CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE

We propose DQF, a novel database query form interface, which is able to dynamically generate query forms.

1.2 OVERALL DESCRIPTION

Query form is one of the most widely used user interfaces for querying databases. Traditional query forms are designed and pre-defined by developers or DBA in various information management systems. With the rapid development of web information and scientific databases, modern databases become very large and complex. Therefore, it is difficult to design a set of static query forms to satisfy various ad-hoc database queries on those complex databases. In this paper, we propose a Dynamic Query Form system: DQF, a query interface which is capable of dynamically generating query forms for users.

Different from traditional document retrieval, users in database retrieval are often willing to perform many rounds of actions before identifying the final candidates. The essence of DQF is to capture user interests during user interactions and to adapt the query form iteratively. Each iteration consists of two types of user interactions: Query Form Enrichment and Query Execution. The basic query form is then enriched iteratively via the interactions between the user and our system until the user is satisfied with the query results. In this paper, we mainly study the ranking of query form components and the dynamic generation of query forms.

CHAPTER 2

SYSTEM STUDY

2.1 EXISTING SYSTEM:

Traditional query forms are designed and pre-defined by developers or DBA in various information management systems. With the rapid development of web information and scientific databases, modern databases become very large and complex. Therefore, it is difficult to design a set of static query forms to satisfy various ad-hoc database queries on those complex databases.

2.2 PROPOSED SYSTEM

We propose a dynamic query form system which generates the query forms according to the user's desire at run time. The system provides a solution for the query interface in large and complex databases. This paper proposes DQF, a novel database query form interface, which is able to dynamically generate query forms. The essence of DQF is to capture a user's preference and rank query form components, assisting him/her to make decisions. The generation of a query form is an iterative process and is guided by the user. At each iteration, the system automatically generates ranking lists of form components and the user then adds the desired form components into the query form. The ranking of form components is based on the captured user preference. A user can also fill the query form and submit queries to view the query result at each iteration. In this way, a query form could be dynamically refined till the user satisfies with the query results.

2.3 FEASIBILITY ANALYSIS

2.3.1 Feasibility Study

Feasibility study is the test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of recourses.

It focuses on the evaluation of existing system and procedures analysis of alternative candidate system cost estimates. Feasibility analysis was done to determine whether the system would be feasible. The development of a computer based system or a product is more likely plagued by resources and delivery dates. Feasibility study helps the analyst to decide whether or not to proceed, amend, postpone or cancel the project, particularly important when the project is large, complex and costly.

Once the analysis of the user requirement is complement, the system has to check for the compatibility and feasibility of the software package that is aimed at. An important outcome of the preliminary investigation is the determination that the system requested is feasible.

2.3.2 Technical Feasibility:

The technology used can be developed with the current equipments and has the technical capacity to hold the data required by the new system. Technical feasibility on the existing system and to what extend it can support the proposed addition.We can add new modules easily without affecting the Core Program. Most of parts are running in the server using the concept of stored procedures.

2.3.3 Operational Feasibility:

This proposed system can easily implemented, as this is based on JSP coding (JAVA) & HTML .The database created is with MySql server which is more secure and easy to handle. The resources that are required to implement/install these are available. The personal of the organization already has enough exposure to computers. So the project is operationally feasible.

CHAPTER 3

SYSTEM REQUIRMENTS

3.1 SOFTWARE REQUIREMENTS:

- Windows OS
- JDK 1.7
- NetBeans 7.2
- SQL YOG enterprise 7.02

3.2 HARDWARE REQUIREMENTS:

Main Processor	:	> 2GHz
Ram	:	1 GB
Hard Disk	:	80GB

CHAPTER 4

LANGUAGE DESCRIPTION

4.1 JAVA

It is a Platform Independent. Java is an object-oriented programming language developed initially by James Gosling and colleagues at Sun Microsystems. The language, initially called Oak (named after the oak trees outside Gosling's office), was intended to replace C++, although the feature set better resembles that of Objective C.

4.1.1 WORKING OF JAVA

For those who are new to object-oriented programming, the concept of a class will be new to you. Simplistically, a class is the definition for a segment of code that can contain both data (called attributes) and functions (called methods).

When the interpreter executes a class, it looks for a particular method by the name of main, which will sound familiar to C programmers. The main method is passed as a parameter an array of strings (similar to the argv[] of C), and is declared as a static method.

To output text from the program, we execute the println method of System.out, which is java’s output stream. UNIX users will appreciate the theory behind such a stream, as it is actually standard output. For those who are instead used to the Wintel platform, it will write the string passed to it to the user’s program.

Java consists of two things :

- Programming language
- Platform.

4.1.2 THE JAVA PROGRAMMING LANGUAGE

Java is a high-level programming language that is all of the following:

- Simple
    - Object-oriented
    - Distributed
    - Interpreted
    - Robust
    - Secure
    - Architecture-neutral
    - Portable
    - High-performance
    - Multithreaded
    - Dynamic

The code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Java is unusual in that each Java program is both co implied and interpreted. With a compiler, you translate a Java program into an intermediate language called **Java byte codes** – the platform independent codes interpreted by the Java interpreter. With an interpreter, each Java byte code instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. This figure illustrates how it works :

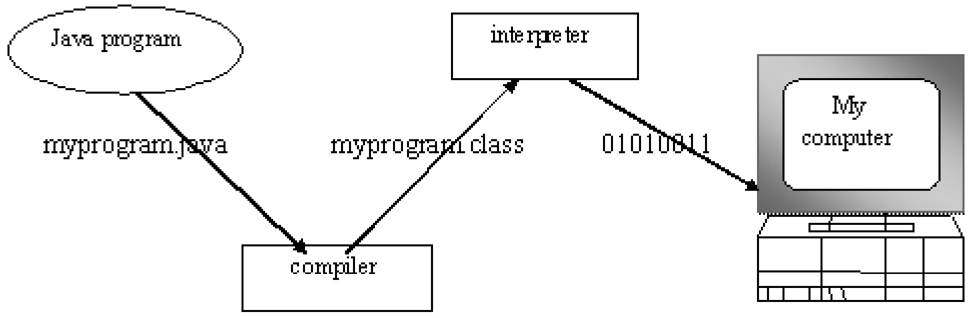


Figure 4.1
Working of java programming language

4.1.3 THE JAVA PLATFORM

A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other, hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components :

- The Java Virtual Machine (JVM)
- The Java Application Programming Interface (Java API)

You’ve already been introduced to the JVM. It’s the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries **(packages)** of related components. The following figure depicts a Java program, such as an application or applet, that’s running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.

