Preferences in Databases

Representation & Composition & Application

Georgia Koutrika(1), Evaggelia Pitoura(2), Kostas Stefanidis(2)

(1) Stanford University, (2) University of Ioannina
Preferences guide human decisions
  e.g., “which ice-cream flavor to buy?”
  “which investment funds to choose?”

Preferences have been studied in philosophy, psychology, economics, etc
  e.g., in philosophy: reasoning on values, desires, duties

TODAY’s topic: Preferences in Databases

G. Koutrika, E. Pitoura and K. Stefanidis
Why considering preferences in databases?

What are the challenges?

What has been done so far?

What next?
Why Preferences in Databases?

- The Boolean database answer model: all or nothing!
  - Empty-answer problem
  - Too-many-answers problem

- Databases on the Web: 7,500TB (19TB is the surface Web)!
  - National Climatic Data Center (NOAA)
  - NASA EOSDIS
  - Alexandria Digital Library
  - JSTOR Project Limited
  - US Census
  - Amazon.com
  - …

G. Koutrika, E. Pitoura and K. Stefanidis
The Boolean database answer model: all or nothing!

- Empty-answer problem
- Too-many-answers problem

Databases on the Web: 7,500TB (19TB is the surface Web!)

- Unknown schema
- Unknown contents

On the Web: Too much information

- Information Overload
- User diversity
Why Preferences in Databases?

Incorporating preferences can help return non-empty answers.

G. Koutrika, E. Pitoura and K. Stefanidis
Incorporating preferences can help return non-empty answers.
Incorporating preferences can help return focused answers

G. Koutrika, E. Pitoura and K. Stefanidis
Incorporating preferences can help return focused answers.
Tutorial Overview

- Preference Representation
- Preference Composition
- Preferential Query Processing
- Preference Learning
Example

<table>
<thead>
<tr>
<th>movie</th>
<th>play</th>
<th>actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>mid</td>
<td>mid</td>
<td>aid</td>
</tr>
<tr>
<td>title</td>
<td>title</td>
<td>aid</td>
</tr>
<tr>
<td>year</td>
<td>year</td>
<td>name</td>
</tr>
<tr>
<td>director</td>
<td>genre</td>
<td>date_of_birth</td>
</tr>
<tr>
<td>language</td>
<td>language</td>
<td></td>
</tr>
<tr>
<td>duration</td>
<td>duration</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>movie</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mid</td>
<td>t₁</td>
</tr>
<tr>
<td>title</td>
<td>m₁ Casablanca</td>
</tr>
<tr>
<td>year</td>
<td>1942</td>
</tr>
<tr>
<td>director</td>
<td>Curtiz</td>
</tr>
<tr>
<td>genre</td>
<td>drama</td>
</tr>
<tr>
<td>language</td>
<td>english</td>
</tr>
<tr>
<td>duration</td>
<td>102</td>
</tr>
<tr>
<td>mid</td>
<td>t₂</td>
</tr>
<tr>
<td>title</td>
<td>m₂ Psycho</td>
</tr>
<tr>
<td>year</td>
<td>1960</td>
</tr>
<tr>
<td>director</td>
<td>Hitchcock</td>
</tr>
<tr>
<td>genre</td>
<td>horror</td>
</tr>
<tr>
<td>language</td>
<td>english</td>
</tr>
<tr>
<td>duration</td>
<td>109</td>
</tr>
<tr>
<td>mid</td>
<td>t₃</td>
</tr>
<tr>
<td>title</td>
<td>m₃ Schindler's List</td>
</tr>
<tr>
<td>year</td>
<td>1993</td>
</tr>
<tr>
<td>director</td>
<td>Spielberg</td>
</tr>
<tr>
<td>genre</td>
<td>drama</td>
</tr>
<tr>
<td>language</td>
<td>english</td>
</tr>
<tr>
<td>duration</td>
<td>109</td>
</tr>
</tbody>
</table>
Formulation

- Qualitative approaches
- Quantitative approaches
Preferences between tuples in the answer to a query are specified directly using binary preference relations.

[Chomicki 2003; Kiessling 2002]

Given a relation $R$:
A preference relation $B$ is a subset of $R \times R$

$a \, B \, b$ between tuples $a$ and $b$ of $R \Rightarrow a$ is preferred over $b$
## Properties of binary relations

<table>
<thead>
<tr>
<th>Property</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reflexive</td>
<td>$a \ B \ a, \ \forall \ a \ in \ R$</td>
</tr>
<tr>
<td>Irreflexive</td>
<td>$\neg (a \ B \ a), \ \forall \ a \ in \ R$</td>
</tr>
<tr>
<td>Symmetric</td>
<td>$a \ B \ b \Rightarrow b \ B \ a, \ \forall \ a, b \ in \ R$</td>
</tr>
<tr>
<td>Transitive</td>
<td>$(a \ B \ b) \land (b \ B \ c) \Rightarrow (a \ B \ c), \ \forall \ a, b, c \ in \ R$</td>
</tr>
<tr>
<td>Asymmetric</td>
<td>$(a \ B \ b) \Rightarrow \neg (b \ B \ a), \ \forall \ a, b \ in \ R$</td>
</tr>
<tr>
<td>Antisymmetric</td>
<td>$(a \ B \ b) \land (b \ B \ a) \Rightarrow (a = b), \ \forall \ a, b \ in \ R$</td>
</tr>
<tr>
<td>Negative transitive</td>
<td>$\neg (a \ B \ b) \land \neg (b \ B \ c) \Rightarrow \neg (a \ B \ c), \ \forall \ a, b, c \ in \ R$</td>
</tr>
<tr>
<td>Connective</td>
<td>$(a \ B \ b) \lor (b \ B \ a) \lor (a = b), \ \forall \ a, b \ in \ R$</td>
</tr>
</tbody>
</table>
Formulation: Qualitative Approaches

Types of binary relations

Tuples in $R$

- Connective
- Irreflexive
- Asymmetric
- Transitive

Total Order

G. Koutrika, E. Pitoura and K. Stefanidis
Formulation: Qualitative Approaches

**Types of binary relations**

- Irreflexive
- Asymmetric
- Transitive

Strict Partial Order

G. Koutrika, E. Pitoura and K. Stefanidis
Formulation: Qualitative Approaches

Types of binary relations

- Tuples in $R$

- Negative transitive
- Irreflexive
- Asymmetric
- Transitive

Weak Order

G. Koutrika, E. Pitoura and K. Stefanidis
Formulation: Qualitative Approaches

Logical formulas

A logical formula $PF$ expresses the constraints two tuples must satisfy so that one is preferred over the other

$$t_i \succ_{PF} t_j \iff t_i[\text{genre}] = t_j[\text{genre}] \land t_i[\text{duration}] < t_j[\text{duration}]$$

Casablanca is preferred over Schindler’s list

G. Koutrika, E. Pitoura and K. Stefanidis
Formulation: Qualitative Approaches

Preference Constructors

A formal language for formulating preference relations using constructors

\[\text{HIGHEST}(A) \quad \{t_i \succ_{p\_new} t_j \iff t_i > t_j\}\]

\[\text{AROUND}(A, z) \quad \{t_i \succ_{p\_new} t_j \iff \text{abs}(t_i - z) < \text{abs}(t_j - z)\}\]

[Kiessling 2002]
Formulation: Qualitative Approaches

Preference Constructors

A formal language for formulating preference relations using constructors

[Kiessling 2002]

<table>
<thead>
<tr>
<th>movie</th>
<th>mid</th>
<th>title</th>
<th>year</th>
<th>director</th>
<th>genre</th>
<th>language</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>m₁</td>
<td>Casablanca</td>
<td>1942</td>
<td>Curtiz</td>
<td>drama</td>
<td>english</td>
<td>102</td>
</tr>
<tr>
<td>t₂</td>
<td>m₂</td>
<td>Psycho</td>
<td>1960</td>
<td>Hitchcock</td>
<td>horror</td>
<td>english</td>
<td>109</td>
</tr>
<tr>
<td>t₃</td>
<td>m₃</td>
<td>Schindler’s List</td>
<td>1993</td>
<td>Spielberg</td>
<td>drama</td>
<td>english</td>
<td>109</td>
</tr>
</tbody>
</table>

POS(genre, {horror})

NEG(year, {1960})

EXP(title, {(Casablanca), (Psycho), (Schindler’s list)})

G. Koutrika, E. Pitoura and K. Stefanidis
Preferences for tuples are expressed using functions that assign a score

\[ t_i \succ_P t_j \text{ for a preference function } f_P \Leftrightarrow f_P(t_i) > f_P(t_j) \]

