Optimization of Preference Queries

Jan Chomicki
University at Buffalo
http://www.cse.buffalo.edu/~chomicki
Plan of the talk

1. Relational query languages: relational algebra, SQL.
2. Relational query evaluation: data structures, algorithms.
5. Preference query evaluation and optimization.
Relational algebra

A set of operators on relations that can be nested to form expressions:

- set-theoretic: union, set difference
- almost set-theoretic: Cartesian product $R \times S$
  \[ r \times s = \{ t : t[R] \in r \land t[S] \in s \} \]
- selection $\sigma_\alpha(R)$
  \[ \sigma_\alpha(r) = \{ t : t \in r \land \alpha(t) \} \]
- projection $\pi_X(R)$
  \[ \pi_X(r) = \{ t[X] : t \in r \} \]
- join
  \[ R \bowtie S = \sigma_{A=B}(R(\ldots, A, \ldots) \times S(\ldots, B, \ldots)). \]
Selection $\sigma_{\text{Price} < 15}(\text{Book})$:

<table>
<thead>
<tr>
<th>Book</th>
<th>Title</th>
<th>Vendor</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>The Flanders Panel</td>
<td>amazon.com</td>
<td>$14.75</td>
</tr>
<tr>
<td>$t_2$</td>
<td>The Flanders Panel</td>
<td>fatbrain.com</td>
<td>$13.50</td>
</tr>
<tr>
<td>$t_3$</td>
<td>The Flanders Panel</td>
<td>bn.com</td>
<td>$18.80</td>
</tr>
<tr>
<td>$t_4$</td>
<td>Green Guide: Greece</td>
<td>bn.com</td>
<td>$17.30</td>
</tr>
</tbody>
</table>
SQL

A hybrid language:

- relational algebra

- relational calculus (= first-order logic)
Basic form:

\[
\text{SELECT } A_1, \ldots, A_n \\
\text{FROM } R_1, \ldots, R_k \\
\text{WHERE } C
\]

This corresponds to the following relational algebra expression:

\[
\pi_{A_1, \ldots, A_n}(\sigma_C(R_1 \times \cdots \times R_k))
\]
Other features of SQL

Nested subqueries which may include quantification.

Aggregation: MAX, MIN, SUM,...

Grouping.

```
SELECT Title,
       MIN(Price) AS MPrice
FROM Book
GROUP BY Title
```

<table>
<thead>
<tr>
<th>Title</th>
<th>MPrice</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Flanders Panel</td>
<td>$13.50</td>
</tr>
<tr>
<td>Green Guide: Greece</td>
<td>$17.30</td>
</tr>
</tbody>
</table>
Relational query evaluation

Indexing:
- fast access to individual rows using the values of one or more index columns
- speeding-up queries $\sigma_{A=c}$
- special data structures (B-trees) and algorithms (hashing)

Joins:
- various methods: nested loops, hash join, sort-merge join, index join
Query optimization

SQL → Parsing → Parse Tree → Logical Plan Generation & Rewriting

→ Physical Evaluation Plan
→ Cost-based Optimisation
→ Logical Query Plan

Statistics
Algebraic query optimization

Using algebraic laws to rewrite logical query plans.

Pushing selection down:

\[ \sigma_\alpha(E_1 \times E_2) = \sigma_\alpha(E_1) \times E_2 \]

if \( F \) involves only the attributes of \( E_1 \).

\[ \sigma_\alpha(E_1 \cup E_2) = \sigma_\alpha(E_1) \cup \sigma_\alpha(E_2). \]

\[ \sigma_\alpha(E_1 - E_2) = \sigma_\alpha(E_1) - \sigma_\alpha(E_2). \]

Also pushing projections down, join reordering, ...
Semantic query optimization

Using integrity constraints to transform the query.

