DYNAMIC QUERY FORMS FOR DATABASE QUERIES

Project report submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA

In Partial fulfillment of the requirements

For the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

DURING 2012-2016

SUBMITTED BY

B. Vishnu Priya	**(12KQ1A0511)**
G. Anil Kumar	**(12KQ1A0527)**
K. Adi Narayana Reddy	**(12KQ1A0519)**
D. Dhana Lakshmi	**(12KQ1A0538)**

Under the esteemed guidance of

Mr. G. Srinivasa Rao M .Tech.,(Ph.D)

Associate Professor, CSE

Department of COMPUTER SCIENCE & ENGINEERING

PACE INSTITUTE OF TECHNOLOGY & SCIENCES

An ISO 9001-2008 Certified Institution

(Affiliated to J.N.T. University, Kakinada

NH-5, Near Valluramma Temple, Ongole, Prakasam (D.t)

Department of COMPUTER SCIENCE & ENGINEERING

PACE INSTITUTE OF TECHNOLOGY & SCIENCES

An ISO 9001-2008 Certified Institution

(Affiliated to J.N.T. University, Kakinada)

NH-5, Near Valluramma Temple, Ongole, Prakasam (D.t)

CERTIFICATE

This is to certify that the project work entitled **"DYNAMIC QUERY FORMS FOR DATABASE QUERIES"** is the result of the bonafide work done by

B. Vishnu Priya	**(12KQ1A0511)**
G. Anil Kumar	**(12KQ1A0527)**
K. Adi Narayana Reddy	**(12KQ1A0519)**
D. Dhana Lakshmi	**(12KQ1A0538)**

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY,

KAKINADA

In partial fulfillment of the requirements

For the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN **COMPUTER SCIENCE & ENGINEERING**

DURING 2012-2016

INTERNAL GUIDE	**HOD**
Mr.G.Srinivasa Rao	**Mr. P.Anil Kumar**
Asst. Professor	Assoc. Professor &HOD

External examiner

DECLARATION

We hereby declare that the project work titled "**Dynamic Query Forms for Database Queries**" done under the guidance of **Mr. G. Srinivasa Rao. , Associate Professor** of CSE Department, Pace Institute of Technology & Sciences, Valluru, being submitted to "The Department of Computer Science & Engineering", Pace Institute of Technology & Sciences, Valluru of our own and has not been submitted to any other University or Educational institution for any degree or diploma.

B. Vishnu Priya (12KQ1A0511)

G. Anil Kumar (12KQ1A0527)

K. Adi Narayana Reddy (12KQ1A0519)

D. Dhana Lakshmi (12KQ1A0538)

ACKNOWLEDGEMENT

We are thankful to our esteemed guide **Mr. G. Srinivasa Rao. , Associate Professor** who has spared his valuable time and append novel ideas to guide us in limelight. We are indebted to his without whom we would not have culminated to pinnacle of the project.

We are thankful to HOD of CSE department **Mr. P.Anil Kumar**, **Associate Professor** for his continuous monitoring and motivation which helps us to concentrate on the project.

We are thankful to our beloved Principal **Dr. C.V. Subbarao** for providing us appropriate environment required for the project to complete.

We are also thankful to the whole CSE DEPARTMENT for their encouragement and cooperation for the successful completion of the project. We are also thankful to all the friends who are directly or indirectly helped us in the completion of the project with flying colors.

We are express my heartfelt gratitude to the management for providing all the necessary support to complete this project successfully.

B. Vishnu Priya **(12KQ1A0511)**

G. Anil Kumar **(12KQ1A0527)**

K. Adi Narayana Reddy **(12KQ1A0519)**

D. Dhana Lakshmi **(12KQ1A0538)**

ABSTRACT

Modern scientific and web databases maintain large and heterogeneous data. These real-world database schemas contain over hundreds or even thousands of attributes and relations. Traditional predefined query forms are not able to satisfy various ad-hoc queries from users. 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 the user preference and rank query form components. The generation of the 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. The user can also fill the query form and submit queries to view the query result at each iteration. In this way, the query form could be dynamically refined until the user satisfies with the query results. We propose a metric for measuring the goodness of a query form. A probabilistic model is developed for estimating the goodness of a query form in DQF. Our experimental evaluation and user study demonstrate the effectiveness and efficiency of the system.

Contents

List of Figures

1. INTRODUCTION

1.1 MOTIVATION:

Query form is one of the most widely used user interfaces for querying databases. Traditional query forms are designed and predefined 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. In natural sciences, such as genomics and diseases, the databases have over hundreds of entities for chemical and biological data resources. Many web databases, such as Freebase and DBPedia, typically have thousands of structured web entities. Therefore, it is difficult to design a set of static query forms to satisfy various ad-hoc database queries on those complex databases. Many existing database management and development tools, such as Easy Query, Cold Fusion, SAP and Microsoft Access, provide several mechanisms to let users create customized queries on databases. However, the creation of customized queries totally depends on users' manual editing. If a user is not familiar with the database schema in advance, those hundreds or thousands of data attributes would confuse him/her.

1.2 OBJECTIVES OF PROJECT:

Modern scientific and web databases maintain large and heterogeneous data. These real-world database schemas contain over hundreds or even thousands of attributes and relations. Traditional predefined query forms are not able to satisfy various ad-hoc queries from users. 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 the user preference and rank query form components. The generation of the 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. We propose a metric for measuring the goodness of a query form.

1. 3 LIMITATIONS OF PROJECT:

In that case, even if we generate lots of query forms to let users find an appropriate and desired query form would be challenging. There are still user queries that cannot be satisfied by any one of query forms. In That Case Time Consuming is more.

1.4 PROBLEM DEFINITION:

In our system, we provide a ranked list of query form components for the user. Problem 1 is the formal statement of the ranking problem.

2. LITERATURE SURVEY

2.1 INTRODUCTION:

Query form is one of the most widely used user interfaces for querying databases. Traditional query forms are designed and predefined 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. In natural sciences, such as genomics and diseases, the databases have over hundreds of entities for chemical and biological data resources. Many web databases, such as Freebase and DBPedia, typically have thousands of structured web entities. Therefore, it is difficult to design a set of static query forms to satisfy various ad-hoc database queries on those complex databases. Many existing database management and development tools, such as Easy Query, Cold Fusion, SAP and Microsoft Access, provide several mechanisms to let users create customized queries on databases.

