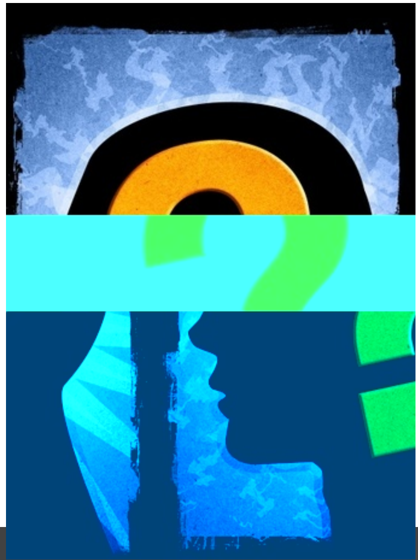
Preferences in Databases



Representation Composition & Application

Georgia Koutrika[1], **Evaggelia Pitoura**[2], **Kostas Stefanidis**[2]

[1] Stanford University, [2] University of Ioannina

Introduction

- **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**

Introduction

- 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

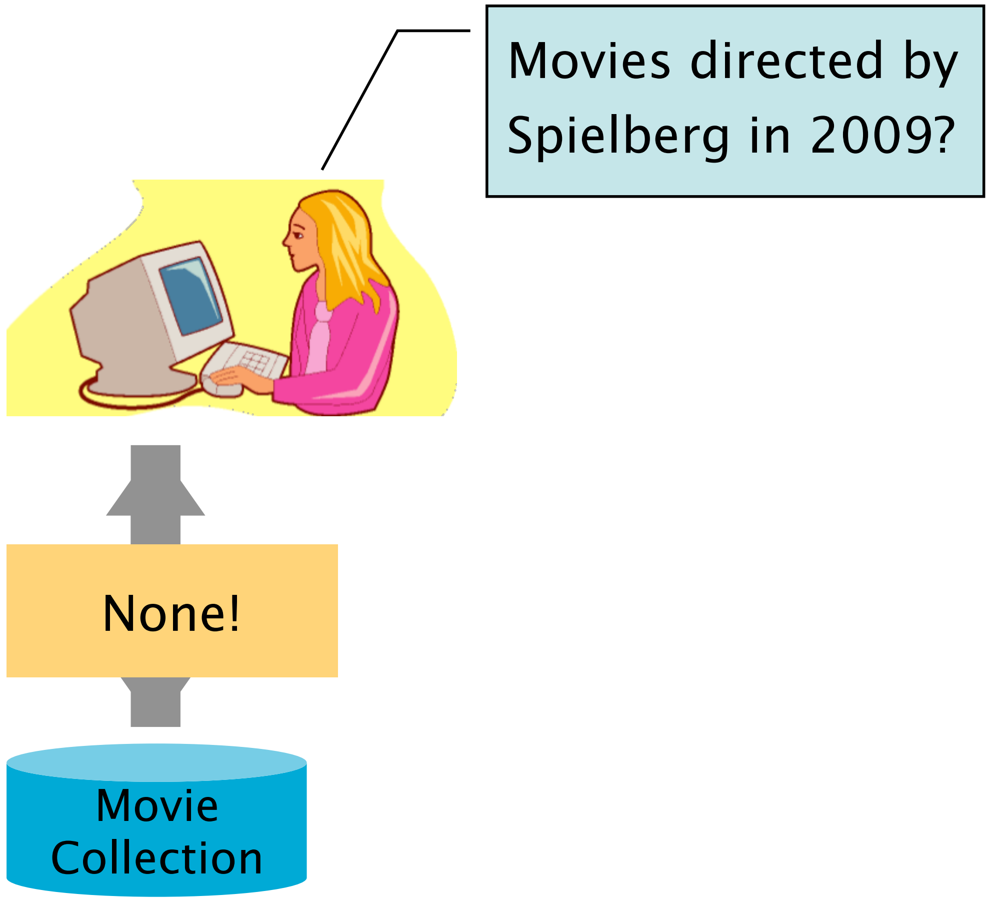
- Databases on the Web: 7,500TB (19TB is the surface Web)!
  - National Climatic Data Center (NOAA)
  - NASA EOSDIS
  - Alexandria Digital Library
  - JSTOR Project Limited
  - US Census
  - Amazon.com
  - …

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!)
  - Unknown schema
  - Unknown contents
- On the Web: Too much information
  - Information Overload
  - User diversity

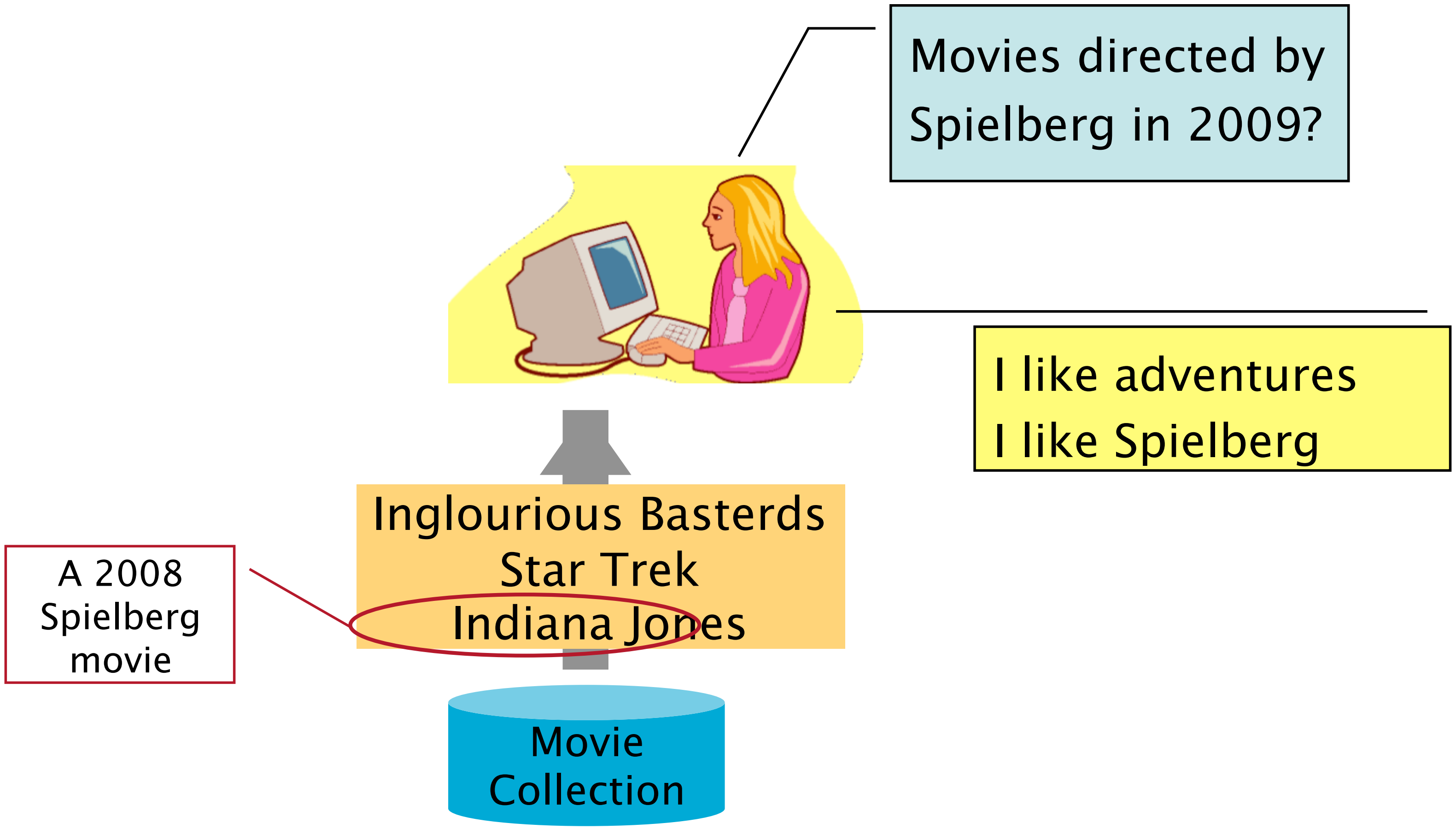
Why Preferences in Databases?

- Incorporating preferences can help return non-empty answers



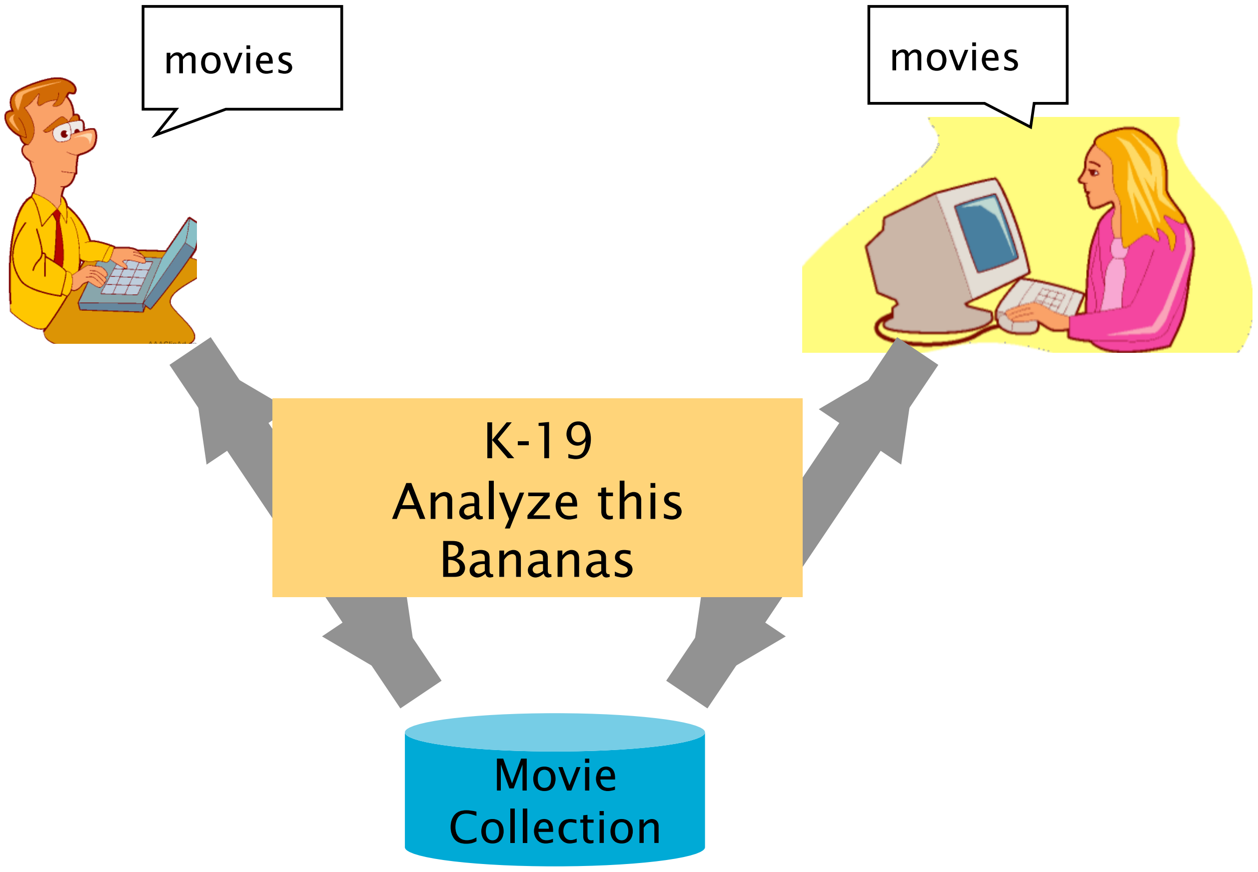
Why Preferences in Databases?