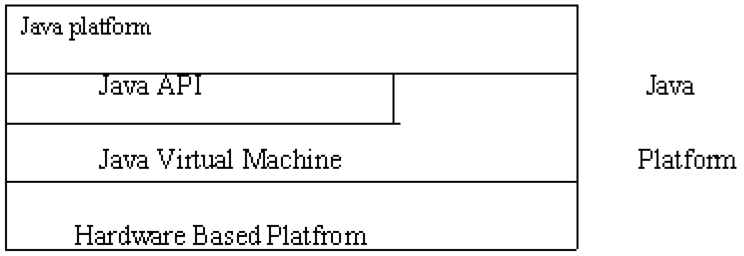


Figure 4.2 Java platform

4.2 NETBEANS

Net Beans refers to both a platform framework for java desktop applications and an integrated development environment(IDE) for developing with java, java script, python, groovy, C,C++, Scala, Clojure and others.

4.2.1 NetBeans platforms

The NetBeans Platform is a reuable framework for simplifying the development of java swing desktop applications, allowing developers to focus on the logic specification to their application. Among the features of the platform are:

User interface management (e.g. menus and toolbars)
User settings management
Storage management (saving and loading any kind of data)
Window management
Wizard framework (supports step-by-step dialogs)
NetBeans visual library
Integrated development tools

4.2.2 NetBeans IDE

NetBeans IDE is an open-source integrated development environment. NetBeans IDE supports development of all java application types among other features are an anti-based project system, Maven supports , refactoring, versioncontrol (supporting CVS, Subversion, Mercurial and Clear case).

Modularity: All the functions of the IDE are provided by modules. Each module provides a well defined function, such as support for the java language, editing , or suppoet for the CVS versioning system, and SVN. NetBeans contains all the modules needed for java development.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

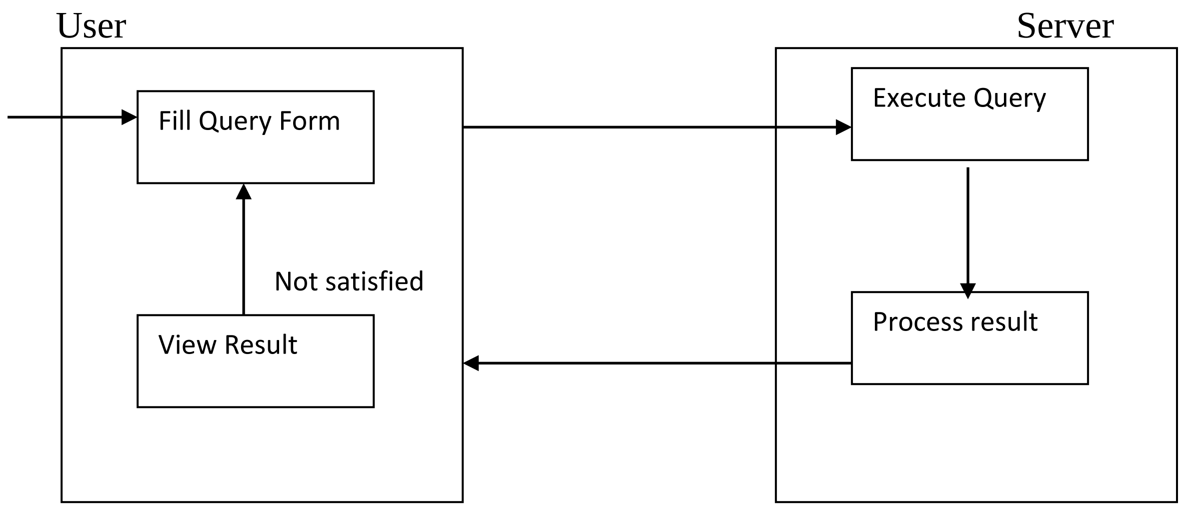
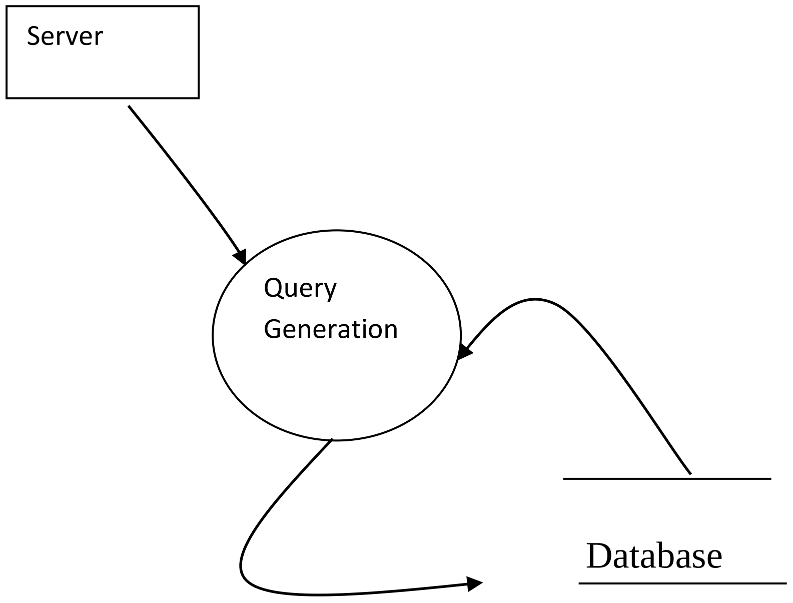


