Consistently Adding Amalgamations to Preference Orders

Ulrich Güntzer
Institut für Informatik
University of Tübingen
Tübingen, Germany
+49 (7071) 29 70437
ulrich.guentzer@informatik.uni-tuebingen.de

Wolf-Tilo Balke
L3S Research Center
University of Hannover
Hannover, Germany
+49 (511) 76217712
balke@l3s.de

ABSTRACT

The use of user preferences for database retrieval promises a high degree of personalization by cooperatively combating the information flood. But most of the time preferences are only given for a subset of the available domain values and thus can still lead to unmanageable result sets especially in the presence of many attributes in a query. To limit such result sets preferences need to be completed interactively in tight cooperation with the user. In our framework the user is therefore enabled to specify additional preference information and also explicit equivalences between certain domain value combinations. However, whenever new information has to be integrated into the retrieval process the system has to make sure that the newly specified information is also consistent with what has been specified before. In this paper we study sufficient conditions for testing the consistency of new preference and equivalence information into a preference-based retrieval process. The benefits are that if the preference information fulfills a quite intuitive condition (which we call Cartesian) all new information can simple be added to the existing information without need for further consistency checks.

1. INTRODUCTION

Cooperative retrieval techniques based on qualitative user preferences have gained a lot of attention in the database community [14], [11], [7]. Realizing a cooperative system behavior queries can be posed in the most user-specific way and in case that no suitable object should be available in the database instance they can be gradually relaxed by the system. Thus, the tedious trial and error querying between empty result set and huge result sizes is avoided. In database research the term 'skyline queries' has been adopted for this class of queries (see e.g., [5], [15], or, [3]) and has recently been extended from using only totally ordered domains to categorical domains [6], [2]. Basically skyline queries allow users to provide a (partial) order of (a subset of) the domain values of each attribute used in a query. Then as a result set the Pareto optimal set with respect to all attributes is retrieved.

However, Skyline results are usually still too big because of the exponentially growing possibilities to create pairwise incomparable objects with respect to the Pareto semantics with growing numbers of attributes in the query [9]. Recent studies show that even for small to medium sized databases result set sizes for as little as five to six independent attributes in the queries are often already in the range of several thousand objects [3]. This problem is even more demanding in face of anti-correlations between pairs of attributes [5]. Though some approaches try to ease the problem of skyline manageability by exploiting suitable structural properties of the preferences e.g. the membership in skylines of lower dimensionality [16] or weak Pareto dominance [2], incorporating additional user feedback still ranges among the more helpful techniques see e.g., [13], [4], or [8].

With respect to query refinement the easiest way of integrating new information is defining new preference information in a single attribute. Since the preferences with respect to each attribute are generally given as partial orders, the user can add new domination relationships or explicitly resolve incomparability relationships by defining equivalences. We have proposed a comprehensive framework for modeling such changes and checking their consistency with already existing preference information in [1]. In a nutshell, given a query containing *n* attributes a new domination relationship between domain values $a < b$ in a preference P_i is extended on object level to all object pairs that have the values *a* and *b* to the *i*-th attribute respectively, and share exactly the same domain value with respect to the remaining (*n*-1) attributes. The characteristic is often referred to as *ceteris-paribus* semantics: in the case that (*n*-1) attribute values are equal, the *n*-th attribute decides about object domination [10].

On the other hand new information on only individual attributes might be a too limited option for adding preference information. Often users want to introduce preferences/equivalences with respect to several attributes. This includes relationships between specific value combinations (so-called *amalgamated* attributes, cf. [1]) and even trade-offs between several preferences. In particular the latter extend the Pareto aggregation semantics, because in a trade-off the value tuples restricted to the amalgamated attributes can be equivalent, or form a domination relationship without actually being strictly better or equivalent in each of the attributes. But also for amalgamated attributes the ceteris paribus semantics has to be used to expand to all attributes with all possible value combinations for the non-amalgamated attributes. The resulting relationships between the completed objects have to be added to the aggregated preference relation. This means a batch of insertions into the aggregated preference relation.

To guarantee that the resulting preferences on object level are still valid (especially: free of cycles) we have to check the overall consistency of the aggregated relation after the batch insert. In [1] we have given criteria for stating the consistency after a single insert of a new domination relationship or equivalence statement. Naively we could check these criteria for every single insert of our batch, but of course this method is computationally prohibitive. In this paper we present a more suitable criterion for checking the consistency of an aggregated preference relation after a batch of preferences/equivalences on object level has been inserted. Moreover, we will prove that this criterion is sufficient and can be efficiently checked.

This paper is organized as follows: in section 2 we will give a short overview of our basic preference model and will motivate the usefulness of attribute amalgamations. Section 3 will deal with a sufficient condition for integrating new preference or equivalence amalgamations into the already stated information. We close with a short summary and outlook.

