isrjournals

International Journal of Advanced Research in Computer Science Engineering and Information Technology

Volume: 4, Issue: 3,Special Issue: 3 ,May,2016 ,ISSN_NO: 2321-3337

Personalized Search and Recommendations For Movies in A Relational Database

S.ARUN RAJ [1], V. LOGANATHAN[2] PG Scholar, Department of CSE,

Saveetha Engineering College, Thandalam-638112, Chennai, India

Assistant Professor, Department of CSE, Saveetha Engineering College,

Thandalam-638112, Chennai, India alrocky07@gmail.com

loganathan@saveetha.ac.in

ABSTRACT—*PrefDB, a preference-aware relational system that transparently and efficiently handles queries with preferences. In its core, PrefDB employs a preference-aware data model and algebra, where preferences are treated as first-class citizens. We define a reference using a condition on the tuples affected, a scoring function that scores these tuples, and a confidence that shows how confident these scores are. In our data model, tuples carry scores with confidences. Our algebra comprises the standard relational operators extended to handle scores and confidences. For example, the join operator will join two tuples and compute a new score-confidence pair by combining the scores and confidences that come with the two tuples. In addition, our algebra contains a new operator, prefer, that evaluates a preference on a relation, i.e., given as inputs a relation and a preference on this relation, prefer outputs the relation with new scores and confidences. During preference evaluation, both the conditional and the scoring part of a preference are used. The conditional part acts as 'soft' constraint that determines which tuples are scored without disqualifying any tuples from the query result. In this way, PrefDB separates preference evaluation from tuple filtering. This separation is a distinguishing feature of our work with respect to previous works. It allows us to define the algebraic properties of the prefer operator and build generic query optimization and processing strategies that are applicable regardless of the type of reference specified in a query or the expected type of answer.*

Keywords—Database personalization, preferences.

1, Introduction

Data mining is the process of extracting patterns from data. As more data are gathered, with the amount of data doubling every three years, data mining is becoming an increasingly important tool to transform these data into information. It is commonly used in a wide range of profiling practices, such as marketing, surveillance, fraud detection and scientific discovery.

Data mining commonly involves four classes of task: Classification - Arranges the data into predefined groups. For example an email program might attempt to classify an email as legitimate or spam. Common algorithms include nearest neighbor, Naive Bayes classifier and neural network.Clustering - Is like classification but the groups are not predefined, so the algorithm will try to group similar items together.

• Regression - Attempts to find a function which models the data with the least error. A common method is to use Genetic Programming.
• Association rule learning - Searches for relationships between variables. For example a supermarket might gather data of what each customer buys. Using association rule learning, the supermarket can work out what products are frequently bought together, which is useful for marketing purposes. This is sometimes referred to as "market basket analysis".

Data mining, the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. The automated, prospective analyses offered by data mining move beyond the analyses of past events provided by retrospective tools typical of decision support systems. Data mining tools can answer business questions that traditionally were too time consuming to resolve. They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations.

2, Requirements

PrefDB takes user profile along with preferences and stores in the database. Already there exists two traditional approaches plug-in and native approach. In plug-in approach preferences are translated into complex queries. In native approach operators are injected into the query engine to execute the queries along with the preferences. The disadvantage of plug-in is,it operates above the query engine. Therefore it is hardwired. The disadvantage in native approach is the entire database core must be changed. PrefDB is the use of an extended relational data model and algebra that allow expressing different flavors of preferential queries.

In prefDB once the user has logged in, the user has to provide his profile information with his interests. For example in movie rental application the user provides his interests in terms of favorite movies, actors, directors, genres. The movie will be recommended by filtering the recommendation in terms of high, medium and low level as per the user’s choice. In high level recommendation there will be more number of constraints ,which results in retrieving closest match to the users interest and preferences that is in a movie rental application you will get the favorite movie .

