# Rewriting preferences as queries

**Nicolas Spyratos**

University of Paris 11
Orsay Centre

**fact:**
**when you query a big data set chances are that the answer will be big as well**

**problem:**
**it is difficult for the user to find an item of interest in a big answer set**

**one solution:**
**partition the answer into smaller sets and show one part at a time**
(showing “best parts” first)
→ ***one way to achieve this is to exploit user preferences***

# various kinds of user preferences



- **in terms of their nature preferences can be:**

**quantitative** (John likes BMWs 80%, and VWs 60%, …)

**qualitative** (I like BMW more than VW)
→ *easy to express by the casual user*

- **in terms of their persistence in time preferences can be:**

**long term preferences**
either discovered by the system (unobtrusively, from query logs) or declared explicitly by the user

**short term preferences**
expressed explicitly by the user, online

- **the nature and persistence in time are orthogonal features of preferences**
- **in this presentation we focus on qualitative preferences** (whether short or long term)

## the basic setting through an example



**Table T** (e-catalogue such as Autoreflex)

| Serial | Make | Color | Mileage | Price | Year |
|---|---|---|---|---|---|
| **1** | BMW | black | 35000 | 3800 | 2002 |
| **2** | Honda | blue | 63000 | 2900 | 2000 |
| **...** | **...** | **...** | **...** | **...** | **...** |

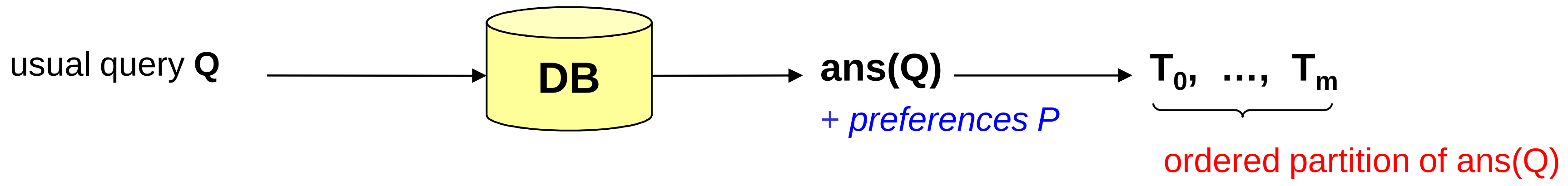
**as T is usually large chances are that query answers are large as well**

**usual query Q: VW∨BMW** (meaning: select * from T where (Make=VW)∨(Make=BMW))

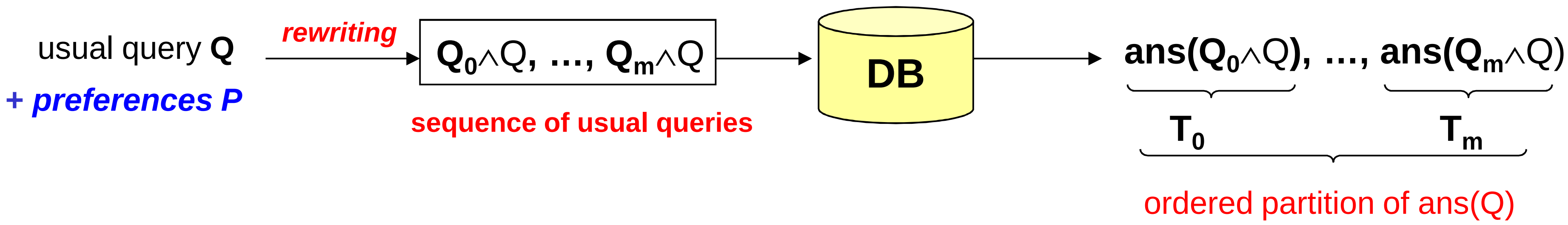
**preference P: Red → Black** (meaning "Red is preferred to Black")

**answer**:
**standard approach:** compute ans(Q) then use P to partition ans(Q) into a sequence of two tables: $T_{Red}$ , $T_{Black}$
**rewriting approach**: use P to rewrite Q into a sequence of two queries: $Q_{Red} \wedge Q$ , $Q_{Black} \wedge Q$ whose answers are $T_{Red}$ , $T_{Black}$

