Preference Queries in Relational Databases

Jan Chomicki

University at Buffalo

http://www.cse.buffalo.edu/~chomicki

Querying with Preferences

Find the best answers to a query, instead of all the answers.

Why:

- too many answers (objectively or subjectively)
- only the best answers will do.

"Find the lowest price for this book on the Web...

... but also keep in mind my preference for `amazon.com`*."*

Scoring functions

An approach grounded in utility theory:

1. construct a real-valued function u such that:

$$t_1 \succ t_2 \equiv u(t_1) > u(t_2)$$

2. return the answers that maximize u in the given instance r.

+ can be implemented using SQL3 user-defined functions [Agrawal&Wimmers,2000; Hristidis et al., 2001]
+ provides an ordering of all the answers
− scoring functions need to be hand-crafted for every input
− not expressive enough: weak order preference relations.

Non-existence of utility functions

	Title	*Vendor*	*Price*
t_1	The Flanders Panel	`amazon.com`	$14.75
t_2	The Flanders Panel	`fatbrain.com`	$13.50
t_3	The Flanders Panel	`bn.com`	$18.80
t_4	Green Guide: Greece	`bn.com`	$17.30

The set of constraints

$$\{u(t_2) > u(t_1) > u(t_3), u(t_4) = u(t_1), u(t_4) = u(t_2)\}$$

is unsatisfiable.

Preference relations as logical formulas

Define preferences explicitly in a logical language:

$$(i, v, p) \succ_1 (i', v', p') \equiv i = i' \wedge p < p'.$$

Query language embedding: new winnow operator returning the tuples in the given instance that are not dominated by any other tuple in the instance.

	Title	*Vendor*	*Price*
t_1	The Flanders Panel	`amazon.com`	\$14.75
t_2	The Flanders Panel	`fatbrain.com`	\$13.50
t_3	The Flanders Panel	`bn.com`	\$18.80
t_4	Green Guide: Greece	`bn.com`	\$17.30

Advantages of the logical approach

Generality:

$>$ unrestricted preference relations

$>$ logical composition of preference relations

$>$ tight integration with the query language

$>$ algebraic and semantic query optimization

$>$ variety of implementation options.

User-friendliness:

$>$ preference formulas are more natural and intuitive than utility functions.

Plan of the talk

1. Preference formulas.
2. Winnow and its properties.
3. Optimization of preference queries.
4. Logical composition of preferences.
5. Extrinsic preferences.
6. Skyline and linear optimization queries.
7. Related and future work.

Definitions

Preference relation: a binary relation $\succ$ between the tuples of a given relation.

Preference formula: a first-order formula defining a preference relation.

Intrinsic preference formula: the definition uses only built-in predicates.

Typical properties of preference relations: irreflexivity, and transitivity, can be effectively checked for intrinsic preference formulas with $=, \neq, <, >, \leq, \geq$.

Winnow

Given a preference relation $\succ$ defined using a preference formula C:

$$\omega_C(r) = \{t \in r | \neg \exists t' \in r.\ t' \succ t\}.$$

Example (*"preference for* `amazon`*"*):

$$(i_1, v_1, p_1) \succ_2 (i_2, v_2, p_2) \equiv i_1 = i_2 \wedge v_1 = '\mathtt{amazon}' \wedge v_2 \neq '\mathtt{amazon}'.$$

	Title	*Vendor*	*Price*
t_1	The Flanders Panel	`amazon.com`	\$14.75
t_2	The Flanders Panel	`fatbrain.com`	\$13.50
t_3	The Flanders Panel	`bn.com`	\$18.80
t_4	Green Guide: Greece	`bn.com`	\$17.30

Some properties of winnow

Winnow can be formulated in relational algebra:

$$Book(I,V,P) - \pi_{I,V,P}(Book(I,V,P) \underset{I=I',P>P'}{\bowtie} Book(I',V',P')).$$

Nonemptiness:

If C defines a preference relation which is a strict partial order and r is finite, then $\omega_C(r)$ is nonempty.

Containment:

If $C_1(t_1,t_2) \Rightarrow C_2(t_1,t_2)$, then for all instances r, $\omega_{C_2}(r) \subseteq \omega_{C_1}(r)$.

Algebraic laws

Commutativity with selection:

If $(C_1(t_2) \wedge C_2(t_1, t_2)) \Rightarrow C_1(t_1)$, then for every r

$$\sigma_{C_1}(\omega_{C_2}(r)) = \omega_{C_2}(\sigma_{C_1}(r)).$$

Distributivity over Cartesian product: For every r_1 and r_2

$$\omega_C(r_1 \times r_2) = \omega_C(r_1) \times r_2.$$

Commutativity of winnow: If $C_1(t_1, t_2) \Rightarrow C_2(t_1, t_2)$ and $\succ_{C_1}$ and $\succ_{C_2}$ are strict partial orders, then for all finite instances r:

$$\omega_{C_1}(\omega_{C_2}(r)) = \omega_{C_2}(\omega_{C_1}(r)) = \omega_{C_2}(r).$$

Semantic query optimization

Using information about integrity constraints to:

- eliminate redundant occurrences of winnow.
- make more efficient computation of winnow possible.

