PREFERENCE RULES IN DATABASE QUERYING

Sergio Greco, Cristian Molinaro and Francesco Parisi
DEIS, University of Calabria, 87036 Rende, Italy

Keywords: Deductive databases, prioritized queries, preferences.

Abstract: The paper proposes the use of preferences for querying databases. In expressing queries it is natural to express preferences among tuples belonging to the answer. This can be done in commercial DBMS, for instance, by ordering the tuples in the result. The paper presents a different proposal, based on similar approaches deeply investigated in the artificial intelligence field, where preferences are used to restrict the result of queries posed over databases. In our proposal a query over a database $\mathcal{DB}$ is a triple $\langle q, \mathcal{P}, \Phi \rangle$, where q denotes the output relation, $\mathcal{P}$ a Datalog program (or an SQL query) used to compute the result and Φ is a set of preference rules used to introduce preferences on the computed tuples. In our proposal tuples which are "dominated" by other tuples do not belong to the result and cannot be used to infer other tuples. A new stratified semantics is presented where the program $\mathcal{P}$ is partitioned into strata and the preference rules associated to each stratum of $\mathcal{P}$ are divided into layers; the result of a query is carried out by computing one stratum at time and by applying the preference rules, one layer at time. We show that our technique is sound and that the complexity of computing queries with preference rules is still polynomial.

1 INTRODUCTION

The growing volume of available information poses new challenges to the database and artificial intelligence communities. Recent researches have investigated new techniques in accessing large volumes of data such as user-centered access to information, information filtering and extraction and policies to reduce data presented to users. An interesting direction deeply studied in the artificial intelligence and non-monotonic reasoning fields consists in the use of preferences to express priorities on the alternative scenarios.

The paper presents a logical framework wherein preferences are used to restrict the result of queries posed over a database. This is an important aspect in querying large databases such as those used by search engines. In this context, the result of a query contains only tuples which are not *dominated* by other tuples and dominated tuples cannot be used to infer new information. The novelty of the presented approach is that preferences are stratified and applied one stratum at time. A second innovative aspect of this proposal is that preferences on both base and derived atoms are considered as well as general (recursive) queries which can be expressed by means of stratified Datalog.

Example 1 Consider a database $\mathcal{DB} = \{\mathtt{fish}, \mathtt{beef}\}$ and a program $\mathcal{P}$ consisting of the two rules:

$\mathtt{red\text{-}wine} \leftarrow \mathtt{beef}$
$\mathtt{white\text{-}wine} \leftarrow \mathtt{fish}$

Assume now to have a query defined by the rules in $\mathcal{P}$ and the preference

$\rho_1 = \mathtt{red\text{-}wine} \succ \mathtt{white\text{-}wine} \leftarrow \mathtt{beef}$

stating that if there is `beef`, we prefer `red-wine` to `white-wine`. The set of preferred atoms contains the base atoms `fish` and `beef` and the derived atom `red-wine` (the atom `white-wine` is not preferred). Assume now to also have the preference $\rho_2 = \mathtt{fish} \succ \mathtt{beef}$ stating that we prefer `fish` to `beef`. In this case, first the preference rule ρ_2, and next the preference rule ρ_1, are considered. However, ρ_1 cannot be applied as `beef` is not in the preferred set

Greco S., Molinaro C. and Parisi F. (2007).
PREFERENCE RULES IN DATABASE QUERYING.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - DISI*, pages 119-124
DOI: 10.5220/0002389901190124

of atoms. Consequently, the set of preferred atoms, with respect to the preference rules ρ_2 and ρ_1, is $\{\texttt{fish}, \texttt{white-wine}\}$. □

Contributions. In this paper we study the use of preferences in querying databases. We consider general (stratified) Datalog queries and general preferences: the head of preference rules may contain atoms belonging to different relations and the body consists of a conjunction of literals. A semantics where both query and preferences are partitioned into strata is defined. Under such a semantics, the query is computed one stratum at time and for each stratum (of the query), the preferences are applied one stratum at time.

