Computing Hierarchical Skyline Queries "On-the-Fly" in a Data Warehouse

Tassadit Bouadi[1], Marie-Odile Cordier[1], and René Quiniou[2]

[1] IRISA - University of Rennes 1
[2] IRISA - INRIA Rennes
Campus de Beaulieu, 35042 RENNES, France
{tassadit.bouadi,marie-odile.cordier}@irisa.fr, rene.quiniou@inria.fr

Abstract. Skyline queries represent a powerful tool for multidimensional data analysis and for decision aid. When the dimensions are conflicting, skyline queries return the best compromises associated with these dimensions. Many studies have focused on the extraction of skyline points in the context of multidimensional databases, but, to the best of our knowledge, none of them have investigated skyline queries, when data are structured along multiple and hierarchical dimensions. This article proposes a new method that extends skyline queries to multiple hierarchical dimensions. Our proposal, *HSky* (Hierarchical Skyline Queries) allows the user to navigate along the dimensions hierarchies (i.e. specialize / generalize) while ensuring an efficient online calculation of the associated skyline.

1 Introduction

Skyline queries represent a powerful tool for multidimensional data analysis and decision-making. When the dimensions are in conflict, the skyline queries return the best compromises on these dimensions. Skyline queries have been extensively studied [1, 2] in the database and the artificial intelligence communities. Several studies have investigated [3, 4] the problem of expressing and evaluating OLAP preferences, but few of them have addressed the problem of skyline computation when dealing with aggregated data and hierarchical dimensions in a data warehouse. The aim is to couple OLAP with skyline analysis to enable the user to select the most interesting facts from the data warehouse. Therefore, the main challenge is to compute skylines efficiently over hierarchical dimensions and over aggregated data. This problem rises several scientific and technical issues. Should the skylines be recomputed at every hierarchical level? Can the skyline of a given level be derived from the skyline at lower or higher level? Can conventional skyline algorithms be extended to cope with hierarchical dimensions?

Recent work [5, 6, 7] has considered the computation of skyline queries over aggregated data. These proposals have focused on the optimization of queries involving both *Skyline* and *Group-By* operators. But, they propose to execute the two operators sequentially without a real coupling. More interestingly, the operator *Aggregate Skyline* proposed by the authors of [8], combines the functionalities of both *Skyline* and *Group-By* operators. The definition of the dominance

L. Bellatreche and M.K. Mohania (Eds.): DaWaK 2014, LNCS 8646, pp. 146–158, 2014.

relation is extended to point groups. This operator enables the user to perform skyline queries on point groups in order to select the most relevant group.

The aim of this work is to propose an efficient approach simulating the effect of the OLAP operators "drill-down" and "roll-up" on the computation of skyline queries. The proposed method *HSky (Hierarchical Skyline queries)* provide the user with an interactive navigation tool that let him specialize/generalize a basic preference and its associated skyline, and derive the corresponding skylines while respecting the hierarchical structure of the involved dimensions. Properties of the hierarchical relationships between preferences associated with the different dimensions are used to design an efficient navigation schema among the preferences, while ensuring an online computation of skyline queries.

In Section 2, we introduce the basic concepts related to skyline queries and preference orders. Section 3 develops the formal aspects of our new approach *HSky* and its implementation. Section 4 gives the results of the experimental evaluation performed on synthetic datasets and highlights the relevance of the proposed solution. Section 5 concludes the paper.

2 Basic Concepts

Let $D = \{d_1, ..., d_n\}$ be an n-dimensional space, E a data set defined in space D and $p, q \in E$. $p(d_i)$ denotes the value of p on dimension d_i. A preference on the domain of a dimension d_i, denoted by $\wp_{d_i}$, is defined by a partial order $\leqslant_{d_i}$. This order can also be represented by the set of binary preferences (possibly infinite) $\wp_{d_i} = \{(u, v) | u \leqslant_{d_i} v\}$, where (u, v) is an ordered pair. $\wp = \bigcup_{i=1}^{|D|} \wp_{d_i}$ denotes the set of preferences associated to the space D. p is said to *dominate* q in D, denoted by $p \prec_D q$, if $\forall d_i \in D$, $p(d_i) \leqslant_{d_i} q(d_i)$ (i.e. p is preferred or equal to q on D) and $\exists d_i \in D$, $p(d_i) <_{d_i} q(d_i)$ (i.e. p is strictly preferred to q on some dimension d_i). For better readability, $p \prec_D q$ is simply noted $p \prec q$.

