

CHAPTER-1

INTRODUCTION

1.1 Database:

A database is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex, they are often developed using formal design and modeling techniques.

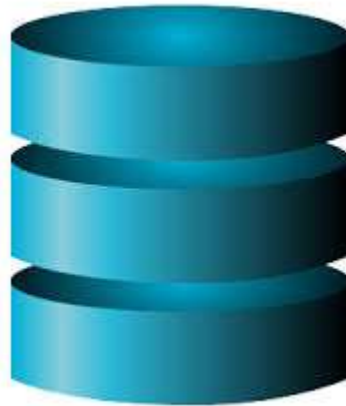


Fig. 1.1. Database icon

There are four main types of database organization:

- **Relational Database:** Data is organized as logically independent tables. Relationships among tables are shown through shared data. The data in one table may reference similar data in other tables, which maintains the integrity of the links among them. This feature is referred to as referential integrity – an important concept in a relational database system. Operations such as "select" and "join" can be performed on these tables. This is the most widely used system of database organization.

- **Flat Database:** Data is organized in a single kind of record with a fixed number of fields. This database type encounters more errors due to the repetitive nature of data.
- **Object-Oriented Database:** Data is organized with similarity to object-oriented programming concepts. An object consists of data and methods, while classes group objects having similar data and methods.
- **Hierarchical Database:** Data is organized with hierarchical relationships. It becomes a complex network if the one-to-many relationship is violated.

1.2 DBMS:

The Database Management System (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS software additionally encompasses the core facilities provided to administer the database. The sum of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database.



Fig. 1.2. Structure of DBMS

DBMS is a software package designed to define, manipulate, retrieve and manage data in a database. A DBMS generally manipulates the data itself, the data format, field names, record structure and file structure. It also defines rules to validate and manipulate this data.

A DBMS relieves users of framing programs for data maintenance. Fourth-generation query languages, such as SQL, are used along with the DBMS package to interact with a database.

Computer scientists may classify database-management systems according to the database models that they support. Relational databases became dominant in the 1980s. These model data as rows and columns in a series of tables, and the vast majority use SQL for writing and querying data. In the 2000s, non-relational databases became popular, referred to as NoSQL because they use different query languages.

Some other DBMS examples include:

- MySQL
- SQL Server
- Oracle
- dBASE
- FoxPro

1.3 SQL:

SQL (Structured Query Language) is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e. data incorporating relations among entities and variables.

Types of SQL Statements:

SQL statements are categorized into four different type of statements, which are

1. DML (Data Manipulation Language)
2. DDL (Data Definition Language)
3. DCL (Data Control Language)

4. TCL (Transaction Control Language)

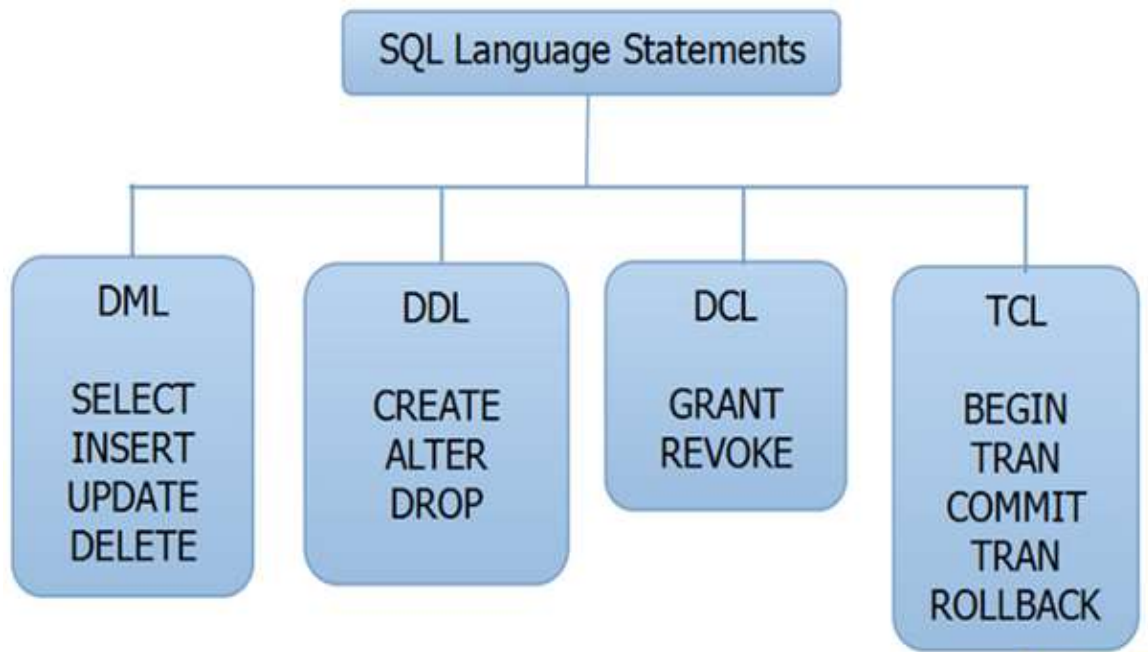


Fig. 1.3. Types of SQL statements

1.4 Advantages of DBMS:

1.4.1. Reducing Data Redundancy:

The file-based data management systems contained multiple files that were stored in many different locations in a system or even across multiple systems. Because of this, there were sometimes multiple copies of the same file which lead to data redundancy.

This is prevented in a database as there is a single database and any change in it is reflected immediately. Because of this, there is no chance of encountering duplicate data.

1.4.2. Sharing of Data:

In a database, the users of the database can share the data among themselves. There are various levels of authorisation to access the data, and consequently the data can only be shared based on the correct authorisation protocols being followed. Many remote users can also access the database simultaneously and share the data between themselves.

1.4.3. Data Integrity:

Data integrity means that the data is accurate and consistent in the database. Data Integrity is very important as there are multiple databases in a DBMS. All these databases contain data that is visible to multiple users. So, it is necessary to ensure that the data is correct and consistent in all the databases and for all the users.

1.4.4. Data Security:

Data Security is vital concept in a database. Only authorised users should be allowed to access the database and their identity should be authenticated using a username and password. Unauthorised users should not be allowed to access the database under any circumstances as it violates the integrity constraints.

1.4.5. Privacy:

The privacy rule in a database means only the authorized users can access a database according to its privacy constraints. There are levels of database access and a user can only view the data he can. For example - In social networking sites, access constraints are different for different accounts a user may want to access.

1.4.6. Backup and Recovery:

Database Management System automatically takes care of backup and recovery. The users don't need to backup data periodically because this is taken care of by the DBMS. Moreover, it also restores the database after a crash or system failure to its previous condition.

1.4.7. Data Consistency:

Data consistency is ensured in a database because there is no data redundancy. All data appears consistently across the database and the data is same for all the users viewing the database. Moreover, any changes made to the database are immediately reflected to all the users and there is no data inconsistency.

