Conference Proceeding Issue Published in International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333, www.ijtrd.com

Recommendations for Personalized Search for Movies in a Relational Database

[1]P. Malathi and [2]Dr. R. Parameswari,
[1]Research Scholar, [2]Professor,
[1,2]Department of Computer Science, School of computing sciences, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Chennai, India

***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*---Preferences, Database Personalization.*

I. INTRODUCTION

Considering query conditions as hard constraints is the cornerstone of the Boolean database query model. A nonempty answer to a database query is returned only if it satisfies all query conditions. However, this exact match model is often too strict. Imagine, for example, a movie rental application. Asearch for recent movies would return several results making ithard for the user to choose. Taking into account that the userprefers comedies and action movies would help focus herquery to fewer recent movies. On the other hand, if the querycriteria are too restrictive, the query might produce no resultsat all. In this case, it may be better to consider the querycriteria as soft (i.e., preferences) and return results that satisfysome of them. Several approaches to integrating preferencesinto database queries have been proposed and can be roughlydivided into two categories. Plug-in approaches operate on topof the database engineand they typicallytranslate preferences into conventional query constructs. Onthe other hand, native approaches focus on supporting moreefficiently specific queries, such as top-k or skyline queries,by injecting new operators inside the database engine.Motivated by these issues, we have developed PrefDB.

Preference-aware relational system that transparently and efficiently handles queries with preferences. PrefDB employs a preference-aware data model and algebra, where preferences are treated as first-class citizens. We define a preference using a condition on the tuples affected, a scoring function that scores these tuples, and a confidence that show show confident these scores are. In our data model, tuples carry scores with confidences. Our algebra comprises the standard relational operators extended to handle scores andconfidences. For example, the join operator will join twotuples and compute a new score-confidence pair by combiningthe scores and confidences that come with the two tuples. Inaddition, our algebra contains a new operator, prefer, thatevaluates a preference on a relation, i.e., given as inputs arelation and a preference on this relation, prefer outputs therelation with new scores and confidences.PrefDB provides a personalization framework that facilitatesthe enrichment of queries with preference semantics such thatquery results match the specified preferences. It offerssimplified engineering for applications that require preferenceprocessing on top of a relational database. Instead of hardwiringthe preference integration and evaluation strategy intothe application logic .

PrefDBsupports declarative formulation and transparent execution fordifferent types of queries with preferences. At the same time,PrefDB's hybrid implementation pushes preference evaluationcloser to the database than plug-in approaches, enablingoperator-level optimizations, without being as obtrusive asnative ones, and remaining compatible with standard relationalDBMSs. Contributions. The contributions of this work can besummarized as follows:

- A preference-aware relational framework where preferences
- Appear inside queries as first-class citizens and preference
- Evaluation is captured as a special operator that can be
- Combined with other relational operators.

II. REQUIREMENT

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.

III. 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, Todescribe 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.

A. Advantages 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.

IV. ARCHITECTURE

A. 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).

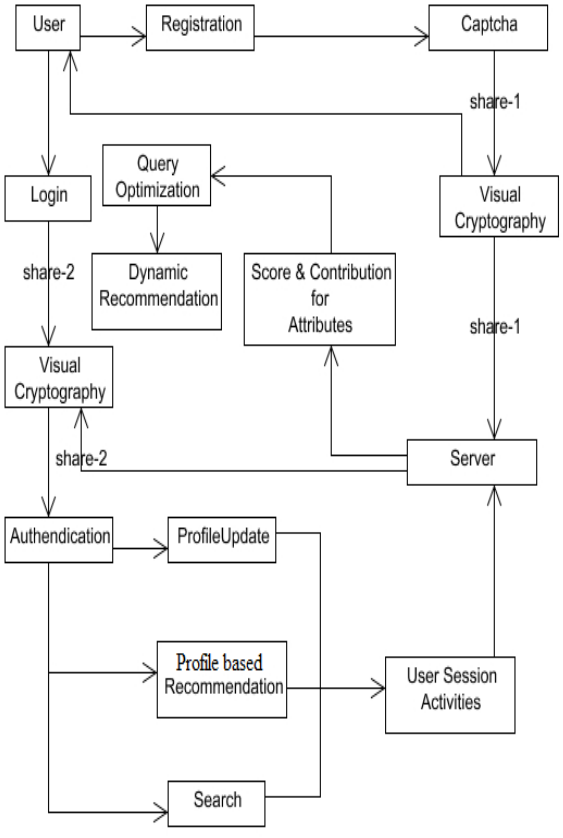

Figure 4.1 System Architecture

V. METHODOLOGY

A. 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.

B. 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 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.

C. 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.

D. 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.

VI. EXPERIMENTAL RESULT AND ANALYSIS

It produces the consequences by comparing the proposed system with the existing system. The movie recommendation based on the user's interest is being compared by some of the constraints.

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.

