Database Preferences – A Unified Model

Roxana Gheorghiu
University of Pittsburgh
roxana@cs.pitt.edu

Alexandros Labrinidis
University of Pittsburgh
labrinid@cs.pitt.edu

Panos K. Chrysanthis
University of Pittsburgh
panos@cs.pitt.edu

ABSTRACT

In this work we present a new model that combines two different types of preferences, qualitative and quantitative. We show how our model can support different types of preferences at different granularity levels and how can an application use these preferences to retrieve a list of tuples. The new model takes advantage of graph representation of preferences where nodes in the graph are SQL predicates, edges between two nodes describe a qualitative preference and edges from a node to the same node capture quantitative preferences. Each edge is labeled with a numeric value, between -1 and 1, to express the intensity of each preference. Using this graph representation we further show how two different preferences can be combined and applied to the existing query result set to filter the result and identify the most relevant tuples first.

1. INTRODUCTION

Preferences have been studied for many years and they were traditionally applied to philosophy, psychology and economics. In the Artificial Intelligence domain they were applied to decision making problems, capturing agents' goals. With the rapid increase of the Web and personal communication technologies on one hand and the explosion of available data on the other hand, preferences have become a requirement for scalable processing of large volumes of data. Query personalization techniques have been proposed both by academia and industry, showing the real interest for techniques that can cope with virtually an unlimited amount of data.

The literature classifies the preferences in two types: *quantitative* and *quantitative*. Pitoura et al. [6] describe the existing work in this area in terms of preference representation, preference composition, and preference query processing.

Quantitative preferences are described by scores attached to each tuple, for which a preference was expressed. As an example, consider the following preference: "I like comedies very much". This can be translated in the following quantitative preference: ("I like comedies", score =1) The score denotes users' interest in one or multiple data tuples. Using these scores we can define a total order over a part of the database tuples.

Qualitative preferences are expressed as pairs of tuples. As an example, consider the preference "I like comedies more than dramas". This can be translated into the following qualitative preference: ("comedies", *preferred over*, "dramas"). When put together, these pairs create a partial order between the tuples in the database.

When all database tuples have a score attached to them they can be ranked from the most preferred to the least preferred. The tuples that do not match any preference can be divided into two categories: equally preferred and incomparable. In the first case, tuples equally preferred can be seen as having attached the same score, whereas, in the second case, tuples that are incomparable cannot be included in the partial or total order defined by the preferences.

Each type of preferences – quantitative and qualitative – has its advantage over the other. However, there are examples when a user's preference can be conveniently expressed using one approach but not the other. For example, it is very easy to express a negative preference in the quantitative model by assigning a negative weight to that particular tuple or to a set of tuples that match a given condition. However, there is no easy way to express a negative preference in the qualitative model, since this will require to explicitly list all tuples that are preferred over the non-preferred ones.

In this work we propose a unified model that is based on graphs and integrates qualitative and quantitative preferences by means of preference intensity values and user profiles. This approach benefits from creating a global view of preferences that will be further used to correctly rank the query results.

Another approach to combine preferences is using PreferenceSQL [2]. This can be seen as a local view of preferences because each preference is given by the user at query time. Preference SQL does not allow different levels of intensity in the preference definition which can lead to an unexpected ordering of results (see example in the next section).

The formal underpinning of our proposed unified model is a preference graph. Each node in the graph represents a query condition; a directed link between two nodes is used to define a preferred order between tuples. We express quantitative preferences using edges that have the same starting and ending point. Qualitative preferences are represented by edges between two different nodes, and each edge is labeled

with a value that represents the intensity of the preference.

In our model, preferences are stored in a user profile and are expressed in the form of a preference graph stored as an adjacency matrix. Each user submits her own preferences, both qualitative and quantitative, along with an intensity value. In this way, users create their own profile by incrementally adding or removing preferences over the database tuples or attributes. Once the profile is created, preferences are used to filter the output result when user's query returns many tuples or arrange the result from the most preferred to the least preferred tuple. In out work the user does not have to decide which preferences are acceptable to her specific query since they are all stored in her profile and the system determines their applicability.