CHAPTER-2

LITERATURE SURVEY

2.1 Introduction:

Generally, when we consider any database with the interface provided by that database, we can handle only one kind of database and the interface is also not user-friendly and gives burden to the user to form his own queries.

2.2 Problem Statement:

The existing system has following drawbacks.

- Heavy load on memory to remember syntaxes.
- User sometimes may not form correct query for requirement.
- Different consoles for different databases.
- Different syntaxes for different databases.
- User feels uneasy to shift between databases to meet his requirements.

2.3 Proposed Work:

- The proposed system provides a single interface to interact with different databases.
- We can run different SQL statements from the browser.
- The results will be displayed on the user interface.
- User can create his own account to utilize the services.
- User feels easy to perform tasks while using the user interface.

2.4 Advantages:

- No need to remember queries to perform operations on databases.
- User friendly interface.
- We can handle multiple databases using a single console.
- Only registered users can utilize the website.
- The results will be displayed on the user interface (UI).

2.5 System Requirements:

2.5.1 Hardware requirements:

- System RAM : 3 Gb
- Hard disk : 20Gb
- Processor : Intel Pentium-IV based

2.5.2 Software Requirements:

- Web Presentation : HTML
- Client-Side Validation : JAVASCRIPT
- Programming Language : Java
- Database Connectivity API : JDBC
- Backend Database : Oracle, My-SQL
- Operating System : Windows XP/2000/2003
- J2EE Web/Application Server : Tomcat
- Browser : IE/Mozilla

CHAPTER-3

SYSTEM DESIGN

3.1 Modules:

3.1.1 Connection module: This module specifically provides the Connection form Database by getting the values from form page.

3.1.2 DB Operations module: This module maintains the information specific to all the Operations of Database.

3.1.3 Create module: This module creates the tables by getting the values from create page.

3.1.4 Drop module: The module maintains the information related to the total sales that are executed upon the system by the sales people as part of their marketing activity. The total atomic information related to the sales is maintained here along with the technical and non-technical information that becomes more important for future reference.

3.1.5 Insert module: This module allows to insert the values into tables created by the users.

3.1.6 Select module: This module displays the user table values on the browser.

3.1.7 Delete module: This module allows the user to delete table rows in the DB from browser.

3.2 UML (Unified Modelling Language):

UML stands for Unified Modelling Language which is used in object-oriented software engineering. Although typically used in software engineering it is a rich language that can be used to model an application structures, behavior and even business processes. Which is designed to provide a standard way to visualize the design of a system.

It was created and developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software during 1994–95 with further development led by them through 1996. In 1997 it was adopted as a standard by the Object Management Group (OMG), and has been managed by this organization ever since. In 2000 the Unified Modelling Language was also accepted by the International Organization for Standardization (ISO) as an approved ISO.

3.2.1 Class Diagram:

Class diagrams are arguably the most used UML diagram type. It is the main building block of any object-oriented solution. It shows the classes in a system, attributes and operations of each class and the relationship between each class. In most modelling tools a class has three parts, name at the top, attributes in the middle and operations or methods at the bottom.

3.2.2 Component Diagram:

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems that has many components. Components communicate with each other using interfaces. The interfaces are linked using connectors.

3.2.3 Deployment Diagram:

A deployment diagram shows the hardware of your system and the software in that hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration.

3.2.4 Object Diagram:

Object Diagrams, sometimes referred as Instance diagrams are very similar to class diagrams. As class diagrams they also show the relationship between objects but they use real world examples. They are used to show how a system will look like at a given time

3.2.5 Use Case Diagram:

Most known diagram type of the behavioral UML diagrams, use case diagrams gives a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions are interacted. It's a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system.

3.2.6 Activity Diagram:

Activity diagrams represent workflows in a graphical way. They can be used to describe business workflow or the operational workflow of any component in a system. Sometimes activity diagrams are used as an alternative to State machine diagrams.

3.2.7 State Machine Diagram:

State machine diagrams are like activity diagrams although notations and usage change a bit. They are sometime known as state diagrams or start chart diagrams as well. These are very useful to describe the behavior of objects that act different according to the state they are now.

3.2.8 Sequence Diagram:

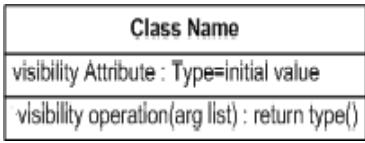
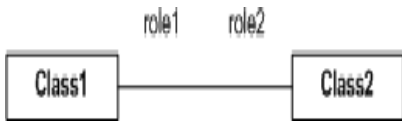



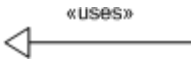

Sequence diagrams in UML shows how object interact with each other and the order those interactions occur. It's important to note that show the interactions for a scenario. The processes are represented vertically and interactions are show as arrows.



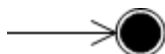


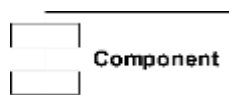



3.2.9 Collaboration Diagram:


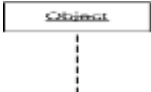
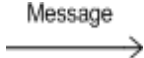
It is like sequence diagrams but the focus is on messages passed between objects.

The same information can be represented using a sequence diagram and different objects.

3.3 List of UML Notations:

S.NO	SYMBOL NAME	SYMBOL	DESCRIPTION
1	Class		Classes represent a collection of similar entities grouped together.
2	Association		Association represents a static relationship between classes.
3	Aggregation		Aggregation is a form of association. It aggregates several classes into single class.
4	Actor		Actors are the users of the system and other external entity that react with the system.
5	Use Case		A use case is an interaction between the system and the external environment.
6	Relation (Uses)		It is used for additional process communication
7	Communication		It is the communication between various use cases.

8	State		It represents the state of a process. Each state goes through various flows.
9	Initial State		It represents the initial state of the object.
10	Final State		It represents the final state of the object.
11	Control Flow		It represents the various control flow between the states.
12	Decision Box		It represents the decision-making process from a constraint.
13	Component		Components represent the physical components used in the system
14	Node		Deployment diagrams use the nodes for representing physical modules, which is a collection of components.
15	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.
16	External Entity		It represents any external entity such as keyboard, sensors etc.

17	Transition		It represents any communication that occurs between the processes.
18	Object Lifeline		Object lifelines represents the vertical dimension that objects communicate.
19	Message		It represents the messages exchanged.