2. THEORETICAL FOUNDATIONS

2.1 Basic Preference Framework

Following previous work in the area of preference-based database retrieval [10], [7] our framework builds on *qualitative preferences* in the form of strict partial orders of domain values. Given n attributes in the query we thus have to consider n partial order preferences $P_1, \ldots, P_n$. We complement these so-called *base preferences* by suitable *domain value equivalences* $Q_1, \ldots, Q_n$ that at least contain identity (i.e. if two database objects share the same domain value with respect to the i-th attribute, they are trivially part of the equivalence relation Q_i). Of course the equivalence relations may also contain more relationships that can be either structural (cf. [2]) or explicitly user defined. However, we will always require equivalence relations to be *compatible* with the respective preference relations, meaning that no equivalence in Q_i contradicts any strict preference in P_i and that domination relationships expressed transitively using P_i and Q_i must always be contained in P_i.

For evaluating skyline queries such base preferences and base equivalences are usually integrated on object level using the Pareto semantics as a fair and intuitive way of aggregation. The Pareto semantics states that an object x dominates another object y (written $x > y$), if x's attribute values are better than or equivalent to the y's attribute value with respect to all n attributes and strictly better with respect to at least one attribute. In general the *aggregated object preference relation*, which we will call P in the following, should always contain the aggregated preference information following the Pareto semantics. We will denote this basic set of preference relationships with respect to a database instance O and n attributes as $Pareto(O, P_1, \ldots, P_n, Q_1, \ldots, Q_n)$. For brevity in the following we generally use the notation $Pareto(O, P_I, Q_I)$ for Pareto aggregation of a database instance O for a set of indices I using all base preferences and base equivalences with an index in I. Of course also the equivalence relations can be aggregated to an *equivalence relation on object level* (which we will call Q in the following) by adding relationships between all database objects that share equivalent attribute values with respect to all n attributes. The set Q is automatically compatible with P.

One of the characteristics implied by the Pareto semantics is separability: each preference on complex objects can be broken down to preferences on the individual attributes. In fact using a Pareto aggregation a database object can only dominate another object, if it is really better or equivalent with respect to *all* base preferences. Therefore, given a preference on object level will always determine adequate preferences on the individual base preferences. In the following we will step by step relax the separability assumption for our modeling framework.

To offer a maximum of flexibility to the user the object level preference P can even extended beyond the Pareto semantics. In our framework users can add more domination relationships, for instance domination relationships explicitly modeled by users on specific domain value combinations only. Of course such new domination relationships have to be *consistent* with all already specified relationships meaning that the aggregated object preference relation must never contain conflicting relationship statements in the form of cycles. In the same way also the equivalence relation can be extended. The *skyline result set* then can be defined as the set of all database objects that are not dominated by any other database object with respect to P. For a detailed discussion of all these concepts see [1].

To illustrate the above let us consider a short example given by Figure 1. Assume a relational database table characterized by three attributes, where the first attribute can have domain values A_0 or A_1, the second B_0, B_1, B_2, or B_3 and the last attribute C_0 or C_1. Every database object is thus characterized by three values. With respect to each attribute a user a user might have preferences on the domain values. For instance a user might prefer database object with domain value A_0 over those with A_1 and those with value B_0 over those with B_1, on the other hand the user might be indifferent between values B_1 and B_2. The three partial orders given in Figure 1 reflect such base preference statements for each attribute. On object level the preferences can be aggregated using the Pareto semantics, for instance object (A_0, B_0, C_0) is considered better with respect to the aggregated preference P than object (A_1, B_3, C_1). In fact, if an object (A_0, B_0, C_0) exists in the database instance it dominates *all* other objects with at least some differing attributes value. Hence, in this case the skyline would only consist of objects of type (A_0, B_0, C_0).

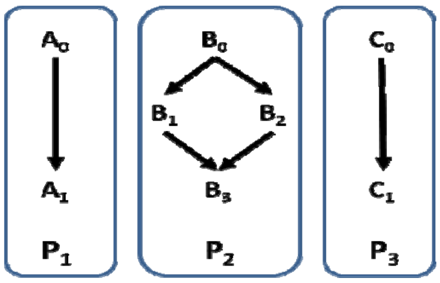

Figure 1. Three base preferences.

Likewise, a user might explicitly state that he/she considers the attribute values B_1 and B_2 as being equally desirable. This equivalence statement is compatible with preference P_2, since no preference relationship is stated with respect to B_1 and B_2 and all induced transitive domination relationships are already stated in P_2.