Eliminating redundancy: Given a set of integrity constraints F, ω_C is redundant w.r.t. F iff F implies the following integrity constraint:

$$\forall t_1.\forall t_2.\ [R(t_1) \wedge R(t_2)] \Rightarrow t_1 \not\succ_C t_2 \wedge t_2 \not\succ_C t_1.$$

Integrity constraints

Constraint-generating dependencies (CGDs) [Baudinet, Chomicki, Wolper, 1995]:

$$\forall t_1. \dots \forall t_n.\ [R(t_1) \wedge \cdots \wedge R(t_n) \wedge \gamma(t_1, \dots t_n)] \Rightarrow \gamma'(t_1, \dots t_n).$$

Implication is decidable for CGDs.

Logical composition of preferences

Single-dimensional:

- Boolean operations, e.g.,:

$$x\ (\succ_1 \cap \succ_2)\ y \equiv x \succ_1 y \wedge x \succ_2 y.$$

- prioritized composition:

$$x\ (\succ_1 \rhd \succ_2)\ y \equiv x \succ_1 y \vee (y \not\succ_1 x \wedge x \succ_2 y).$$

Multi-dimensional:

- lexicographic and Pareto composition

Scoring functions: composition much more problematic

Extrinsic preferences

Preference relation depends not only on the components of the tuples being compared but also on other factors:

- the presence or absence of other tuples in the database
- computed or aggregate values.

Solution: winnow + SQL.

Preference for a lower total cost of a book (including shipping and handling).

Vendor	*SH*
amazon.com	$6.99
fatbrain.com	$3.99
bn.com	$5.99

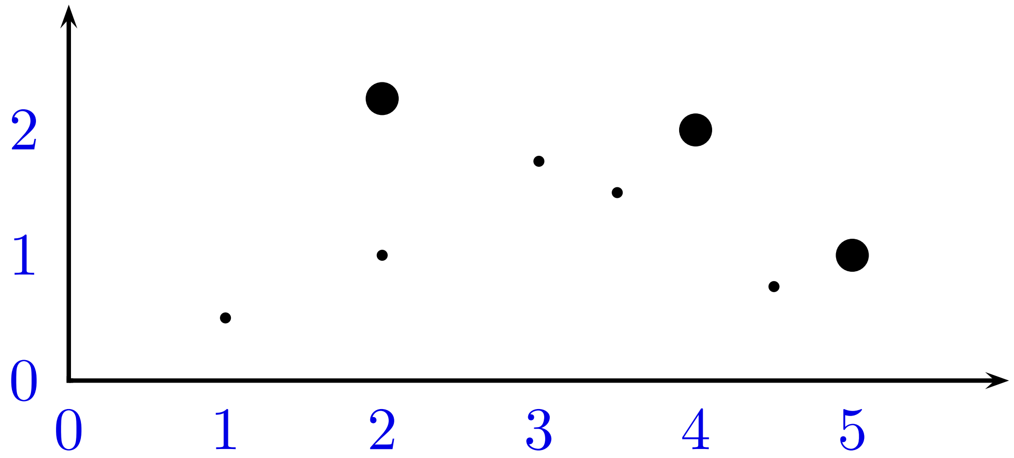
Apply winnow to the following view:

```
CREATE VIEW TotalCost(Title, Vendor, Cost) AS
SELECT Book.Title, Book.Vendor, Book.Price + SHCosts.SH
FROM Book, SHCosts WHERE Book.Vendor = SHCosts.Vendor
```

Problem: Cartesian products in views.

Skyline queries

Find all the tuples that are not dominated by any other tuples in all dimensions [Börzsönyi, Kossmann, Stocker, 2001] (Pareto composition).



Properties:

- skyline: maximums of monotone scoring functions.

Skyline in SQL

```
SELECT ... FROM ... WHERE ...
  GROUP BY ... HAVING ...
  SKYLINE OF A1 [MIN | MAX | DIFF]
             ...
             An [MIN | MAX | DIFF]
```

A `DIFF` attribute indicates that tuples with different values of that attribute are incomparable.