- Incorporating preferences can help return non-empty answers



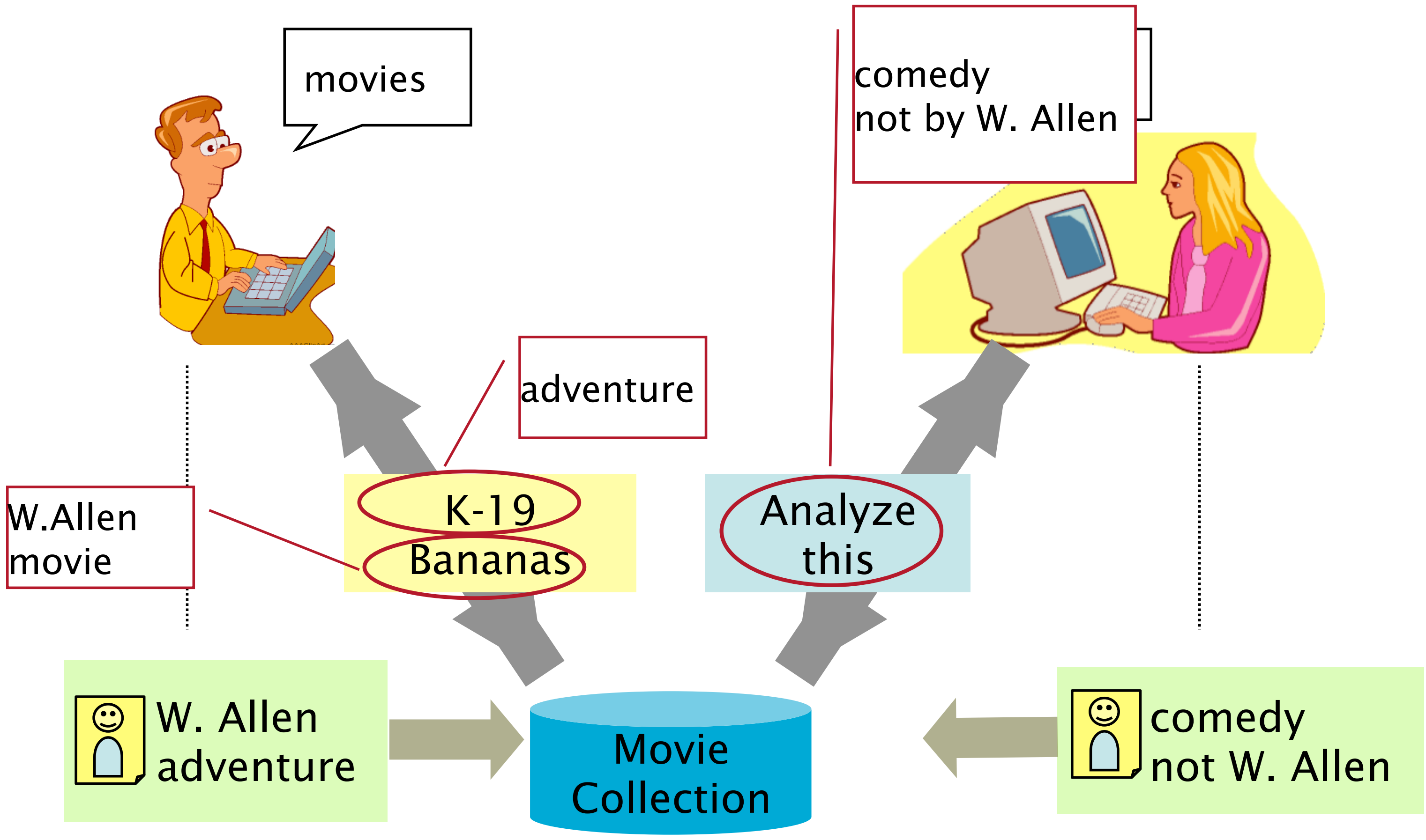
Why Preferences in Databases?

- Incorporating preferences can help return focused answers



Why Preferences in Databases?

- Incorporating preferences can help return focused answers



Tutorial Overview

Preference Representation

Preference Composition

Preferential Query Processing

Preference Learning

Example

movie

<u>mid</u>	title	year	director	genre	language	duration

play

<u>mid</u>	<u>aid</u>

actor

<u>aid</u>	name	date_of_birth

movie

	<u>mid</u>	*title*	*year*	*director*	*genre*	*language*	*duration*
t_1	m_1	Casablanca	1942	Curtiz	drama	english	102
t_2	m_2	Psycho	1960	Hitchcock	horror	english	109
t_3	m_3	Schindler's List	1993	Spielberg	drama	english	109

Tutorial Overview

Preference Representation

- Formulation
- Granularity
- Context
- Aspects

sing

Preference Learning

Formulation

- Qualitative approaches
- Quantitative approaches

Binary preference relations

Preferences between tuples in the answer to a query are specified directly using binary preference relations

[Chomicki 2003; Kiessling 2002]

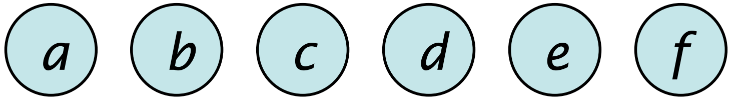
Given a relation R:
A preference relation B is a subset of $R \times R$

a B b between tuples a and b of R => a is preferred over b

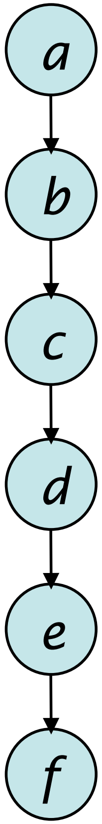
Properties of binary relations

Reflexive:	a B a, $\forall$ a in R
Irreflexive	$\neg(a$ B $a)$, $\forall$ a in R
Symmetric	a B $b \Rightarrow b$ B a, $\forall$ a, b in R
Transitive	$(a$ B $b) \wedge (b$ B $c) \Rightarrow (a$ B $c)$, $\forall$ a, b, c in R
Asymmetric	$(a$ B $b) \Rightarrow \neg(b$ B $a)$, $\forall$ a, b in R
Antisymmetric	$(a$ B $b) \wedge (b$ B $a) \Rightarrow (a = b)$, $\forall$ a, b in R
Negative transitive	$\neg(a$ B $b) \wedge \neg(b$ B $c) \Rightarrow \neg(a$ B $c)$, $\forall$ a, b, c in R
Connective	$(a$ B $b) \vee (b$ B $a) \vee (a = b)$, $\forall$ a, b in R

Types of binary relations



Tuples in R



Connective

Irreflexive

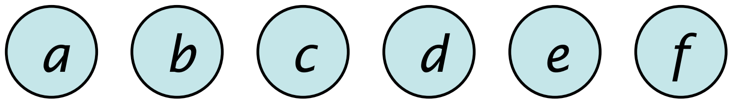
Asymmetric

Transitive

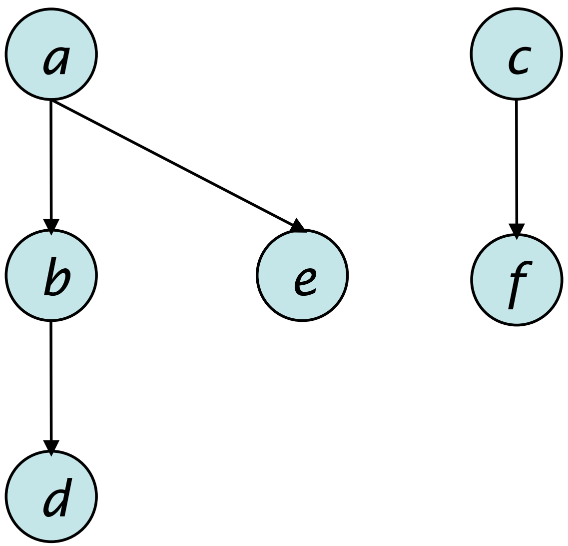
Total Order

Formulation: Qualitative Approaches

Types of binary relations



Tuples in *R*



Irreflexive

Asymmetric

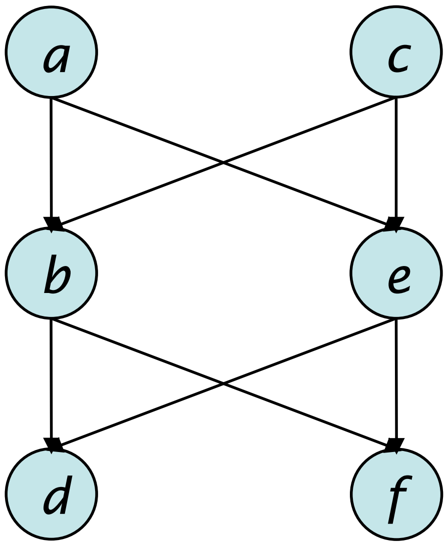
Transitive

Strict Partial Order

Formulation: Qualitative Approaches

Types of binary relations

a *b* *c* *d* *e* *f* Tuples in *R*



Negative transitive

Irreflexive

Asymmetric

Transitive

Weak Order

Logical formulas

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

[Chomicki 2003; Georgiadis et al. 2008]

movie

	mid	*title*	*year*	*director*	*genre*	*language*	*duration*
t_1	m_1	Casablanca	1942	Curtiz	drama	english	102
t_2	m_2	Psycho	1960	Hitchcock	horror	english	109
t_3	m_3	Schindler's List	1993	Spielberg	drama	english	109

