DYNAMIC QUERY FORM

Abhisha KJ, Bachelor of Technology in CSE, NCERC

Adithyan Shaji E , Bachelor of Technology in CSE, NCERC

Akash KS, Bachelor of Technology in CSE, NCERC

Ashik M, Bachelor of Technology in CSE, NCERC

Mrs. Gracemol Thankachan, Assistant Professor, Department of CSE, NCERC

---***---

1. ABSTRACT

Query form is one of the most widely used ~~user interfaces~~ for querying databases. Modern scientific and web databases maintain large and heterogeneous data. With the rapid development of web information and scientific databases, modern databases become very large and complex. It is Difficult to design a set of static query forms to satisfy various ad-hoc database queries on complex databases.

Here we propose a novel database query form interface, which is able to dynamically generate query forms. The generation of the query form is an iterative process and is guided by the user. The essence of dynamic query form 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. 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

2. INTRODUCTION

Databases and database systems are an essential component of life in modern society. Databases and database technology have a major impact on the growing use of computers. It is fair to say that databases play a critical role in almost all areas where computers are used, including business, electronic commerce, engineering, medicine, genetics, law, education, and library science.

A database is a collection of related data, by data, we mean known facts that can be recorded and that have implicit meaning. A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested.

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications. Constructing the database is the process of storing the data on some storage medium that is controlled by the DBMS. Manipulating a database includes functions such as querying the database to retrieve specific data, updating the database to reflect changes in the mini world, and generating reports from the data. Sharing a database allows multiple users and programs to access the database simultaneously. An application program accesses th database by sending queries or requests for data to the DBMS. A query typically causes some data to be retrieved; a transaction may cause some data to be read and some data to be written into the database. Other important functions provided by the DBMS include protecting the database and maintaining it over a long period of time. Protection includes system protection against hardware or software malfunction (or crashes) and security protection against unauthorized or malicious access.

3. LITERATURE SURVEY

3.1. A Case for A Collaborative Query Management System

Nodira Khoussainova, Magdalena Balazinska, Wolfgang Gatterbauer, YongChul Kwon, and Dan Suciu

New environments are emerging where large numbers of users need to develop and run complex queries over a very large, shared\ data repository. Examples include large scientific databases and Web-related data. These users are not SQL savvy, yet they need to perform complex analysis on the data and are further constraine by the high cost of running and testing their queries, often on a shared server cluster.

3.2. Probabilistic Information Retrieval Approach for Ranking of Database Query Results

SURAJIT CHAUDHURI

We proposed a completely automated approach for the Many-Answers Problem which\ leverages data and workload statistics and correlations. Our ranking functions are based upon the probabilistic IR models, judiciously adapted for structured data. We presented results of preliminary experiments which demonstrate the efficiency as well as the quality of our ranking system.

3.3. USHER: Improving Data Quality with Dynamic Forms

Kuang Chen #1, Harr Chen _2, Neil Conway

In this paper, we have shown that probabilistic approaches can be used to design intelligent data entry forms that promote high data quality. USHER leverages data-driven insights to automate multiple steps in the data entry pipeline. Before entry, we find an ordering of form fields that promotes rapid information capture, driven by a greedy information gain principle. During entry, we use the same principle to dynamically adapt the form based on entered values

3.4. Automated Ranking of Database Query Results

Sanjay Agrawal Microsoft Research

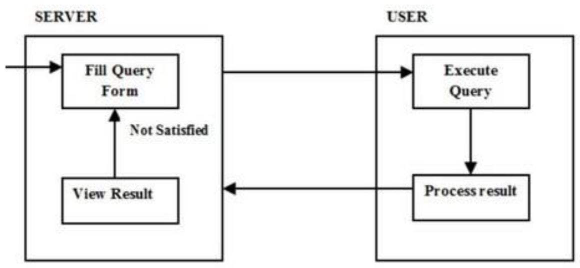
In this paper, we have presented our experience in attempting to build a generic automated ranking infrastructure for SQL databases. This is consistent with our research philosophy of seeding the relational database management infrastructure with functionality necessary and useful for data exploration

3.5. SnipSuggest: ContextAware Autocompletion for SQL

Nodira Khoussainova, YongChul Kwon, Magdalena Balazinska

In this paper, we present SnipSuggest, a system that provides on the- go, context- aware assistance in the SQL composition process. SnipSuggest aims to help the increasing population of non-expert database users, who need to perform complex analysis on their large-scale datasets, but have difficulty writing SQL queries. As a user types a query, SnipSuggest recommends possible additions to various clauses in the query using relevant snippets collected from a log of past queries.