Related Work. The increased interest in preferences in logic programs is reflected by an extensive number of proposals and systems for preference handling. Most of the approaches propose an extension of logic programming by adding preference information. The most common form of preference consists in specifying a strict partial order on rules (Delgrande et al., 2003; Gelfond and Son, 1997; Sakama and Inoue, 2000; Zhang and Foo, 1997), whereas more sophisticated forms of preferences also allow priorities to be specified between conjunctive (disjunctive) knowledge with preconditions (Brewka et al., 2003; Sakama and Inoue, 2000) and numerical penalties for suboptimal options (Brewka, 2004).
Considering the use of preferences in querying databases, an extension of relational calculus expressing preferences for tuples in terms of logical conditions has been proposed in (Lacroix and Lavency, 1987). Preferences requiring non-deterministic choice among atoms which minimize or maximize the value of some attribute has been proposed in (Greco and Zaniolo, 2002). An extension of Datalog with preference relations, subsuming the approach proposed in (Lacroix and Lavency, 1987), has been proposed in (Kostler et al., 1995), whereas an extension of SQL including preferences has been proposed in (Kieβling, 2002; Kieβling and Kostler, 2002). In the last proposal several built-in operators and a formal definition of their combinations (i.e. intersection, union, Pareto composition, etc.) has been considered. Borzsonyi et al. proposed the *skyline* operator (Borzsonyi et al., 2001), to filter out a set of "interesting" point (i.e. not dominated by any other point) from a potential large set of points. An extension of SQL with a skyline operator has been also proposed. A framework for specifying preferences using logical formulas and its embedding into relational algebra has been introduced in (Chomicki, 2003). The paper also introduces the *winnow* operator which generalizes the skyline operator. The implementation of winnow and ranking is also studied in (Torlone and Ciaccia, 2002). Algorithms for computing skyline operators are also studied in (Kossmann et al., 2002; Papadias et al., 2003; Chomicki et al., 2003). In (Agrawal and Wimmers, 2002) the use of quantitative preferences (scoring functions) in queries is proposed.
In this work, in contrast with previous proposals, general preferences and a different (stratified) semantics, which we believe to be more intuitive, are considered.

2 BACKGROUND

Familiarity with disjunctive logic programs and disjunctive deductive databases is assumed (Ullman, 1988).

Datalog Programs. A *term* is either a constant or a variable. An *atom* is of the form $p(t_1,\ldots,t_h)$, where p is a *predicate symbol* of arity h and $t_1,\ldots,t_h$ are terms. A *literal* is either an atom A or its negation *not* A. A *(Datalog) rule* r is a clause of the form

$$A \leftarrow B_1,\ldots,B_m, not\ B_{m+1},\ldots, not\ B_n, \varphi \qquad n \geq 0$$

where $A, B_1,\ldots,B_n$ are atoms, whereas φ is a conjunction of built-in atoms of the form $u\theta v$ where u and v are terms and θ is a comparison predicate. A is the *head* of r (denoted by $Head(r)$), whereas the conjunction $B_1,\ldots,B_m, not\ B_{m+1},\ldots, not\ B_n, \varphi$ is the *body* of r (denoted by $Body(r)$). It is assumed that each rule is safe, i.e. a variable appearing in the head or in a negative literal also appears in a positive body literal.
A *(Datalog) program* is a finite set of rules. A *not*-free program is called *positive*. The *Herbrand Universe* $\mathcal{U}_{\mathcal{P}}$ of a program $\mathcal{P}$ is the set of all constants appearing in $\mathcal{P}$, and its *Herbrand Base* $\mathcal{B}_{\mathcal{P}}$ is the set of all ground atoms constructed from the predicates appearing in $\mathcal{P}$ and the constants from $\mathcal{U}_{\mathcal{P}}$. A term (resp. an atom, a literal, a rule or a program) is *ground* if no variable occurs in it. A rule r' is a *ground instance* of a rule r if r' is obtained from r by replacing every variable in r with some constant in $\mathcal{U}_{\mathcal{P}}$; $ground(\mathcal{P})$ denotes the set of all ground instances of the rules in $\mathcal{P}$.
An *interpretation* M for a Datalog program $\mathcal{P}$ is any subset of $\mathcal{B}_{\mathcal{P}}$; M is a *model* of $\mathcal{P}$ if it satisfies all rules in $ground(\mathcal{P})$. The (model-theoretic) semantics for positive $\mathcal{P}$ assigns to $\mathcal{P}$ the set of its *minimal models* $\mathcal{MM}(\mathcal{P})$, where a model M for $\mathcal{P}$ is minimal if no proper subset of M is a model for $\mathcal{P}$. For any interpretation M, $\mathcal{P}^M$ is the ground positive program derived from $ground(\mathcal{P})$ by 1) removing all rules that contain