$t_i \succ_{\mathbf{PF}} t_j \Leftrightarrow t_i[\text{genre}] = t_j[\text{genre}] \wedge t_i[\text{duration}] < t_j[\text{duration}]$

Casablanca is preferred over Schindler's list

Preference Constructors

A formal language for formulating preference relations using constructors

[Kiessling 2002]

HIGHEST(A) $\{t_i \succ_{P\_new} t_j \text{ iff } t_i > t_j\};$

AROUND(A, z) $\{t_i \succ_{P\_new} t_j \text{ iff } abs(t_i - z) < abs(t_j - z)\};$

Preference Constructors

A formal language for formulating preference relations using constructors

[Kiessling 2002]

movie

	mid	*title*	*year*	*director*	*genre*	*language*	*duration*
t_1	m_1	Casablanca	1942	Curtiz	drama	english	102
t_2	m_2	Psycho	1960	Hitchcock	horror	english	109
t_3	m_3	Schindler's List	1993	Spielberg	drama	english	109

POS(genre, {horror})

NEG(year, {1960})

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

Preference Functions

Preferences for tuples are expressed using functions that assign a score

[Agrawal et al. 2000]

$t_i \succ_P t_j$ for a preference function $f_P \Leftrightarrow f_P(t_i) > f_P(t_j)$

(with exceptions [Guo et al. 2008])

Formulation: Quantitative Approaches

Preference Functions

Example

movie

	mid	*title*	*year*	*director*	*genre*	*language*	*duration*	
t_1	m_1	Casablanca	1942	Curtiz	drama	english	102	→ 0.102
t_2	m_2	Psycho	1960	Hitchcock	horror	english	109	→ 0.109
t_3	m_3	Schindler's List	1993	Spielberg	drama	english	109	→ 0.109

$f_P(t_i) = 0.001 \times t_i[\textit{duration}]$

Formulation: Quantitative Approaches

Degrees of Interest

Preferences for tuples are expressed by specifying constraints for the tuples and assigning scores in these constraints

[Koutrika et al. 2004; Stefanidis et al. 2007]

Preference *(Condition, Score)*:

Condition: $A_1 \ \theta_1 \ v_1 \wedge A_2 \ \theta_2 \ v_2 \wedge \ldots \wedge A_n \ \theta_n \ v_n$

Score belongs to a predefined numerical domain

movie.genre = 'drama', 0.9

movie.year > 1990, 0.8

Formulation

Incompleteness

Represents a gap in our knowledge

Indifference

$t_i \sim t_j \Leftrightarrow \neg(t_i \succ_{PR} t_j) \wedge \neg(t_j \succ_{PR} t_i)$ qualitative

$\Leftrightarrow f_P(t_i) = f_P(t_j)$ quantitative

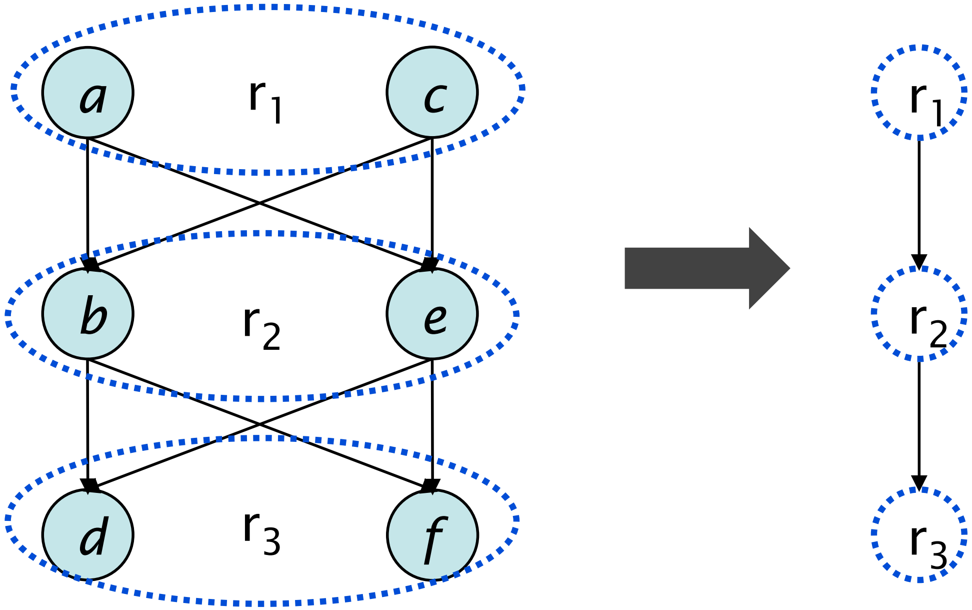
Incomparability

Tuples that cannot be compared in some fundamental way

Equivalence classes

If a preference relation $\succ_{PR}$ is weak order, then indifference is an equivalence class

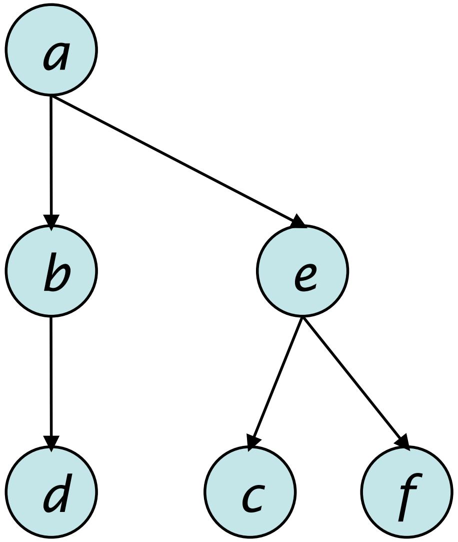
A binary relation is an equivalence class if it is reflexive, symmetric and transitive



Formulation

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

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

movie

	mid	*title*	*year*	*director*	*genre*	*language*	*duration*
t_1	m_1	Casablanca	1942	Curtiz	drama	english	102
t_2	m_2	Psycho	1960	Hitchcock	horror	english	109
t_3	m_3	Schindler's List	1993	Spielberg	drama	english	109

t_3 is preferred over t_1 and t_2 is incomparable

Preference Representation

Preference representation dimensions

- Formulation
- Granularity
- Context
- Aspects

Tuple Preferences

Preferences expressed directly for tuples and their values

movie.genre = ‘drama’, 0.9

movie.mid = cast.mid and
cast.aid = actor.aid and
actor.name = ‘J. Roberts’, 0.7

[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

Granularity

Attribute Preferences

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

$P_{director} \succ P_{genre}$ [Georgiadis et al 2008]

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

{title, genre, language}, 1

{year, director, duration}, 0.3

[Miele at al 2009]

Granularity

Relationship Preferences

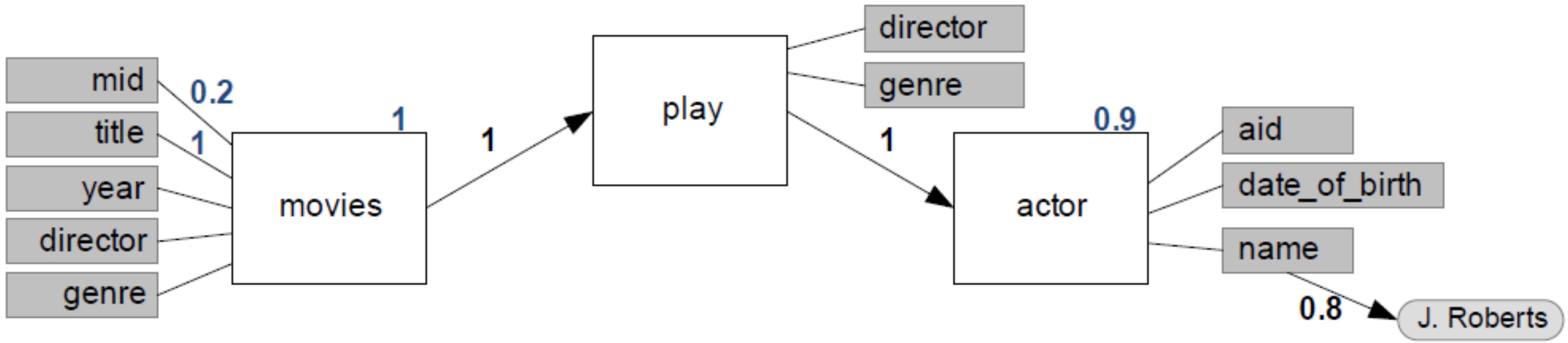
They are expressed on relationships between two types of entities or two particular entities

(movie.mid = play.mid, 1) [Koutrika, Ioannidis 2004]

A director has directed many movies

Julia Roberts has acted in Ocean's Eleven

One more example…



Preference Representation

Preference representation dimensions

- Formulation
- Granularity
- Context
- Aspects

Context

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 (in preferences)

