Logical Foundations of Preference Queries

Jan Chomicki
University at Buffalo
chomicki@buffalo.edu

Abstract

The notion of preference plays an increasing role in today's information systems. In particular, preferences are used to specify which query answers are the best from the user's point of view. In this article, we discuss the work done over the last 10 years in the context of preference queries to relational database systems. We focus on preferences that are specified logically. *We show how such preferences can be embedded into relational query languages. By separating the preferences from the query, preference-specific query evaluation and optimization techniques can be formulated and preference modification studied. We conclude with an outline of future prospects for preference research.*

1 Introduction

Preference is one of the dimensions of personalization. Two different users querying a database using the same query may expect different answers because they may use different, implicit criteria of which answers are the *best* and the *most preferred*. For example, when purchasing a book online one user may be primarily interested in obtaining the lowest price available, while another may be concerned with the reliability of the vendor.

We discuss here a formal framework in which user preferences are formulated explicitly using first-order logic. Relying on that representation, many issues germane to preferences like composition, elicitation, or revision have been studied independently of queries. Moreover, it has been shown that preferences and queries can be combined in a clean fashion, yielding *preference queries*. In the context of such queries, classical database issues like query evaluation and optimization have been revisited, yielding new, preference-specific techniques.

Preferences have been studied for a long time in decision theory and philosophy [11, 14]. The interest in preferences in artificial intelligence [4] and databases [24] is more recent.

The following mock car-shopping dialogue illustrates some aspects of preferences and preference queries addressed in this paper:

Maggie (salesperson): What kind of car do you prefer?

Fred (customer): The newer the better, if it is the same make. And cheap, too.

Maggie: Which is more important for you: the age or the price?

Fred: The age, definitely.

Maggie: Those are the best cars, according to your preferences, that we have in stock.

Fred: Wait...it better be a BMW.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

First, Maggie elicits Fred's preferences involving age and price. Second, she gets Fred to commit to a *prioritized composition* of those preferences (age more important than price). Third, Maggie runs a *preference query* returning the best cars, according to Fred's preferences. Fourth, Fred changes his mind and *revises* his preferences.

2 Preference relations

We view preferences as *binary relations* between objects of the same kind. *Attribute* preference relations relate attribute values (typically, constants), *tuple* preference relations relate tuples in the same relation. To refer to the former we use the symbol $'' >''$, to the latter, the symbol $'' \succ''$, both with subscripts if necessary. If $x \succ y$, we say that x is *is preferred to* y (x *is better than* y, x *dominates* y).

Finite preference relations are defined by enumerating their elements. To define infinite preference relations – common in the presence of infinite domains – we use *first-order logic formulas.*

Example 1: Throughout this paper, we will repeatedly use the database relation $Car(Make, Year, Price)$ and the following preference relations:

$$p >_{price} p' \equiv p < p'$$
$$y >_{age} y' \equiv y > y'$$
$$(m,y,p) \succ_{C_1} (m',y',p') \equiv m = m' \wedge (y > y' \wedge p \leq p' \vee y \geq y' \wedge p < p').$$

The first two are attribute preference relations, the third, a tuple preference relation.

Typically, logic formulas defining preferences (*preference formulas*) contain constants, variables, comparison operators (like $'' >''$) and Boolean connectives. Thus, it is possible to check whether one tuple is preferred to another by substituting tuple attribute values into the preference formula and computing the truth value of the resulting formula. Such preferences – based only on the contents of the tuples being compared – are called *intrinsic* [8], in contrast to *extrinsic* preferences which may also refer to the contents of database relations. Also, intrinsic preference formulas usually admit quantifier elimination, and thus quantifiers are not needed in preference specification.

We also make use of several *derived* binary relations:

- non-strict attribute preference: $x \geq_A x' \equiv x >_A x' \vee x = x'$;
- non-strict tuple preference: $t \succeq_C t' \equiv t \succ_C t' \vee t = t'$;
- tuple indifference: $t \sim_C t' \equiv t \not\succ_C t' \wedge t' \not\succ_C t$.