Goals:

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

3.4 UML Diagrams:

3.4.1 Class Diagram:

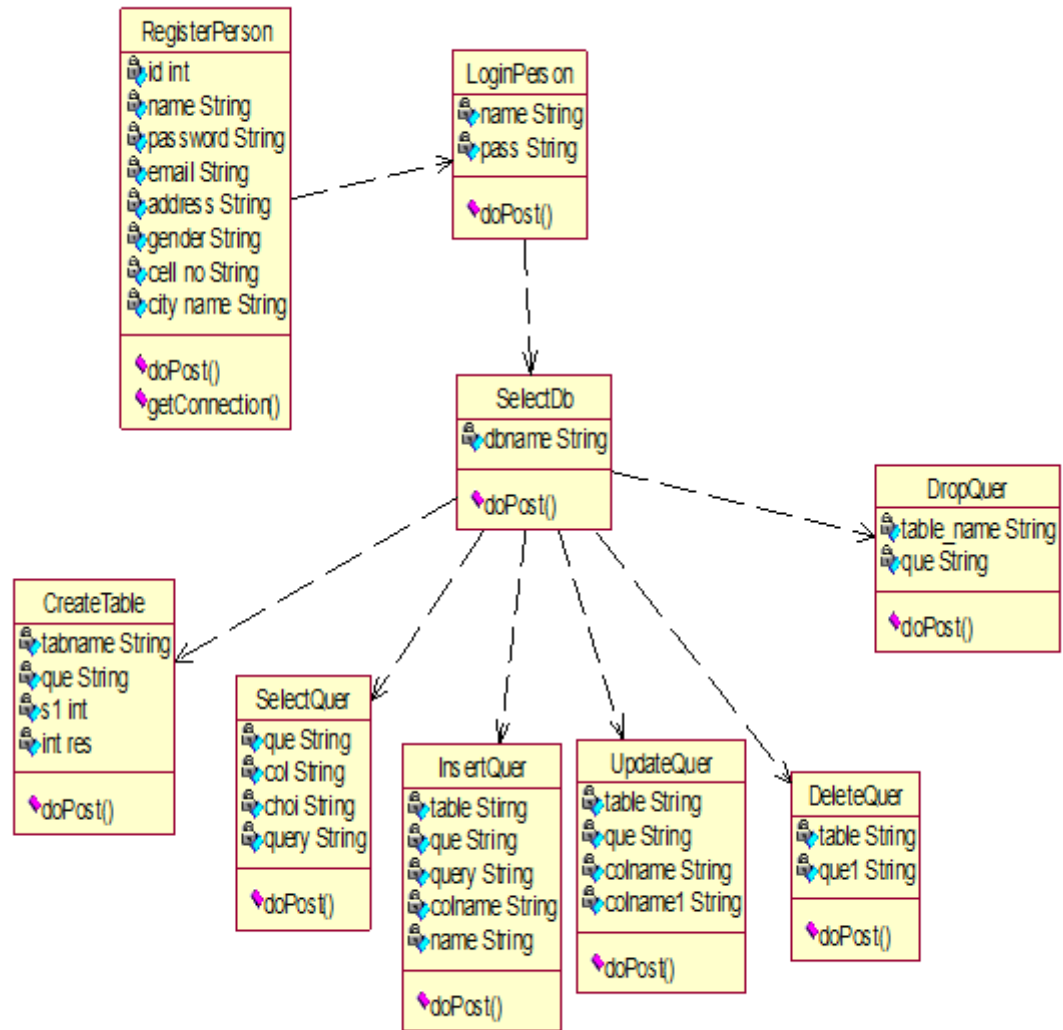


Fig 3.4.1 Class Diagram

3.4.2 Sequence Diagram:

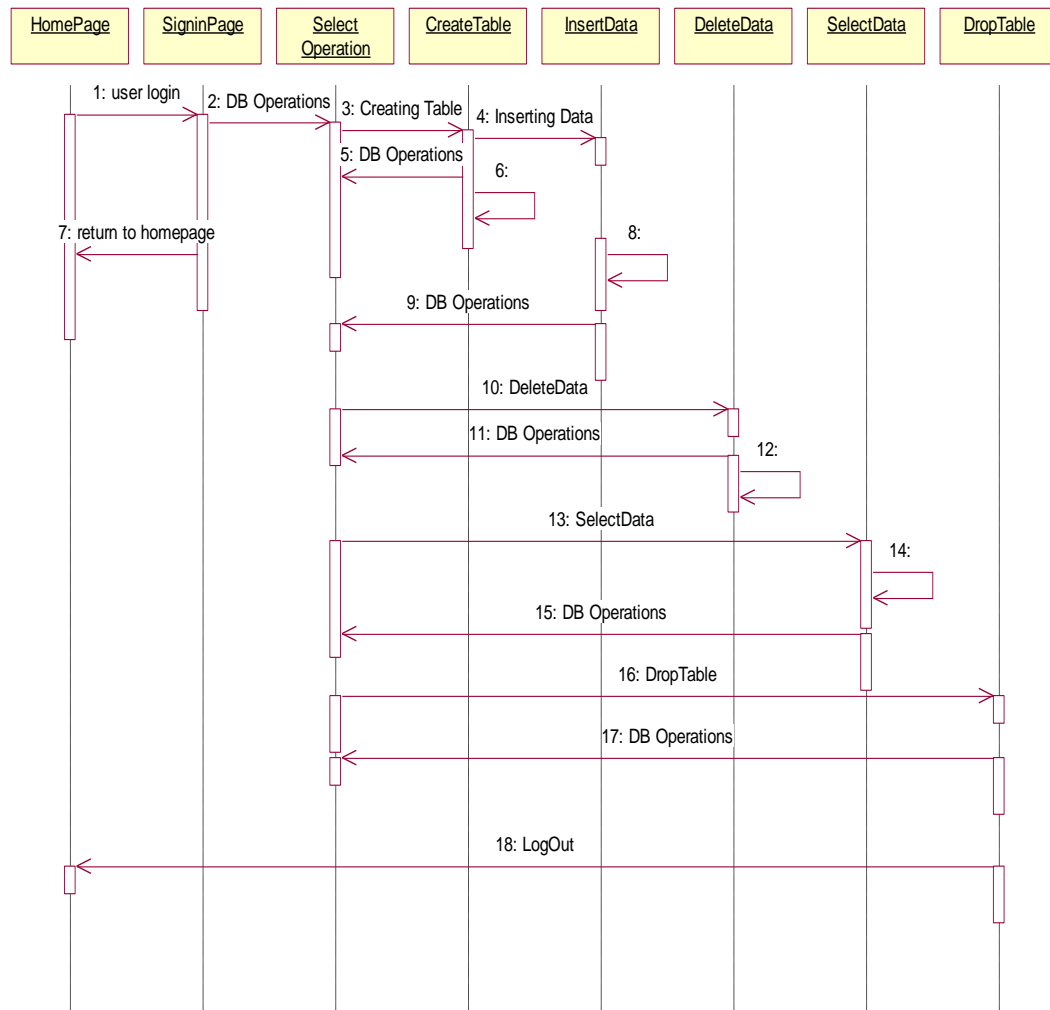


Fig 3.4.2. Sequence Diagram

3.4.3 Collaboration Diagram:

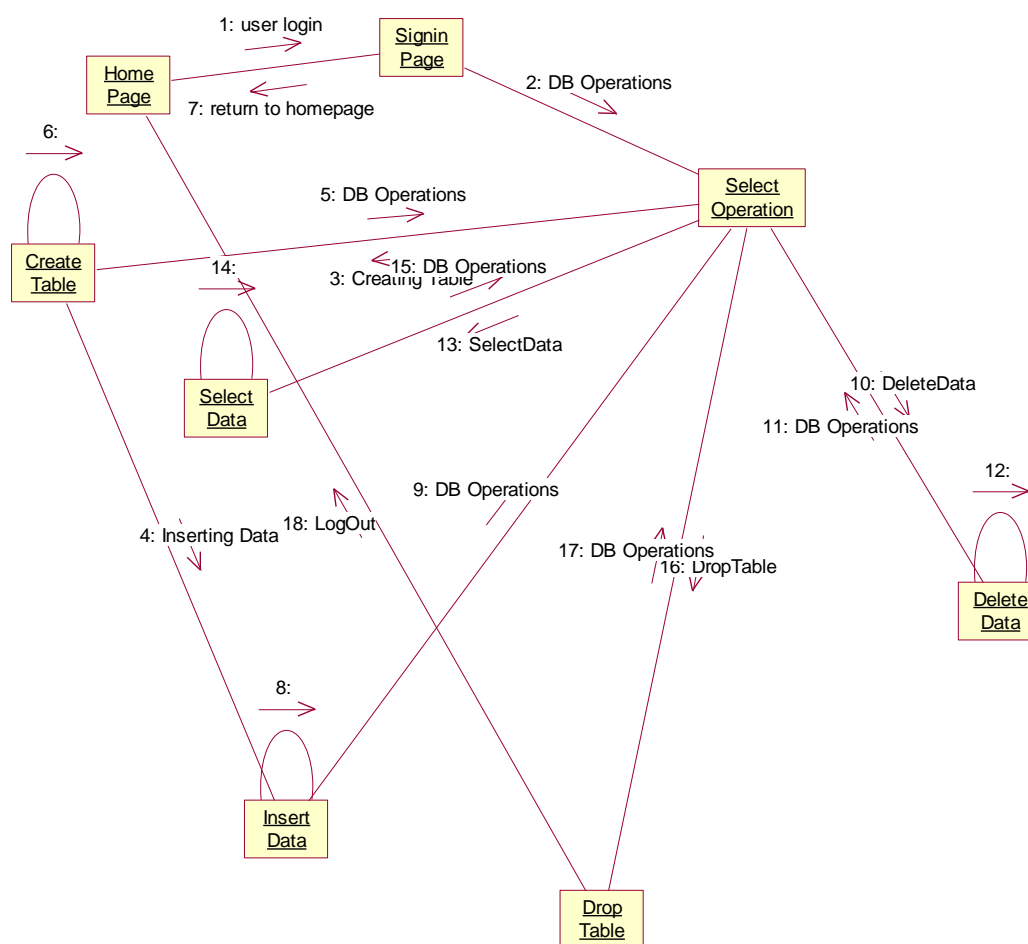


Fig. 3.4.3. Collaboration Diagram

3.4.4 Use Case Diagram:

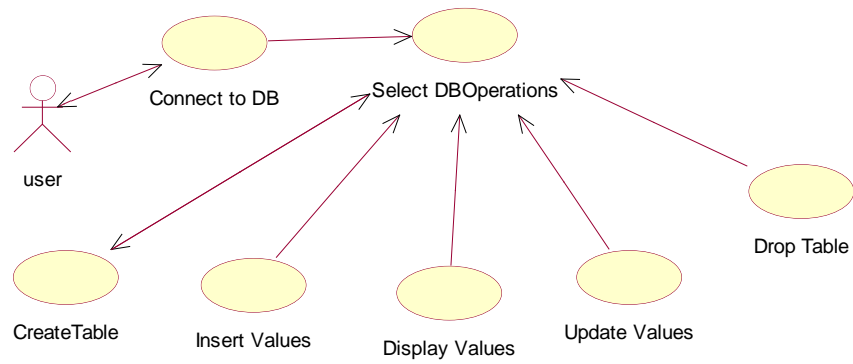


Fig.3.4.4 Use Case Diagram

CHAPTER-4

SYSTEM IMPLEMENTATION

4.1 Technologies:

4.1.1 Java Technology:

Initially the language was called as “oak” but it was renamed as “Java” in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer’s language.
- Java is cohesive and consistent.
- Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.

Finally, Java is to Internet programming where C was to system programming.

Importance of Java to the Internet:

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

Java can be used to create two types of programs:

Applications and Applets: An application is a program that runs on our Computer under the operating system of that computer. It is like one creating using C or C++. Java’s

ability to create Applets makes it important. An Applet is an application designed to be transmitted over the Internet and executed by a Java –compatible web browser. An applet is a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

Features of Java:

1) Security:

Every time you that you download a “normal” program, you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scanned them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a “firewall” between a network application and your computer.

When you use a Java-compatible Web browser, you can safely download Java applets without fear of virus infection or malicious intent.

2) Portability:

For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed. As you will see, the same mechanism that helps ensure security also helps create portability. Indeed, Java’s solution to these two problems is both elegant and efficient.

3) The Byte code:

The key that allows the Java to solve the security and portability problems is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions

designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for byte code.

