

Optimization of Preference Queries under Hard Sum Constraints

Markus Endres and Werner Kießling

Institute for Computer Science
University of Augsburg
Universitätsstr. 14
86159 Augsburg, Germany
{endres, kiessling}@informatik.uni-augsburg.de

Abstract

Many important applications, e.g. planning tasks, demand the flexible and efficient use of personalization and preference handling techniques. Applying preference-based search technology could improve things quite a lot, e.g. using *Preference SQL* where preferences (i.e. soft constraints) can be combined with hard constraints. However, there are still fundamental efficiency issues that need to be addressed. In this paper we study preference database queries involving hard constraints over the sum of multiple attributes. We develop algebraic optimization techniques to transform a preference query with a sum constraint in order to enable its efficient processing by database engines. For this purpose we present new transformation laws for an efficient solution of this problem.

Introduction

Preferences are an integral part of our private and business-related life. Personal preferences are often expressed in the terms of wishes. In case of failure for the perfect match, people are often willing to accept worse alternatives or to negotiate compromises. In some instances, preference queries challenge traditional query processing and optimization, as illustrated by the following example.

Example 1: Consider a database storing nutritional information for single servings of different kinds of food relations like Soups, Meats and Beverages. A user, Mrs. Diet, is interested in finding meals that satisfy nutritional requirements such as a restriction on the number of calories (cal). For example, the recommendations for a 30-year old female, who is moderately active, are at most 1100 calories (USDA 2007).

However, in real life, each user has preferences concerning its meals. Mrs. Diet for example likes chicken soup as starter. The main course should be beef and the total lipid (abbr. 'fat') of the beef should be as little as possible. For beverage she likes red wine. The complete meal must fulfill the restriction of 1100 calories. Mrs. Diet wants to find all such meals which fulfill the hard constraint and satisfy her preferences best possible.

Using Kießling's approach of modelling preferences as strict partial orders (Kießling 2002; 2005), the above mentioned hard and soft constraints can be expressed by Preference SQL (Kießling and Köstler 2002) as follows:

```
SELECT S.name, M.name, B.name
FROM   Soups S, Meats M, Beverages B
WHERE  S.cal + M.cal + B.cal <= 1100
PREFERRING
      S.name IN ('Chicken soup')           AND
      M.name IN ('Beef') AND M.fat LOWEST AND
      B.name IN ('Red wine')
```

This query expresses Mrs. Diet's preferences after the keyword PREFERRING. It is a Pareto preference (AND) consisting of preferences on soups, meats and beverages. IN denotes a preference for members of a given set, a POS-preference. The whole preference is evaluated on the result of the hard sum constraint.

In this paper we call the query given in the example a **Preference SUM-constraint query**, since there is a hard sum constraint over multiple attributes belonging to several relations and user preferences which should be fulfilled.

Conventional approaches implement this type of preference query by composing pair-wise joins and then evaluating the preferences with an algorithm like (Börzsönyi, Kossmann, and Stocker 2001; Preisinger and Kießling 2007). Because each condition refers to attributes from more than two relations, pair-wise join operators may fail to remove intermediate results based on these conditions. Thus, producing cartesian products of relations may lead to high memory and computation costs.

In this paper we address such queries and focus on the critical issue of preference query optimization in combination with hard sum constraints over multiple attributes from different database relations. We introduce query rewriting techniques that can be used by join operators to remove tuples from relations that do not lead to any results at the end of the evaluation.

First, let's take a look at related work. Afterwards we describe the background of our preference model and discuss the Preference SUM-constraint problem. Thereafter we develop preference optimization techniques for this problem. Further on, we present experimental results, and finally we conclude with a summary and outlook.

Related Work

Queries with constraints over attributes belonging to several relations occur frequently in the real world, e.g. in the context of document retrieval or in e-commerce. However, they have not been intensively researched in the database context.

