ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 IEEE International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st&22ndMarch, Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

Optimizing Evaluated Preference Data in Relational Database Using Query Optimization

Fathima Sanjas .M,Shoba Rani .P

Student, Dept Of Computer Science and Engineering, R.M.D. Engineering College, Thiruvallur Dist., T.N. ,India.

Associate Professor, Dept of Computer Science and Engineering, R.M.D. Engineering College, Thiruvallur Dist., T.N. ,India.

Abstract: Preference-aware queries are needed to be processed closer to the DBMS. Preference database tuples are elaborated from the preference-aware relational data model and the queries with the preferences are processed using an extended algebra. Query optimization strategies are provided for extended query plan based on the set of algebraic properties and cost model. Further illustration of an query execution algorithm that blends preference evaluation with query execution, simultaneously utilizing the native query engine. The framework and methods have been implemented in a prototype system, PrefDB. Transparent and efficient evaluations of preferential queries of a relational DBMS are allowed by PrefDB. This results in experimenting extensive evaluation on two real world data sets which illustrates the feasibility and advantages of the framework. Early pruning of results based on score or confidence during query processing are enabled by combining the prefer operator with the rank and rank join operators.

Keywords: Database Personalization, Personalization search engine, Preferences.

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. A search for recent movies would return several results making it hard for the user to choose. Taking into account that the user prefers comedies and action movies would help focus her query to fewer recent movies. On the other hand, if the query criteria are too restrictive, the query might produce no results at all. In this case, it may be better to consider the query criteria as soft (i.e., preferences) and return results that satisfy some of them.

Several approaches to integrating preferences into database queries have been proposed and can be roughly divided into two categories. **Plug-in approaches** operate on top of the database engine and they typically translate preferences into conventional query constructs. On the other hand, **native approaches** focus on supporting more efficiently specific queries, such as top-k or skyline queries, by injecting new operators inside the database engine.

PrefDB have been developed, 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 preference 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. 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 a '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 preference specified in a query or the expected type of answer.

For processing a query with preferences, we follow a hybrid approach with respect to plug-in and native approaches: we first construct an extended query plan that contains all operators that comprise a query and we 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, we apply 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, we describe an improved version of our processing algorithm (GBU). 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. We provide a detailed experimental evaluation of our query optimization and processing techniques. We evaluate both greedy and dynamic programming plan enumeration methods and compare their performance against two plug-in algorithms. We compare the effectiveness of our optimization methods with the optimal query plan as produced by an exhaustive search algorithm, both in terms of execution and optimization times. Finally, we perform additional sensitivity analysis experiments with respect to the query results size and the preference selectivity.

PrefDB provides a personalization framework that facilitates the enrichment of queries with preference semantics such that query results match the specified preferences. It offers simplified engineering for applications that require preference processing on top of a relational database. Instead of hard-wiring the preference integration and evaluation strategy into the application logic, PrefDB supports declarative formulation and transparent execution for different types of queries with preferences. At the same time, PrefDB's hybrid implementation pushes preference evaluation closer to the database than plug-in approaches, enabling operator-level optimizations, without being as obtrusive as native ones, and remaining compatible with standard relational DBMSs.

II. RELATED WORK

a. Context Sensitive Ranking

Contextual preferences take the form that item i1 is preferred to item i_2 in the context of X. Preferences are provided independently by various sources therefore preferences contain cycle and contradictions. The preferences gradually increasing from various sources are reconciled to create a priori orderings of tuples in an off-line preprocessing step. Few representative orders corresponding to a contexts are saved. Ranked answers are provided when orders and their concern context are used at query time. Contextual preferences provide algorithms for creating orders and processing queries, and present experimental results that show their efficacy and practical utility. For example, a preference might state the choice for Nicole Kidman over Penelope Cruz in drama movies, whereas another preference might choose Penelope Cruz over Nicole Kidman in the context of Spanish dramas.