Translating a Java program into byte code helps makes it much easier to run a program in a wide variety of environments. The reason is, once the run-time package exists for a given system, any Java program can run on it.

Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-fly compilation of byte code into native code. Sun has just completed its Just In Time (JIT) compiler for byte code. When the JIT compiler is a part of JVM, it compiles byte code into executable code in real time, on a piece-by-piece, demand basis. It is not possible to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. The JIT compiles code, as it is needed, during execution.

4) Java Virtual Machine (JVM):

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that's has been generated by the compiler will not corrupt the machine that it's loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So, byte code verification is integral to the compiling and executing of Java code.

Overall Description:

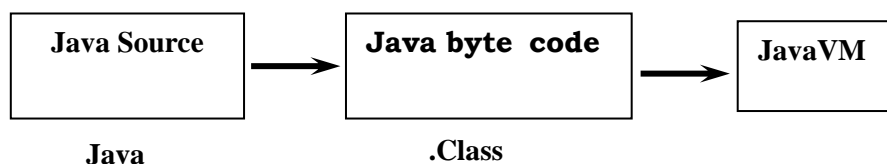


Fig. 4.1 Picture showing the development process of JAVA Program

Java programming uses to produce byte codes and executes them. The first box indicates that the Java source code is in a .java file that is processed with a Java compiler called javac. The Java compiler produces a file called a .class file, which contains the byte code. The .class file is then loaded across the network or loaded locally on your machine into the execution environment is the Java virtual machine, which interprets and executes the byte code.

5) Java Architecture:

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

6) Compilation of code:

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

7) Simple:

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ programmer, learning Java will be even easier. Because Java inherits the C/C++ syntax and many of the object-oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

8) Object-Oriented:

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

9) Robust:

The multi-platform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and run time. Java virtually eliminates the problems of memory management and deallocation, which is completely automatic. In a well-written Java program, all run time errors can and should be managed by your program.