Special case of winnow

Skyline:

`SKYLINE OF A DIFF, B MAX, C MIN`

The corresponding preference formula:

$$(x, y, z) \succ (x', y', z') \equiv x = x' \wedge y \geq y' \wedge z \leq z' \wedge (y > y' \vee z < z').$$

Skyline using BNL

Block-nested-loops algorithm [Börzsönyi, Kossmann, Stocker, 2001]:

1. initialize the window W and the temporary file F to empty;
2. repeat the following until the input is empty:
3. for every tuple t in the input:
   - t is dominated by a tuple in $W \Rightarrow$ ignore t,
   - t dominates some tuples in $W \Rightarrow$ eliminate the dominated tuples and insert t into W,
   - t is incomparable with all tuples in $W \Rightarrow$ insert t into W (if there is room), otherwise add t to F;
4. output the tuples from W that were added there when F was empty,
5. make F the input, clear F.

Skyline using SFS

Sort-Filter-Skyline [Chomicki, Godfrey, Gryz, Liang, 2003]:

1. sort the input file according to any monotone scoring function;
2. initialize the window W and the temporary file F to empty;
3. repeat the following until the input is empty:
4. for every tuple t in the input:
   - t is dominated by a tuple in $W \Rightarrow$ ignore t,
   - t is incomparable with all tuples in $W \Rightarrow$ insert t into W (if there is room), otherwise add t to F;
5. output the tuples from W.
6. make F the input, clear F.

Comparison of algorithms

SFS:

- progressive
- window contains only skyline tuples:
  - $>$ skyline result fits in the window $\Rightarrow$ single pass
- partial-order preference relations: topological sort necessary.

BNL:

- output delayed
- window can contain non-skyline tuples:
  - $>$ skyline result fits in the window $\Rightarrow$ more than one pass may be necessary
- works for any partial-order preference relation.

Both algorithms can be extended for iterated skylines: multiple windows.

Other algorithms for skylines

Main-memory algorithms for the maximal vector problem in E^d:

- based on Divide-and-Conquer
- $\mathcal{O}(n(\log n)^{d-2}) + \mathcal{O}(n \log n)$ [Kung et al., 1975]
- result size k: $\mathcal{O}(n \log k)$ for $d = 2$ ([Kirkpatrick et al, 1985]

Algorithms based on nearest-neighbor search:

- optimal, progressive [Papadias et al., SIGMOD 2003]
- also [Kossmann et al., VLDB 2002]

Those algorithms do not generalize to arbitrary preference relations.

Linear optimization queries

Query formulation:

Find the input tuples that maximize $\sum_{i=1}^{n} a_i x_i$.

The preference relation:

$$\bar{x} \succ \bar{y} \equiv \sum_{i=1}^{n} a_i x_i > \sum_{i=1}^{n} a_i y_i.$$

Query evaluation methods (*top K* answers):

- convex hull of the input [Chang et al.,SIGMOD 2000]
- ranked materialized views [Hristidis et al.,SIGMOD 2001]

Related work

Preference queries [Lacroix, Lavency, VLDB 1987]:

> Pick the tuples of R satisfying $Q \wedge P_1 \wedge P_2$; if the result is empty, pick the tuples satisfying $Q \wedge P_1 \wedge \neg P_2$; if the result is empty, pick the tuples satisfying $Q \wedge \neg P_1 \wedge P_2$.

This can be expressed as

$$\omega_{C_2}(\omega_{C_1}(\sigma_Q(R)))$$

where

$$C_1(t_1, t_2) \equiv P_1(t_1) \wedge \neg P_1(t_2)$$

$$C_2(t_1, t_2) \equiv P_2(t_1) \wedge \neg P_2(t_2).$$

[Kiessling, Güntzer, 1994], [Govindaran, Jayaraman, Mantha, 2001]:

- clausally-defined preference relations
- extension of Datalog, requires a special evaluation method.

[Kiessling et al., VLDB 2002]:

- atomic and composite preference specifications
- winnow, skyline
- no logical framework
- implementation: Preference SQL compiled to SQL
- deployed applications: personalized search engines and shopping agents

Future work

Preference management:

- elicitation:
  - past research concentrated on eliciting multi-attribute utility functions
  - how to construct preference formulas?
- revision

General preference queries:

- queries with extrinsic preferences
- preferences over sets
- XML?

Evaluation and optimization of preference queries:

- rewrite rules
- semantic query optimization:
  - other techniques?
  - more general integrity constraints
- cost models:
  - skylines [Godfrey, FOIKS'04]
- indexing? materialized views?
- new classes of preference queries:
  - relaxation queries

Papers

1. J. Chomicki “Querying with Intrinsic Preferences,” Proc. EDBT’2002, Prague, March 2002.

2. J. Chomicki “Preference Formulas in Relational Queries,” ACM Transactions on Database Systems, December 2003.

3. J. Chomicki ”Semantic Optimization of Preference Queries,” Symposium on Applications of Constraint Databases, Paris, June 2004.

4. J. Chomicki, P. Godfrey, J. Gryz, D. Liang “Skyline with Presorting,” Poster at ICDE’03.