Contributions: The contributions of our work are as follows:

- We propose a new model that incorporates qualitative and quantitative preferences expressed as a preference graph. Our model can be used to define both tuple-based and predicate-based preferences and can easily support negative preferences.
- We handle preferences at different levels of granularity (e.g., tuple level vs. attribute level). Preferences are defined by the user and they will be used by our system anytime they apply.
- The preference graph is used to create an order over the tuples in the database based on intensity of preference.

Roadmap: Section 2 covers the related work and the differences between out proposed model and similar existing models. Section 3 defines our unified model for preferences, its implementation and preference specifications through examples. Section 4 introduces the composition techniques used to combine the two preference types. Section 5 gives a brief overview about how can the model be used and Section 6 summarizes our contributions and states the future work.

2. RELATED WORK

In the database domain preferences are seen as *soft* criteria, used to rank the results in terms of how well they match the query predicate. In contrast, the predicates in the SQL WHERE clause are seen as *hard* constraints and a non-empty result is returned only when all conditions are met.

Many solutions have been proposed for working with preferences[6]. In most of the cases the designed systems can handle only one type or preference (e.g., qualitative *or* quantitative). Our proposed model combines these two different approaches into a unified model.

Kiessling et al.[1] propose a framework that can support a hybrid version of both qualitative and quantitative preferences. PreferenceSQL system introduces a new clause, PREFERRING, in which the user can state their preferences relative to the current query. All preferences are connected with an AND operator except for the case when a qualitative preference is defined, in which case a PRIOR TO operator is used. In this framework users need to fully describe their preferences for each query.

Assume the following preferences: "I like white cars slightly better than yellow cars and I prefer cars around 5 years old". Also assume that we have the following tuples in the database: t1: (color=yellow, age=4), t2: (color=white, age=20) and t3: (color=red, age=5). Using PreferenceSQL we have two ways to write this preference. First way is to say we prefer (color=white PRIOR TO color=yellow) PRIOR TO (age around 5). This will result in the following order: t2, t1, t3. Another way is to say we prefer (age around 5) PRIOR TO ((color=white) PRIOR TO (color=yellow)). This will return tuples in the following order: t3, t1, t2. However when submitting the query, we expect the following ranking of the results: {**t1**, t2, t3} or {**t1**, t3, t2} because both preferences can be applied on tuple t1 whereas tuple t2 is preferred only in the first preference and tuple t3 is preferred only in the second preference. Our system can correctly rank tuple t1 as the first tuple. The reason for this is because our system takes a global view of all preferences which, in the end, will create an explicit ranking. As opposed to our work, Preference SQL takes a local restricted view of preferences, creating an implicit ranking induced by PRIOR TO function.

To summarize, our work differs from Kissling et al. because each preference is enhanced with intensity information to allow an ordering over the database tuples in the query result. Our work also handles both qualitative and quantitative preferences through user profiles. Both dimensions are important when we consider preferences because, together, they can generate a global ranking of tuples by (1) adding intensity values to every tuple for which a preference can be applied that can be further used to rank the query results and (2) dynamically modify the intensity values of tuples when a new preference is introduced in the profile and that is connected to the already existing ones.

The work done by Koutrika and Ioannidis [4] is the other most related to ours. In their work the preferences are kept as query predicates with an intensity value attached. In contrast to our work, they only record quantitative preferences and they are using them to create a preference network (i.e., a directed acyclic graph) that will allow an efficient identification of relevant preferences. This graph is used to depict the relation between preferences (i.e., each node in the network refers to a subclass of entities that its parent refers to) whereas in our case the graph depicts the flow of the preferences from the most preferred ones to the least preferred.

In contrast to [4], our work keeps track of preferences in any form (qualitative and quantitative) and our graph representation captures user specific order of tuples as they will show up in the final response after preferences are applied.

3. UNIFIED MODEL FOR PREFERENCES

A graph representation is the most natural way of exemplifying the connections between tuples in a database and visually depicting their relationships. The purpose of our preference graph is to connect two different preference approaches into a unified model.

