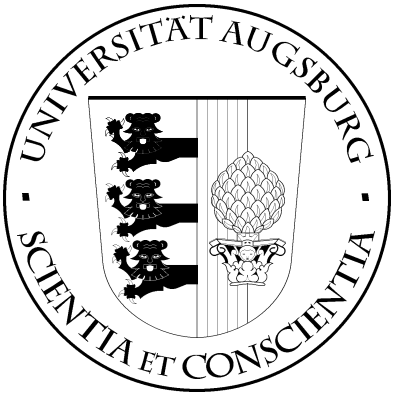
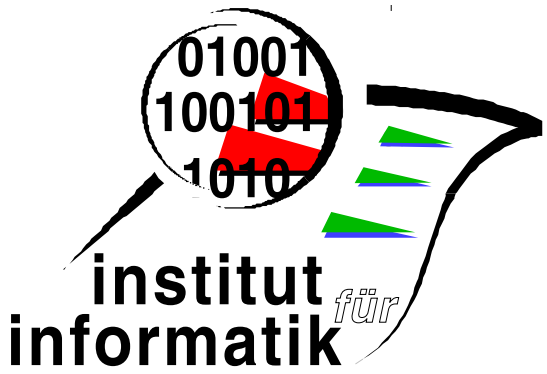
Universität Augsburg



Optimizing Preference Queries for Personalized Web Services

Werner Kießling, Bernd Hafenrichter

Report 2002-12 Juli 2002



Institut für Informatik
D-86135 Augsburg

Optimizing Preference Queries for Personalized Web Services

Werner Kießling, Bernd Hafenrichter

Institute of Computer Science, University of Augsburg, D-86135 Augsburg, Germany
Contact: Prof. Dr. Werner Kießling (kiessling@informatik.uni-augsburg.de)

Abstract

Personalization of Web services requires a powerful preference model that smoothly and efficiently integrates with standard database query languages. We make the case for preferences as strict partial orders, supported in Preference SQL and Preference XPATH. Performance of Web services will crucially depend on various architectural design decisions. We pointed out that a central server architecture is desirable. Concerning the implementation of preference queries we investigated the tightly coupled architecture, presenting a novel approach for algebraic optimization based on preference algebra. We provided new transformation laws and gave evidence for the power of this heuristic optimization. This forms the basis for a new preference query optimization methodology, promising sufficient performance even for complex Web services.

Keywords: Personalized Web services, preference engineering, focused search, preference query optimization

1 Introduction

Preferences are an indispensable part of our daily life, be it private or business–related. Thus the notion of preferences should be a key element in designing personalized software and Internet-based information systems ([1]). Personal preferences are often expressed in the sense of wishes: Wishes are free, but there is no guarantee that they can all be satisfied at all times. In case of failure for the perfect match people are not always, but usually prepared to accept worse alternatives or to negotiate compromises. Thus preferences in the real world require a paradigm shift from exact matches towards *match-making*, which means to find the best possible matches between one's wishes and the reality. In other words, preferences lead to the notion of *soft constraints*. Since data is predominantly stored in object-relational databases (and recently in XML databases), amalgamating preferences with declarative query languages like SQL or XPATH is considered one key challenge towards successfully implementing personalized Web services.

The contributions of this paper are part of the research program *"Preference World"*, which has been originated already back in 1995. One of its major triggers had been the ubiquitous observation that the Internet hosts an abundance of inadequate search engines, suffering from the *empty result* effect and the *flooding* effect. In fact, one reason for this deficiency is that the notion of preferences has not been dealt with properly in the past. The demand for appropriate and efficient solutions gets even more challenging when *mobile Web services* are considered, where *focused search* is really essential ([11]).

In this paper we will elaborate on *preference-driven database technology* as a backbone for personalized Web services. To set the stage, in section 2 we revisit recent solutions on preference engineering. In section 3 we investigate the performance impact of architectural design decisions for Web services. Novel contributions concerning the algebraic optimization of preference queries are presented in section 4. A summary and outlook on current research in 'Preference World' are given in section 5.

2 Preference Engineering

Intuitively people express their wishes in the form " *I like A better than B*". Since this can be understood by everybody, it is a natural way of modeling user preferences. Mathematically such preferences can be characterized by *strict partial orders* ([10], [5])

2.1 The preference model

Preferences can be engineered inductively as follows. Depending on the application domain, a set of pre-defined so-called *base preferences* are assumed to be defined. For e-commerce this set includes e.g. so-called `around` preferences, `between` preferences, `positive` or `negative` preferences, preferences for `lowest` or `highest` values (see [10, 12] for details).

Given single preferences, more complex ones can be gained by means of three fundamental operators.

Complex preference constructors

1. *Pareto* preference: $\mathbf{P := P_1 \otimes P_2 \otimes ... \otimes P_n}$
   P is a combination of *equally important* preferences, following the well-known Pareto principle.
2. *Prioritized* preference: $\mathbf{P := P_1 \,\&\, P_2 \,\&\, ... \,\&\, P_n}$
   P evaluates *more important* preferences earlier, similar to a lexicographical ordering. P_1 is most important, P_2 next, etc.
3. *Ranked* preference: $\mathbf{P := rank_F(P_1, P_2, ..., P_n)}$

P combines individual numerical scores SCORE(P_i), $1 \leq i \leq n$, by means of some ranking function F.

Proposition 1 Preference construction is closed under strict partial orders.

Further fundamental algebraic properties of strict partial order preferences are given in [10].