Using the user-provided equivalence information we can derive equivalence relationships e.g. between (A_0, B_1, C_0) and (A_0, B_2, C_0) in *Q*. But due to A_0 dominating A_1 and B_1, B_2 being equivalent, we can also derive a dominance relation in *P* between objects (A_0, B_1, C_0) and (A_1, B_2, C_0). Please note that this last relationship is not part of *P* under strict Pareto semantics, i.e. Pareto semantics where each Q_i is the equality relation on the *i*-th attribute.

2.2 Dealing with Amalgamated Attributes

Moreover, often users want to introduce new relationships with respect to only a subset of the attributes. For instance, a user might not want to state a general equivalence between B_1 and B_2 in the above example, but the equivalence should only hold for a value of A_0 with respect to the first attribute. Then he/she might explicitly define an equivalence between (A_0, B_1, C_0) and (A_0, B_2, C_0), as well as between (A_0, B_1, C_1) and (A_0, B_2, C_1). Due to the Pareto semantics this in turn also leads to the new domination relationships (A_0, B_1, C_0) > (A_0, B_2, C_1) and (A_0, B_2, C_0) > (A_0, B_1, C_1).

A special case are so-called *trade-offs* where a user can explicitly add relationships extending the expressivity of the basic Pareto semantics on the cost of separability like for example that the combination between A_0 and C_1 is preferred over A_1 and C_0. Whenever new relationships involve more than one attribute we call the respective attributes *amalgamated attributes*. To be self-contained we will give the formal definition of amalgamated preferences and equivalences from [1]:

Definition 1: (Amalgamated Preferences and Equivalences Functions)
Let $\mu \subseteq \{1, \dots, m\}$ be a set with cardinality *k* and the complement $\overline{\mu} := \{1, \dots, m\} \setminus \mu$. Further let π_I be the projection in the sense of relational algebra on all attributes with index in I, we define the function:
$\boldsymbol{AmalPref(x_\mu, y_\mu)} : (\times_{i \in \mu} D_i)^2 \to O^2$ where $(x_\mu, y_\mu) \mapsto$
$\{(o_1,o_2) \in O^2 \mid (\pi_\mu(o_1) = x_\mu) \wedge (\pi_\mu(o_2) = y_\mu) \wedge (\pi_{\overline{\mu}}(o_1) = \pi_{\overline{\mu}}(o_2))\}$
as well as
$\boldsymbol{AmalEq(x_\mu, y_\mu)} : (\times_{i \in \mu} D_i)^2 \to O^2$ where $(x_\mu, y_\mu) \mapsto$
$\{(o_1,o_2) \in O^2 \mid [(\pi_\mu(o_1) = x_\mu \wedge \pi_\mu(o_2) = y_\mu) \vee (\pi_\mu(o_1) = y_\mu \wedge \pi_\mu(o_2) = x_\mu)] \wedge (\pi_{\overline{\mu}}(o_1) = \pi_{\overline{\mu}}(o_2))\}$

This means: Given two tuples x_μ, y_μ from the same amalgamated domains of attributes described by μ, the function *AmalPref*(x_μ, y_μ) returns a set of relationships between database objects of the form (o_1, o_2) where the attributes of o_1 projected on the amalgamated domains equal those of x_μ, the attributes of o_2 projected on the amalgamated domains equal those of y_μ and furthermore all other attributes which are not within the amalgamated attributes are identical for o_1 and o_2. Similarly the amalgamated equivalence contains all object pairs where attribute values coincide with x_μ, resp. y_μ, in the amalgamated attributes and the same values are shared with respect to the remaining attributes. In both cases the last requirement denotes the well-known *ceteris paribus* condition ("all other things being equal", see e.g. [10]).

Basically, whenever two or more attributes have been amalgamated, the Pareto semantics and transitivity can be used to derive all new domination relationships extending the original *P*. Assume for instance that the first and the third preference P_1 and P_3 have been amalgamated by stating that the combination between A_0 and C_1 is preferred over A_1 and C_0. Then both base preferences are transformed into a single preference P_{13} and the set of valid Pareto domination relationships (projected on the two preferences) {(A_0, C_0) > (A_0, C_1), (A_0, C_0) > (A_1, C_0), (A_0, C_0) > (A_1, C_1), (A_0, C_1) > (A_1, C_1), (A_1, C_0) > (A_1, C_1)} is extended by {(A_0, C_1) > (A_1, C_0)}. Now a Pareto aggregation between P_{13} and P_2 will lead to the correct aggregated object preference relation *P* that can be used to evaluate the respective skyline query.

As we have seen, a new domination relationship extending the Pareto semantics has been added to P_{13}. But now the new preference cannot be factorized into the base preferences P_1 and P_3 anymore, because a simple re-aggregation into P_{13} using the Pareto semantics would lose the information that (A_0, C_1) > (A_1, C_0). Therefore, having performed an amalgamation between P_1 and P_3, it is difficult to perform another amalgamation with any of the factors, for example between P_1 and P_2. One possibility to keep all factors separate in the face of amalgamations is to add all necessary preference/equivalence relationships on object level. That means, the original *P* is calculated once using the Pareto semantics (generalized by allowing equivalence information) and then only extended by new relationships between objects that are induced by additional preferences, equivalences or amalgamations.