DEFINITION 1. *We define the graph of preferences PG =(PV,PE) as a labeled directed graph where:*

- *PV is the set of vertices where each vertex represents a tuple in the database or a query predicate (e.g., a set of tuples).*
- *PE is the set of edges where each edge (v_i, v_j, s) defines a direction and is labeled with a score s. An edge from v_i to v_j captures a qualitative preference (e.g, the value in vertex v_i is preferred over the value in vertex v_j)*

whereas an edge from v_i to itself will describe a quantitative preference. The score s is a value between -1 and 1 that captures the preference intensity.

Using the preference graph description above, one preference is defined by the triplet (v_i, v_j, s), where v_i, $v_j \in$ PV and (v_i, v_j, s) $\in$ PE. When i=j, the triplet (v_i, v_j, s) represents a qualitative preference, and when i≠j, it represents a quantitative preference.

In our preference graph, intensity is a value between -1 and 1. All negative values are used to express negative preferences at different intensities, -1 being used to express complete dislike. In a similar way, all positive values are used to express positive preferences and 1 is used to capture the most preferred tuple. Zero is a special value used to express *equally* preferred tuples, in the case of qualitative preferences, and *indifference*, in the case of quantitative preferences.

For a quantitative preference, the intensity value expresses the likelihood of preferring one particular tuple (or a set of tuples for the predicate-based case) over all other tuples in the database. In this case, a large intensity value describes a strong preference towards that particular tuple or set of tuples.

For a qualitative preference, the intensity value expresses the likelihood of preferring one tuple over another. In this case, a small positive value will express a similarity on preferences (i.e., one tuple is almost as preferred as the other tuple).

Intensity can be a constant value or a function to allow dynamic ranking of preferences. As an example, consider the preference: "I like recent comedies", where recent can be expressed as a function on the year a movie was released and normalized in the proper range (i.e., [-1, 1]).

As mentioned in the definition, a vertex in the graph can represent a single tuple in the database (*tuple preference graph*), or a set of tuples if it is defined as a query predicate (*predicate preference graph*).

A tuple-based preference graph is usually not scalable because, for each tuple that matches a preference, a new vertex is created in the preference graph. However, this type of preference graph can be seen as a materialized database view and it is useful especially for cases when the preference has a low probability of changing (i.e., it is highly probable that the user will not change this particular preference often).

A predicate-based preference graph is a scalable version of the tuple-based one and it is used for preferences that apply to a large set of tuples. This type of preference graph is also useful for preferences that, by their nature, are temporary and will be removed and reinserted many times. There are two reasons for doing this: (1) it is much easier to remove only one row from the adjacency matrix as opposed to removing all rows linked to tuples that match a particular preference when that preference does not hold anymore; and (2) when one preference applies to many tuples, a system that creates only one row in the matrix is more scalable then one that introduces one row for every tuple that matches the preference.

From the representation point of view, both types of graphs are similar, and for this reason we will make a detailed presentation of tuple based preference graph in Sec. 3.1. The predicate-based preference graph model follows the same specifications and, given the space constraints, we will only point out the differences, in Sec. 3.3.

In our model, a graph will be created for each user profile in order to accurately return the most preferred tuples first, relative to each user's preferences. The preference graph is stored using an adjacency matrix. Each cell in this matrix contains the intensity value associated with one particular preference, when a preference is defined, and is *empty* otherwise. Following the definition, the intensity values of all quantitative preferences are values on the adjacency matrix's diagonal whereas the intensity of all qualitative preferences are values in all other cells.

3.1 Preference Graph Model Implementation

When creating a preference, the intensity of that preference allows one to decide, before hand, how important is that particular preference with respect to all preferences (for a quantitative preference) or with respect to another preference (for a qualitative preference). These values can be given by the user or they can be inferred from the preferences using Natural Language Processing algorithms for sentiment identification.