Whereas, in medium and low level recommendation the number of results will be more, if the level is medium then the results will be based on favorite movie and director and if the level is low the results will be based on all the four categories , but the movies which satisfy any one constraint will be selected . Similar to prefDB there is careDB which is used in directing the users to the interested restaurants based on his cuisine interest .Here, the external factors like traffic, climate, waiting in restaurants, etc are also included. ThecareDB along mapping software, location detection software (GPRS, Antenna) processes the query and provides the user with the best result of the preferred restaurant of the user in his handheld device. The result will be based on the score and confidence will vary the result based on the previous choice of restaurants chosen by the customer.

The constraints for the query formation will include the environmental factors like climatic changes, road block, traffic etc... And other external factors like long waiting time in restaurants,restaurants closed preferred cuisine unavailable etc... The users get the query result in their handheld device with a map layout for showing the direction.

- It provides several query optimization strategies for extended query plans.
- It describes a query execution algorithm that blends preference evaluation with query execution, while making effective use of the native query engine.
- PrefDB implements the framework and methods in a prototype system,that allows the transparent and efficient evaluation of preferential queries on top of a relational DBMS.
- The extended query plan is constructed which contains all the operators that comprise a query and optimize it.
- The goal of query optimization is to minimize the cost related with preference evaluation.

3, PROPOSED METHOD

The extended query plan is constructed which contains all the operators that comprise a query and optimize it. Then, for processing the optimized query plan, our general strategy is to blend query execution with preference evaluation and leverage the native query engine to process parts of the query that do not involve a prefer operator. Given a query with preferences, the goal of query optimization is to minimize the cost related with preference evaluation. Based on the algebraic properties of prefer, Toapply a set of heuristic rules aiming to minimize the number of tuples that are given as input to the prefer operators. We further provide a cost-based query optimization approach. Using the output plan of the first step as a skeleton and a cost model for preference evaluation, the query optimizer calculates the costs of alternative plans that interleave preference evaluation and query processing in different ways. Two plan enumeration methods, i.e., a dynamic programming and a greedy algorithm are proposed. For executing an optimized query plan with preferences, To describe an improved version of our processing algorithm (GBU) (an earlier version is described in. The improved algorithm uses the native query engine in a more efficient way by better grouping operators together and by reducing the out-of-the-engine query processing.

3.1 Advantage of Proposed System

- A preference aware relational framework is done.
- A prototype system implementation is done.
- Cost based query optimization where the cost is reduced.
- Improved query execution improves the performance.
- Score and confidence method is used.

4, ARCHITECTURE

4.1 Introduction

System architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. The system comprised the components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture; collectively these are called architecture description languages (ADLs).

4.2, MODULES DESCRIPTION

4.2.1 Registration & Interest Sum Up:

During Registration, each and every user will provide their basic information for authentication. After that, user has to provide their profile information and their interests about their movie. Based upon their, and with our movie datasets, we can be able to analyze their interest about the movie and have to provide the recommended movies to the particular user.