(with exceptions [Guo et al. 2008])

[Agrawal et al. 2000]
Formulation: Quantitative Approaches

Preference Functions

Example

<table>
<thead>
<tr>
<th>movie</th>
<th>mid</th>
<th>title</th>
<th>year</th>
<th>director</th>
<th>genre</th>
<th>language</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_1</td>
<td>m_1</td>
<td>Casablanca</td>
<td>1942</td>
<td>Curtiz</td>
<td>drama</td>
<td>english</td>
<td>102</td>
</tr>
<tr>
<td>t_2</td>
<td>m_2</td>
<td>Psycho</td>
<td>1960</td>
<td>Hitchcock</td>
<td>horror</td>
<td>english</td>
<td>109</td>
</tr>
<tr>
<td>t_3</td>
<td>m_3</td>
<td>Schindler's List</td>
<td>1993</td>
<td>Spielberg</td>
<td>drama</td>
<td>english</td>
<td>109</td>
</tr>
</tbody>
</table>

\[ f_P(t_i) = 0.001 \times t_i[\text{duration}] \]
Preferences for tuples are expressed by specifying constraints for the tuples and assigning scores in these constraints


Preference \((\text{Condition}, \text{Score})\):

\text{Condition}: A_1 \theta_1 v_1 \land A_2 \theta_2 v_2 \land \ldots \land A_n \theta_n v_n

\text{Score} \text{ belongs to a predefined numerical domain}

\text{movie.genre = ‘drama’, 0.9}

\text{movie.year > 1990, 0.8}
Formulation

**Incompleteness**

Represents a gap in our knowledge

**Indifference**

\[ t_i \sim t_j \iff \neg (t_i >_{PR} t_j) \land \neg (t_j >_{PR} t_i) \]  
 qualitative

\[ \iff f_P(t_i) = f_P(t_j) \]  
 quantitative

**Incomparability**

Tuples that cannot be compared in some fundamental way

G. Koutrika, E. Pitoura and K. Stefanidis
If a preference relation $\succ_{\text{PR}}$ is weak order, then indifference is an equivalence class.

A binary relation is an equivalence class if it is reflexive, symmetric and transitive.
Incomparability

Example

\[ a \] dominates \( e \) and \( b \)

\( e \) and \( b \) are indifferent

\( b \) and \( c \) are indifferent

**BUT**: \( e \) dominates \( c \)

The indifference relation fails to capture incomparable versus equally important tuples

G. Koutrika, E. Pitoura and K. Stefanidis
Qualitative vs Quantitative

In a quantitative way: I like comedies a lot!
Qualitative cannot capture priority, importance, feeling

In a qualitative way: between two movies of the same kind,
I prefer the shortest
Quantitative is more restricted

Example

<table>
<thead>
<tr>
<th>mid</th>
<th>title</th>
<th>year</th>
<th>director</th>
<th>genre</th>
<th>language</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>m₁</td>
<td>Casablanca</td>
<td>1942</td>
<td>Curtiz</td>
<td>drama</td>
<td>english</td>
<td>102</td>
</tr>
<tr>
<td>m₂</td>
<td>Psycho</td>
<td>1960</td>
<td>Hitchcock</td>
<td>horror</td>
<td>english</td>
<td>109</td>
</tr>
<tr>
<td>m₃</td>
<td>Schindler's List</td>
<td>1993</td>
<td>Spielberg</td>
<td>drama</td>
<td>english</td>
<td>109</td>
</tr>
</tbody>
</table>

\[ t₃ \text{ is preferred over } t₁ \text{ and } t₂ \text{ is incomparable} \]

G. Koutrika, E. Pitoura and K. Stefanidis
Preference representation dimensions

- Formulation
- Granularity
- Context
- Aspects
Granularity

Tuple Preferences

Preferences expressed directly for tuples and their values

\[
\begin{align*}
\text{movie.genre} &= \text{‘drama’}, & 0.9 \\
\text{movie.mid} &= \text{cast.mid and} \\
\text{cast.aid} &= \text{actor.aid and} \\
\text{actor.name} &= \text{‘J. Roberts’}, & 0.7 \\
\end{align*}
\]

[Koutrika and Ioannidis 2010]
Granularity

Set Preferences

Preferences expressed based on the properties of a group of tuples as a whole

[Zhang and Chomicki 2008]

I want to see three movies of the same director

G. Koutrika, E. Pitoura and K. Stefanidis
Granularity

Attribute Preferences

They can set priorities among tuple preferences expressed over the values in the corresponding attributes

\[ P_{\text{director}} > P_{\text{genre}} \]  
[Georgiadis et al 2008]

They can set priorities among the attributes to be displayed in the results

\{\text{title, genre, language}\}, 1  
\{\text{year, director, duration}\}, 0.3  
[Miele at al 2009]

G. Koutrika, E. Pitoura and K. Stefanidis
Granularity

Relationship Preferences

They are expressed on relationships between two types of entities or two particular entities

\[ (\text{movie}.\text{mid} = \text{play}.\text{mid}, 1) \]  [Koutrika, Ioannidis 2004]

A director has directed many movies

Julia Roberts has acted in Ocean’s Eleven

G. Koutrika, E. Pitoura and K. Stefanidis
One more example…
Preference representation dimensions

- Formulation
- Granularity
- Context
- Aspects
Context is any information that can be used to characterize the situation of an entity

An entity is a person, place, object that is considered relevant to the interaction between a user and an application, including the user and the application themselves

[Dey 2001]

User preferences can be part of the user context!

We study how context determined when user preferences hold
Context is any external to the database information that can be used to characterize the situation of a user or any internally stored information that can be used to characterize the data per se.
Context

Contextual Preferences

(C, P), where C defines the context and P defines the preference

C → **internal contextual preferences**
  e.g., for **dramas**, I prefer movies directed by **Spielberg**

→ **external contextual preferences**
  e.g., when with **friends**, I prefer to watch **horror** movies

<table>
<thead>
<tr>
<th>movie</th>
<th>mid</th>
<th>title</th>
<th>year</th>
<th>director</th>
<th>genre</th>
<th>language</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>t&lt;sub&gt;1&lt;/sub&gt;</td>
<td>m&lt;sub&gt;1&lt;/sub&gt;</td>
<td>Casablanca</td>
<td>1942</td>
<td>Curtiz</td>
<td>drama</td>
<td>english</td>
<td>102</td>
</tr>
<tr>
<td>t&lt;sub&gt;2&lt;/sub&gt;</td>
<td>m&lt;sub&gt;2&lt;/sub&gt;</td>
<td>Psycho</td>
<td>1960</td>
<td>Hitchcock</td>
<td>horror</td>
<td>english</td>
<td>109</td>
</tr>
<tr>
<td>t&lt;sub&gt;3&lt;/sub&gt;</td>
<td>m&lt;sub&gt;3&lt;/sub&gt;</td>
<td>Schindler's List</td>
<td>1993</td>
<td>Spielberg</td>
<td>drama</td>
<td>english</td>
<td>109</td>
</tr>
</tbody>
</table>

G. Koutrika, E. Pitoura and K. Stefanidis
Internal Contextual Preferences

Given a relation with attributes $A_1, \ldots, A_d$, an internal context is:

$$\wedge_{j \in L}(A_j = v_j), \ L \subseteq \{A_1, \ldots, A_d\}$$

Example

<table>
<thead>
<tr>
<th>movie</th>
<th>mid</th>
<th>title</th>
<th>year</th>
<th>director</th>
<th>genre</th>
<th>language</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_1</td>
<td>m_1</td>
<td>Casablanca</td>
<td>1942</td>
<td>Curtiz</td>
<td>drama</td>
<td>english</td>
<td>102</td>
</tr>
<tr>
<td>t_2</td>
<td>m_2</td>
<td>Psycho</td>
<td>1960</td>
<td>Hitchcock</td>
<td>horror</td>
<td>english</td>
<td>109</td>
</tr>
<tr>
<td>t_3</td>
<td>m_3</td>
<td>Schindler's List</td>
<td>1993</td>
<td>Spielberg</td>
<td>drama</td>
<td>english</td>
<td>109</td>
</tr>
</tbody>
</table>

{director = ‘Spielberg’ > director = ‘Curtiz’ | genre = ‘drama’}  

$t_3$ is preferred over $t_1$
Internal Contextual Preferences

Example [Chomicki 2003]

