar Varchar2	Varchar
binary object	Binary (fixed up to 8K) Varbinary (<8K) Image (<2GB)	Long Raw	Blob Text

SQL Commands

→ This command creates a table by name emp with 3 fields: id, name and dept

```
SQL> create table emp (id number, name varchar(20), dept varchar(20), address varchar(20), salary number);
```

→ This command will insert a record into emp table with values for the fields.

```
SQL> insert into emp values (120, ‘john','technology training', 'baltimore', 60000);
```

→ This command updates a record ' john' to ‘John snow' in emp table.

```
SQL> update emp set name=‘john snow' where name=‘john';
```

→ The following command removes a record where name is joe.

```
SQL> delete from emp where name=‘joe';
```

SQL Commands (contd.)

→ Drop TABLE removes the table from the database:

```
SQL > drop table emp;
```

→ To remove all the records of the table:

```
SQL> TRUNCATE TABLE emp;
```

→ Following command will adda field address to the emp table:

```
SQL> alter table emp add address varchar(20);
```

→ Following command displays all the records from emp table:

```
SQL>Select * from emp;
```

SQL SELECT Command

```
Select * from emp where name=‘John';
Select * from emp where salary > 50000;
Select * from emp order by name;
Select * from emp order by name desc;
Select * from emp where salary between 10000 to 20000;
Select * from emp where Address IN (‘cal', ‘Arizona');
```

SQL Functions

→ Following SQL will display average, minimum and maximum salary.

```
SQL> select avg(salary), min(salary), max(salary) from emp;
```

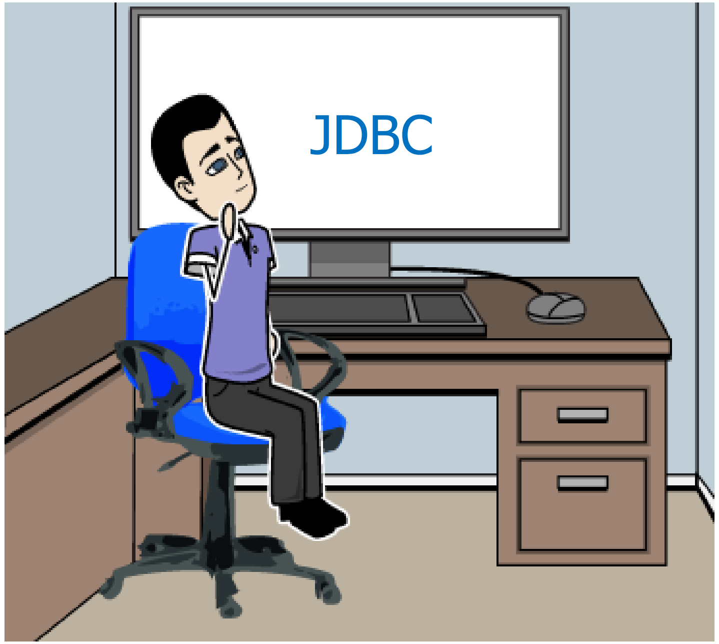
→ Following command display number of records in the table emp using count(*)

```
SQL> select count(*) from emp;
```

JDBC

Introduction

In continuation of the above case, John wants to develop/generate reports from Java program. Instead of writing his own program for database operations, he uses JDBC and fetches the records and displays the data in the required format.



JDBC

→ JDBC Stands for Java database connectivity.

→ JDBC API is the industry standard for database-independent connectivity.

→ JDBC interface is not specific to a database.

→ JDBC is a Java Library which is used to connect a database.

Features of JDBC

→ Obtains the connection with the database.

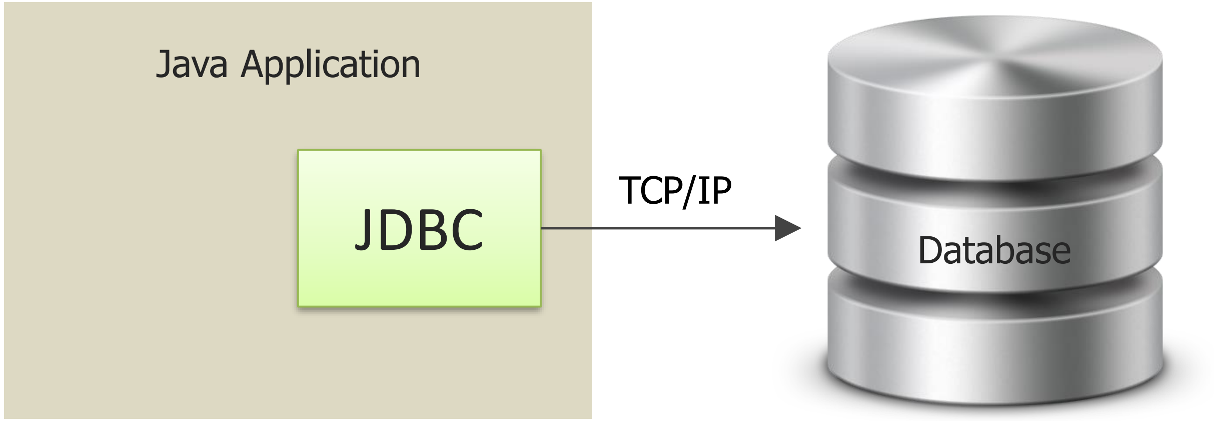
→ Connection usage: Every time for every database operation, a new connection need not be obtained. Once a connection is obtained with the database, this connection can be used for many database operations and towards the database connection can be closed.

→ Supports BLOB (Binary Large Object) and CLOB (Character Large Object.): BLOB is used to store binary data like voice, image etc., CLOB is used for storing large character data. It can also store single byte character or multi byte character data which support different languages like French, German etc.,

→ Batch updates.