Definition 1. ***(Skyline)*** *Let $\wp$ be the preference set on D. The skyline of the dataset E on the dimension space D is the set of points that are not dominated by any point in E: $Sky(D, E)_\wp = \{p \in E | \neg(\exists q \in E, q \prec_D p)\}$.*

Property 1. *[9]* ***(Monotonicity of preference extension)*** *Let $\wp'$ and $\wp''$ be two preference sets on D. If $\wp' \subseteq \wp''$ (i.e. $\wp''$ is an extension of $\wp'$) then $Sky(D, E)_{(Z,\wp'')} \subseteq Sky(D, E)_{(Z,\wp')}$.*

Example 1. ***(Running example)*** *In this paper, we will use as running example the dataset E in Table 1. It contains 6 agricultural plots described by 3 dimensions: location (Loc), nitric pollution rate (Np) kgN/ha/year and crop yield (Yd) kg/ha. A basic preference $\wp^0_{d_i}$ is defined for each dimension d_i of the space D. The order relations $\leqslant_{Yd}$ and $\leqslant_{Np}$ are based on the order relation on natural numbers, and specify that plots with the highest crop yield (Yd) and with the lowest nitric pollution rate (Np) are preferred. The values of dimension Loc are associated with the preference order $\{Brittany <_{Loc} Epte, Yeres <_{Loc} Normandy\}$. The remaining values of dimension Loc are left unordered. $Sky(D, E)_{\wp^0} = \{a, b, e, c, d, f\}$.*

Table 1. A set of agricultural plots

ID plot	Loc	Np	Yd
a	Pays de la Loire (PL)	16	200
b	Yeres (YRS)	24	500
c	Vilaine (VLN)	36	100
d	Yar	30	200
e	ALL	23	400
f	Epte (EPT)	30	300

2.1 Hierarchy Formalization

In data warehouses, the domain values of dimensions are structured in hierarchies. Each dimension $d_i \in D$ is associated with a hierarchy that is represented by a directed acyclic graph whose nodes represent subsets of d_i domain values and edges represent set inclusion relationships. The most general value is at the root of the hierarchy and the leaves correspond to the most specific values.

Definition 2. ***(Hierarchy)*** *Let $H_D = \{h_{d_1}, ..., h_{d_{|D|}}\}$ be the set of hierarchies associated with the dimensions of space D where:*

- *h_{d_i} represents the hierarchical relationship, possibly empty (i.e. the graph reduced to the single node ALL), between the values of dimension $d_i \in D$,*
- *h_{d_i} is a directed acyclic graph,*
- *for each node $n_j \in h_{d_i}$, $label(n_j) = v_j$ with $v_j \in dom(d_i)$,*
- *for each value $v_j \in dom(d_i)$, $\exists n_j \in h_{d_i}$, $label(n_j) = v_j$.*

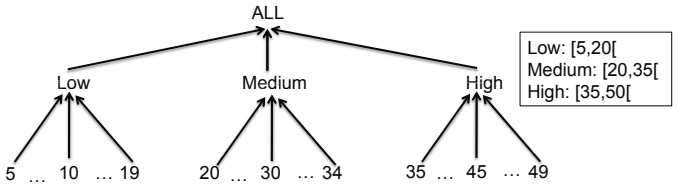
$\widehat{v_j}$ denotes the set of n_j ancestor (direct or indirect) labels in the hierarchy h_{d_i} and $\breve{v_j}$ denotes the set of its descendant labels.

The value ALL, the most general value in the hierarchy, is required to belong to the domain values of every d_i.