<table>
<thead>
<tr>
<th>movie</th>
<th>mid</th>
<th>title</th>
<th>year</th>
<th>director</th>
<th>genre</th>
<th>language</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>m₁</td>
<td>Casablanca</td>
<td>1942</td>
<td>Curtiz</td>
<td>drama</td>
<td>english</td>
<td>102</td>
</tr>
<tr>
<td>t₂</td>
<td>m₂</td>
<td>Psycho</td>
<td>1960</td>
<td>Hitchcock</td>
<td>horror</td>
<td>english</td>
<td>109</td>
</tr>
<tr>
<td>t₃</td>
<td>m₃</td>
<td>Schindler’s List</td>
<td>1993</td>
<td>Spielberg</td>
<td>drama</td>
<td>english</td>
<td>109</td>
</tr>
</tbody>
</table>

\[ t_i \succ_{PF} t_j \iff (t_i[\text{genre}] = t_j[\text{genre}] \land t_i[\text{genre}] = 'drama' \land t_i[\text{director}] = 'Spielberg' \land t_j[\text{director}] = 'Curtiz') \lor (t_i[\text{genre}] = t_j[\text{genre}] \land t_i[\text{genre}] = 'thriller' \land t_j[\text{director}] = 'Spielberg' \land t_i[\text{director}] = 'Curtiz') \]
Given a set of contextual parameters $C_1, \ldots, C_n$, an external context is:
a n-tuple $(c_1, \ldots, c_n)$, where $c_i \in C_i$

Example

CP1: (Time_period = ‘All’, genre = ‘adventure’)
CP2: (Time_period = ‘Holidays’, language = ‘Greek’)
CP3: (Time_period = ‘Holidays’, director = ‘Hitchcock’)

G. Koutrika, E. Pitoura and K. Stefanidis
Preference representation dimensions

- Formulation
- Granularity
- Context
- Aspects
Aspects

Intensity

It shows the degree of desire expressed in a preference

- Weak preferences
  \[ \text{movie.genre} = \text{‘cartoons’}, 0.4 \]

- Strong preferences
  \[ \text{movie.genre} = \text{‘comedy’}, 0.9 \]
Aspects

Necessity

It shows whether a preference should be met

- Hard/mandatory preferences
  
  When with friends, I do not want to see a drama movie

- Soft OPTIONAL preferences

  An optional preference for director W. Allen
Aspects

Feeling

It shows how one feels about something

- Positive preferences
  movie.genre = 'drama', 0.9

- Negative preferences
  movie.genre = 'horror', -0.5

G. Koutrika, E. Pitoura and K. Stefanidis
Preference representation approaches w.r.t. preference formulation, granularity and context

<table>
<thead>
<tr>
<th>Reference</th>
<th>Formulation</th>
<th>Granularity</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Agrawal and Wimmers 2000]</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>[Agrawal et al. 2006]</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Bunningen et al. 2006; 2007]</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[Chomicki 2002; 2003]</td>
<td>✓</td>
<td></td>
<td>✓ ✓</td>
</tr>
<tr>
<td>[Georgiadis et al. 2008]</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Holland and Kiessling 2004]</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Kiessling 2002]</td>
<td>✓ ✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Koutrika and Ioannidis 2004; 2005]</td>
<td>✓ ✓</td>
<td></td>
<td>✓ ✓</td>
</tr>
<tr>
<td>[Miele et al. 2009]</td>
<td>✓ ✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>[Stefanidis et al. 2006; 2007]</td>
<td>✓ ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Zhang and Chomicki 2008]</td>
<td>✓</td>
<td>sets ✓</td>
<td></td>
</tr>
</tbody>
</table>
Preference representation approaches w.r.t preference aspects (T=tuple, C=relation, A=attribute, R=relationship)

<table>
<thead>
<tr>
<th>Aspects</th>
<th>Intensity</th>
<th>Necessity</th>
<th>Feeling</th>
<th>Complexity</th>
<th>Attitude</th>
<th>Elasticity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Strong</td>
<td>Weak</td>
<td>Hard</td>
<td>Soft</td>
<td>Positive</td>
<td>Negative</td>
</tr>
<tr>
<td>[Agrawal and Wimmers 2000]</td>
<td>T</td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td>[Agrawal et al. 2006]</td>
<td>T</td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td>[Bunningen et al. 2006; 2007]</td>
<td>T</td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td>[Chomicki 2002; 2003]</td>
<td>T</td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td>[Holland and Kiessling 2004]</td>
<td>T</td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td>[Kiessling 2002]</td>
<td>T</td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td>[Koutrika and Ioannidis 2004; 2005]</td>
<td>T</td>
<td>T</td>
<td>TR</td>
<td>TR</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>[Stefanidis et al. 2006; 2007]</td>
<td>T</td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>T</td>
<td>-</td>
</tr>
<tr>
<td>[Zhang and Chomicki 2008]</td>
<td>T</td>
<td>T</td>
<td>-</td>
<td>T</td>
<td>T</td>
<td>-</td>
</tr>
</tbody>
</table>
Composition mechanisms defined over preference relations

- Prioritized Composition
  - E.g., $P_x$ is considered more important than $P_y$
- Pareto Composition
  - Equally important preference relations
- Pair-wise Comparisons Composition
- Set-oriented Composition
  - Intersection, Union, Difference

In following, we assume composition of two preferences $P_x$ and $P_y$; generalizing to $n > 2$ preferences is straightforward
Let $P_x, P_y$ be two preference relations defined over the relational schema $R$

- The prioritized preference composition relation $\succ_{P_x \& P_y}$ is defined over $R$, such that, $\forall t_i, t_j \text{ of } R$, $t_i \succ_{P_x \& P_y} t_j$, iff:
  $$(t_i \succ_{P_x} t_j) \lor (t_i \sim_{P_x} t_j \land t_i \succ_{P_y} t_j)$$
Prioritized Composition

Example:
P1: dramas over horrors
P2: long movies over short ones

For $t_i$, $t_j$, $t_i \succ_{P1\&P2} t_j$, iff:

$$(t_i[\text{genre}] = \textit{drama} \land t_j[\text{genre}] = \textit{horror}) \lor
(t_i[\text{genre}] \neq \textit{drama} \land t_i[\text{duration}] > t_j[\text{duration}]) \lor
(t_j[\text{genre}] \neq \textit{horror} \land t_i[\text{duration}] > t_j[\text{duration}])$$

$t_3$ is preferred over $t_1$
$t_1$ is preferred over $t_2$

<table>
<thead>
<tr>
<th>movie</th>
<th>mid</th>
<th>title</th>
<th>year</th>
<th>director</th>
<th>genre</th>
<th>language</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>$m_1$</td>
<td>Casablanca</td>
<td>1942</td>
<td>Curtiz</td>
<td>drama</td>
<td>english</td>
<td>102</td>
</tr>
<tr>
<td>$t_2$</td>
<td>$m_2$</td>
<td>Psycho</td>
<td>1960</td>
<td>Hitchcock</td>
<td>horror</td>
<td>english</td>
<td>109</td>
</tr>
<tr>
<td>$t_3$</td>
<td>$m_3$</td>
<td>Schindler’s List</td>
<td>1993</td>
<td>Spielberg</td>
<td>drama</td>
<td>english</td>
<td>109</td>
</tr>
</tbody>
</table>
Prioritized composition over different relational schemas

**Lexicographical Composition**

For $P_x, P_y$ defined over $R, R’$ with attribute domains $\text{dom}(A), \text{dom}(A’)$

- The *lexicographical preference composition* relation $\succ_{P_x&P_y}$ defined over $R \times R’$, is a subset of $\text{dom}(A) \times \text{dom}(A’)$, such that,

$$(t_i, t_i’) \succ_{P_x&P_y} (t_j, t_j’), \text{ iff: } (t_i \succ_{P_x} t_j) \lor (t_i \sim_{P_x} t_j \land t_i’ \succ_{P_y} t_j’)$$

$t_i, t_j$ are tuples of $R$ and $t_i’, t_j’$ tuples of $R’$

[Chomicki 2003]:

- Total and weak orders are preserved by the prioritized and lexicographical composition
- Strict partial order is not

G. Koutrika, E. Pitoura and K. Stefanidis
For $P_x, P_y$ defined over $R$

- The **pareto preference composition** relation $\succ_{P_x \circ P_y}$ is defined over $R$, such that, $\forall t_i, t_j$ of $R$, $t_i \succ_{P_x \circ P_y} t_j$, iff:

$$(t_i \succ_{P_x} t_j \land \neg (t_j \succ_{P_y} t_i)) \lor (t_i \succ_{P_y} t_j \land \neg (t_j \succ_{P_x} t_i))$$

Intuitively, under pareto composition, a tuple dominates another if it is at least as good (i.e., not worse) under one preference and strictly better under the other.

