Adaptive Indexing in Very Large Databases

Andrew Johnson, and Farshad Fotouhi
Wayne State University

Very large databases contain large amounts of interrelated information. This information is often stored in relational databases with hundreds of tables and thousands of rows per table. Clustering is an effective way to reduce the information-overhead associated with finding information among these tables, allowing the user to browse through the clusters as well as the individual tables. In this paper, we compare the use of two adaptive algorithms (genetic algorithms, and neural networks) in clustering the tables of a very large database. These clusters allow the user to index into this overwhelming number of tables and find the needed information quickly. We cluster the tables based on the user’s queries and not on the content of the tables, thus the clustering reflects the unique relationships each user sees among the tables. The original database remains untouched, however each user will now have a personalized index into this database.

Relational databases are designed to deal with limited ranges of data on specific topics. The form of the data is known ahead of time and the database tables and their relationships are clearly defined before the data is entered. Very large databases can contain a much larger amount of data on many different, but related topics. The form of the data is often not known ahead of time as the data is collected from a wide range of sources.

Very large databases are used in many areas. They store statistics such as economic data and census data. They store inventory data. They store data from scientific experiments and simulations such as climate modelling and human genome mapping. They are accessed by users from a wide range of disciplines, mostly unfamiliar with databases and their associated query languages. These users need to search for specific pieces of data quickly, and browse through related information to see if it is of value to them. They need to relate information from different tables in the database.

Much of the difficulty in accessing data in very large databases comes from the enormous amount of data that is involved; but the organization of this data is also a major problem (French et al., 1990.) Users from various communities see different relationships between sets of data. Certain information is important to certain users and certain information is not. As time passes, the data in the database will be important to different users. Typically a generic interface is provided to the data. This gives users from all backgrounds a way to access the data, but each of the users must conform to this generic structuring of the data.

When an interface to a very large database is designed, its creator typically imposes a generic structure on the database - a hierarchical menu system allowing the user to move through an ordering of the tables. This gives users from all backgrounds a way to access the data, but each of the users must conform to this generic structuring of the data. This approach has several shortcomings: 1) The menu system does not provide enough flexibility for a wide range of researchers, 2) The users may not know enough about the domain to make appropriate choices, 3) It does not help the user with ill-defined queries.

Graphical query languages have been proposed to simplify the interface (Kuntz and Melichert, 1989, Ozsoyoglu and Wang, 1993.) Graphical query languages make the database schema more visible, reduce typing, and allow users to rely on recognition rather than memorization. This approach has several shortcomings: 1) The schema of scientific databases

Manuscript originally submitted October 13, 1993; Revised April 29, 1994; Accepted June 17, 1994 for publication.
are so large, and complicated that the user rapidly runs out of screen real-estate, 2) The graphical metaphor quickly becomes cumbersome for complicated queries.

Ioannidis, et al. (1992; 1993) developed a graphical interface for the management of scientific experiments and data using the Object-Oriented data model MOOSE. The user interacts with the database through the schema. The system makes large schemas more manageable by allowing the user to hide parts of the schema, collapse sections of the schema into nodes, and use reference nodes to eliminate long arcs. While useful for people involved in the original experiment, this approach has several shortcomings for users less familiar with the original experiment: 1) The users may not know what data is available, 2) The users may not know enough about the domain to make appropriate choices, 3) It gives users a variety of choices without sufficient descriptive material to make that choice, 4) The original schema may not match the relationships seen by all users.

Instead of forcing the users to conform to the structure of the database, we can mold the structure of the database to the needs of the individual users and thereby reduce their confusion when interacting with the database. Our solution is to cluster the tables of a very large database based on the user’s queries. The user of the database needs the benefits of clustering so we should involve the user in the clustering process. The content of the tables is irrelevant, only the fact that the user sees a relationship between the tables is important. This flexibility will be increasingly important as the size, breadth, and accessibility of very large databases increases.

We collect data on the user’s queries. This information is then given to a genetic algorithm or to a neural network which partitions the tables of the database into a hierarchy of clusters. The genetic algorithm and neural network generate very good clusterings much faster than a deterministic exhaustive search algorithm. Each user now has the choice of browsing through the existing generic interface, or using a hierarchy of clusters as an index to move quickly to the appropriate tables in the database. The original database remains untouched, however each user now has a personal index into this document. See Figure 1.

Clustering has been found to be a very effective means of reducing information overhead in other large information systems with related components such as hypertext systems Botafogo and Schneiderman, 1991, Haru et al., 1991.) Genetic algorithms have been found to be useful in improving document retrieval accuracy by integrating usage patterns from various users (Gordon, 1988.)

As new users begin to work with this database, they can choose from the existing clusterings. A new user can choose to look at the database from various points of view. This gives each user a starting point nearer their own needs than the generic interface. Experienced users will also be able to access their data faster and more conveniently because the database will adapt to them. We have tested this approach using users from several disciplines who found it to be useful in accessing a very large database.

Section 2 discusses how we collect information from the user. Section 3 discusses clustering in very large databases. Sections 4 and 5 respectively discuss how we use a genetic algorithm and neural network to create the hierarchy of clusters. Section 6 discusses our implementation of both clustering methods on a very large database. Section 7 discusses uses for this clustering. Finally, Section 8 gives our conclusions and plans for future work.

Information Collection

Users retrieve data from a relational database using a query language such as SQL. We monitor the user’s queries, and store information about the tables accessed in a list.

Each line of the list contains information on a single table in the database. This line lists all of the columns that have been displayed for that table, and all the columns that have been used to connect this table to other tables in the database. The format of each line of the list is shown below:

\[
table_i : \{selectCol_i\} \cdot \{joinCol_i \cdot joinCol_j @ table_j\} \cdot *
\]

where:

- \(table_i\) is the table we are interested in.
- \(selectCol_i\) are the columns in table that have been selected.
- \(joinCol_j\) are the columns in table that have been joined with columns \(joinColumn_j\) of table \(table_j\).

Three small sample SQL queries are converted into their list form as shown in Figure 2. When a user begins to search for information in the database they either choose to work with an existing list or create a new empty list. The user can give this new empty list an appropriate name. As the user works with the database, the list keeps track of the tables and columns the user is accessing through their queries.
Related Content

An Empirical Study on the Determinants of Effective Database Management
[www.igi-global.com/article/empirical-study-determinants-effective-database/51099?camid=4v1a](www.igi-global.com/article/empirical-study-determinants-effective-database/51099?camid=4v1a)

CMU-WEB: A Conceptual Model for Designing Usable Web Applications
[www.igi-global.com/article/cmu-web-conceptual-model-designing/51225?camid=4v1a](www.igi-global.com/article/cmu-web-conceptual-model-designing/51225?camid=4v1a)

Scalable XML Filtering for Content Subscriptions
Ryan Choi and Raymond Wong (2011). *Theoretical and Practical Advances in Information Systems Development: Emerging Trends and Approaches* (pp. 120-152).
[www.igi-global.com/chapter/scalable-xml-filtering-content-subscriptions/52955?camid=4v1a](www.igi-global.com/chapter/scalable-xml-filtering-content-subscriptions/52955?camid=4v1a)

An Ontological Analysis Framework for Domain-Specific Modeling Languages
[www.igi-global.com/article/an-ontological-analysis-framework-for-domain-specific-modeling-languages/201041?camid=4v1a](www.igi-global.com/article/an-ontological-analysis-framework-for-domain-specific-modeling-languages/201041?camid=4v1a)