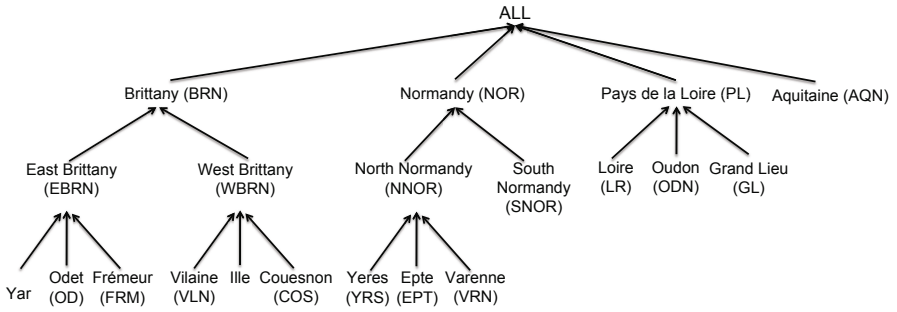
Example 2. *Figure 1 describes examples of hierarchies defined on dimensions Loc and Np. The hierarchy of dimension Loc is represented by three hierarchical levels: region (Brittany, Normandy,...), sub-region (West Brittany, North Normandy,...) and catchment (Yar, Epte,...). The domain values of Loc contains all these values associated with hierarchical levels. Similarly, the Np hierarchy introduces categorical values abstracting numerical values associated with Np.*

A hierarchy allows representing incomplete information. For example, plot *e* in table 1 is described by the value *ALL* on the dimension *Loc*. This means that there is no information on the exact location where plot *e* is located. Similarly, plot *a* in table 1 is described by the value *Pays de Loire* on dimension *Loc*. This means that the most detailed and most accurate value that exists about the location of plot *a* is its region (*Pays de Loire*). In contrast, plots *b*, *c*, *d* and *f* are described by the most precise values of the *Loc* domain.

Defining hierarchies on dimensions allows the user to generalize or specialize preferences for computing related skyline. For example, once domain values of



(a) The hierarchy of dimension Np



(b) The hierarchy of dimension *Loc*

Fig. 1

Np are partitioned in categorical values (e.g. the rates below 10 – *low* – have a low impact, the rates between 20 and 34 – *medium* – have a medium impact and the rates higher than 35 – *high* – have a strong impact), the user can express abstract preferences, and submit related abstract queries, instead of formulating preferences between individual values only (e.g. a plot with a nitric pollution rate of 16 is better than a plot with a rate of 30). Similarly, a user may be interested in more specific phenomena, e.g. green algae proliferation in coastal agricultural catchments and is likely to pay attention to specific regions of dimension *Loc* like Brittany, where this proliferation problem is well-known.

2.2 Hierarchical Relationships between Preferences

After defining the structure of hierarchical dimensions, we focus now on preference properties introduced by such dimensions.

Definition 3. ***(Preference consistency)*** *Let* $\wp_{d_i} = \{(v_1, v_2), ..., (v_n, v_m)\}$ *be a preference on the hierarchical dimension* d_i*. The preference* $\wp_{d_i}$ *is consistent if and only if it does not contain binary preference ordering a value and its ancestors:* $\nexists (v_k, v_j) \in \wp_{d_i}, k \neq j, (v_k \in \widehat{v_j} \vee v_j \in \widehat{v_k})$.

In the rest of the paper, we only consider consistent preferences and we assimilate preferences to their hierarchical closures.

Definition 4. ***(Hierarchical closure)*** *Let* $\wp_{d_i}$ *be a preference associated with the hierarchical dimension* d_i *of the multidimensional space* D*. The hierarchical*

closure of preference $\wp_{d_i}$*, noted* $(\wp_{d_i})^H$*, is defined as the set of binary preferences resulting from the transitive closure over the descendants values of* $\wp_{d_i}$*:* $(\wp_{d_i})^H = \{(v_p, v_q) \mid \exists\ (v_n, v_m) \in \wp_{d_i},\ (v_p \in \widetilde{v_n} \vee v_p = v_n) \wedge (v_q \in \widetilde{v_m} \vee v_q = v_m)\}$.

Example 3. *Let* $\wp_{Loc} = \{(BRN, EPT)\}$*. The hierarchical closure of* $\wp_{Loc}$ *is* $(\wp_{Loc})^H = \{(BRN, EPT), (EBRN, EPT), (WBRN, EPT), (Yar, EPT), (OD, EPT), (FRM, EPT), (Ille, EPT), (VLN, EPT), (COS, EPT)\}$.