(Agarwal et al. 1998) address queries with linear constraints, and (Guha et al. 2003) address queries with aggregation constraints. However, their work is only valid for queries on one relation. (Ilyas, Aref, and Elmagarmid 2003) developed algorithms for top-k queries that can be extended to implement queries with a constraint on the value of a monotone function. (Liu, Yang, and Foster 2005) and (Nestorov, Liu, and Foster 2007) integrated constraint-programming techniques with traditional database techniques to solve sum constraint queries by modifying existing nested-loop-join operators. (Döring, Preisinger, and Endres 2008) present a first approach for processing of queries dealing with individual and global preferences of customers in combination with a hard constraint. (Hafenrichter and Kießling 2005) and (Chomicki 2003) developed transformation laws for preference queries with hard constraints, but these rules only correspond to the one-relation case.

However, none of them consider a sum constraint over a set of attributes belonging to several relations in combination with preference handling. But in the database query context hard constraints over a set of attributes are very important, particularly in combination with user preferences.

Our techniques are based on query rewriting in combination with preferences. This leads to algebraic optimization techniques for preference database queries and can easily be integrated in a Hill-Climbing algorithm.

Note that our approach does not intend to replace other optimization methods. Instead, our method can be combined with other algorithms to allow more efficient execution of *Preference Queries under Hard Sum Constraints* on relational database systems.

Background

In this section we want to give some background concerning the preference technology.

Preference Algebra

Preference frameworks tailored to standard database systems have been introduced in (Kießling 2002) and (Chomicki 2003). We depict the preference algebra from (Kießling 2002) which is a direct mapping to relational algebra and declarative query languages. This preference model is based on *strict partial orders* and is semantically rich, easy to handle and very flexible to represent user preferences which are ubiquitous in our life.

Definition 1. Preference

Let $A = \{A_1, \dots, A_n\}$ be a set of attribute names with corresponding domains of values $dom(A_i)$. The domain of A is defined as $dom(A) = dom(A_1) \times \dots \times dom(A_n)$. Then a preference P is a strict partial order $P = (A, <_P)$, where $<_P \subseteq dom(A) \times dom(A)$.

The term $x <_P y$ is interpreted as "I like y more than x ".

Having defined preferences as strict partial orders we provide a variety of intuitive and customizable **base preference constructors** for categorical and numerical domains which can intuitively be combined to build complex preferences still yielding partial orders. Formally, a base preference constructor has one or more arguments, the first characterizing the attributes A and the others the strict partial order $<_P$, referring to A .

For example, the POS-preference $POS(A, <_P)$ on an attribute A expresses that a special value of an attribute is preferred to all others (compare the IN-clause above). There is also a NEG-preference constructor which represents the disliked values of a person. Moreover, it is possible to combine these preferences to POS/NEG or POS/POS .

If we want to focus on preferences where the domain is a numerical data type, e.g. decimal, which can be infinite, we can use a number of *numerical preference constructors* which are defined by (Kießling 2002), e.g. LOWEST, HIGHEST, AROUND, BETWEEN and the SCORE preference. LOWEST(fat), for example, represents that a person prefers lower values for 'fat' over higher values.

Definition 2. Extremal preferences

We define the LOWEST and HIGHEST preference that the desired value should be as low (high) as possible. Formally:

- P is called a LOWEST preference, if: $x <_P y$ iff $x > y$
- P is called a HIGHEST preference, if: $x <_P y$ iff $x < y$

In the following we will briefly discuss a **complex preference constructor**, namely the *Pareto preference* (AND, \otimes).

Definition 3. Pareto preference: $P_1 \otimes P_2$

Given two preferences $P_1 = (A, <_{P_1})$ and $P_2 = (B, <_{P_2})$, for $x, y \in dom(A) \times dom(B)$ we define

$$x <_{P_1 \otimes P_2} y \text{ iff} \\ (x_1 <_{P_1} y_1 \wedge (x_2 <_{P_2} y_2 \vee x_2 = y_2)) \vee \\ (x_2 <_{P_2} y_2 \wedge (x_1 <_{P_1} y_1 \vee x_1 = y_1))$$

$P = (A \cup B, <_{P_1 \otimes P_2})$ is called *Pareto preference modelling* P_1 and P_2 as **equally important**.