a negative literal $not A$ in the body and $A \in M$, and 2) removing all negative literals from the remaining rules. An interpretation M is a *stable model* of $\mathcal{P}$ if and only if $M \in \mathcal{MM}(\mathcal{P}^M)$ (Gelfond and Lifschitz, 1988). For general $\mathcal{P}$, the stable model semantics assigns to $\mathcal{P}$ the set $\mathcal{SM}(\mathcal{P})$ of its stable models. It is well-known that stable models are minimal models (i.e. $\mathcal{SM}(\mathcal{P}) \subseteq \mathcal{MM}(\mathcal{P})$) and that for negation free programs minimal and stable model semantics coincide (i.e. $\mathcal{SM}(\mathcal{P}) = \mathcal{MM}(\mathcal{P})$).

Given a Datalog program $\mathcal{P}$, $\mathcal{G}_g(\mathcal{P}) = (V_g, E_g)$ denotes the dependency graph associated with $ground(\mathcal{P})$ where V_g consists of all ground atoms appearing in $ground(\mathcal{P})$, whereas there is an arc from B to A in E_g if there is a rule r in $ground(\mathcal{P})$ such that $Head(r) = A$ and $B \in Body(r)$; the arc is said to be marked negatively if B appears negated in the body of r. The dependency graph $\mathcal{G}(\mathcal{P}) = (V, E)$ associated with $\mathcal{P}$ is built by considering the ground program derived from $\mathcal{P}$ by eliminating all terms (i.e. every atom $p(t)$ is replaced by p). A ground atom $p(t)$ *depends* on a ground atom $q(u)$ if there is a path in $\mathcal{G}_g(\mathcal{P})$ from $q(u)$ to $p(t)$. Analogously, a predicate symbol p *depends* on a predicate symbol q if there is a path in $\mathcal{G}(\mathcal{P})$ from q to p. The dependency is negated if there is an arc marked negatively in the path.

A partition $\pi_0, \ldots, \pi_k$ of the set of all predicate symbols of a Datalog program $\mathcal{P}$, where each π_i is called *stratum*, is a *stratification* of $\mathcal{P}$ if for each rule r in $\mathcal{P}$ the predicates that appear only positively in the body of r are in strata lower than or equal to the stratum of the predicate in the head of r, and the predicates that appear negatively are in strata lower than the stratum of the predicate in the head of r. The stratification of the predicates defines a stratification of the rules of $\mathcal{P}$ into strata $\langle \mathcal{P}_1, \ldots, \mathcal{P}_k \rangle$ where a stratum $\mathcal{P}_i$ contains rules which define predicates in π_i. A Datalog program is called *stratified* if it has a stratification. Stratified (normal) programs have a unique stable model which coincides with the *stratified model*, obtained by computing the fixpoints of every stratum in their order.

Queries. Predicate symbols are partitioned into two distinct sets: *base predicates* and *derived predicates*. Base predicates correspond to database relations defined over a given domain and they do not appear in the head of any rule, whereas derived predicates are defined by means of rules. Given a set of ground atoms $\mathcal{DB}$, a predicate symbol p and a stratified program $\mathcal{P}$, $\mathcal{DB}[p]$ denotes the set of p-tuples in $\mathcal{DB}$, while $\mathcal{P}_{\mathcal{DB}}$ denotes the program derived from the union of $\mathcal{P}$ with the facts in $\mathcal{DB}$, i.e. $\mathcal{P}_{\mathcal{DB}} = \mathcal{P} \cup \mathcal{DB}$. The semantics of $\mathcal{P}_{\mathcal{DB}}$ is given by the stratified model (which coincide with the unique stable model) of $\mathcal{P}_{\mathcal{DB}}$. The answer to a query $Q = (g, \mathcal{P})$ over a database $\mathcal{DB}$, denoted by $Q(\mathcal{DB})$, is given by $\mathcal{M}[g]$ where $\mathcal{M} = \mathcal{SM}(\mathcal{P}_{\mathcal{DB}})$. In the following we also denote with $\mathcal{P}(\mathcal{DB}) = \mathcal{SM}(\mathcal{P}_{\mathcal{DB}})$ the application of $\mathcal{P}$ to $\mathcal{DB}$; therefore $Q(\mathcal{DB}) = \mathcal{P}(\mathcal{DB})[g]$.