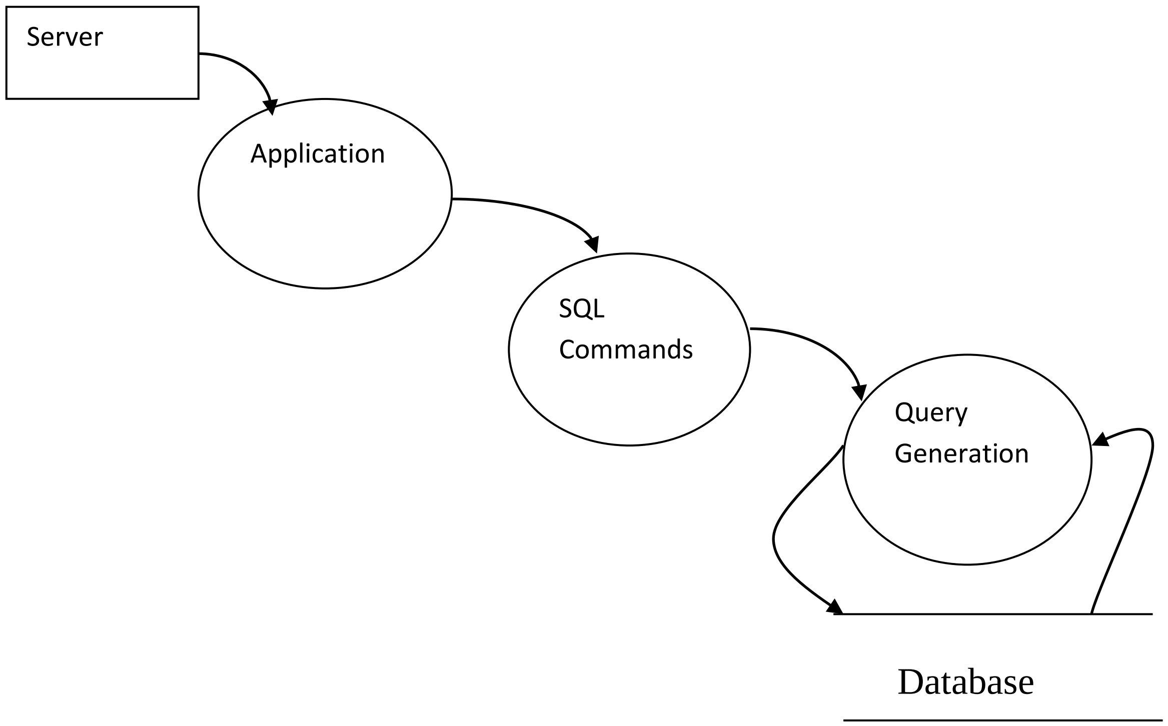
Figure 5.1
System Architecture

5.2 DATAFLOW DIAGRAMS

5.2.1 DFD-0



5.2.2 DFD-1



5.2.3 DFD-2

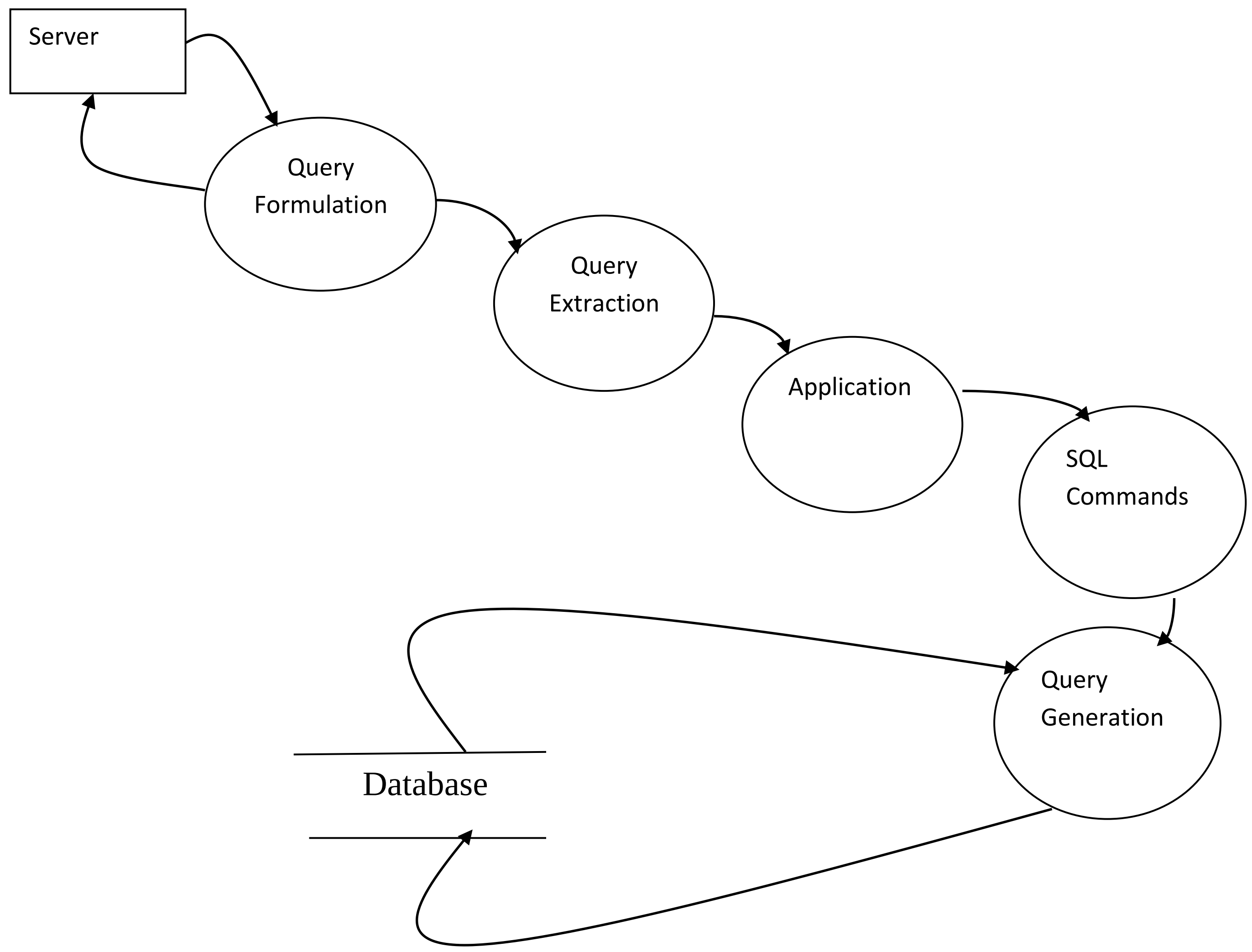


Figure 5.1,5.2 and 5.3 Dataflow Diagram

5.3 USECASE

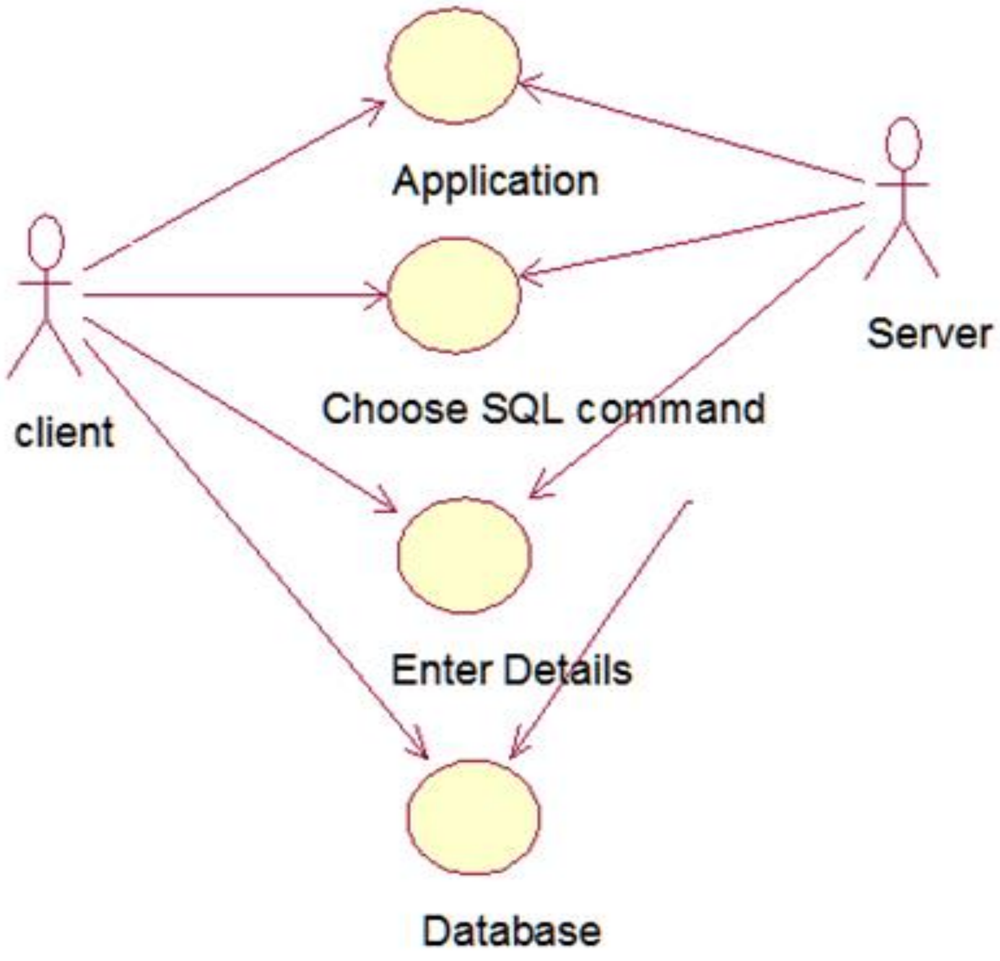


Figure 5.4 Usecase Diagram

5.4 CLASS DIAGRAM

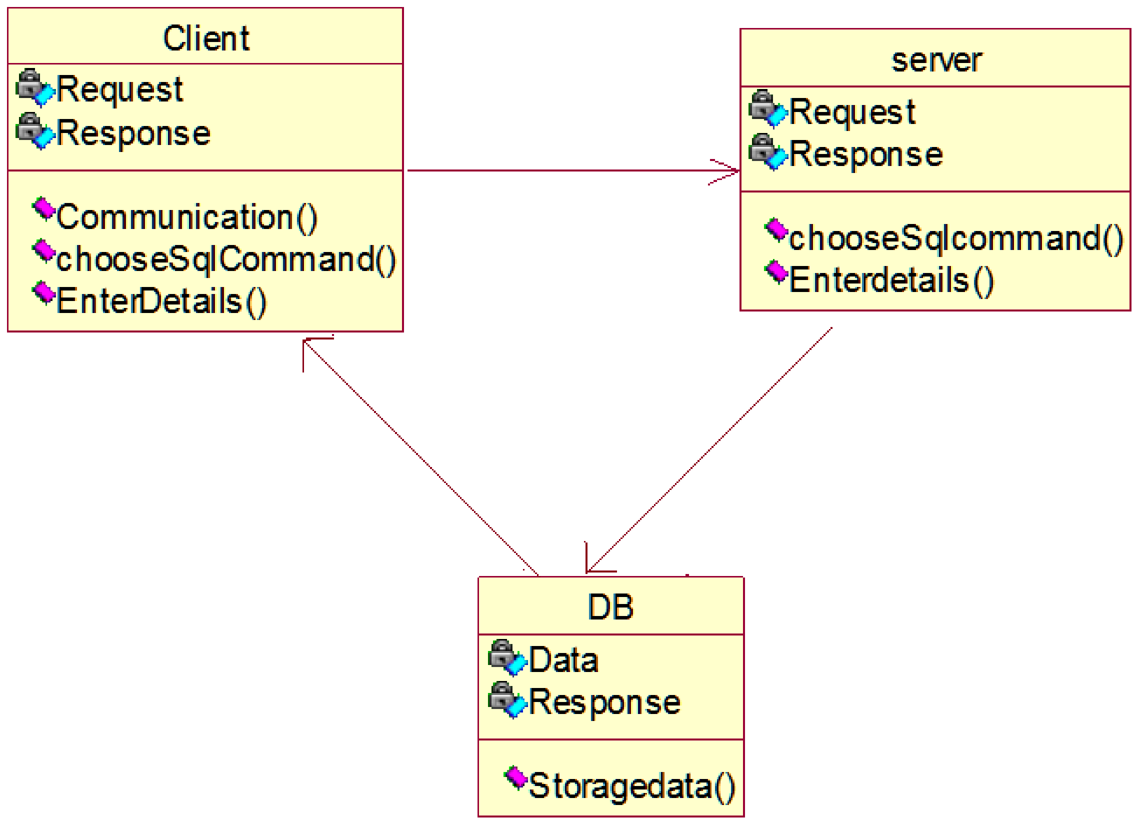


Figure 5.5 Class Diagram

5.5 SEQUENCE

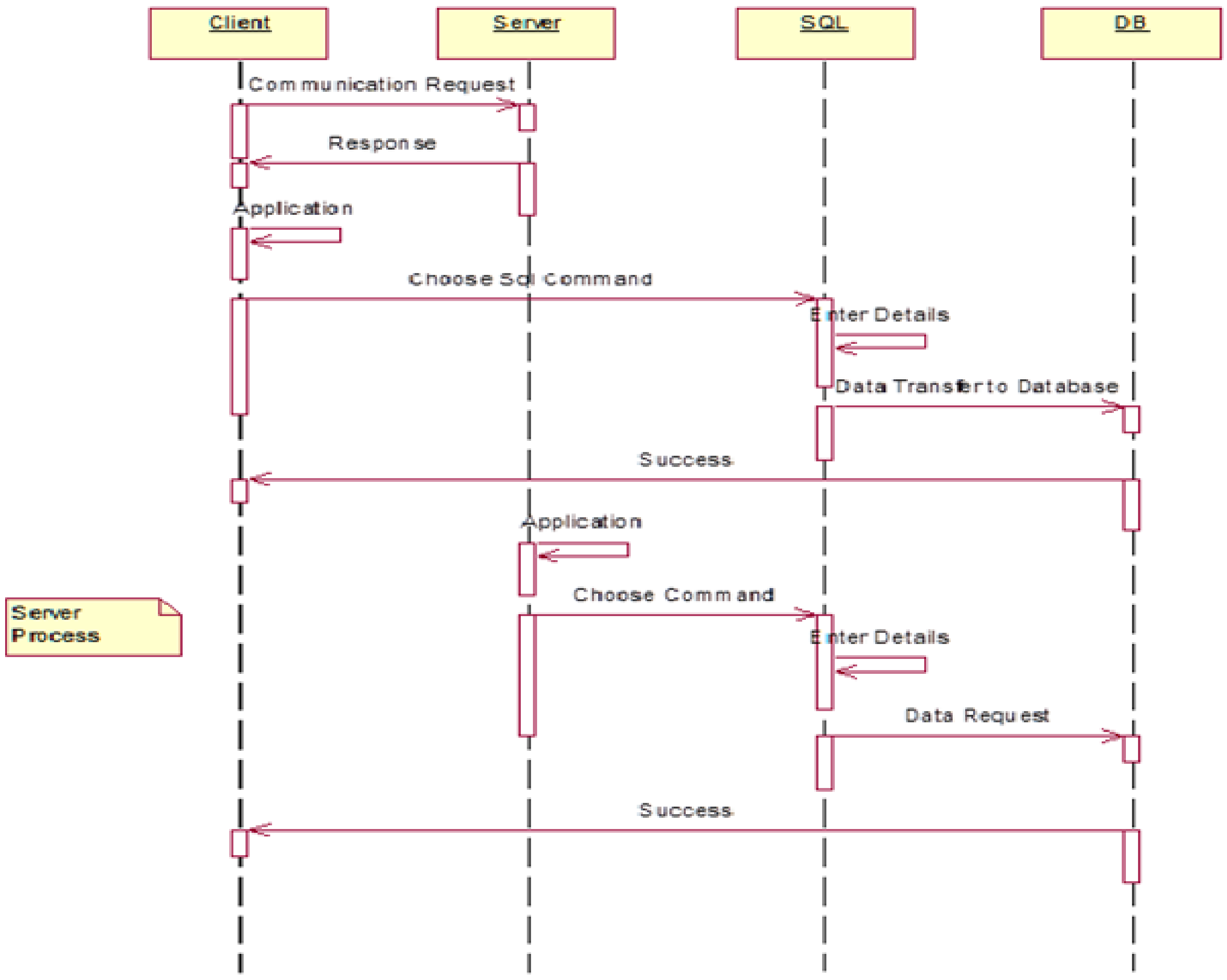


Figure 5.6 Sequence Diagram

5.6 COLLABORATION

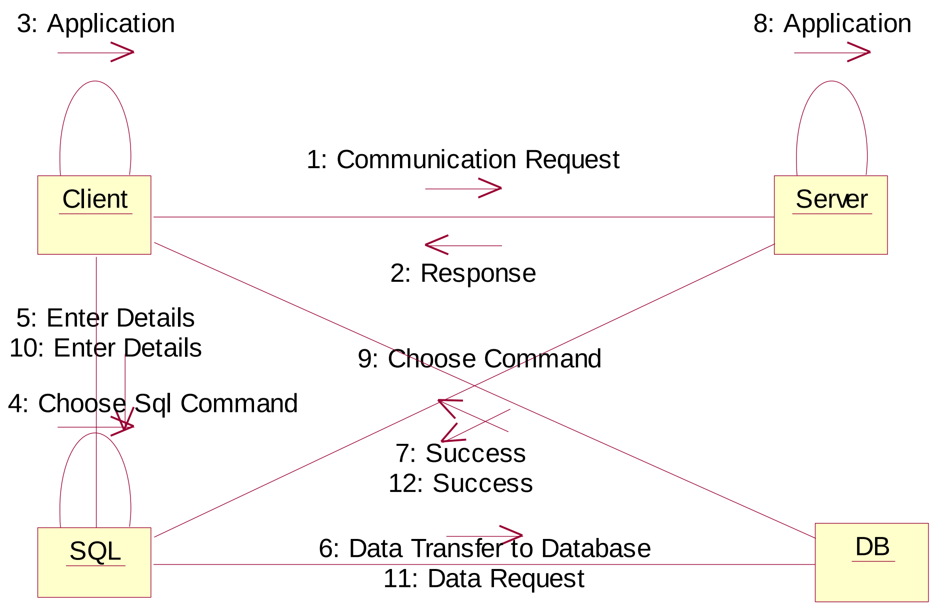


Figure 5.7 Colaboration Diagram

5.7 ACTIVITY

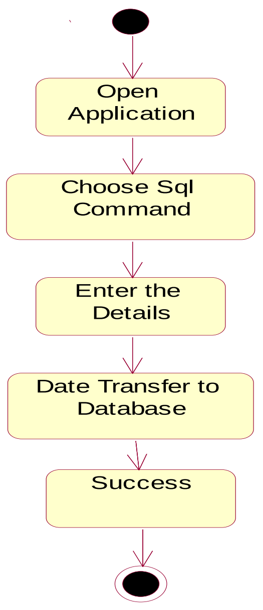


Figure 5.8 Activity Diagram

5.8 SAMPLE CODING

```
package dqf;

public class QueryForm implements ActionListener, ItemListener

 {

   public static final String type1Class="sun.jdbc.odbc.JdbcOdbcDriver"; //database
connectivity

   public static final String type1Dsn="Jdbc:Odbc:";

   public static final String oracle10gClass="oracle.jdbc.driver.OracleDriver";

   public static final String oracle10gDsn="jdbc:oracle:thin:@localhost:1521:Xe";

   public static final String mySqlClass="com.mysql.jdbc.Driver";

   public static final String mySqlDsn="jdbc:mysql://localhost:3306/";

   public static String dsnText="";

   public static String selectTableName="";

   public static String className="";

   public static String dsnName="";

   public static Connection connection;

   public static ResultSet resultSet;

   public static ResultSetMetaData rstMetaData;

   public static Statement statement;

   public static DatabaseMetaData dbmd;

    public static void main(String args[])

    { QueryForm qb=new QueryForm();}

  public void actionPerformed(ActionEvent e)

   {
```

```
 try
 {
 if(e.getSource()==connect)
 {      connectDialog();     }
else if(e.getSource()==tableField)
 {      addAvailColumnList();    }
  else if(e.getSource()==queryButton)
{   queryArea.setText(getQuery());}
 else if(e.getSource()==viewupdate)
{
    queryArea.setText(getupdateQuery());
    JPasswordField jf=new JPasswordField();
 String pass=JOptionPane.showInputDialog(jf,"Enter DataBase Server Password to
Update");
    if(pass.equals("password"))
    {
    statement.executeUpdate(queryArea.getText());
     queryButton.setEnabled(true);
    goButton.setEnabled(true);
    deleteButton.setEnabled(true);
    clearAll();
    }
    else
    {JOptionPane.showMessageDialog(null,"Invalid Password");}
}
```

```
   else if(e.getSource()==deleteButton)
    {
        queryButton.setEnabled(false);
        goButton.setEnabled(false);
        updateButton.setEnabled(false);
        queryArea.setText(getdeleteQuery());