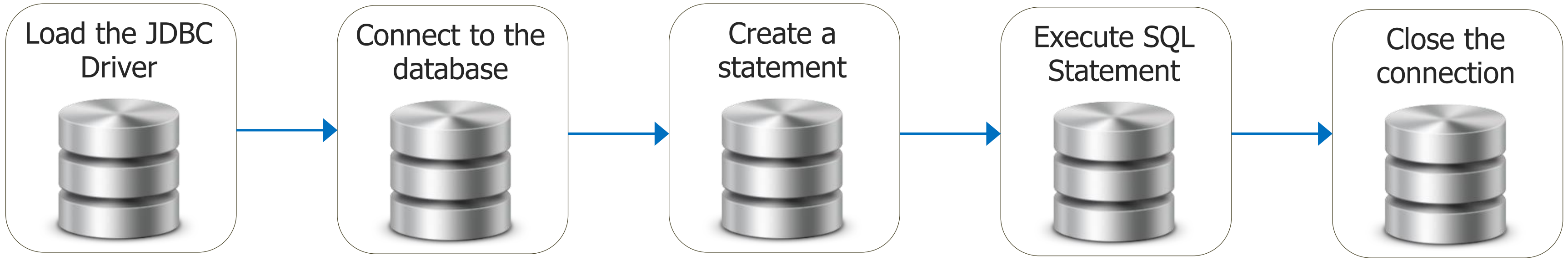
Connection using JDBC

edureka!



JDBC Procedure

edureka!



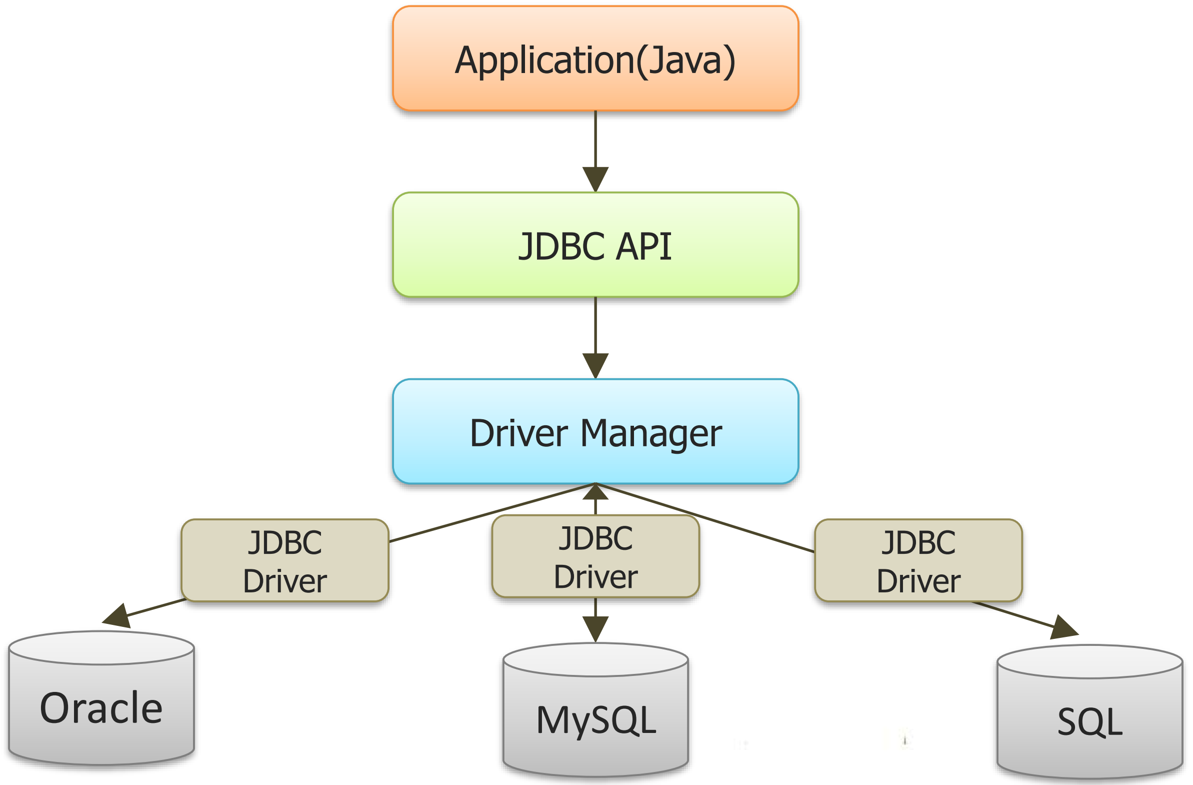
Software Required for JDBC

→ Oracle/MySQL installation.

→ JDBC driver provided by the database vendor (jar file).