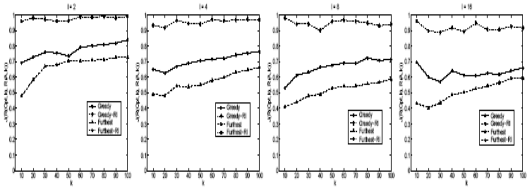


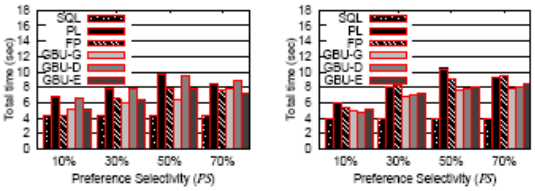
Figure 4: Jaccard coefficient of the top-k results.

b.A Framework for Expressing and combining Preferences

The advent of the World Wide Web has created an explosion in the available on-line information. The time and effort required to sort through them also expands as the range of potential choices expands. This problem could be addressed using a formal framework for expressing and combining user preferences. Preferences can be used to focus search queries and to order the search results.

c.PrefDB: Bringing Preferences Closer to the DBMS

A preference-aware relational query answering system called PrefDB and PrefDB Admin is used for demonstration. The user preferences and the profiles are aggregated and stored into the PrefDB database by the profile manager. By using the graphical tool PrefDB Admin, the preference can be explicitly defined. The preference selection algorithm is applied by the profile manager when the preference is not provided with the input query.



(a) IMDB-1 (b) IMDB-2

d.Towards Preference Aware Relational database.

A Plug-In approach was built on top of the database engine as a straight forward approach for implementing a preference-aware query processing. The expressivity and the performance of queries with preference are affected treating the DBMS as a black box. This argues in pushing the preference aware query processing closer to the DBMS. Therefore this concept illustrates a preference-aware relational data model that extends database tuples with preferences and an extended algebra that captures the essence of processing queries with preferences. The preference model is defined in three dimensions proving the tuples affected the preference scores and the preference credibility. The algebraic property for finer-grained query optimization has the ability to influence through pushing the preference evaluation inside the query plan by query processing strategies. This paper compares the framework to the plug-in approach implementation and has shown the feasibility.

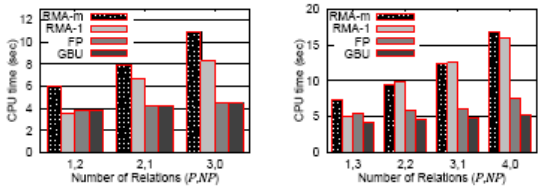
m_id	score	conf
m_1	1.769	2
m_2	0.83	1
m_3	1.847	2
m_4	0.997	1
m_5	1.778	2

(a) $\lambda_{pa}(MOVIES)$

m_id	score	conf
m_1	1.769	2
m_2	1.305	1.5
m_3	2.297	2.5
m_4	1.480	1.5
m_5	1.778	2

(b)$\lambda_{pb}(\lambda_{pa}(MOVIES))$

Fig. 4. Examples using the prefer operator



(a) IMDB-1 (b) IMDB-2

Fig. 11. Processing time vs $|P|/|NP|$

e.Preference Formulas in Relational Queries

The volumes of data presented for the user are reduced through the preferences used for information filtering and extraction. User profiles are kept track to formulate policies to improve and automate the decision making. Preferences are formulated as preference formulas using a simple logical framework. The framework does not impose any restrictions on the preference relations, and allows arbitrary operation and predicate signatures in preference formulas and also makes the straight forward composition of preference relations. Winnow operator parameterized by a preference formula is proposed for embedding of preference formulas into a relational algebra. The formulation of preference queries are made possible through embedding.

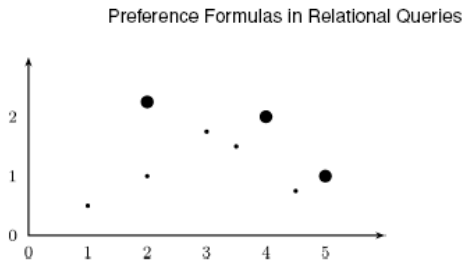


Fig. 5. Two-dimensional skyline.

III THE PREFERENCE DATABASE SYSTEM

Preference database is a prototype system based on the preference and extended relational data model.

OVERVIEW

Figure 1 depicts the system architecture. Two alternative query options are offered by preference DB: Input query consisting of preferences where preferences are specified in a declarative way or a non-preferential query can be enriched by the system with a related preference where the relevant preferences are provided by the profile manager. In preference DB the preferences specified by the users are stored using a visual tool as well as the preference depending on the past queries. Query parser receives both the query option as input.