        JPasswordField jf=new JPasswordField();
       String  pass=JOptionPane.showInputDialog(jf,"Enter DataBase Server Password to
Delete");
        if(pass.equals("password"))
        {
        statement.executeUpdate(queryArea.getText());
         queryButton.setEnabled(true);
        goButton.setEnabled(true);
        updateButton.setEnabled(true);
        clearAll();
        }
        else
        { JOptionPane.showMessageDialog(null,"Invalid Password");}
    }

    else if(e.getSource()==updateButton)
    {   updialog();
        viewupdate.setEnabled(true);
        queryButton.setEnabled(false);
```

```
      goButton.setEnabled(false);
      deleteButton.setEnabled(false);
      queryArea.setText(getupdateQuery());}
   else if(e.getSource()==orderby)
  {   Orderdialog();
      viewupdate.setEnabled(true);
      queryButton.setEnabled(true);
      goButton.setEnabled(true);
      deleteButton.setEnabled(true);
      queryArea.setText(getOrderQuery());}
 else if(e.getSource()==addOrder)
  {temp=new Vector<String>();
    temp.add((String)orderCombo.getSelectedItem());}
else if(e.getSource()==okOrder)
 { orderVector=new Vector<String>();
   Enumeration em=temp.elements();
   while(em.hasMoreElements())
    { orderVector.add((String)em.nextElement());}
    orderVector.add((String)orderCombo.getSelectedItem());
   temp.clear();
  dialog.setVisible(false);
 }
}
     if(e.getSource()==okConnect)
    {
     try{
           String dbname=dbfield.getSelectedItem().toString();
            int cout=0;
             className=mySqlClass;
             dsnName=mySqlDsn+dbname;
              Class.forName(className);
              //connection=DriverManager.getConnection(dsnName,"root","password");
              connection=DriverManager.getConnection(dsnName,userName.getText(),new
String(passwordField.getPassword()));
          statement=connection.createStatement();
          resultSet=statement.executeQuery("select atime from rankdb.totaldb where
dbname='"+dbname+"'");
          if(resultSet.next())
          {cout=resultSet.getInt("atime");}
          statement.executeUpdate("update rankdb.totaldb set atime='"+(cout+1)+"' where
dbname='"+dbname+"'");
        setConnection(className,dsnName);
         dialog.setVisible(false);  }
    catch(Exception ex)
     { int col=ex.toString().indexOf(':');
       String tp=ex.toString().substring(0,col);
        String msg=ex.toString().substring(col+1)+"from Data Source";
       System.out.println(ex);
       JOptionPane.showMessageDialog(frame,msg,tp,JOptionPane.ERROR_MESSAGE);
     }} }
     catch(Exception exx)