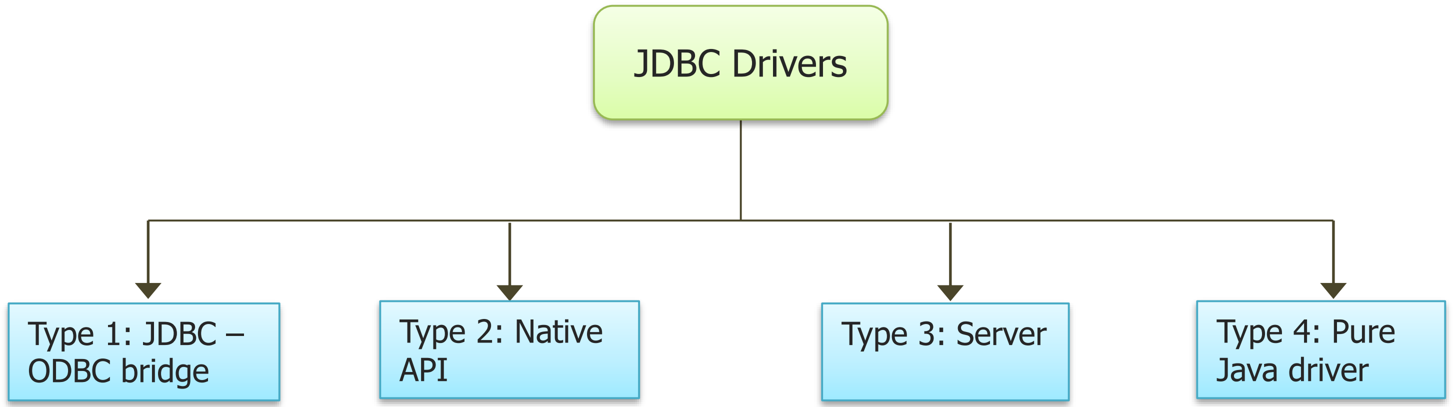
→ Remember the User ID and password of the database while installing. We need it while connecting database from JDBC.

Java – DB Connectivity



Driver Types

There are 4 types of JDBC drivers.



Annie's Question

What does ODBC stand for?

Annie's Answer

Open database connectivity.

What is ODBC?

- → ODBC (Open Database Connectivity) is a standard programming language middleware API for accessing database management systems (DBMS).
- → The designers of ODBC aimed to make it independent of database and operating systems.
- → An application written using ODBC can be ported to other platforms, both on the client and server side, with few changes to the data access code.

Why use ODBC at all?

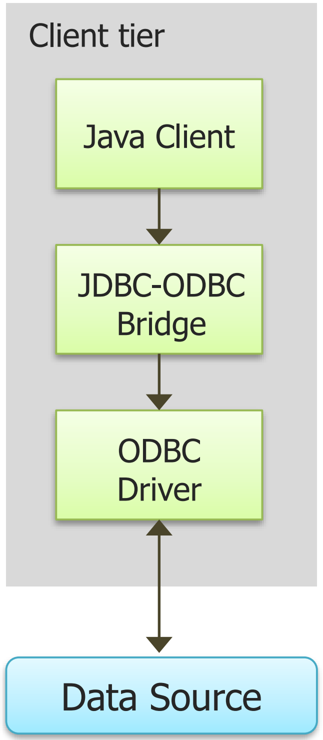
The ideal is "Pure Java": no native code, no platform dependent features. But you may need to begin your development effort right away, without waiting for your DBMS to offer a Java-only JDBC driver.

Partly native drivers, such as the JDBC - ODBC Bridge, lets you create programs that easily adapt to pure Java drivers as they become available.

Ref - http://docs.oracle.com/javase/7/docs/technotes/guides/jdbc/bridge.html

Type 1 - Driver

edureka!



→ Type 1 driver is JDBC-ODBC bridge.

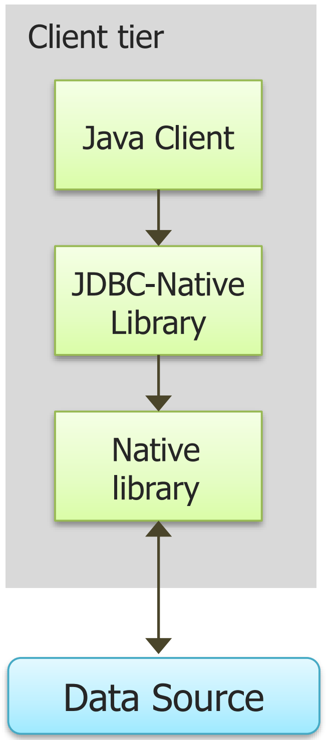
→ ODBC is a standard API which is used to connect to the database.

→ For Type-1 driver, Java communicates with JDBC.

→ JDBC communicates with ODBC and ODBC communicates with DB and vice-versa.

→ In the initial stages ODBC was only the option for JDBC, hence JDBC is used to communicate database through ODBC.

Type 2 - Driver



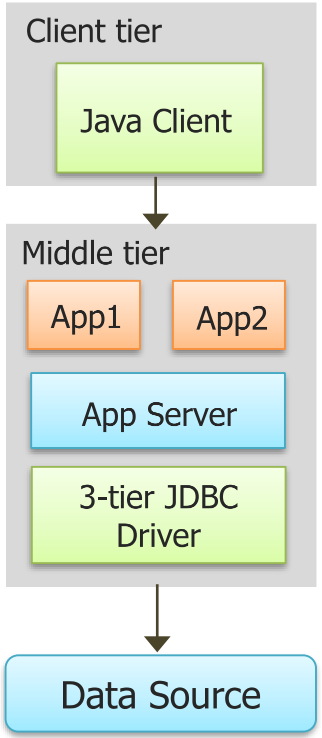
→ In Type-2 driver, JDBC calls are converted to native API code which is given by the database vendor itself.

→ Native API calls could be written either in C or in C++.

→ JDBC is depending on Native API to communicate with Database.

→ Since Native API is given by the database vendor, if the database is changed then its corresponding Native API should be used.

Type 3 - Driver

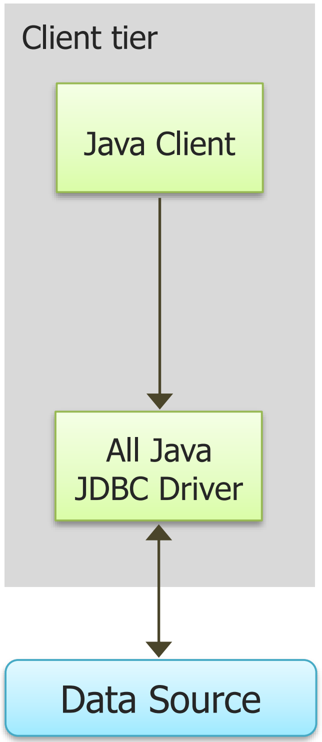


→ JDBC will communicate with a server using network sockets.