3.6. Architecture of the System



4. METHODOLOGY

4.1. System Design

The System Design outlines the overall architecture of the application, breaking down how the components interact and function together.

1. TableSelectionDialog:

◦ Components: Combobox for data source/schema selection, JList for displaying tables, and CRUD operation buttons.

◦ Responsibilities: Fetches and displays tables from the database, allowing CRUD operations.

2. Database Connection:

◦ Interaction: Uses JDBC for MySQL connection and SQL queries.

◦ Operations: Fetches tables using DatabaseMetaData and performs INSERT, DELETE, UPDATE, and SELECT operations.

3. Table Display:

◦ JTable: Displays selected table data.

◦ Functionality: Allows users to modify and update table data after performing operations.

4.2. System Workflow

1. The user selects a data source (e.g., employee database) and a schema(e.g., <DEFAULTSCHEMA>)from
dropdownmenus.

2. . Select Table:

- A list of available tables is displayed, and the user selects the table they want to interact with.

3. . View Table Data:

5. The user clicks the OK button after selecting a table, which opens a new window displaying the table's data in a JTable.

6. Perform CRUD Operations:

- add: The user can add a new row to the table by filling in the required fields in a form.
- Update: The user selects a row in the table and can edit the values in the form. Upon submitting, the selected row is updated.
- Delete: The user selects a row and deletes it from the table based on the primary key.

7. Data Refresh:

- After any CRUD operation, the table data is refreshed to reflect the changes made.

5. IMPLEMENTATION AND RESULTS

5.1. System Development

• GUI Development: The system uses Java Swing components such as JComboBox, JList, JTable, and JButton for user interaction and displaying data.

• Database Interaction: JDBC is employed to connect to a MySQL database, retrieve available tables using DatabaseMetaData, and perform CRUD operations on the selected tables.

• Security: Though security is not heavily emphasized in the system, SQL injection prevention is implemented using Prepared Statement for executing SQL queries.

5.2. System Testing

• Unit Testing: Individual methods, such as fetching table names and executing CRUD operations, were tested to ensure correct functionality. Each method was tested for edge cases, including empty results and database errors.

• Integration Testing: Integration between the GUI components and the database was tested by simulating user interactions such as selecting a table, adding a row, and updating/deleting records. The database interactions were confirmed to work as expected, ensuring that data reflected changes in both the table and the database.

- User Acceptance Testing : End users interacted with the system to validate the overall user experience. Feedback indicated that the interface was intuitive,and users could easily perform operations without issues. All user-reported bugs were addressed and resolved