***standard approach:***




***rewriting approach:***




**advantage of rewriting** :
rewriting is transparent to the system, it is possible to do incremental evaluation under user control

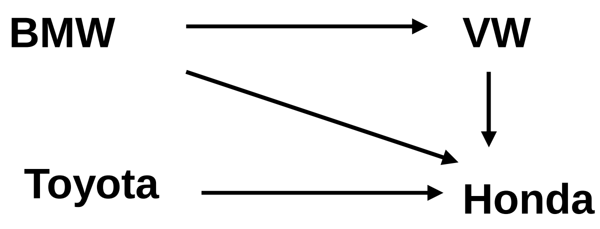
**the problem we consider**: **rewrite P into the sequence $Q_0$, …, $Q_m$**
(forgetting about Q for the moment, i.e. P ***is*** the query)

***ICDE 2008 , ADBIS 2011, ASSETS Project** (2010-2012), **ICFCA 2012,***

***preference over attribute A*** **: pair of values (v, w) from the domain of A** ($v \neq w$)
(read as " v is preferred to w" or " v precedes w", or "v dominates w", denoted as $v \rightarrow w$)

***preference relation or preference graph over A*** **: a set P.A of preferences over A**

**P.Make:**




**assumption :** the only values of interest to the user are those expressed in the preference graph (hence a sort of "closed world assumption")
as a consequence, the only tuples of interest are those in the answer to the following query:
***induced query:*** Q(P.A)= disjunction of all values in P.A
in our example: Q(P.Make)= BMW∨Toyota∨VW∨Honda

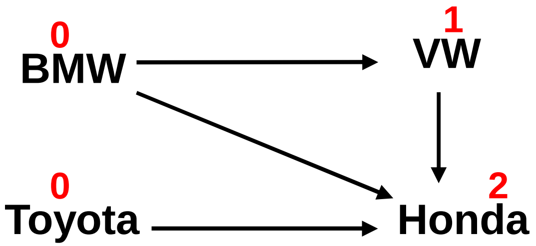
**no other assumption is made on the preference relation**
(in particular, the preference relation may not be transitive and may contain cycles)

## ranking of nodes in the preference graph :

(assuming acyclicity for now and *we'll see shortly how we can handle cycles)*

***if*** *v* is a root *then* rank(*v*)=0

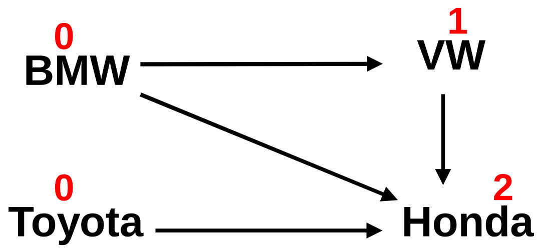
***else*** rank(*v*)= the maximal length of path among all paths going from a root to *v*




$$B_i = \{v / \text{rank}(v) = i\}$$

- **the further away from the roots the less preferred a value is**
- **each non root value of rank i>0 is preceded by a value of rank i-1**

## rewriting a preference graph into a sequence of queries based on ranks:




node ranking →

$B_0$= {BMW, Toyota}
$B_1$= {VW}
$B_2$= {Honda}

↓ query definition

$Q_0$= BVW∨Toyota
$Q_1$=VW
$Q_2$= Honda

**actually, we have an ordered partition of the induced query:**

induced query : BMW∨Toyota ∨ VW ∨ Honda

rewritten query: $\underbrace{\text{BMW}\vee\text{Toyota}}_{Q_0} \vee \underbrace{\text{VW}}_{Q_1} \vee \underbrace{\text{Honda}}_{Q_2}$

- **each query $Q_i$ is evaluated in order of increasing rank** (hence best answers first)
- **evaluation is under user control** (incremental evaluation)
- **result presentation adapts to user preferences**

*(i.e. the answer to the same induced query will be presented differently for different preferences)*

- **skyline of T:**

**it is defined to be the set of all non dominated tuples of T** (w.r.t. the given preferences), **hence it is the answer to the first query $Q_i$ such that ans($Q_i$)≠∅**
(and this $Q_i$ depends on the current instance of T)

# additional issues

**user profile:**

**the sequence $Q_0$, $Q_1$, $Q_2$ , …, $Q_m$ into which the preference graph is rewritten can be seen as the user's prefrence profile** (to be stored, and invoked in subsequent query sessions)