G. Koutrika, E. Pitoura and K. Stefanidis
Qualitative Composition

Pareto Composition

Example:
P1: dramas over horrors
P2: long movies over short ones

For \( t_i, t_j \), \( t_i \succ_{P1 \otimes P2} t_j \), iff: 
\[
(t_i[\text{genre}] = 'drama' \land t_j[\text{genre}] = 'horror' \land t_i[\text{duration}] \geq t_j[\text{duration}]) \lor
(t_i[\text{duration}] > t_j[\text{duration}] \land t_j[\text{genre}] \neq 'drama') \lor
(t_i[\text{duration}] > t_j[\text{duration}] \land t_j[\text{genre}] = 'drama' \land t_i[\text{genre}] \neq 'horror')
\]

\( t_3 \) is preferred over \( t_1 \)
\( t_1, t_2 \) are incomparable

<table>
<thead>
<tr>
<th>movie</th>
<th>mid</th>
<th>title</th>
<th>year</th>
<th>director</th>
<th>genre</th>
<th>language</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>( m_1 )</td>
<td>Casablanca</td>
<td>1942</td>
<td>Curtiz</td>
<td>drama</td>
<td>english</td>
<td>102</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>( m_2 )</td>
<td>Psycho</td>
<td>1960</td>
<td>Hitchcock</td>
<td>horror</td>
<td>english</td>
<td>109</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>( m_3 )</td>
<td>Schindler's List</td>
<td>1993</td>
<td>Spielberg</td>
<td>drama</td>
<td>english</td>
<td>109</td>
</tr>
</tbody>
</table>
Pareto composition over different relational schemas

**Multidimensional Pareto Composition**

For $P_x$, $P_y$ defined over $R$, $R'$ with attribute domains $\text{dom}(A)$, $\text{dom}(A')$

- The *multidimensional pareto preference* relation $\succ_{P_x \otimes P_y}$ defined over $R \times R'$ is a subset of $\text{dom}(A) \times \text{dom}(A')$, such that,

\[
(t_i, t'_i) \succ_{P_x \otimes P_y} (t_j, t'_j), \text{ iff: } (t_i \succ_{P_x} t_j \land \neg (t'_j \succ_{P_y} t'_i)) \lor (t'_i \succ_{P_y} t'_j \land \neg (t_j \succ_{P_x} t_i))
\]

$t_i, t_j$ are tuples of $R$ and $t'_i, t'_j$ tuples of $R'$
Motivation: Voting theory [Condorcet 1785]

Given a set of preference relations:

$t_i$ is preferred over $t_j$, iff, $t_i$ is preferred over $t_j$ for the majority of the preference relations

Other methods of voting theory:

- Given a set of rankings, tuples are ordered based on the number of times each one appears first
- [Borda 1781]: determine the position of a tuple by the sum of its positions in the initial rankings

G. Koutrika, E. Pitoura and K. Stefanidis
Qualitative Composition

For $P_x$, $P_y$ defined over the relational schema $R$

- The **intersection preference relation** $>_{P_x \land P_y}$ is defined over $R$, such that, $\forall t_i, t_j$ of $R$, $t_i >_{P_x \land P_y} t_j$, iff:
  \[ t_i >_{P_x} t_j \land t_i >_{P_y} t_j \]

- The **union preference relation** $>_{P_x + P_y}$ is defined over $R$, such that, $\forall t_i, t_j$ of $R$, $t_i >_{P_x + P_y} t_j$, iff:
  \[ t_i >_{P_x} t_j \lor t_i >_{P_y} t_j \]

- The **difference preference relation** $>_{P_x - P_y}$ is defined over $R$, such that, $\forall t_i, t_j$ of $R$, $t_i >_{P_x - P_y} t_j$, iff:
  \[ t_i >_{P_x} t_j \land \neg(t_i >_{P_y} t_j) \]

G. Koutrika, E. Pitoura and K. Stefanidis
**Intersection example:**
P1: dramas over horrors
P2: long movies over short ones

P1 \land P2: t_i \succ_{P_1 \land P_2} t_j, iff:
(t_i[genre] = ‘drama’ \land t_j[genre] = ‘horror’) \land (t_i[duration] > t_j[duration])

[Chomicki 2003]:
- Strict partial order is preserved by intersection but not by difference or union
- None of the set-oriented composition operators preserve the weak and the total order

G. Koutrika, E. Pitoura and K. Stefanidis
Preference composition mechanism categories:

- Qualitative composition
- **Quantitative composition**
  - Combine preferences expressed as scores over a set of tuples and assign final scores to these tuples
- Heterogeneous composition
Given:
- Two preferences $P_x, P_y$ over $\mathbb{R}$ defined through preference functions $f_{P_x}, f_{P_y}$
- A combining function $F : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$

$\forall t_i, t_j \in \mathbb{R}, t_i \succ_{\text{rank}_F(P_x,P_y)} t_j$, iff: $F(f_{P_x}(t_i), f_{P_y}(t_i)) > F(f_{P_x}(t_j), f_{P_y}(t_j))$
To assign importance to preferences, weights can be used

**Example:**
- P1: $f_{P1}(t_i) = 0.001 \times t_i[\text{duration}]$
- P2: $f_{P2}(t_i) = 0.0001 \times t_i[\text{year}]$

$F(P1, P2): F(f_{P1}(t_i), f_{P2}(t_i)) = 0.1 \times f_{P1}(t_i) + 0.9 \times f_{P2}(t_i)$

Under this preference:
- score(t1) = 0.185
- score(t2) = 0.187
- score(t3) = 0.199

<table>
<thead>
<tr>
<th>movie</th>
<th>mid</th>
<th>title</th>
<th>year</th>
<th>director</th>
<th>genre</th>
<th>language</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>m1</td>
<td>Casablanca</td>
<td>1942</td>
<td>Curtiz</td>
<td>drama</td>
<td>english</td>
<td>102</td>
</tr>
<tr>
<td>t2</td>
<td>m2</td>
<td>Psycho</td>
<td>1960</td>
<td>Hitchcock</td>
<td>horror</td>
<td>english</td>
<td>109</td>
</tr>
<tr>
<td>t3</td>
<td>m3</td>
<td>Schindler's List</td>
<td>1993</td>
<td>Spielberg</td>
<td>drama</td>
<td>english</td>
<td>109</td>
</tr>
</tbody>
</table>

Also: Numerical composition over different relational schemas

G. Koutrika, E. Pitoura and K. Stefanidis
Other types of combining functions:
- The \textit{min} and \textit{max} functions

Three classes of combining functions:
- \textbf{Inflationary}: the preference in a tuple increases with the number of preferences that satisfy it
- \textbf{Dominant}: the most important preference dominates
- \textbf{Reserved}: the preference in a tuple is between the highest and the lowest degrees of interest among the preferences satisfied

[\textit{Koutrika and Ioannidis 2005b}]
Let $P_x$, $P_y$ be two preferences defined over the relational schema $R$.

If $P_x$ refers to a subset of tuples that $P_y$ refers to, the more specific one, i.e., $P_x$, overrides the more generic one.

Example:

$P_1$: movie: (movie.genre = 'comedy', 0.9)

$P_2$: movie: (movie.genre = 'comedy' and movie.director = 'Stiller', -0.9)

$P_2$ overrides $P_1$ whenever they both apply.

[Koutrika and Ioannidis 2010]
Qualitative vs. Quantitative Composition

Note:

Every composition mechanism defined over preference relations can be applied to preferences defined using functions or degrees of interest

This way:
- Prioritized, lexicographical, pareto, intersection, union and difference composition are also applicable to numerical preferences

G. Koutrika, E. Pitoura and K. Stefanidis
So far, we have distinguished composition methods based on the tuple ranking criterion between:

- **Qualitative**
- **Quantitative**

Distinguish composition methods based on the user attitude:

- **Overriding attitude**: Preference $P_x$ overriding $P_y$ means that $P_y$ is applicable only if $P_x$ does not apply
- **Dominant attitude**: The most or least important preference determines the tuple ranking
- **Combinatory attitude**: Both $P_x$ and $P_y$ contribute to the tuple ranking
Preference composition w.r.t. tuple ranking and user attitude

<table>
<thead>
<tr>
<th>Tuple Ranking</th>
<th>Qualitative</th>
<th>Overriding</th>
<th>Dominant</th>
<th>Combinatory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>prioritized, lexicographical</td>
<td>--</td>
<td></td>
<td>pareto, multidimensional pareto, pair-wise comparisons, intersection, difference, union</td>
</tr>
<tr>
<td></td>
<td>syntactic overriding</td>
<td>max, min</td>
<td></td>
<td>average, weighted average, ...</td>
</tr>
</tbody>
</table>
So far, we have focused on:
- Mechanisms for composing preferences for tuples

Is this the only direction?