4.1.2 Servlets:

Introduction:

The Java web server is Java Soft's own web Server. The Java web server is just a part of a larger framework, intended to provide you not just with a web server, but also with tools. To build customized network servers for any Internet or Intranet client/server system. Servlets are to a web server, how applets are to the browser.

About Servlets:

Servlets provide a Java-based solution used to address the problems currently associated with doing server-side programming, including inextensible scripting solutions, platform-specific APIs, and incomplete interfaces.

Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side -

object byte codes that can be dynamically loaded off the net. They differ from applets in that they are faceless objects (without graphics or a GUI component). They serve as platform independent, dynamically loadable, pluggable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

For example, an HTTP Servlets can be used to generate dynamic HTML content. When you use Servlets to do dynamic content you get the following advantages:

- They're faster and cleaner than CGI scripts
- They use a standard API (the Servlets API)
- They provide all the advantages of Java (run on a variety of servers without needing to be rewritten).

Attractiveness of Servlets:

There are many features of Servlets that make them easy and attractive to use. These include

- Easily configured using the GUI-based Admin tool
- Can be loaded and invoked from a local disk or remotely across the network.
- Can be linked together, or chained, so that one Servlets can call another Servlets, or several Servlets in sequence.
- Can be called dynamically from within HTML pages, using server-side include tags.
- Are secure - even when downloading across the network, the Servlets security model and Servlets sandbox protect your system from unfriendly behavior.

Advantages of the Servlet API:

- One of the great advantages of the Servlet API is protocol independence. It assumes nothing about:
 - The protocol being used to transmit on the net
 - How it is loaded

- The server environment it will be running in

These qualities are important, because it allows the Servlet API to be embedded in many kinds of servers. There are other advantages to the Servlet API as well. These include:

- It's extensible - you can inherit all your functionality from the base classes made available to you.
- it's simple, small, and easy to use.

Features of Servlets:

- Servlets are persistent. Servlets are loaded only by the web server and can maintain services between requests.
- Servlets are fast. Since Servlets only need to be loaded once, they offer much better performance over their CGI counterparts.
- Servlets are platform independent.
- Servlets are extensible. Java is a robust, object-oriented programming language, which easily can be extended to suit your needs
- Servlets are secure.
- Servlets can be used with a variety of clients.

Loading Servlets:

Servlets can be loaded from three places

1) From a directory that is on the CLASSPATH. The CLASSPATH of the Java Webserver includes service root/classes/ which is where the system classes reside.

2) From the <SERVICE_ROOT /Servlets/ directory. This is not in the server's class path. A class loader is used to create Servlets from this directory. New Servlets can be added - existing Servlets can be recompiled and the server will notice these changes.

3) From a remote location. For this a code base like `http://nine.eng/classes/foo/` is required in addition to the Servlets class name. Refer to the admin GUI docs on Servlet section to see how to set this up.

Loading Remote Servlets:

Remote Servlets can be loaded by:

- Configuring the Admin Tool to setup automatic loading of remote Servlets
- Setting up server side include tags in .html files
- Defining a filter chain configuration

Invoking Servlets:

A Servlet invoker is a Servlet that invokes the "service" method on a named Servlet. If the Servlet is not loaded in the server, then the invoker first loads the Servlet (either from local disk or from the network) and then invokes the "service" method. Also, like applets, local Servlets in the server can be identified by just the class name. In other words, if a Servlet name is not absolute, it is treated as local.

A client can invoke Servlets in the following ways:

- The client can ask for a document that is served by the Servlet.
- The client (browser) can invoke the Servlet directly using a URL, once it has been mapped using the Servlet Aliases section of the admin GUI.
- The Servlet can be invoked through server side include tags.
- The Servlet can be invoked by placing it in the Servlets/ directory.
- The Servlet can be invoked by using it in a filter chain.

4.1.3 JavaScript:

JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of

both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then updates the browser's display accordingly.

Even though JavaScript supports both client and server Web programming, we prefer JavaScript at Client-side programming since most of the browsers supports it. JavaScript is almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags.

<SCRIPTS>..
</SCRIPT>.

<SCRIPT LANGUAGE = "JavaScript">

//JavaScript statements

</SCRIPT>

Here are a few things we can do with JavaScript:

- Validate the contents of a form and make calculations.
- Add scrolling or changing messages to the Browser's status line.
- Animate images or rotate images that change when we move the mouse over them.
- Detect the browser in use and display different content for different browsers.
- Detect installed plug-ins and notify the user if a plug-in is required.

We can do much more with JavaScript, including creating entire application.

4.1.4 JavaScript Vs Java:

JavaScript and Java are entirely different languages. A few of the most glaring differences are:

- Java applets are generally displayed in a box within the web document; JavaScript can affect any part of the Web document itself.
- While JavaScript is best suited to simple applications and adding interactive features to Web pages; Java can be used for incredibly complex applications.

There are many other differences but the important thing to remember is that JavaScript and Java are separate languages. They are both useful for different things; in fact, they can be used together to combine their advantages.

Advantages:

- JavaScript can be used for Server-side and Client-side scripting.
- It is more flexible than VBScript.
- JavaScript is the default scripting languages at Client-side since all the browsers supports it.

4.1.5 Hyper Text Markup Language:

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

Basic HTML Tags:

<!-- -->	Specifies comments
<A>.....	Creates hypertext links
.....	Formats text as bold
<BIG>.....</BIG>	Formats text in large font.
<BODY>...</BODY>	Contains all tags and text in the HTML document
<CENTER>...</CENTER>	Creates text in center
<DD>...</DD>	Definition of a term
<DL>...</DL>	Creates definition list
...	Formats text with a particular font
<FORM>...</FORM>	Encloses a fill-out form
<FRAME>...</FRAME>	Defines a particular frame in a set of frames
<H#>...</H#>	Creates headings of different levels
<HEAD>...</HEAD>	Contains tags that specify information about a document
<HR>...</HR>	Creates a horizontal rule

<code><HTML>...</HTML></code>	Contains all other HTML tags
<code><META>...</META></code>	Provides meta-information about a document
<code><SCRIPT>...</SCRIPT></code>	Contains client-side or server-side script
<code><TABLE>...</TABLE></code>	Creates a table
<code><TD>...</TD></code>	Indicates table data in a table
<code><TR>...</TR></code>	Designates a table row
<code><TH>...</TH></code>	Creates a heading in a table

Advantages:

- A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.
- HTML is platform independent.
- HTML tags are not case-sensitive.

4.1.6 Java Database Connectivity:

What Is JDBC?

JDBC is a Java API for executing SQL statements. (As a point of interest, JDBC is a trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the Java programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API.

Using JDBC, it is easy to send SQL statements to virtually any relational database. One can write a single program using the JDBC API, and the program will be able to send SQL statements to the appropriate database. The combinations of Java and JDBC lets a programmer write it once and run it anywhere.

What Does JDBC Do?

Simply put, JDBC makes it possible to do three things:

- Establish a connection with a database
- Send SQL statements
- Process the results.

