Preference Queries in Relational Databases

Jan Chomicki

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

University at Buffalo
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.

... but also keep in mind my preference for amazon.com.

Find the lowest price for this book on the Web...
Scoring functions

An approach grounded in utility theory:

1. Construct a real-valued function $u$ such that:

$$t_1 \preceq t_2 \iff u(t_1) \leq u(t_2)$$

2. Return the answers that maximize $u$ in the given instance $r$.

+ Scoring functions need to be hand-crafted for every input.
+ Provides an ordering of all the answers.
+
[Agrawal & Wimmers, 2000; Hristidis et al., 2001]
+ Can be implemented using SQL3 user-defined functions.

3. Not expressive enough: weak order preference relations.

- Scoring functions need to be hand-crafted for every input.
Non-existence of utility functions is unsatisfiable.

\[
\{(t_2)_n = (t_4)_n, (t_4)_n = (t_3)_n, (t_3)_n < (t_1)_n, (t_2)_n < (t_1)_n, n\}
\]

The set of constraints

<table>
<thead>
<tr>
<th>Price</th>
<th>Vendor</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>$17.30</td>
<td>bn.com</td>
<td>Green Guide: Greece</td>
</tr>
<tr>
<td>$18.80</td>
<td>bn.com</td>
<td>The Flanders Panel</td>
</tr>
<tr>
<td>$13.50</td>
<td>fatbrain.com</td>
<td>The Flanders Panel</td>
</tr>
<tr>
<td>$14.75</td>
<td>amazon.com</td>
<td>The Flanders Panel</td>
</tr>
</tbody>
</table>

The Flanders Panel
Preferencerelationsaslogicalformulas

Define preferences explicitly in a logical language:

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

\( \bar{d} \ast_d \bar{d} \lor \bar{d} = \bar{d} \equiv ( \bar{d}, \rho, \bar{d}, \rho ) \prec_1 ( \bar{d}, \rho, \bar{d}, \rho ) \)
<table>
<thead>
<tr>
<th>Title</th>
<th>Vendor</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green Guide: Greece</td>
<td>fatbrain.com</td>
<td>$17.70</td>
</tr>
<tr>
<td>The Flanders Panel</td>
<td>bn.com</td>
<td>$18.80</td>
</tr>
<tr>
<td>The Flanders Panel</td>
<td>fatbrain.com</td>
<td>$13.50</td>
</tr>
<tr>
<td>The Flanders Panel</td>
<td>amazon.com</td>
<td>$14.75</td>
</tr>
</tbody>
</table>
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

I. Preference formulas.
II. Winnow and its properties.
III. Optimization of preference queries.
IV. Logical composition of preferences.
V. Extrinsic preferences.
VI. Skyline and linear optimization queries.
VII. Related and future work.
Definitions

Preferencerelation:abinaryrelationbetweenthetuplesofagivenrelation.

Preferenceformula:afirst-orderformuladefiningapreferencerelation.

Intrinsicpreferenceformula:thedefinitionusesonlybuilt-inpredicates.

Typicalpropertiesofpreferencerelations:Irreflexivity,transitivity,and

dependences.

Intrinsicpreferencerelation:thedefinitionusesonlybuilt-in

Relations.

Preferenceformulaafirst-orderformuladefiningapreferencerelation

Givenrelation.

Preferencerelation:abinaryrelationbetweenthetuplesofa

Definitions
Given a preference relation \( \preceq \) defined using a preference formula

Example ("preference for Amazon"):

\[
\{(i_1, v_1, p_1) \preceq (i_2, v_2, p_2) : i_1 \neq i_2 \land v_1 = v_2 \land p_1 \leq p_2 \}
\]

\[
C = \{(i, v) \in \mathcal{R} \mid i \in I \land v \in V\}
\]

Given a preference relation \( \preceq \) defined using a preference formula
<table>
<thead>
<tr>
<th>Price</th>
<th>Vendor</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>$17.30</td>
<td>bn.com</td>
<td>Green Guide: Greece</td>
</tr>
<tr>
<td>$18.80</td>
<td>fatbrain.com</td>
<td>The Flanders Panel</td>
</tr>
<tr>
<td>$13.50</td>
<td>amazon.com</td>
<td>The Flanders Panel</td>
</tr>
<tr>
<td>$14.75</td>
<td></td>
<td>The Flanders Panel</td>
</tr>
</tbody>
</table>
Some properties of Winnow

Winnow can be formulated in relational algebra:

\[
\text{Book}(I', V', P', I, V, P) \subseteq \text{Book}(I, V, P) \setminus \text{Winnow}(\text{Winnow}(I, V, P, \text{Book}(I, V, P), I, V, P))
\]

If \( C_1(I_1, I_2) \subseteq C_2(I_1, I_2) \), then for all instances \( r \),

\( C_1(\text{Winnow}(r)) \subseteq C_2(\text{Winnow}(r)) \).