2.2 EXISTING SYSTEM:

Recently proposed automatic approaches to generate the database Query forms without user participation presented a data-driven method. It first finds a set of data attributes, which are most likely queried based on the database schema and data instances. Then, the query forms are generated based on the selected attributes. One problem of the aforementioned approaches is that, if the database Schema is large and complex, user queries could be quite diverse. In that case, even if we generate lots of query forms in advance, there are still user queries that cannot be satisfied by any one of query forms. Another problem is that, when we generate a large number of query forms, how to let users find an appropriate and desired query form would be challenging. A solution that combines keyword search with query form generation is proposed. It automatically generates a lot of query forms in advance. 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. However, it is not appropriate when the user does not have concrete keywords to describe the queries at the beginning, especially for the numeric attributes.

2.3 DISADVANTAGES OF EXISTING SYSTEM:

In that case, even if we generate lots of query forms to let users find an appropriate and desired query form would be challenging. There are still user queries that cannot be satisfied by any one of query forms. In That Case Time Consuming is more.

2.4 PROPOSED SYSTEM:

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 (i.e., refining query conditions) 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 following figure shows the work-flow of DQF. It starts with a Basic query form which contains very few primary attributes of the database. 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.

3. SYSTEM ANALYSIS

3.1 INTRODUCTION:

We implemented the dynamic query forms as a web based system using JDK 1.6 with Java Server Page. The dynamic web interface for the query forms used open-source java script library jQuery 1.4. We used MySQL 5.1.39 as the database engine. All experiments were run using a machine with Intel Core 2 CPU @2.83GHz, 3.5G main memory, and running on Windows XP SP2.

3.2 REQUIREMENT SPECIFICATION:

3.2.1 Hardware Requirements:

Processor	-	Pentium –III / Intel Core
Speed	-	1.1 GHz
RAM	-	256 MB (min)
Hard Disk	-	250 GB

3.2.2 Software Requirements:

Operating System	:	Windows
Technology	:	JAVA, JFC (Swing), J2me
Database	:	My SQL
Database Connectivity	:	JDBC
Web Server	:	Tomcat.

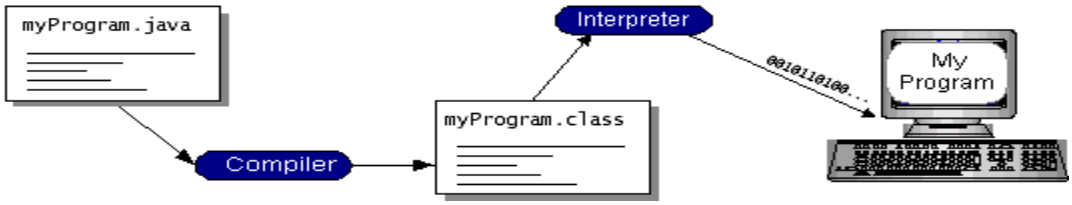
SOFTWARE DESCRIPTIONS:

Java Technology

Java technology is both a programming language and a platform. The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

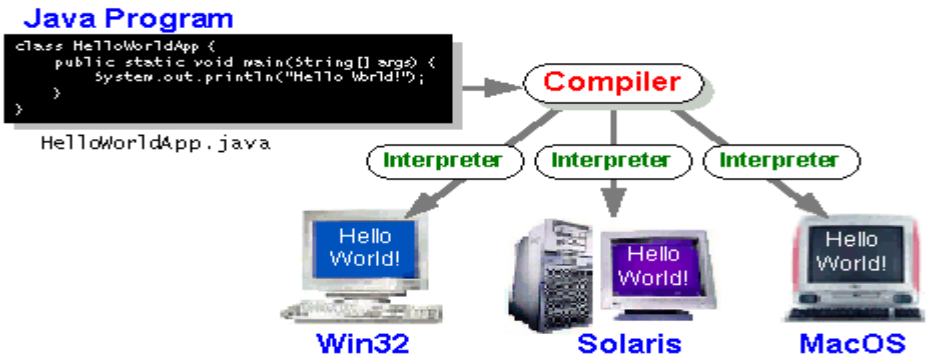
- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



3.1. Java Runtime Environment Diagram

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



3.2 Java Program compiler Diagram

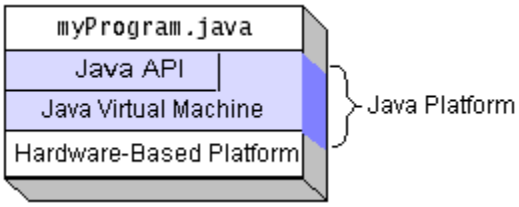
The Java Platform

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and Mac OS. Most platforms can be described as a combination of the operating system and hardware. 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. The Java platform has two components

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

You've already been introduced to the Java VM. 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 of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide. The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

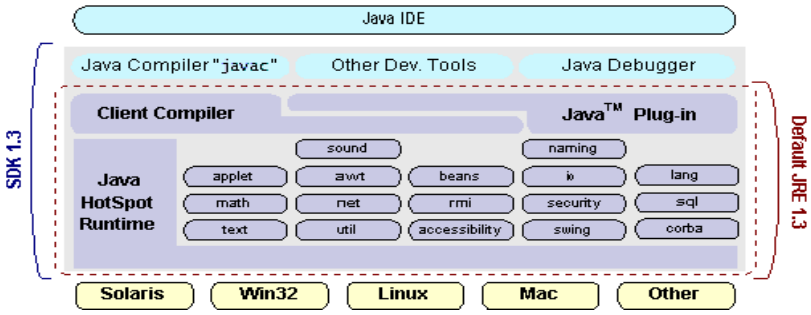
What Can Java Technology Do?

The most common types of programs written in the Java programming language are applets and applications. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser. However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server. How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials:** Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.

- **Applets**: The set of conventions used by applets.
- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components**: Known as Java Beans TM, can plug into existing component architectures.
- **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBCTM)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



3.3 Java IDE Process Diagram

How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.
- **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
- **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.
- **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
- **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java TM Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
- **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
- **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a

coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change. Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN. The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources. From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to

those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on. To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution. JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after. The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java. The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. SQL Level API

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to "generate" JDBC code and to hide many of JDBC's complexities from the end user.