2.1 Strict partial orders

Here we list some typical properties of binary relations. A binary relation R is

- *irreflexive* if $\forall x\ (\neg R(x,x))$,
- *transitive* if $\forall x,y,z\ (R(x,y) \wedge R(y,z) \rightarrow R(x,z))$,
- *connected* if $\forall x,y\ (R(x,y) \vee R(y,x) \vee x = y)$,
- a *strict partial order* (SPO) if it is irreflexive and transitive,

- a *weak order* (WO) if it is an SPO such that $\forall x, y, z\ (R(x,y) \rightarrow R(x,z) \vee R(z,y))$,
- *a total order* if it is a connected SPO.

Commonly, preference relations are required to be SPOs. It is obvious that irreflexivity should hold: preferring an object over itself seems to violate the basic intuitions behind preference. But transitivity is debatable. On one hand, it captures the *rationality* of preferences [11, 12]. On the other, transitivity is sometimes violated by preference aggregation in voting scenarios [25]. We note that quantifier elimination provides procedures for checking whether a binary relation satisfies the desirable order properties.

Preferences are often captured using numeric-valued *scoring functions*. Such functions constitute special cases of preference relations. Indeed, a scoring function f represents a preference relation $\succ_f$ such that

$$x \succ_f y \equiv f(x) > f(y).$$

It is easy to see that preference relations represented by scoring functions are *weak orders*. Suppose $x \succ_f y$. Then $f(x) > f(y)$. So for every z, $f(x) > f(z)$ or $f(z) > f(y)$, and thus $x \succ_f z$ or $z \succ_f y$. The WO property has an important consequence: preference relations that are SPOs but not weak orders *cannot be represented using scoring functions*. Such preferences are common: the preference relation $\succ_{C_1}$ in Example 1, which is a *skyline* preference relation, falls into that category.

2.2 Combining preferences

There are many different ways in which preferences can be combined. We discuss *preference composition* and *preference accumulation*. Both of them are defined *logically*: if the preference relations being combined are defined by logic formulas, so is the resulting preference relation.

Preference composition combines preference relations about objects of the *same kind*: constants or tuples from the same database relation. The result is a preference relation of that kind. Therefore, the "dimensionality" of preference is not increased. The most common composition operators [1] are:

- union: $x \succ y \equiv x \succ_1 y \ \vee \ x \succ_2 y$, similarly intersection;
- prioritized composition: $x \succ y \equiv x \succ_1 y \vee (y \not\succ_1 x \wedge x \succ_2 y)$;
- Pareto composition: $x \succ y \equiv (x \succ_1 y \wedge y \not\succ_2 x) \vee (x \succ_2 y \wedge y \not\succ_1 x)$.

One of the potential applications of preference composition is *preference revision* [9]. This is further described in Section 6.

Example 2: The notation $X \succ Y$ means that $\forall x \in X, y \in Y\ (x \succ y)$. Consider two preference relations $\succ_1$ and $\succ_2$: *wine* $\succ_1 \{$*tea*,*coffee*$\} \succ_1$ *juice* and $\{$*tea*,*juice*$\} \succ_2$ *coffee* $\succ_2$ *wine*. Then the prioritized composition $\succ_3$ of $\succ_1$ and $\succ_2$ is *wine* $\succ_3$ *tea* $\succ_3$ *coffee* $\succ_3$ *juice* and the Pareto composition $\succ_4$ of $\succ_1$ and $\succ_2$ is *tea* $\succ_4 \{$*coffee*,*juice*$\}$. According to the indifference relation corresponding to $\succ_4$, wine and all the other drinks are mutually indifferent.