A generalization of the Pareto preference constructor to more than two preferences is obvious. An extended definition can be found in (Preisinger and Kießling 2007).

Example 2: As in example 1, beef and fat as little as possible are equally important for Mrs. Diet. With preference algebra we describe her preferences as:

$$P = POS(Meats, \{Beef\}) \otimes LOWEST(fat)$$

There exist further complex preference constructors, e.g. the Prioritization, where a preference P_1 is considered **more important** than a preference P_2 . Detailed information on all preference constructors are given in (Kießling 2002).

The BMO Query Model

Given preferences over a set of attributes a central question is to determine an outcome that is preferentially optimal with respect to the preference statements. Whether preferences can be satisfied depends on the current database contents. Thus a match-making between wishes and data has to

be made. For this purpose the **Best-Matches-Only (BMO)** query model has been proposed by (Kießling 2002).

Definition 4. BMO-Set

The **Best-Matches-Only** result set contains only the best matches w.r.t the strict partial order of a preference P . It is a selection of unordered result tuples where all tuples in the BMO-set are undominated by others regarding the preference P .

In principle, efficient BMO query evaluation requires two new relational operators. We define

$$\sigma[P](R) := \{t \in R \mid \neg \exists t' \in R : t[A] <_P t'[A]\}$$

as **preference selection**. It finds all best matching tuples t for a preference $P = (A, <_P)$ with $A \subseteq \text{attr}(R)$ ¹. If none exists, it delivers *best-matching alternatives*, but *nothing worse*.

A preference can also be evaluated in grouped mode, given some $B \subseteq \text{attr}(R)$. This can be expressed as the **grouped preference selection**

$$\sigma[P \text{ groupby } B](R) := \{t \in R \mid \neg \exists t' \in R : t[A] <_P t'[A] \wedge t[B] = t'[B]\}.$$

$\sigma[P](R)$ and $\sigma[P \text{ groupby } B](R)$ can perform the match-making process as required by BMO semantics.

The Preference SUM-Constraint Problem

In this section we want to discuss the *Preference SUM-constraint problem* and we work out the idea behind our new approach. Consider Mrs. Diet’s preference query from example 1 in our preference model:²

$$\sigma[P_1 \otimes P_2 \otimes P_3] \sigma_{S.cal+M.cal+B.cal \leq 1100}(S \times M \times B)$$

with preferences

- $P_1 = POS(Soups, \{Chicken\})$
- $P_2 = POS(Meats, \{Beef\}) \otimes LOWEST(fat)$
- $P_3 = POS(Beverages, \{Red\ wine\})$

Conventional approaches implement such queries by a set of binary join operators and evaluate the hard sum constraint. Afterwards the user preferences as soft selection combined with the Pareto operator (\otimes) are evaluated by a skyline algorithm, e.g. (Börzsönyi, Kossmann, and Stocker 2001), to retrieve all combinations that fulfill the preferences best possible.

Because the hard constraint refers to attributes from more than two relations, pair-wise join operators may fail to remove intermediate results based on these condition. For a hard constraint such as $A_1 + \dots + A_r \Theta c$, $\Theta \in \{<, >, \leq, \geq, =, \neq\}$ and c a number, current join operators cannot test the satisfiability of an intermediate tuple until all variables have been determined. A consequence of this inability to remove intermediate tuples that will not lead to any results

¹We use $\text{attr}(R)$ to denote all attributes of a relation R

² $\sigma[P]$ means preference selection, i.e. a soft constraint, whereas σ_F denotes a classical relational algebra selection, i.e. a hard constraint.

is that the query evaluation process must ultimately evaluate the cartesian product of all tuples of all join relations, which lead to high memory and computation costs, particularly if the relations are large.

