A NEW TOP-K CONDITIONAL XML PREFERENCE QUERIES

Shaikhah Alhazmi[1] and Mourad Ykhlef[2]

Information Systems Department, King Saud University, Kingdom of Saudi Arabia

ABSTRACT

Preference querying technology is a very important issue in a variety of applications ranging from e-commerce to personalized search engines. Most of recent research works have been dedicated to this topic in the Artificial Intelligence and Database fields. Several formalisms allowing preference reasoning and specification have been proposed in the Artificial Intelligence domain. On the other hand, in the Database field the interest has been focused mainly in extending standard Structured Query Language (SQL) and also eXtensible Markup Language (XML) with preference facilities in order to provide personalized query answering. More precisely, the interest in the database context focuses on the notion of Top-k preference query and on the development of efficient methods for evaluating these queries. A Top-k preference query returns k data tuples which are the most preferred according to the user's preferences. Of course, Top-k preference query answering is closely dependent on the particular preference model underlying the semantics of the operators responsible for selecting the best tuples. In this paper, we consider the Conditional Preference queries (CP-queries) where preferences are specified by a set of rules expressed in a logical formalism. We introduce Top-k conditional preference queries (Top-k CP-queries), and the operators BestK-Match and Best-Match for evaluating these queries will be presented.

KEYWORDS

XML Query Processing, Preference-Based Retrieval, Conditional Preference, Top-K preference Query, Query Evaluation

1. INTRODUCTION

The handling of preferences is becoming an increasingly important issue in present-day information systems [1]. Among others, preferences are used for information filtering and extraction to reduce the volume of data presented to the user. There are some motivations for such an issue [2], [3], [4]. First, it has appeared to be desirable to offer more expressive query languages that can be more faithful to what a user intends to say. Second, the introduction of preferences in queries provides a basis for rank-ordering the retrieved items, which is especially valuable in case of large sets of items satisfying a query. Third, on the contrary, a classical query may also have an empty set of answers, while a relaxed (and thus less restrictive) version of the query might be matched by some items in the database.

Preferences can be of many kinds: qualitative or quantitative, conditional, and positive or negative [5]; for example, the preference is qualitative as in ("I like A better than B"), or quantitative as in ("I like A at level 10 and B at level 11") and the conditional preference as in ("If A happens, then I prefer B to C"), positive preference as in ("I like A, and I like B even more than A"), or negative as in ("I don't like A, and I really don't like B"). In order to fulfill the user's preferences to the best possible degree, one must support reasoning about qualitative statements of preference that all of us express in our everyday activities. Several models for representation and reasoning about such statements have been proposed in AI, of which the most studied to date is the CP-nets model [6], where CP stands for Conditional Preference. CP-nets are devoted to

DOI : 10.5121/ijaia.2014.5505

reasoning about qualitative (and possibly conditional) statements of preference where each statement expresses preference over the values of a single property of the outcomes, such as "*I prefer a red dress to a yellow dress if the shoe is black*", or "*If the car is convertible, I prefer a soft top to a hard top*". The core notion exploited by the CP-nets model is ceteris paribus (all else being equal) interpretation of the statements [7].

On the other hand, XML is widely used as a base technology for knowledge management within enterprises and dissemination of data on the Web (like e.g., product catalogues), because it allows to organize and handle semi-structured data. A collection of XML documents is viewed as a forest of node labeled trees. The data generally can be queried on both structure and content using advanced retrieval languages such as XPath [8] or XQuery [9]. User queries over XML usually are structured to express the user's information needs as hard constraints with *exact-match semantic*, where user's need can be satisfied completely or not at all delivering exactly the information if they are there and otherwise reject the user's request. The deficiency here is users could face the *empty-result problem* which is disappointing to them [4]. The other extreme deficiency, if the query is built by disjunctive sub-queries then instead of getting an empty answer the user overloaded with lots of mostly irrelevant information. This is the notation of *flooding-effect problem* [4]. The exact-match query model can become a real nuisance for users. However, XML only supports hard constraints with *exact-match semantic*. In contrast, mostly people include preferences in their decisions both in their daily lives as well as in business. But the very nature of preferences is different from hard constraints. Preferences are like soft constraints, requiring *a match-making process* instead: If my favorite choice is available, I will take it. Otherwise, instead of getting nothing, I am open to alternatives, but show me only the best ones according to their relevance to the query which is more appropriate than exact-match queries. Therefore, it would be useful to provide an operator which can produce a set of tuples which is reasonably interesting to the user and also he can control the size of the preferred set (Top-k querying).