     {  System.out.println(exx.getMessage());}
  }
    public void itemStateChanged(ItemEvent e)
    { JCheckBox jc=(JCheckBox)e.getItem();}
   public String getupdateQuery()
    {  String query="update ";
     query+=tableField.getSelectedItem();
     query+=" set "+columnField1.getSelectedItem()+"="+textFieldValue1.getText();
     if(!fieldVector.isEmpty())
        {  query+=" where ";
          for(int i=0;i<fieldVector.size();i++)
           {
             query+=" "+(String)relationVector.get(i)+" "+fieldVector.get(i)+"
"+operatorVector.get(i);
             int
t=(Integer)availColumnTypeList.get(availColumnList.indexOf(fieldVector.get(i)));
             if(t==Types.VARCHAR || t==Types.DATE || t==Types.CHAR)
               { query+="'"+valuesVector.get(i)+"'";}
             else { query+=valuesVector.get(i);}
}}
     return query;
    }


  public String getdeleteQuery()
    { String query="delete ";
```

```
    query+="from "+tableField.getSelectedItem();
    if(!fieldVector.isEmpty())
      {  query+=" where ";
         for(int i=0;i<fieldVector.size();i++)
          {  query+=" "+(String)relationVector.get(i)+" "+fieldVector.get(i)+"
"+operatorVector.get(i);
            int
t=(Integer)availColumnTypeList.get(availColumnList.indexOf(fieldVector.get(i)));
            if(t==Types.VARCHAR || t==Types.DATE || t==Types.CHAR)
              { query+="'"+valuesVector.get(i)+"'"; }
            else {query+=valuesVector.get(i);}
} }
    else
    {
        JOptionPane.showMessageDialog(null, "You have to set where field to use Delete
operation");
    }
    return query;
  }
  public void setConnection(String className,String dsnName)
   {
    try
     {
       tableVector=new Vector<String>();
       tableHashSet=new LinkedHashSet<String>();
      // Class.forName(className);
```

```
        // connection=DriverManager.getConnection(dsnName);
         dbmd=connection.getMetaData();
         resultSet=dbmd.getTables(null,null,null,new String[]{"Table"});
          while(resultSet.next())
           { String str=resultSet.getString(3);
             tableVector.add(str);
             tableHashSet.add(str);
           }
//add table in Table ComboBox
         Iterator<String> it=tableHashSet.iterator();
         while(it.hasNext())
          {  tableField.addItem(it.next());}
        }
      catch(Exception e)
        {
         JOptionPane jop=new JOptionPane();
         jop.showMessageDialog(frame,e);
         System.out.println(e);
        }
      }
    // Connect Dialog Box
  public void connectDialog()throws Exception
   {
      panel.add(new JLabel("<html><big>Select Data
Source...</big></html>"),BorderLayout.NORTH);