JDBC Driver Types:

The JDBC drivers that we are aware of at this time fit into one of four categories:

- JDBC-ODBC bridge plus ODBC driver
- Native-API partly-Java driver
- JDBC-Net pure Java driver
- Native-protocol pure Java driver

4.1.7 Java Server Pages (JSP):

Java server Pages is a simple, yet powerful technology for creating and maintaining dynamic-content web pages. Based on the Java programming language, Java Server Pages offers proven portability, open standards, and a mature re-usable component model. The Java Server Pages architecture enables the separation of content generation from content presentation. This separation not eases maintenance headaches, it also allows web team members to focus on their areas of expertise. Now, web page designer can concentrate on layout, and web application designers on programming, with minimal concern about impacting each other's work.

Features of JSP:

Portability:

Java Server Pages files can be run on any web server or web-enabled application server that provides support for them. Dubbed the JSP engine, this support involves recognition,

translation, and management of the Java Server Page lifecycle and its interaction components.

Components:

It was mentioned earlier that the Java Server Pages architecture can include reusable Java components. The architecture also allows for the embedding of a scripting language directly into the Java Server Pages file. The components currently supported include Java Beans, and Servlets.

Processing:

A Java Server Pages file is essentially an HTML document with JSP scripting or tags. The Java Server Pages file has a JSP extension to the server as a Java Server Pages file. Before the page is served, the Java Server Pages syntax is parsed and processed into a Servlet on the server side. The Servlet that is generated outputs real content in straight HTML for responding to the client.

Access Models:

A Java Server Pages file may be accessed in at least two different ways. A client's request comes directly into a Java Server Page. In this scenario, suppose the page accesses reusable Java Bean components that perform well-defined computations like accessing a database. The result of the Beans computations, called result sets is stored within the Bean as properties. The page uses such Beans to generate dynamic content and present it back to the client.

In both above cases, the page could also contain any valid Java code. Java Server Pages architecture encourages separation of content from presentation.

Steps in the execution of a JSP Application:

- The client sends a request to the web server for a JSP file by giving the name of the JSP file within the form tag of a HTML page.

- This request is transferred to the Java Webserver. At the server-side Java Webserver receives the request and if it is a request for a jsp file server gives this request to the JSP engine.
- JSP engine is program which can understand the tags of the jsp and then it converts those tags into a Servlet program and it is stored at the server side. This Servlet is loaded in the memory and then it is executed and the result is given back to the Java Webserver and then it is transferred back to the result is given back to the Java Webserver and then it is transferred back to the client.

4.1.8 JDBC connectivity

The JDBC provides database-independent connectivity between the J2EE platform and a wide range of tabular data sources. JDBC technology allows an Application Component Provider to:

- Perform connection and authentication to a database server
- Manager transactions
- Move SQL statements to a database engine for preprocessing and execution
- Execute stored procedures
- Inspect and modify the results from Select statements.

4.2 Sample Code:

MakeConnection.java:

```
import java.sql.*;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.DriverManager;

import javax.servlet.RequestDispatcher;
```



```
import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class MakeConnection extends HttpServlet

{

    Connection con=null;

    public void doGet(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException{

        try {

            response.setContentType("text/html");

            PrintWriter out = response.getWriter();

            RequestDispatcher

rd1=request.getRequestDispatcher("ConnFailure.jsp");

            RequestDispatcher rd=request.getRequestDispatcher("Link.jsp");

            String name=request.getParameter("dbname");

            String comname=request.getParameter("compname");

            String sidname=request.getParameter("sid");

            String users=request.getParameter("user");

            String pass=request.getParameter("password");

            //String url="jdbc:oracle:thin:@newmm9:1521:orcl";
```

```
        if(name.equals("oracle"))
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");

con=DriverManager.getConnection("jdbc:oracle:thin:"+comname+":1521:"+sidname,use
rs,pass);

            if(con!=null)
            {
                request.getSession().setAttribute("comname1",comname);

                request.getSession().setAttribute("sidname1",sidname);

                request.getSession().setAttribute("users1",users);

                request.getSession().setAttribute("pass1",pass);

            }

            rd.forward(request,response);

        }

        rd1.forward(request,response);

        out.flush();

        out.close();

    }/* catch (ClassNotFoundException e) {e.printStackTrace();}*/

    catch (Exception e) {//e.printStackTrace();}

}
```

4.3 Results:

Home Page:

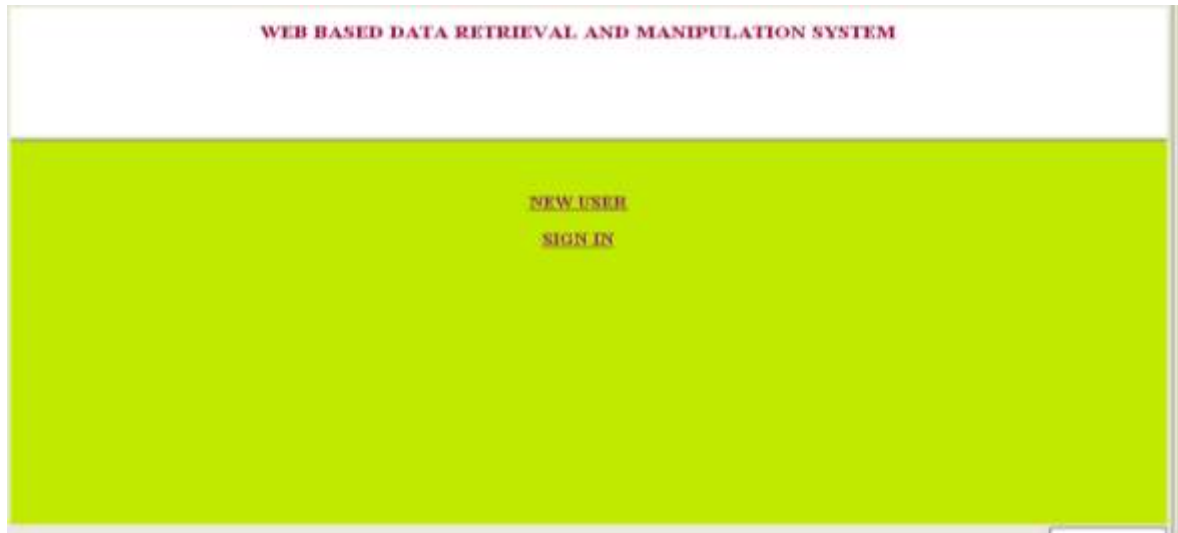


Fig 4.3.1 Home Page

Description:

- 1) On clicking NEW USER, the user navigates to the registration page.
- 2) On clicking the SIGN IN, the user navigates to the sign-in page.