In this paper, our contribution is to make a step towards bridging the gap between expressiveness capabilities of XML technology and attractiveness of preferences elicitation for developing preferences elicitation through proposing an approach of Top-k Conditional Preference Queries (Top-k CP-queries) and the operators for evaluating these queries which are *BestK-Match* and *Best-Match* which follows the lines of the Best Match Only (BMO) [10], [11], [12].

Users can specify their conditional preferences in the form of (IF-THEN) statements, each preference separated by the connective AND. The users could provide an optional non-negative integer *k* to control the result set size. If *k* is provided; the query is evaluated using *BestK-Match* operator and returns the top *k* best matches, if *k* is not given; the query is evaluated using *Best-Match* operator and returns the set of most preferred tuples. Both operators perform the dominance tests between each two tuples in the dataset. We use the graphical representation, Better-Than graph, which can be used for specifying tuples relations between each other (e.g. an arrow from *a* to *b* means that *a* better-than or dominates *b*). By this graph, we can elicit the level for each tuple; the lowest level gives the tuple that is the most match or favorable for the given user preferences.

This paper is organized as follows. Section 2 summarizes some related works. Section 3 briefly presents the formalism for specifying and reasoning with conditional preferences over XML. We present the semantic of the proposed approach and the operators *BestK-Match* and *Best-Match* in section 4. Finally, we conclude the paper and discuss our future work in section 5.

2. RELATED WORK

The research literature on preference modeling, reasoning and eliciting is extensive [13]. The approach of CP-Nets [6], [14] uses a very simple graphical model which captures user's qualitative conditional preference over tuples, under a ceteris paribus semantics. The approach of TCP-Nets [15] generalizes the CP-Nets by introducing the ability of expressing absolute and relative importance of object attributes. Reference [16] Introduced an approach uses a logical framework for expressing conditional preference statements. It generalizes the CP-Nets, by relaxing the ceteris paribus semantics and allowing more expressiveness, and is orthogonal to TCP-Nets. In the database area, the problem of enhancing well-known query languages with preference features has been tackled in several recent and important works in the area. A pioneer work concerns the *skyline* operator introduced in [17].

In reference [11], a simple logical framework is proposed for expressing preferences by preference formulae. These formulae are incorporated into the Relational Algebra and into SQL, through the operator winnow parameterized by a preference formula. Reference [18] Introduced Preference SQL which extends SQL with some built-in base preference constructors and a pareto accumulation and a prioritized accumulation constructors. The optimizer uses an efficient rewriting procedure which transforms preference queries into standard SQL queries.

The work presented in [19] also proposes a bridge between the treatment of preferences in the AI and DB communities, by transforming the TCP-Net queries of [16] into preference database queries. No algorithm for selecting the best tuples has been proposed neither in [19] and further work (to the best of our knowledge).