Property 2. *The hierarchical closure of a preference* $\wp$ *is consistent iff* $\wp$ *is consistent.*

Starting from an initial preference $\wp^0$, called the *base preference*, we want to provide the user with means to navigate through the associated hierarchies, i.e. to generalize or to specialize the values ordered in the base preference. Specializing a preference removes indifference: it introduces a partial or total order on the direct descendants of some value in the base preference. These descendants must be non ordered initially. If an order is already specified, its completion at the same level is not considered to be a specialization but an extension of the order (cf. property 1). Specialization can be done in two ways: introducing new preference pairs either on values explicitly mentioned in the base preference or on values ignored in the base preference. Thus, the domain values of some dimension, e.g. *Loc*, can be divided into two sets: values that can be specialized (ordered values – colored green in Figure 2 – and non ordered values – colored blue) and those that cannot be specialized (colored red in Figure 2). A border can be traced between these two kinds of values (cf. Figure 2).

Example 4. *Let* $\wp^0_{Loc}$ *and* $\wp'_{Loc}$ *be two preferences associated with the hierarchical dimension Loc.* $\wp^0_{Loc} = (\{(BRN, EPT)\})^H = \{(BRN, EPT), (BBRN, EPT), (HBRN, EPT), (Yar, EPT), (OD, EPT), (FRM, EPT), (Ille, EPT), (VLN, EPT), (COS, EPT)\}$, $\wp'_{Loc} = \wp^0_{Loc} \bigcup \{(Yar, VLN)\}$ *and* $\wp''_{Loc} = \wp^0_{Loc} \bigcup \{(LR, GL)\}$. $\wp'_{Loc}$ *is a specialization of* $\wp^0_{Loc}$ *since the values of* $\{(Yar, VLN)\}$ *are descendants of* BRN *in* h_{Loc}.

$\wp''_{Loc}$ *is a specialization of* $\wp^0_{Loc}$ *with respect to* h_{Loc}*. The ancestor of* GL *and* LR *(i.e.* PL*) does not belong to* $\wp^0_{Loc}$*. This means that the value* PL *is not explicitly ordered in* $\wp^0_{Loc}$ *with respect to the other values (i.e.* BRN *and* NOR*). Let now* $\wp^0_{Loc} = (\{(BRN, EPT)\})^H$ *and* $\wp'''_{Loc} = \wp^0_{Loc} \bigcup \{(VLN, YRS)\}$. $\wp'''_{Loc}$ *is an extension of* $\wp^0_{Loc}$ *but* $\wp'''_{Loc}$ *is not a specialization of* $\wp^0_{Loc}$ *since the value* YRS *mentioned in* $\wp'''_{Loc}$ *is not the specialization of a value mentioned in* $\wp^0_{Loc}$ *(i.e. values colored green or blue Figure 2).*

Preference generalization, the dual operation of preference specialization, adds indifference between values ordered in the preference. Definition 3 gives the formal specification of preference specialization / generalization:

Definition 5. ***(Preference specialization/generalization)*** *Let* $\wp_{d_i} = \{(v_1, v_2), ..., (v_n, v_m)\}$ *and* $\wp'_{d_i} = \{(v'_1, v'_2), ..., (v'_p, v'_q)\}$ *be two preferences associated with the hierarchical dimension* d_i.