5.3 RESULTS

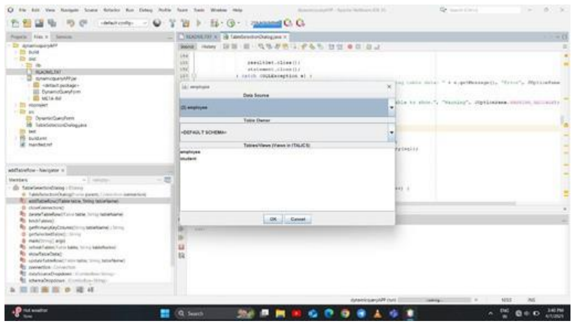


Fig 1.data source

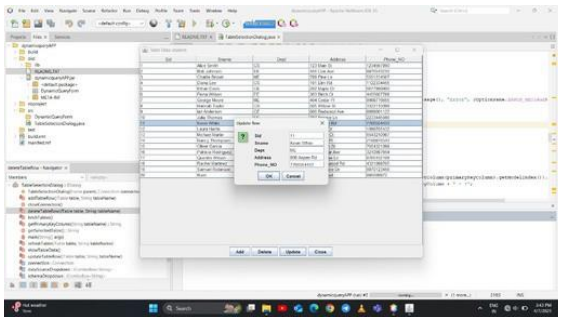


Fig 5.updating a student record

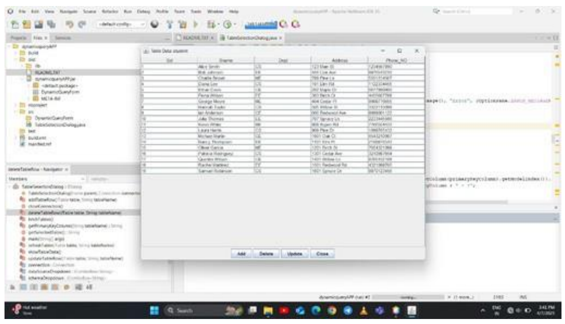


fig 2. student data table

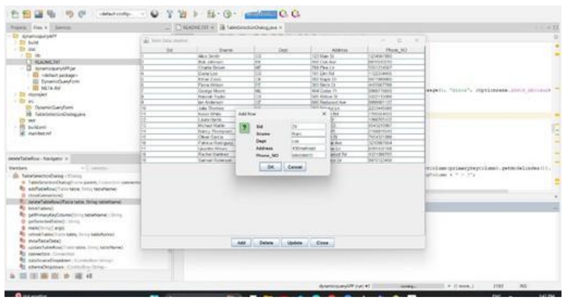


fig 3.adding a new record

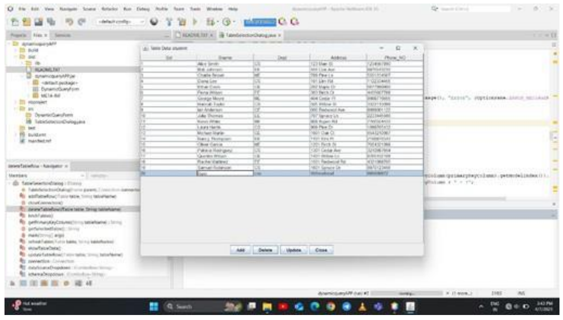


Fig 4.student data table display

6. ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my Mentor Mrs.Gracemol Thankachan for their invaluable guidance, encouragement, and continuous support throughout the development of this project. Their expertise and insightful feedback played a crucial role in enhancing my understanding and improving the project's quality. I extend my heartfelt thanks to my team members for their dedication, teamwork, and commitment, which were instrumental in successfully implementing the Dynamic Query Form Their contributions in various aspects of design, development, and testing greatly enriched the project.
I am also thankful to Nehru College of Engineering and Research Centre, Thiruvillamala for providing the learning environment, enabling me to explore and apply my technical knowledge effectively.
Finally, I deeply appreciate the unwavering support and encouragement from my family and friends, whose motivation kept me inspired throughout this journey. This project has been a significant learning experience, and I am truly grateful to everyone who contributed to its successful completion.

7. CONCLUSION

We propose a dynamic query form generation approach; query form is one of the most widely used user interfaces for querying databases which helps users dynamically generate query forms. 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. The key idea is to use a probabilistic model to rank form components based on use 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 make it easier for users to customize query forms. As future work, we will study how our approach can be extended to non-relational data. 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 textbox for users to input some keywords queries.

REFRENCES

[1] Cold Fusion. http://www.adobe.com/products/coldfusion/.
[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. Diversifying
search results. In Proceedings of WSDM, pages 5–14,
Barcelona, Spain, February 2009.
S. Agrawal, S. Chaudhuri, G. Das, and A.
Gionis. Automated
ranking of database query results. In CIDR, 2003.
[7] 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.
[8] 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.
[9] 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.
11] K. Chen, H. Chen, N. Conway, J. M. Hellerstein, and T. S.
Parikh. Usher: Improving data quality with dynamic forms.
In Proceedings of ICDE conference, pages 321–332, Long Beach,
California, USA, March 2010.
[12] E. Chu, A. Baid, X. Chai, A. Doan, and J.
F. Naughton.
Combining keyword search and forms for ad hoc querying of
databases. In Proceedings of ACM SIGMOD Conference, pages
349–360, Providence, Rhode Island, USA, June 2009.