Assignment no:2

Title:
Client-application data-server,three tier.

Problem Definition: DBMS using connections(Client-application server data-server,three tier)oracle/mysql(odbc/jdbc),sql joints prompts.

Learning Objective:
1.To learn about three-tier architecture.
2.To learn about jsp.
Learning Outcome:
After successfully completing this assignment,we learnt about jsp,jdbc,mysql.

Software hardware requirements:
1.Apache-tomcat 8.0.9.
2.Fedora 20,eclipse kepler.
3.MYSQL.

Mathematical Model:

Let the solution system be S.

S={s,e,X,Y,DD,NDD,Fme,F}
$where,$
$s\rightarrow$Start State
e$\rightarrow$End State
X$\rightarrow$Set of Inputs
Y$\rightarrow$Output
DD$\rightarrow$Deterministic Data

NDD$\rightarrow$Non Deterministic Data

F_{me} $\rightarrow$Main Function={f1,f2,f3}

F_{f1}:Function to establish the connection.

F_{f2}:Function to execute query.

F_{f3} : Function to display result.

F_{friend} $\rightarrow$Friend Function={f$_{query1}, f_{query2}, f_{query3}, f_{query4}, f_{query5}$}

f_{query1} : find names of professors who do not work for computer dept.
$X = \{X1, X2, X3\}$
$Y : Y \epsilon X3(dno, dname)'COMP'$

f_{query2} : find names of professor who works in city pune.
$X = \{X1, X2, X3\}$
$Y : Y_i \epsilon X, city \epsilon' pune'$

f_{query3} : find names of employee who works indept pune or mumbai.
$X = \{X1, X2, X3\}$
$Y : Y_i \epsilon X, city \epsilon' pune' or' mumbai'$.

f_{query3} : find names,address,city of professors who work for computer earn more than 50,000.
$X = \{X1, X2, X3\}$
$Y : Y_i \epsilon deptname' comp' sal > 50000$.

f_{query5} : show professor id,name.salary,dno,average salary in their department.
$X = \{X1, X2, X3\}$
$Y : Y_i \epsilon X1, X2, X3, avg(sal)$.

Let X be the input to the system S. $X = \{X1, X2, X3\}$
$X1 = \{id, pname, addr, city, age, doj, sal\}$
where,
id:{id|$id \epsilon [0-9]*$}
$pname$:{pname|$pname \epsilon [A-Z] * [a-z]*$}
$addr$:{addr|$addr \epsilon [a-z] * [A-Z] * [0-9]*$}
age:{age|$age \epsilon [0-9]*$}
doj:{doj|$doj \epsilon [0-9]*$}
sal:{sal|$sal \epsilon [0-9]*$}

$X2 = \{id, dno\}$
dno:{dno|$dno \epsilon [0-9]*$}

$X3 = \{dno, dname, addr, city\}$
$dname$:{dname|$dname \epsilon [A-z] * [a-z]*$}
$addr$:{addr |$addr \epsilon [A-Z] * [a-z] * [0-9]*$}
$city$:{city|$city \epsilon [A-Z] * [a-z] * [0-9]*$}

$Y = \{Y1, Y2, Y3\}$
$Y1$: set of output
$Y1_i = \{x1, x2, ...xn\}$
$x_i \epsilon X_1$

$Y2$: set of output
$Y2_i = \{x1, x2, ...xn\}$
$x_i \epsilon X_2$

$Y3$: set of output

$Y3_i = \{x1, x2, ...xn\}$
$x_i \epsilon X_3$

Success case:It is a case when all inputs are entered correctly.
Failure case:It is a case where input does not match the validation criteria.