Next, we focus on:
- Combining preferences of different granularity
Mechanisms for composing preferences of different granularity

- Combine preferences expressed at tuple and relationship level
- Combine preferences expressed at tuple and attribute level
Heterogeneous Composition

Combine preferences expressed at **tuple** and **relationship** level

To do this:
Compose implicit preferences by other composeable ones

\( \text{P}_x \) and \( \text{P}_y \) are composeable, iff:

i. \( \text{P}_x \) is a join preference of the form \( \text{R}_x: (q_x, d_x) \) connecting \( \text{R}_x \) to a relation \( \text{R}_y \) and

ii. \( \text{P}_y \) is a join or selection preference on \( \text{R}_y \), i.e., \( \text{R}_y: (q_y, d_y) \)

[Koutrika and Ioannidis 2005b]

\( q_x \) and \( q_y \) are conditions, \( d_x \) and \( d_y \) are scores, \( \text{P}_x \) and \( \text{P}_y \) can be viewed as queries that select tuples from relations \( \text{R}_x \), \( \text{R}_y \) that satisfy \( q_x \), \( q_y \)

G. Koutrika, E. Pitoura and K. Stefanidis
Heterogeneous Composition

Combine preferences expressed at **tuple** and **relationship** level

**Example:**
Selection preference: actor: (actor.name = ‘Roberts’, 0.8)
Join preferences: movie: (movie.mid = play.mid, 1)
    play: (play.aid = actor.aid, 1)

**Implicit preference** for movies with Julia Roberts:
movie: (movie.mid = play.mid and
    play.aid = actor.aid and
    actor.name = ‘Roberts’, 0.8)
Heterogeneous Composition

Combine preferences expressed at **tuple** and **attribute** level

Employ attribute preferences to express priorities among tuple preferences

[Georgiadis et al. 2008]

**Example:**

**Tuple preferences:** Hitchcock is preferred to Curtiz or Spielberg ($P_D$)
- horror movies are preferred to dramas ($P_G$)

**Attribute preference:** the director of a movie is as important as its genre ($P_{DG}$)

$P_D$ and $P_G$ are combined by taking the **pareto preference composition** $P_D \otimes P_G$

- $P_{DG}$ expresses that $P_D$ and $P_G$ are equally important

<table>
<thead>
<tr>
<th>t2 is preferred to t1 and t3</th>
<th>t1, t3 are incomparable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>movie</strong></td>
<td></td>
</tr>
<tr>
<td>$mid$</td>
<td>title</td>
</tr>
<tr>
<td>t1</td>
<td>m1</td>
</tr>
<tr>
<td>t2</td>
<td>m2</td>
</tr>
<tr>
<td>t3</td>
<td>m3</td>
</tr>
</tbody>
</table>
Preference composition w.r.t. granularity

<table>
<thead>
<tr>
<th></th>
<th>Tuple</th>
<th>Relation</th>
<th>Attribute</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation</td>
<td></td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relationship</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Given a set of preferences:

How we can employ them to compute query results?

**Goal**: Exploit preferences to provide users with customized answers by *changing the order* and *possibly the size of results*
Tutorial Overview

Preference Representation

Preference Composition

Preferential Query Processing

- Expand Database Queries with Preferences
- Pre-compute Rankings of Tuples
- Top-k Processing
Three fundamental steps:

- **Preference relatedness**: determine which preferences are related and applicable to a query

- **Preference filtering**: identify which of the related preferences should be integrated into the query

- **Preference integration**: integrate the selected preferences into the original query to enable preferential query answering
Preference Relatedness

From a set of preferences known for a user at query time:

- All preferences may be considered related to the query
- Only a subset of preferences may be considered related to the query

Which of the available preferences we will use?
Assume:

**Example:**

\((C, P): (\text{Accompanying\_people} = 'friends', \text{genre} = 'horror')\)

\((C_Q, Q): (\text{Accompanying\_people} = 'friends', \text{SELECT title, FROM movie, WHERE director} = 'Hitchcock')\)
A preference \((C, P)\) is related to a query \((C_Q, Q)\) if:
- The external part of \(C\) matches \(C_Q\) and the internal part of \(C\) matches \(Q\)
- The preference part \(P\) is applicable to \(Q\)'s results

In what follows, we elaborate each part of the definition separately:
- **Context matching**
- Preference applicability
Use a metric for measuring the \textit{distance}, \textit{similarity} or \textit{difference} of two contexts:

- Vector-based approaches
  - Represent query and preference contexts as vectors and measure their similarity

- Hierarchical-based approaches

\[\text{[Agrawal et al. 2006]}\]
Context Matching: Hierarchical Approach

For context parameters that take values from hierarchical domains:

- Compare contexts expressed at different levels of abstraction

Given a preference \((C, P)\) and a query with context \(C_Q\):

- \(C\) is related to \(C_Q\), if \(C\) is equal or more general than \(C_Q\)

[Stefanidis et al. 2007a]

Example:
For the context parameter **Time_period**, the value **Holidays** is more general than the value **Christmas**
Hierarchical distance
Distance between C and C_Q: Sum of distances of the levels of all context parameters
- Distance between two levels: Minimum path between them in the hierarchy

Example:
The contexts (Athens, warm) and (Greece, good) have distance 1+1=2

A similar metric is used by [Miele et al. 2009]
- Take into account the depth of context values in the hierarchy

G. Koutrika, E. Pitoura and K. Stefanidis
Locate the related preferences using the profile tree
- Exploit the repetition of context values in contexts

Preferences (C, P):
- ((all, all, all), P₀)
- ((friends, good, summer holidays), P₁)
- ((family, good, summer holidays), P₂)
- ((friends, all, holidays), P₃)
- ((family, all, holidays), P₄)
- ((family, all, all), P₅)
- ((all, all, holidays), P₆)
A context parameter may be relaxed:
- **Upwards** by replacing its value by a more general one
- **Downwards** by replacing its value by a set of more specific ones
- **Sideways** by replacing its value by sibling values in the hierarchy

But how well C matches C’?
- Employ metrics that exploit the number of relaxed parameters and the depth of relaxations

[Stefanidis et al. 2007b]
Preference Relatedness

With context matching, we identify:
- Preferences that are valid in a query context
- Preferences that are out of context

It does not guarantee that a preference can be combined with the query and yield an interesting, non-empty output

Preference Applicability

We consider the following cases of preference applicability:

- Instance applicability
- Semantic applicability
- Syntactic applicability
Instance Applicability

P is **instantly applicable** to Q if:
Q, combined conjunctively with P, is executed over the current database instance and its result set is not empty

**Example:**
For a Q about **recent movies** and a P for movies directed by Spielberg:
- P is instantly applicable to Q only if the database contains recent entries of Steven Spielberg
For semantic applicability, additional knowledge, outside the database, is needed.

**Example:**
For a Q about **comedies**:
- A preference for movies directed by **Allen** is applicable
- A preference for **Tarkovsky** is not applicable
Semantic Applicability

For *semantic applicability*, additional knowledge, outside the database, is needed

**Note:**

When $P$ is instantly applicable to $Q$, then $P$ is also semantically applicable to $Q$
- The reverse does not apply

**Example:** For a $Q$ about recent movies and a $P$ for movies directed by Tarantino
- $P$ is semantically applicable to $Q$
- Assuming that our database is not updated, $P$ is not instantly applicable to $Q
A preference P is **syntactically applicable** to a query Q w.r.t. their structure
- That is, according to the relations, attributes and values P and Q contain

A P for the tuples of a relation R is applicable to Q, if:
- R is referenced in Q
- P is expressed over an attribute in Q

[ Koutrika and Ioannidis 2004 ]

**Note:**

The set of semantically applicable preferences for a query is a superset of the syntactically applicable ones

G. Koutrika, E. Pitoura and K. Stefanidis
Assume the query:
Q: (Time_period = ‘Christmas’, SELECT title FROM movie
WHERE genre = ‘horror’ AND language = ‘English’)

and the preferences:
CP1: (Time_period = ‘All’, genre = ‘adventure’)
CP2: (Time_period = ‘Holidays’, language = ‘Greek’)
CP3: (Time_period = ‘Holidays’, director = ‘Hitchcock’)

