Preference SQL - Design, Implementation, Experiences

Werner Kießling
Institute of Computer Science
University of Augsburg

Gerhard Köstler
Database Preference
Software GmbH Augsburg
(now at Intershop, Jena)
Overview

1. Motivation
2. Preference Query Model
3. Language Design
4. Application Study
5. Implementation Issues
6. Summary and Outlook
1. Motivation

Personalization of search engines and of Web services:
- Deficiencies: “empty result“ effect, “flooding“ effect
- Insufficient ad-hoc remedies:
  Parametric search, boolean “expert” search, ...

Demand for database support of personalization:
- Need for a suitable preference model
- Need for a suitable declarative preference query model
- Compatibility with industry standards (SQL, XML)
- Time to market
- Efficiency
2. Preference Query Model

Preferences are NOT:
• hard constraints ... necessarily numbers ... necessarily total orders

Preferences are:
• Intuitively: Personalized wishes “I like A better than B”
• Formally: (for details see my yesterday’s talk)
  - Given a set $A$ of attribute names with a domain of values $\text{dom}(A)$,
    a preference $P$ is a strict partial order $P = (A, <P)$ on $\text{dom}(A)$.
  - $x <P y$ is interpreted as “I like $y$ better than $x$”.

Specification of preferences:
• People are used to express “better than” qualitatively.
• Preference attributes may have categorical or numerical domains.
• Preferences may be complex, covering multiple attributes.
Preference SQL - Design, Implementation, Experiences

Preference Query Model (cont’d)

Preference Engineering

- **Base preference constructors in Preference SQL:**
  - On categorial attributes:
    - **POS, NEG, POS/NEG, POS/POS, EXP**
  - On numerical attributes:
    - **AROUND, BETWEEN, LOWEST, HIGHEST**

- **Complex preference constructors in Preference SQL:**
  - Pareto preference P: \( P = P_1 \otimes P_2 \) (equal importance)
  - Prioritized preference P: \( P = P_1 \& P_2 \) (ordered importance)
Preference queries:

- Given $P = (A, \prec_P)$ and a relation $R$, $R[A] \subseteq \text{dom}(A)$, a preference query $\sigma[P](R)$ is a **soft selection** operation on $R$.

- **Perfect choices** are maximal elements of $P$.

- **Best-Matches-Only (BMO)** query model:
  - Retrieve **perfect choices**, if present in $R$.
  - Otherwise deliver **best-matching alternatives**, but **nothing worse**.

- Nuisances from empty result / flooding effect are defeated.
3. Language Design

- **Preference SQL** = **Standard SQL** + Preferences

- **Preference SQL Query Block**:

  ```sql
  SELECT <selection>
  FROM <table_references>
  WHERE <hard_conditions>
  PREFERING <soft_conditions>
  GROUPING <attribute_list>
  BUT ONLY <but_only_condition>
  ORDER BY <attribute_list>
  ```
Preference SQL examples:
(base preference constructors)

SELECT * FROM trips
PREFERRING duration AROUND 14;

SELECT * FROM apartments
PREFERRING HIGHEST(area);

SELECT * FROM programmers
PREFERRING exp IN ('java', 'C++');

SELECT * FROM hotels
PREFERRING location <> 'downtown';
Preference SQL examples:
(complex preference constructors)

SELECT * FROM computers
PREFERRING HIGHEST(main_memory)
    AND HIGHEST(cpu_speed);

SELECT * FROM car
WHERE make = 'Opel'
PREFERRING (category = 'roadster' ELSE
    category <> 'passenger'
    AND price AROUND 40000
    AND HIGHEST(power))
    CASCADE color = 'red'
    CASCADE LOWEST(mileage);
Answer explanation: TOP, LEVEL, DISTANCE

```sql
SELECT
    ident, color, age,
    LEVEL(color),
    DISTANCE(age)
FROM oldtimer
PREFERING
    color = 'white' ELSE
    color = 'yellow'
AND age AROUND 40;
```

<table>
<thead>
<tr>
<th>oldtimer</th>
<th>ident</th>
<th>color</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maggie</td>
<td>white</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>Bart</td>
<td>green</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>Homer</td>
<td>yellow</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>Selma</td>
<td>red</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Smithers</td>
<td>red</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>Skinner</td>
<td>yellow</td>
<td>51</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ident</th>
<th>color</th>
<th>age</th>
<th>level</th>
<th>distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selma</td>
<td>red</td>
<td>40</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Homer</td>
<td>yellow</td>
<td>35</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Maggie</td>
<td>white</td>
<td>19</td>
<td>1</td>
<td>21</td>
</tr>
</tbody>
</table>
Quality control: BUT ONLY

```sql
SELECT *
FROM trips
PREFERRING start_day AROUND '1999/7/3'
    AND duration AROUND 14
BUT ONLY  DISTANCE(start_day) <= 2
    AND DISTANCE(duration) <= 2;
```

The GROUPING clause:

```sql
SELECT *
FROM   oldtimer
PREFERRING age AROUND 40
GROUPING color;
```
4. Application Study

- Design issues of personalized search engines:
  - Hard selections (WHERE) vs. soft (PREFERING)?
  - Which quality control (BUT ONLY condition)?
  - Importance of multi-attribute preferences (AND vs. CASCADE)?
  - Result presentation (use of TOP, LEVEL, DISTANCE)?