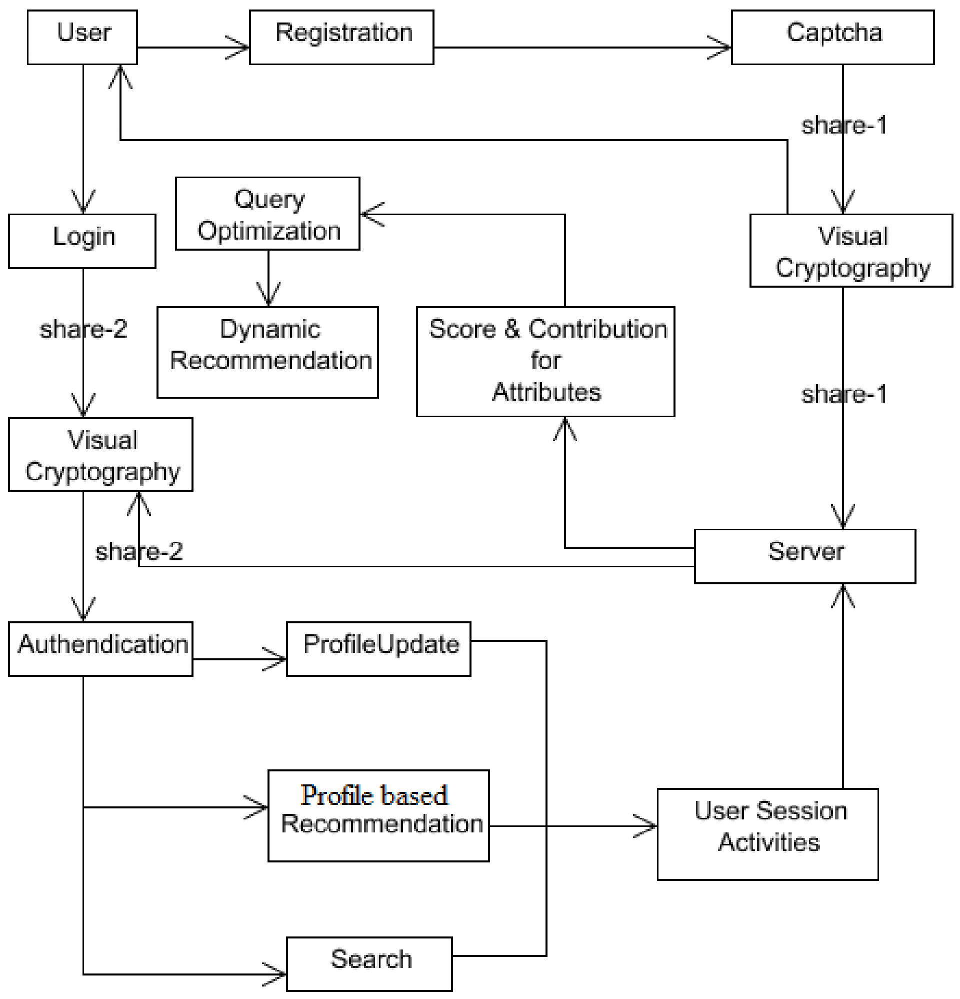

Figure 4.1 System Architecture

4.2.2 Query Optimization & Execution:

Extended relational operators and the prefer operator do not change how tuples are filtered or joined; for instance, prefer operator does not filter any tuples. Therefore our extended relational operators do not affect the non-preference related cost. Thus, we can expect that the join order that is suggested by the native query optimizer for a query if no prefer operators were present, will still yield good performance for the non-preference part of the same query with the prefer operators. Based on this observation, we will keep the suggested join order and we will consider the non-preference related cost as fixed. Then, the goal of our query optimizer will be to minimize the cost related with preference evaluation. Typically, the most critical parameter that shapes the processing cost of query evaluation is the disk I/Os, which is proportional to the number of tuples

Volume: 4, Issue: 3,Special Issue: 3 ,May,2016 ,ISSN_NO: 2321-3337

flowing through the operators in the query plan. Assuming a fixed position for the other operators, the goal of our query optimizer is essentially to place the prefer operators inside the plan, such that the number of tuples flowing through the score tables is minimized. The execution engine of PrefDB is responsible for processing a preferential query and supports various algorithms.

4.2.3 Query Formation:

A preferential query combines p-relations, extended relational and prefer operators and returns a set of tuples that satisfy the boolean query conditions along with their score and confidence values that have been calculated after evaluating all prefer operators on the corresponding relations. Intuitively, the better a tuple matches preferences and the more (or more confident) preferences it satisfies, the higher its final score and confidence will be, respectively. The query parser adds a prefer operator for each preference. Finally, the query parser checks for each preference, whether it involves an attribute (either in the conditional or the scoring part) that does not appear in the query and modifies project operators, such that these attributes will be projected as well.

4.2.4 Query Reformulation:

Query Reformulation is a process of modifying the Object Oriented Query with users Current Preference which is extracted previously from users session information with some special operators.Here as soon as the Object Oriented Query is injected in the execution engine. It will provide a Result set which will be scrootinized and sorted set.No need to perform Filtering and sorting operations which is done in multiple levels in existing systems for redundant data elimination and ranking the most appropriate results thereby achieving personalization in user results.