However, for efficient computation it may be helpful to apply the soft selection first (Hafenrichter and Kießling 2005; Chomicki 2003). Then we can neglect dominated tuples which do not satisfy the user’s preferences, maybe before evaluating the cartesian product. Building the cartesian product after this elimination is much faster, since the number of tuples in each relation maybe reduced. However, Mrs. Diet’s rewritten query

$$\sigma_{S.cal+M.cal+B.cal \leq 1100}(\sigma[P_1](S) \times \sigma[P_2](M) \times \sigma[P_3](B))$$

could produce an empty result set. Maybe there is no solution which perfectly satisfies all preferences **and** the hard sum constraint. If we first evaluate the preferences, we get the best-matches. But for the hard constraint, which is an absolute must, no combinations could exist.

Example 3: Considering the example database (table 1).

Table 1: Example database

Soups	ID	Name	Cal
	S1	Vegetable	59
	S2	Chicken	198
	S3	Noodle	453

Meats	ID	Name	Cal	Fat
	M1	Turkey	818	0.4
	M2	Pork	857	0.8
	M3	Beef	911	0.2

Beverages	ID	Name	Cal
	B1	Red Wine	85
	B2	Lemonade	181
	B3	Coke	400

Mrs. Diet’s rewritten preference query

$$\sigma_{S.cal+M.cal+B.cal \leq 1100}(\sigma[P_1](S) \times \sigma[P_2](M) \times \sigma[P_3](B))$$

gives an empty result, since

- $\sigma[P_1](Soup) = \{S2\}$
- $\sigma[P_2](Meats) = \{M3\}$
- $\sigma[P_3](Beverages) = \{B1\}$

and the sum of calories of this combination (1194 kcal) exceeds the hard constraint of $c \leq 1100$ kcal. On the other hand, building the cartesian product, selecting all combinations with the sum constraint below 1100 kcal and evaluating the preference query afterwards leads to the solution $\{S1, M3, B1\}$ with 1055 kcal.

This shows that a simple change of hard and soft constraints is not allowed. However, there is a possibility to ‘push the preferences’ over the cartesian product and execute them before evaluating the cartesian product.

In the next section we introduce a new approach for such preference SUM-constraint queries that enables search space refinement by inserting rewritten preferences into the cartesian product.

Preference Optimization

In this section we present new transformation laws for an efficient evaluation of *Preference SUM-constraint queries*.

Dominance Criterion

Since the target is to minimize the costs for computing the cartesian product, we introduce a *dominance criterion*. The dominance criterion allows us to eliminate dominated tuples from our relations which definitely can never be in the BMO-set. This can be done before building the cartesian product and therefore speeds up the evaluation.

Theorem 1. Dominance-Criterion

Assume a query

$$Q := \sigma[P_1 \otimes \dots \otimes P_r] \sigma_{\sum_{i=1}^r A_i \Theta c} (R_1 \times \dots \times R_r)$$

with

- $R_i(A_i, B_i)$, $i = 1, \dots, r$ database relations
- A_i numerical attributes
- $P_i = (B_i, <_{P_i})$ base preferences
- $\sum_{i=1}^r A_i \Theta c$
a SUM-constraint with $\Theta \in \{<, \leq, >, \geq, =, \neq\}$.

Let tuples $t, t' \in R_i$ such that

$$t[A_i] \hat{\Theta} t'[A_i] \wedge t[B_i] <_{P_i} t'[B_i] \quad (*)$$

and $\hat{\Theta}$ defined as

$$\hat{\Theta} := \begin{cases} \geq & \text{iff } \Theta \in \{\leq, <\} \\ \leq & \text{iff } \Theta \in \{\geq, >\} \\ = & \text{iff } \Theta \in \{=, \neq\} \end{cases}$$

Then an optimal solution exists for our Preference SUM-constraint query Q **without** the tuple $t \in R_i$, i.e. using the dominance criterion (*) for each relation, Q leads to a correct and complete solution.