Assuming we have a free text interface, there are two ways to handle sentiment identification. One way is to parse the preference sentence and collect any words that may be referring to the intensity of a preference – such as *"very much", "like", "don't like", "acceptable"*. This is an orthogonal problem and is not the subject of our work. Another way to handle this situation is to create a simple mapping between terms and intensity values. For example, "very much" can be mapped by an intensity value 1 whereas "don't like" can be mapped by an intensity value -1. In this case we can provide a list of terms and let the user decide what is the most related value that match her preference.

3.2 Preference Specification Examples

The previous section explained how preferences can be created and inserted into a user profile, along with an intensity value. In the next six subsections we exemplify how different types of preferences can be expressed using our proposed preference graph model. We will use the classic toy example of *Movies* table (see Table 1). We start with an empty preference graph and we incrementally add new preferences in this graph. In Figures 1-5, along with the graph representation we also display the adjacency matrix that resulted when a new preference is added and will be used to internally store the preference graph. For simplicity, we will make a detailed presentation for the tuple-based preference case, but, as we mentioned in the previous section, the predicate-based preference case works similarly.

3.2.1 Negative Preference (Figure 1)

Assume a free text representation of preference, as follows: **Free text representation**: "I don't like horror movies". This type of preference is very useful in cases where the user knows what she does not want to see in the final query result. It can be easily expressed using a quantitative preference (i.e., by assigning an intensity value equal to -1) but is virtually impossible using a qualitative preference approach. If we would rewrite this in a calculus representation, we get: **Calculus representation**: ∀m∈Movie: m[genre]="horror" then m is not preferred.

movie_id	title	year	director	genre	language	duration
m1	Casablanca	1942	M. Curtiz	drama	english	102
m2	Psycho	1960	A. Hitchock	horror	english	109
m3	Schindler's List	1993	S. Spielberg	drama	english	195
m4	White Christmas	1954	M. Curtiz	comedy	english	120
m5	The Adventures of Tintin	2011	S. Spielberg	comedy	english	110

Table 1: The Movie Relation

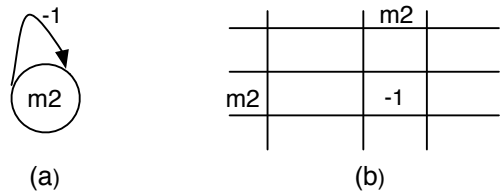


Figure 1: Specification of negative preferences (a) Graph; (b) Associated adjacency matrix

In our database example tuple m2 will match this preference therefore it will be incorporated in the preference graph. Following our proposed graph-based model we have a graph representation:
Graph representation: m2 ∈ PV, e1 =(m2 → m2) ∈ PE with score(e1) = −1

3.2.2 Relative Preference (Figure 2)

Free text representation: "If two movies have the same genre, I prefer the longer movie".
This is an example of a qualitative preference that cannot be expressed as a quantitative preference. This preference constructs a partial order between two tuples that match one common condition (i.e., they have the same genre) but are different in another (i.e., different duration). As an example from our database, tuple m4 will be preferred over tuple m5, and tuple m3 will be preferred over tuple m1.
Calculus representation: $\forall m_i, m_j \in$Movie:
m_i[genre]=m_j[genre] and m_i[duration]>m_j[duration] then m_i is preferred over m_j.
Graph representation: add m1, m3, m4, m5 ∈ PV and (m3, m1, 0.8), (m4, m5, 0.8) ∈ PE. Assuming that no intensity value was provided, a default intensity value equal to 0.8 will be added in cells [m3, m1] and [m4, m5] .

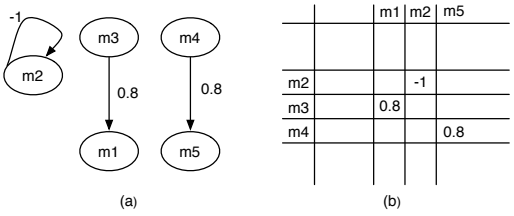


Figure 2: Specifications of relative preferences (a) Graph; (b) Associated adjacency matrix