2.2 Integration into Database Query Languages

Standard SQL queries only support hard selection conditions. To express preferences we extended the syntax of SQL by an additional PREFERRING clause for complex preference constructors. This extension is called *Preference SQL* (see [12] for details), currently supporting all previously mentioned base preferences together with the constructors for Pareto and prioritized preferences:

```
SELECT  <selection> FROM  <tables>
WHERE       <hard-conditions>
PREFERRING <soft-conditions>
```

Preference queries are evaluated following the **BMO query model**, treating preferences as *soft constraints* with a cooperative answer semantics: ***B**est **M**atches **O**nly* w.r.t. the given strict partial order are returned. Clearly this avoids both the empty result and the flooding effect.

Example: Get the best offers for compact hi-fi systems including a CD player in a price range of 100 to 115$.

The subsequent Preference SQL query uses the base preferences 'CONTAINS' and 'BETWEEN'; 'AND' denotes the construction of a Pareto preference:

```
SELECT description, price FROM  devices
WHERE category = 'hi-fi systems'
PREFERRING description CONTAINS 'CD'
       AND price BETWEEN(100, 115)
```

Suppose that this query is evaluated for mobile commerce where focused search is essential.

Figure 1: Using a Preference SQL WAP service

As illustrated in Fig. 1, the BMO semantics avoided the empty result effect (none of the item on sales satisfied the BETWEEN clause perfectly), hence the best alternatives (and only those, avoiding the flooding effect) were selected. A similar extension of the query language XPATH for XML has been carried out, resulting in **Preference XPATH** (seen [10]).

In future Web-based information systems, automatic *query expansion* may lead to very complex preference queries. E.g., a cognitive query builder may adapt focused search by long-term user preferences, user intentions and emotional state, situational awareness and domain knowledge from ontologies ([11], [9]). Consequently, there is a high demand for good optimization techniques for preference queries.

3 Architecural Issues of Web-Services

Before investigating preference query optimization let us elaborate architectural impacts of Web service design.

3.1 The central server architecture

As known from [3], doing heavy online access to heterogeneous Internet sources leads to poor performance for Web services, given current bandwidth and Internet response times. Thus a central server architecture as shown in figure 2 is more promising for the time being.

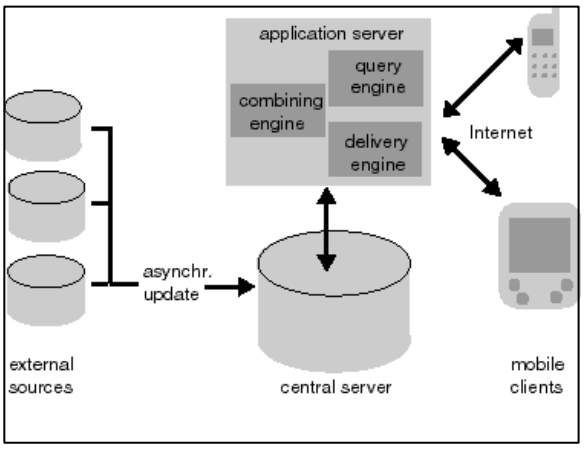


Figure 2: Central server architecture for Web services

A typical application scenario is e.g. a *mobile route planner*. A prototype under development at 'Preference World' finds best routes from city A to city B, given personal preferences on travel expenses, road conditions and traffic jam awareness.[1] The asynchronous update can be performed by Unix cron-jobs feeding a central server SQL database, which also stores an autobahn map. Mobile clients are supported using XSLT technology.

If Pareto or prioritization are chosen for preference construction, Preference SQL can do the job of the *combining engine*. Alternatively we combined single preferences by means of a variant of *Quick-Combine* ([7]), efficiently implementing ranked preferences.

[1] External sources used are www.wetter.de, www.wdr.de and www.bmvbw.de, respectively.

3.2 The Pre-Processor Architecture

Given a central server architecture for Web services we now focus in more detail on efficiency issues for preference queries. Preference SQL queries can be implemented on top of a relational database system by translating them into pre-optimized SQL queries as depicted in figure 3. This achieves a rapid integration into all database systems supporting the ODBC or JDBC standard. Details of this pre-processor approach are presented in [12].

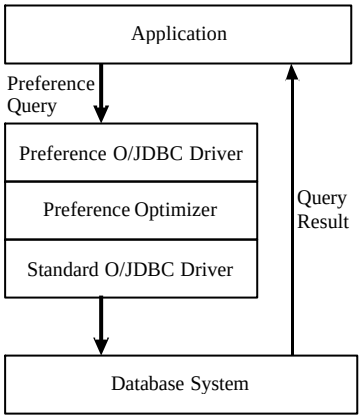


Figure 3: Pre-processor approach for preference queries

The application issues a query to the Preference-Driver, rewriting it into a complex SQL query which in turn is optimized thereafter by the SQL query optimizer at hand. This approach is very flexible, relatively easy to implement and performs sufficiently well for a large choice of applications ([12]).

3.3 The Tightly-Coupled Architecture

Obviously, by integrating the preference optimizer and the SQL optimizer more tightly even better performance can be expected[2]. The results in [4] point in this direction. So let's study this architecture as depicted in figure 4 more closely. The main obstacle of this approach is that a practical implementation requires the close cooperation with SQL database manufacturers, which is not easy to obtain. However, the expected performance boost might be a rather strong incentive. So let's explore the performance advantages we can possibly expect.