Proof. Let $w := (t_1, \dots, t, \dots, t_r)$ and $v := (t_1, \dots, t', \dots, t_r)$ two tuples in $R := R_1 \times \dots \times R_r$ which only differ in t and t' with $t, t' \in R_i$ and

$$t[A_i] \hat{\Theta} t'[A_i] \wedge t[B_i] <_{P_i} t'[B_i]$$

Then, it is evident that:

- 1) if the hard constraint fails for v , also the hard constraint fails for w , since $t[A_i] \hat{\Theta} t'[A_i]$ (SUM is monotone). Therefore w is not an element of the solution.
- 2) if v fulfills the hard constraint, then
 - a) if w fails the hard constraint, then w is not an element of the solution.
 - b) if w also fulfills the hard constraint, then we know $t[B_i] <_{P_i} t'[B_i]$, i.e. tuple t' is preferred to t . It follows from the Pareto preference $P_1 \otimes \dots \otimes P_r$ that v is preferred over w since t' is preferred w.r.t P_i and all others are equal. \square

Example 4: Revisit example 1 with Mrs. Diet's hard constraint of maximal 1100 kcal (Θ is \leq) and her preferences concerning the chicken soup. Table 2 represents a simple soup relation.

Table 2: Example for the dominance criterion

Soups	ID	Name	Cal
	S1	Vegetable	59
	S2	Chicken	198
	S3	Noodle	453

Since 'S3' has more calories than 'S2' ($\hat{\Theta}$ is \geq) and 'S3' is worse than 'S2' concerning her soup preference, tuple 'S3' is dominated, i.e. we have not to consider 'S3' in the cartesian product and the hard selection. This results in the undominated tuples of 'S1' and 'S2'. Tuple 'S1' is worse than 'S2' concerning the same preference, but due to lower calories and the hard constraint we must take into account tuple 'S1'.

Transformation laws

(Hafenrichter and Kießling 2005) constructed a preference query optimizer as an extension of a classical Hill-Climbing algorithm. We added the dominance criterion to this preference query optimizer. For this we have to introduce a new complex preference constructor called CUTOFF.

Definition 5. CUTOFF Preference Constructor

Given preferences $\hat{P} = (A_1, <_{\hat{P}})$ and $P = (A_2, <_P)$, $x = (x_1, x_2)$, $y = (y_1, y_2) \in \text{dom}(A_1) \times \text{dom}(A_2)$.

Then $P_c := \text{CUTOFF}(\hat{P}, P)$, if:

$$x <_{P_c} y \text{ iff } (x_1 = y_1 \vee x_1 <_{\hat{P}} y_1) \wedge x_2 <_P y_2$$

CUTOFF is a partial order and therefore a preference according to (Kießling 2002).

For an evaluation of the dominance criterion from theorem 1 we use the $\text{CUTOFF}(\hat{P}, P)$ preference constructor described above within a BMO algorithm.

Corollary 1. Given a relation $R(A, B)$ with A a numerical attribute and $P = (B, <_P)$ a user preference. Then the preference constructor

$$P_c := \text{CUTOFF}(\hat{P}, P)$$

models the dominance criterion (*) from theorem 1 in preference algebra. For this we have to set \hat{P} as follows:

- if $\Theta \in \{\leq, <\}$, then $\hat{P} = \text{LOWEST}(A)$
- if $\Theta \in \{\geq, >\}$, then $\hat{P} = \text{HIGHEST}(A)$
- if $\Theta \in \{=, \neq\}$, then $\hat{P} = A^{\leftrightarrow}$

A^{\leftrightarrow} is called **anti-chain** preference on the attribute A and returns all elements of the input relation (Kießling 2002).

Proof. We prove this only for $\hat{P} = \text{LOWEST}(A)$. The other cases can be done analogously.

Using definition 5 we get for $\hat{P} = \text{LOWEST}(A)$

$$\begin{aligned} x <_{P_c} y & \text{ iff} \\ (x_1 = y_1 \vee x_1 <_{\hat{P}} y_1) \wedge x_2 <_P y_2 & \Leftrightarrow \\ x_1 \geq y_1 \wedge x_2 <_P y_2 & \end{aligned}$$

which corresponds to the dominance criterion (*). \square

With the help of theorem 1 and corollary 1 we can develop transformation laws for preference relational algebra that allow us to eliminate dominated tuples before building the cartesian product by *inserting the CUTOFF preference* into the query. For simplicity we reduced the number of preferences and relations to two, but all laws can be extended arbitrarily (cp. theorem 1).