This procedure is straightforward if explicit relationships between objects are added. However, since base preferences/equivalences and amalgamations define relationships only on a subset of the attributes, the object relationships have to be derived. Here again the notion of *ceteris paribus* comes into play. If a preference/equivalence has been specified on an attribute subset only (like above for P_1 and P_3) the object-level preferences/equivalences have to be added for every domain value combination for every remaining preference (in the above case for all possible values of P_2, i.e. specifying the trade-off (A_0, C_1) > (A_1, C_0) leads to adding object-level preferences {(A_0, B_0, C_1) > (A_1, B_0, C_0)}, (A_0, B_1, C_1) > (A_1, B_1, C_0)}, (A_0, B_2, C_1) > (A_1, B_2, C_0)}, (A_0, B_3, C_1) > (A_1, B_3, C_0)} to *P*).

As we can see each new piece of information stated by the user thus results in a batch of relationships, which have to be integrated into the already existing information. In any case, the new relationships created may only be incorporated into *P* and *Q* as long as they don't violate *P*'s or *Q*'s consistency. In [1] we have in detail investigated necessary and sufficient criteria for checking the consistency of a single new relationship between objects with already existing preference/equivalence information. Thus, one obvious way of performing the consistency check for our amalgamations here would be to create all necessary new relationships following the ceteris paribus semantics and then checking the consistency for each single statement successively. Only if all statements can be added consistently, the amalgamation can be integrated with the existing information.

Of course even if only a few dimensions are not part of the amalgamation, the number of statements to check is the product of their respective domain value cardinalities. This is even more difficult for numerical domains where the values have to be restricted to those actually occurring in the respective database instance. Hence, this method is obviously computationally prohibitive and we have to find an easier test, whether a batch of new relationships can be consistently added to the existing information. The next section will address this problem in more detail.

2.3 Pareto Aggregation and Trade-Offs

We have seen the Pareto aggregation to be a very intuitive concept, which, however, by design cannot deal with trade-offs. The reason is that the Pareto paradigm is completely compositional with respect to the attributes, i.e. if all preferences for each individual attribute are known, then also the product preference is known and vice versa. For illustration assume three attributes and preferences P_1, P_2, and P_3, as well as equivalences Q_1, Q_2, and Q_3 on these attributes. Then for a Pareto aggregation P and objects (x_1, x_2, x_3) and (y_1, y_2, y_3), we always have:

$$((x_1, x_2, x_3), (y_1, y_2, y_3)) \in P \;\Leftrightarrow\; \forall\, 1 \le i \le 3: (x_i, y_i) \in P_i \cup Q_i \;\wedge\; \exists\, i: (x_i, y_i) \in P_i$$

and $\quad ((x_1, x_2, x_3), (y_1, y_2, y_3)) \in Q \;\Leftrightarrow\; \forall\, 1 \le i \le 3: (x_i, y_i) \in Q_i$

However, the compositional character has to be sacrificed for higher expressiveness in the face of trade-offs. Although the product preference should always contain the Pareto aggregation (i.e. the '$\Leftarrow$' part of the above equivalences always holds, which is a very natural assumption), once a trade-off has been specified the product preference P is enhanced beyond Pareto aggregation and generally violates the '$\Rightarrow$' direction (i.e. some new product preference relationships may not extend to the individual attribute preferences).

The new product preference thus introduces higher expressiveness at the cost of more complexity. Still this complexity is restricted by postulating that none of these new preference/equivalence relationships conflicts with already stated information.

3. GUARANTEEING CONSISTENCY

In this section we will define a new sufficient criterion for checking if amalgamated preferences/equivalences are consistent with already specified information. We prove that if the new relationships show a special characteristic for all possible extensions, which we call Cartesian characteristic, the new information can be consistently added to what has been specified before. Let us first define the concept of Cartesian preferences and equivalences:

Definition 2: (Cartesian preference/equivalence for preference amalgamations)
Let $\mu \subset \{1, \ldots, m\}$ be a set with cardinality $k < m$, and the complement $\overline{\mu} := \{1, \ldots, m\} \backslash \mu$. Further let π_I be the projection in the sense of relational algebra on all attributes with index in I. For some k-dimensional amalgamated preference $(y_\mu, x_\mu) \in (\times_{i \in \mu} D_i)^2$ we call preference and equivalence relation P and Q *Cartesian with respect to preference* (y_μ, x_μ), if and only if for all $(x, y) \in (P \cup Q)$ such that $(\pi_\mu(x) = x_\mu) \wedge (\pi_\mu(y) = y_\mu)$ holds: $(\pi_{\overline{\mu}}(x), \pi_{\overline{\mu}}(y)) \in Pareto(O, P_{\overline{\mu}}, Q_{\overline{\mu}})$

