A Review on Preference Based Information Retrieval Models

Manisha Singh
Shri. Ramdeobaba College of Engineering and Management, Computer Science, Nagpur, India

ABSTRACT

Personalization of e-services demands powerful and flexible preference modeling techniques to cope up with the new challenges in information retrieval models. This paper presents a historical review of major milestones in development of various preference based information retrieval models over time covering topics like Pareto preference, Skyline operator, Best Match Only(BMO) query model, Top-k query model, Preference SQL, etc. by constructive analysis through summary and comparison along with merits and drawbacks as applicable for each model. It is concluded by providing practical utility and benefits of these models and a general relation between them. Lastly, some important open fields of research in Preference based Information Retrieval area is jotted down.

KEY WORDS: PREFERENCE BASED IR, SKYLINE OPERATOR, PARETO PREFERENCE, BMO, PREFERENCE SQL.

INTRODUCTION

Most of the database query models work on "hard constraint" principle, i.e. one must specify the requirement in a rather rigid way. If the constraint matches, the result is displayed or else empty set will come. But then there is a "real world"! Natural human instinct is to look for the best match available for our need. But if that is not possible, people are willing to compromise with other alternatives available for the same. Consider a scenario where the user must book a flight or a hotel for a specific region under a certain price range. After all the work of setting the explicit filters of distance, cost, brands, etc. the user notices "flights not found" or "hotels not found" as an output. The user is then asked to refine his query or change the filters to see other options. This is quite a tedious process.

In such scenarios, users will be at ease if the system instead of working on hard constraints, considers the query given by the user as a soft constraint or just a preference. This means considering the query entered by the user as user's first preference. If found, then return the exact match. Else, instead of giving an empty set, return the other possible options. Now consider the reverse scenario where the query is not so well formulated or too generalized. There the user will be flooded with results which might not even be relevant. These are the two classical issues, the infamous "empty set" and "flooding effect" which researchers are trying to solve using preference based information retrieval models to make the system more flexible and efficient to handle real world human behavior.

Many studies and researches have been carried out in the area of preference based information retrieval since decades. This paper tries to outline some of the important aspects and milestones of these studies ranging from 1987 to as recent as possible. Section 2 covers the discussion on 4 different research papers which have proposed different models for preference based query construction and evaluation. Section 2.1 talks about the research paper "Preferences: Putting More Knowledge into Queries"[Lacroix, 1987] which is based on preference clause PREFER. Section 2.2 talks about research paper

ARTICLE INFORMATION

**Corresponding Author: singhmm@rknec.edu*
Received 20th Oct 2020 Accepted after revision 30th Dec 2020
Print ISSN: 0974-6455 Online ISSN: 2321-4007 CODEN: BBRCBA

Thomson Reuters ISI Web of Science Clarivate Analytics USA and Crossref Indexed Journal

NAAS Journal Score 2020 (4.31)
A Society of Science and Nature Publication, Bhopal India 2020. All rights reserved.
Online Contents Available at: http//www.bbrc.in/
Doi: http://dx.doi.org/10.21786/bbrc/13.14/40

"The Skyline Operator"[Borzsony, 2001] based on a very practical requirement of getting the result based on multiple contradicting preferences. Section 2.3 covers the discussion on the research paper "Foundations of Preferences in Database Systems"[Kießling, 2002] which has proved to be one of the most significant research in the field of preference models. And last but not the least, section 2.4 discusses about the research paper "The Preference SQL System – An Overview"[Kießling, 2011] which is an extension to SQL and a practical working model with good query optimization techniques. Section 3 covers conclusion as a comparative study of these discussed models.

2. Preference Models

2.1 Preferences: Putting More Knowledge into Queries: The model given by M. Lacroix and P. Lavency in 1987 is one of the earliest approaches to model preferences in classical (relational) databases [Lacroix, 1987]. It proposes a preference mechanism which is presented as an extension of a language of the Domain Relational Calculus family (DRC) by adding a preference clause PREFER in the existing traditional query language [Lacroix, 1987]. It first evaluates the result without the preference clause. Then it applies the preference clause on the result obtained by previous evaluation. If after applying the preference clause the result set turns to be empty, the preference clause is treated as void. Else, in best case it reduces the cardinality of the result to the most preferred options by the user thus trying to handle both the issues of "empty set" and "flooding effect".