```

```
      panel.add(new JScrollPane(centerPanel),BorderLayout.CENTER);

      panel.add(new JScrollPane(southPanel),BorderLayout.SOUTH);

      dialog.add(panel);

      dialog.setVisible(true);

      dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);

      Class.forName("com.mysql.jdbc.Driver");
connection=DriverManager.getConnection("jdbc:mysql://localhost:3306/","root","password");

      Connection
connection1=DriverManager.getConnection("jdbc:mysql://localhost:3306/","root","password"
);

      Statement stt=connection1.createStatement();

      ResultSet rss=stt.executeQuery("select * from rankdb.totaldb");

      resultSet=connection.getMetaData().getCatalogs();

      if(rss.next())

      {while(resultSet.next())

    {

        String tabs=resultSet.getString("TABLE_CAT");

        stt.executeUpdate("insert into rankdb.totaldb values('"+tabs+"','1')");

    }

      }

      rss=stt.executeQuery("select * from rankdb.totaldb order by atime desc"); //RANKING

      while(rss.next())

      {

         //System.out.println("...HAVING..");

        dbfield.addItem(rss.getString("dbname"));

      }  }
```

```
  public String getQuery()
   {
       String query="Select ";
       if(vList2.isEmpty())
         query+=" * ";
       else
        {
          for(int i=0;i<vList2.size();i++)
           {  if(i>0)
               query+=" ,";
             query+=vList2.get(i);}
        }
      query+=" from "+tableField.getSelectedItem();
      if(!fieldVector.isEmpty())
        {  query+=" where ";
           for(int i=0;i<fieldVector.size();i++)
            {
              query+=" "+(String)relationVector.get(i)+" "+fieldVector.get(i)+"
"+operatorVector.get(i);
              int
t=(Integer)availColumnTypeList.get(availColumnList.indexOf(fieldVector.get(i)));
              if(t==Types.VARCHAR || t==Types.DATE || t==Types.CHAR)
                {query+="'"+valuesVector.get(i)+"'";}
              else
              { query+=valuesVector.get(i);}
            } }
      return query;
   }
  public String getOrderQuery()
   {  String query="Select ";
       if(vList2.isEmpty())
         query+=" * ";
       else
        {  for(int i=0;i<vList2.size();i++)
           {  if(i>0)
               query+=" ,";
             query+=vList2.get(i);
           }  }
      query+=" from "+tableField.getSelectedItem();
      if(!fieldVector.isEmpty())
        { query+=" where ";
           for(int i=0;i<fieldVector.size();i++)
            {              query+=" "+(String)relationVector.get(i)+" "+fieldVector.get(i)+"
"+operatorVector.get(i);
              int