On the other hand, the topic of preference query evaluation has been extensively studied in the literature, in the context of *pareto* and *skyline* queries. In [17], the basic BNL (block-nested loop) algorithm has been introduced for evaluating skyline queries. In [20], the algorithm BNL+ for Pareto query evaluation was introduced. In this more general setting, the domains of the attribute values are partially ordered. In [21] the authors proposed the algorithm BNL++ for pareto query evaluation in a particular case, where the ordering over the attribute domains is a weak order. This implies nice properties in the better-than graph, which are not verified in the general setting of the algorithm BNL+ of [20]. Due to these nice properties, BNL++ is able to identify some important pruning conditions in the better-than graph, which results in a far better performance when compared to the BNL and BNL+ algorithms. Further improvements of the BNL algorithm, in the context of *pareto* queries evaluation, were achieved in [20] (algorithms BBS+, SDC and SDC+). The algorithm BNL* which proposed in [22] for evaluating the *Select-Best* operator in the context of CP-queries follows the lines of the BNL algorithm and uses the technique introduced in [23] for transitive closure evaluation.

In reference [24], [25], [26] the Top-k queries have been introduced in a quantitative preference model setting, that is, where preference between tuples is expressed by a score function defined over the dataset. The *Top-k dominating queries* have been introduced in [27] as an extension of the skyline queries of [17] which were originally designed to return the most preferred tuples, without any user control on the size of the result. A *Top-k dominating query* returns the *k* tuples which dominated the maximum amount of tuples in the database. This concept is orthogonal to the skyline and pareto queries, as well as to our CP-queries. In [27] the authors propose the algorithm BBS (Branch and Bound Skyline) based on nearest-neighbor search on multidimensional access methods for evaluating the Top-k dominating queries. In [28] and [29] a set of algorithms are introduced for Top-k dominating queries which significantly outperforms the algorithm BBS of [24].

3. XML CONDITIONAL PREFERENCES QUERIES

This section aims to present preferences queries formalism over XML with support of conditional preferences.

Beyond simple lookups modern database retrieval which enables users to express certain preferences. Such preferences state what a user likes and preference query processing attempts to retrieve all best possible matches in a cooperative fashion avoiding empty result sets. In relational databases such preferences are specified over data values. But, since in XML retrieval also the structure of the document often carries semantics described by a DTD or XML schema, user preferences can also be specified over the structure.

A Top-k conditional preferences query (*Top-k CP-query*) incorporates the usual hard constraints declared inside square brackets as well as soft constraints specified by a set of preferences rules (CP-rules). The general syntax of our preferences query is:

CREATE PREFERENCE AS <*pref-name*>
<*list-of-CP-rules*>

… /<xml-document-instance>/… [hard-constraints]
ACCORDING TO PREFERENCES *(k, pref-name)*

Definition 1 (Top-k CP-query) A Top-k CP-query is preference query containing the ACCORDING TO PREFERENCES clause. The parameter k is a non-negative integer and is optional. If k is not provided, the query is evaluated using *Best-Match* operator and returns the set of most preferred tuples, without specifying the size of this set. If $k \geq 0$ is given, then the evaluation uses the *BestK-Match* operator.

Our framework is obtained by extending the XML capabilities with the proposed preferences query block which is given above. The *list-of-CP-rules* inside CREATE PREFERENCE command is a list of CP-rules declared in the form of IF … THEN …, each rule separated by the connective AND.

Example 1 Let us suppose we are given a XML file named *Movies* storing information about some movies. The XML schema of *Movies* as follows:

```
<xs:element name="tilte" type="xs:string"/>
<xs:element name="director">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="first-name"  type="xs:string"/>
      <xs:element name="last-name"  type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="category" type="xs:string"/>
<xs:element name="year" type="xs:string"/>
```