The idea behind this criterion is related to the general notion of the Pareto semantics and the idea of trade-offs. Consider a dominance relationship for certain domain values $y_\mu > x_\mu$ with respect to the amalgamated attributes. If, however, any two database objects showing these specific attribute values are nevertheless in the *reverse domination relationship* $x > y$ when considering all attributes, then the relationship between all non-amalgamated attributes must have 'compensated' the lack of preference in the amalgamated attributes. Therefore the non-amalgamated attribute value combination should be in a preference relationship following the Pareto semantics $\pi_{\overline{\mu}}(x) > \pi_{\overline{\mu}}(y)$. Being Cartesian thus is a quite natural concept.

Also for the case of equivalences specified over several amalgamated attributes we can define the notion of being Cartesian:

Definition 3: (Cartesian preference/equivalence for equivalence amalgamations)
Let $\mu \subset \{1, \ldots, m\}$ be a set with cardinality $k < m$, and the complement $\overline{\mu} := \{1, \ldots, m\} \backslash \mu$. Further let π_I be the projection in the sense of relational algebra on all attributes with index in I. For some k-dimensional amalgamated equivalence $(y_\mu, x_\mu) \in (\times_{i \in \mu} D_i)^2$ we call preference and equivalence relation P and Q *Cartesian with respect to* (y_μ, x_μ), if and only if for all $(x, y) \in (P \cup Q)$ such that $(\pi_\mu(x) = x_\mu \vee \pi_\mu(x) = y_\mu) \wedge (\pi_\mu(y) = y_\mu \vee \pi_\mu(y) = x_\mu)$ holds: $(\pi_{\overline{\mu}}(x), \pi_{\overline{\mu}}(y)) \in (Pareto(O, P_{\overline{\mu}}, Q_{\overline{\mu}}) \cup \times_{i \in \overline{\mu}} Q_i)$
and if $(x, y) \in P$, then $(\pi_{\overline{\mu}}(x), \pi_{\overline{\mu}}(y)) \in Pareto(O, P_{\overline{\mu}}, Q_{\overline{\mu}})$

Again this condition is quite natural: if two database objects x and y are considered equivalent or x is preferred over y , but we know that their amalgamated part is definitely considered equivalent by the user, then the non-amalgamated part has to cause the overall relationship, i.e. it also has to be considered equivalent or x's attribute values have to dominate y's. In the case that x is definitely preferred over y, the non-amalgamated components even do have to be in a domination relationship.

It is obvious due to separability that the simple preference/and equivalence relation induced by Pareto aggregation (without any amalgamations) is always Cartesian with respect to any subset of attributes. Whatever subset is chosen, to be in the Pareto preference/equivalence relation a tuple has also to be part of the preference/equivalence relation projected on the subset's complement. However, in the case of amalgamations we lose the separability characteristic, thus preference/equivalence relations will not always be Cartesian.

3.1 Motivation of the Cartesian Semantics

To better understand what it means that a preference/equivalence is Cartesian with respect to preference/equivalence let us consider a short example of a non-Cartesian preference/equivalence under the conditions of definition 3. Assume three attributes and preferences P_1, P_2, and P_3, as well as equivalences Q_1, Q_2, and Q_3 on these attributes. Moreover let us assume that a user has introduced a trade-off equivalence $Q_{1,2}$ with respect to the first two attributes such that $(y_1, y_2) \approx (x_1, x_2)$.

If the product preference P would not be Cartesian, there would exist attribute values x_3 and y_3 such that:

- if $((x_1, x_2, x_3), (y_1, y_2, y_3)) \in Q$, then $(x_3, y_3) \notin P_3 \cup Q_3$,
- if $((x_1, x_2, x_3), (y_1, y_2, y_3)) \in P$, then $(x_3, y_3) \notin P_3$.

In both case either $(y_3, x_3) \in P_3$ holds, or x_3 and y_3 are incomparable with respect to P_3 (also in the second case $(x_3, y_3) \notin Q_3$, since otherwise with $Q_{1,2}$ and Q_3 the pair $((x_1, x_2, x_3), (y_1, y_2, y_3))$ would have to be in Q). Now for the sake of contradiction let us assume that $(y_3, x_3) \in P_3$ holds. Then due to the ceteris paribus semantics we know $((x_1, x_2, y_3), (x_1, x_2, x_3)) \in Pareto(O, P_1,\ldots,P_3,Q_1,\ldots,Q_3) \subseteq P$. Since on the other hand $((y_1, y_2, y_3), (x_1, x_2, y_3)) \in Q$ due to ceteris paribus, we could derive $((y_1, y_2, y_3), (x_1, x_2, x_3)) \in P$ in contrast to our two cases above. Hence, x_3 and y_3 are really incomparable with respect to P_3. That means in a non-Cartesian preference/equivalence an *incomparable pair of values* can cause an equivalence relationship, or (what is even more surprising) a strict preference relationship between certain objects which are equivalent with respect to the amalgamated parts. Generally excluding these non-intuitive cases from our consideration does indeed seem sensible.