Preference accumulation combines preferences over *objects* to yield preferences over Cartesian products of objects. In this way, the "dimensionality" of preference is increased. The most common accumulation operators are:

- prioritized accumulation $\succ = \succ_1 \& \succ_2$: $(x_1, x_2) \succ (y_1, y_2) \equiv x_1 \succ_1 y_1 \vee (x_1 = y_1 \wedge x_2 \succ_2 y_2)$, and
- Pareto accumulation $\succ = \succ_1 \otimes \succ_2$: $(x_1, x_2) \succ (y_1, y_2) \equiv (x_1 \succ_1 y_1 \wedge x_2 \succeq_2 y_2) \vee (x_1 \succeq_1 y_1 \wedge x_2 \succ_2 y_2)$.

We note that both prioritized and Pareto accumulation are *associative* (Pareto is also commutative).

[1] The preference relation which is the result of the composition will be denoted by $\succ$.

3 Skylines

Among all preference relations, *skyline* preference relations, defined using Pareto accumulation, have been the most extensively studied [6]. Given attribute preference relations $>_{A_1}, \ldots, >_{A_n}$, the *skyline* preference relation $\succ$ is defined as:

$$\succ = >_{A_1} \otimes >_{A_2} \otimes \cdots \otimes >_{A_n} .$$

Unfolding the definition of Pareto accumulation:

$$(x_1, \ldots, x_n) \succ (y_1, \ldots, y_n) \equiv \bigwedge_i x_i \geq_{A_i} y_i \wedge \bigvee_i x_i >_{A_i} y_i. \tag{1}$$

If we fix the attribute preferences to be the standard orderings of the reals, then we get a *Euclidean* skyline preference relation.

Example 3: Given a two-dimensional Euclidean skyline preference relation $\succ$:

$$(x_1, x_2) \succ (y_1, y_2) \equiv x_1 \geq y_1 \wedge x_2 > y_2 \vee x_1 > y_1 \wedge x_2 \geq y_2$$

and a finite set of points S in the 2-dimensional space, the skyline consists of $\succ$-maximal elements of S. Figure 1 shows an example skyline (the skyline points are solid black).

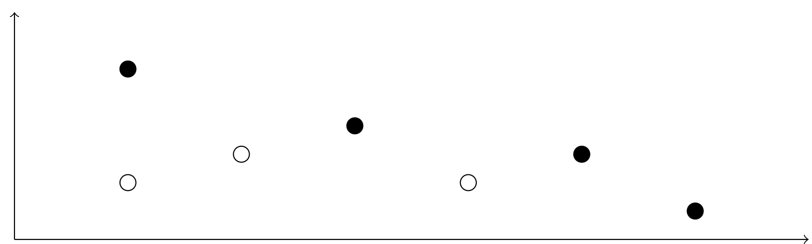

Figure 1: Two-dimensional skyline

Skyline preference relations and skylines enjoy several properties that make them attractive in the context of preference queries:

- *invariance*: a skyline preference relation is unaffected by *scaling* or *shifting* in any dimension;
- *universality*: a skyline consists of maxima of *monotonic* scoring functions.

We note that usually skyline preference relations are not weak orders.

Example 4: In two-dimensional Euclidean space:

$$(3,0) \succ (2,0), (3,0) \not\succ (1,1), (1,1) \not\succ (2,0).$$

Actually, the skyline preference relations in [6] admit a form slightly more general than the formula in Equation 1. The preference relation is defined not only in terms of attribute preference relations but also attribute equality. This achieves the effect of GROUP-BY and can be conceptually viewed as defining multiple skylines.

Example 5: Considering Example 1, the preference relation $\succ_{C_1}$ is defined as

$$(m,y,p) \succ_{C_1} (m',y',p') \equiv m = m' \wedge (y > y' \wedge p \leq p' \vee y \geq y' \wedge p < p').$$

Only the cars of the same make can be compared (are in the same group). In the SQL extension proposed in [6], the above preference relation is expressed as

```
SKYLINE Make DIFF, Year MAX, Price MIN
```

Skylines have also been generalized to *p-skylines*, defined using not only Pareto but also prioritized accumulation [21]. In this way priorities between different attribute preferences are captured.