And the input data is represented as XML file as shown below:

```
<?xml version="1.0"?>
<movies name="Movies">
   <movie movieid="1" >
      <title>The Family</title>
      <director>
         < first-name >John</ first-name >
```

```
        <last-name>Trump</ last-name >
    </director>
    <category>Drama</ category >
    <year>90</ year >
  </movie>
  <movie movieid ="2" >
    <title>X-File</title>
    <director>
        < first-name >James</ first-name >
        <last-name>Brose</ last-name >
    </director>
    <category>Drama</ category >
    <year>80</ year >
  </movie>
  <movie movieid ="3" >
    <title>The Robot</title>
    <director>
        < first-name > Stephen</ first-name >
        <last-name> Crane</ last-name >
    </director>
    <category>Drama</ category >
    <year>80</ year >
  </movie>
  <movie movieid ="4" >
    <title>Adams Family</title>
    <director>
        < first-name > Stephen </ first-name >
        <last-name> Crane </ last-name >
    </director>
    <category>Comedy</ category >
    <year>80</ year >
  </movie>
  <movie movieid ="5" >
    <title>The Toy</title>
    <director>
        <first-name >John</ first-name >
        <last-name>Trump</ last-name >
    </director>
    <category>Comedy</ category >
    <year>80</ year >
  </movie>
  <movie movieid ="6" >
    <title>Black Cat</title>
    <director>
        < first-name >James</ first-name >
        <last-name>Brose</ last-name >
    </director>
    <category>Action</ category >
    <year>00</ year >
  </movie>
</movies>
```

Suppose the following statements express my preferences about movies: (1) I prefer those movies produced in the 90's, if both belongs to the same category; (2) For the movies produced in the 80's I prefer drama ones; (3) For the comedies produced in the 80's I prefer those directed by John Trump; (4) I prefer those movies produced in and after 2000, if both directed by the same director.

I would like to list 2 titles of the movies which most fulfil my wishes among those stored in the XML file, but I don't want thriller movies to figure in my list.

The corresponding Top-k CP-query associated to the example is:

CREATE PREFERENCE AS *myPref*
IF <*same category*> THEN *year = 90* AND
IF *year = 80* THEN *category = "Drama"* AND
IF *category = "Comedy"* and *year = 80* THEN *director = "John Trump"* AND
IF <*same director*> THEN *year >= 2000*

/movies / movie [category != "Thriller"] / title
ACCORDING TO PREFERENCES (*2 , myPref*)

In this query, the hard constraint is category != *"Thriller"* and the soft constraints are given by the rules *myPref*, denoted as: P1: same category → 90's; P2: 80's → drama; P3: comedy + 80's → John Trump; P4: same director → 00's.

Note that the parameter *k* = 2 appear inside the clause ACCORDING TO PREFERENCES since it is part of the soft constraint expressed by the preferences rules. In general, if *k* is not specified the most preferred tuples will be returned.

A naive execution plan is as follows:

1. Select the data from XML file that satisfy the hard constraints, and then convert it to relation.
2. Select the tuples that match the given preferences from the relation.
3. Return preferred results.

The evaluation of the *BestK-Match* involves the evaluation of the most preferred tuples first. This task is achieved by the *Best-Match* operator. The semantics of this operator is the same as the *winnow* operator [10] of SQL query returning the tuples which are not dominated by others. In the next section we present the semantic of our proposed approach of *Top-k CP-queries* and the operators *BestK-Match* and *Best-Match* for evaluating these queries.

4. TOP-K QUERIES SEMANTIC

In this section we aim to represent the semantic of the approach of Top-k Conditional Preference Queries (Top-k CP-queries). We introduce our proposed approach focusing on the new operators BestK-Match and Best-Match.

The approach of *Top-k CP-queries* is the first proposed solution for aspects of XML preference query processing that return Top-k answers. It consists of two operators that evaluate CP-queries, which are *Best-Match* and *BestK-Match*. The *Best-Match operator* returns the most preferred tuples satisfying a given CP-queries which are dominate the others. The execution of *Best-Match* on XML data consists in finding the set of tuples verifying the soft constraints. If this set is non-empty only these tuples are returned. Otherwise, all tuples that satisfied the hard constraints are returned. In the other hand, the *BestK-Match operator* enables user control on the size of the answer set. The *BestK-match operator* returns the first top *k* tuples satisfying a given CP-queries.