With tight coupling the *preference query evaluation model* should be an extension of classic relational algebra by appropriate new preference-specific operators to deal with soft selection. Given that, we can follow the classical paradigm of database query optimization ([13]):

1. Given the preference query, build an *initial operator tree* T_0 (also called preference query execution plan).

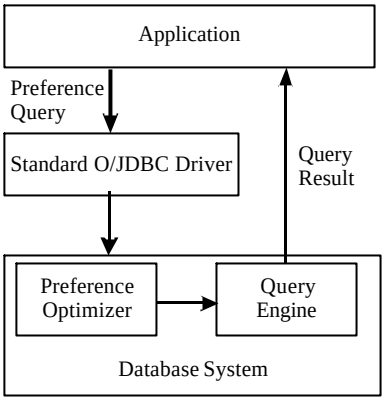


Figure 4: Tightly coupled preference evaluation

2. Given a set of algebraic equations, optimize T_0 by applying them, if a subtree matches one side of a law. Which laws to apply, and in what order, has to be controlled by some *heuristics* that intelligently prune the exponentially large search space (typically greedy hill-climbing is used). Let T_{fin} denote the final tree emerging from this phase.
3. *Cost-based* optimization: The abstract operators contained in T_{fin} must be mapped to concrete evaluation algorithms. If there are several choices for a particular operator, a cost model has to decide.

For the scope of this paper we will concentrate on stage 2, which is known to frequently achieve the biggest optimization gains for SQL databases, sometimes in the order of several magnitudes[3].

4 Algebraic Preference Query Optimization

The subsequent results are founded on the preference algebra defined in [10]. Due to lack of space we omit all proofs (but [8] will have the details). Moreover, we only state those laws from [8] that are required here.

4.1 Preference Relational Algebra

We propose to extend conventional relational algebra by two operators to deal with soft selections (their precise declarative semantics is given in [10]):

- *Preference selection* σ**[P](R)** finds all best matching objects for preference P in relation R.
- *Grouped preference selection* σ**[P groupby A](R)** partitions R by equal A-values and performs a preference selection on each partition.

Let P denote the preference specified in the PREFERRING-clause. Then the *operational semantics* of a Preference SQL query looks as follows:

$$\pi_{\textbf{selection}}(\sigma[P](\sigma_{\textbf{hard-conditions}}(R_1 \times \ldots \times R_n)))$$

[2] Some familiarity of the reader with basic algebraic query optimization techniques for relational database systems is helpful.

[3] Quick-Combine or the Skyline-operator of [4], being a restricted form of a Pareto preference, are instances of stage 3.

This expression canonically defines our initial operator tree T_0. Heuristic query optimization relies on the assumption that reducing the sizes of intermediate results leads to more efficient query plans. The classical most promising strategies are 'push hard selection' and 'push projection' ([13]). We will adopt this idea here by developing transformation rules subsequently that allow us to perform '*push preference*' within query execution plans.

4.2 Transformation Laws

Here are first transformation laws involving preferences.

Proposition 2

Let A denote the set of preference attributes of P.

(L1) *Push projection over preference selection*

a) $\pi_X(\sigma[P](R)) = \sigma[P](\pi_X(R))$ if $A \subseteq X$

b) $\pi_X(\sigma[P](R)) = \pi_X(\sigma[P](\pi_{X \cup A}(R)))$ otherwise

(L2) *Push preference selection over Cartesian product*

$\sigma[P](R \times T) = \sigma[P](R) \times T$ if $A \subseteq R$

(L3) *Push preference selection over union*

$\sigma[P](R \cup T) = \sigma[P](\sigma[P](R) \cup \sigma[P](T))$ ¨

Next we state laws that can introduce grouped preference selection into the query execution plan.

Proposition 3

Let preference P_1 be defined on a set of attributes A_1.

(L4) *Split prioritization into grouping*

$\sigma[P_1 \,\&\, P_2](R) = \sigma[P_2 \text{ groupby } A_1](\sigma[P_1](R))$

Proposition 4

Let A be the set of preference attributes of P.

(L5) *Push projection over preference selection*

a) $\pi_X(\sigma[P \text{ groupby } B](R)) = \sigma[P \text{ groupby } B](\pi_X(R))$ if $A \subseteq X$, $B \in X$

b) $\pi_X(\sigma[P \text{ groupby } B](R)) = \pi_X(\sigma[P \text{ groupby } B](\pi_{X \cup A \cup B}(R)))$ otherwise ¨

Queries may be a combination of hard selections and preference selections. Thus it is important to analyze the interaction of those two. Pushing $\sigma[P]$ over σ_H for some hard condition H will turn out to cause major obstacles in general. Thus, since a relational *join* operation is defined as a hard selection over a Cartesian product, pushing $\sigma[P]$ over a join becomes more difficult. However, *foreign key* relationships from relational schema information will enable further transformations.

Proposition 5

Let A be the set of attributes of preference P, $A \subseteq R$, and R.X be a foreign key referring to T.X.

(L6) *Push preference selection over a join selection*

$\sigma[P](\sigma_{R.X = T.X}(R \times T)) = \sigma_{R.X = T.X}(\sigma[P](R \times T))$ ¨

Note that the right-hand side of L6 can be transformed further by L2, achieving to push preference selection into a join in summary.

Proposition 6

Let A_1 be the set of attributes of preference P_1, $A_1 \subseteq R$, and R.X be a foreign key referring to T.X.

(L7) *Split Pareto preference and push over join*

$\sigma[P_1 \otimes P_2](\sigma_{R.X = T.X}(R \times T)) =$

$\sigma[P_1 \otimes P_2](\sigma_{R.X = T.X}(\sigma[P_1 \text{ groupby } X](R) \times T))$ ¨

4.3 Case Studies

Let us now demonstrate the potential of algebraically optimizing preference queries by two case studies.