2. SQL Conformance

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

3. JDBC must be implemental on top of common database interfaces. The JDBC SQL API must "sit" on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

4. Provide a Java interface that is consistent with the rest of the Java system

Because of Java's acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

5. Keep it simple

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

6. Use strong, static typing wherever possible

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

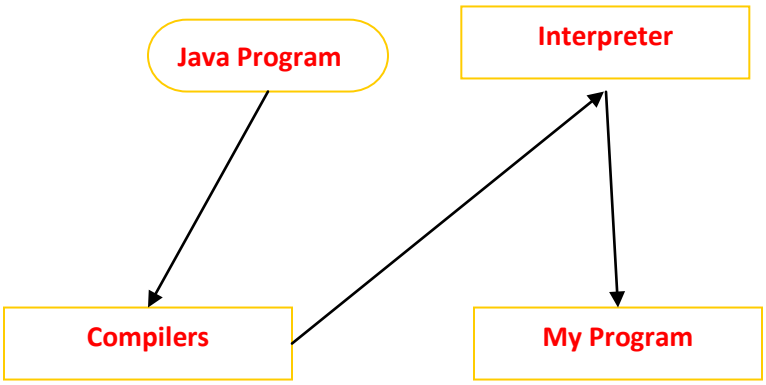
7. Keep the common cases simple

Because more often than not, the usual SQL calls used by the programmer are simple SELECT's, INSERT's, DELETE's and UPDATE's, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Finally we decided to precede the implementation using Java Networking. And for dynamically updating the cache table we go for MS Access database. Java ha two things: a programming language and a platform. Java is a high-level programming language that is all of the following

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

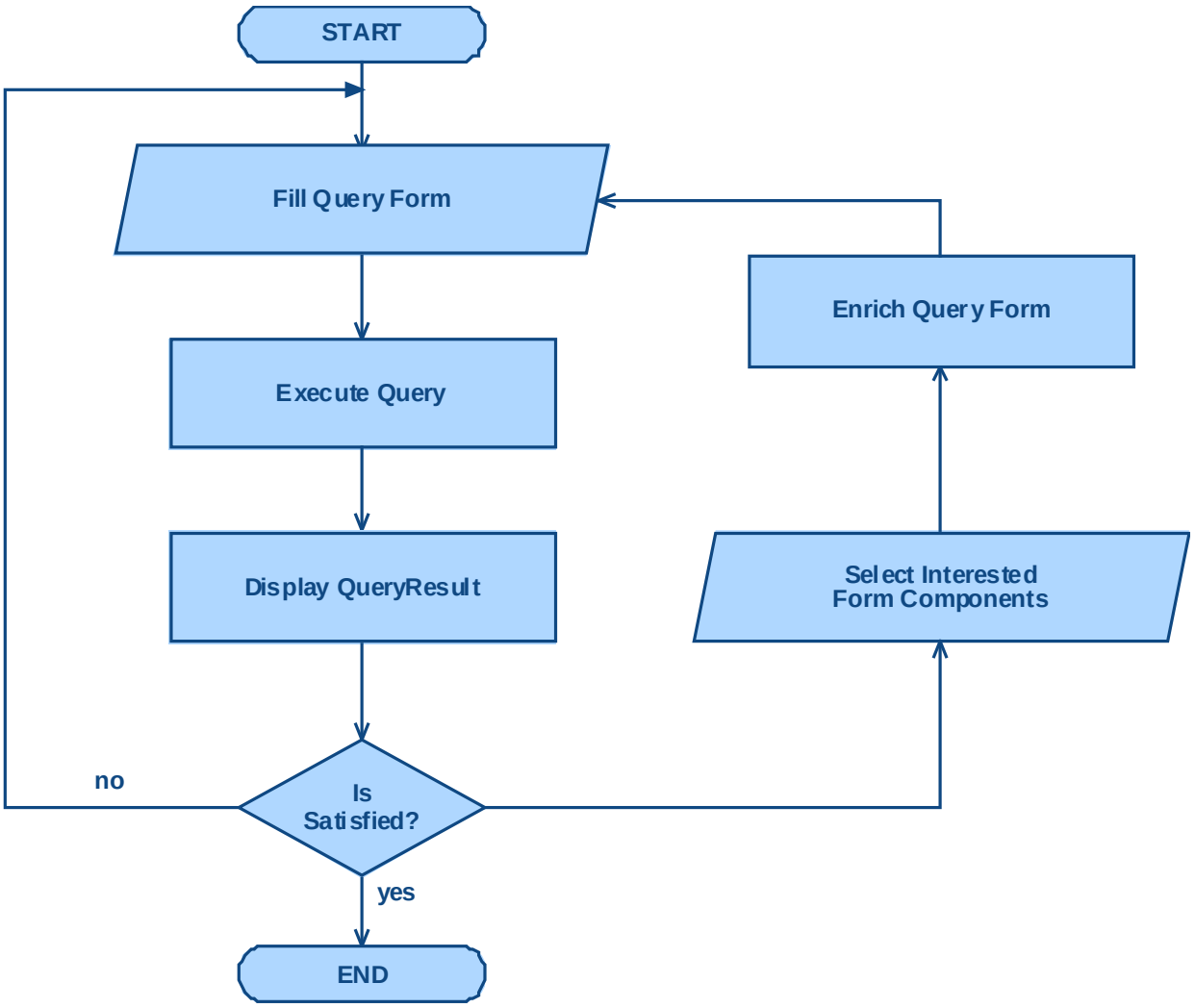
Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.



3.4 Java Program Execution Process

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware. Java byte codes help make "write once, run anywhere" possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

3. 3 CONTENT DIAGRAM OF PROJECT:



3.5 Content Diagram of Project

4. SYSTEM DESIGN

4.1 INTRODUCTION:

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. UML is a very important part of developing objects oriented software and the software development process. UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

Definition:

UML is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of the software system.

UML is a language:

It will provide vocabulary and rules for communications and function on conceptual and physical representation. So it is modeling language.

UML Specifying:

Specifying means building models that are precise, unambiguous and complete. In particular, the UML address the specification of all the important analysis, design and implementation decisions that must be made in developing and displaying a software intensive system.

UML Visualization:

The UML includes both graphical and textual representation. It makes easy to visualize the system and for better understanding.

UML Constructing:

UML models can be directly connected to a variety of programming languages and it is sufficiently expressive and free from any ambiguity to permit the direct execution of models.

UML Documenting:

UML provides variety of documents in addition raw executable codes.