We conclude this section by noting that all the preferences that can be expressed in the framework of Kießling [8] can also be expressed using intrinsic first-order logic formulas.

4 Preference queries

The most common kind of preference query has been formalized as the *winnow* operator ω [8], also called *BMO* [8] and *Best* [26]. The operator retrieves all the best (undominated) tuples from a given relation.

Formally, given a preference relation $\succ$ and a database relation r:

$$\omega_{\succ}(r) = \{t \in r \mid \neg\exists t' \in r.\, t' \succ t\}.$$

If a preference relation $\succ_C$ is defined using a formula C, then we write $\omega_C(r)$, instead of $\omega_{\succ_C}(r)$.

It is clear that a skyline is the result of computing winnow under the skyline preference relation. This simple observation has important consequences: the techniques for evaluating and optimizing winnow apply immediately to skylines.

Winnow seamlessly integrates with the operators of the relational algebra. In fact, winnow can be expressed in relational algebra and provides the nonmonotonic functionality equivalent to that of set difference [8].

Example 6: Consider the preference relation $\succ_{C_1}$ from Example 1, and the following relation instance $\{t_1 : (mazda, 2009, 20K), t_2 : (ford, 2008, 15K), t_3 : (ford, 2007, 15K)\}$ Winnow returns the tuples t_1 and t_2. The tuple t_3 is dominated by t_2.

There are two algorithms that compute winnow for arbitrary SPO preference relations: BNL [6] and SFS [7]. Those algorithms were first proposed in the context of skylines but they only require irreflexivity and transitivity of preferences [8]. Many algorithms for computing skylines and their variants have been proposed in the literature: we mention three influential ones: BBS [23], LESS [13], and Salsa [2].

5 Query optimization

A major advantage of the logic-based approach to preference queries is that *rewrite-based query optimization* is done in a natural and clean way. For example, the algebraic laws involving winnow are formulated analogously to the well-known laws of relational algebra [8, 8, 15]. However, often the laws do not hold unconditionally. Consider commuting winnow and selection. We have $\sigma_C(\omega_\gamma(r)) = \omega_\gamma(\sigma_C(r))$ for every r if the formula $\forall t_1, t_2.[C(t_2) \wedge \gamma(t_1, t_2) \Rightarrow C(t_1)]$ is valid.

Example 7: Under the preference relation $\succ_{C_1}$ from Example 1, the selection $\sigma_{Price<20K}$ commutes with ω_{C_1} but $\sigma_{Price>20K}$ does not.

Other laws involving winnow were studied in [8, 15].

Semantic query optimization (query optimization using integrity constraints) also fits in very well here. As shown in [10], the information about integrity constraints can be used to eliminate redundant occurrences of winnow and make more efficient computation of winnow possible. We say that ω_C is *redundant w.r.t.* a set of integrity constraints F if $w_C(r) = r$ for all r satisfying F. Now ω_C is *redundant w.r.t.* F iff F implies the formula $\forall t_1, t_2.\ R(t_1) \wedge R(t_2) \Rightarrow t_1 \sim_C t_2$. The latter formula is a *constraint-generating dependency*. The properties of such dependencies were studied in [3].

6 Prospects for preferences

Preference modification Preferences are rarely static. Desires and needs fluctuate, priorities change. Several different preference modifications operators have been considered in the literature. In *preference revision* [9], a *revising* preference relation $\succ_0$ is *composed* with the original preference relation $\succ$ using one of the composition operators: union, prioritized or Pareto composition. Under certain restrictions on preference relations, the composition eliminates preference conflicts. Moreover, to guarantee SPO properties the result of the composition is transitively closed. Preference revision is suitable in scenarios where new preference information augmenting or contradicting the existing one comes to light. In *preference contraction* [20], the new information consists of a *contractor* relation CON. The revised preference relation is now a maximal SPO subset of $\succ$ disjoint with CON. Contraction is appropriate if preferences are being cancelled or withdrawn. In *substitution* [5], the new information is a set of *indifference* pairs. The paired objects are supposed to become mutually substitutable. This is possible if additional preferences are added so that the paired objects have the same sets of dominating and dominated objects. Clearly, having a general framework encompassing the above approaches (and other that can be envisioned) would be useful.