3 PREFERENCE RULES AND QUERIES

This section presents a framework for expressing preferences in the evaluation of queries posed on a given database. The framework is based on the introduction of *preference rules*, whose syntax is inspired to the management of priorities in the artificial intelligence field, logic programming and database querying (Brewka et al., 2003; Delgrande et al., 2003; Gelfond and Son, 1997; Sakama and Inoue, 2000; Zhang and Foo, 1997).

3.1 Syntax

A *prioritized program* consists of a set of standard rules (Datalog program) and a set of preference rules. As rules expressing preferences eliminate tuples which are derived by means of standard rules (Datalog program) we first introduce a standard stratification of the Datalog program to fix the order in which standard rules are applied. Preference rules are associated to each subprogram (stratum) and applied after the subprogram has been evaluated. Let start by introducing the concept of standard stratification.

Definition 1 The *standard stratification* of a stratified program $\mathcal{P}$ consists of k strata $\langle \mathcal{P}_1, ..., \mathcal{P}_k \rangle$ where k is the minimal value such that for each $\mathcal{P}_i$ and for each pair of predicates p and q defined in $\mathcal{P}_i$ either they are mutually recursive or they are independent (i.e. p does not depend on q and q does not depend on p). □

In the following, given an atom $p(t)$, $str(p(t))$ denotes the stratum of the predicate symbol p (or equivalently of the subprogram in which p is defined) in the standard stratification.

Definition 2 A *preference rule* ρ is of the form:

$$A \succ C \leftarrow B_1, ..., B_m, not\ B_{m+1}, ..., not\ B_n, \varphi \qquad (1)$$

where where $A, C, B_1, \ldots, B_n$ are atoms, and φ is a conjunction of built-in atoms. □

Also in this case we assume that rules are safe. In the above definition $A \succ C$ is called head of the preference rule (denoted as $Head(\rho)$), whereas the conjunction $B_1, ..., B_m, not\ B_{m+1}, ..., not\ B_n, \varphi$ is called body (denoted as $Body(\rho)$). Moreover, we denote with $Head_1(\rho)$ and $Head_2(\rho)$ the first and the second atom in the head of ρ, respectively (i.e. $Head_1(\rho) = A$ and $Head_2(\rho) = C$).

The intuitive meaning of a ground preference rule ρ is that if the body of ρ is true, then the atom A is *preferable* to C (we also say that the atom C is *dominated* by the atom A). This means that in the evaluation of a prioritized program $\langle \mathcal{P}, \Phi \rangle$ the model defining its semantics cannot contain the atom C if it contains the atom A and the body of the preference rule is true.

Let Φ be a *preference program*, i.e. a set of preference rules. The transitive closure of $ground(\Phi)$ is $\Phi_g^* = ground(\Phi) \cup \{(A \succ C \leftarrow body_1, body_2 \mid \exists A \succ B \leftarrow body_1 \in \Phi_g^* \wedge \exists B \succ C \leftarrow body_2 \in \Phi_g^* \}$. Analogously, we define Φ^* as the closure of the set of ground preference rules derived from Φ by replacing every atom $p(t)$ with p and deleting built-in atoms.

Definition 3 A (ground) preference program Φ_g^* is *layered* if it is possible to partition it into n layers $\langle \Phi_g^*[1], \ldots, \Phi_g^*[n] \rangle$ as follows:

- For each ground atom A such that there is no ground rule $\rho \in \Phi_g^*$ such that $Head_2(\rho) = A$, $layer(A) = 0$;
- For every ground atom C such that there is a rule ρ of the form (1) (i.e. such that $Head_2(\rho) = C$), $layer(C) > \max\{layer(B_1), \ldots, layer(B_n), 0\}$ and $layer(C) \geq layer(A)$;
- The layer of a preference rule $\rho \in \Phi_g^*$, denoted as $layer(\rho)$, is equal to $layer(Head_2(\rho))$;
- $\Phi_g^*[i]$ consists of all preference rules associated with the layer i. □

Example 2 Consider the set of preference rules Φ:

ρ_1 : `fish` $\succ$ `beef` $\leftarrow$
ρ_2 : `red-wine` $\succ$ `white-wine` $\leftarrow$ `beef`
ρ_3 : `white-wine` $\succ$ `red-wine` $\leftarrow$ `fish`

The transitive closure Φ^* consists of the rules ρ_1, ρ_2, ρ_3 plus the following rules

ρ_4 : `red-wine` $\succ$ `red-wine` $\leftarrow$ `beef`, `fish`
ρ_5 : `white-wine` $\succ$ `white-wine` $\leftarrow$ `fish`, `beef`

Φ^* is partitioned into the two layers $\Phi^*[1] = \{\rho_1\}$ and $\Phi^*[2] = \{\rho_2, \rho_3, \rho_4, \rho_5\}$. □

As it will be clear in the next subsection, preference rules of the form $A \succ A \leftarrow body$ are useless and can be deleted. Therefore, in the above example $\Phi^*[2] = \{\rho_2, \rho_3\}$.

Example 3 Consider the set of preference rules Φ:

ρ_1 : `fish` $\succ$ `beef` $\leftarrow$ `white-wine`
ρ_2 : `red-wine` $\succ$ `white-wine` $\leftarrow$ `beef`

According to ρ_1 the layer of `beef` must be greater than the layer of `white-wine`, whereas according to ρ_2 the layer of `white-wine` must be greater than the layer of `beef`. Thus, the set of preference rules is not layered. □

Observe that in the above definition, in order to compute the closure of the ground instantiation of Φ, we need to know the database $\mathcal{DB}$ containing all constants in the database domain. Therefore, checking whether Φ_g^* can be partitioned into layers cannot be done at compile-time. It is possible to define sufficient conditions which guarantee that the set of preference rules can be partitioned into layers by considering the (ground) program Φ^* instead of the program Φ_g^*. This means that if Φ^* can be partitioned into layers, the set Φ_g^* can be partitioned into layers as well, although the layers of Φ_g^* may be different from the layers of Φ^* (the layers of Φ_g^* define a "refinement" of the layers of Φ^*).

Definition 4 A *prioritized query* is of the form $\langle q, \mathcal{P}, \Phi \rangle$ where q is a predicate symbol denoting the output relation, $\mathcal{P}$ is a (stratified) Datalog program and Φ is a set of preference rules. □

As said before, the intuitive meaning of a prioritized query $\langle q, \mathcal{P}, \Phi \rangle$ over a database $\mathcal{DB}$ is that the atoms derived from $\mathcal{P}$ and $\mathcal{DB}$ must satisfy the preference conditions defined in Φ.

Definition 5 A prioritized query $Q = \langle q, \mathcal{P}, \Phi \rangle$ is said to be *well formed* if Φ_g^* is layered and for every ground atom C such that there is a rule ρ of the form (1) (i.e. such that $Head_2(\rho) = C$) it holds that

1. $str(C) \geq \max\{str(A), str(B_1), \ldots, str(B_n)\}$, and
2. $A, B_1, ..., B_m$ do not depend on C in $\mathcal{P}$. □

In the following we assume that our queries are well formed. Sufficient conditions can be defined on the base of the dependency graph $\mathcal{G}(\mathcal{P})$.

3.2 Semantics

First we analyze the case where Φ defines preferences on databases atoms and next we consider the case where Φ expresses preferences on base and derived atoms, i.e. also on atoms defined in $\mathcal{P}$.

3.2.1 Preferences On Base Atoms

It is assumed here to have a query $Q = \langle q, \mathcal{P}, \Phi \rangle$ and that the preference rules in Φ express preferences only among base atoms. As said before, Φ_g^* can be partitioned into n layers $\widehat{\Phi}_g^* = \langle \Phi_g^*[1], ..., \Phi_g^*[n] \rangle$.

Definition 6 Let $\mathcal{DB}$ be a set of ground atoms, Φ a set of preference rules such that $\widehat{\Phi}_g^* = \langle \Phi_g^*[1], ..., \Phi_g^*[n] \rangle$, and t, u two atoms in $\mathcal{DB}$. We say that t is *preferable* to u with respect to $\Phi_g^*[i]$ (denotes as $t \sqsupset_{\Phi[i]} u$) if

- $\exists (t \succ u \leftarrow body_1) \in \Phi_g^*[i]$ s.t. $\mathcal{DB} \models body_1$, and
- $\nexists (u \succ t \leftarrow body_2) \in \Phi_g^*[i]$ s.t. $\mathcal{DB} \models body_2$.