Preference Selection:
- CP2 and CP3 are more closely related to Q
- CP2 is not applicable to Q
- CP3 is syntactically, instantly and semantically applicable
Expand Database Queries

Three fundamental steps:

- **Preference relatedness**: determine which preferences are related and applicable to a query

- **Preference filtering**: identify which of the related preferences should be integrated into the query

- **Preference integration**: integrate the selected preferences into the original query to enable preferential query answering
All preferences related to a query may be used for ranking and selecting the tuples returned by the query

Alternatively: Rank preferences based on their:
- Relatedness score, capturing the degree to which a preference is related to a query
- Preference score, showing their intensity

Subsequently, select the top preferences for ranking the query results
Filtering based on Relatedness Score

Rank preferences based on their relatedness score
- Use a function to capture how well a preference context matches a query context

Use the cosine similarity to match contexts [Agrawal et al. 2006]

For hierarchical contexts:
Employ distance metrics that combine:
- The number of parameters in which the contexts differ
- The level at which such differences occur in the context hierarchies [Stefanidis et al. 2007a; Miele et al. 2009]

G. Koutrika, E. Pitoura and K. Stefanidis
Quantitative preferences are ordered in decreasing preference score and the top K ones are selected for expanding the query.
Filtering based on Preference Score

Extract the **top K related preferences** from a set $U$
- These preferences are stored explicitly in $U$ or are derived implicitly through preference composition

[ Koutrika and Ioannidis 2004 ]

**Example:**
**Selection preference:** actor: (actor.name = ‘Roberts’, 0.8)
**Join preferences:** movie: (movie.mid = play.mid, 1)
    play: (play.aid = actor.aid, 1)

**Implicit preference** for movies with Julia Roberts:
movie: (movie.mid = play.mid and play.aid = actor.aid and actor.name = ‘Roberts’, 0.8)
Preference Selection Algorithm

**Input:** Q, preferences U, interest criterion CI

**Output:** a set \( P_K \) of the top K related preferences derived from U

Start from the related to the query preferences \( Q_p \)

Iteratively consider additional preferences that are composeable with those already known

- At each round, pick from \( Q_p \) the candidate preference \( P \) with the highest degree of interest
  - A selection preference is added in \( P_K \), if it satisfies CI
  - A join preference is combined with the stored, composeable preferences to infer implicit preferences that can be applied to the query and satisfy CI
    - These implicit preferences are inserted into \( Q_p \)
- The algorithm stops when no other preferences satisfying CI can be derived and returns \( P_K \)

CI examples: preferences with degrees of interest greater than a threshold, at most x preferences could be output etc.
Three fundamental steps:

- **Preference relatedness**: determine which preferences are related and applicable to a query

- **Preference filtering**: identify which of the related preferences should be integrated into the query

- **Preference integration**: integrate the selected preferences into the original query to enable preferential query answering
Preferences expressed as query conditions can be naturally integrated into a query

- **Query rewriting** approaches leverage the power of SQL to return results that satisfy the user preferences

Use the top K preferences for query personalization

- Query results satisfy **at least L of the K preferences**
  - K: Desired degree of personalization
  - L: Minimum number of criteria that an answer should meet

  [Koutrika and Ioannidis 2004]

Two different query re-writing mechanisms:

i. **Single query**: A conjunction of query conditions with the disjunction of all possible conjunctions of the L out of K preferences

ii. **K queries**: Augment the initial query with one of the K preferences
  - Each tuple that appears at least L times is output
Example:
Assume the query

Q: SELECT title FROM movie WHERE director = ‘Spielberg’

and the preferences

P1: (genre = ‘drama’)
P2: (language = ‘English’)  \( (L = 1) \)

Mechanism ii

SELECT distinct title FROM (  
(SELECT distinct title FROM movie  
WHERE director = ‘Spielberg’ AND genre = ‘drama’) 
UNION ALL  
(SELECT distinct title FROM movie  
WHERE director = ‘Spielberg’ AND language = ‘English’)  
)
Blocks, or groups, of equivalent queries
- Each block consists of a set of queries that generate equally preferable results

[Georgiadis et al. 2008]

Example preferences:
- Hitchcock is preferred over Curtiz or Spielberg
- Horror movies are preferred over dramas
- The director of a movie is as important as its genre
Three fundamental steps:

- **Preference relatedness**: determine which preferences are related and applicable to a query
  - All preferences
  - Context matching
  - Preference applicability

- **Preference filtering**: identify which of the related preferences should be integrated into the query
  - Preference relatedness
  - Preference score

- **Preference integration**: integrate the selected preferences into the original query to enable preferential query answering
A taxonomy of approaches that expand database queries with preferences

<table>
<thead>
<tr>
<th></th>
<th>Preference Relatedness</th>
<th>Preference Filtering</th>
<th>Preference Integration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All Preferences</td>
<td>Context Matching</td>
<td>Preference Applicability</td>
</tr>
<tr>
<td>[Agrawal et al. 2006]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Bunningen et al. 2006]</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>[Georgiadis et al. 2008]</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>[Koutrika and Ioannidis 2004; 2005]</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>[Miele et al. 2009]</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>[Stefanidis et al. 2007]</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Preference integration

- **Employ preference operators**
Preferences can be embedded into query languages through preference-related operators
- Select from input the set of the most preferred tuples

Two fundamentals approaches to handle preference operators:
- **Operator implementation**
  - Operators are implemented inside the database engine
    - Employ special evaluation algorithmic techniques
- **Operator translation**
  - Operators are translated into other, existing relational algebra operators

G. Koutrika, E. Pitoura and K. Stefanidis
In following, we focus on:

- **Defining preference operators**
- Implementing preference operators
- Translating preference operators
The winnow operator: Pick from an instance $r$ the set of the most preferred tuples w.r.t. a preference relation $P$ [Chomicki 2003]

Definition

Given an instance $r$ of a relational schema $R$ and a $P$ over $R$:
The winnow operator $w_P(r)$ is

$$w_P(r) = \{t_i \in r \mid \nexists t_j \in r, \text{ such that } t_j \succ_P t_i \}$$

Winnow can be used to select tuples from more than one relation
- Apply winnow to the result of queries defined over more than one relation

G. Koutrika, E. Pitoura and K. Stefanidis
The Winnow Operator: Properties

- If $\succ_P$ is a total order, $w_P(r)$ includes just one tuple.
Employ Preference Operators: Definition

The Winnow Operator: Properties

- If $\succ_p$ is a total order, $w_p(r)$ includes just one tuple.

- If $\succ_p$ is a weak order, tuples in $\text{win}_p(r)$ are tuples of the top equivalence class of $r$ defined by $\succ$. 

G. Koutrika, E. Pitoura and K. Stefanidis
Employ Preference Operators: Definition

The Winnow Operator: Properties

- If $\succ_\mathcal{P}$ is a total order, $w_\mathcal{P}(r)$ includes just one tuple.

- If $\succ_\mathcal{P}$ is a weak order, tuples in $\text{win}_\mathcal{P}(r)$ are tuples of the top equivalence class of $r$ defined by $\succ$.

- If $\succ_\mathcal{P}$ is a strict partial order, $w_\mathcal{P}(r)$ is non-empty (for every finite, non-empty instance $r$ of $\mathcal{R}$).

G. Koutrika, E. Pitoura and K. Stefanidis
Employ Preference Operators: Definition

The Winnow Operator: Properties

- If $\succ_p$ is a **total** order, $w_p(r)$ includes just one tuple.

- If $\succ_p$ is a **weak** order, tuples in $\text{win}_p(r)$ are tuples of the top equivalence class of $r$ defined by $\sim$.

- If $\succ_p$ is a **strict partial** order, $w_p(r)$ is non-empty (for every finite, non-empty instance $r$ of $R$).