Having justified the concept of Cartesian preferences/equivalences let us now turn to the problem whether an amalgamation can be added to a preference and corresponding equivalence relation consistently. Since the entire concepts are independent of the specific order of the attributes, let us for ease of notation and without loss of generality assume for the remainder of this paper that the first k attributes have been amalgamated and the last $(m\text{-}k)$ attributes are non-amalgamated. Further like above we will denote the set of indexes for amalgamated attributes as μ and the respective complement as $\bar{\mu}$. In the following we always will denote the existing preference/equivalence relations P and Q and assume them to be Cartesian with respect to an amalgamation (y_μ, x_μ).

As we have seen in section 2 respecting the ceteris paribus semantics requires us to add all possible extensions of an amalgamation to the preference/equivalence relation. There are two cases: either the amalgamation specified a preference between several attributes, or the amalgamation specified an equivalence between the attributes. We will deal with both cases separately.

3.2 Preferences on Amalgamations

For an amalgamation (y_μ, x_μ) that expresses a preference between y_μ and x_μ we will denote the set of relationships that have to be added under the ceteris paribus semantics as:

$$S := \{[(y_\mu, z), (x_\mu, z)] \mid z \in (\times_{i\in\bar{\mu}} D_i)\}.$$

Although in principle relationships in this set can be added incrementally one by one, it is necessary to know in advance, if the entire set can be added; otherwise effort will be wasted. In [1] theorem 1 states as a criterion under which new relationships can be consistently added that once the new preference information in S has been added to the existing information $(P \cup Q)$, the new set $(P \cup Q \cup S)$ is not allowed to contain cycles involving any preference information, i.e. an edge from $(P \cup S)$. The next theorem will show that this can never be the case under the property of the relations being Cartesian with respect to the amalgamation.

Theorem 1: (Adding preference amalgamations)
Let P, Q, S be as above and P, Q be Cartesian with respect to an amalgamation (y_μ, x_μ) that defines $y_\mu > x_\mu$. Then no cycle in $(P \cup Q \cup S)$ contains an element from $(P \cup S)$.

Proof: We have to show that any cycle with edges from $(P \cup Q \cup S)$ can only consist of edges in Q. Let assume that there exists a cycle C in $(P \cup Q \cup S)$. If the cycle does not contain an edge from S, there is nothing to show since $(P \cup Q)$ can only have cycles with edges from Q. For the sake of contradiction therefore let us assume that there exists j edges from S in the cycle of the form $[(y_\mu, z_i), (x_\mu, z_i)]$ $(1 \leq i \leq j)$ and if we cut the cycle open at $[(y_\mu, z_1), (x_\mu, z_1)]$ that they occur in this order.

Between each two edges $[(y_\mu, z_i), (x_\mu, z_i)]$ and $[(y_\mu, z_{i+1}), (x_\mu, z_{i+1})]$ there are only edges from $(P \cup Q)$. Thus using transitivity we can contract them into a single edge of the form $[(x_\mu, z_i), (y_\mu, z_{i+1})]$ and this edge is also in $(P \cup Q)$. Since P and Q are Cartesian with respect to (y_μ, x_μ), we always get $(z_i, z_{i+1}) \in Pareto(O, P_{\bar{\mu}}, Q_{\bar{\mu}})$. However, because C is a cycle we have $z_{j+1} = z_1$ and hence there would exist a cycle in $Pareto(O, P_{\bar{\mu}}, Q_{\bar{\mu}})$, which is a contradiction to the Pareto semantics. ■

3.3 Equivalences on Amalgamations

For an amalgamation (y_μ, x_μ) that expresses an equivalence between y_μ and x_μ we will have to add all symmetric relationships under the ceteris paribus semantics as:

$$E := \{[(y_\mu, z), (x_\mu, z)] \mid z \in (\times_{i\in\bar{\mu}} D_i)\}$$
$$\cup \{[(x_\mu, z), (y_\mu, z)] \mid z \in (\times_{i\in\bar{\mu}} D_i)\}.$$

Again we have to show that under the notion of Cartesian relations the new information can always be added consistently to what has been specified before.

Theorem 2: (Adding equivalence amalgamations)
Let P, Q, S be as above and P, Q be Cartesian with respect to an amalgamation (y_μ, x_μ) that defines y_μ and x_μ to be equivalent. Then no cycle in $(P \cup Q \cup E)$ contains an element from P.