→ This server will convert the request given by JDBC into the commands which database understands and communicates with the db. Db will perform the required operation.

→ This is called as 3-tier approach as JDBC, Server and DB are involved in it for DB transaction.

Type 4 - Driver



→ Type 4 is pure Java driver.

→ JDBC does not depend on any other software/hardware to communicate with database.

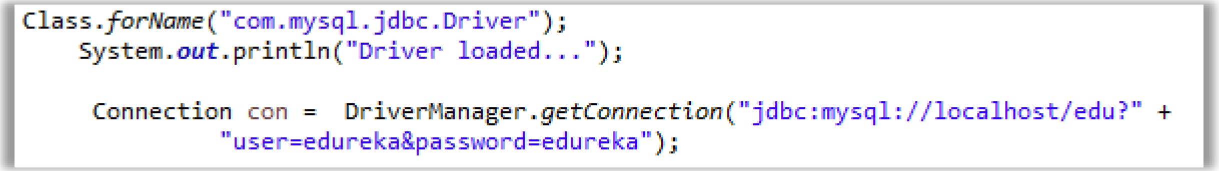
→ JDBC driver has to be provided by the database vendor.

JDBC Connection String

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

Creating Connection for Mysql Database:

Host Name: localhost
Database Name: edu
Username: edureka
Password: edureka

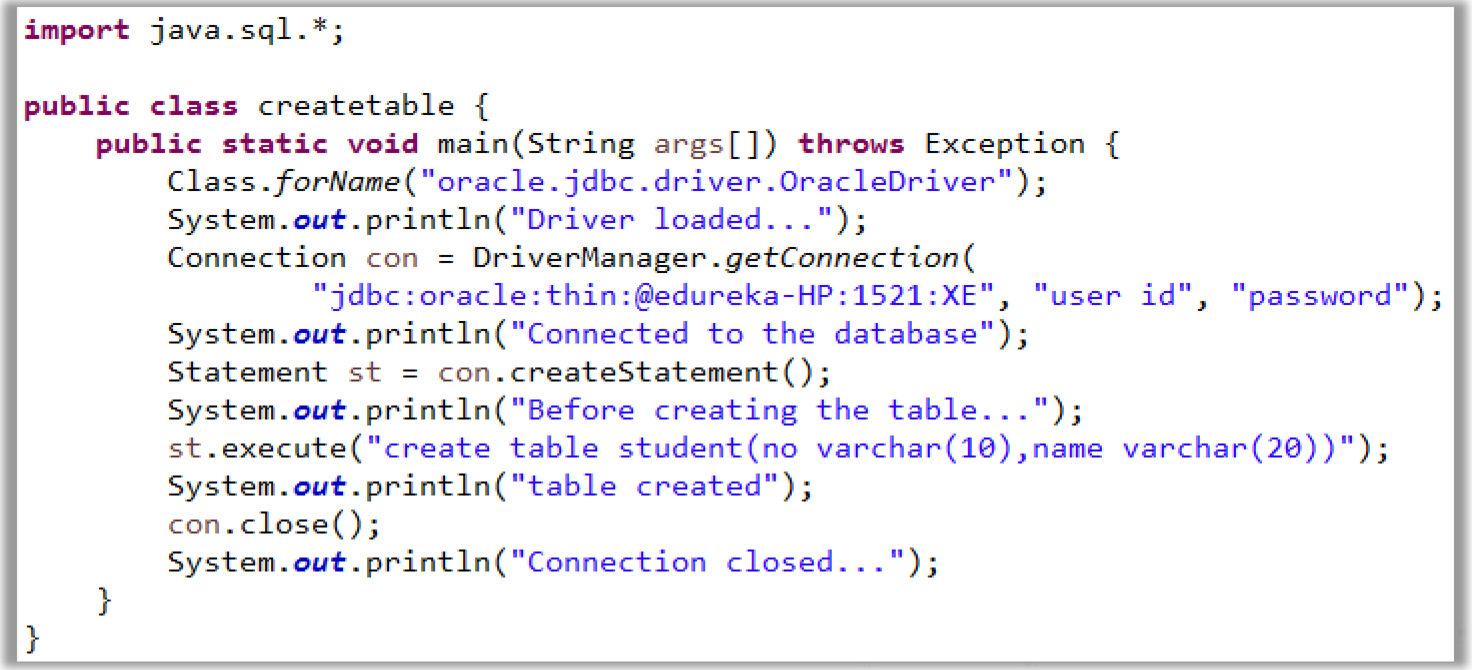


```
Class.forName("com.mysql.jdbc.Driver");
    System.out.println("Driver loaded...");

     Connection con =  DriverManager.getConnection("jdbc:mysql://localhost/edu?" +
              "user=edureka&password=edureka");
```

Program to Create Table in the db through JDBC

edureka!



```java
import java.sql.*;

public class createtable {
    public static void main(String args[]) throws Exception {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        System.out.println("Driver loaded...");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@edureka-HP:1521:XE", "user id", "password");
        System.out.println("Connected to the database");
        Statement st = con.createStatement();
        System.out.println("Before creating the table...");
        st.execute("create table student(no varchar(10),name varchar(20))");
        System.out.println("table created");
        con.close();
        System.out.println("Connection closed...");
    }
}
```

Program to Create Table in db through JDBC (contd.)

Class.forName loads the JDBC driver into memory.

Class Connection → The Connection class represents a connection with a specific database
Interface Statement → The object used for executing a static SQL statement and returning the results it produces.

DriverManager.getConnection() → Connects with the database.