**Nonemptiness:**

If \( C \) defines a preference relation which is a strict partial order and \( r \) is finite, then \( \text{Winnow}(r) \) is nonempty.

**Containment:**

If \( C_1 \subseteq C_2 \), then for all instances \( r \),

\( \text{Winnow}(C_1(r)) \subseteq \text{Winnow}(C_2(r)) \).
Algebraic laws

Commutativity with selection:
If \( C_1(t_1; t_2) \) and \( C_2(t_1; t_2) \) are strict partial orders, then for all finite instances \( r \),
\[
\forall r \in C_1 \Rightarrow C_2(r) = C_2(r) \Rightarrow C_1(r).
\]

Distributivity over Cartesian product:
For every \( r_1 \) and \( r_2 \),
\[
C(r_1 \times r_2) = (C(r_1)) \times (C(r_2)).
\]

Commutativity of winnow:
If \( C_1(t_1; t_2) \) and \( C_2(t_1; t_2) \) are strict partial orders, then for every finite instance \( r \),
\[
\forall r \in C_1 \Rightarrow C_2(r) = C_2(r) \Rightarrow C_1(r).
\]
Semantic query optimization

Using information about integrity constraints to:

- Eliminate redundant occurrences of winnow.

Using information about integrity constraints to:

- Make more efficient computation of winnow possible.

Eliminating redundancy: Given a set of integrity constraints \( \mathcal{F} \),

\[
\forall t_1, t_2. \quad [R(t_1) \land R(t_2)] \iff [t_1 \land t_2 \lor t_1 \land t_2].
\]

Constraint:

\( \psi \) is redundant w.r.t. \( \mathcal{F} \) iff \( \mathcal{F} \) implies the following integrity constraint:

\[
\forall t_1, t_2: \quad [t_1 \land t_2 \lor t_1 \land t_2].
\]
Integrity constraints

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

\[ R(t_1) \land \cdots \land R(t_n) \land \left( \bigwedge_{i=1}^n \left[ R(t_i) \land \bigwedge_{j \neq i} (t_j) \right] \right) \]

Implication is decidable for CGDs.

Chomicki, Wolper, 1995: Constraint-generating dependencies (CGDs) [Baudinet, Chomicki, Wolper, 1995]:
Logical composition of preferences

Single-dimensional: $\text{Boolean operations, e.g.,: } x \left( \bar{\bar{1}} \bar{\bar{2}} \right) y \left( \bar{\bar{1}} \bar{\bar{2}} \right) x \left( \bar{\bar{1}} \bar{\bar{2}} \right)$

Prioritized composition: $x \left( \bar{\bar{1}} \bar{\bar{2}} \right) x \left( \bar{\bar{1}} \bar{\bar{2}} \right)$

Multi-dimensional: $\text{lexicographic and Pareto composition}$

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
- Other factors not on the components of the tuples being compared

Solution: Winnow + SQL
Preference for a lower total cost of a book (including shipping).

Vendor: 
- fatbrain.com: $3.99
- bn.com: $5.99
- fatbrain.com: $3.99
- amazon.com: $6.99
- SH

Apply window 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 [Borsos and V. Kossmann, Stocker '2001] (Pareto composition).

Properties:

- Skyline: maximums of monotone scoring functions.
A DIFF attribute indicates that tuples with different values of that attribute are incomparable.

```
SELECT ... FROM ... WHERE ...
GROUP BY ... HAVING ...
SELECT ... FROM ...
```

**Skyline in SQL**
The corresponding preference formula:

\[
\left( z > z \land f < h \right) \lor \left( \exists z \land f \geq h \right) \lor x = x \equiv (z, f, x) \prec (z, f, x)
\]

Special case of window
Skyline Using BNL

Block-nested-loops algorithm [Borzsonyi, 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:

4. Output the tuples from $W$ that were added there when $F$ was not empty.

5. Make $F$ the input, clear $F$.

6. Output the tuples from $W$ that were added there when $F$ was empty.

7. For every tuple $t$ in the input:

8. If $t$ is incomparable with all tuples in $W$:

9. Insert $t$ into $W$.

10. If $t$ is comparable with some tuples in $W$:

11. Eliminate the dominated tuples and insert $t$ into $W$.

12. If $t$ dominates some tuples in $W$:

13. Eliminate the dominated tuples.

14. If $t$ is dominated by a tuple in $W$:

15. Ignore $t$.

16. Otherwise:

17. Repeat the following until $F$ is empty:

18. For every tuple $t$ in the input:

19. Output $t$.

20. If there is room, otherwise add $t$ to $F$.

21. For every tuple $t$ in the input:

22. Make $F$ the input, clear $F$. 

23. Block-nested-loops algorithm [Borzsonyi, Kossmann, Stocker, 2001]:
Skyline using SFS

Chomicki, Godfrey, Gryz, Liang, 2003:

1. sort the input 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:
      * if $t$ is incomparable with all tuples in $W$ (if there is room), otherwise add $t$ to $F$;
      * if $t$ is dominated by a tuple in $W$ ignore $t$;
5. output the tuples from $W$.
6. make $F$ the input, clear $F$.

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

SFS:
- Progressive window contains only skyline tuples: \( \geq \) skyline results in the window
- Single pass may be necessary

BNL:
- Output delayed
- Window can contain non-skyline tuples:
  - More than one pass may be necessary
  - Skyline result fits in the window

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

- Works for any partial-order preference relation.
Other algorithms for skyline queries:

Main-memory algorithms for the maximal vector problem in $E^p$

Main-memory algorithms for the maximal vector problem in $E^p$

Those algorithms do not generalize to arbitrary preference relations.

Also [Kossmann et al., VLDB 2002]

•optimal, progressive [Papadias et al., SIGMOD 2003]

Algorithms based on nearest-neighbor search:

Result size $k$: $O(\log u) + (u \log u) + (u \log u) O$

Also [Kirkpatrick et al., 1985]

$\Omega$ for $d = 2$

$\Omega$ for $d = 2$

$\Omega$ for $d = 2$

Also [Papadias et al., SIGMOD 2003]

•optimal, progressive [Papadias et al., SIGMOD 2003]

Algorithms based on nearest-neighbor search:
Query formulation:

Find the input tuples that maximize

\[ \sum_{i=1}^{n} a_i x_i \]
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 $Q \wedge P_2$. If the result is empty, pick the tuples satisfying $Q \wedge P_1 \wedge \emptyset$. If the result is empty, pick the tuples satisfying $Q \wedge P_1 \wedge P_2$.

This can be expressed as

$$\mathcal{C}_2(\mathcal{C}_1(r^1, r^2) \equiv \mathcal{D}_1(r^1, r^2) \equiv \mathcal{C}_1(r^1, r^2) \equiv \mathcal{D}_2(r^1, r^2)),$$

where

$$((\mathcal{C}_2(\mathcal{C}_1(r^1, r^2) \equiv \mathcal{D}_1(r^1, r^2) \equiv \mathcal{C}_1(r^1, r^2) \equiv \mathcal{D}_2(r^1, r^2))) \circ \mathcal{C}_1(r^1, r^2) \equiv \mathcal{D}_1(r^1, r^2) \equiv \mathcal{C}_1(r^1, r^2) \equiv \mathcal{D}_2(r^1, r^2)).$$

Related work
Shopping agents

- deployed applications: personalized search engines and
  - clausally-defined preference relations
  - extension of Datalog, requires a special evaluation method.
- no logical framework
  - winnow, skyline
  - atomic and composite preference specifications

[Kiessling et al., VLDB 2002]:

Implementation: Preference SQL compiled to SQL

- two SQL views
- winnow, skyline

Deployment: personalized search engines and shopping agents

[Kiessling, Günther, 1994], [Govindarajan, Jayaraman, Mantha, 2001]:

Personalized search engines

- Preference SQL compiled to SQL
- two SQL views
- winnow, skyline

Implementation: Preference SQL compiled to SQL

- clausally-defined preference relations
Future work

Preferencemanagement:

- elicitation:
  - pastresearchconcentratedonelicitingmulti-attributeutilityfunctions
  - how to construct preferenceformulas?

- preferences over sets
- queries with extrinsic preferences
- General preferencequeries:
  - revision
  - utility functions
  - pastresearchconcentratedonelicitingmulti-attributeelicitation:

Preference management:

XML?
Evaluation and optimization of preference queries:

- Rewrite rules
- Semantic query optimization
- Other techniques

Cost models:
- More general integrity constraints

Evaluation and optimization of preference queries:

Rewrite rules

Other techniques
Papers


2002.


Papers