- **Case study 1:**

The subsequent query asks for cars, preferably being around eight years old, which is considered most important ('PRIOR TO'). Less important is the preference on brand, being positively for a BMW or a Mercedes.

```
SELECT u.price, d.brand, u.age
FROM   used_cars u, car_detail d
WHERE  u.id = d.id and u.color = 'Red'
PREFERRING u.age around 8
  PRIOR TO d.brand in ('BMW', 'Mercedes');
```

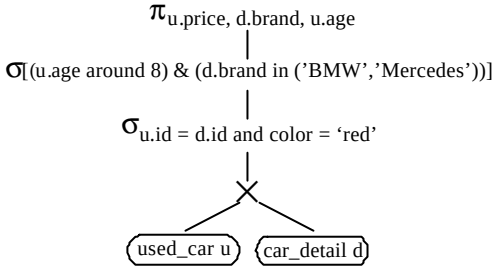


Figure 5: Initial query plan for case study 1

A *preference query optimizer* can process the initial query plan in figure 5 as follows, applying the strategy to reduce intermediate result sizes early and using a greedy hill-climbing strategy to prune the search space (for classical relational algebra transformations we refer to [13]):

1. Split the hard selection and push $\sigma_{color = 'red'}$ down to used_car.
2. Apply law L4 to split prioritization into grouping.
3. Assuming foreign key d.id referring to u.id, law L6 can be fired next, pushing σ[age around 8] into the left branch of $\times$.
4. Apply L5a to push $\pi_{u.price,\ d.brand,\ u.age}$ over σ[brand in ('BMW', 'Mercedes') groupby age], followed by pushing projection further down as far as possible using L1a.
5. Combine $\sigma_{u.id = d.id}$ followed by $\times$ into an equijoin.

The final query execution plan is depicted in figure 6.

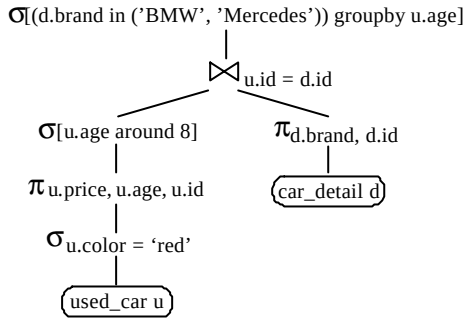


Figure 6: Optimized query plan for case study 1

The result of this joint work of classical relational algebra optimization extended by preference-specific laws is quite remarkable. In addition to the usual optimization result, we succeeded to push the highly selective σ[age around 8] into the join, which can achieve big performance speed-ups given a large used_car relation.

Let's evaluate this plan, given the following small sample database.

used_cars

Id	Price	Age	Color
1	20.000	6	red
2	16.000	8	blue
3	30.000	10	red
4	30.000	11	red
1	30.000	11	blue
4	15.000	12	yellow

car_details

Id	Brand	Type
1	BMW	Cabriolet
2	Renault	Coupé
3	Fiat	Coupé
4	Audi	Avant

First, $\sigma_{color = 'red'}$ filters off all 3 non-red tuples from used_cars. Then σ[age around 8] does not find perfect matches, but selects tuples with age 6 or 10 as best alternatives, whereas the tuple with age 11 is worse and gets discarded. After the join σ[brand in ('BMW', 'Mercedes') groupby age] is evaluated. Because none of the tuples are equal in age, nothing is filtered off. So we get this BMO query result:

Price	Brand	Age
20.000	BMW	6
30.000	Fiat	10

- **Case study 2:**

The subsequent query asks for cars preferably being around eight years old, they should be a BMW or Mercedes and the preferred color is red. All these preferences are supposed to be equally important, hence getting combined into a Pareto preference ('AND'):

```
SELECT u.price, d.brand, u.age
FROM   used_cars u, car_detail d
WHERE  u.id = d.id
PREFERRING u.age around 8
       AND d.brand in ('BMW', 'Mercedes')
       AND u.color in ('red');
```

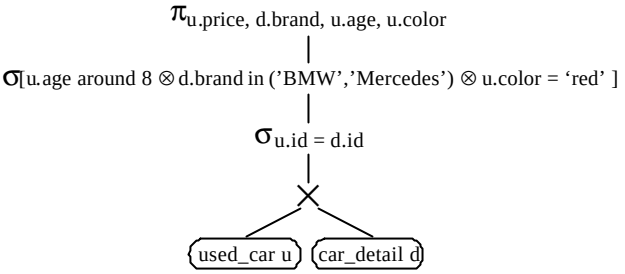


Figure 7: Initial query plan for case study 2

Let's put our sample preference query optimizer to work as before, given the initial query plan in figure 7:

1. Since Pareto preferences are associative and commutative, the preference selection can be rearranged as σ[(age around 8 ⊗ color = 'red') ⊗ (brand in ('BMW', 'Mercedes'))].
2. Assuming foreign key d.id referring to u.id, L7 can be used to split above rearranged Pareto preference, pushing its first operand into the join.
3. Push the projection over preference selection and further down the tree by applying the familiar laws.
4. Combine $\sigma_{u.id = d.id}$ followed by × into an equijoin.

The resulting query tree is given in figure 8. This time we succeeded for Pareto preferences to push a grouped preference selection into the join, which can achieve further performance speed-ups for large used_car relations.

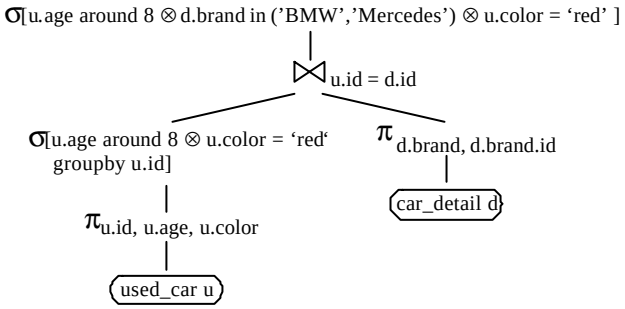


Figure 8: Optimized query plan for case study 2

We evaluate this plan, given the same sample database as in case study 1. The grouped Pareto preference selection filters off the following tuples from used_cars:

Id	Price	Age	Color
1	30.000	11	blue
4	15.000	12	yellow

Note that e.g. the tuple with id = 1 is discarded, because the other tuple with id = 1 in used_cars has a preferable color and a better age, thus dominating it under Pareto preference semantics.