A Top-k queries processing consists of the followed steps:

- Convert the XML file to a relation after satisfied the hard constraints.
- Testing the tuples dominance and construct a Better-Than Graph (BTG).
- Assign level to each node in the graph.
- Find the best match answers.
- Return the top *k* answers.

Definition 2 (Dominance Test) Let us suppose x_1 and x_2 are tuples over a relation *R*, and let *G* be the BTG graph of *R* that will be generated using dominance test. We say that:

- x_1 *dominates* x_2, if x_1 satisfy one preference or more over x_2. Denoted as: $x_1 >_{Pi} x_2$. Add arrow from x_1 to x_2 in *G* as shown in Figure 1.a.
- x_1 *equal* x_2, if both satisfy the same preference P_i. Denoted as: $x_1 =_{Pi} x_2$. Add nodes x_1 and x_2 to *G* without arrow between them as shown in Figure 1.b.
- x_1 *incomparable with* x_2, if there is no preference relation between x_1 and x_2. Denoted as: $x_1 <> x_2$. Do nothing in *G*.

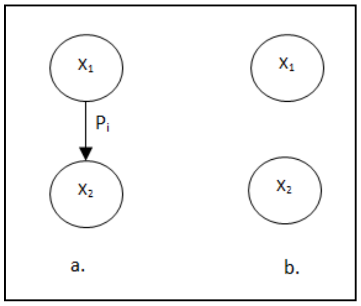

Figure 1. Dominance Illustration

The following example aims to illustrate our dominance notation of Top-k conditional preferences queries.

Example 1 (continued) To test the tuples dominance, convert the XML file *Movies* to a relation after satisfying the hard constraint as represented in Table 1. Then, test the tuples dominance and construct the BTG graph as shown in Figure 2:

- M1 *dominates* M2 and M3 according to P1. Add arrow from node M1 to M2 and M3.
- M1 *incomparable with* M4, M5 and M6. Do nothing.
- M2 *equal* M3 according to P2. Do nothing.
- M2 *dominates* M4 and M5 according to P2. Add arrow from node M2 to M4 and M5.
- M2 *dominated by* M6 according to P4. Add arrow from node M6 to M2.
- M3 *dominates* M4 and M5 according to P2. Add arrow from M3 to M4 and M5.
- M3 *incomparable with* M6. Do nothing.
- M4 *dominated by* M5 according to P3. Add arrow from M5 to M4.
- M4 *incomparable with* M6. Do nothing.
- M5 *incomparable with* M6. Do nothing.

Table 1. The *Movies* relation

	Title	**Director**		**Category**	**Year**
		first-name	**Last-name**		
M1	The Family	John	Trump	Drama	90
M2	X-File	James	Brose	Drama	80
M3	The Robot	Stephen	Crane	Drama	80
M4	Adams Family	Stephen	Crane	Comedy	80
M5	The Toy	John	Trump	Comedy	80
M6	Black Cat	James	Brose	Action	00

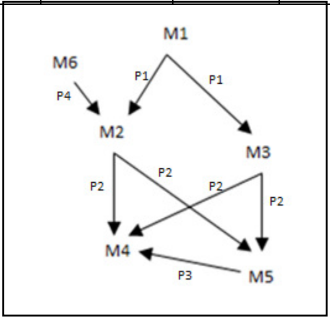
$$\begin{tikzcd}
M6 \arrow[dr, "P4"'] & & M1 \arrow[dl, "P1"'] \arrow[drr, "P1"] & \\
& M2 \arrow[d, "P2"'] \arrow[drr, "P2"] & & M3 \arrow[dll, "P2"'] \arrow[d, "P2"] \\
& M4 & & M5 \arrow[ll, "P3"]
\end{tikzcd}$$

Figure 2. Better-Than Graph for relation *Movies*

The time complexity of dominance testing process is *((n(n-1))/2)m* in worse case, where *n* is the number of tuples of a relation and *m* is the number of preferences.