3.2.3 Intensity (Figure 3)

A qualitative preference alone is defined only in terms of pairs of tuples (usually the first tuple in the pair is preferred over the second tuple), but it cannot capture how strong is the feeling related to that particular preference. Our model incorporates an intensity value to cope with this problem. The next example will illustrate this situation.
Free text representation: "I like drama movies a bit more than horror movies".
In our database, tuples m3 and m1 are preferred over m2.
Calculus representation: $\forall m_i, m_j \in$Movie:
if m_i[genre]="drama" and m_j[genre]="horror" then m_i is preferred over tuple m_j with intensity (m_i, m_j)
Graph representation: For each pair of tuples (m_i, m_j) that satisfies the condition m_i[genre]="drama" and m_j[genre]="horror", an edge from m_i to m_j is created and labeled with 0.2.

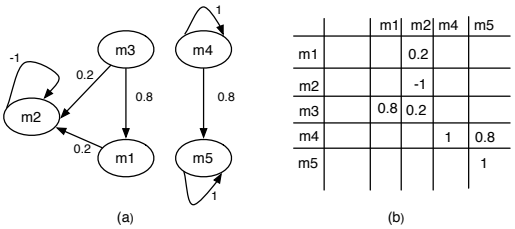
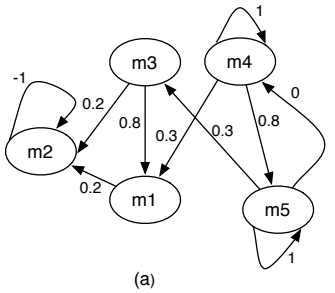


Figure 3: Specification of Intensity of Preferences (a) Graph; (b) Associated adjacency matrix

3.2.4 Sets (Figure 4)

Preference over a set of tuples is another example that can be expressed as a qualitative preference but not as a quantitative one. For example, assume the following preference:
Free text representation: "From a collection of movies D, I would prefer one comedy and as many movies as possible to have the same director".
Calculus representation:$\exists m_i \in$ (D∈Movie):
m_i[genre]="comedy" and $\forall m_j \in$D m_j[director]=m_i[director].
In our database, tuple m4 and m5 are preferred because they refer to comedies and the pairs of tuples: (m5, m3) and (m4, m1) are almost similar in preference (i.e., intensity value is 0.3) because they have the same director.
Graph representation: If two movies have the same director then an edge is created between the two tuples and a small value is used for intensity (e.g., 0.3) to suggest that the two tuples are similar in terms of preference. Tuple m4 is slightly more preferred than tuple m1 because it is also a comedy. For any tuple that represents a comedy, a self-addressed edge is created with a preference score (i.e., label) of 1. Moreover, there are two tuples in our database that represent a comedy: m5 and m4. To emphasize that comedies are equally preferred in this query, one edge (from m5 to m4) will be created in out graph and labeled with intensity 0. Notice that another edge, from m4 to m5 can be added



	m1	m2	m3	m4	m5
m1		0.2			
m2		-1			
m3	0.8	0.2			
m4	0.3			1	0.8
m5			0.3	0	1

(b)

Figure 4: Specification of set preferences (a) Graph; (b) Associated adjacency matrix

in the graph but since these edges are labeled with 0, and there is already one edge from m4 to m5 we keep only the one that does not overlap.

3.2.5 *Preferences at different granularity (Figure 5)*

Our proposed preference model can be used at different granularity levels. In previous sections we showed how it can be used to handle preferences that can be defined in one model but not in the other. In this section we describe how we can use the same model to support preferences over attributes, in addition to with preferences over values.

Free text representation: "I am interested in directors but not in genres".

Structured representation: In this case, a new preference graph over attributes with its corresponding adjacency matrix is created. Since the node's values are different than before (now Ai is an attribute whereas before, ti was a tuple) we cannot keep all the preferences together in the same graph.

Graph Representation: Two nodes are inserted into the graph: A1-Director attribute and A2-Producer attribute. A2 has a self-addressed edge labeled with -1. We also create an edge from A1 to A2 and label it with 0.8. This captures two similar meanings. The first one is that the user is more interested in directors than producers (0.8) whereas the second one captures the fact that the user is not interested in producers at all.