- Where do preferences come from:
  - Hard-wired into search mask by the e-service provider?
  - Explicitly entered by the e-customers?
  - Implicitly entered vendor preferences?
  - Implicitly entered domain knowledge?
Preference SQL - Design, Implementation, Experiences

Application Study (cont’d)

Please describe your preferred car:

- Manufacturer: BMW
- Model & Color: 7er
- Price Range: 0-75000
- Kms: 0-30000
- Registration Year: 1997-1999
- Power in horsepower:
- Zip Code: (prefix is possible)
- Diesel Fuel:

Options:
- Airbag
- Automatic Transmission
- Used Car Warranty
- Air Conditioning
- Navigation Aid
- Used Car Certification
- Anti-Theft Device
- Central Locking
Preference SQL - Design, Implementation, Experiences

Application Study (cont’d)

```
SELECT *,
    TOP(manufacturer),
    TOP(model), TOP(price),
    TOP(mileage), TOP(regyear),
    TOP(diesel), TOP(airbag),
    TOP(autotransmission),
    TOP(aircondition)
FROM used_cars
PREFERRING (manufacture = 'BMW' AND model = '7')
    CASCADE (price BETWEEN 0, 75000
                AND mileage BETWEEN 0, 30000
                AND regyear between 1997, 1999
                AND diesel = 'yes'
                AND airbag = 'yes'
                AND autotransmission = 'yes'
                AND aircondition = 'yes');
```
Preference SQL - Design, Implementation, Experiences

Application Study (cont’d)

<table>
<thead>
<tr>
<th>Picture</th>
<th>BMW</th>
<th>BMW</th>
<th>BMW</th>
<th>BMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturer</td>
<td>7er</td>
<td>7er</td>
<td>7er</td>
<td>7er</td>
</tr>
<tr>
<td>Model</td>
<td>DEM 59900</td>
<td>DEM 52900</td>
<td>DEM 64900</td>
<td>DEM 38000</td>
</tr>
<tr>
<td>Price</td>
<td>23000</td>
<td>25000</td>
<td>15000</td>
<td>33000</td>
</tr>
<tr>
<td>Registration Year</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diesel Fuel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Airbag</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automatic transmission</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Air Condition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© W. Kießling, 28th Intern. Conf. on Very Large Databases (VLDB 2002), Hong Kong, 22.08.02
5. Implementation Issues

- Plug-and-go application integration: (loose coupling)

Diagram:

- E/M-Commerce application
- Preference SQL Optimizer
- Standard SQL DB system
- Standard ODBC/JDBC driver
- Preference ODBC/JDBC driver
The Preference SQL Optimizer:

- Preference SQL - Design, Implementation, Experiences

**Implementation Issues (cont’d)**

The Preference SQL Optimizer:

```
CREATE VIEW Aux AS
SELECT *
FROM Cars
PREFERRING Make = 'Audi'
AND Diesel = 'yes';
```

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Make</th>
<th>Model</th>
<th>Price</th>
<th>Mileage</th>
<th>Airbag</th>
<th>Diesel</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Audi</td>
<td>A6</td>
<td>40000</td>
<td>15000</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>BMW</td>
<td>5 series</td>
<td>35000</td>
<td>30000</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>Volkswagen</td>
<td>Beetle</td>
<td>20000</td>
<td>10000</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

```
CREATE VIEW Aux AS
SELECT *, CASE WHEN Make = 'Audi'
THEN 1 ELSE 2 END
AS Makelevel,
CASE WHEN Diesel = 'yes'
THEN 1 ELSE 2 END
AS Diesellevel
FROM Cars;
```

```
INSERT INTO Max
SELECT Identifier, Make, Model,
Price, Mileage, Airbag, Diesel
FROM Aux A1
WHERE NOT EXISTS
(SELECT 1 FROM Aux A2
AND
A2.Diesellevel <= A1.Diesellevel
AND
A2.Diesellevel < A1.Diesellevel));
```

```
SELECT *
FROM Cars
PREFERRING Make = 'Audi'
AND Diesel = 'yes';
```

© W. Kießling, 28th Intern. Conf. on Very Large Databases (VLDB 2002), Hong Kong, 22.08.02
A large scale performance benchmark:
6. Summary and Outlook

Product development of Preference SQL:

- It is the very first instance of a declarative preference query language, compatibly extending standard SQL.
- It has been invented already in 1997.
- Its first commercial product release was by spin-off company Database Preference Software GmbH, Augsburg in 1999.
- It works on top of all major commercial database systems.
- It has been deployed in commercial e-commerce platforms as Preference Search Cartridge for Intershop 4 and enfinity.
Salient features of Preference SQL:

- It implements the Best-Matches-Only (BMO) query model for strict partial order preferences.
- It’s much more powerful than the Skyline-operator (proposed in 2001).
- It’s a highly convenient and cost-effective way to develop advanced personalized search engines:
  - **Focused search**: Empty result & flooding effects are defeated.
    (This is even more important for mobile commerce.)
  - Parametric search and boolean “expert” search have become obsolete.
- Its quality functions are a great basis for cooperative database interfaces.
- Sufficient performance for many e-commerce applications by loose coupling.
Preference SQL – Design, Implementation, Experiences

Summary and Outlook (cont’d)

“it’s a Preference World”

Preference research at the Univ. of Augsburg:

• Preference SQL is used in: P-Services, P-Bargainer, P-Agent, P-News
• Preference SQL approach is transferred to XML: Preference XPATH
• Performance issues (tight coupling):
  - Integration of efficient Skyline-operators for subset of Preference SQL.
  - P-Optimizer: Algebraic optimization using preference algebra.
• Preference mining and maintenance: P-Miner, P-Repository

Funding: DFG, FORSIP (www.forsip.de)

© W. Kießling, 28th Intern. Conf. on Very Large Databases (VLDB 2002), Hong Kong, 22.08.02