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

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

movie

	mid	*title*	*year*	*director*	*genre*	*language*	*duration*
t_1	m_1	Casablanca	1942	Curtiz	drama	english	102
t_2	m_2	Psycho	1960	Hitchcock	horror	english	109
t_3	m_3	Schindler's List	1993	Spielberg	drama	english	109

Internal Contextual Preferences

Given a relation with attributes A_1, … A_d, an internal context is:
$\wedge_{j \in L}(A_j = v_j)$, $L \subseteq \{A_1, \ldots A_d\}$

[Agrawal et al 2006]

Example

movie

	mid	*title*	*year*	*director*	*genre*	*language*	*duration*
t_1	m_1	Casablanca	1942	Curtiz	drama	english	102
t_2	m_2	Psycho	1960	Hitchcock	horror	english	109
t_3	m_3	Schindler's List	1993	Spielberg	drama	english	109

{director = 'Spielberg' ≻ director = 'Curtiz' | genre = 'drama'}

t_3 is preferred over t_1

Internal Contextual Preferences

Example [Chomicki 2003]

movie

	mid	*title*	*year*	*director*	*genre*	*language*	*duration*
t_1	m_1	Casablanca	1942	Curtiz	drama	english	102
t_2	m_2	Psycho	1960	Hitchcock	horror	english	109
t_3	m_3	Schindler's List	1993	Spielberg	drama	english	109

$$
\begin{aligned}
t_i \succ_{PF} t_j \Leftrightarrow\ & (t_i[\text{genre}] = t_j[\text{genre}] \wedge t_i[\text{genre}] = \text{‘drama’} \wedge \\
& \quad t_i[\text{director}] = \text{‘Spielberg’} \wedge t_j[\text{director}] = \text{‘Curtiz’}) \vee \\
& (t_i[\text{genre}] = t_j[\text{genre}] \wedge t_i[\text{genre}] = \text{‘thriller’} \wedge \\
& \quad t_j[\text{director}] = \text{‘Spielberg’} \wedge t_i[\text{director}] = \text{‘Curtiz’})
\end{aligned}
$$

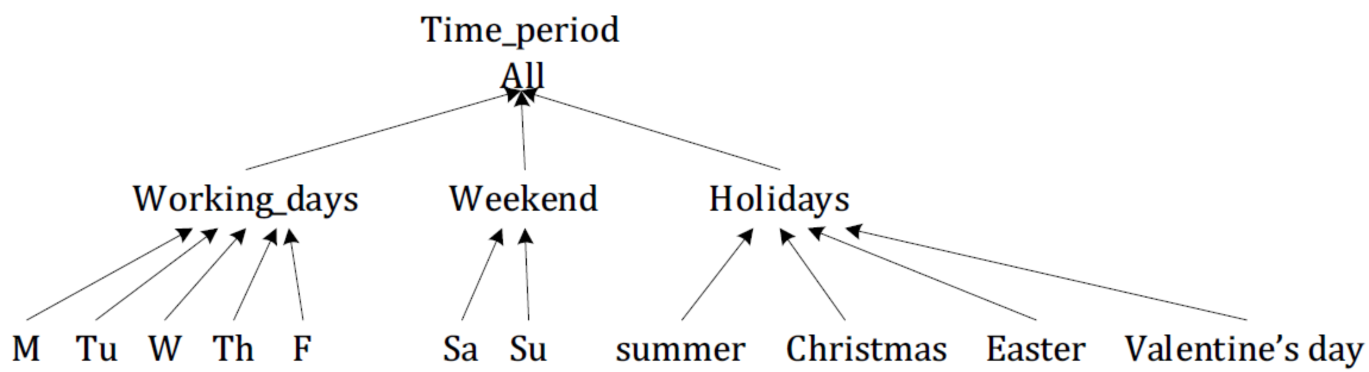
External Contextual Preferences

Given a set of contextual parameters C_1, … C_n, an external context is: a n-tuple (c_1, … c_n), where $c_i \in C_i$

[Stefanidis et al. 2007; Miele et al. 2009]

Example

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



Preference Representation

Preference representation dimensions

- Formulation
- Granularity
- Context
- Aspects

Aspects

Intensity

It shows the degree of desire expressed in a preference

- Weak preferences

  movie.genre = ‘cartoons’, 0.4

- Strong preferences

  movie.genre = ‘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

Preference Representation: Summary

Preference representation approaches w.r.t. preference formulation, granularity and context

	Formulation		Granularity				Context		
	Qualitative	Quantitative	Tuple	Relation	Attribute	Relationship	Context-free	Internal	External
[Agrawal and Wimmers 2000]		✓	✓				✓		
[Agrawal et al. 2006]	✓		✓					✓	
[Bunningen et al. 2006; 2007]		✓	✓						✓
[Chomicki 2002; 2003]	✓		✓				✓	✓	
[Georgiadis et al. 2008]	✓		✓		✓		✓		
[Holland and Kiessling 2004]	✓		✓						✓
[Kiessling 2002]	✓	✓	✓				✓		
[Koutrika and Ioannidis 2004; 2005]		✓	✓			✓	✓		
[Miele et al. 2009]		✓	✓		✓				✓
[Stefanidis et al. 2006; 2007]		✓	✓				✓		✓
[Zhang and Chomicki 2008]	✓		sets				✓		

Preference Representation: Summary

Preference representation approaches w.r.t preference aspects (T=tuple, C=relation, A=attribute, R=relationship)

	Aspects												
	Intensity		Necessity		Feeling			Complexity		Attitude		Elasticity	
	Strong	Weak	Hard	Soft	Positive	Negative	Indifferent	Simple	Compound	Presence	Absence	Exact	Elastic
[Agrawal and Wimmers 2000]	T	T	-	T	T	-	T	T	T	T	T	T	T
[Agrawal et al. 2006]	T	T	-	T	T	-	-	T	T	T	-	T	-
[Bunningen et al. 2006; 2007]	T	T	-	T	T	-	-	T	T	T	T	T	-
[Chomicki 2002; 2003]	T	T	-	T	T	-	T	T	T	T	T	T	-
[Georgiadis et al. 2008]	TA	TA	A	T	TA	-	TA	T	T	TA	-	TA	-
[Holland and Kiessling 2004]	T	T	-	T	T	T	-	T	T	T	T	T	T
[Kiessling 2002]	T	T	-	T	T	T	-	T	T	T	T	T	T
[Koutrika and Ioannidis 2004; 2005]	T	T	TR	TR	T	T	T	TR	TR	T	T	T	T
[Miele et al. 2009]	TA	TA	A	TA	TA	-	-	TA	TA	TA	T	TA	-
[Stefanidis et al. 2006; 2007]	T	T	-	T	T	-	-	T	T	T	T	T	
[Zhang and Chomicki 2008]	T	T	-	T	T	-	T	T	T	T	T	T	-

Tutorial Overview

- Preference Representation
- Preference Composition
  - Qualitative Composition
  - Quantitative Composition
  - Heterogeneous Composition
- Preference Learning

Qualitative Composition

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

Qualitative Composition

Prioritized Composition

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 of R, $t_i \succ_{P_x \& P_y} t_j$, iff:

  $(t_i \succ_{P_x} t_j) \vee (t_i \sim_{P_x} t_j \wedge t_i \succ_{P_y} t_j)$

Qualitative Composition

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[genre] = 'drama' $\wedge$ t_j[genre] = 'horror') $\vee$

(t_i[genre] $\neq$ 'drama' $\wedge$ t_i[duration] > t_j[duration]) $\vee$

(t_j[genre] $\neq$ 'horror' $\wedge$ t_i[duration] > t_j[duration])

t3 is preferred over t1

t1 is preferred over t2

movie

	mid	*title*	*year*	*director*	*genre*	*language*	*duration*
t_1	m_1	Casablanca	1942	Curtiz	drama	english	102
t_2	m_2	Psycho	1960	Hitchcock	horror	english	109
t_3	m_3	Schindler's List	1993	Spielberg	drama	english	109

Qualitative Composition

Prioritized composition over different relational schemas

Lexicographical Composition

For P_x, P_y defined over R, R' with attribute domains dom(A), dom(A')

- The lexicographical preference composition relation $\succ_{Px\&Py}$ defined over R×R', is a subset of dom(A)×dom(A'), such that,

  $(t_i, t'_i) \succ_{Px\&Py} (t_j, t'_j)$, iff: $(t_i \succ_{Px} t_j) \vee (t_i \sim_{Px} t_j \wedge t'_i \succ_{Py} t'_j)$

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

Qualitative Composition

Pareto Composition

For P_x, P_y defined over R

- The pareto preference composition relation $\succ_{Px \otimes Py}$ is defined over R, such that, $\forall t_i$, t_j of R, $t_i \succ_{Px \otimes Py} t_j$, iff:

  $(t_i \succ_{Px} t_j \wedge \neg(t_j \succ_{Py} t_i)) \vee (t_i \succ_{Py} t_j \wedge \neg(t_j \succ_{Px} t_i))$

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

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: ($t_i$[genre] = 'drama' ∧ t_j[genre] = 'horror' ∧ t_i[duration] ≥ t_j[duration]) ∨

(t_i[duration] > t_j[duration] ∧ t_j[genre] ≠ 'drama') ∨

(t_i[duration] > t_j[duration] ∧ t_j[genre] = 'drama' ∧ t_i[genre] ≠ 'horror')

t3 is preferred over t1

t1, t2 are incomparable

movie

	mid	*title*	*year*	*director*	*genre*	*language*	*duration*
t_1	m_1	Casablanca	1942	Curtiz	drama	english	102
t_2	m_2	Psycho	1960	Hitchcock	horror	english	109
t_3	m_3	Schindler's List	1993	Spielberg	drama	english	109

Pareto composition over different relational schemas

Multidimensional Pareto Composition

For P_x, P_y defined over R, R' with attribute domains dom(A), dom(A')

- The multidimensional pareto preference relation $\succ_{Px \otimes Py}$ defined over R×R' is a subset of dom(A)×dom(A'), such that,