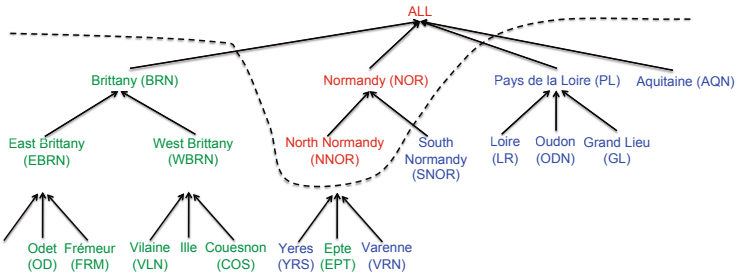


Fig. 2. Specialization/generalization implicit border

- $\wp'_{d_i}$ *is a specialization of* $\wp_{d_i}$ *(denoted* $\wp_{d_i} \subset_h \wp'_{d_i}$*) if:*
  1. $\wp_{d_i} \subset \wp'_{d_i}$ *i.e.* $\wp'_{d_i}$ *is an extension of* $\wp_{d_i}$*,*
  2. $\forall (v'_i, v'_j) \in \wp'_{d_i}$*:*
     - *either (a):* $(v'_i, v'_j) \in \wp_{d_i}$*,*
     - *or (b):*
       - $\exists (v_k, v_m) \in \wp_{d_i}$*,* $(v'_i \in \breve{v_k} \wedge v'_j \in \breve{v_k}) \vee (v'_i \in \breve{v_m} \wedge v'_j \in \breve{v_m})$ *or,*
       - $\hat{v'_i} \cap \hat{v'_j} \neq \varnothing \wedge \forall v \in \hat{v'_i} \cap \hat{v'_j}$*,* v *is not ordered in* $\wp_{d_i}$*.*

     and there exists at least one pair (v'_i, v'_j) *which verifies property (b).*
- $\wp_{d_i}$ *is a generalization of* $\wp'_{d_i}$ *if* $\wp'_{d_i}$ *is a specialization of* $\wp_{d_i}$*.*
- $\wp_{d_i}$ *and* $\wp'_{d_i}$ *are incomparable if there exists no specialization nor generalization relation between these values* $(\neg(\wp_{d_i} \subset_h \wp'_{d_i}) \wedge \neg(\wp'_{d_i} \subset_h \wp_{d_i}))$*.*

We extend the definition of specialization/generalization to preferences expressed on several hierarchical dimensions.

Definition 6. *Let* $\wp = \bigcup_{d_i \in D} \wp_{d_i}$ *and* $\wp' = \bigcup_{d_i \in D} \wp'_{d_i}$ *be two preferences associated with dimension space* D*.* $\wp'$ *is a specialization of* $\wp$ *(i.e.* $\wp$ *is a generalization of* $\wp'$*) iff* $\forall d_i \in D$*,* $\wp_{d_i} \subset_h \wp'_{d_i} \vee \wp_{d_i} = \wp'_{d_i} \wedge \exists d_i \in D$*,* $\wp_{d_i} \subset_h \wp'_{d_i}$*.*

The set of specializations and generalizations of a preference associated with a set of dimensions is partially ordered with respect to the specialization / generalization relation. For example, the set of specializations and generalizations of the base preference $\wp^0$ specified on the dimensions of Table 1, induces the hierarchical structure of Figure 3. Nodes in this structure are associated with a specialization or a generalization of the base preference $\wp^0$. To improve the readibility of figures, the notation $\wp^{kl}$, where k is a level in the structure and l is a rank, is used to specify the nodes at level k. The preferences associated with the direct descendants (resp. the direct ancestors) of node N with associated preference $\wp$ are called direct specializations (resp. direct generalizations) of $\wp$.

3 Hierarchical Skyline Queries

In this section, we give some properties of hierarchical preferences and a materialization method for storing related skylines which is grounded on these properties.

Following definition 3, the specialization of a preference is also an extension of its preference, but the converse is not true. The corollary of property 1 asserts the monotonicity of preference specialization (resp. generalization).

Corollary 1. ***(Hierarchical monotonicity)*** *Let $\wp$ and $\wp'$ be two preferences on dimension space D. If $\wp'$ is a specialization of $\wp$ ($\wp \subset_h \wp'$) then $\wp'$ is also an extension of $\wp$ and, by property 1, $Sky(D,E)_{\wp'} \subseteq Sky(D,E)_{\wp}$.*

The hierarchical monotonicity property states that every skyline point associated with a given preference remains skyline when considering a generalization of this preference. In the sequel, the hierarchical preferences designate the set of specializations and generalizations of some base preference $\wp$.

3.1 Hierarchical Skyline Query: Algorithm *HSky*

Our goal is to minimize the number of dominance test for computing efficiently the skyline sets while navigating in the hierarchical dimensions. To do so, we characterize the skyline points that remains skyline and those that become skyline or non skyline after specializing (drill-down) or generalizing (roll-up) hierarchical preferences. We propose a compromise between (i) *materializing* all skyline points of every hierarchical preferences associated with the base preference $\wp^0$, and (ii) *compute*, for every user query, the skyline points associated with the hierarchical preferences formulated in the query.