Definition 3 (BestK-Match(k,R)) an operator which returns the set of first *k* tuples having the minimum number of tuples that dominated them in the relation *R*. Besides this operator, our approach includes also the *Best-Match* operator.

Definition 4 (Best-Match(R)) an operator which returns the most preferred tuples (those which are not dominated by others in the relation *R*). In order to define the semantics of the *BestK-match* operator, we introduce the notion of level.

Definition 5 (Levels) Given a BTG graph *G*, and *a* is node in *G*, we say level(*a*) (denoted *l(a)*) is *x* if the input arrows into *a* are *x*.

level(a) = no. of a's input arrows in better-than graph

As a special case, we consider *a* as best match tuple when *level(a)* = 0.
Example 1 (continued) Based on better-than graph as in Figure 2, we have: *l(M1)* = *l(M6)* = 0, *l(M3)* = 1, *l(M2)* = *l(M5)* = 2, and *l(M4)* = 3.

BestK-Match(k, R) returns the set of *k* tuples with the smallest levels. However, *Best-Match*(*R*) returns the most preferred tuples. For example, *BestK-Match(3, Movies)* = {M1, M6, M3}, *BestK-Match(4, Movies)* = {M1, M6, M3, M2}. We remark that *l(M2)*=*l(M5)*= 2, but M5 is not considered in the answer, since it comes after M2 in the Table 1. Note that *BestK-Match(0, Movies)* = ∅ and *Best-Match*(*Movies*) = {M1, M6} which is the set of the most preferred tuples, those which are not dominated by others.

For instance, the Top-k CP-query of Example 1 executed over relation *Movies* of Table 1 is evaluated by *BestK-Match* operator and returns {M1, M6} according to the BTG graph illustrated in Figure 2.

5. CONCLUSION AND FUTURE WORK

In this paper, we introduced the conditional preference queries (CP-queries), which allow users to specify preferences over the values of attributes depending on the values of other attributes in the tuples. We proposed our approach of Top-k CP-queries allowing user preference to be taken into account in the query answering process, and the operators *BestK-Match* and *Best-Match* for evaluating these queries. A lot of work remains to do; we are intending to design a quantitative model of preferences, that users can specify their preferences both as likings (positive) and disliking (negative) form with different important levels.

REFERENCES

[1] Wang, H., Zhou, X., Chen, W. and Ma, P. 2013. Top-k Retrieval Using Conditional Preference Networks. CIKM'12, Maui, HI, USA.

[2] A. Hadjali, S. Kaci, H. Prade. Database Preference Queries – A Possibilistic Logic Approach with Symbolic Priorities, Proc. of the 5th Symposium on the Foundations of Information and Knowledge Systems (FoIKS'08), Pisa, Italy, 2008.

[3] P.Bosc, A.Hadjali, O.Pivert. An approach to competitive conditional fuzzy preferences in database flexible querying. Intelligent Systems, 2008. IS '08. 4th International IEEE Conference , vol.2, no., pp.11-2,11-7, 6-8 Sept. 2008.

[4] Peti, L., de Amo, S., Roncancio, C., and Labbé, C. Top-k Context-Aware Queries on Streams. DEXA 2012, Part I, LNCS 7446, pp. 397–411.

[5] S.Bistarelli, M.Silvia Pini, F.Rossi, and B.Venable. Positive and Negative Preferences. Pisa, Italy.

[6] C. Boutilier, R. Brafman, H. Hoos, and D.Poole. Cp-nets: A tool for representing and reasoning about conditional ceteris paribus preference statements. Journal of Artificial Intelligence Research, 21:135–191, 2004.

[7] C. Domshlak, S. Prestwich, F. Rossi, K. B. Venable and T. Walsh. Hard and soft constraints for reasoning about qualitative conditional preferences. Journal of Heuristics, pages 263-285, 2006.

[8] XPATH, [Online]. Available: http://en.wikipedia.org/wiki/XPath.