After the join, the final preference selection yields this BMO query result:

Id	Price	Age	Color	Brand
1	20.000	6	red	BMW
2	16.000	8	blue	Renault
3	30.000	10	red	Fiat

5 Summary and Outlook

Advanced personalization in Web services requires a powerful and intuitive preference model. We have made the case for preferences as strict partial orders, enabling a systematic preference engineering. Database query languages like Preference SQL and Preference XPATH, which treat preferences as soft selection conditions under the BMO query model, are the key towards focused multi-attribute search. Since next generation Web services will also perform extensive automatic query expansion, optimization of complex preference queries becomes a key challenge. As a unique novel contribution in this paper we have given several algebraic transformation laws for preference queries, aiming to push preference selections down a preference relational algebra tree. Two case studies exemplified the potential of this approach.

This work is currently extended towards a full-scale preference query optimizer, which will be implemented in a tightly coupled architecture with existing query optimizers. For rapid prototyping we are going to use the XXL-library from [2]. Two advanced preference application projects will be used as major test environments. The *P-News* project[4] will support sophisticated query expansion in the context of a personalized news service for MPEG7 digital libraries. The *COSIMA* project[5] ([6]) aims to build a new generation of agent-based Web information systems, where the user can bargain with an emotional, speaking smart avatar. At the heart of this is again the need for complex preference engineering and preference query optimization to achieve real-time performance over the Internet

[4] P-News is funded by the German Research Association DFG. (www.informatik.uni-augsburg.de/lehrstuehle/info2/p-news)

[5] COSIMA is funded by the Bavarian Research Cooperation FORSIP on *Situated, Individualized and Personalized Man-Machine Interaction*. (www.myCosima.com/englishbargain)

Acknowledgments

We are grateful to Georg Struth for helpful technical comments and suggestions.

References

[1] Special issue of the Communications of the ACM on *Personalization*, vol. 43, Aug. 2000.

[2] J. Bercken, B. Blohsfeld, J. Dittrich, et al.: *XXL - A Library Approach to Supporting Efficient Implementations of Advanced Database Queries*. Proc. 27th Intern. Conf. on Very Large Databases (VLDB), Rome, Sept. 2001.

[3] N. Bruno, L. Gravano, A. Marian: *Evaluating top-k queries over web-accessible databases*. Intern. Conf. on Data Engineering (ICDE), San Jose, USA, Febr. 2002.

[4] S. Börzsönyi, D.Kossmann, Konrad Stocker: *The Skyline Operator*. Proc 17th Intern. Conf. on Data Engineering, Heidelberg, April 2001.

[5] J. Chomicki: *Querying with intrinsic preferences*. Proc. Intern. Conf. on Advances in Database Technology (EDBT), Prague, Czech Republic, March 2002, pp. 34-51.

[6] S. Fischer, W. Kießling, S.Holland, M. Fleder: *The COSIMA Prototype for Multi-Objective Bargaining*. First Intern. Conf. On Autonomous Agents and Multiagent Systems (AAMAS), Bologna, July 2002, to appear.

[7] U. Güntzer, W.-T. Balke, W. Kießling: *Optimizing Multi-Feature Queries for Image Databases*. Proc. 26th Intern. Conf. on Very Large Databases (VLDB), Cairo, Egypt, 2000, pp. 419-428.

[8] B. Hafenrichter, W. Kießling: *Optimizing Preference Queries for Personalized Web Services*. Extended version of this paper (in preparation).

[9] I. Horrocks: *DAML+OIL: a reason-able web ontology language*. Proc. Intern. Conf. on Advances in Database Technology (EDBT), Prague, March 2002, pp. 2-13.

[10] W. Kießling: *Foundations of Preferences in Database Systems.* Tech. Rep. 2001-8, Institute of Comp. Science, Univ. of Augsburg. Accepted for 28th Intern. Conf. on Very Large Databases (VLDB), Hong Kong, Aug. 2002.

[11] W. Kießling, W.T. *Balke: Mobile Search in a Preference World*. Proc. Workshop on Mobile Search in conj. with 11th Intern. World Wide Web Conf., Honolulu, May 2002, pp. 32-38.

[12] W. Kießling, G. Köstler: *Preference SQL - Design, Implementation, Experiences*. Tech. Rep. 2001-7, Institute of Comp. Science, Univ. of Augsburg. Accepted for 28th Intern. Conf. on Very Large Databases (VLDB), Hong Kong, Aug. 2002.

[13] J. Ullman: *Principles of Database and Knowledge-Base Systems*. Vol. 1, Computer Science Press, 1989.

Proof of Laws:

Definition 1: A *"preference"* $P=(A,<_P)$ defines a strict partial order $<_P \subseteq dom(A) \times dom(A)$.

Definition 2: The *"preference selection s[P](R)"* is defined as $\sigma[P](R) = \{ w \in R \mid \neg\exists v \in R : w < v \}$

Intuitively, this query select all maximal tuples that are not dominated by other tuples.