Preference DB modules are:

a) Profile Manager
b) Query Parser
c) Query Optimizer
d) Execution Engine
e) Admin
f) Weka Tool
g) ARRF File

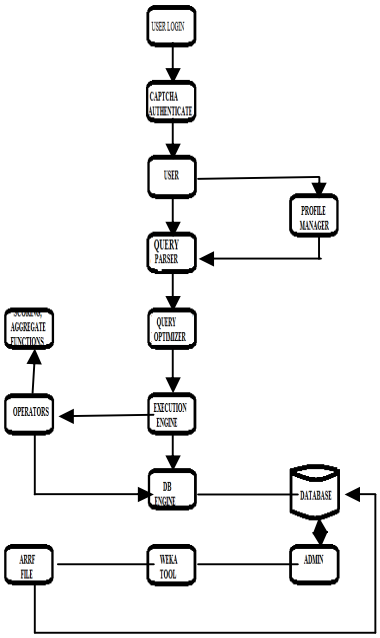


Figure 1: System Architecture

IV QUERY OPTIMIZATION

It is safe to assume to assume that the total cost consists of two parts, the cost related with the non preference operations and the cost related with preference evaluation and score/confidence aggregation on score relations.

The purpose of the step is twofold: (i) The number of tuples that are given as input to the prefer operators are minimized. (ii) Suboptimal plans limiting the search space of alternative plans are pruned. Two plan enumeration algorithms are proposed: (i) Alternative positions of the preferred operators are examined, (ii) The cost of the respective plans are estimated. (iii) Cheapest estimated cost with the plan are selected.

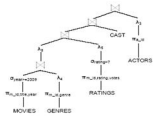


Figure 7. Effect of Rule-Based Query Optimization

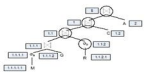


Figure 8. Alternative Positions for ⋈.

V IMPLEMENTATION

Preference database is a prototype system based on the preference and extended relational data model.

Implementation

Implementation of the p-relation and the operators in preference database is discussed below.

Implementing p-relations: Scores and confidence will be query dependent for any database relation and many tuples will remain unaffected by any preferences in a single query. a corresponding score table Rs(Pk,score,conf) is maintained for each base relation Rb affected by preferences in a query, where Pk is a primary

key of Rb. Score table Rs contains only tuples with non-default scores and confidences in order to save space. Each time an operator is executed the score table is updated with new score and confidence.

P-Operators implementation: All extended operators with associated relations with no scored tuples can be minimized to standard relational operators. Preference Database checks if the input score relation is empty before executing each operation; if it is empty then the corresponding operation is forwarded to the native query engine. Otherwise Preference Database uses the operator implementation.

Operators have to be implemented on both base and a score table. For example, tuples do not satisfy the conditions that are filtered out from both relations when a select operator is evaluated. Project operators involve both as well. If the Project operator does not appear in the score table then the projected attribute does not belong to the primary key. In that case the project operator only concerns the base table.

Evaluating a P-Operator is more complicated. First the preference's conditional part is executed on both Rb and Rs. Rs's qualifying tuples have non-default scores and confidence assigned, which is updated with new values. All qualifying tuples of Rb that do not appear in Rs have to be assigned with new non-default scores and confidences and added to Rs. The scoring part is executed on the qualifying tuples of the table Rb, inorder to calculate the new scores for both sets of tuples, then the corresponding aggregate function is called to calculate the final scores.

VI CONCLUSION

In this work a preference-aware data model where preferences appear as first-class citizens and preference evaluation is captured as a special 'prefer' operator has been illustrated. The algebraic properties of the new operator and applied them in order to develop cost-based query optimizations and holistic query processing methods are studied. A framework have been presented 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. A prototype system implementation demonstrated the performance advantages of our methods when compared with two variation of a plug-in strategy are experimented.

REFERENCES

[1] R. Agrawal, R. Rantzau, and E. Terzi. Context-sensitive ranking. In SIGMOD, page 383–394, 2006.

[2] R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences. In SIGMOD, pages 297–306, 2000.

[3]A. Arvanitis and G. Koutrika. PrefDB: Bringing preferences closer to the DBMS. In SIGMOD, pages 665–668, 2012.

[4]A. Arvanitis and G. Koutrika. Towards preference-aware relational databases. In ICDE, pages 426–437, 2012.

[5] J. Chomicki. Preference formulas in relational queries. TODS, 28(4):427–466,2003.

[6] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In PODS, pages 102–113, 2001.

[7] W. Kießling. Foundations of preferences in database systems. In VLDB, pages 311–322, 2002.

[8] G. Koutrika and Y. E. Ioannidis. Personalization of queries in database systems. In ICDE, pages 597–608, 2004.

[9] C. Li, K. C.-C. Chang, I. F. Ilyas, and S. Song. RankSQL: Query algebra and optimization for relational top-k queries. In SIGMOD, pages 131–142, 2005.

[10] K. Stefanidis, E. Pitoura, and P. Vassiliadis. Adding context to preferences. In ICDE, pages 846–855, 2007.