Preference elicitation. While preference formulas concisely and precisely capture preferences, they are not easy to construct. Instead of requiring that the user build a preference formula from scratch, one could imagine a step-by-step preference specification process in which the user could provide additional feedback. Recent work suggests that the feedback in the form of good or bad objects may be used in a variety of ways. In [16], the user analyzes the result of a skyline query and labels some of the skyline objects as *superior* (should be in the skyline) or *inferior* (should not be in the skyline). Then the attribute preference relations are revised in such a way that the revised skyline query is guaranteed to return the superior objects and not to return inferior objects. Under the same model, [21] propose that it is the skyline query expression that needs to be revised by replacing some occurrences of Pareto accumulation operators by the occurrences of prioritized accumulation. In this way relative priorities of attribute preference relations are captured. It would be natural to consider other kinds of user feedback, for example answers to dominance queries: *"does a dominate b?"* In general, it is a considerable challenge to uncover the preferences underlying the kinds of preference-related information available, for example, in social networks.

Preference and uncertainty. Several papers have studied the evaluation of skyline queries over uncertain (probabilistic) data [22]. It would be interesting to consider the same problem for more general variants of skyline queries. [19] study the problem of skyline querying in the presence of nulls. There, a tuple t_1 dominates a tuple t_2 if t_1 is *known* to be better than t_2 in some dimension and *not known* to be worse in any other dimension. What is the right logic for defining such preference relations?

Preferences over sets. In some applications it is natural to consider preferences over *sets of objects*. Those sets may be *homogenous* (consisting of objects of the same type) or *heterogenous* (consisting of objects of different types). Preferences over homogenous sets arise, for example, in committee selection, or employee or student recruiting. To address this topic, [28] (generalizing the approach of [1]) propose a two-layered approach. In the first layer, *set profiles*, which are tuples of *features*, are defined. An example feature is an aggregate value of some attribute of the set, e.g., `SUM`. In the second layer, tuple preferences among profiles are specified. Queries return the best profiles – such profiles correspond to the best subsets of a given set. The algorithmic challenge is to improve on the brute-force enumeration of all the subsets. Preferences over heterogenous sets arise in product configuration. An example product is a vacation package, consisting of hotel, plane, and rental reservations. The best products are computed using the skyline semantics [27]. The algorithmic challenge is to avoid the materialization of all possible products.

Preference networks. We note that an influential approach to preference queries and personalization [17] is based on the notion of *preference network* in which numeric scores are associated with query conditions to capture their degree of satisfaction. It would be interesting to develop logical semantics for that approach.

References

[1] Maxim Binshtok, Ronen I. Brafman, Carmel Domshlak, and Solomon Eyal Shimony. Generic Preferences over Subsets of Structured Objects. *Journal of Artificial Intelligence Research*, 34:133–164, 2009.

[2] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. Efficient Sort-Based Skyline Evaluation. *ACM Transactions on Database Systems*, 33(4), 2008.

[3] M. Baudinet, J. Chomicki, and P. Wolper. Constraint-Generating Dependencies. *Journal of Computer and System Sciences*, 59:94–115, 1999. Preliminary version in ICDT'95.

[4] R. I. Brafman and C. Domshlak. Preference Handling – An Introductory Tutorial. *AI Magazine*, 30(1):58–86, 2009.

[5] W-T. Balke, U. Güntzer, and W. Siberski. Exploiting Indifference for Customization of Partial Order Skylines. In *International Database Engineering and Applications Symposium (IDEAS)*, pages 80–88, 2006.