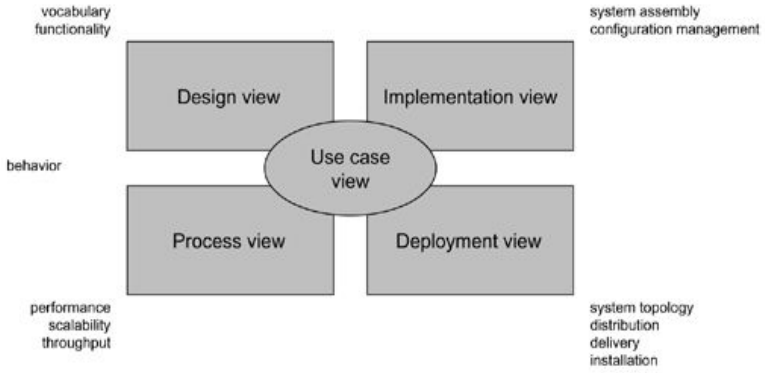


Figure: 4.1Modeling a System Architecture using views of UML

The use case view of a system encompasses the use cases that describe the behavior of the system as seen by its end users, analysts, and testers. The design view of a system encompasses the classes, interfaces, and collaborations that form the vocabulary of the problem and its solution. The implementation view of a system encompasses the components and files that are used to assemble and release the physical system. The deployment view of a system encompasses the nodes that form the system's hardware topology on which the system executes.

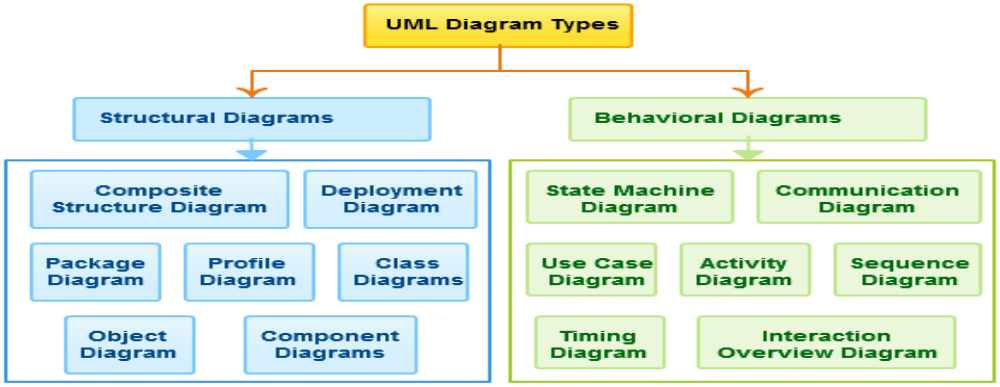


Figure: 4.2 UML diagram types

Uses of UML:

The UML is intended primarily for software intensive systems. It has been used effectively for such domain as:

- Enterprise Information System
- Banking and Financial Services
- Telecommunications
- Transportation
- Defense/Retails
- Medical Electronics
- Scientific Fields
- Distributed Web

Building blocks of UML:

The vocabulary of the UML encompasses 3 kinds of building blocks

- Things
- Relationships
- Diagrams

Things:

Things are the data abstractions that are first class citizens in a model. Things are of 4 types Structural Things, Behavioral Things, Grouping Things, A notational Things.

Relationships:

Relationships tie the things together. Relationships in the UML are Dependency, Association, Generalization, and Specialization.

UML Diagram types:

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships).There are two types of diagrams, they are:

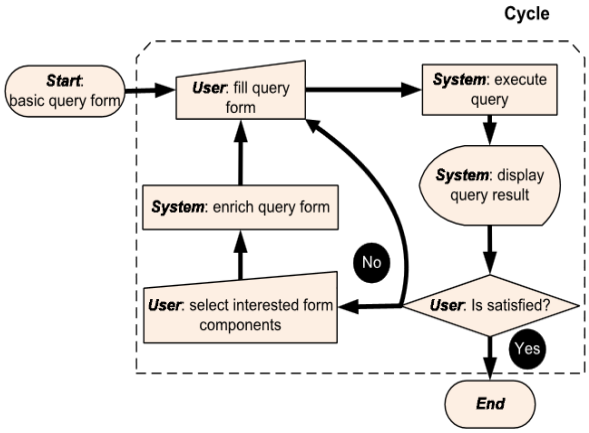
- Structural and Behavioral Diagrams.

Structural Diagrams:

The UML‘s four structural diagrams exist to visualize, specify, construct and document the static aspects of a system. I can View the static parts of a system using one of the following diagrams. Structural diagrams consist of Class Diagram, Object Diagram, Component Diagram, and Deployment Diagram.

Behavioral Diagrams:

The UML’s five behavioral diagrams are used to visualize, specify, construct, and document the dynamic aspects of a system. The UML’s behavioral diagrams are roughly organized around the major ways which can model the dynamics of a system. Behavioral diagrams consists of Use case Diagram, Sequence Diagram, Collaboration Diagram, State chart Diagram, Activity Diagram.