Corollary 2. Insert CUTOFF into Cartesian Product

Let $R_i(A_i, B_i)$ be database relations, A_i numerical attributes, $P_i = (B_i, <_{P_i})$ base preferences, c a hard constraint and $\Theta \in \{<, \leq, >, \geq, =, \neq\}$. Then

$$\frac{\sigma[P_1 \otimes P_2] \sigma_{A_1+A_2 \Theta c}(R_1 \times R_2)}{\sigma[P_1 \otimes P_2] \sigma_{A_1+A_2 \Theta c}(\sigma[P_{c_1}](R_1) \times \sigma[P_{c_2}](R_2))}$$

where $P_{c_i} = CUTOFF(\hat{P}, P_i)$ and \hat{P} conditioned by Θ .

Proof. The proof is given by theorem 1. \square

In the case of joins like $R_1 \bowtie_{R_1.X=R_2.X} R_2$ we have to ensure that we do not eliminate join partners, i.e. for each tuple in the first relation there must exist a join partner in the second relation. To get rid of this problem we have to evaluate the CUTOFF preference as a **grouped preference selection**, see (Hafenrichter and Kießling 2005). Therefore, the CUTOFF preference is only evaluated for tuples in the same equivalence class, i.e. grouped by X .

Example 5: Revisit example 1 once more, but now we extend the query by the taste of the meal. The complete food combination should have the same taste, i.e. it should be bitter, salty, sour, sweet or umami ('savoury') and not mixed.

```
SELECT S.name, M.name, B.name
FROM Soups S, Meats M, Beverages B
WHERE S.cal + M.cal + B.cal <= 1100
      AND S.taste = M.taste
      AND M.taste = B.taste
PREFERRING ...
```

Applying the dominance criterion without grouping by the attribute 'taste' maybe eliminates all umami wines and no overall combination exists with an umami taste.

Corollary 3. Insert CUTOFF into Join

Let $R_i(A_i, B_i, X)$ be database relations, A_i numerical attributes, $P_i = (B_i, <_{P_i})$ base preferences, $X \subseteq attr(R_1) \cap attr(R_2)$, c a hard constraint and $\Theta \in \{<, \leq, >, \geq, =, \neq\}$. Then

$$\frac{\sigma[P_1 \otimes P_2] \sigma_{A_1+A_2 \Theta c}(R_1 \bowtie_{R_1.X=R_2.X} R_2)}{\sigma[P_1 \otimes P_2] \sigma_{A_1+A_2 \Theta c}(\sigma[P_{c_1} \text{ groupby } X](R_1) \bowtie_{R_1.X=R_2.X} \sigma[P_{c_2} \text{ groupby } X](R_2))}$$

where $P_{c_i} = CUTOFF(\hat{P}, P_i)$ and \hat{P} conditioned by Θ .

Proof. This is a consequence from theorem 1 and (Hafenrichter and Kießling 2005), theorem L6 and L7. \square

Notice, the selectivity of a groupby-preference depends on the distribution of the attribute X . In the worst case, X is a unique attribute and no tuples are dominated by the grouping operation. Otherwise, if the distribution of X is sparse, a good selectivity can be achieved.