5, CONCLUSION.

The existing system which uses the plug-in and native approaches are to be less effective when compared with the proposed system. In plug-in methods, the way preferences is used, for example as additional query constraints or as ranking constructs, the query execution flow as well as the expected type of answer (e.g., top-k or skyline) are all hard-wired in this method, which hinders application development and maintenance. On the other hand, native methods consider preference evaluation and filtering as one operation. Due to this tight coupling, these methods are also tailored to one type of query.

Furthermore, they require modifications of the database core, which may not be feasible or practical in real life. Thus, these both approaches do not offer a holistic solution to flexible processing of queries with preferences and it is less effective. The proposed system overcomes this problem by introducing the PreDB which helps the user by providing their interests in terms of favorite movies, actors, directors, genres. The movie is recommended based on filtering the recommendation in terms of high, medium and low level as per the user’s choice where it produces to be very effective. Based on this, the search results in the existing system produces the search for the recent movies returns several results making it hard for the user to choose.

Taking into account that the user prefers comedies and action movies would help focus on query to fewer recent movies. On the other hand, if the query criteria are too restrictive, the query might produce no results at all. This makes the search to be very expensive. Whereas, the proposed system produces the exact search results based on the user's choice by the preference evaluation and filtering. This results to be an inexpensive process. Another important criteria is that the proposed system even produces the results of a movie recommendation for the user which is external to the database whereas the existing system produces the results only which contains in the database and produces no results when external to the database.

6, FUTURE ENHANCEMENT

In future research, a preference-aware data modelwhere preferences appear as first-class citizens and preference evaluation is captured as a special 'prefer' operator.Westudied the algebraic properties of the new operator and applied them in order to develop cost-based query optimizations and holistic query processing methods.Wepresented a framework that is (i) flexible in handling different flavors of preferential queries, (ii) closer to the database than plug-in approaches, (iii) yet non-obtrusive to the database engine. Our experiments using a prototype system implementation demonstrated the performance advantages of our methods when compared with two variation of a plug-in strategy.In the future, we aim to explore combining the prefer operator with the rank and rank join operators defined in order to enable early pruning of results based on score or confidence during query processing.

REFERENCES

#1. G. Adomavicius and A. Tuzhilin (Jun. 2005), "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," IEEE Trans. Knowl. Data Eng., vol. 17, no. 6,pp. 734–749,.

#2. R. Agrawal, R. Rantzau, and E. Terzi,(2006) "Context-sensitive ranking," in Proc. SIGMOD, Chicago, IL, USA, pp. 383–394.

#3. R. Agrawal and E. L. Wimmers, (2000) "A framework for expressing and combining preferences," in Proc. SIGMOD, Dallas, TX, USA, pp. 297–306.

#4. H.Andreka, M.Ryan, and P.-Y.Schlobbens,(2002) " Operators and Laws for Combining Preferential Relations,Journal of Logic and Computation"12(1).pp13-53.

#5. A. Arvanitis and G.Koutrika,(2012) "Towards preference-aware relational databases". inProc. IEEE 28th ICDE, Washington, DC, USA,.

#6. A. Arvanitis and G. Koutrika, (2012) "PrefDB: Bringing preferences closer to the DBMS," in Proc. SIGMOD, New York, NY, USA, pp. 665–668.

#7. A. Arvanitis and G. Koutrika, (2012) “PrefDB: Supporting preferences as first-class citizens in relational databases,” Tech. Rep.,

#8. A. Arvanitis and G. Koutrika, (2012) “Towards preference-aware relational databases,” in Proc. IEEE 28th ICDE, Washington, DC, USA, pp. 426–437.

#9. W.-T. Balke, and U. Guntzer, (2004) "Multi-Objective Query Processing for Database Systems".In VLDB.

#10. S. Börzsönyi, D. Kossmann, and K. Stocker, (2001) “The skyline operator,” in Proc. 17th ICDE, Heidelberg, Germany, pp. 421–430.