**$Q_i$ 's with large answer sets:**

**$Q_i$ can be modified using order-by and/or top k and/or "hard conditions"**
(eventually through a dialogue with the user)

**Ex:**

Q : select * from T where (P.Make: BMW→VW) order-by Km
is rewritten as:
Q0: select * from T where (Make=BMW) order-by Km
Q1: select * from T where (Make=VW) order-by Km

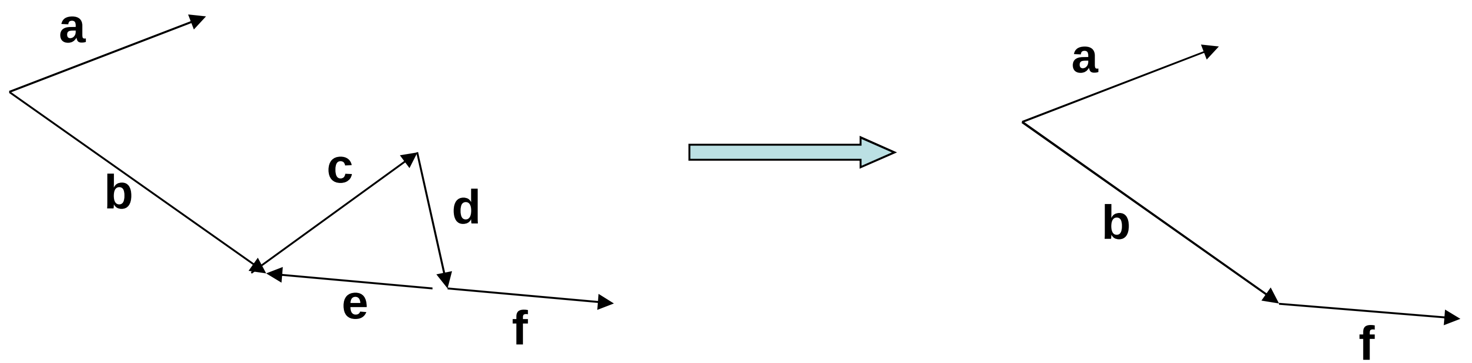
# handling cycles in the preference graph

1. **with the help of the user** : show the cycles and ask the user to ‘break’ them
   → need for algorithms for finding all cycles in the preference graph

**2. without user involvement**:

2.1 refusing the addition of a preference if the current acyclic graph becomes cyclic

2.2 considering all values in a cycle to be of equal preference, thus turning each maximal cycle into one node
→ need for algorithms for finding all cycles in the preference graph

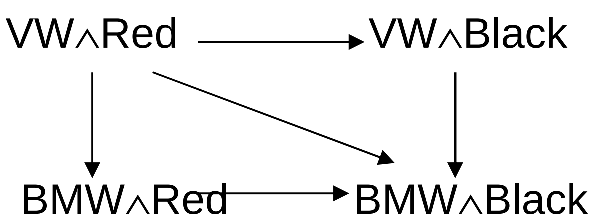


# extending the approach

**preferences over more than one attribute can be declared either <u>conjunctively</u> or <u>independently</u>**

**in a conjunctive declaration we proceed as in the case of a single attribute**
(the only difference is that the nodes are conjunctions of values instead of single values)