Proof: The proof is similar to the proof of theorem 1, but due to the value substitutability introduced by equivalence of the amalgamated part, there are more possibilities to construct cycles. This time we have to show that any cycle with edges from $(P \cup Q \cup E)$ can only consist of edges in $(Q \cup E)$.

Let assume that there exists a cycle C in $(P \cup Q \cup E)$. If the cycle does not contain an edge from E, there is nothing to show since $(P \cup Q)$ can only have cycles with edges from Q. For the sake of contradiction therefore let us again assume that there exist j edges from E in the cycle. But since y_μ and x_μ are considered equivalent the edges can have the form $[(y_\mu, z_i), (x_\mu, z_i)]$, $[(x_\mu, z_i), (y_\mu, z_i)]$ $(1 \leq i \leq j)$.

Again, between each two edges there are only edges from $(P \cup Q)$. But this time there are four possible cases between which these edges are enclosed: between $[(y_\mu, z_i), (x_\mu, z_i)]$ and $[(y_\mu, z_{i+1}), (x_\mu, z_{i+1})]$, or between $[(x_\mu, z_i), (y_\mu, z_i)]$ and $[(x_\mu, z_{i+1}), (y_\mu, z_{i+1})]$, or between $[(x_\mu, z_i), (y_\mu, z_i)]$ and $[(y_\mu, z_{i+1}), (x_\mu, z_{i+1})]$, or between $[(y_\mu, z_i), (x_\mu, z_i)]$ and $[(x_\mu, z_{i+1}), (y_\mu, z_{i+1})]$.

Depending on the case and again using transitivity we can contract them into a single edge of either one of four forms $[(x_\mu, z_i), (y_\mu, z_{i+1})]$, $[(y_\mu, z_i), (x_\mu, z_{i+1})]$, $[(x_\mu, z_i), (x_\mu, z_{i+1})]$, or $[(y_\mu, z_i), (y_\mu, z_{i+1})]$. and either of these edges is also in $(P \cup Q)$. Since P and Q are Cartesian with respect to (y_μ, x_μ), we always get $(z_i, z_{i+1}) \in Pareto(O, P_{\bar{\mu}}, Q_{\bar{\mu}}) \cup \times_{i\in\bar{\mu}} Q_i$.

Now, if the cycle would have an element from P, one of the intermediate edge combinations would be in P, too. Therefore the corresponding (z_i, z_{i+1}) would lie in $Pareto(O, P_{\bar{\mu}}, Q_{\bar{\mu}})$. However, because C is a cycle we have $z_{j+1} = z_1$ and hence there would also exist a cycle in $Pareto(O, P_{\overline{\mu}}, Q_{\overline{\mu}}) \cup \times_{i \in \overline{\mu}} Q_i$ with at least one strict edge, which again is a contradiction to the Pareto semantics. ■

Now we have shown for both cases preference amalgamations and equivalence amalgamations that the intuitive condition of being Cartesian with respect to the amalgamation is already sufficient to allow for a consistent integration of the trade-off information with what a user has specified before. Please note, however, that although being sufficient, the condition is not necessary.

3.4 A Test for the Cartesian Condition

One disadvantage of the conditions for checking the consistency of a preference/equivalence relationships that have been presented in [1], is that they have to be tested for every single relationship. Given the need of observing the ceteris paribus semantics for extending amalgamation, this can result in a high number of possible relationships. In contrast, the test if a preference/equivalence relation is Cartesian with respect to a certain amalgamation can be performed set-based over the database instance. For instance, the work in [12] presents PreferenceSQL, an extension of standard SQL to evaluate skyline queries over relational databases. When evaluating the Pareto-optimal result set, PreferenceSQL uses temporal views to store tuples under consideration for the skyline and checks the direct and transitive domination relationships on the attributes as given by the query.

Basically, once the domination relationships are stored in a table and the Pareto aggregation can be computed on the fly, the test whether a preference/equivalence relation is Cartesian with respect to an amalgamation can be performed by a simple SQL query. After calculating and storing the Pareto aggregation for the non-amalgamated attributes in the query in a (temporary) view, we have to select all possible extensions in the database instance. Therefore, all object pairs have to be selected from the database, whose attribute values equal the values of the respective attributes specified in the amalgamation. Only if the difference of this set of object pairs with the object pairs stored in the view is empty, then the preference/equivalence relation is Cartesian with respect to the amalgamation. In this way we can profit from the set-oriented evaluation of SQL and PreferenceSQL.