Computation of Hierarchical Skyline Queries. In order to compute the skyline points for a user query related to some specialization $\wp'$ of preference $\wp$, we introduce the set of *hierarchical skyline* points, $HNSky(D,E)_{(\wp',\wp)}$ where $\wp$ is a direct ancestor of $\wp'$ in the specialization/generalization structure. This set gathers the skyline points that are disqualified when $\wp$ is specialized in $\wp'$ or, conversely, the skyline points introduced when $\wp'$ is generalized in $\wp$.

Definition 7. ***(HNSky: Hierarchical New Skyline)*** *Let $\wp$ be a preference defined on D, $Sky(D,E)_{\wp}$ its associated skyline, $\wp'$ a direct specialization of $\wp$, and $Sky(D,E)_{\wp'}$ its associated skyline. By definition: $HNSky(D,E)_{(\wp,\wp')} = \{p \in Sky(D,E)_{\wp} | p \notin Sky(D,E)_{\wp'}\}$.*

Example 5. *Let $\wp = (\{(BRN, EPT)\})^H \bigcup \wp_{Sn} \bigcup \wp^0_{Re}$ and $\wp' = \wp \bigcup \{(Yar, VLN)\}$ be two preferences defined on space D. $\wp'$ is a direct specialization of $\wp^0$. $Sky(D,E)_{\wp} = \{a,b,e,c,d,f\}$ and $Sky(D,E)_{\wp'} = \{a,b,d,e,f\}$. Consequently, $HNSky(D,E)_{(\wp,\wp')} = \{c\}$.*

The computation of $HNSky(D,E)_{(\wp,\wp')}$ does not require to compute the whole skyline associated with $\wp'$ but only to verify that the skyline points associated with $\wp$ remains skyline for $\wp'$. When specializing preference $\wp$ into $\wp'$, the skyline associated with $\wp'$ may be obtained by removing from the skyline associated with $\wp$ the points disqualified by the specialization (i.e. the set $HNSky$).

We want to build a memorization data structure $HSky$ for storing efficiently the pre-computed information. Our goal is to avoid computing and storing all the

skyline points associated with every possible hierarchical preference defined on D. $HSky$ is a graph data structure whose nodes represent preferences and whose arcs connecting a child node associated with some preference $\wp'$ to a parent node associated with preference $\wp$ is labeled by $HNSky(D,E)_{(\wp,\wp')}$.

Once the base preference $\wp^0$ is chosen, the $HSky$ building process navigates in the specializations and the generalizations of its associated node:

- generate all the direct and indirect specializations and generalizations of $\wp^0$ and build the preference hierarchy. This gives the general structure of $HSky$. The node at the top of the structure denotes the empty preference $\varnothing$,
- for each arc of $HSky$, compute the set $HNSky$ that stores the disqualified points (resp. introduced) when going from a preference to a direct specialization (resp. generalization) of this preference.

In practice, these two operations are performed simultaneously. Thanks to property 1, every point in $Sky(D,E)_{\wp^0}$ belongs to the skyline associated with each parent node (generalization) of $\wp^0$. Consequently, to compute the set $HNSky$ associated with an arc going from node $\wp^0$ to one of its generalizations N, it is