ex: **P.{Make, Colour}** = {VW∧Red→VW∧Black, BMW∧Red→BMW∧Black}




node ranking

$B_0$= {VW∧Red}
$B_1$= {VW∧Black, BMW∧Red}
$B_2$= {BMW∧Black}

query definition

$Q_0$= {VW∧Red}
$Q_1$={(VW∧Black)∨(BMW∧Red)}
$Q_2$= BMW∧Black

**an independent declaration can be converted into a conjunctive one in two steps**

**Step 1: combine the values present in the preference relations:**

VW $\rightarrow$ BMW
Red $\rightarrow$ Black

**combining values** $\Longrightarrow$

VW $\wedge$ Red    VW $\wedge$ Black

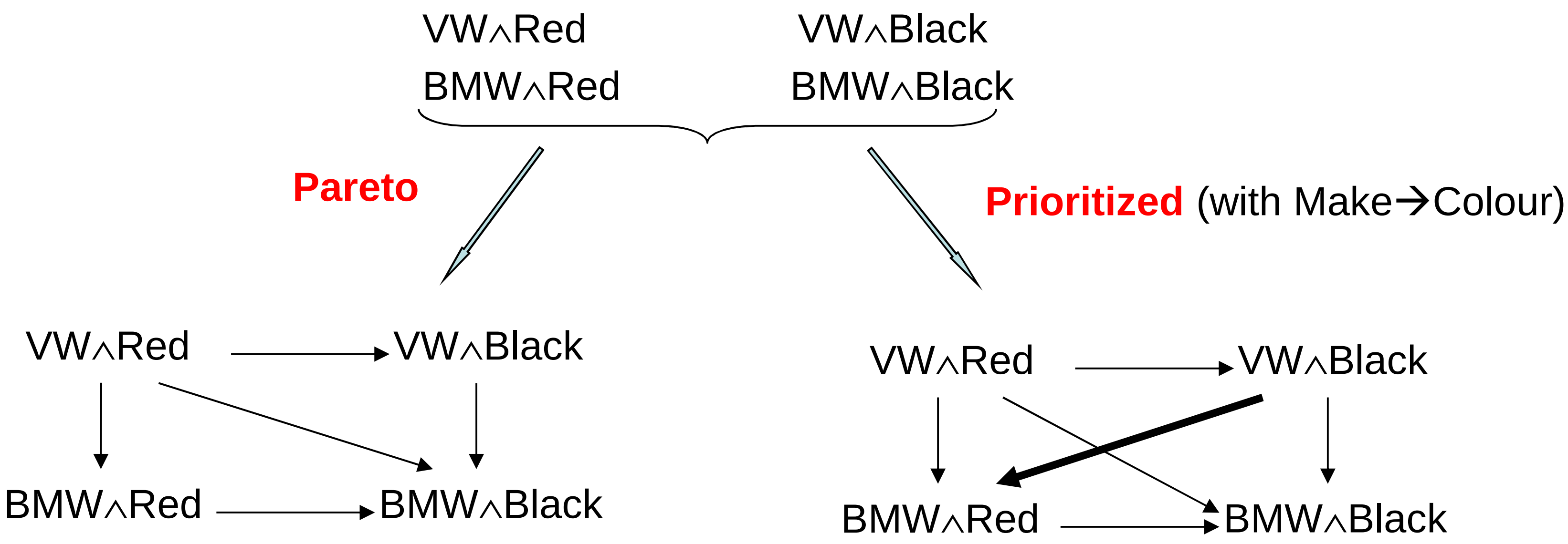
BMW $\wedge$ Red    BMW $\wedge$ Black

**Step 2: define a preference rel. over the combined values based on the given preference rel.**

**to perform this second step we need a preference rel. over the attributes** (given by the user)

**Pareto: all attributes are of the same importance**
**Prioritized: there is a prioritiy over attributes**
(priority: linear ordering of the attributes over which preferences are declared)

**Preferences :** **P.Make**= {VW→BMW}, **P.Colour**= {Red→Black}




***once the graph is constructed we proceed to rewriting as for single attributes:***

**Pareto:** $B_0$= {VW∧Red}, $B_1$= {VW∧Black**,** BMW∧Red}, $B_2$= {BMW∧Black}

**Prioritized:** $B_0$= {VW∧Red}, $B_1$= {VW∧Black}**,** $B_2$= {BMW∧Red}, $B_3$= {BMW∧Black}

***question:* can we find the $\mathbf{B_0}$, $\mathbf{B_1}$, …. *without constructing the derived graphs?***
*(i.e. based on the graphs of the individual attributes)*

## avoiding derived graphs:



**P.Make**= {VW→BMW} $B_0^M$ = {VW} , $B_1^M$ = {BMW}

**P.Colour**= {Red→Black} $B_0^C$ = {Red}, $B_1^C$ = {Black}

**Prioritized :**
(Make→ Colour)

$B_0 = B_0^M \times B_0^C = \{VW \wedge Red\}$

$B_1 = (B_0^M \times B_1^C) = \{VW \wedge Black\}$

$B_2 = B_1^M \times B_0^C) = \{BMW \wedge Red\}$

$B_3 = B_1^M \times B_1^C) = \{BMW \wedge Black\}$

**skyline**: the answer of the first query $Q_i$ such that $ans(Q_i) \neq \emptyset$

**Pareto :**

$B_0 = B_0^M \times B_0^C = \{VW \wedge Red\}$

$B_1 = (B_0^M \times B_1^C) \cup (B_1^M \times B_0^C) = \{VW \wedge Black, BMW \wedge Red\}$