Definition 3: The "*Grouped preference selection s[P groupby A](R)"* is defined as :

$$\sigma[P \text{ groupby } A](R) = \{ w \in R \mid \neg\exists v \in R : w <_P v \wedge w[A] = v[A] \}$$

Definition 4: Given the relations R and T, the *"extension of R by T"* through a function f is defined as:

$$R \text{ ext}_f T = \{(r,t) \mid r \in R, t \in T, f(r,t) \}$$

Furthermore, we claim that for every tuple r exists at least one tuple in t, which satisfies f(r,t).

Definition 5: Given an attribute B, the *"anti-chain preference"* $B^{\leftrightarrow} = (B, <_{\leftrightarrow})$ is defined as $<_{\leftrightarrow} = \emptyset$. This denotes that all elements are incomparable to each other.

Lemma 1: Let $P=(A,<_P)$ and a relation R , $A \subseteq$ attributes(R). Then P induces an order onto $R \times R$

$a, b \in R, a < b \Leftrightarrow \pi_A(a) <_P \pi_A(b)$.

Lemma 2: Let denote A and B sets of attributes with $A \subseteq B$. Then $\pi_A(\pi_B(R)) = \pi_A(R)$

Lemma 3: Let $P=(A,<_P)$, $A \subseteq X$.
Then $\pi_X(a) < \pi_X(b) \Leftrightarrow a < b$

$\pi_X(a) < \pi_X(b)$ Le1
$= \pi_A(\pi_X(a)) < \pi_A(\pi_X(b))$ Le2
$= \pi_A(a) <_P \pi_A(b)$ Le1
$= a < b$ ¨

Lemma 4: Let A denote a set of attributes, R and T database relations, with $A \subseteq$ attributes(R). Then $\pi_A(R \times T) = \pi_A(R)$.

Lemma 5: Let ext_f be an extension of R by T. Then $\sigma[P](R \text{ ext}_f T) = \sigma[P](R) \text{ ext}_f T$ is valid, if $\text{attrib}(P) \subseteq R$.

$\sigma[P](R \text{ ext}_f T)$ D4
$= \{ w \in R \text{ ext}_f T \mid \neg\exists v \in R \text{ ext}_f T : w < v \}$ Le1
$= \{ w \in R \text{ ext}_f T \mid \neg\exists v \in R \text{ ext}_f T : \pi_A(w) < \pi_A(v) \}$
$= \{ w \in R \text{ ext}_f T \mid \neg\exists v \in R : \pi_A(w) < \pi_A(v) \}$
$= \{ w \in R \mid \neg\exists v \in R : \pi_A(w) < \pi_A(v) \} \text{ ext}_f T$
$= \sigma[P](R) \text{ ext}_f T$ ¨

Lemma 6: The preference selection $\sigma[P \text{ groupby } B](R)$ denotes a grouped preference selection. This selection can be expressed in terms of a prioritized anti chain preference $\sigma[B^{\leftrightarrow} \& P](R)$.

$\sigma[P \text{ groupby } B](R)$ D3
$= \{ w \in R \mid \neg\exists v \in R : w[B]=v[B] \wedge w <_P v \}$ D5
$= \{ w \in R \mid \neg\exists v \in R : w[B] <_{B\leftrightarrow} v[B] \vee (w[B] = v[B] \wedge w <_P v) \}$ Def.
$= \sigma[B^{\leftrightarrow} \& P](R)$ ¨

Lemma 7: Let $\sigma[P1](R)$ and $\sigma[P2](R)$ denote preferences. Then

$$\sigma[P1](R) \subseteq \sigma[P2](R) \Rightarrow \sigma[P1](R) = \sigma[P1](\sigma[P2](R))$$

(L1) Push projection over preference selection

a) $\pi_X(\sigma[P](R)) = \sigma[P](\pi_X(R))$ if $A \subseteq X$
b) $\pi_X(\sigma[P](R)) = \pi_X(\sigma[P](\pi_{X \cup A}(R)))$ otherwise

Proof (a): Let $P=(A,<_P)$ and $A \subseteq X$, i.e. π_X preserves the preference Attributes A. Then:

$\pi_X(\sigma[P](R))$ D2
$= \pi_X(\{w \in R \mid \neg\exists v \in R : w < v\})$ Le1
$= \{\pi_X(w) \in R \mid \neg\exists v \in R : \pi_A(w) <_P \pi_A(v) \}$ Le2
$= \{w \in \pi_X(R) \mid \neg\exists v \in \pi_X(R) : w < v\}$ D2
$= \sigma[P](\pi_X(R))$ ¨

Proof (b): Let $P=(A,<_P)$ and $A \not\subset X$, i.e. π_X does not preserves the preference Attributes A. Then:

$\pi_X(\sigma[P](R))$ Le2
$= \pi_X(\pi_{X \cup A}(\sigma[P](R)))$ L1a
$= \pi_X(\sigma[P](\pi_{X \cup A}(R))$ ¨

(L2) Push preference selection over Cartesian product

$$\sigma[P](R \times T) = \sigma[P](R) \times T \quad \text{if } A \subseteq R$$

Proof: The cartesian product $R \times T$ can be rewritten in terms of an extension function.

$$R \times T = R \text{ ext}_{\times} T$$
$$\times(r,t) = \text{true}$$

$\sigma[P](R \times T)$ D4
$= \sigma[P](R \text{ ext}_{\times} T)$ Le5
$= \sigma[P](R) \text{ ext}_{\times} T$
$= \sigma[P](R) \times T$ ¨

(L3) Push preference selection over union

s **[P](R** $\dot{\cup}$ **T) =** s **[P](**s **[P](R)** $\dot{\cup}$ s **[P](T))**

Proof:

$R' = \sigma[P](R) \cup \sigma[P](T)$

$\sigma[P](\sigma[P](R) \cup \sigma[P](T)) =$ D3
$= \{ w \in R' \mid \neg\exists v \in R' : w <_P v\}$ R'
$= \{ w \in R' \mid \neg((\exists v \in \sigma[P](R) : w < v) \vee$
$(\exists v \in \sigma[P](T) : w < v)) \}$
$= \{ w \in R' \mid \neg((\exists v \in R : w < v \wedge v \in \sigma[P](R)) \vee$
$(\exists v \in T : w < v \wedge v \in \sigma[P](T)))\}$

Since $<_P$ is transitive, we immediately get

$= \{ w \in R' \mid \neg((\exists v \in R : w < v) \vee$
$(\exists v \in T : w < v))\}$
$= \{ w \in R \cup T \mid \neg(\exists v \in R \cup T : w < v) \wedge$
$w \in \sigma[P](R) \cup \sigma[P](T) \}$

Again, since $<_P$ is transitive, we immediately get

$= \{ w \in R \cup T \mid \neg(\exists v \in R \cup T : w <_P v)\}$
$= \sigma[P](R \cup T)$ ¨

(L4) Split prioritization into grouping

s **[**P_1 **&** P_2**](R) =** s **[**P_2 **groupby** A_1**](**s **[**P_1**](R))**

Proof:

$\sigma[P1\&P2](R) =$ D2
$= \{w \in R \mid \neg (\exists v \in R : \pi_{A1}(w) <_{P1} \pi_{A1}(v) \vee$
$(\pi_{A1}(w)=\pi_{A1}(v) \wedge \pi_{A2}(w) <_{P2} \pi_{A2}(v)))\}$
$= \{w \in R \mid \neg ((\exists v \in R: \pi_{A1}(w) <_{P1} \pi_{A1}(v)) \vee$
$(\exists v' \in R: (\pi_{A1}(w)=\pi_{A1}(v')$
$\wedge \pi_{A2}(w) <_{P2} \pi_{A2}(v'))))\}$
$= \{w \in R \mid (\neg\exists v \in R: \pi_{A1}(w) <_{P1} \pi_{A1}(v)) \wedge$
$(\neg\exists v' \in R: (\pi_{A1}(w)=\pi_{A1}(v')$
$\wedge \pi_{A2}(w) <_{P2} \pi_{A2}(v'))))\}$
$= \sigma[P1](R) \cap \sigma[P1 \text{ groupby } A1](R)$

$\sigma[P_2 \text{ groupby } A1](\sigma[P_1](R))$ Le6
$= \sigma[A_1{}^{\leftrightarrow} \text{ \& } P_2](\sigma[P_1](R))$ D2
$=\{ w \in \sigma[P_1](R) \mid \neg\exists v \in \sigma[P_1](R): \pi_{A1}(w)=\pi_{A1}(v)$
$\wedge w <_{P_2} v \}$
$=\{ w \in R \mid (\neg\exists v \in \sigma[P_1](R) : w[A_1]=v[A_1] \wedge$
$w <_{P_2} v) \wedge$
$w \in \sigma[P1](R) \}$
$=\{ w \in R \mid (\neg\exists v \in \sigma[P_1](R) : \pi_{A1}(w)= \pi_{A1}(v) \wedge$
$w <_{P_2} v) \wedge$
$w \in \sigma[P1](R) \}$
$=\{ w \in R \mid (\neg\exists v \in \sigma[P_1](R) : \pi_{A1}(w)= \pi_{A1}(v) \wedge$
$w <_{P_2} v) \wedge$
$w \in \sigma[P1](R) \}$
$=\{ w \in R \mid (\neg\exists v \in R : \pi_{A1}(w)= \pi_{A1}(v) \wedge$
$w <_{P_2} v) \wedge w \in \sigma[P1](R)\}$
$=\{ w \in R \mid (\neg\exists v \in R : w[A_1]=v[A_1] \wedge w <_{P_2} v)\}$
$\cap \{w \in \sigma[P1](R) \}$
$= \sigma[P1](R) \cap \sigma[P1 \text{ groupby } A1](R)$

$\Rightarrow \sigma[P_1 \text{ \& } P_2](R) = \sigma[P_2 \text{ groupby } A_1](\sigma[P_1](R))$ ¨

(L5) Push projection over preference selection

Proof: Immediately from L1, since groupby-selection can be rewritten in terms or prioritization, using an anti-chain order on attribute B, as defined in Lemma 5.

a) $p_X($**[P groupby B](R)) =**
s **[P groupby B](**p_X**(R))** if $A \subseteq X$, $B \in X$

$\pi_X(\sigma[P \text{ groupby } B](R))$ Le6
$= \pi_X(\sigma[B^{\leftrightarrow} \text{ \& } P](R))$ L1a
$= \sigma[B^{\leftrightarrow} \text{ \& } P](\pi_X(R))$ Le6
$= \sigma[P \text{ groupby } B](\pi_X(R))$ ¨

b) $p_X($**[P groupby B](R)) =**
$p_X($**[P groupby B](**$p_{X \dot{\cup} A \dot{\cup} B}$ **(R)))** otherwise

$\pi_X(\sigma[P \text{ groupby } B](R))$ Le6
$= \pi_X(\sigma[B^{\leftrightarrow} \text{ \& } P](R))$ L2a
$= \pi_X(\sigma[B^{\leftrightarrow} \text{ \& } P](\pi_{X \cup A \cup B}(R))$ Le6
$= \pi_X(\sigma[P \text{ groupby } B](\pi_{X \cup A \cup B}(R))$ ¨

(L7) *Push preference selection over a join selection*
s **[P](**$s_{R.X = T.X}$**(R** ´ **T)) =** $s_{R.X = T.X}$**(**s **[P](R** ´ **T))**

Proof: Let A be the set of attributes of preference P, $A \subseteq$ R, and R.X be a foreign key referring to T.X.