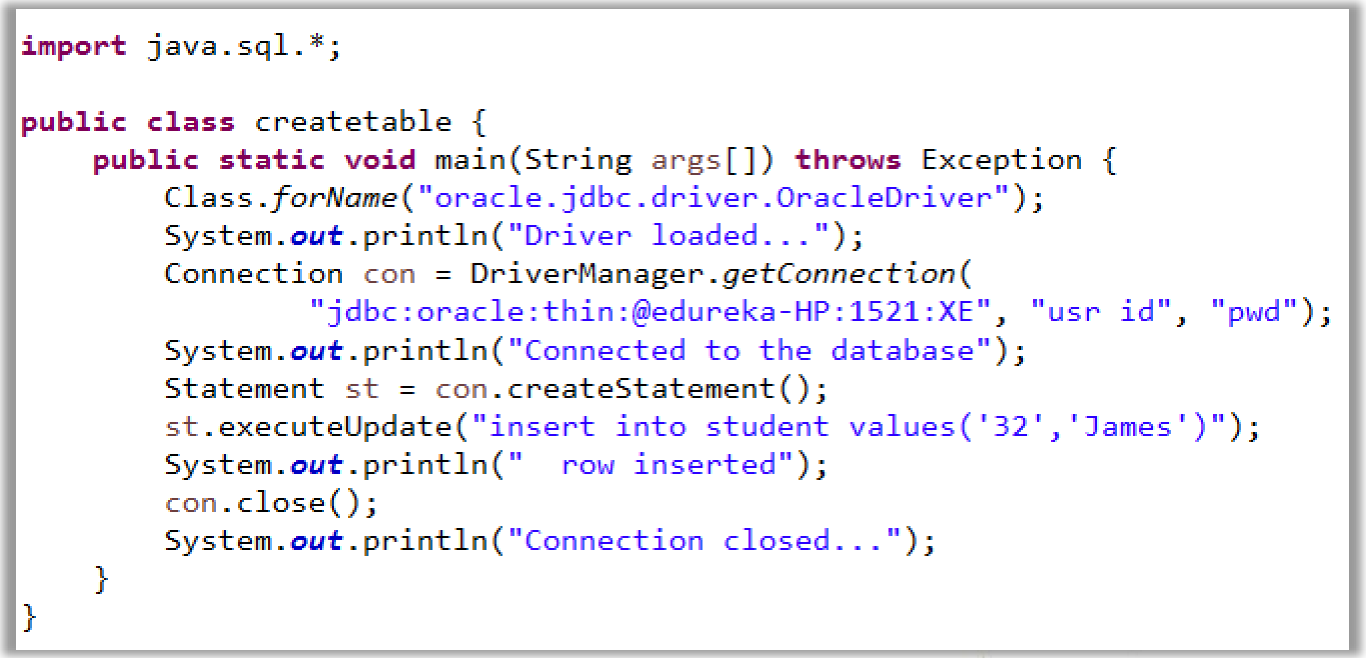
createStatement() → Creates a statement to store the SQL Query.

execute() method executes the SQL in the statement in database.

close() method is used to close the connection.

This program will create student table in the database with 2 fields no and name as characters.

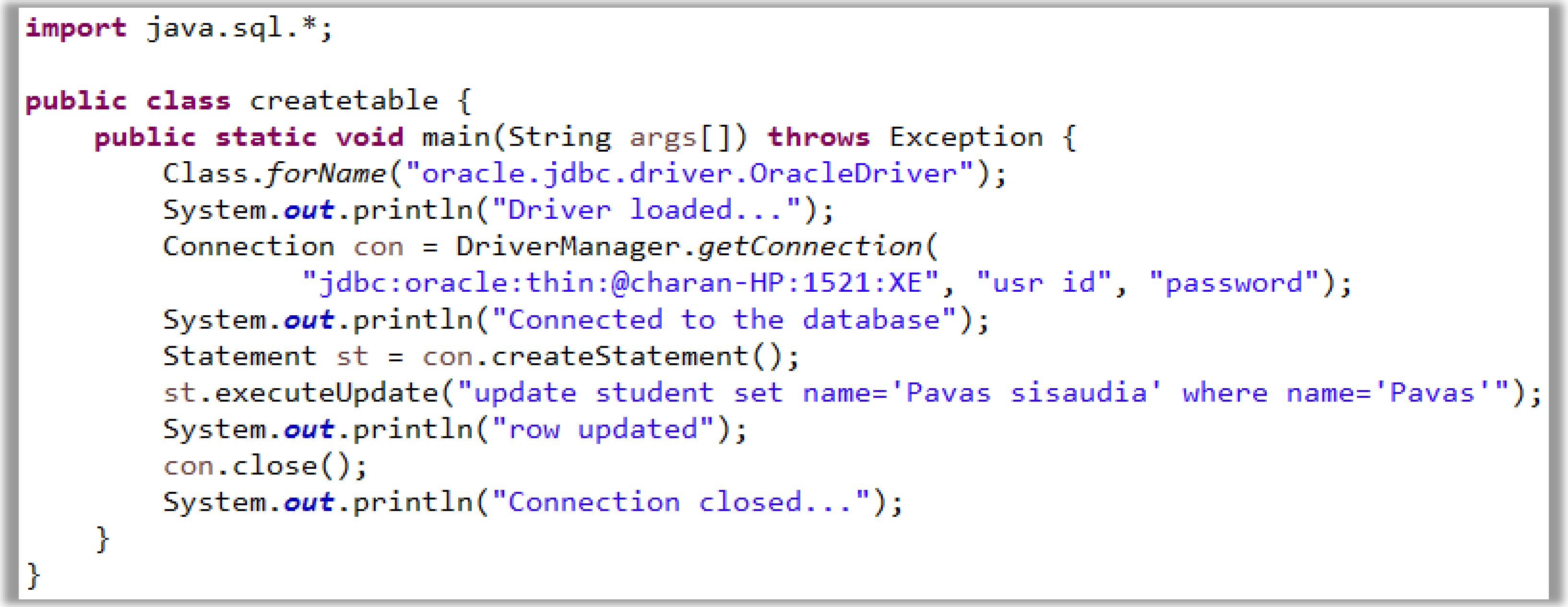
Program to Insert Data in db



```
import java.sql.*;

public class createtable {
    public static void main(String args[]) throws Exception {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        System.out.println("Driver loaded...");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@edureka-HP:1521:XE", "usr id", "pwd");
        System.out.println("Connected to the database");
        Statement st = con.createStatement();
        st.executeUpdate("insert into student values('32','James')");
        System.out.println("  row inserted");
        con.close();
        System.out.println("Connection closed...");
    }
}
```