2.1.1 Simple preference clauses

SELECT houses HAVING status = "built complete"
FROM WHICH PREFER THOSE
HAVING roomsize = "2BHK"

Consider a scenario where a new house is to be found to move in. The above query will first select the houses whose construction is completed. And then it will check among those houses which have 2BHK and return the new reduced cardinality result set if applicable, else in case of empty set, it will not restrict it to 2BHK..

2.1.2 Compound preference clauses (multiple preferences)

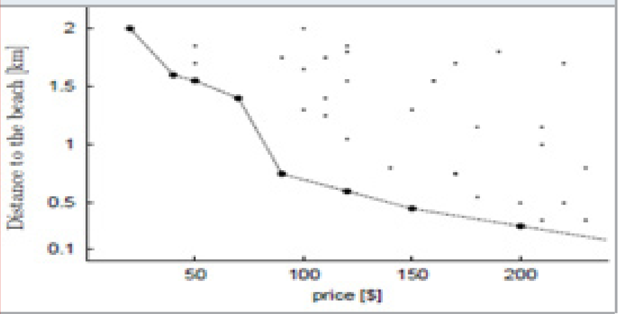
Nested preferences: This is considered as a multi level preference. First preference clause being the highest priority, then on that evaluated result, apply next filter of second preference clause as next priority and so on. For relative importance preferences, repeat the "from which" clause. The result is unrestricted by the preference clause if it is not found on given data set or it makes the result set empty at any stage. The priority is decided as the order in which the preference clauses including "from which" is written.

Equally important preferences: This is similar to the nested preferences except that here all the preference clauses are of same priority. Its functioning is more like "OR" operation. For equally important preferences, repeat the "prefer those" clause.

In case of very large programs one may need the same qualification of preference for different modules which can result in a lot of repetition. This can be avoided using second order constructs provided by this model. Other approaches [C.L. Chang, 1976] have been proposed before to handle preferences but it works on numerical metadata and calculates distance, etc. to find best match. This numerical data is not always available. Also, how that numerical value is assigned plays a very important role in query evaluation and can dominate the result. This issue is resolved in Lacroix's and Lavency's model as they do not rely on numerical metadata. Though this model provides a good insight and approach in dealing with the preferences, but the fact that preference clause is integrated in Domain Relational Calculus (DRC) makes it less efficient because DRC queries are combinational functions over the preferences which make it very complex. Also, in case of complex compound queries the time complexity increases as it doesn't deal with any optimization techniques in proposed approach.

2.2 The Skyline Operator: This model was given by Börzsönyi, Kossmann and Stocker in 2001. It is based on practical real time issue of contrasting preferences. The Skyline is defined as those points which are not dominated by any other point. A point dominates another point if it is as good or better in all dimensions and better in at least one dimension [Borzsony, 2001]. This type of dominance is called Pareto Dominance. Let's take the example given in paper [Borzsony, 2001] where a person needs to travel to Nassau (Bahamas) and is looking for cheap hotels near the beach. These two are contrasting preferences as hotels near beaches are comparatively costly. The Skyline operator will try to filter out interesting hotels from potentially large set of hotels in Nassau by applying filter of min(cost) and min(distance) from beach. The skyline query constructor extends the SQL's SELECT statement by optional SKYLINE OF clause as follows:

Figure 1: Skyline of hotels



SELECT ... FROM ... WHERE ...
GROUP BY ... HAVING ...
SKYLINE OF [distinct] d1 [min | max | diff]...... dm [min | max | diff]..... ORDER BY ...

An advantage of this approach is that only simple modifications to parser and query optimizer is required making integration of the Skyline operator into a traditional SQL query processor extremely simple. Important skyline implementation factor is the transitivity of dominance i.e. if p dominates q and q dominates r, then p also dominates r. One-dimensional Skyline is equivalent to a min, max, or distinct SQL query without a SKYLINE OF clause and can be done easily with the help of sorting. But simple sorting doesn't work on two-dimensional or multi-dimensional skylines. It needs special algorithms for that.