  $(t_i, t'_i) \succ_{Px \otimes Py} (t_j, t'_j)$, iff: $(t_i \succ_{Px} t_j \wedge \neg(t'_j \succ_{Py} t'_i)) \vee$
  $(t'_i \succ_{Py} t'_j \wedge \neg(t_j \succ_{Px} t_i))$

t_i, t_j are tuples of R and t'_i, t'_j tuples of R'

Qualitative Composition

Motivation: Voting theory [Condorcet 1785]

Pair-wise Comparisons Composition

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

Qualitative Composition

Set-oriented Composition

For P_x, P_y defined over the relational schema R

- The intersection preference relation $\succ_{Px \wedge Py}$ is defined over R, such that, $\forall t_i$, t_j of R, $t_i \succ_{Px \wedge Py} t_j$, iff:

$$t_i \succ_{Px} t_j \wedge t_i \succ_{Py} t_j$$

- The union preference relation $\succ_{Px+Py}$ is defined over R, such that, $\forall t_i$, t_j of R, $t_i \succ_{Px+Py} t_j$, iff:

$$t_i \succ_{Px} t_j \vee t_i \succ_{Py} t_j$$

- The difference preference relation $\succ_{Px-Py}$ is defined over R, such that, $\forall t_i$, t_j of R, $t_i \succ_{Px-Py} t_j$, iff:

$$t_i \succ_{Px} t_j \wedge \neg(t_i \succ_{Py} t_j)$$

Qualitative Composition

Intersection example:

P1: dramas over horrors

P2: long movies over short ones

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

$(t_i[\text{genre}] = \text{‘drama’} \wedge t_j[\text{genre}] = \text{‘horror’}) \wedge (t_i[\text{duration}] > t_j[\text{duration}])$

[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

Preference Composition

Preference composition mechanism categories:

- Qualitative composition
- <u>Quantitative composition</u>
  - Combine preferences expressed as scores over a set of tuples and assign final scores to these tuples
- Heterogeneous composition

Quantitative Composition

Definition

Given:

- Two preferences P_x, P_y over R defined through preference functions f_{Px}, f_{Py}
- A combining function $F : \mathbb{R}\times\mathbb{R}\rightarrow\mathbb{R}$

$\forall t_i, t_j$ in R, $t_i \succ rank_F(P_x,P_y)\ t_j$, iff: $F(f_{Px}(t_i), f_{Py}(t_i)) > F(f_{Px}(t_j), f_{Py}(t_j))$

Quantitative Composition

To assign importance to preferences, weights can be used

Example: P1: $f_{P1}(t_i) = 0.001 \times t_i[\text{duration}]$
P2: $f_{P2}(t_i) = 0.0001 \times t_i[\text{year}]$
$\text{rank}_F(P1, P2)$: $F(f_{P1}(ti), f_{P2}(ti)) = 0.1 \times f_{P1}(t_i) + 0.9 \times f_{P2}(t_i)$

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

movie

	mid	*title*	*year*	*director*	*genre*	*language*	*duration*
t_1	m_1	Casablanca	1942	Curtiz	drama	english	102
t_2	m_2	Psycho	1960	Hitchcock	horror	english	109
t_3	m_3	Schindler's List	1993	Spielberg	drama	english	109

Also: Numerical composition over different relational schemas

Quantitative Composition

Other types of combining functions:

- The min and max functions

Three classes of combining functions:

- Inflationary: the preference in a tuple increases with the number of preferences that satisfy it
- Dominant: the most important preference dominates
- Reserved: the preference in a tuple is between the highest and the lowest degrees of interest among the preferences satisfied

[Koutrika and Ioannidis 2005b]

Preference Overriding

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

[Koutrika and Ioannidis 2010]

Example:

P1: movie: (movie.genre = ‘comedy’, 0.9)

P2: movie: (movie.genre = ‘comedy’ and movie.director = ‘Stiller’, -0.9)

P2 overrides P1 whenever they both apply

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

User Attitude

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

		Attitude		
		Overriding	Dominant	Combinatory
Tuple Ranking	Qualitative	prioritized, lexicographical	--	pareto, multidimensional pareto, pair-wise comparisons, intersection, difference, union
	Quantitative	syntactic overriding	max, min	average, weighted average, …

Heterogeneous Composition

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

Heterogeneous Composition

Mechanisms for composing preferences of different granularity

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

Combine preferences expressed at tuple and relationship level

To do this:

Compose implicit preferences by other composeable ones

P_x and P_y are composeable, iff:

i. P_x is a join preference of the form R_x: (q_x, d_x) connecting R_x to a relation R_y and
ii. P_y is a join or selection preference on R_y, i.e., R_y: (q_y, d_y)

[Koutrika and Ioannidis 2005b]

q_x and q_y are conditions, d_x and d_y are scores, P_x and P_y can be viewed as queries that select tuples from relations R_x, R_y that satisfy q_x, q_y

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)

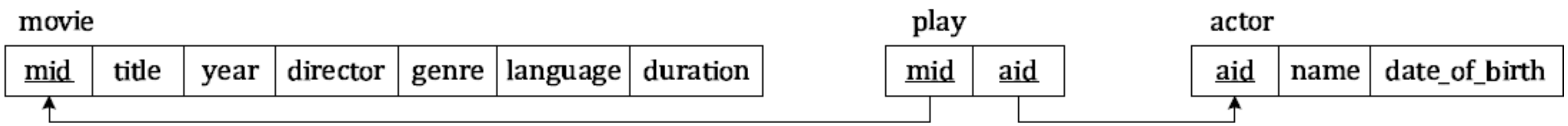
play: (play.aid = actor.aid, 1)

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

t2 is preferred to t1 and t3

t1, t3 are incomparable

movie

	mid	*title*	*year*	*director*	*genre*	*language*	*duration*
t_1	m_1	Casablanca	1942	Curtiz	drama	english	102
t_2	m_2	Psycho	1960	Hitchcock	horror	english	109
t_3	m_3	Schindler's List	1993	Spielberg	drama	english	109

Preference Composition: Summary

Preference composition w.r.t. granularity

	Tuple	Relation	Attribute	Relationship
Tuple	[Agrawal and Wimmers 2000; Agrawal et al. 2006; Bunningen et al. 2006; 2007; Chomicki 2002; 2003; Georgiadis et al. 2008; Holland and Kiessling 2004; Kiessling 2002; Koutrika and Ioannidis 2004; 2005b; Miele et al. 2009; Stefanidis et al. 2006; 2007; Zhang and Chomicki 2008]	--	[Georgiadis et al. 2008]	[Koutrika and Ioannidis 2004; 2005b]
Relation		--	--	--
Attribute			[Georgiadis et al. 2008; Miele et al. 2009]	--
Relationship				[Koutrika and Ioannidis 2004; 2005b]

Preferential Query Processing

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

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

Expand Database Queries

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?

Expand Database Queries

Preference Relatedness

Assume:

rnal

Example:

(**C**, **P**): (Accompanying_people = '**friends**',
genre = '**horror**')

($\mathbf{C_Q}$, **Q**): (Accompanying_people = '**friends**',
SELECT title
FROM movie
WHERE director = 'Hitchcock')

Preference Relatedness

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

Expand Queries: Preference Relatedness

Context Matching

Use a metric for measuring the distance, similarity or difference of two contexts:

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

[Agrawal et al. 2006]

- Hierarchical-based approaches

Expand Queries: Preference Relatedness

Context Matching : Hierarchical Approach

For context parameters that take values from hierarchical domains:

– Compare contexts expressed at <u>different levels of abstraction</u>

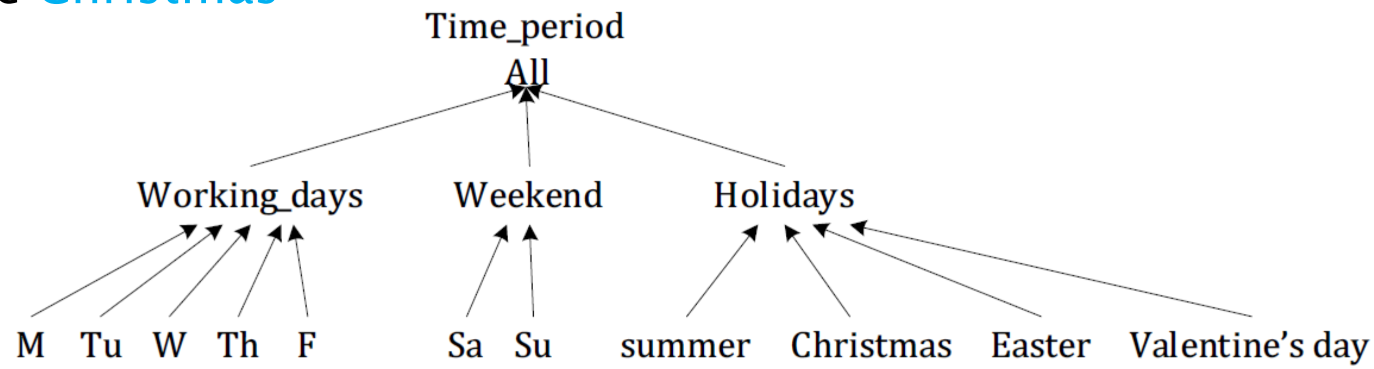
Given a preference (C, P) and a query with context C_Q:

– C is related to C_Q, if C is equal or more general than C_Q

[Stefanidis et al. 2007a]

<u>Example</u>:

For the context parameter Time_period, the value Holidays is more general than the value Christmas