t=(Integer)availColumnTypeList.get(availColumnList.indexOf(fieldVector.get(i)));
              if(t==Types.VARCHAR || t==Types.DATE || t==Types.CHAR)
                { query+="'"+valuesVector.get(i)+"'";}
              else
             { query+=valuesVector.get(i); }
  } }
      query+=" order by "+columnField2.getSelectedItem()+"
"+operatorField2.getSelectedItem();
      return query;
   }
   public void showTable()
   { int p=0,k=0;
     String str[][];
     String strr[];
      try
       {    dbmd=connection.getMetaData();
             statement=connection.createStatement();
             Statement st1=connection.createStatement();
             resultSet=dbmd.getTables(null,null,null,new String[]{"Table"});
             resultSet=statement.executeQuery(queryArea.getText());
             ResultSet rs1=st1.executeQuery(queryArea.getText());
             rstMetaData=resultSet.getMetaData();
             int count=rstMetaData.getColumnCount();
             strr=new String[count];
             for(int i=0;i<count;i++)
              {  strr[i]=rstMetaData.getColumnName(i+1); }
          while(resultSet.next())
           {   k++; }
            str=new String[k][count];
```

```
            int j=0;
            while(rs1.next())
             {  for(int i=0;i<count;i++)
                { str[j][i]=rs1.getString(i+1);
                }  j++;
   }
           // Table Box
              tableDialog=new JDialog(frame,true);
             tableDialog.setSize(450,350);
             tableDialog.setTitle("Table");
             JPanel tablePanel=new JPanel(new BorderLayout());
             try { table=new JTable(str,strr);}
             catch(Exception et)
              { JOptionPane jop=new JOptionPane();
                jop.showMessageDialog(frame,et)}
             int v=ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;
             int h=ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;
             JScrollPane jsp=new JScrollPane(table,v,h);
             tablePanel.add(jsp);
             JPanel southPanel=new JPanel(new BorderLayout());
             closeButton=new JButton("Close");
             closeButton.addActionListener(this);
             tablePanel.add(closeButton,BorderLayout.SOUTH);
             tableDialog.add(tablePanel);
             tableDialog.setVisible(true);
```

```
        }
      catch(Exception ex)
       {
          int col=ex.toString().indexOf(':');
          String tp=ex.toString().substring(0,col);
          String msg=ex.toString().substring(col+1)+"from Data Source";
          JOptionPane.showMessageDialog(frame,msg,tp,JOptionPane.ERROR_MESSAGE);
       }
   }
 }
```