This preference example states that attribute *director* is more important than attribute *genre*. For the cases where attribute values are missing, the tuples that have a value for the director field but no value for the genre field will be preferred over tuples without a value for director but with a value for genre. Another useful case for this type of preference will be when it is combined with preferences over attribute values. For example, assume the following two extra preferences: "I prefer comedy movies" and "I prefer movies directed by Spielberg". In this case, tuple m5 will be preferred over tuple m4 because the director is more important than the genre of the movie.

3.3 Predicate preference

Section 3.1 described in detail the idea of our proposed preference graph model for the case where each node in the graph is associated with a tuple in the database. For the case of predicate-based preference graph everything mentioned in the previous section holds. The only difference now is that

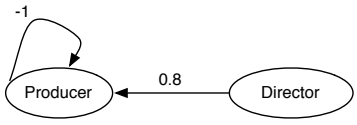


Figure 5: Specification of Relation preferences

each row in the adjacency matrix is a predicate which possibly matches multiple tuples in the database. This is why, each node in the graph is now a predicate and possibly multiple tuples in the database can be linked to that particular node.

As an example, consider the following two preferences:

- P1: "I prefer comedy over drama movies", score = 0.7 which can be translated as: P11 is preferred over P12 with intensity 0.7, where the new preferences are defined as:
  - P11: genre="comedy"
  - P12: genre="drama"
- P2: "I prefer drama movies 60% of the times", which can be translated into: genre="drama", score =0.6

	P11	P12
P11		0.7
P12		0.6

Figure 6: Adjacency Matrix for a predicate-based preference

4. PREFERENCE COMPOSITION

Each preference in a set of preferences can be applied to one or more tuples in the database. When two preferences affect the same tuple(s) or predicates it is necessary to have a mechanism of combining them into a single preference.

The literature describes two types of qualitative composition methods based on *attitude*: overriding attitude and combinatory attitude [6]. In the overriding attitude one preference has priority over the other, meaning that the lower priority preference is applicable only if the higher priority preference is not. In the combinatory attitude, as the name suggests, both preferences are combined and used. All techniques described in the literature handle the composition for the same type of preferences (i.e., combining two quantitative preferences or combining two qualitative preferences). Our model is designed to handle also compositions of one qualitative and one quantitative preference.

In our proposed unified model there is a score associated with each preference. In the case of a quantitative preference, the score is assigned to each tuple that matches the preference. In the case of qualitative preference, the score is attached to a pair of tuples and is recorded in the adjacency matrix in the cell corresponding to row t_i and column t_j. For example, assume we have the following preference: ("I like comedies more that drama", intensity=0.8). This does not mean that all tuples that match the predicate m_i[genre]="comedy" should get the intensity value 0.8 because this intensity value only states what should happen

when tuples that have genre="comedy" are compared with tuples that have genre"drama". When compared with tuples that have genre="horror" we should not assume any intensity.

Koutrika and Ioannidis [3] defined three types of behavior when combining two preference values. In their work, the resulting score can be: *inflationary* when the final preference value is larger then the initial values, *dominant* when one preference value dominates the final result, or *reserved*, when the final value lies between two preference values combined. In the final case the combined preference value decreases the value of the already assigned preference, whenever the value is larger than the new preference value, or will increase it if the already assigned value is smaller than the new value.

In order to take advantage of all information stored in our graph model we create a hybrid composition technique. In our system all tuples that match a preference (qualitative or quantitative) will have an implicit or explicit intensity value assigned. The explicit value is given by the user and is applied only to quantitative preferences. The implicit intensity value is computed internally and is used to readjust the intensity value already given or to assign an intensity value when there is no value given. The later case applies to cases where both a qualitative and a quantitative preference can be applied over the same tuple. In terms of graph notation, the later case applies when a tuple has a self addressed edge along with an incoming or outgoing edge.