4. SUMMARY AND OUTLOOK

In database retrieval skyline queries promise intuitive querying for the user, because the user just specifies the set of attributes he/she is interested in and a set of basic preferences (in the form of strict partial orders), but does not have to specify complex utility functions for ordering the result set. The skyline result then is the Pareto-optimal set of ‘best’ objects that are not dominated by any other object in the database instance. However, the ease of use comes at a price: since all attributes are considered to be of equal importance skyline results tend to contain large portions of the database instance for higher numbers of attributes in the query.

To ease this problem we advocate the elicitation of additional preference/equivalence information from the user. Moreover, in our framework a user is even enabled to specify new information with respect to a subset of the attributes using so-called amalgamations. But since to the ceteris paribus semantics has to be observed, specifying an amalgamation leads to the necessary integration of preference/equivalence relationships for all possible extensions of the amalgamated values with respect to the database instance. Of course once new information is introduced by the user, the system has to check whether this information is consistent with what has been specified before, to avoid cyclic preferences in the aggregated preference relation.

In this paper we have presented a sufficient and natural condition under which new preference or equivalence information with respect to amalgamations can consistently be integrated into the aggregated preference relation. We have introduced the concept of a preference relation being *Cartesian* with respect to an amalgamation, which in the presence of preferences between database objects enforces some simple conditions on the non-amalgamated attributes. Moreover, we have shown that if the preference relation is Cartesian, then a new amalgamation in the form of a preference or an equivalence can always be consistently integrated without complex checks.

Our future work will aim at refining the concept of Cartesian preference relations with respect to amalgamations. To test the applicability and practical implications extensive experiments on real world data and suitable user profiles are necessary.

5. ACKNOWLEDGMENTS

This research has been partially supported by the German Research Foundation (DFG) within the Emmy Noether Program of Excellence.

6. REFERENCES

[1] W.-T. Balke, U. Güntzer, C. Lofi. Incremental Trade-Off Management for Preference Based Queries. In International Journal of Computer Science & Applications (IJCSA), Vol. 4(1), 2007.

[2] W.-T. Balke, U. Güntzer, W. Siberski. Restricting Skyline Sizes using Weak Pareto Dominance. In *Informatik - Forschung und Entwicklung (IFE)*, Vol. 21(3), Springer, 2007

[3] W.-T. Balke, U. Güntzer, J. Zheng. Efficient Distributed Skylining for Web Information Systems. In *Proc. of the Int. Conf. on Extending Database Technology (EDBT)*, LNCS 2992, Heraklion, Crete, Greece, 2004.

[4] W.-T. Balke, J. Zheng, U. Güntzer. Approaching the Efficient Frontier: Cooperative Database Retrieval Using High-Dimensional Skylines. In *Proc. of the Int. Conf. on Database Systems for Advanced Applications (DASFAA)*, Beijing, China, 2005.

[5] S. Börzsönyi, D. Kossmann, K. Stocker. The Skyline Operator. In *Proc. of the Int. Conf. on Data Engineering (ICDE)*, Heidelberg, Germany, 2001.

[6] C. Chan P. Eng, K. Tan. Stratified Computation of Skylines with Partially Ordered Domains. In *Proc. of the Int. Conf. on Management of Data (SIGMOD)*, Baltimore, MD, USA, 2005.

[7] J. Chomicki. Preference Formulas in Relational Queries. *In ACM Transactions on Database Systems (TODS)*, Vol. 28(4), 2003.´

[8] J. Chomicki. Iterative Modification and Incremental Evaluation of Preference Queries. *Int. Symp. on Found. of Inf. and Knowledge Systems (FoIKS)*, Budapest, Hungary, 2006.

[9] P. Godfrey. Skyline Cardinality for Relational Processing. In *Proc. of the Int Symp. on Foundations of Information and Knowledge Systems (FoIKS)*, Wilhelminenburg Castle, Austria, 2004.

[10] S. O. Hansson. Preference logic. Vol. 4 of Handbook of Philosophical Logic, 2nd Edition. Kluwer, 2002.

[11] W. Kießling. Foundations of Preferences in Database Systems. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, Hong Kong, China, 2002.

[12] W. Kießling, G. Köstler. Preference SQL - Design, Implementation, Experiences. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, Hong Kong, China, 2002.

[13] D. Kossmann, F. Ramsak, S. Rost. Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, Hong Kong, China, 2002.

[14] M. Lacroix, P. Lavency. Preferences: Putting more Knowledge into Queries. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, Brighton, UK, 1987.

[15] D. Papadias, Y. Tao, G. Fu, et.al. An Optimal and Progressive Algorithm for Skyline Queries. In *Proc. of the Int. ACM SIGMOD Conf. on Management of Data*, San Diego, USA, 2003.

[16] J. Pei, W. Jin, M. Ester, Y. Tao. Catching the Best Views of Skyline: A Semantic Approach Based on Decisive Subspaces. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, Trondheim, Norway, 2005.