New User Registration:



The screenshot shows a web browser window with a title bar. The page has a header with the text "WEB BASED DATA RETRIEVAL AND MANIPULATION SYSTEM FOR ORACLE DATABASE" in red. Below the header is a large yellow rectangular area containing a form titled "USER INFORMATION". The form has the following fields: "USER ID:" with a text input, "USERNAME" with a text input, "PASSWORD:" with a text input, "(RE ENTER PASSWORD)" with a text input, "EMAIL" with a text input, "ADDRESS:" with a text input, "GENDER" with two radio buttons labeled "Male" and "Female", "CELL NUMBER:" with a text input, and "CITY" with a text input. At the bottom right of the form are two buttons: "Register" and "Clear".

Fig 4.3.2 New User Registration

Description: Here user can enter his details to create an account in the website. After, entering all the details, the user clicks on Register to get registered.

Sign-in:



The screenshot shows a web application interface. At the top, a header bar contains the text "WEB BASED DATA RETRIEVAL AND MANIPULATION SYSTEM FOR MULTIPLE DATABASES" in red. Below the header, the main content area has a light green background. In the center, there is a "Sign In" section. It includes a "User Name" label and a text input field, followed by a "Password" label and another text input field. Below these fields are two buttons: "Submit" and "Clear". In the top right corner of the main content area, there is a "Logout" link.

Fig. 4.3.3 Sign-in Page

Description: Already registered users can enter his credentials to log-in into the website to utilize its services. User needs to click on submit after entering his credentials.

Select Operation:



Fig 4.3.4 Select Operation

Description: User needs to select which operation he needs to perform on the database among the various options displayed on the screen.

Create Table

WEB BASED DATA RETRIEVAL AND MANIPULATION SYSTEM FOR MULTIPLE DATABASES

Logout

EnterTableName:

CHECK AVAILABILITY

COLUMN NAME:

TYPE: VARCHAR2 SIZE: 5 CONSTRAINT: PRIMARY KEY

AddColumn

Create

[Back](#)

Fig.4.3.5 Create Table

Description: Here user first checks the availability of the table name and then adds required columns and finally enters table name and creates his table on clicking create table.

Inserting into Table:

WEB BASED DATA RETRIEVAL AND MANIPULATION SYSTEM FOR MULTIPLE DATABASES

[LogOut](#)

ENTER THE TABLE VALUES HERE

NAME

NO

ABCD5

ColumnName	ColumnTypeName
NAME	VARCHAR2
NO	NUMBER

[Back](#)

Fig. 4.3.6 Insert into Table

Description: Here the user enters the values which he wants to insert into the table. For the user reference the datatype of each columns is also shown.

Update Table:

The screenshot displays a web application titled "WEB BASED DATA RETRIEVAL AND MANIPULATION SYSTEM". In the top right corner, there is a "Logout" link. The main content area has a light blue background. At the top center, a table labeled "ABCD5" shows column details: "Column Name" and "Column Type Name". Below this, a table with two columns, "NAME" and "NO", is shown. The "NAME" column contains the value "orbit" and the "NO" column contains the value "735". To the right of this table is a "Button" column with an "Update" button and a "STATUS" column with a radio button. Below the table, there is a "Back" link.

Column Name	Column Type Name
NAME	VARCHAR2
NO	NUMBER

NAME	NO	Button	STATUS
orbit	735	Update	<input type="radio"/>

[Back](#)

Fig.4.3.7 Update Table

Description: Here complete table is displayed with each value in a text box. Here user can edit the value in a row where he needs to update later selects status and the clicks update to get the values updated.

Delete Row:

WEB BASED DATA RETRIEVAL AND MANIPULATION SYSTEM FOR MULTIPLE DATABASES

LogOut

ENTER THE COLUMN VALUE HERE

NAME

Delete

ABCD5

ColumnName	ColumnTypeNames
NAME	VARCHAR2
NO	NUMBER

NAME	NO
orha	234

Fig 4.3.8 Delete Row

Description: Here user enters the value. The rows which contains that value gets deleted after clicking on delete button.

Display Table:

The screenshot shows a web application titled "WEB BASED DATA RETRIEVAL AND MANIPULATION SYSTEM FOR MULTIPLE DATABASES". The interface has a pink background. In the top right corner, there is a "LogOut" link. The main content area displays "TABLE NAME: ABCD5". Below this, there is a table with two columns: "NAME" and "NO". The first row of data shows "orbit" and "234". Under the table, there is a section for sorting. It says "ORDER BY: NAME" with a dropdown arrow. Below that, there are two radio buttons: "ASCENDING" (which is selected) and "DESCENDING". At the bottom of this section, there is a "Submit" button and a "Back" link.

NAME	NO
orbit	234

ORDER BY: NAME

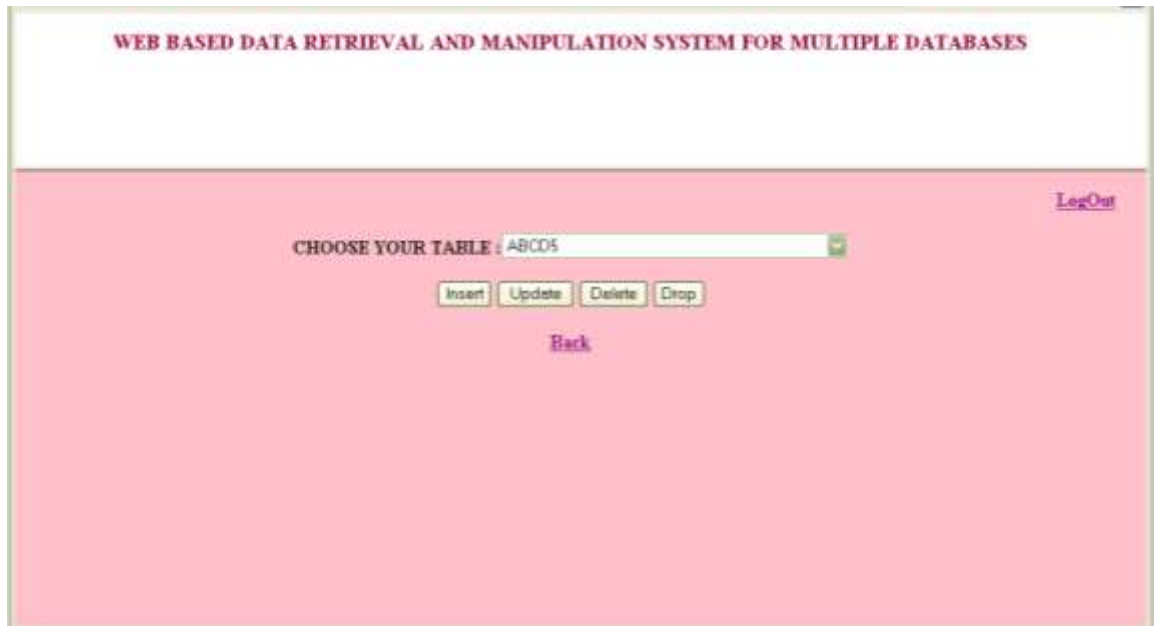
☒ ASCENDING ☐ DESCENDING

[Back](#)

Fig 4.3.9 Display Table

Description: Here user can see complete table and sort them as he wishes.