- For any two tuples $t_i$ and $t_j$ of $r$ of $w_p(r)$, it holds that $t_i \succ t_j$
  - $t_i$ and $t_j$ are indifferent

[Chomicki 2003]
The **skyline operator**: Pick the tuples of \( r \) that are not dominated by any other tuple in \( r \)

- A tuple dominates another tuple if:
  - It is as good or better w.r.t. a set of preferences
  - It is better in at least one preference

**Is there any relation with pareto composition?**

[Borzsonyi et al. 2001]: Skylines in multidimensional Euclidean spaces

- The dominance relationship is \( > \) or \( < \)
- Attributes are partitioned into DIFF, MAX and MIN
- Only tuples with identical values on all DIFF attributes are comparable
  - Among those, MAX attribute values are maximized and MIN values are minimized

G. Koutrika, E. Pitoura and K. Stefanidis
k-dominant skyline: $t_i$ k-dominates $t_j$ if there are $k$ dimensions, or preferences, in which $t_i$ is better than or equal to $t_j$, and $t_i$ is better in at least one of these $k$ dimensions

[Chan et al. 2006]

k-representative skyline: select $k$ tuples, such that, the number of tuples that are dominated by at least one of these $k$ tuples is maximized

[Lin et al. 2007]

$\varepsilon$-skyline: compute the set of tuples that are not $\varepsilon$-dominated by any other tuple

- Given a set of preferences, $t_i$ $\varepsilon$-dominates $t_j$ if it is as good, better or slightly worse (up to $\varepsilon$) w.r.t. all preferences and better in at least one preference

[Xia et al. 2008]
Winnow and skyline operators select the most preferred tuples

For ranking all input tuples: Apply multiple times the operators

The Iterated Winnow Operator

Given an instance \( r \) of a relational schema \( R \) and a \( P \) over \( R \), the iterated winnow operator, \( \text{win}^i_p(r) \), of level \( i \), \( i > 0 \), is:

\[
\begin{align*}
\text{win}^1_p(r) &= w_p(r) \\
\text{win}^{i+1}_p(r) &= w_p(r - \bigcup_{k=1}^{i} \text{win}^k_p(r))
\end{align*}
\]

[Chomicki 2003]

The iterated winnow operator, called Best operator, is independently defined by [Torlone and Ciaccia 2003]
In following, we focus on:

- Defining preference operators
- Implementing preference operators
- Translating preference operators
The naïve approach: Nested-Loop method
  - Compare each tuple with every other tuple
    o Nested-Loop requires scanning the whole input for each tuple
Employ Preference Operators: Implementation

Within The Query Engine

A more efficient implementation: **Block-Nested-Loop method**
[Borzsonyi et al. 2001]

**Input**: instance r
**Variables**: window W and table T that are empty

At each iteration:
- All tuples in r are read
- When a tuple t is read, t is compared with all tuples in W
  1. If t is dominated by a tuple in W, then **t is discarded**
  2. If t dominates one or more of the tuples in W, these tuples are discarded and **t is inserted into W**
  3. If t is indifferent with all tuples in W
     - If there is room in W, **t is inserted into W**
     - Otherwise, **t is stored in T**

At the end of each iteration:
- All tuples added to W when T was empty are output
- The next iteration uses T as input
Employ Preference Operators: Implementation

**Winnow for Weak Orders** [Chomicki 2007]
- **Advantage**: All tuples in the winnow belong to a single equivalence class

An input tuple $t$:
- is dominated by all tuples in $W$, in which case $t$ is discarded
- dominates all tuples in $W$, in which case the whole $W$ is replaced by $t$
- is indifferent to all tuples in $W$, in which case $t$ is added to $W$

In all cases: A single comparison of $t$ with just one tuple in $W$ suffices

G. Koutrika, E. Pitoura and K. Stefanidis
Employ Preference Operators: Implementation

Within The Query Engine

**Sort-Filter-Skyline** algorithm [Chomicki et al. 2003]
- Add a preprocessing step to BNL that sorts all tuples in $r$
  - If $t_i \succ_p t_j$, then $t_i$ precedes $t_j$ in the produced order

**Basic Idea**
- Produce an order by topologically sorting the preference graph of $r$
- Process the tuples following this order
  - When a tuple is inserted into $W$, it belongs to the winnow, thus it can be output immediately

For SFS to work, $\succ_p$ must be at least a strict partial order

G. Koutrika, E. Pitoura and K. Stefanidis
Employ Preference Operators: Implementation

**Iterated winnow operator implementation**
- Apply one of the previous algorithms (e.g., the NL or SFS) multiple times
  - First, apply on $r$ to produce $\text{win}_1^p(r)$
  - Then, apply on $(r - \bigcup_{k=1}^{i} \text{win}_k^p(r))$ to produce $\text{win}_{i+1}^p(r)$

**Evaluating Best Operator** algorithm [Torlone and Ciaccia 2003]
- **BNL variation**
  - Compute $\text{win}_{i+1}^p(r)$ from those tuples that were found to be directly dominated by a tuple in $\text{win}_i^p(r)$

G. Koutrika, E. Pitoura and K. Stefanidis
In following, we focus on:

- Defining preference operators
- Implementing preference operators

  **Translating preference operators**
Is the only solution to implement preference operators?
- **Translate operators** into existing relational algebra operators

[Kießling 2002] defines preference queries with two new relational operators:

1. **Preference selection operator**: corresponds to the winnow operator \( w_p(r) \)
2. **Grouped preference selection operator**: apply preference selection within groups

Given an attribute set \( B \):

- Tuples are partitioned into groups with same values in \( B \)
- The grouped preference selection operator selects the dominating tuples in each group
Preference queries expressed using operators can be translated into standard SQL queries

**Preference SQL**: Extent SQL with the preference constructors of [Kießling 2002]

[Kießling and Kostler 2002]

**Example:**
SELECT * FROM movies PREFERING duration BETWEEN [170, 200]
- Return movies with duration in [170, 200]
- If such movies do not exist, return movies with duration closer to the interval limits

G. Koutrika, E. Pitoura and K. Stefanidis
A taxonomy of approaches employing preference operators

<table>
<thead>
<tr>
<th>Query Model</th>
<th>Implementation Level</th>
<th>Evaluation Techniques</th>
<th>Operator Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best Answers</td>
<td>Evaluation Techniques</td>
<td>winnow, skyline</td>
<td>preference selection, grouped preference selection</td>
</tr>
<tr>
<td>Ranking</td>
<td></td>
<td>iterated winnow</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Chomicki 2003; Torlone and Ciaccia 2003; Georgiadis et al. 2008; Drosou et al. 2009]</td>
<td>--</td>
</tr>
</tbody>
</table>
Numerous evaluation methods for preference queries
- Only a few are implemented within the core of a database system

**FlexPref: A framework for extensible preference evaluation in database systems**

Integration with FlexPref: register the functions that implement a preference method
- Once integrated, the preference method “lives” at the core of the database

[Levandoski et al. 2010]
Preferential query processing methods:

- Expand regular database queries with preferences
- Pre-compute rankings of database tuples based on preferences
- Top-k processing
Perform some pre-processing offline to make online processing of queries fast

How?
- Employ preferences to construct offline representative rankings
- At query time, select the relevant rankings and use them to report results

We organize existing approaches into:
- Context-based approaches
- Context-free approaches

G. Koutrika, E. Pitoura and K. Stefanidis
Pre-compute Rankings: Context-based Approaches

Pre-compute representative rankings of database tuples based on contextual preferences

But how the representative rankings are constructed?

G. Koutrika, E. Pitoura and K. Stefanidis
Pre-compute Rankings: Context-based Approaches

[Agrawal et al. 2006]
- Construct a ranking for each set of preferences with the same context
- Maintain only a set of representative rankings

How to select the representative rankings?
- **Greedy Algorithm**
  - Begin from all rankings
  - Remove at each step the ranking that is the most similar to the remaining ones
- **Furthest Algorithm**
  - Select randomly a ranking
  - At each step, pick the ranking which is furthest from the already selected ones
  - Continue up to collect the desirable number of representative rankings

The distance between two rankings may be computed using either the [Spearman footrule](https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient) or the [Kendall tau distance](https://en.wikipedia.org/wiki/Kendall_rank_correlation_coefficient).
Pre-compute Rankings: Context-based Approaches

[Stefanidis and Pitoura 2008]
- Create groups of similar preferences
- Construct a ranking for each group

Which preferences are similar?