[9] XQuery, [Online]. Available: http://en.wikipedia.org/wiki/XQuery

[10] W. Kießling. Foundations of preferences in database systems. Proceedings of 28th International Conference on Very Large Data Bases, 2002, Hong Kong, China, pages 311–322, 2002.

[11] J. Chomicki. Preference formulas in relational queries. ACM Transactions on Database Systems, pages 427–466, 2003.

[12] J. Chomicki. Semantic optimization techniques for preference queries. Information Systems, 32, 5:670–684, 2007.

[13] Lukasiewicz, T., Martinez, M., Simari, G., and Tifrea-Marciuska, O. 2014. Preference-Based Query Answering in Probabilistic Datalog+/–Ontologies. Berlin Heidelberg 2014.

[14] Dubois, D., Prade, H. and Touazi, F. 2013. Conditional Preference-Nets, Possibilistic Logic, and the Transitivity of Priorities. Switzerland.

[15] Brafman, R., Domshlak, C., and Shimony, S. 2006. On graphical modeling of preference and importance. Journal of Artificial Intelligence Research 25, 389–424.

[16] Wilson, N. 2004. Extending cp-nets with stronger conditional preference statements. In AAAI. 735–741.

[17] Borzsonyi, S., Kossmann, D., and Stocker, K. 2001. The skyline operator. In Proc. 17th International Conference on Data Engineering (ICDE 2001), Germany. 412–430.

[18] Kießling, W. and Köstler, G. 2002. Preference sql - design, implementation, experiences. In VLDB. 990–1001.

[19] Endres, M. and Kießling, W. 2006. Transformation of tcp-net queries into preference database queries. In Proc. Of the ECAI 2006 Multidisciplinary Workshop on Advances in Preference Handling. 23–30.

[20] Chan, C.-Y., Eng, P.-K., and Tan, K.-L. 2005a. Efficient processing of skyline queries with partially-ordered domains. In ICDE. 190–191.

[21] Preisinger, T., Kießling, W., and Endres, M. 2006. The bnl++ algorithm for evaluating pareto preference queries. In 2nd Multidisciplinary Workshop on Advances in Preference Handling, ECAI 2006.

[22] De Amo, S. and Ribeiro, M. R. 2009. Cpref-sql: A query language supporting conditional preferences. In 24th Annual ACM Symposium on Applied Computing (ACM SAC), March 2009, Honolulu, Hawaii, USA. 1573–1577.

[23] Agrawal, R., Borgida, A., and Jagadish, H. 1989. Efficient management of transitive relationships in large data and knowledge bases. In Proc. ACM SIGMOD Internacional Conference on Management of Data. 253–262.

[24] Hristidis, V., Koudas, N., and Papakonstantinou, Y. 2001. Prefer: A system for the efficient execution of multiparametric ranked queries. In Proceedings of ACM SIGMOD, Santa Barbara, CA, USA. 259–270.

[25] Loyer, Y., Sadoun, I., and Zeitouni, K. 2013. Preferences Chain Guided Search and Ranking Refinement. DEXA 2013, Part I, LNCS 8055, pp. 9–24.

[26] Jabbour, S., Sais, L., and Salhi, Y. 2013. The Top-k Frequent Closed Itemset Mining Using Top-k SAT Problem. ECML PKDD 2013, Part III, LNAI 8190, pp. 403–418.

[27] D. Papadias, Y. Tao, G. Fu, B. Seeger. Progressive Skyline Computation in Database Systems, ACM Transactions on Database Systems, Vol.30, No.1, pp.41-82, 2005.

[28] Yiu, M. L. and Mamoulis, N. 2007. Efficient processing of Top-k dominating queries on multi-dimensional data. In Proceedings of VLDB 2007, Vienna, Austria. 483–494.

[29] Yiu, M. L. and Mamoulis, N. 2009. Multi-dimensional Top-k dominating queries. VLDB Journal 18(3), 695–718.