Various techniques:

- join elimination/introduction
- predicate elimination/introduction
- detecting empty results
Predicate elimination

If we know that faculty members are at least 30 years old, then

SELECT Name FROM Faculty WHERE AGE > 25

can be rewritten as

SELECT Name FROM Faculty
Cost-based query optimization

Estimating the cost of physical evaluation plans:

- number of I/O operations (or an approximation)
- based on stored statistics

Enumerating equivalent evaluation plans to find a plan of least cost.
Preference queries

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

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

... but also keep in mind my preference for amazon.com.”
Preferences as first-order formulas

[Chomicki, EDBT’02].

Relation $Book(Title, Vendor, Price)$.

Preference:

$$(i, v, p) \succ_{C_1} (i', v', p') \equiv i = i' \land p < p'.$$

Indifference:

$$(i, v, p) \sim_{C_1} (i', v', p') \equiv i \neq i' \lor p = p'.$$
Relational algebra embedding

[Chomicki, EDBT’02; Kiessling, VLDB’02]:

New winnow operator returning the tuples in the
given instance that are not dominated by any other
tuple in the instance.

<table>
<thead>
<tr>
<th>Book</th>
<th>Title</th>
<th>Vendor</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>The Flanders Panel</td>
<td>amazon.com</td>
<td>$14.75</td>
</tr>
<tr>
<td>$t_2$</td>
<td>The Flanders Panel</td>
<td>fatbrain.com</td>
<td>$13.50</td>
</tr>
<tr>
<td>$t_3$</td>
<td>The Flanders Panel</td>
<td>bn.com</td>
<td>$18.80</td>
</tr>
<tr>
<td>$t_4$</td>
<td>Green Guide: Greece</td>
<td>bn.com</td>
<td>$17.30</td>
</tr>
</tbody>
</table>
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 ($\Rightarrow$ strict partial orders), can be effectively checked for intrinsic preference formulas with $=, \neq, <, >, \leq, \geq$. 
Weak orders

Weak order: a strict partial order with transitive indifference.
Utility (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.

Typically, top $K$ answers are requested.
Properties of scoring functions

+ can be implemented using SQL3 user-defined functions  
  [Agrawal et al, SIGMOD’00] [Hristidis et al., SIGMOD’01]

+ provide an ordering of all the answers

+ capture preference intensity

+ can be numerically aggregated
  – need to be hand-crafted for every input
  – hard to logically aggregate
  – not expressive enough: only weak order pref. relations.
Non-existence of utility functions

<table>
<thead>
<tr>
<th></th>
<th>Title</th>
<th>Vendor</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>The Flanders Panel</td>
<td>amazon.com</td>
<td>$14.75</td>
</tr>
<tr>
<td>$t_2$</td>
<td>The Flanders Panel</td>
<td>fatbrain.com</td>
<td>$13.50</td>
</tr>
<tr>
<td>$t_3$</td>
<td>The Flanders Panel</td>
<td>bn.com</td>
<td>$18.80</td>
</tr>
<tr>
<td>$t_4$</td>
<td>Green Guide: Greece</td>
<td>bn.com</td>
<td>$17.30</td>
</tr>
</tbody>
</table>

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.
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.com"): 

\[
(i, v, p) \succ_2 (i', v', p') \equiv i = i' \\
\land v = 'amazon.com' \land v' \neq 'amazon.com'
\]
<table>
<thead>
<tr>
<th></th>
<th><strong>Title</strong></th>
<th><strong>Vendor</strong></th>
<th><strong>Price</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>The Flanders Panel</td>
<td>amazon.com</td>
<td>$14.75</td>
</tr>
<tr>
<td>$t_2$</td>
<td>The Flanders Panel</td>
<td>fatbrain.com</td>
<td>$13.50</td>
</tr>
<tr>
<td>$t_3$</td>
<td>The Flanders Panel</td>
<td>bn.com</td>
<td>$18.80</td>
</tr>
<tr>
<td>$t_4$</td>
<td>Green Guide: Greece</td>
<td>bn.com</td>
<td>$17.30</td>
</tr>
</tbody>
</table>
Applications of winnow

Preference SQL [Kiessling et al, VLDB’02].

Skyline queries [Börzsönyi et al, ICDE’01]:

- find all the tuples that are not dominated by any other tuple in every dimension (Pareto set).