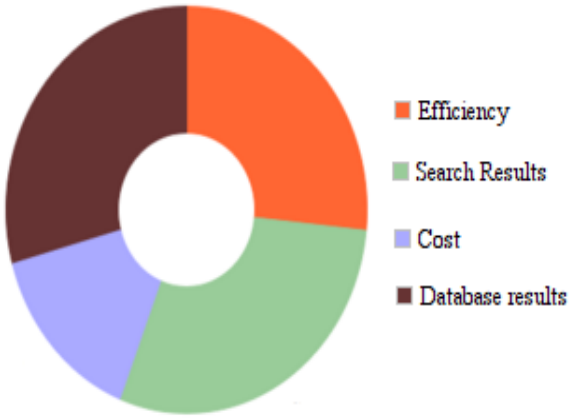

Fig Result analysis

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.

CONCLUSION AND FUTURE ENHANCEMENT

A preference-aware data model is presented where preferences appear as a first-class citizens and preference evaluation is captured as a special 'prefer' operator. The algebraic properties of the new operator is applied to develop a cost-based query optimizations and holistic query processing methods. It adds the advantage by providing the flexibility in handling the different flavors of preferential queries, and also it is closer to the database than plug-in approaches and non-obtrusive to the database engine.

In this work we presented a preference-aware data model where preferences appear as first-class citizens and preference valuation is captured as a special 'prefer' operator. We studied the algebraic properties of the new operator and applied them in order to develop cost-based query optimizations and holistic query processing methods. We presented 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 databaseengine. Our experiments using a prototype systemimplementation demonstrated the performance adventages ofour methods when compared with two variation of a plug-instrategy.In the future, we aim to explore combining the preferoperator with the rank and rank join operators defined in orderto enable early pruning of results based on score or confidenceduring 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.

[11] J. Chomicki, (Dec. 2003) "Preference formulas in relational queries," ACM Trans. Database Syst., vol. 28, no. 4, pp. 427–466,

[12] V. Christophides, D. Plexousakis, (2003) M. Scholl, and S. Tourtounis, "On labeling schemes for the semantic web," in Proc. 12th Int.Conf. WWW, Budapest, Hungary, pp. 544–555.

[13] W. W. Cohen, R. E. Schapire, and Y. Singer, (Jan. 1999) "Learning to order things," J. Artif. Intell. Res., vol. 10, no. 1, pp. 243–270.

[14] R. Fagin, A. Lotem, and M. Naor,(2001) "Optimal aggregation algorithms for middleware," in Proc. 20th PODS, Santa Barbara, CA, USA, pp. 102–113.

[15] P. Georgiadis, I. Kapantaidakis, V. Christophides, E. M. Nguer, and N. Spyratos, (2008) "Efficient rewriting algorithms for preference queries," in Proc. IEEE 24th ICDE, Cancun, Mexico, pp. 1101–1110.

[16] S. Holland, M. Ester, and W. Kießling, (2003) "Preference mining:A novel approach on mining user preferences for personalized applications," in Proc. 7th European Conf. PKDD, Cavtat- Dubrovnik, Croatia, pp. 204–216.

[17] T. Joachims, (2002) "Optimizing search engines using clickthrough data," in Proc. 8th KDD, Edmonton, AB, Canada, pp. 133–142.

[18] W. Kießling, (2002) "Foundations of preferences in database systems," in Proc. 28th Int. Conf. VLDB, Hong Kong, China, pp. 311–322.

[19] W. Kießling and G. Köstler, (2002) "Preference SQL - Design, implementation, experiences," in Proc. 28th Int. Conf. VLDB, Hong Kong, China, pp. 990–1001.

[20] G. Koutrika and Y. E. Ioannidis, (2004) "Personalization of queries in database systems," in Proc. 20th ICDE, Washington, DC, USA, pp. 597–608.

[21] M. Lacroix and P. Lavency, (1987) "Preferences: Putting more knowledge into queries," in Proc. 13th Int. Conf. VLDB, Brighton, U.K., pp. 217–225.

[22] J. Levandoski, M. Mokbel, and M. Khalefa,(2010) "FlexPref: A framework for extensible preference evaluation in database systems," in Proc. IEEE 26th ICDE, Long Beach, CA, USA, pp. 828–839.

[23] C. Li, K. C.-C. Chang, I. F. Ilyas, and S. Song, (2005) "RankSQL: Query algebra and optimization for relational top-k queries," in Proc.SIGMOD, Baltimore, MD, USA, pp. 131–142.

[24] C. Mishra and N. Koudas.(2008) "Stretch 'n' shrink: Resizing queries to user preferences". In SIGMOD, pages 1227–1230,

[25] D. Papadias, Y. Tao, G. Fu, and B. Seeger.(2003) "An optimal and progressive algorithm for skyline queries". In SIGMOD, pages 467–478.

[26] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, (1979) "Access path selection in a relational database management system," in Proc. SIGMOD, pp. 23–34.

[27] K. Stefanidis, M. Drosou, and E. Pitoura.(2010) " Perk: personalized keyword search in relational databases through preferences". In EDBT, pages 585–596.

[28] K. Stefanidis, E. Pitoura, and P. Vassiliadis, (2007) "Adding context to preferences," in Proc. IEEE 23rd ICDE, Istanbul, Turkey, pp. 846–855.

[29] R.Torlone, and P.Ciaccia, (2002) "Which Are My Preferred Item" In Recommendation & Personalization in eCommerce

[30] Y. Yuan, X. Lin, Q. Liu, W. Wang, J.X. Yu, and Q. Zhang, (2005) " Efficient Computation of the Skyline Cube", In VLDB.