2.3 Foundations of Preferences in Database Systems: This preference model was proposed by Werner Kießling in 2003 [Kießling, 2002]. It works on the principle of strict partial order. It considers preference in terms of "better than" perspective, mathematically which can directly be mapped as strict partial order. Strict partial order can be represented as:

Preference P = (A, <P) on dom(A)
"x <P y" is interpreted as "I like y better than x"

Among all other models available, it can be said that this model is one of the most rich, simple and flexible model semantically. It has proved to be a milestone in building of personalized applications. The preference constructor here covers a wide range with many different criteria as follows:

2.3.1 Base preference constructors

Non-numerical:
POS preference: POS(A, POS-set) : preference given to elements mentioned in POS set.
NEG preference: NEG(A, NEG-set) : don't prefer the elements of NEG set unless the result set is turning empty.
POS/NEG preference: POS/NEG(A, POS-set; NEG-set) : prefer POS set elements, try eliminating NEG set
POS/POS preference: POS/POS(A, POS1-set; POS2-set) : acts as a two level preference
EXPLICIT preference: EXP(A, E-graph) : explicitly specify the preferred elements/criteria

Numerical:
AROUND preference: AROUND(A, z) : prefer values around the given value (min difference)
BETWEEN preference: BETWEEN(A, [low, up]) : prefer values in the given range
LOWEST, HIGHEST preference: LOWEST(A), HIGHEST(A) : prefer the lowest and highest value in given domain
SCORE preference: SCORE(A, f) : uses a function to calculate a score, which can later be used in rank retrieval

If preferred data available, consider that. Else, keep the result unrestricted to avoid returning empty set.

2.3.2 Complex preference constructors

Pareto preference: P1⊗ P2 : both preference are equally important.

Prioritized preference: P1 & P2: preference P1 is first priority followed by preference P2

Numerical preference: rank(P1, P2) : calculate rank based on score, then return top-n results (only with score)

Apart from these there is also aggregating preference constructors like intersection, disjoint and union preferences.

BMO Query Model: The exact match query model adapted by SQL doesn't necessarily hold in real world. Thus, preference works on Best Match Only (BMO) query model which is a match-making between wishes and reality. Return perfect matches if they exist, else, deliver best alternatives, but never worse objects (effect of discarding non-maximal values on the fly). In BMO, query relaxation is implicitly applied and the behavior is always non-monotonous depending on the quality of data rather than quantity.

Efficiency and optimization issues are not directly addressed in this paper. But, it does provide a backbone for optimization approaches like divide-and-conquer by laying the foundation in the form of decomposition of Pareto preferences into '+' and '♦', which in turn can be decomposed further. Merit of this model holds on Pareto accumulation that gives user the best-match automatically without any overload of explicit query refinement.

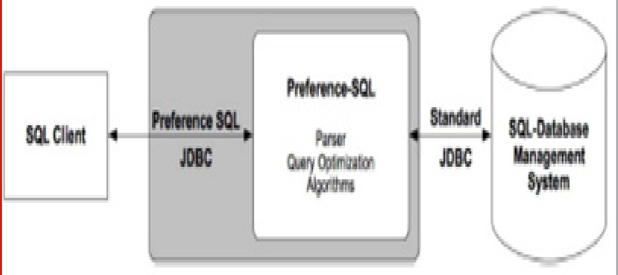
2.4 The Preference SQL System: This preference model was proposed by Endres, Kießling, & Wenzel in 2011 [Kießling, 2011]. Preference SQL is a declarative extension of standard SQL by strict partial order preferences, behaving like soft constraints under the BMO query model, discussed in previous model. By default, Preference SQL has implicit SV-Semantics but it is still under development and enhancement (as of 2019). The preference constructor here is derived from previous model with an addition of more advanced functionalities. The schematic query structure of Preference SQL is as follows:

```
SELECT ... <selection>
FROM ... <table_references>
WHERE ... <hard_conditions>
PREFERRING ... <soft_conditions>
GROUPING ... <attribute_list>
TOP ... <k>
BUT ONLY ... <but_only_condition>
GROUP BY ... <attribute_list>
HAVING ... <hard_conditions>
ORDER BY ... <attribute_list>
LIMIT ... <n>
```

Statements select, from, where, group by, having, and order by are the standard SQL keywords. Along with that, special preference based keywords are integrated into it. The evaluation order first groups Preference Selection then Top-k Interface. Here But Only is a hard selection (after-filter) which helps in reducing the novel "flooding effect". HAVING is a hard selection too for groups using grouping attributes or aggregate functions.