Due to the knowledge that there exists a foreign key relationship between R.X and T.X, the join can be rewritten in terms of an extension function.

$\sigma_{R.X = T.X}(R \times T) = R \text{ ext}_{join} T$
$join(r,t) = \{ true : r[x] = t[x], false otherwise \}$

$\sigma[P](\sigma_{R.X = T.X}(R \times T))$ D4
$= \sigma[P](R \text{ ext}_{join} T)$ Le5
$= \sigma[P](R) \text{ ext}_{join} T$ D4
$= \sigma_{R.X = T.X}(\sigma[P](R) \times T)$ ¨

(L7) Split Pareto preference and push over join

Let A_1 be the set of attributes of preference P_1, $A_1 \subseteq R$, $A_2 \subseteq T$, and R.X be a foreign key referring to T.X.

$$\sigma[\mathbf{P_1} \ddot{A} \mathbf{P_2}](\sigma_{\mathbf{R.X = T.X}}(\mathbf{R} \acute{} \mathbf{T})) =$$
$$\sigma[\mathbf{P_1} \ddot{A} \mathbf{P_2}](\sigma_{\mathbf{R.X = T.X}}(\sigma[\mathbf{P_1\ groupby\ X}](\mathbf{R}) \acute{} \mathbf{T}))$$

Proof:

$\sigma[P1 \otimes P2](R)$ Def.

$= \{ w \in R \mid \neg\exists v \in R : w <_{P1 \otimes P2} v \}$ Def.

$= \{ w \in R \mid \neg(\exists v \in R:$
$(\pi_{A1}(w) <_{P1} \pi_{A1}(v) \wedge$
$(\pi_{A2}(w)=\pi_{A2}(v) \vee \pi_{A2}(w) <_{P2} \pi_{A2}(v))) \vee$
$(\pi_{A2}(w) <_{P2} \pi_{A2}(v) \wedge$
$(\pi_{A1}(w)=\pi_{A1}(v) \vee \pi_{A1}(w) <_{P1} \pi_{A1}(v))))$

$= \{ w \in R \mid \neg(\exists v \in R:$
$(\pi_{A1}(w) <_{P1} \pi_{A1}(v) \wedge \pi_{A2}(w)=\pi_{A2}(v)) \vee$
$(\pi_{A1}(w) <_{P1} \pi_{A1}(v) \wedge \pi_{A2}(w) <_{P2} \pi_{A2}(v)) \vee$
$(\pi_{A2}(w) <_{P2} \pi_{A2}(v) \wedge \pi_{A1}(w)=\pi_{A1}(v)))$

$= \{ w \in R \mid$
$\neg((\exists v \in R: (\pi_{A1}(w) <_{P1} \pi_{A1}(v) \wedge \pi_{A2}(w)=\pi_{A2}(v)))$
$\vee (\exists v \in R: (\pi_{A1}(w) <_{P1} \pi_{A1}(v) \wedge \pi_{A2}(w) <_{P2} \pi_{A2}(v)))$
$\vee (\exists v \in R: (\pi_{A2}(w) <_{P2} \pi_{A2}(v) \wedge \pi_{A1}(w)=\pi_{A1}(v))))$

$= \{ w \in R \mid$
$(\neg\exists v \in R: (\pi_{A1}(w) <_{P1} \pi_{A1}(v) \wedge \pi_{A2}(w)=\pi_{A2}(v)))$
$\wedge (\neg\exists v \in R: (\pi_{A1}(w) <_{P1} \pi_{A1}(v) \wedge \pi_{A2}(w) <_{P2} \pi_{A2}(v)))$
$\wedge (\neg\exists v \in R: (\pi_{A2}(w) <_{P2} \pi_{A2}(v) \wedge \pi_{A1}(w)=\pi_{A1}(v)))$

$= \sigma[P_1\ groupby\ A_2](R)$
$\cap\ \sigma[P_2\ groupby\ A_1](R)$
$\cap\ \{w \in R \mid \exists v \in R: (\pi_{A2}(w) <_{P2} \pi_{A2}(v) \wedge$
$\pi_{A1}(w)=\pi_{A1}(v))\}$

$\Rightarrow \sigma[P_1 \otimes P_2](R) \subseteq \sigma[P_1\ groupby\ A_2](R)$

Due to Lemma 7, we can conclude that

$\sigma[P_1 \otimes P_2](R) = \sigma[P_1 \otimes P_2](\sigma[P_1\ groupby\ A_2](R))$

The foreign key R.X establishes a functional dependency between [R.X] and the grouping Attribute A_2. Therefore, it is possible to substitute Attribute A_2 by the foreign key attribute X.

$\Rightarrow \sigma[P_1 \otimes P_2](R) = \sigma[P_1 \otimes P_2](\sigma[P_1\ groupby\ X](R))$

$\sigma[P_1 \otimes P_2](\sigma_{R.X = T.X}(R \times T))$
$= \sigma[P_1 \otimes P_2](\sigma[P_1\ groupby\ X](\sigma_{R.X = T.X}(R \times T)))$ Le6
$= \sigma[P_1 \otimes P_2](\sigma[X^{\leftrightarrow}\ \&\ P_1](\sigma_{R.X = T.X}(R \times T)))$ L7
$= \sigma[P_1 \otimes P_2](\sigma_{R.X = T.X}(\sigma[X^{\leftrightarrow}\ \&\ P_1](R) \times T))$ Le6
$= \sigma[P_1 \otimes P_2](\sigma_{R.X = T.X}(\sigma[P_1\ groupby\ X](R) \times T))$ ¨