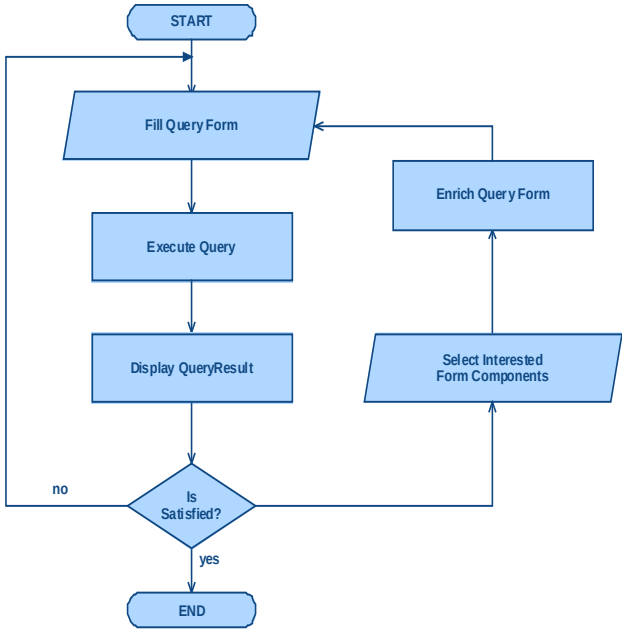


4.3 System Architecture

4.2 DFD / ER / UML DIAGRAMS:

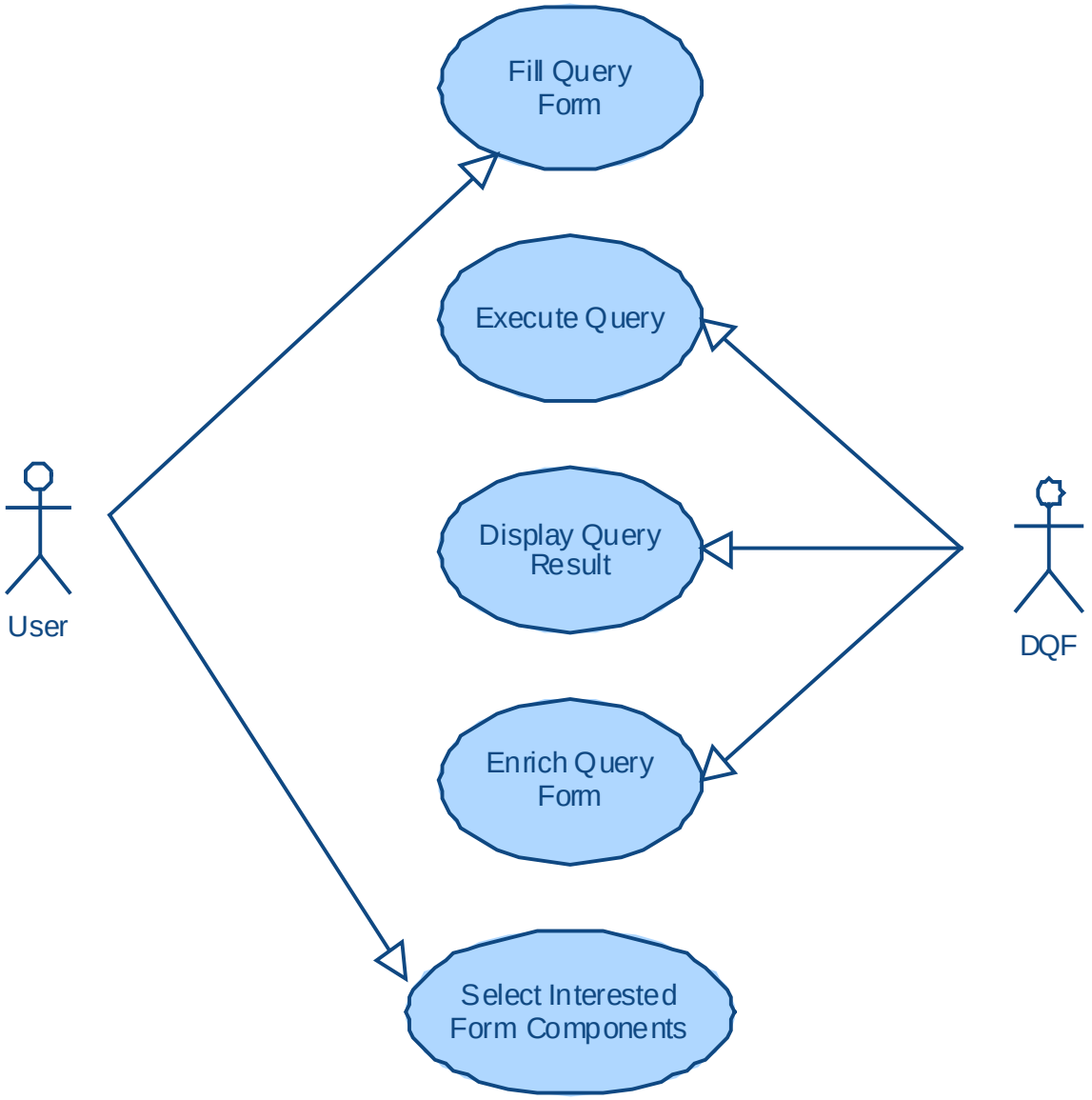
The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

FLOW CHART:



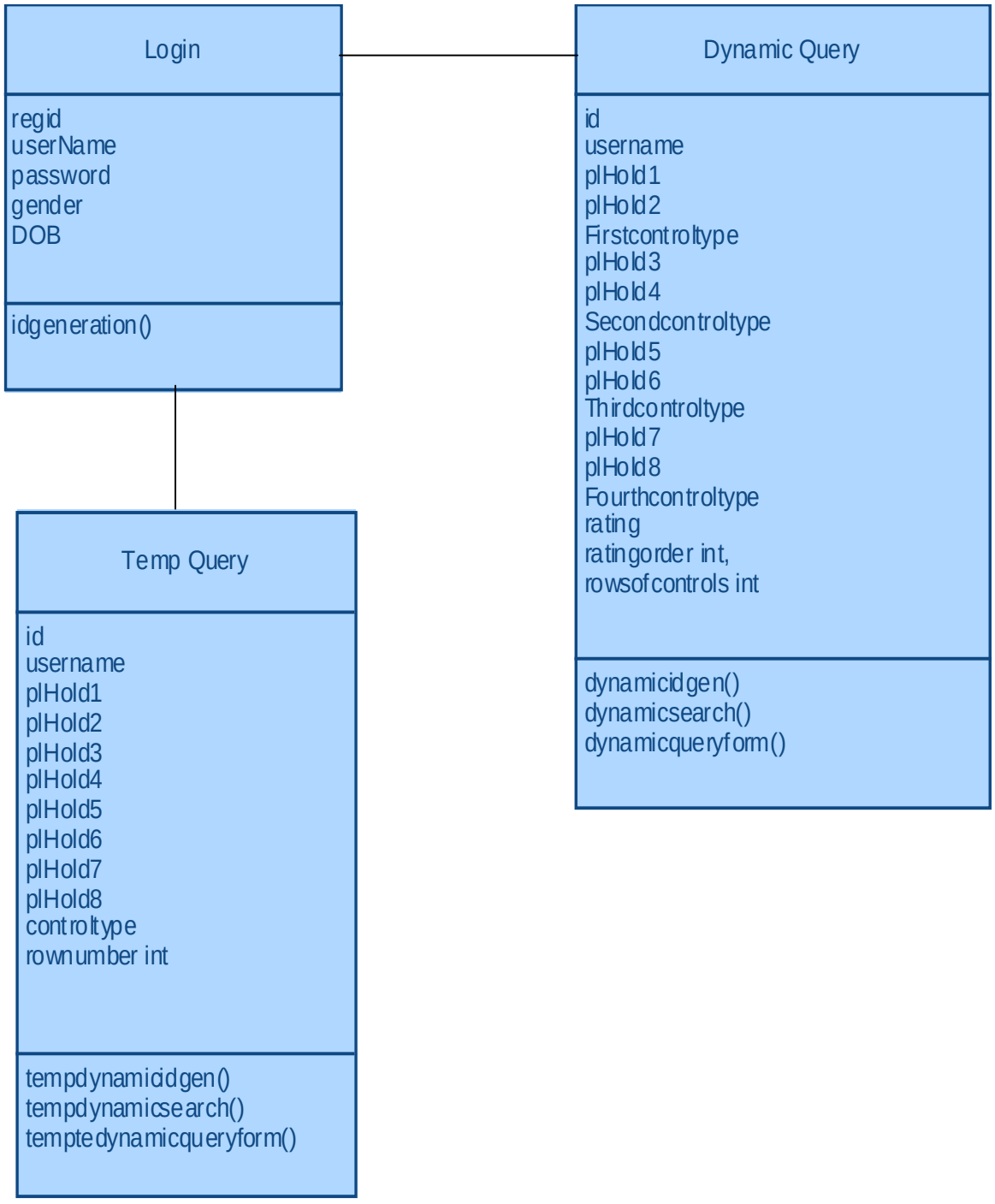
4.4 Flow Chart Diagram

USE CASE DIAGRAM:



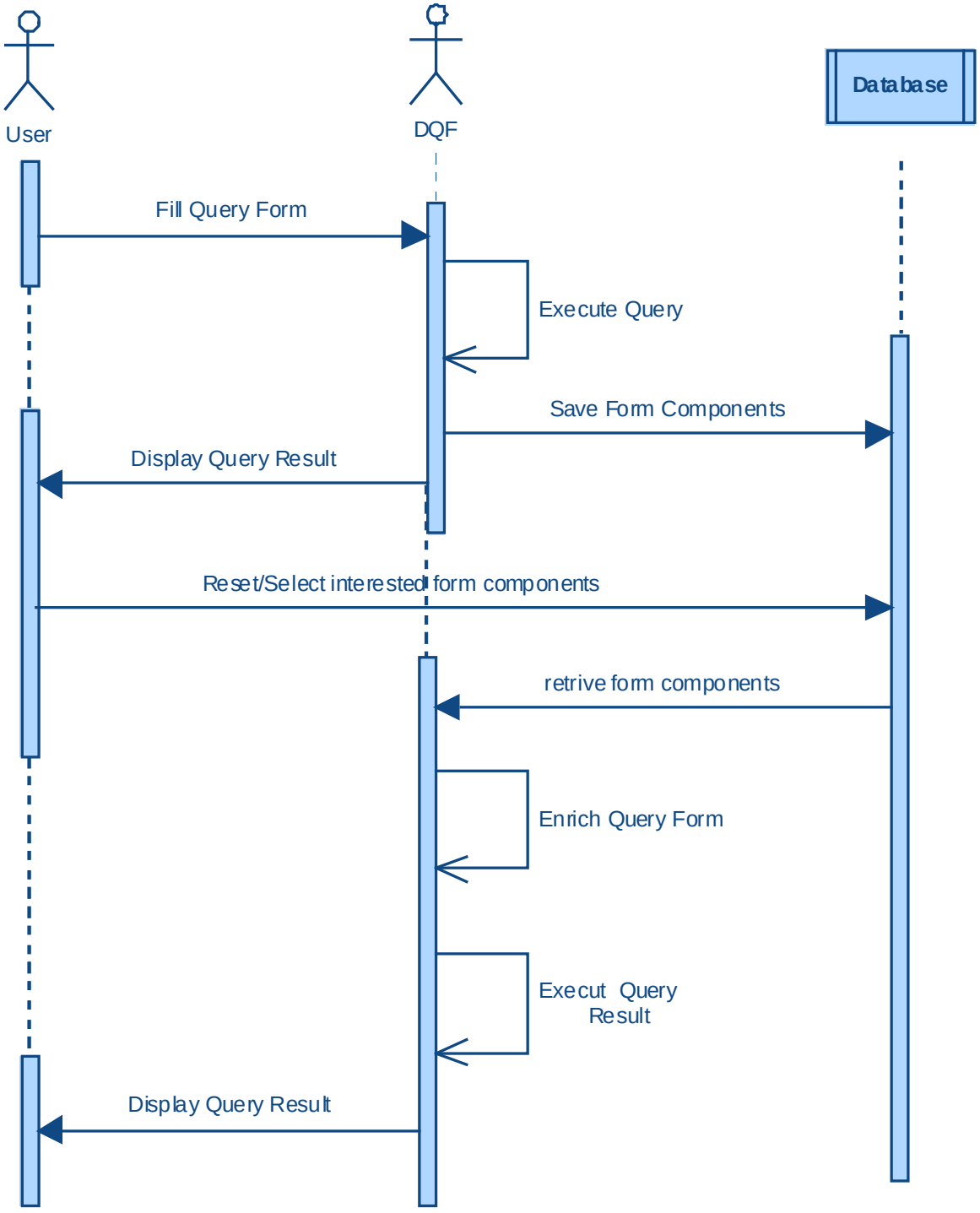
4.5 Use Case Diagram

CLASS DIAGRAM:



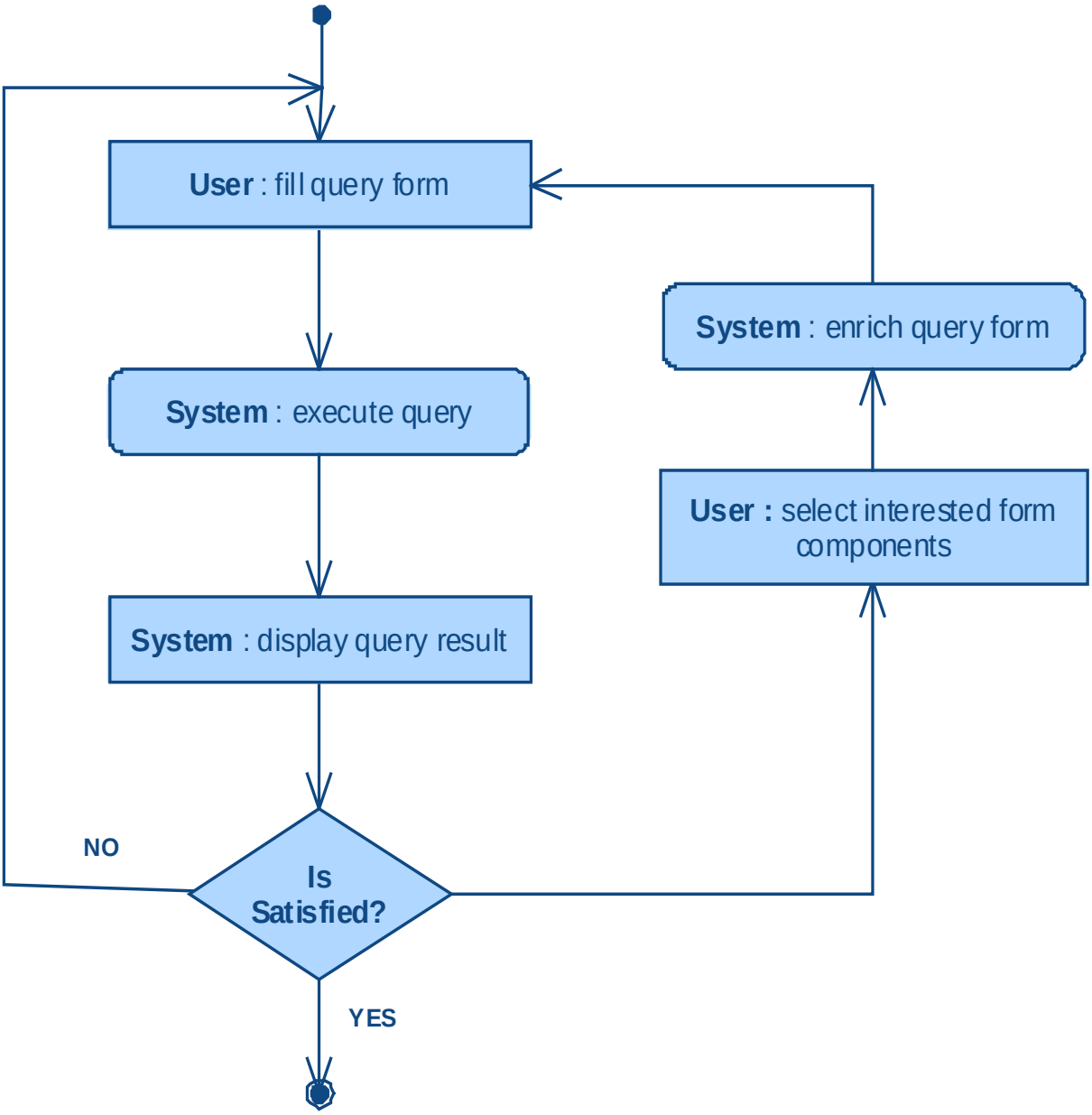
4.6 Class Diagram

SEQUENCE DIAGRAM:



4.7 Sequence Diagram

ACTIVITY DIAGRAM:



4.8 Activity Diagram

5. SYSTEM IMPLEMENTATION

5.1 INTRODUCTION:

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

5.2 METHOD OF IMPLEMENTATION:

5.2.1 Module Forms

The system is proposed to have the following modules along with functional requirements.

- Query Form Enrichment
- Query Execution
- Customized Query Form
- Database Query Recommendation

1. Query Form Enrichment:

- DQF recommends a ranked list of query form components to the user.
- The user selects the desired form components into the current query form.

2. Query execution:

- The user fills out the current query form and submits a query.
- DQF executes the query and shows the results.
- The user provides the feedback about the query results.

3. Customized Query Form:

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. If proposed a system which allows end-users to customize the existing query form at run time. However, an end-user may not be familiar with the database. If the database schema is very large, it is difficult for them to find appropriate database entities and attributes and to create desired query forms.

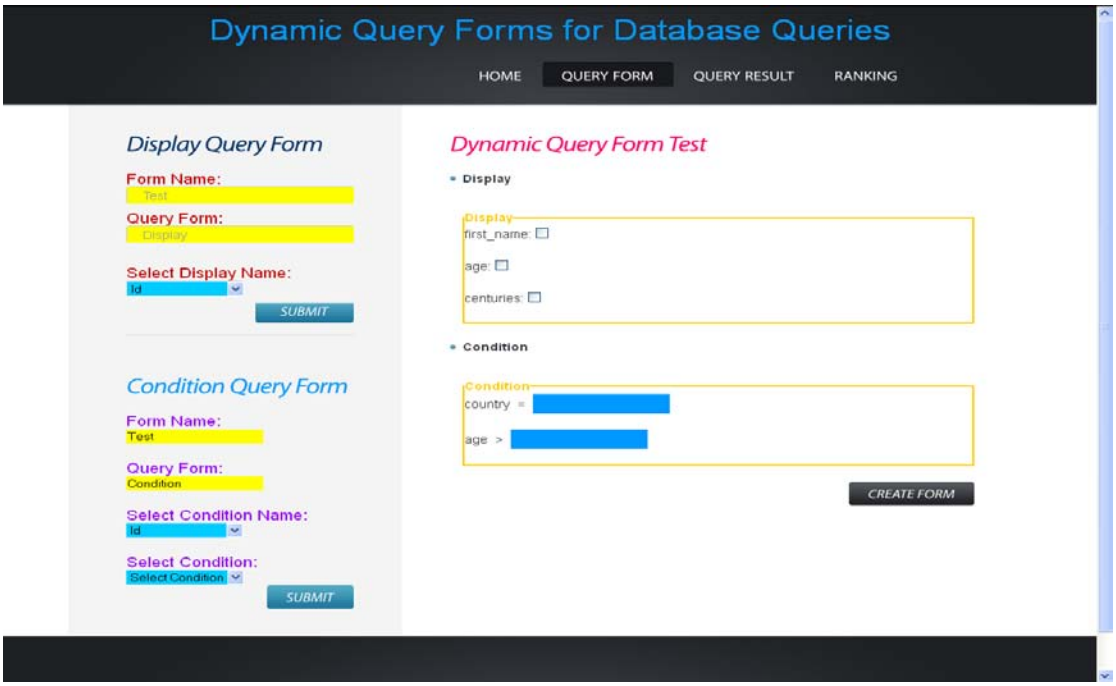
4. Database Query Recommendation:

Recent studies introduce collaborative approaches to recommend database query components for database exploration. They treat SQL queries as items in the collaborative filtering approach, and recommend similar queries to related users.