Expand Queries: Preference Relatedness

Context Matching : Hierarchical Approach

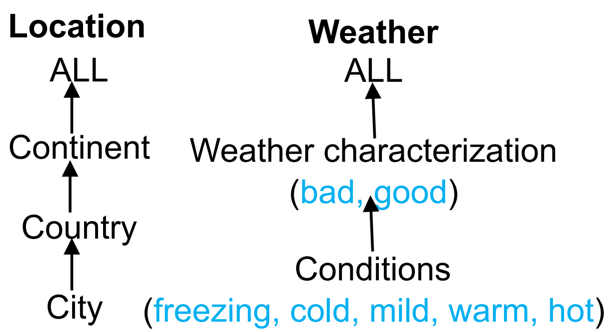
Hierarchical distance

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

Expand Queries: Preference Relatedness

Context Matching : Hierarchical Approach

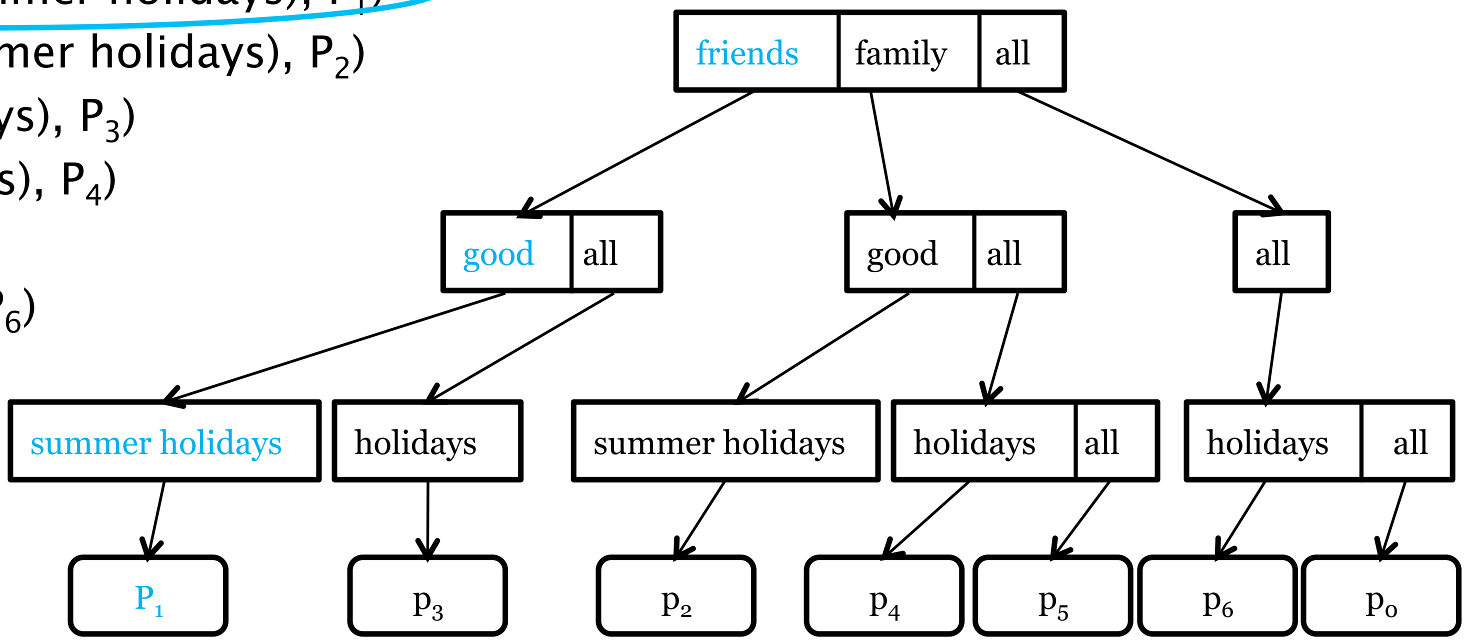
Locate the related preferences using the profile tree

- Exploit the repetition of context values in contexts

[Stefanidis et al. 2007a]

Preferences (C, P):

- ((all, all, all), P_0)
- ((friends, good, summer holidays), P_1)
- ((family, good, summer holidays), P_2)
- ((friends, all, holidays), P_3)
- ((family, all, holidays), P_4)
- ((family, all, all), P_5)
- ((all, all, holidays), P_6)



Context Matching: Relaxation Types

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]

Expand Queries: Preference Relatedness

Preference Applicability

With context matching, we identify:

- Preferences that are valid in a query context
- Preferences that are out of context

We consider the following cases of preference applicability:

- Instance applicability
- Semantic applicability
- Syntactic applicability

Expand Queries: Preference Relatedness

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

Expand Queries: Preference Relatedness

Semantic Applicability

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

Expand Queries: Preference Relatedness

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

Expand Queries: Preference Relatedness

Syntactic Applicability

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

Preference Relatedness Example

Assume the query:

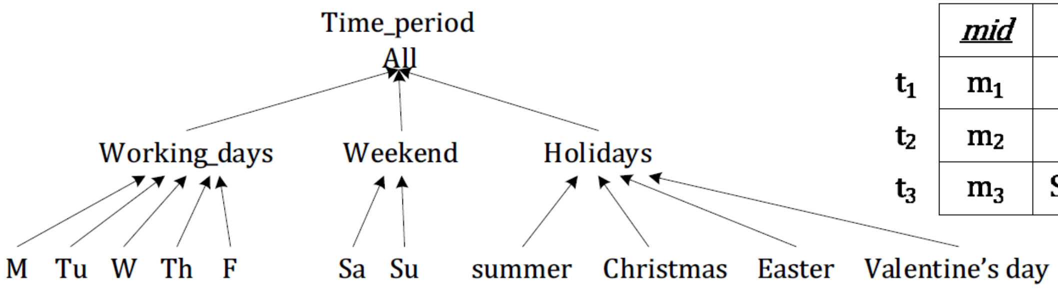
Q: (Time_period = 'Christmas', SELECT title FROM movie
WHERE genre = 'horror' AND language = 'English')

and the preferences:

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

<u>Preference Selection</u>:

- CP2 and CP3 are more closely related to Q
- CP2 is not applicable to Q
- CP3 is syntactically, instantly and semantically applicable



movie

	mid	*title*	*year*	*director*	*genre*	*language*	*duration*
t_1	m_1	Casablanca	1942	Curtiz	drama	english	102
t_2	m_2	Psycho	1960	Hitchcock	horror	english	109
t_3	m_3	Schindler's List	1993	Spielberg	drama	english	109

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

Expand Queries: Preference Filtering

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

Expand Queries: Preference Filtering

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]

Expand Queries: Preference Filtering

Filtering based on Preference Score

Quantitative preferences are ordered in decreasing preference score and the top K ones are selected for expanding the query

Expand Queries: Preference Filtering

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)

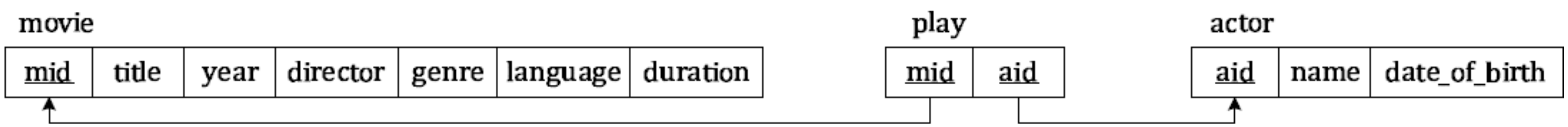
Join preferences: movie: (movie.mid = play.mid, 1)

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.

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

Expand Queries: Preference Integration

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

Query Re-Writing Mechanism Example

Example:

Assume the query

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

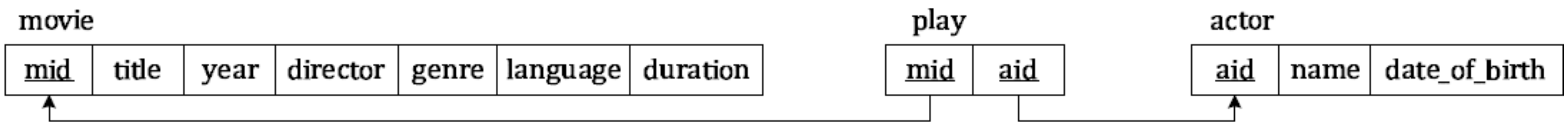
and the preferences

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')
                                    )
```



A Lattice-based Approach

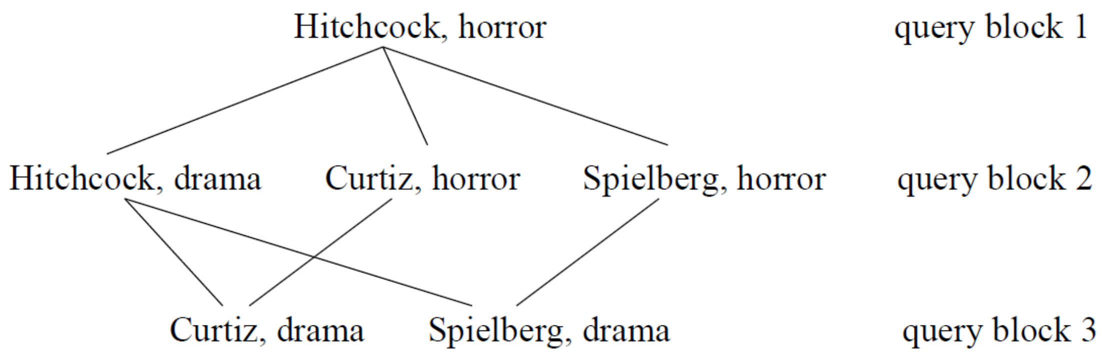
Blocks, or groups, of equivalent queries

- 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



Expand Database Queries: Summary

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

Expand Database Queries: Summary

A taxonomy of approaches that expand database queries with preferences

	Preference Relatedness			Preference Filtering		Preference Integration	
	All Preferences	Context Matching	Preference Applicability	Preference Score	Preference Relatedness	Top-K Queries	Order All Queries
[Agrawal et al. 2006]		internal			✓		
[Bunningen et al. 2006]		external	✓		✓		✓
[Georgiadis et al. 2008]	✓						✓
[Koutrika and Ioannidis 2004; 2005]			✓	✓		✓	
[Miele et al. 2009]		external			✓	✓	
[Stefanidis et al. 2007]		external			✓	✓	

Expand Database Queries

Preference integration

- Employ preference operators

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

Employ Preference Operators

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

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

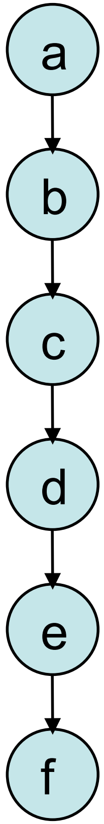
Winnow can be used to select tuples from more than one relation

- Apply winnow to the result of queries defined over more than one relation

Employ Preference Operators: Definition

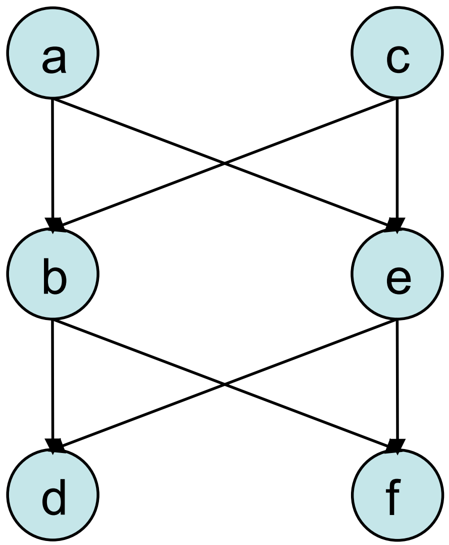
The Winnow Operator: Properties

- If $\succ_P$ is a total order, $w_P(r)$ includes just one tuple



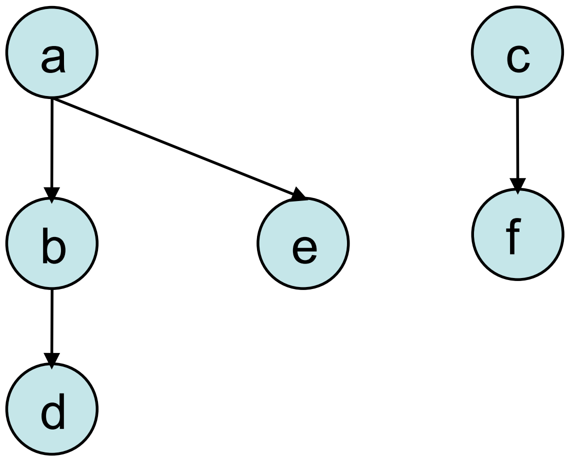
The Winnow Operator: Properties

- If $\succ_P$ is a total order, $w_P(r)$ includes just one tuple
- If $\succ_P$ is a weak order, tuples in $win_P(r)$ are tuples of the top equivalence class of r defined by $\succ$



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 $win_P(r)$ are tuples of the top equivalence class of r defined by $\succ$
- If $\succ_P$ is a strict partial order, $w_P(r)$ is non-empty (for every finite, non-empty instance r of R)



Employ Preference Operators: Definition

The Winnow Operator: Properties

- If $>_P$ is a total order, $w_P(r)$ includes just one tuple
- If $>_P$ is a weak order, tuples in $win_P(r)$ are tuples of the top equivalence class of r defined by ~
- If $>_P$ 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 $t_i \not> t_j$
  - t_i and t_j are indifferent

[Chomicki 2003]

Employ Preference Operators: Definition

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

Employ Preference Operators: Definition

Other Definitions of Skylines

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]

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

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

[Xia et al. 2008]

Employ Preference Operators: Definition

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, $win^i_P(r)$, of level i, i > 0, is:

- $win^1_P(r) = w_P(r)$
- $win^{i+1}_P(r) = w_P(r - \cup^i_{k=1} win^k_P(r))$

[Chomicki 2003]

The iterated winnow operator, called Best operator, is independently defined by [Torlone and Ciaccia 2003]

Employ Preference Operators

In following, we focus on:

- Defining preference operators
- Implementing preference operators
- Translating preference operators

Employ Preference Operators: Implementation

Within The Query Engine

The naïve approach: Nested-Loop method

- Compare each tuple with every other tuple
  - 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

Within The Query Engine

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

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

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 $win^1_P(r)$
  - Then, apply on $(r - \cup^i_{k=1} win^k_P(r))$ to produce $win^{i+1}_P(r)$

Evaluating Best Operator algorithm [Torlone and Ciaccia 2003]

BNL variation

- Compute $win^{i+1}_P(r)$ from those tuples that were found to be directly dominated by a tuple in $win^i_P(r)$

Employ Preference Operators

In following, we focus on:

- Defining preference operators
- Implementing preference operators
- Translating preference operators

Employ Preference Operators: Translation

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

Employ Preference Operators: Translation

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 PREFERRING 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

Employ Preference Operators: Summary

A taxonomy of approaches employing preference operators

		Implementation Level	
		Evaluation Techniques	Operator Translation
Query Model	Best Answers	winnow, skyline [Chomicki 2002; Borzsonyi et al. 2001; Tan et al. 2001; Kossman et al. 2002; Papadias et al. 2003; Yuan et al. 2005; Pei et al 2005; Tao et al. 2006; Chan et al. 2006; Lin et al. 2007; Xia et al. 2008]	preference selection, grouped preference selection [Kiessling 2002; Kiessling and Kostler 2002]
	Ranking	iterated winnow [Chomicki 2003; Torlone and Ciaccia 2203; Georgiadis et al. 2008; Drosou et al. 2009]	--

Expand Database Queries: Summary

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

Preferential query processing methods:

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

Pre-compute Rankings

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

Pre-compute Rankings: Context-based Approaches

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

But how the representative rankings are constructed?

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

<u>How to select the representative rankings</u>?

- <u>Greedy Algorithm</u>
  - Begin from all rankings
  - Remove at each step the ranking that is the most similar to the remaining ones
- <u>Furthest Algorithm</u>
  - 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 <u>Spearman footrule</u> or the <u>Kendall tau distance</u>

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]

Pre-computing Rankings: Summary

A taxonomy of pre-computing rankings approaches

		Context	
		Context-based	Context-free
Formulation	Qualitative	[Agrawal et al. 2006]	--
	Quantitative	[Stefanidis and Pitoura 2008; You and Hwang 2008]	[Hristidis and Papakonstantinou 2004; Das et al. 2006; Yi et al. 2003]

Preferential Query Processing

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

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

Example: FA Algorithm

S1 = < A 0.9, **C 0.8,** <u>**E**</u> **0.7, B 0.5, F 0.5, G 0.5, H 0.5** >
S2 = < **B 1.0,** <u>**E**</u> **0.8, F 0.7, A 0.7, C 0.5, H 0.5, G 0.5** >
S3 = < A **0.8, C 0.8,** <u>**E**</u> **0.7, B 0.5, F 0.5, G 0.5, H 0.5** >

<u>Which is the top-1 item</u>?

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

Top-k Processing

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

S1 = < A 0.9, C 0.8, E 0.7, B 0.5, F 0.5, G 0.5, H 0.5 >
S2 = < B 1.0, E 0.8, F 0.7, A 0.7, C 0.5, H 0.5, G 0.5 >
S3 = < 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:

score(A) = 0.9 + 0.7 + 0.8 = 2.4

score(B) = 0.5 + 1.0 + 0.5 = 2.0

threshold_value = 0.9 + 1.0 + 0.8 = 2.7 Continue since 2.7 > 2.4

Step2:

score(C) = 0.8 + 0.5 + 0.8 = 2.1

score(E) = 0.7 + 0.8 + 0.7 = 2.2

threshold_value = 0.8 + 0.8 + 0.8 = 2.4 Stop since score(A) = threshold_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]

Top-k Processing: Summary

A taxonomy of top-k query processing techniques

		Implementation Level	
		Application level	Within engine
Query Model	Top-k tuples	[Fagin et al. 2001; Nepal and Ramakrishna 1999; Guntzer et al. 2000]	--
	Top-k joined tuples	[Natsev et al. 2001]	[Ilyas et al. 2004]
	Top-k groups of tuples	--	[Li et al. 2006]

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)

Preference Learning

- 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.

[Jiang et al. 2008], [Wong et al. 2007]

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.

Preference Learning

[Cohen et al. 1999]

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

Model: *Pref* (i_1; i_2), *Pref* : I x I → [0; 1], returns a value indicating which item is ranked higher.

Learning: At each round, items are ranked with respect to *Pref*. Then, the learner receives feedback from the environment. Given that *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
  - Existing methods tackle specific aspects of the problem
  - A holistic preference representation approach is missing
  - Complete understanding of user preferences is missing – (psychology?)
  - New types of preferences (membership, uncertain, …)
- **Preference Composition**
- **Preferential Query Processing**
- **Preference Learning**

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**

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**

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

Future Directions

- Leveraging the wisdom of crowds
  - Learning preferences
- Preference-aware query engine
  - Making preferences first-class citizens
  - Holistic optimizer

The End

References

1. Agrawal, R., Rantzau, R., and Terzi, E. 2006. *Context-sensitive ranking*. In *SIGMOD*. 383–394.
2. Agrawal, R. and Wimmers, E. L. 2000. *A framework for expressing and combining preferences*. In *SIGMOD*. 297–306.