Program to Update a Record in db

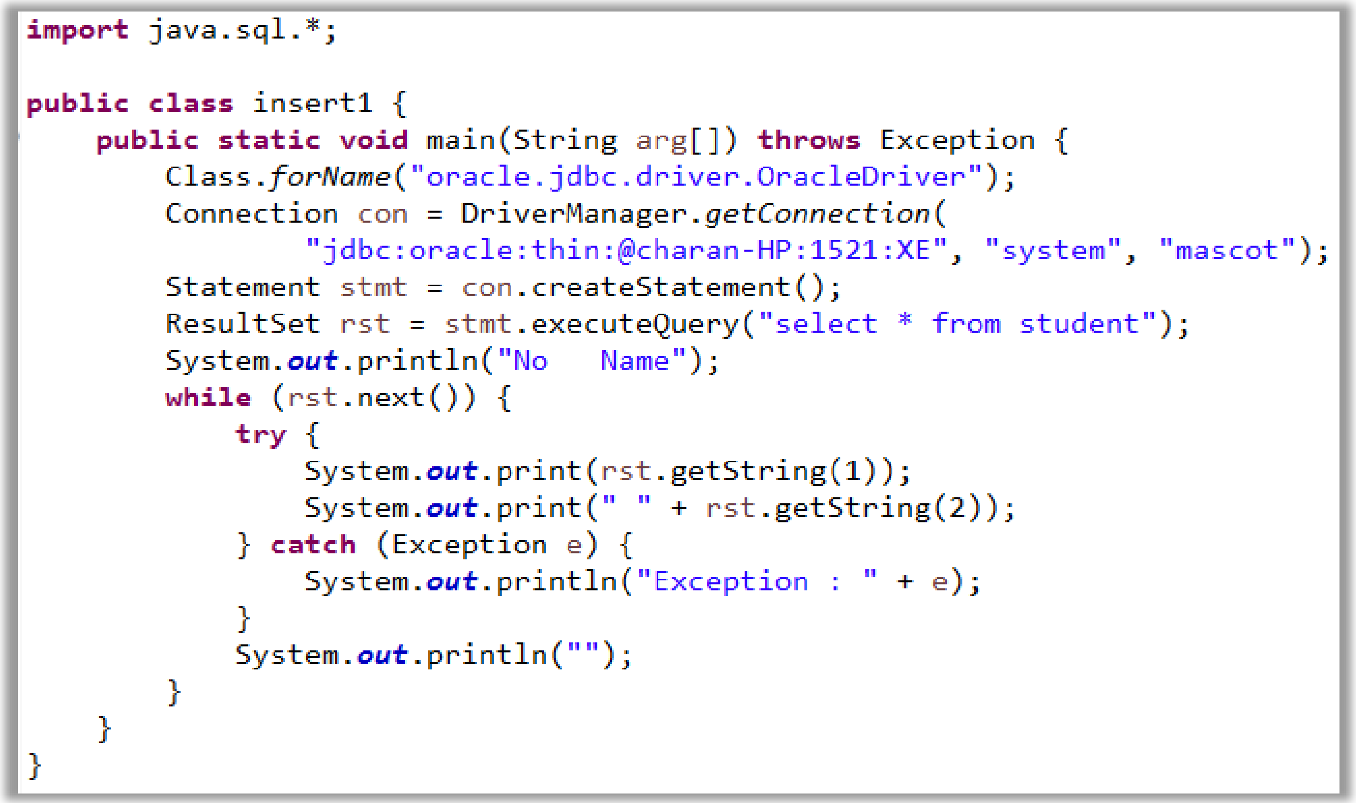


```
import java.sql.*;

public class createtable {
    public static void main(String args[]) throws Exception {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        System.out.println("Driver loaded...");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@charan-HP:1521:XE", "usr id", "password");
        System.out.println("Connected to the database");
        Statement st = con.createStatement();
        st.executeUpdate("update student set name='Pavas sisaudia' where name='Pavas'");
        System.out.println("row updated");
        con.close();
        System.out.println("Connection closed...");
    }
}
```

This program will update a record if the name is Pavas to Pavas sisaudia in the student table.

To Display Records using Select



```
import java.sql.*;

public class insert1 {
    public static void main(String arg[]) throws Exception {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@charan-HP:1521:XE", "system", "mascot");
        Statement stmt = con.createStatement();
        ResultSet rst = stmt.executeQuery("select * from student");
        System.out.println("No   Name");
        while (rst.next()) {
            try {
                System.out.print(rst.getString(1));
                System.out.print(" " + rst.getString(2));
            } catch (Exception e) {
                System.out.println("Exception : " + e);
            }
            System.out.println("");
        }
    }
}
```

To Display Records using Select (contd.)

When SQL Select query is given to the database, there is a possibility that database returns multiple records. These records are stored in ResultSet Object.