Linear optimization queries:

- find all the tuples that maximize \( \sum_{i=1}^{n} a_i x_i \).
Winnow evaluation: BNL

[Börzsönyi et al, ICDE’01].

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 \( W \) ⇒ ignore \( t \),
   - \( t \) dominates some tuples in \( W \) ⇒ eliminate them and insert \( t \) into \( W \),
   - \( t \) is incomparable with all tuples in \( W \) ⇒ 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.
Algebraic laws [Chomicki, TODS’03]

Commutativity with selection:

If the formula

$$\forall t_1, t_2. \ (\alpha(t_2) \land \gamma(t_1, t_2)) \Rightarrow \alpha(t_1)$$

is valid, then for every $r$

$$\sigma_\alpha(\omega_\gamma(r)) = \omega_\gamma(\sigma_\alpha(r)).$$
Example

The preference relation

\[(i, v, p) \succ_{C_1} (i', v', p') \equiv i = i' \land p < p'.\]

The selection \(\sigma_{Price<20}\) commutes with \(\omega_{C_1}\) because

\[\forall p, p', i, i'[(p' < 15 \land i = i' \land p < p') \Rightarrow p < 15]\]

is a valid formula.

The selection \(\sigma_{Price>20}\) does not commute with \(\omega_{C_1}\) because

\[\forall p, p', i, i'[(p' > 15 \land i = i' \land p < p') \Rightarrow p > 15]\]

is not a valid formula.
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 $\succeq_{C_1}$
and $\succeq_{C_2}$ are strict partial orders, then for all finite instances $r$:

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

Also commutativity with projection.
Semantic query optimization

[Chomicki, CDB’04].

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 $F, \omega_C$ is redundant w.r.t. $F$ iff $F$ entails the formula

$$\forall t_1, t_2. \ R(t_1) \land R(t_2) \Rightarrow t_1 \sim_C t_2.$$
Integrity constraints

Constraint-generating dependencies (CGDs) [Baudinet et al, JCSS’99]:

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

Entailment is **decidable** for CGDs by reduction to the validity of \( \forall \)-formulas in the constraint theory.
Example

Relation $Book(Title, Vendor, Price)$.

For the preference relation

$$(i, v, p) \succ_{C_1} (i', v', p') \equiv i = i' \land p < p'$$

$\omega_{C_1}(Book)$ is redundant w.r.t. FD $Title \rightarrow Price$, because the formula

$$(i_1 \neq i_2 \lor p_1 = p_2) \land i_1 = i_2 \land p_1 < p_2$$

is unsatisfiable.
Cost-based optimization

Little known about result size estimates for preference queries. For skylines [Buchta, 1989; Godfrey, FOIKS’04]:

The expected cardinality of a $d$-dimensional skyline of $n$ tuples is equal to $H_{d-1,n}$, the $d-1$-order harmonic of $n$ (under attribute independence).

Asymptotically: $H_{d,n} \in \Theta((\ln n)^d/d!)$.  

Some values:

\[
\begin{align*}
H_{2,10^6} &= 104 \\
H_{6,10^6} &= 14,087
\end{align*}
\]
Extension: extrinsic preference

Extrinsic 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).

<table>
<thead>
<tr>
<th>Vendor</th>
<th>SH</th>
</tr>
</thead>
<tbody>
<tr>
<td>amazon.com</td>
<td>$6.99</td>
</tr>
<tr>
<td>fatbrain.com</td>
<td>$3.99</td>
</tr>
<tr>
<td>bn.com</td>
<td>$5.99</td>
</tr>
</tbody>
</table>

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: computing Cartesian products.
Extension: preferences between sets

A best set does not necessarily consist of the best individuals:

- bundling [Chang et al, EC’03]
- complementarity
- diversity \(\Rightarrow\) College Admissions Problem

Design query language extensions in which:

- sets are first-class citizens: powerset? nondeterminism?
- solutions can be constrained
- set winnow is available.
Future work

Preference management:

• elicitation: how to construct preference formulas?

• aggregation

Decision components:

• preferences between actions: workflows, ECA systems

• preferences between E-services

Preferences for XML?