$B_2 = B_1^M \times B_1^C) = \{BMW \wedge Black\}$

**skyline**: the answer of the first query $Q_i$ such that $ans(Q_i) \neq \emptyset$ is a <u>subset</u> of the skyline , not necessarily the whole skyline

(because of the unions)

**1:** $a \rightarrow b \rightarrow c$ $B_0^1 = \{a\}$ , $B_1^1 = \{b\}$, $B_2^1 = \{c\}$

**2:** $x \rightarrow y \rightarrow z$ $B_0^2 = \{x\}$ , $B_1^2 = \{y\}$, $B_2^2 = \{z\}$

**Pareto over 1, 2:**

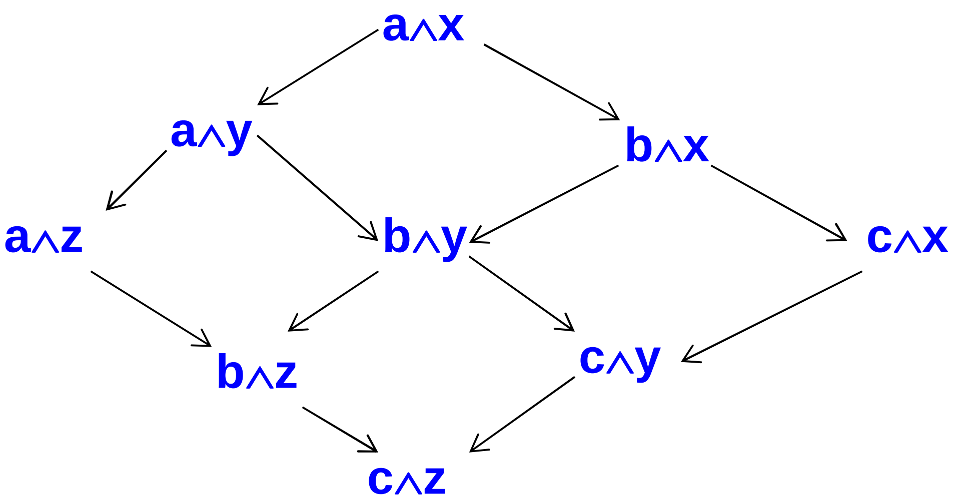
$Q_0 = a \wedge x$

$Q_1 = (a \wedge y) \vee (b \wedge x)$

$Q_2 = (a \wedge z) \vee (b \wedge y) \vee (c \wedge x)$

$Q_3 = (b \wedge z) \vee (c \wedge y)$

$Q_4 = c \wedge z$




assuming $ans(a \wedge x) = \emptyset$ and $ans(b \wedge x) = \emptyset$,
the skyline is $ans(a \wedge y) \cup ans(c \wedge x)$

- skyline can be computed incrementally (under user control)
- can be used to compute skylines over joins <u>without</u> computing the join

# summarizing

**preference query**= **usual query** + **preferences** + **priority** + **skyline** + **.....**

(**Price**<2000)**&**(**Km**<60000)

**P.Model** = VW→Clio
**P.Colour**: Red→Black

**Priority**: Model→Colour

. . . . . . . . . . . . . . . . . . . .

**usual query**

**preferences**

**Priority**

**............**

**preference query**

## concluding remarks



- **attribute values organized in a taxonomy** (domains of relational tables are unordered sets)
- **do we need to impose conditions on preference relations** (e.g. transitivity)
- **do we need more general kinds of priorities?**
  (e.g. priority over attribute sets, each set assumed to be Pareto)
- **what happens with attributes over which no preferences are expressed**
  don't care assumption? (as usual in databases), ceteris paribus assumption? a preference specification language for expressing preferences and conditions?
- **combining usual queries and preferences** (priority to the query? to the preferences? no priority?)

E
E N D
D