ResultSet is like a pointer for the result obtained from the Select query.

ResultSet object will be pointing to before the first record initially. When next() is called in the loop, recordset pointer moves to the first record. Every time next() is called cursor moves to the next record. record.

getString(column number) → Returns the data of the column given by the column number in the record pointed by resultset.

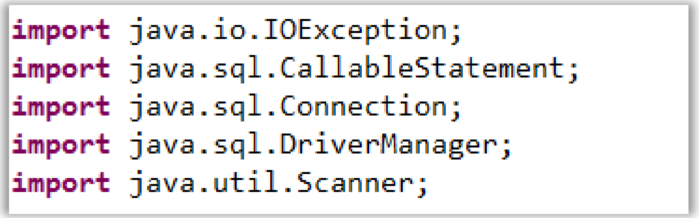
Calling Stored Procedures - Imports

Creating a Stored Procedure:

Create a stored procedure in MySQL with the below given code:

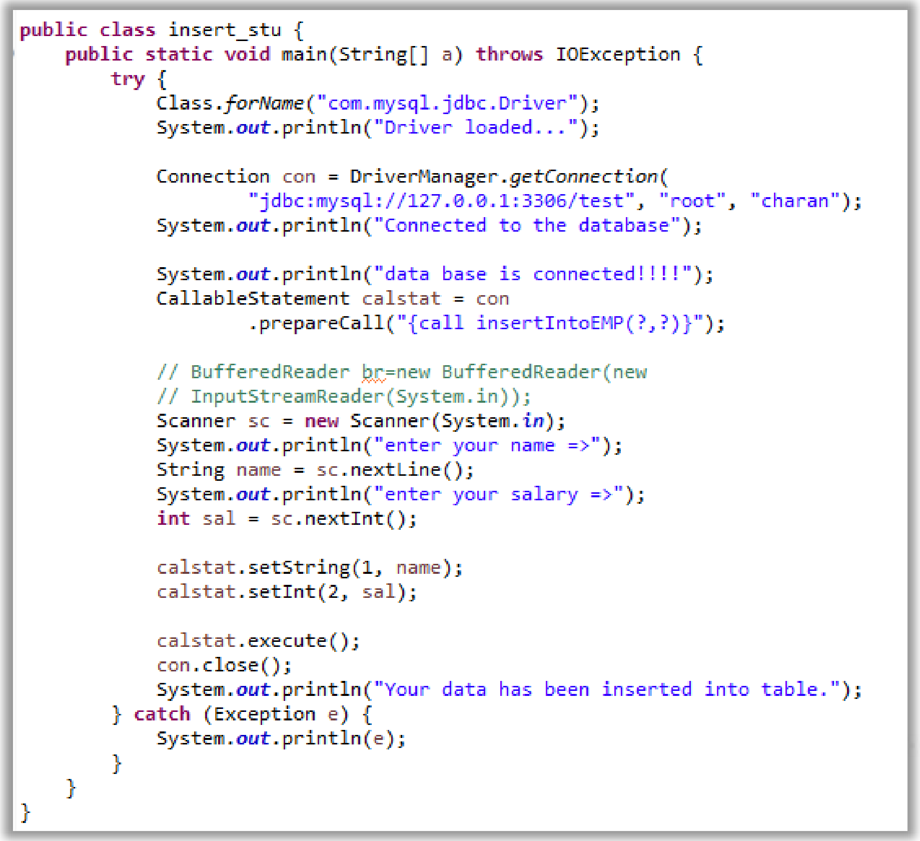
```
CREATE  PROCEDURE insertIntoEMP(IN name1 VARCHAR(50), IN sal1 INT)
BEGIN
insert into emp values (name1, sal1);
END
```

Imports Required:



```
import java.io.IOException;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.util.Scanner;
```