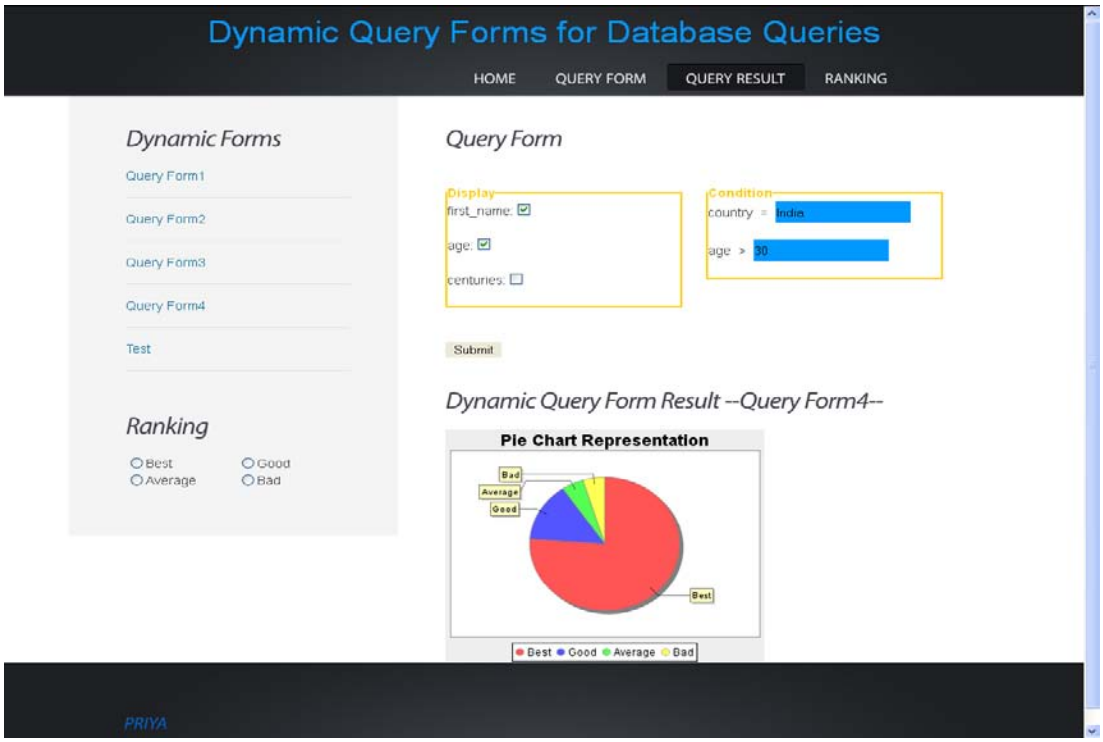
5.2.2 Screenshots:

Registration Page

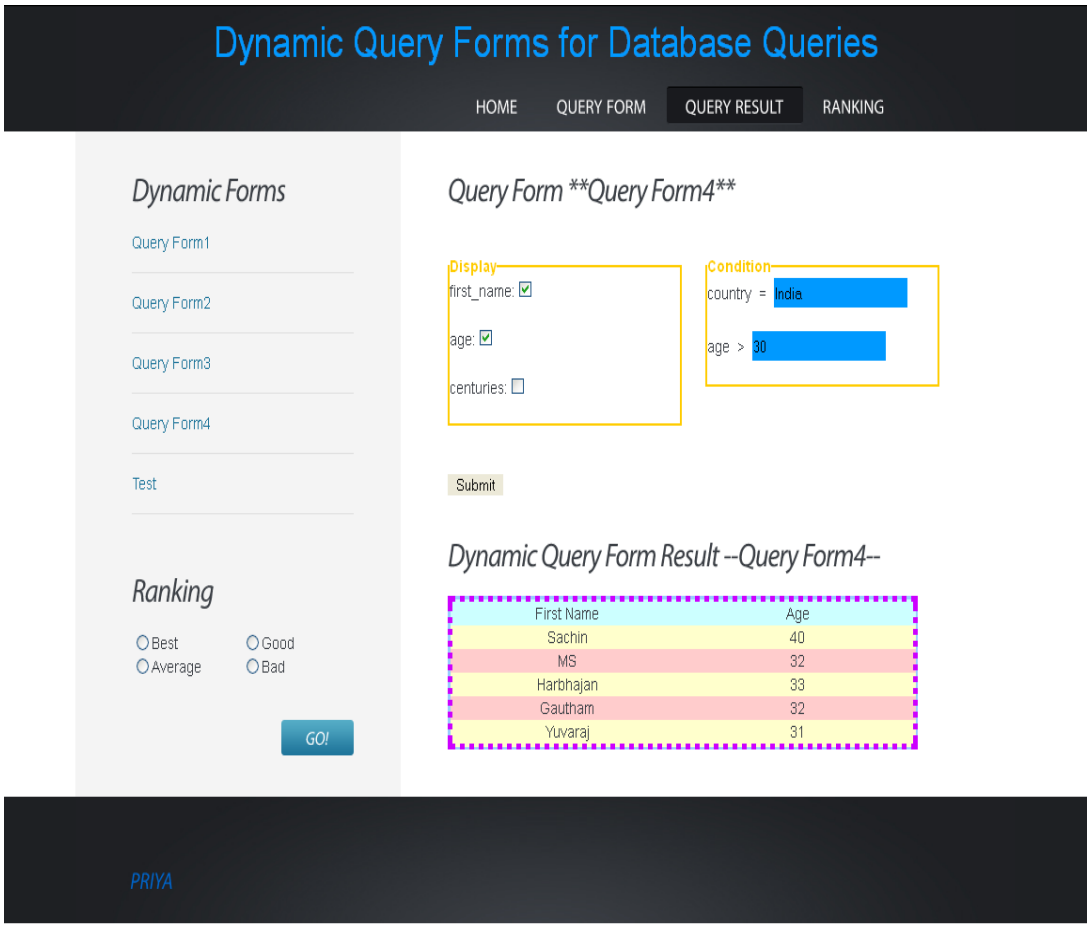
Home Page & Login Page Screen



Create Query Forms Screen



Query Result Screen



5.5

Dynamic Query Forms Screen

6. TESTING

6.1 INTRODUCTION:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product it is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 DESIGN OF TEST CASES AND SCENARIOS:

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases. Test strategy and approach.

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing:

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.3 VALIDATION:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify

Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7. CONCLUSION

In this paper we propose a 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. As future work, we will study how our approach can be extended to non relational data.

As for the future work, we plan to develop multiple methods to capture the user's interest for the queries besides the click feedback. For instance, we can add a text-box for users to input some keywords queries. The relevance score between the keywords and the query form can be incorporated into the ranking of form components at each step. As for the future work, we plan to develop multiple methods to capture the user's interest for the queries besides the click feedback. For instance, we can add a text-box for users to input some keywords queries. The relevance score between the keywords and the query form can be incorporated into the ranking of form components at each step.

8. REFERENCES

[1] Cold Fusion. http://www.adobe.com/products/coldfusion/.

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

[3] Easy Query. 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. Diversifying 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.

[8] S. Boriah, V. Chandola, and V. Kumar. Similarity measures for categorical data: A comparative evaluation. In Proceedings of SIAM International Conference on Data Mining (SDM 2008), pages 243–254, Atlanta, Georgia, USA, April 2008.

[9] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query recommendations for interactive database exploration. In Proceedings of SSDBM, pages 3–18, New Orleans, LA, USA, June 2009.

[10] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic information retrieval approach for ranking of database query results. ACM Trans. Database Syst. (TODS), 31(3):1134 – 1168, 2006.

Sites Referred:

http://java.sun.com

http://www.roseindia.com/

http://www.java2s.com/