Drop Table:



The screenshot shows a web application interface with a title bar at the top that reads "WEB BASED DATA RETRIEVAL AND MANIPULATION SYSTEM FOR MULTIPLE DATABASES". Below the title bar, on the right side, is a "LogOut" link. The main content area has a pink background and contains a form titled "CHOOSE YOUR TABLE :". The form includes a text input field with the value "ABCD5" and a green dropdown arrow on the right. Below the input field are four buttons: "Insert", "Update", "Delete", and "Drop". At the bottom of the form is a "Back" link.

Fig 4.3.10 Drop Table

Description: Here user can select a table to drop it from the database.

CHAPTER-5

SYSTEM TESTING

5.1 Test Case Description:

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later during maintenance also.

Psychology of Testing:

The aim of testing is often to demonstrate that a program works by showing that it has no errors. The basic purpose of testing phase is to detect the errors that may be present in the program. Hence one should not start testing with the intent of showing that a program works, but the intent should be to show that a program doesn't work. Testing is the process of executing a program with the intent of finding errors.

Testing Objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

- Testing is a process of executing a program with the intent of finding an error.
- A successful test is one that uncovers a yet undiscovered error.
- A good test case is one that has a high probability of finding error, if it exists.
- The tests are inadequate to detect possibly present errors.
- The software conforms to the quality and reliable standards.

Levels of Testing:

In order to uncover the errors, present in different phases we have the concept of levels of testing.

System Testing:

The philosophy behind testing is to find errors. Test cases are devised. A strategy employed for system testing is code testing.

Code Testing:

This strategy examines the logic of the program. To follow this method, we developed some test data that resulted in executing every instruction in the program and module i.e. every path is tested. Systems are not designed as entire nor are they tested as single systems. To ensure that the coding is perfect two types of testing is performed or for that matter is performed or that matter is performed or for that matter is performed on all systems.

5.2 Types of Testing

5.2.1 Unit Testing:

Unit testing focuses verification effort on the smallest unit of software i.e. the module. Using the detailed design and the process specifications testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the start of the integration testing begins.

5.2.2 Link Testing:

Link testing does not test software but rather the integration of each module in system. The primary concern is the compatibility of each module. The Programmer tests where modules are designed with different parameters, length, type etc.

5.2.3 Integration Testing:

After the unit testing, we must perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. This testing activity can be considered as testing the design and hence the emphasis on testing module interactions.

5.2.4 System Testing:

Here the entire software system is tested. The reference document for this process is the requirements document, and the goals to see if software meets its requirements.

5.2.5 Acceptance Testing:

Acceptance Test is performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here is focused on external behavior of the system; the internal logic of program is not emphasized.

5.2.6 White Box Testing:

This is a unit testing method where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors. I tested step wise every piece of code, taking care that every statement in the code is executed at least once. The white box testing is also called Glass Box Testing.

5.2.7 Black Box Testing:

This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level. Here the module will be treated as a block box that will take some input and generate output. Output for a given set of input combinations are forwarded to other modules.

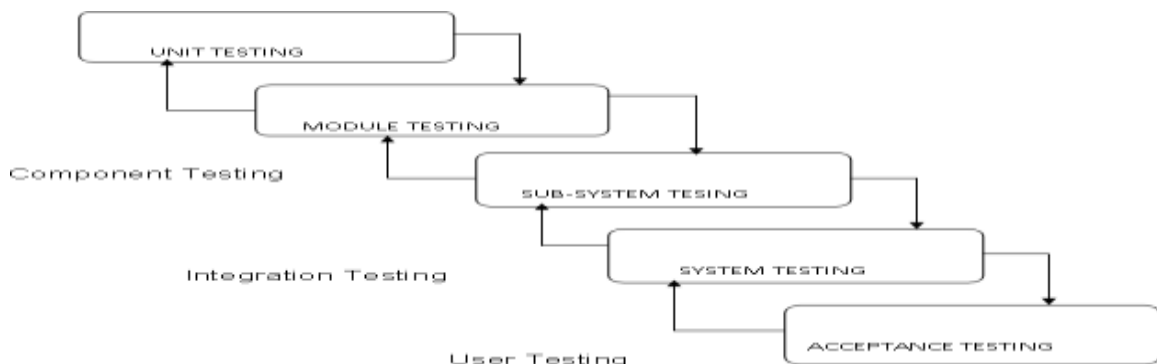


Fig 5.2 Testing Cycle

5.3 Test Cases:

Test Case Name	Test Case Description	Test Steps			Test Case Status
		Step	Expected	Actual	
Registration	New user must enter all details to register	Enter letters in mobile no	Wrong format for mobile.	Wrong format for mobile.	Pass
Sign in	User must enter login details	Enter correct username and wrong password	Username or password incorrect.	Username or password incorrect.	Pass
Sign in	User must enter login details	Enter wrong username and wrong password	Username or password incorrect.	Username or password incorrect.	Pass
Sign in	User must enter login details	Enter wrong username and correct password	Username or password incorrect.	Username or password incorrect.	Pass
Create table	User must enter table name and its columns	Give number at start of the table name	Can't create table	Can't create table	Pass
Insert values into table	Select the table to insert	Enter the wrong datatype values	Values not inserted	Values not inserted	Pass
Update the table	Select the table to update	Select a column and enter the wrong datatype values	Table can't be updated	Table can't be updated	Pass
Registration	New user must enter all details to register	Enter existing username	User already exist	User already exist	Pass
Insert values into table	Select the table to insert	Enter the NULL value in primary key	Values not inserted	Values not inserted	Pass

Display table contents	Select a table to display	Select no columns to display	Table contents can't be displayed	Table contents can't be displayed	Pass
Drop a table	Select a table	Select drop	Table dropped	Table dropped	Pass
Delete a table	Select a table to del-ete	Select delete	Table deleted	Table deleted	Pass
Update the table	Select the table to update	Select a column and change the values	Table updated	Table updated	Pass

CHAPTER-6

CONCLUSION AND FUTURE ENHANCEMENT

This project is very useful to those who want to perform basic operations within a database very easily without taking much time and user can also utilize the services of multiple databases using a single interface only which is user-friendly.

In this project we can just perform the basic queries regarding the database. As a future enhancement this project can be further extended to perform various operations that can be performed in a database. Within this project we have just used two different databases that can be accessed from an interface, this can be further extended for multiple databases more than two.

BIBLIOGRAPHY

References for the Project Development Were Taken From the following Books and Web Sites.

Oracle

- PL/SQL Programming by Scott Urman.
- SQL complete reference by Livion.

JAVA Technologies

- JAVA Complete Reference, Herbert Schildt.
- Java Script Programming by Yehuda Shiran.
- J2EE Professional by Shadab Siddiqui.
- JAVA Server pages by Nick Todd.

HTML

- HTML Black Book by Holzner.

JDBC

- Java Database Programming with JDBC by Patel moss.

Software Engineering

- Software Engineering by Roger Pressman.