3. Aho, A., Sagiv, Y., and Ullman, J. D. 1979. *Equivalence of relational expressions*. *SIAM J. of Computing* 8, 2, 218–246.
4. Amer-Yahia, S., Roy, S. B., Chawla, A., Das, G., and Yu, C. 2009. *Group recommendation: Semantics and efficiency*. *PVLDB* 2, 1, 754–765.
5. Bolchini, C., Curino, C., Quintarelli, E., Schreiber, F. A., and Tanca, L. 2007. *A data-oriented survey of context models*. *SIGMOD Record* 36, 4, 19–26.
6. Borda, J.-C. 1781. *Mémoire sur les élections au scrutin. Histoire de l' Académie Royale des Sciences.*
7. Borzsonyi, S., Kossmann, D., and Stocker, K. 2001. *The skyline operator*. In *ICDE*. 421–430.
8. Boutilier, C., Brafman, R. I., Hoos, H. H., and Poole, D. 1999. *Reasoning with conditional ceteris paribus preference statements*. In *Sym. on Uncertainty in AI*. 71–80.
9. Brown, P., Bovey, J., and Chen, X. 1997. *Context-aware applications: From the laboratory to the marketplace*. *IEEE Personal Communications* 4, 5, 5864.
10. Bunningen, A. H., Feng, L., and Apers, P. M. G. 2006. *A context-aware preference model for database querying in an ambient intelligent environment*. In *DEXA*. 33–43.

References

11. Chan, C. Y., Jagadish, H. V., Tan, K.-L., Tung, A. K. H., and Zhang, Z. 2006. *Finding k-dominant skylines in high dimensional space*. In *SIGMOD*. 503–514.
12. Chekuri, C. and Rajaraman, A. 1997. *Conjunctive query containment revisited*. In *ICDT*. 56–70.
13. Chen, G. and Kotz, D. 2000. *A Survey of Context-Aware Mobile Computing Research*. Tech. Rep. TR2000-381, Dartmouth College, Computer Science. November.
14. Chomicki, J. 2002. *Querying with intrinsic preferences*. In *EDBT*. 34–51.
15. Chomicki, J. 2003. *Preference formulas in relational queries*. *ACM Trans. Database Syst.* 28, 4, 427–466.
16. Chomicki, J. 2007. *Semantic optimization techniques for preference queries*. *Inf. Syst.* 32, 5, 670–684.
17. Chomicki, J., Godfrey, P., Gryz, J., and Liang, D. 2003. *Skyline with presorting*. In *ICDE*. 717–719.
25. Condorcet, J. A. N. 1785. *Essai Sur L' application De L' analyse a La Probabilité Des Décisions Rendues a La Pluralité Des Voix. Kessinger Publishing.*
26. Das, G., Gunopulos, D., Koudas,N., and Tsirogiannis, D. 2006. *Answering top-k queries using views*. In *VLDB*. 451–462.
27. Delgrande, J. P., Schaub, T., and Tompits, H. 2003. *A framework for compiling preferences in logic programs*. *TPLP* 3, 2, 129–187.
28. Dey, A. K. 2001. *Understanding and using context*. *Personal Ubiquitous Comput*. 5, 1, 4–7.

References

22. Doyle, J. 2004. *Prospects for preferences*. *Computational Intelligence* 20, 2.
23. Drosou, M., Stefanidis, K., and Pitoura, E. 2009. *Preference-aware publish/subscribe delivery with diversity*. In *DEBS*. 1–12.
24. Fagin, R. 1999. *Combining fuzzy information from multiple systems. Journal of Computer and System Sciences* 58, 1, 83–99.
25. Fagin, R., Lotem, A., and Naor, M. 2001. *Optimal aggregation algorithms for middleware*. In *PODS*.
26. Fishburn, P. C. 1999. *Preference structures and their numerical representations*. *Theoretical Computer Science* 217, 2, 359–383.
27. Gaasterland, T. and Lobo, J. 1994. *Qualified answers that reflect user needs and preferences*. In *VLDB*. 309–320.
28. Georgiadis, P., Kapantaidakis, I., Christophides, V., Nguer, E. M., and Spyratos, N. 2008. *Efficient rewriting algorithms for preference queries*. In *ICDE*. 1101–1110.
36. Guntzer, U., Balke, W.-T., and Kießling, W. 2000. *Optimizing multi-feature queries for image databases*. In *VLDB*. 419–428.
37. Guntzer, U., Balke, W.-T., and Kießling, W. 2001. *Towards efficient multi-feature queries in heterogeneous environments*. In *ITCC*. 622–628.
38. Guo, L., Amer-Yahia, S., Ramakrishnan, R., Shanmugasundaram, J., Srivastava, U., and Vee, E. 2008. *Efficient top-k processing over query-dependent functions*. In *VLDB*. 1044–1055.
39. Hafenrichter, B. and Kießling, W. 2005. *Optimization of relational preference queries*. In *ADC*. 175–184.

References

33. Hansson, S. O. 2001. *Preference logic. Handbook of Philosophical Logic* (D. Gabbay, Ed.) 8.
34. Holland, S. and Kießling,W. 2004. *Situated preferences and preference repositories for personalized database applications*. In *ER*. 511–523.
35. Hristidis, V. and Papakonstantinou, Y. 2004. *Algorithms and applications for answering ranked queries using ranked views*. *VLDB J*. 13, 1, 49–70.
36. Ilyas, I. F., Aref, W. G., and Elmagarmid, A. K. 2004. *Supporting top-k join queries in relational databases. VLDB J*. 13, 3, 207–221.
37. Ilyas, I. F., Beskales, G., and Soliman, M. A. 2008. *A survey of top-k query processing techniques in relational database systems. ACM Comput. Surv*. 40, 4.
38. Ilyas, I. F., Shah, R., Aref, W. G., Vitter, J. S., and Elmagarmid, A. K. 2004. *Rank-aware query optimization*. In *SIGMOD*. 203–214.
39. Kendall, M. G. 1945. *The treatment of ties in ranking problems. Biometrika* 33, 3, 239–251.
40. Kießling, W. 2002. *Foundations of preferences in database systems*. In *VLDB*. 311–322.
49. Kießling, W. 2005. *Preference queries with sv-semantics*. In *COMAD*. 15–26.
50. Kießling, W. and Kostler, G. 2002. *Preference sql - design, implementation, experiences*. In *VLDB*. 990–1001.
51. Kossmann, D., Ramsak, F., and Rost, S. 2002. *Shooting stars in the sky: an online algorithm for skyline queries*. In *VLDB*. 275–286.

References

44. Koutrika, G. and Ioannidis, Y. 2005a. *Constrained optimalities in query personalization*. In *SIGMOD*. 73–84.
45. Koutrika, G. and Ioannidis, Y. 2005b. *Personalized queries under a generalized preference model*. In *ICDE*. 841–852.
46. Koutrika, G. and Ioannidis, Y. 2010. *Answering queries based on preference hierarchies*. *TODS*.
47. Koutrika, G. and Ioannidis, Y. E. 2004. *Personalization of queries in database systems*. In *ICDE*. 597–608.
48. Lacroix,M. and Lavency, P. 1987. *Preferences: putting more knowledge into queries.* In *VLDB*. 217–225.
49. Levandoski, J., Mokbel, M. F., and Khalefa, M. 2010. *Flexpref: A framework for extensible preference evaluation in database systems*. In *ICDE*.
50. Li, C., Chang, K. C.-C., and Ilyas, I. F. 2006. *Supporting ad-hoc ranking aggregates*. In *SIGMOD*. 61–72.
58. Lin, X., Yuan, Y., Zhang, Q., and Zhang, Y. 2007. *Selecting stars: The k most representative skyline operator*. In *ICDE*. 86–95.
59. Miele, A., Quintarelli, E., and Tanca, L. 2009. *A methodology for preference-based personalization of contextual data*. In *EDBT*. 287–298.
60. Natsev, A., Chang, Y.-C., Smith, J. R., Li, C.-S., and Vitter, J. S. 2001. *Supporting incremental join queries on ranked inputs*. In *VLDB*. 281–290.

References

54. Nepal, S. and Ramakrishna, M. V. 1999. *Query processing issues in image (multimedia) databases*. In *ICDE*. 22–29.
56. Papadias, D., Tao, Y., Fu, G., and Seeger, B. 2003. *An optimal and progressive algorithm for skyline queries*. In *SIGMOD*. 467–478.
57. Pei, J., Jin, W., Ester, M., and Tao, Y. 2005. *Catching the best views of skyline: A semantic approach based on decisive subspaces*. In *VLDB*. 253–264.
58. Ross, K. A., Stuckey, P. J., and Marian, A. 2007. *Practical preference relations for large data sets*. In *ICDE Workshops*. 229–236.
59. Schmidt, A., Aidoo, A. K., Takaluoma, A., Tuomela, U., Laerhoven, K., and de Velde, M. 1999. *Advanced interaction in context*. In *Handheld and Ubiquitous Computing*. 89101.
60. Stefanidis, K. and Pitoura, E. 2008. *Fast contextual preference scoring of database tuples*. In *EDBT*. 344–355.
61. Stefanidis, K., Pitoura, E., and Vassiliadis, P. 2006. *Modeling and storing context-aware preferences*. In *ADBIS*. 124–140.
68. Stefanidis, K., Pitoura, E., and Vassiliadis, P. 2007a. *Adding context to preferences*. In *ICDE*. 846–855.
69. Stefanidis, K., Pitoura, E., and Vassiliadis, P. 2007b. *On relaxing contextual preference queries*. In *MDM*. 289–293.

References

63. Tan, K.-L., Eng, P.-K., and Ooi, B. C. 2001. *Efficient progressive skyline computation*. In *VLDB*. 301–310.
64. Tao, Y., Xiao, X., and Pei, J. 2006. *Subsky: Efficient computation of skylines in subspaces*. In *ICDE*. 65.
65. Torlone, R. and Ciaccia, P. 2002. *Finding the best when it's a matter of preference*. In *SEBD*. 347–360.
66. Torlone, R. and Ciaccia, P. 2003. *Management of user preferences in data intensive applications*. In *SEBD*. 257–268.
67. Vee, E., Srivastava, U., Shanmugasundaram, J., Bhat, P., and Amer-Yahia, S. 2008. *Efficient computation of diverse query results*. In *ICDE*. 228–236.
73. Wellman, M. P. and Doyle, J. 1991. *Preferential semantics for goals*. *AAAI*. 698703.
74. Xia, T., Zhang,D., and Tao, Y. 2008. *On skylining with flexible dominance relation*. In *ICDE. 1397–1399.*
75. Yi, K., Yu, H., Yang, J., Xia, G., and Chen, Y. 2003. *Efficient maintenance of materialized top-k views*. In *ICDE. 189–200.*
76. You, G. and Hwang, S. 2008. *Search structures and algorithms for personalized ranking*. *Information Sciences 178, 20, 3925–3942.*
77. Yuan, Y., Lin, X., Liu, Q.,Wang, W., Yu, J. X., and Zhang, Q. 2005. *Efficient computation of the skyline cube*. In *VLDB. 241–252.*
78. Zhang, X. and Chomicki, J. 2008. *Profiling sets for preference querying*. In *SEBD. 34–44.*