Calling Stored Procedures - Example



```
public class insert_stu {
    public static void main(String[] a) throws IOException {
        try {
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("Driver loaded...");

            Connection con = DriverManager.getConnection(
                    "jdbc:mysql://127.0.0.1:3306/test", "root", "charan");
            System.out.println("Connected to the database");

            System.out.println("data base is connected!!!!");
            CallableStatement calstat = con
                    .prepareCall("{call insertIntoEMP(?,?)}");

            // BufferedReader br=new BufferedReader(new
            // InputStreamReader(System.in));
            Scanner sc = new Scanner(System.in);
            System.out.println("enter your name =>");
            String name = sc.nextLine();
            System.out.println("enter your salary =>");
            int sal = sc.nextInt();

            calstat.setString(1, name);
            calstat.setInt(2, sal);

            calstat.execute();
            con.close();
            System.out.println("Your data has been inserted into table.");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Calling Stored Procedures – Methods Used

In a JDBC program:

→ prepareCall() method is used to specify which stored procedure is to be executed.

→ Parameters can be filled in at runtime as using set() methods.

→ prepareCall() will return CallableStatement object to invoke stored procedures.

→ With execute() of CallableStatement object, stored procedure is executed.

JDBC - Exceptions

→ JDBC returns SQLException.

→ Object of SQLException in the catch block can be printed to see the exception.

→ This exception has getErrorCode() method which retrieves the vendor-specific exception code for this SQLException object.

Few basic methods are:

Method	Description
getErrorCode( )	Gets the error number for the exception.
getMessage( )	Gets the JDBC driver's error message for an error
printStackTrace( )	Prints the current exception, or throwable, and its backtrace to a standard error stream.
printStackTrace(PrintStream s)	Prints this throwable and its backtrace to the print stream you specify.

Example - Batch Processing and Transaction Management

Transaction management: When multiple operations are performed on the database, if one or more transaction fails then it is required to bring the database to the original state.

For example, if the bank wants to perform all the debit and credit transactions done on that day and if few transactions fail then it is not the correct state of the database data after performing the rest of the transactions. To maintain the correct state of the data, transaction management is required. By using Commit(), changes done in the db will be made permanent. By using RollBack() changes done in the db will be removed.

Batch Processing: When multiple sql queries are performed one after the other then JDBC will fire so many SQL commands. If 1000 queries need to be executed then 1000 times database is contacted for database operation. This will degrade the performance of the project.
To increase the performance, all the queries are placed in one batch and is transferred to the database as one batch and executed on the database.

Annie's Question

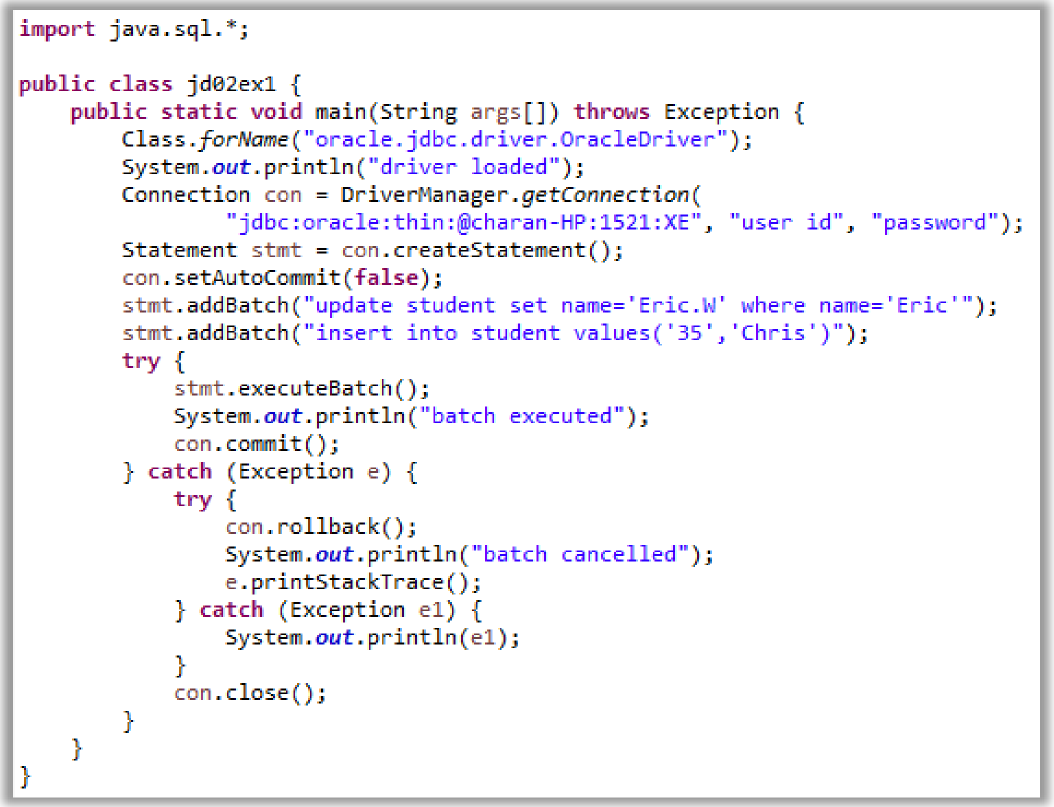
Where do we need batch processing?

Annie’s Answer

edureka!

We can save time if we put all the required statements as part of batch as it reduces the human intervention for every statement execution.

Program on Batch Processing and Transaction Management



```java
import java.sql.*;

public class jd02ex1 {
    public static void main(String args[]) throws Exception {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        System.out.println("driver loaded");
        Connection con = DriverManager.getConnection(
                "jdbc:oracle:thin:@charan-HP:1521:XE", "user id", "password");
        Statement stmt = con.createStatement();
        con.setAutoCommit(false);
        stmt.addBatch("update student set name='Eric.W' where name='Eric'");
        stmt.addBatch("insert into student values('35','Chris')");
        try {
            stmt.executeBatch();
            System.out.println("batch executed");
            con.commit();
        } catch (Exception e) {
            try {
                con.rollback();
                System.out.println("batch cancelled");
                e.printStackTrace();
            } catch (Exception e1) {
                System.out.println(e1);
            }
            con.close();
        }
    }
}
```

Program on Batch Processing and Transaction Management (contd.)

```
catch(Exception e)
{
try{
con.rollback();
System.out.println("batch cancelled");
e.printStackTrace();
}
catch(Exception e1)
{
System.out.println(e1);
}
con.close();
} } }
```

Batch Processing and Transaction Management(contd.)

By using the method setAutoCommit(false), developer has to use commit() and rollback() manually otherwise JDBC will take care.

AddBatch() is used to set of SQL commands of the batch.

ExecuteBatch() is used to execute the SQL statements in the batch.

If everything goes well then commit() is executed to make the transactions permanent.

If there is an exception then rollback() is executed to revert the changes done to the database from this batch commands.

edureka!

QUESTIONS

Assignment

Question Statement 1:

Write programs to create/insert/update/delete/select student table in the db. Student table will have the following fields:

→ Student ID
→ Name
→ Class
→ Marks

Question Statement 2:

Write a program to perform Batch processing and Transaction management for Student table created in Problem 1.

Agenda for the Next Class

edureka!

In the next class, you will be able to:

→ Understand Client Server Architecture

→ Understand Server Types

→ Use Http and its methods

→ Understand servlet types and exceptions

→ Implement servlets

→ Understand and Implement Session Tracking

→ Use Filters

Pre-work

- Read web technologies.

Survey

Your feedback is important to us, be it a compliment, a suggestion or a complaint. It helps us to make the course better!

Please spare few minutes to take the survey after the webinar.

edureka!