The set of tuples in $\mathcal{DB}$ which are *preferred* with respect to $\Phi_g^*[i]$ is $\Phi_g^*[i](\mathcal{DB}) = \{t \mid t \in \mathcal{DB} \wedge \nexists u \in \mathcal{DB}$ s.t. $u \sqsupset_{\Phi[i]} t\}$. $\Box$

Observe that Φ_g^* could contain preference rules of the form $A \succ A \leftarrow body$. Such preferences are useless as they are not used to infer preferences among ground atoms and can be deleted from Φ_g^*.

Example 4 Consider the database $\mathcal{DB} = \{$`fish`, `beef`, `red-wine`, `white-wine`, `pie`, `ice-cream`$\}$ and the following preference rules Φ:

ρ_1 : `pie` $\succ$ `ice-cream` $\leftarrow$
ρ_2 : `red-wine` $\succ$ `white-wine` $\leftarrow$ `fish`
ρ_3 : `white-wine` $\succ$ `red-wine` $\leftarrow$ `beef`

The set Φ_g^* consists, without considering useless rules, of a unique layer $\Phi_g^*[1] = \{\rho_1, \rho_2, \rho_3\}$. The application of $\Phi_g^*[1]$ to $\mathcal{DB}$ gives the set $\Phi_g^*[1](\mathcal{DB}) = \{$`fish`, `beef`, `red-wine`, `white-wine`, `pie`$\}$ $\Box$

Definition 7 Let $\mathcal{DB}$ be a database and $Q = \langle q, \mathcal{P}, \Phi \rangle$ be a query such that Φ expresses preferences only on base atoms and the set of ground preference rules Φ_g^* is layered into $\widehat{\Phi}_g^* = \langle \Phi_g^*[1], ..., \Phi_g^*[n] \rangle$. Then the set of preferred tuples with respect to $\widehat{\Phi}_g^*$ is $\mathcal{M} = \mathcal{P}(\widehat{\Phi}_g^*(\mathcal{DB}))$

$$= \mathcal{P}(\Phi_g^*[n](\Phi_g^*[n-1] \cdots (\Phi_g^*[1](\mathcal{DB})) \cdots))).$$

The answer to the query Q is given by $\mathcal{M}[q]$. $\Box$

Example 5 Consider the database $\mathcal{DB} = \{$`fish`, `beef`, `red-wine`, `white-wine`, `pie`$\}$ and the preference rules Φ of Example 2. Φ_g^* is equal to Φ and it is layered into $\widehat{\Phi}_g^* = \langle \Phi_g^*[1], \Phi_g^*[2] \rangle = \langle \{\rho_1\}, \{\rho_2, \rho_3\} \rangle$
The application of $\Phi_g^*[1]$ to $\mathcal{DB}$ gives the set $\mathcal{M}_1 = \Phi_g^*[1](\mathcal{DB}) = \{$`fish`, `red-wine`, `white-wine`, `pie`$\}$
The application of $\Phi_g^*[2]$ to $\mathcal{M}_1$ gives the set $\mathcal{M}_2 = \Phi_g^*[2](\mathcal{M}_1) = \{$`fish`, `white-wine`, `pie`$\}$ $\Box$

3.2.2 General Preferences

We consider now general prioritized queries $Q = \langle q, \mathcal{P}, \Phi \rangle$ where $\mathcal{P}$ is a stratified Datalog program and Φ expresses preferences also on derived atoms.
Let $\langle q, \mathcal{P}, \Phi \rangle$ be a prioritized query and $\mathcal{DB}$ a database. Let $\langle \mathcal{P}_1, \ldots, \mathcal{P}_k \rangle$ be the standard stratification of $ground(\mathcal{P})$ and let $\mathcal{P}_0 = \{A \leftarrow \mid A \in \mathcal{DB}\}$. Then, $\Phi_g^*[\mathcal{P}_i]$, for $i \in [0..k]$, denotes the following set of preference rules in Φ_g^*:

$$\Phi_g^*[\mathcal{P}_i] = \{A \succ C \leftarrow body \mid \exists (C \leftarrow body') \in \mathcal{P}_i\}$$