Corollary 4. Further transformation laws

Let $R_i(A_i, B_i, X)$ be database relations, A_i numerical attributes, $P_i = (B_i, <_{P_i})$ a base preference, $X \subseteq attr(R_1) \cap attr(R_2)$, c a hard constraint and $\Theta \in \{<, \leq, >, \geq, =, \neq\}$. Then

- a) $\sigma[P_1] \sigma_{A_1 \Theta c}(R_1) = \sigma[P_1] \sigma_{A_1 \Theta c}(\sigma[P_{c_1}](R_1))$
- b) $\frac{\sigma[P_1] \sigma_{A_1+A_2 \Theta c}(R_1 \times R_2)}{\sigma[P_1] \sigma_{A_1+A_2 \Theta c}(\sigma[P_{c_1}](R_1) \times R_2)}$
- c) $\frac{\sigma[P_1] \sigma_{A_1+A_2 \Theta c}(R_1 \bowtie R_2)}{\sigma[P_1] \sigma_{A_1+A_2 \Theta c}(\sigma[P_{c_1} \text{ groupby } X](R_1) \bowtie R_2)}$

Proof. These transformation laws are direct consequences from corollary 2 and 3. \square

With the help of these new transformation laws the evaluation of such preference queries is evident: Our Preference SQL engine (Kießling and Köstler 2002) applies the CUTOFF preference independently on each stream of tuples. Afterwards it performs the cartesian product (checking the hard sum constraint) and finally the Pareto preference.

Remark: All transformation laws are also valid for a Prioritization instead of a Pareto preference in the selection query. We skipped this for simplicity and limited space.

Experimental Results

In order to evaluate our rewriting technique, we performed several experiments using real-world data. We used a food database published by the (USDA 2007). This database contains nutritional facts for more than 7000 types of food. From this database we created three relations: *Soups*, *Meats* and *Beverages* containing information about their eponymous types of food. The sizes of these relations are as follows: There are 433 soups, 1057 meats and 261 beverages available, i.e. about 120.000.000 possible combinations

We run all test queries on an Oracle 10.0 database system in combination with our Preference SQL engine (Kießling and Köstler 2002). The system is running on a Linux machine (Intel Dual Core CPU 1.6 GHz, 2GB main memory). We evaluated the efficiency of our rewriting technique by comparing the response times of several sum constraint queries with and without joins respectively **with and without rewriting**. **Due to limited space we only report one of them**, with representative performance, shown in figure 1 and 2. The test query is based on example 1 and contains a constraint requiring the sum of attributes *cal* to be less or equal than a value called **max_cal**.

```
SELECT S.name, M.name, B.name
FROM Soups S, Meats M, Beverages B
WHERE S.cal + M.cal + B.cal <= max_cal
PREFERRING
      S.name IN ('Chicken soup') AND
      M.name IN ('Beef') AND M.fat LOWEST AND
      B.name IN ('Red wine')
```

Notice, varying the parameter **max_cal** varies the selectivity of the query, while varying the size of the relations changes the size of the problem to be solved. Therefore, we varied the required calories (**max_cal**) in order to change the selectivity (see figure 1) and we varied the size of the relations (see figure 2).

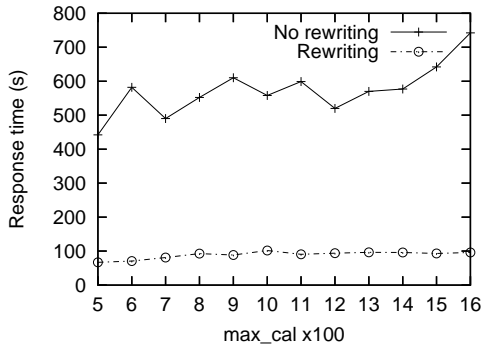


Figure 1: Performance results for different **max_cal**.

Since the dominance criterion only depends on the preferences and not on **max_cal**, the response time for the preference query with different **max_cal** is nearly constant. Obviously our rewriting techniques speeds-up the evaluation because of writing in the *CUTOFF* preference into the query.

Further on, we run our query with different relation sizes (but fixed **max_cal** = 1100) and demonstrate the performance results in figure 2.

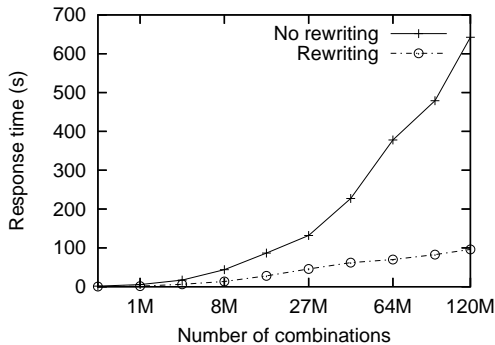


Figure 2: Performance results for different relation sizes.