[6] S. Börzsönyi, D. Kossmann, and K. Stocker. The Skyline Operator. In *IEEE International Conference on Data Engineering (ICDE)*, pages 421–430, 2001.

[7] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with Presorting. In *IEEE International Conference on Data Engineering (ICDE)*, pages 717–719, 2003.

[8] J. Chomicki. Preference Formulas in Relational Queries. *ACM Transactions on Database Systems*, 28(4):427–466, December 2003.

[9] J. Chomicki. Database Querying under Changing Preferences. *Annals of Mathematics and Artificial Intelligence*, 50(1-2):79–109, 2007.

[10] J. Chomicki. Semantic Optimization Techniques for Preference Queries. *Information Systems*, 32(5):660–674, 2007.

[11] P. C. Fishburn. *Utility Theory for Decision Making*. Wiley & Sons, 1970.

[12] P. C. Fishburn. Preference Structures and their Numerical Representations. *Theoretical Computer Science*, 217:359–383, 1999.

[13] P. Godfrey, R. Shipley, and J. Gryz. Algorithms and Analyses for Maximal Vector Computation. *VLDB Journal*, 16:5–28, 2007.

[14] S. O. Hansson. Preference Logic. In D. Gabbay, editor, *Handbook of Philosophical Logic*, volume 4. Kluwer, 2001.

[15] Bernd Hafenrichter and Werner Kießling. Optimization of Relational Preference Queries. In *Australasian Database Conference (ADC)*, pages 175–184, 2005.

[16] Bin Jiang, Jian Pei, Xuemin Lin, David W. Cheung, and Jiawei Han. Mining Preferences from Superior and Inferior Examples. In *KDD*, pages 390–398, 2008.

[17] G. Koutrika and Y. E. Ioannidis. Personalizing Queries Based on Networks of Composite Preferences. *ACM Transactions on Database Systems*, 35(2), 2010.

[18] W. Kießling. Foundations of Preferences in Database Systems. In *International Conference on Very Large Data Bases (VLDB)*, pages 311–322, 2002.

[19] Mohamed E. Khalefa, Mohamed F. Mokbel, and Justin J. Levandoski. Skyline Query Processing for Incomplete Data. In *IEEE International Conference on Data Engineering (ICDE)*, pages 556–565, 2008.

[20] D. Mindolin and J. Chomicki. Contracting Preferences for Database Applications. *Artificial Intelligence*, 175(7-8):1092–1121, May 2011. Special issue on Representing, Processing, and Learning Preferences.

[21] D. Mindolin and J. Chomicki. Preference Elicitation in Prioritized Skyline Queries. *VLDB Journal*, 20(2):157–182, 2011. Special issue: selected papers from VLDB 2009.

[22] Jian Pei, Bin Jiang, Xuemin Lin, and Yidong Yuan. Probabilistic Skylines on Uncertain Data. In *VLDB*, pages 15–26, 2007.

[23] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive Skyline Computation in Database Systems. *ACM Transactions on Database Systems*, 30(1):41–82, 2005.

[24] K. Stefanidis, G. Koutrika, and E. Pitoura. A Survey on Representation, Composition and Application of Preferences in Database Systems. *ACM Transactions on Database Systems*, 36(4), 2011.

[25] A. K. Sen and P. K. Pattanaik. Necessary and Sufficient Conditions for Rational Choice under Majority Decision. *Journal of Economic Theory*, 1:178–202, 1969.

[26] R. Torlone and P. Ciaccia. Which Are My Preferred Items? In *Workshop on Recommendation and Personalization in E-Commerce*, May 2002.

[27] Qian Wan, Raymond Chi-Wing Wong, Ihab F. Ilyas, M. Tamer Özsu, and Yu Peng. Creating competitive products. *PVLDB*, 2(1):898–909, 2009.

[28] X. Zhang and J. Chomicki. Preference Queries over Sets. In *IEEE International Conference on Data Engineering (ICDE)*, pages 1019–1030, April 2011.