Deterministic Data:

- $I \epsilon \phi i.e. all constraints which are given to input$

Non-Deterministic Data:

- $I \phi i.e. input is not within constraint$

Theory:

JOINS:
A SQL join clause combines records from two or more tables in a database. It creates a set that can be saved as a table or used as it is. A JOIN is a means for combining fields from two tables by using values common to each. ANSI-standard SQL specifies five types of JOIN: INNER, LEFT OUTER, RIGHT OUTER, FULL OUTER and CROSS. As a special case, a table (base table, view, or joined table) can JOIN to itself in a self-join.

A natural join is a type of equi-join where the join predicate arises implicitly by comparing all columns in both tables that have the same column-names in the joined tables. The resulting joined table contains only one column for each pair of equally named columns. In the case that no columns with the same names are found, a cross join is performed.

An outer join does not require each record in the two joined tables to have a matching record. The joined table retains each recordeven if no other matching record exists. Outer joins subdivide further into left outer joins, right outer joins, and full outer joins, depending on which table's rows are retained (left, right, or both).

A right outer join (or right join) closely resembles a left outer join, except with the treatment of the tables reversed. Every row from the "right" table (B) will appear in the joined table at least once. If no matching row from the "left" table (A) exists, NULL will appear in columns from A for those records that have no match in B.

Conceptually, a full outer join combines the effect of applying both left and right outer joins. Where records in the FULL OUTER JOINed tables do not match, the result set will have NULL values for every column of the table that lacks a matching row. For those records that do match, a single row will be produced in the result set (containing fields populated from both tables).

JSP:

JSP allows Java code and certain pre-defined actions to be interleaved with static web markup content, with the resulting page being compiled and executed on the server to deliver a document.

The compiled pages, as well as any dependent Java libraries, use Java bytecode rather than a native software format. Like any other Java program, they must be executed within a Java virtual machine (JVM) that integrates with the server's host operating system to provide an abstract platform-neutral environment.

JSP pages use several delimiters for scripting functions.
Performance is significantly better because JSP allows embedding dynamic elements in HTML pages itself instead of having a separate CGI files.

JSP are always compiled before it proceed by the server unlike CGI/Perl which requires the server to load an interpreter and target script each time the page is requested.

They can be used in combination with serclet thet handle the business logic,the model supported by Java servlet template engineers.

Three-tier archtecture:

Multi-tier architecture (often referred to as n-tier architecture) is a clientserver architecture in which presentation, application processing, and data management functions are physically separated. The most widespread use of multi-tier architecture is the three-tier architecture.
N-tier application architecture provides a model by which developers can create flexible and reusable applications. By segregating an application into tiers, developers acquire the option of modifying or adding a specific layer, instead of reworking the entire application.

A Three-tier architecture is typically composed of a presentation tier, a domain logic tier, and a data storage tier.

A three-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms.

Three-tier architecture is a software design pattern and a well-established software architecture.

The three tiers in a three-tier architecture are:

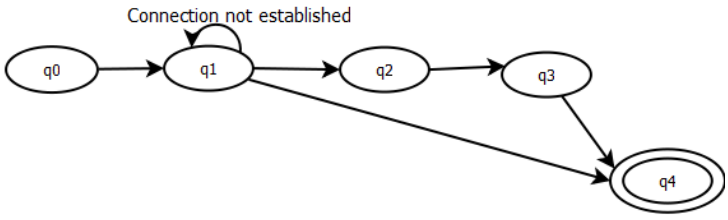
Presentation Tier: Occupies the top level and displays information related to services available on a website. This tier communicates with other tiers by sending results to

the browser and other tiers in the network.

Application Tier: Also called the middle tier, logic tier, business logic or logic tier, this tier is pulled from the presentation tier. It controls application functionality by performing detailed processing.

Data Tier: Houses database servers where information is stored and retrieved. Data in this tier is kept independent of application servers or business logic.

State Diagram:



q0:Start state
q1:Connecting to database.
q2:Insert entries into table.
q3:Display the queries.
q4:Exit state.

Conclusion:

Hence we have successfully executed the assignment for three-tier architecture.We learnt about HTML tags,JSP tags.