Assume we have two preferences: P1 is a quantitative preference with intensity score 0.8 and P2 is a qualitative preference with intensity score 0.3. Also assume that there are two vertices, v1 and v2, part of this composition of preferences, with the following properties: v1 is preferred in P1 and v1 is preferred over v2 in P2. In this case the edge from v1 to v1 is labeled with 0.8, the edge from v1 to v2 is labeled with 0.3, and v2 does not have any label. Our algorithm will increase the value on the self addressed edge of node v1 and will create a self addressed edge to v2 with a value smaller than the one on v1. This algorithm comes from the following reasoning. First, for the vertex v1 the user expressed two preferences: one qualitative and one quantitative. That means that these particular tuples are preferred in general (quantitative preference) and they are also preferred when compared with other tuples. Because of that, we state that the intensity value for this vertex should increase, since it is part of two preferences and is preferred in both cases. Second, for the vertex v2 we add a new value, since v2 is does not have any intensity value defined. All we know at this point is that tuples in vertex v2 are less preferred than tuples in vertex v1. Because of that, vertex v2 is labeled with a lower intensity value than vertex v1 and inversely proportional with the value on the edge between v1 and v2.

5. HOW ARE PREFERENCES USED

In our model we construct a preference graph for each user profile. When a new preference is added, or learned from the user's behavior, the graph is modified to include the new knowledge. The insertion of the new preference needs to take into account the existing preferences. If the new preference shares the same predicate with another preference that is already defined in the user's profile, then the new preference added to the graph has to be connected to the existing preference. For example, assume that we have the following entries in the preference graph: the set of vertices: PV={v1,v2} and the set of edges: PE={(v1,v2,0.8)} where v1 is genre[m_i]=drama and p2 is genre[m_j]=horror meaning that in 80% of the time the user prefers drama over horror movies. Additionally assume we have a new preference added, p3:"Comedies are preferred over drama movies". Since p3 shares the same predicate as p1, we need to add a new node, p3=(genre="comedy") and a new edge from p3 to p1.

In all other cases, the new preference is independent of all existing preferences, so there is no connection between the existing graph and the new vertex is simply added.

We are currently implementing this model in Astroshelf [5]. Preferences are added using a simple web page where users can define one or more preferences, using a single predicate or a disjunction of conjunctions when multiple predicates are part of the same preference.

6. CONCLUSIONS AND FUTURE WORK

In this work we presented a new model that combines two different types of preferences often studied before, qualitative and quantitative. We showed how our model could support different types of preferences at different granularity levels and how an application can use this preference model to retrieve a list sorted from the most preferred to the least preferred tuples. This is an important desideratum that accommodates two powerful approaches of preferences using a unified model.

As part of our future work, we are currently implementing the full version of the theoretical model described here to find its strengths and weaknesses, in particular in terms of its performance/scalability. Another interesting problem we plan to investigate is the recommendation of *interesting* data tuples, by employing collaborative filtering techniques over the set of user-specified preferences.

7. ACKNOWLEDGMENTS

This research was supported in part by NSF career awards IIS-0746696 and IIS-0952720 and NSF grant OIA-1028162.

8. REFERENCES

[1] W. Kiessling. Foundations of preferences in database systems. In *VLDB 2002*, pages 311–322, 2002.

[2] W. Kiessling and G. Köstler. Preference sql: design, implementation, experiences. In *VLDB 2002*, pages 990–1001, 2002.

[3] G. Koutrika and Y. Ioannidis. Personalized queries under a generalized preference model. In *ICDE 2005*, pages 841–852, 2005.

[4] G. Koutrika and Y. Ioannidis. Personalizing queries based on networks of composite preferences. *ACM Trans. Database Syst.*, 35(2):13:1–13:50, May 2010.

[5] P. Neophytou, R. Gheorghiu, R. Hachey, T. Luciani, D. Bao, A. Labrinidis, E. G. Marai, and P. K. Chrysanthis. Astroshelf: Understanding the universe through scalable navigation of a galaxy of annotations. *SIGMOD 2012*, pages pp. 1–4, 2012.

[6] K. Stefanidis, G. Koutrika, and E. Pitoura. A survey on representation, composition and application of preferences in database systems. *ACM Trans. Database Syst.*, 36(3):19:1–19:45, 2011.