- **Contextual clustering**
  - Consider as similar the preferences with similar context
- **Predicate clustering**
  - Consider as similar the preferences with similar predicates and scores
Pre-compute Rankings: Context-free Approaches

Such approaches employ **materialized preference views**
- Relational views ordered according to a preference, or scoring, function

**Main goal:** Locate the $k$ results that maximize (or minimize) a combining preference function in a pipelined manner
  
e.g., [Hristidis and Papakonstantinou 2004]
A taxonomy of pre-computing rankings approaches

<table>
<thead>
<tr>
<th>Formulation</th>
<th>Qualitative</th>
<th>Quantitative</th>
<th>Context-based</th>
<th>Context-free</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Agrawal et al. 2006]</td>
<td>[Stefanidis and Pitoura 2008; You and Hwang 2008]</td>
<td></td>
<td>[Hristidis and Papakonstantinou 2004; Das et al. 2006; Yi et al. 2003]</td>
</tr>
</tbody>
</table>
Preferential query processing methods:

- Expand regular database queries with preferences
- Pre-compute rankings of database tuples based on preferences
- Top-k processing
Top-k Processing

Top-k query: provide the k most important results

Basic Idea

- Assign scores to all tuples based on a scoring function or an aggregation of a set of functions
- Report the k tuples with the highest scores
Top-k Processing

Methods for compounding a set of rankings to an aggregate one:

- **FA Algorithm**
  - Do sorted access to each ranking until there is a set of k tuples, such that each of these tuples has been seen in each of the rankings
  - For each tuple that has been seen, do random accesses to retrieve the missing scores
  - Compute the aggregate score of each tuple that has been seen
  - Rank the tuples based on their aggregate scores and select the top-k ones

  [Fagin et al. 2001]

- **TA Algorithm**

  Sorted access enables tuple retrieval in a descending order of their scores
  Random access enables retrieving the score of a specific tuple in one access

G. Koutrika, E. Pitoura and K. Stefanidis
Example: FA Algorithm

\[
S_1 = < A \ 0.9, \ C \ 0.8, \ E \ 0.7, \ B \ 0.5, \ F \ 0.5, \ G \ 0.5, \ H \ 0.5 >
\]
\[
S_2 = < B \ 1.0, \ E \ 0.8, \ F \ 0.7, \ A \ 0.7, \ C \ 0.5, \ H \ 0.5, \ G \ 0.5 >
\]
\[
S_3 = < A \ 0.8, \ C \ 0.8, \ E \ 0.7, \ B \ 0.5, \ F \ 0.5, \ G \ 0.5, \ H \ 0.5 >
\]

Which is the top-1 item?

Compute aggregate scores for A, B, C, E, F

Note:

FA is correct when the aggregate tuple scores are obtained by combining their individual scores using a monotone function

G. Koutrika, E. Pitoura and K. Stefanidis
Methods for compounding a set of rankings to an aggregate one:

- **FA Algorithm**
- **TA Algorithm**
  - Do sorted access to each ranking: For each tuple seen, do random accesses to retrieve their missing scores
  - Compute the aggregate score of each tuple that has been seen, rank the tuples based on their aggregate scores and select the top-k ones
  - Stop to do sorted accesses when the aggregate scores of the k tuples are at least equal to a threshold value
    - Threshold value: the aggregate score of the scores of the last tuples seen in each ranking

[Fagin et al. 2001; Nepal and Ramakrishna 1999; Guntzer et al. 2000]

Sorted access enables tuple retrieval in a descending order of their scores
Random access enables retrieving the score of a specific tuple in one access
Example: TA Algorithm

\[ S_1 = < A 0.9, C 0.8, E 0.7, B 0.5, F 0.5, G 0.5, H 0.5 > \]
\[ S_2 = < B 1.0, E 0.8, F 0.7, \sqrt{0.7}, C 0.5, H 0.5, G 0.5 > \]
\[ S_3 = < A 0.8, C 0.8, E 0.7, B 0.5, F 0.5, G 0.5, H 0.5 > \]

Which is the top-1 item?

Step1:
\[ \text{score}(A) = 0.9 + 0.7 + 0.8 = 2.4 \]
\[ \text{score}(B) = 0.5 + 1.0 + 0.5 = 2.0 \]
\[ \text{threshold}_\text{value} = 0.9 + 1.0 + 0.8 = 2.7 \]
Continue since \( 2.7 > 2.4 \)

Step2:
\[ \text{score}(C) = 0.8 + 0.5 + 0.8 = 2.1 \]
\[ \text{score}(E) = 0.7 + 0.8 + 0.7 = 2.2 \]
\[ \text{threshold}_\text{value} = 0.8 + 0.8 + 0.8 = 2.4 \]
Stop since \( \text{score}(A) = \text{threshold}_\text{value} \)

The stopping condition of TA occurs at least as early as the stopping condition of FA.
Above: Aggregate rankings that contain the same set of tuples
- The produced ranking consists of the same tuple set

**Top-k Joined Tuples**
Report the k joined tuples with the largest interest scores
- Tuples of different rankings are joined w.r.t. specific join conditions
- Each tuple has a score computed from the scores of the participating tuples

[Natsev et al. 2001; Ilyas et al. 2004]

**Top-k Groups of Tuples**
Report the k groups of tuples with the largest interest scores
- Scores are computed using a group aggregation function

[Li et al. 2006]
A taxonomy of top-k query processing techniques

<table>
<thead>
<tr>
<th>Query Model</th>
<th>Implementation Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Application level</td>
</tr>
<tr>
<td>Top-k tuples</td>
<td>[Fagin et al. 2001; Nepal and Ramakrishna 1999; Guntzer et al. 2000]</td>
</tr>
<tr>
<td>Top-k joined tuples</td>
<td>[Natsev et al. 2001]</td>
</tr>
<tr>
<td>Top-k groups of tuples</td>
<td>--</td>
</tr>
</tbody>
</table>
Tutorial Overview

- Preference Representation
- Preference Composition
- Preferential Query Processing
- Preference Learning
Model Learnt

- Pairwise orderings (i.e., qualitative preferences)
- Utility function (i.e., quantitative preferences)
Input

- Positive examples
- Explicit feedback

- Negative examples
- Implicit feedback
Preference Learning

- Method
  - Association rule mining
  - Clustering
  - Classification
Holland et al. [2003]

**Input**: User logs, no explicit ranking information
x is preferred over y, if and only if, freq(x) > freq(y).

**Model learnt**:
Preferences between values of individual attributes are used to infer positive and negative preferences, numerical preferences and complex preferences [Kießling 2002].

**An important assumption**, for learning negative preferences or dislikes, is the close world assumption indicating that a user knows all possible values of an attribute.
Model Learnt: a preference relation in the form of partial order

Input: set of superior and inferior examples

Output: a strict partial order, such that, every item is dominated by at least one item in the set of superior examples and it is not dominated by any other item in the set of inferior examples.
[Cohen et al. 1999]

**Input**: Feedback that an item should be ranked higher than another.

**Model**: $\text{Pref} (i_1; i_2)$, $\text{Pref} : I \times I \rightarrow [0; 1]$, returns a value indicating which item is ranked higher.

**Learning**: At each round, items are ranked with respect to $\text{Pref}$. Then, the learner receives feedback from the environment. Given that $\text{Pref}$ is a weighted linear combination of $n$ primitive functions, at each round the weights are updated with respect to the user feedback and loss, where loss is the normalized sum of disagreements between function and feedback.
Conclusions

- Preference Representation
- Preference Composition
- Preferential Query Processing
- Preference Learning
Conclusions

- **Preference Representation**
  - Existing methods are divided into qualitative and quantitative

- **Preference Composition**
  - Existing methods tackle specific aspects of the problem

- A holistic preference representation approach is missing

- **Preferential Query Processing**
  - Complete understanding of user preferences is missing – (psychology?)

- **Preference Learning**
  - New types of preferences (membership, uncertain, ...)

G. Koutrakia, E. Pitoura and K. Stefanidis
Conclusions

- Preference Representation

- Preference Composition
  - Existing works follow a uniform approach to representation and composition
  - Qualitative composition applies to preferences represented in either way
  - Most approaches deal with tuple-to-tuple preference composition
  - There are combinations that have not been touched at all
  - Can composition be used as a means to resolve conflicts?

- Preferential Query Processing

- Preference Learning

G. Koutrika, E. Pitoura and K. Stefanidis
Conclusions

- Preference Representation

- Preference Composition

- Preferential Query Processing
  - An approach for matching both internal and external preference context to query context is missing
  - Approaches that deal with instance and semantic applicability are missing
  - Embed preferences in the database
  - Query + Preferences = ?

- Preference Learning

G. Koutrika, E. Pitoura and K. Stefanidis
Conclusions

- Preference Representation
- Preference Composition
- Preferential Query Processing
- Preference Learning
  - Learning preferences following db-specific models is highly unexplored
  - Learning context-aware and privacy-aware preferences (too)
  - Sufficient information for deriving user preferences is missing
Future Directions

- Hybrid preference models
  Combining qualitative and quantitative aspects

- Group preferences
  Merging individual preferences  [Amer-Yahia et al. 2009]

- Social preferences
  User preferences over the social graph

G. Koutrika, E. Pitoura and K. Stefanidis
Future Directions

- Leveraging the wisdom of crowds
  - Learning preferences

- Preference-aware query engine
  - Making preferences first-class citizens
  - Holistic optimizer

G. Koutrika, E. Pitoura and K. Stefanidis
The End