The PreferenceSQL prototype is based on a Java-Middleware (JDBC) containing PSQL-JDBC-Driver (Server) and Preference SQL (Parser, Optimizer, Algorithms). It is a declarative extension of SQL by preferences. The preference query optimizer performs algebraic transformations of preference relational algebra as well as cost-based algorithm selection e.g. Hexagon algorithm for efficient Pareto / skyline [Kießling, 2007]. As PreferenceSQL system lies only on the server, it becomes easily maintainable and simple extension of PreferenceSQL without changing the standard SQL component of database systems along with no updates of the clients as it is in the form of JDBC middleware. But it does face higher runtime or even worse performance issues as more time is needed for computations based on middleware. More efficient algorithms and join approaches are needed for overcoming this issue.

Figure 2: System architecture of Preference SQL



CONCLUSION

This paper presents an overview of the key models for preference based information retrieval along with merits and areas of improvement as applicable for each. All the discussed models are built up on SQL, letting integration with standard database query model possible. Pareto preference has played a major role in all 4 discussed models. Apart from that, Lacroix & Lavency's model incorporates prior preference, the skyline model includes skyline preference and both rearmost models encompasses prior and basis preferences. Lacroix & Lavency's model is not as efficient and user friendly as the latter three models. In terms of practical usage, the skyline model and the Preference SQL have a strong utilization aspect. BNL-style algorithm is used in Lacroix & Lavency's model. The latter 3 models cover different algorithms like BNL, SFS, SaLSa, LESS, Scalagon, etc. The Lacroix & Lavency's model was only a prolog (prototype) whereas the skyline model has an implementation model built on PostgreSQL.

The model given by Kießling [Kießling, 2002] is one of the richest models which have laid a foundation for various implementations based on preference based information retrieval. The PreferenceSQL is an empirical adaption of Kießling's model. The well known implementations of Endres's models are Preference SQL, an implementation based on MS SQL Server, PostgreSQL and EXASolution. Though many studies and researches have been going on in the field of Preference based information retrieval, but it still lags behind compared to exact match standard SQL models. The current day-to-day applications having exact match query model as base can be attempted based on preference based information retrieval for efficiency and least repetitive query refinement from user's end. A very crucial open area of research in this domain includes optimization techniques for complex preferences.

REFERENCES

Borzsony, Kossmann, & Stocker. "The Skyline operator", in proceedings 17th International Conference on Data Engineering, Heidelberg, Germany, 2001.

C. L. Chang. "The Deduce - A deductive query language for relational data base", Pattern Recognition and Artificial Intelligence, Academic Press, 1976.

Eder, Wei. "Evaluation of skyline algorithms in PostgreSQL", in Proceedings of the 2009 International Database Engineering & Applications Symposium on – IDEA, 2009.

Kießling, Endres, Wenzel . "The PreferenceSQL System - An Overview", in IEEE Computer Society Technical Committee on Data Engineering, 2011.

Kießling, Kostler. "Preference SQL - Design, Implementation, Experiences", in VLDB'02: Proceedings of the 28th International Conference on Very Large Database, 2001.

Kießling, Preisinger. "The Hexagon Algorithm for Pareto Preference Queries", in VLDB, Vienna, Austria, 2007.

Kießling. "Foundations of Preferences in Database Systems", in proceedings of the 28th International Conference on Very Large Databases, Hong Kong, China, 2002.

Lacroix and Lavency. "Preferences: Putting More Knowledge Into Queries", in proceedings of the 13th VLDB Conference, Brighton, 1987.

Rahangdale, Agrawal. "Information Extraction Using Discourse Analysis from Newswires", International Journal of Information Technology Convergence and services (IJITCS), 2014.