CHAPTER 6

IMPLEMENTATION AND TESTING

6.1Modules

6.1Query Form

6.2Add Database

6.3Execute Query

6.4Reconstruct QF

6.1.1 **Query Form**

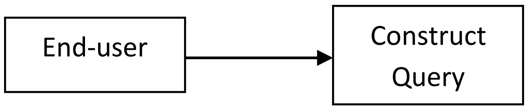


Figure 6.1 Module 1 – Query form

At present, query forms have been utilized in most real-world business or scientific information systems. Existing database clients and tools make great efforts to help developers design and generate the query forms, such as Microsoft Access and so on. They provide visual interfaces for developers to create or customize query forms. The problem of those tools is that, they are provided for the professional developers who are familiar with their databases, not for end-users. This Dynamic query forms allows end-users to customize the existing query form at run time.

6.1.2 Add Database

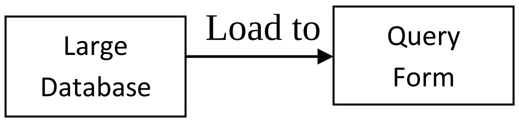


Figure 6.2 Module 2 - Add Database

Database contains the collection of dataset. These large databases are loaded to the Query form page. There are several types of Database drivers are available like Jdbc, Odbc and so on. Each database will be loaded to form and accessed with the help of these drivers. Novel user interfaces have been developed to assist the user to type the database queries based on the query workload, the data distribution and the database schema.

6.1.3 Execute Query

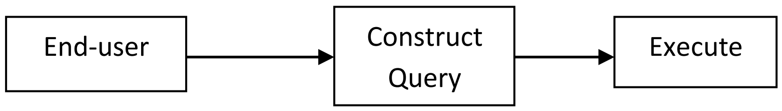


Figure 6.3 Module 3 – Execute Query

It first finds a set of data attributes, then that are retrieved based on the database schema and data instances. Then, the query forms are generated based on the selected attributes. The query forms are then generated based on those representative queries. If the database schema is large and complex, user queries could be quite diverse. In that case, the end-user can regenerate the query form and can execute that as a new query.

6.1.4 Reconstruct QF

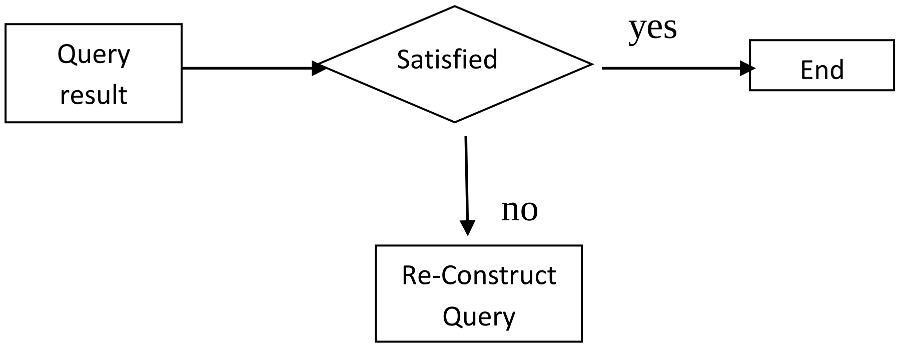


Figure no 6.4   Module – Reconstruct QueryForm

If the database schema is large and complex, user queries could be quite diverse. In that case, the end-user can regenerate the query form and can execute that as a new query. A solution is that the user inputs several keywords to find relevant query forms from a large number of pre-generated query forms. It works well in the databases which have rich textual information in data tuples and schemas.

6.2 SOFTWARE TESTING

6.2.1 INTRODUCTION

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error.

6.3 TYPES OF TESTING

6.3.1 WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function.

Basis Path Setting

- Flow Graph Notation
- Cyclometric Complexity
- Deriving Test Cases

6.3.2 BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that "all gears mesh", that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph Based Test Methods
- Equivalence Partitioning
- Boundary Value Analysis
- Comparison Testing

6.3.3 SECURITY TESTING

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

6.3.4 UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing **.**

6.3.5 STRESS TESTING

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

6.3.6 VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

6.3.7 INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION

Dynamic query form generation approach which helps users dynamically generate query forms. The key idea is to use a probabilistic model to rank form components based on user preferences. We capture user preference using both historical queries and run-time feedback such as click through. Experimental results show that the dynamic approach often leads to higher success rate and simpler query forms compared with a static approach. The ranking of form components also makes it easier for users to customize query forms.

FUTURE ENHANCEMENT

The repeating cost of preparing a dynamic statement can make the performance worse than that of static SQL statements. However, if you execute the same SQL statement often, we can use the dynamic statement cache to decrease the number of times that those dynamic statements must be prepared.

REFERENCES

[1] Dynamic Query Forms for Database Queries, Liang Tang, Tao Li ,Yexi Jiang and Zhiyuyan Chen

[2] DBPedia. http://DBPedia.org.

[3] EasyQuery. http://devtools.korzh.com/eq/dotnet/.

[4] Freebase. http://www.freebase.com.

[5] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In Proceedings of VLDB, pages 81–92, Berlin, Germany, September 2003.

[6] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diver-sifying search results. In Proceedings of WSDM, pages 5–14, Barcelona, Spain, February 2009.

[7] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In CIDR, 2003.