Definition 8 Let $\mathcal{DB}$ be a database and let $Q = \langle q, \mathcal{P}, \Phi \rangle$ be a prioritized query and $\langle \mathcal{P}_1, ..., \mathcal{P}_k \rangle$ the standard stratification of $\mathcal{P}$. The application of $\mathcal{P}$ and Φ to $\mathcal{DB}$ is defined as follows: $\mathcal{M}_0 = \widehat{\Phi}_g^*[\mathcal{P}_0](\mathcal{DB})$ and for each i in $[1..k]$, $\mathcal{M}_i = \widehat{\Phi}_g^*[\mathcal{P}_i](\mathcal{P}_i(M_{i-1}))$.
The answer to the query Q over the database $\mathcal{DB}$, denoted as $Q(\mathcal{DB})$, is given by $\mathcal{M}_k[q]$. $\Box$

Our proposal is sound, i.e. for each ground preference rule $A \succ C \leftarrow body$ in Φ_g^*, if $M_k \models (body \wedge A)$ then $M_k \not\models C$. Moreover, it can be shown that the computational complexity of $Q(\mathcal{DB})$ is polynomial time.

4 CONCLUSIONS

This paper has introduced *prioritized queries*, a form of queries well-suited for expressing preferences among tuples either belonging to the source database or derived by means of the program specified in the query. It has been shown that prioritized queries are well-suited to express queries wherein we are interested only in *preferred tuples*. A stratified semantics for computing prioritized queries has been presented where the program $\mathcal{P}$ is partitioned into strata and the preference rules associated to each stratum of $\mathcal{P}$ are divided into layers; a query is evaluated by computing one stratum at time and by applying the preference rules, one layer at time. The computational complexity of computing prioritized queries remains polynomial.

REFERENCES

Agrawal, R., and Wimmers, E. L. (2002). A framework for expressing and combining preferences. *Proc. SIGMOD*, pp. 297-306.

Borzsonyi S., Kossmann D., Stocker K. (2001). The skyline operator, *Proc. ICDE*, 421-430.

Brewka, G. (2004). Complex Preferences for Answer Set Optimization, *KR*, 213-223.

Brewka G., Niemela I., Truszczynski M. (2003). Answer Set Optimization. *IJCAI*, 867-872.

Chomicki, J. (2003). Preference Formulas in Relational Queries. *ACM TODS*, 28(4), 1-40.

Chomicki, J., Godfrey, P., Gryz, J., and Liang, D. (2003). Skyline with presorting. *Proc. ICDE*.

Delgrande, J., P., Schaub, T., Tompits, H. (2003). A Framework for Compiling Preferences in Logic Programs. *TPLP*, 3(2), 129-187.

Gelfond, M., Son, T.C. (1997). Reasoning with prioritized defaults. *LPKR*, 164-223.

Gelfond, M., Lifschitz, V. (1988). The Stable Model Semantics for Logic Programming, *ICLP*.

Greco S., Zaniolo C. (2002). Greedy by Choice, *Proc. PODS*.

Kieβling, W. (2002). Foundations of preferences in database systems, *Proc. VLDB*.

Kieβling, W., Kostler, G. (2002). Preference SQL - Design, Implementation, experience, *VLDB*.

Kossmann, D., Ramsak, F., and Rost, S. (2002). Shooting stars in the sky: An online algorithm for skyline queries. *Proc. VLDB*.

Kostler, G., Kieβling, W., Thone, H., Guntzer, U. (1995). Fixpoint iteration with subsumption in deductive databases. *JIIS*, 4, 123-148.

Lacroix M., Lavency P.(1987). Prefences: Putting More Knowledge Into Queries. *VLDB*, 217-225.

Papadias, D., Tao, Y., Fu, G., and Seeger, B. (2003). An optimal and progressive algorithm for skyline queries, *Proc. SIGMOD*, pp. 467-478.

Sakama, C., Inoue, K. (2000). Priorized logic programming and its application to commonsense reasoning. *Artificial Intelligence*, 123, 185-222.

Torlone, R., Paolo Ciaccia. (2002). Finding the Best when it's a Matter of Preference, *Proc. SEBD*, pp. 347-360.

Ullman, J. K. (1988). *Principles of Database and Knowledge-Base Systems*, Vol. 1, Computer Science Press.

Zhang, Y., Foo, N. (1997). Answer sets for prioritized logic programs. *ILPS*, 69-83.