From the experimental results of our benchmark queries, we can see that our proposed rewriting techniques improve the query performance consistently for different types of sum constraint queries.

Summary and Outlook

In this paper we have investigated *Preference SUM-constraint queries*. Despite its practical importance like e.g. in planning tasks or tourism, few optimization results have been known so far. Finding efficient query optimization techniques for this class of problems is beneficial for a variety of practical database applications. The key to our efficient evaluation is a dominance criterion and the new *CUTOFF* preference constructor which leads to algebraic transformation laws. For experimental results we used real-world data from the USDA. Performance comparison results with a standard evaluation approach demonstrates the enormous speed-ups that have been achieved. For future work we want to consider multiple constraints on multiple attributes belonging to several relations and we want to develop further transformation laws that lead to fast evaluation of preference

queries under hard constraints. Further on, our method is not limited to linear sum constraints, indeed any monotone function (e.g. the product) can be used. However, this needs a more formal specification and will be the next step.

References

- Agarwal, P. K.; Arge, L.; Erickson, J.; Franciosa, P. G.; and Vitter, J. S. 1998. Efficient Searching with Linear Constraints. In *PODS*, 169–178.
- Börzsönyi, S.; Kossmann, D.; and Stocker, K. 2001. The Skyline Operator. In *Proceedings of the 17th International Conference on Data Engineering (ICDE)*, 421–430. IEEE Computer Society.
- Chomicki, J. 2003. Preference Formulas in Relational Queries. In *ACM Transactions on Database Systems (TODS)*, volume 28, 427–466. ACM Press.
- Döring, S.; Preisinger, T.; and Endres, M. 2008. Advanced Preference Query Processing for E-Commerce. In *Proceedings of 23rd Annual ACM Symposium on Applied Computing (SAC)*, 1457–1462. ACM.
- Guha, S.; Gunopoulos, D.; Koudas, N.; Srivastava, D.; and Vlachos, M. 2003. Efficient approximation of optimization queries under parametric aggregation constraints. In *Proceedings of the 29th international conference on Very large data bases (VLDB)*, 778–789. VLDB Endowment.
- Hafenrichter, B., and Kießling, W. 2005. Optimization of Relational Preference Queries. In *The 16th Australasian Database Conference (ADC)*, 175–184.
- Ilyas, I. F.; Aref, W. G.; and Elmagarmid, A. K. 2003. Supporting Top-k Join Queries in Relational Databases. In *Proceedings of the 29th international conference on Very large data bases (VLDB)*, 754–765. VLDB Endowment.
- Kießling, W., and Köstler, G. 2002. Preference SQL - Design, Implementation, Experiences. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 990–1001.
- Kießling, W. 2002. Foundations of Preferences in Database Systems. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 311–322.
- Kießling, W. 2005. Preference Queries with SV-Semantics. In *Proceedings of the 11th International Conference on Management of Data (COMAD)*, 15–25.
- Liu, C.; Yang, L.; and Foster, I. 2005. Efficient Relational Joins with Arithmetic Constraints on Multiple Attributes. In *Proceedings of the 9th International Database Engineering & Application Symposium (IDEAS)*, 210–220. IEEE Computer Society.
- Nestorov, S.; Liu, C.; and Foster, I. T. 2007. Efficient Processing of Relational Queries with Sum Constraints. In *APWeb/WAIM*, 440–451.
- Preisinger, T., and Kießling, W. 2007. The Hexagon Algorithm for Evaluating Pareto Preference Queries. In *Proceedings of the Multidisciplinary Workshop on Advances in Preference Handling (VLDB)*.
- USDA. 2007. USDA national nutrient database for standard reference. <http://www.nal.usda.gov/fnic/>.