Algorithm 1. $HSky(\wp^0, \wp, Sky(D,E)_{\wp^0})$

input : $\wp^0$: base preference associated with start node, $Sky(D,E)_{\wp^0}$: skyline associated with preference $\wp^0$, $\wp$: preference associated with target node
output: $Sky(D,E)_{\wp}$: skyline associated with $\wp$

```
1  Sky ← Sky(D,E)_{℘^0}
2  if ℘ ⊂_h ℘^0 // Test whether ℘ is a generalization of ℘^0
3  then
4      foreach parent node ℘^1 of ℘^0 do
5          if ℘ = ℘^1 then
6              Sky ← Sky(D,E)_{℘^0} ⋃ HNSky(D,E)_{(℘^1,℘^0)}
7              Sky(D,E)_{℘} ← Sky
8          else
9              if ℘ ⊂_h ℘^1 // Test whether ℘ is a generalization of ℘^1
10             then
11                 Sky ← Sky(D,E)_{℘^0} ⋃ HNSky(D,E)_{(℘^1,℘^0)}
12                 HSky(℘^1, ℘, Sky)
13                 Exit
14 if ℘^0 ⊂_h ℘ // Test whether ℘ is a specialization of ℘^0
15 then
16     foreach child node ℘^1 of ℘^0 do
17         if ℘ = ℘^1 then
18             Sky ← Sky(D,E)_{℘^0} - HNSky(D,E)_{(℘^0,℘^1)}
19             Sky(D,E)_{℘} ← Sky
20         else
21             if ℘^1 ⊂_h ℘ // Test whether ℘ is a specialization of ℘^1
22             then
23                 Sky ← Sky(D,E)_{℘^0} - HNSky(D,E)_{(℘^0,℘^1)}
24                 HSky(℘^1, ℘, Sky)
25                 Exit
26 Return Sky(D,E)_{℘}
```

sufficient to test the dominance of points that does not belong to the sets $Sky(D, E)_{\wp^0}$ or to the sets $HNSky$ associated with arcs targeting node N. Similarly, to compute the set $HNSky$ associated with any specialization of preference $\wp^0$, only points belonging to $Sky(D, E)_{\wp^0}$ must be tested. Figure 3 gives the structure $HSky$ of the running example.

Query Evaluation. In traditional OLAP, the *drill-down* and *roll-up* operators are applied on dimension hierarchies. In our approach, these operators are applied on hierarchical preferences associated with skyline queries. The data structure $HSky$ helps to reduce the runtime computation of skyline points associated with hierarchical preferences, which enhances interactivity. Below, the $HSky$ structure depicted in Figure 3 is used to illustrate a navigation from the base preference $\wp^0 = (\{(BRN, EPT)\})^H \bigcup \wp^0_{Sn} \bigcup \wp^0_{Re}$ and the computation of the related skylines. The skyline associated with $\wp^0$ is $Sky(D, E)_{\wp^0} = \{a, b, e, c, d, f\}$. We show how to use the structure $HSky$ for computing the skyline points associated with the preferences $\wp' = \wp^0 \bigcup \{(LR,GL),(Yar,VLN)\}$.

Skyline of a Specialized Preference. The skyline associated with preference $\wp'$, a specialization of the base preference $\wp^0$, is computed as follows. The search starts from the node associated with $\wp^0$ and explores recursively its children node depth-first, looking for the node associated with $\wp'$ (cf. algorithm 1 line 16). When the searched node is reached, $Sky(D, E)_{\wp'}$ is computed by subtracting from $Sky(D, E)_{\wp^0}$ each set $HNSky$ labeling an arc of the path going from $\wp^0$ to

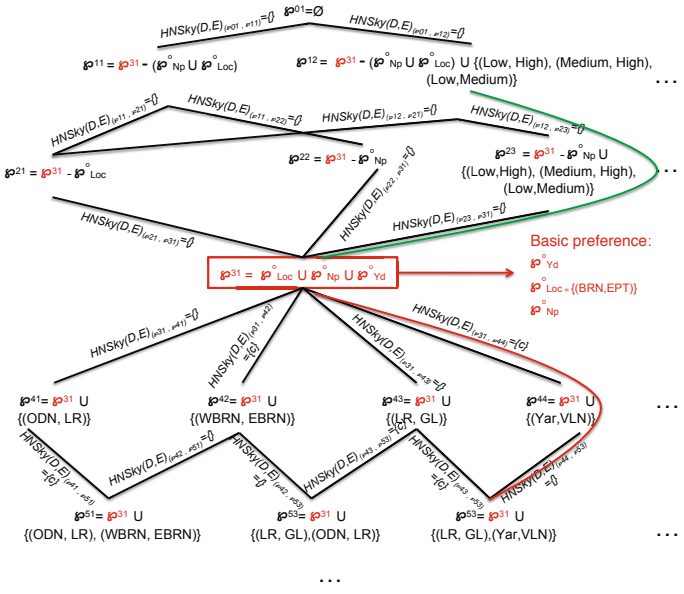


Fig. 3. The specialization / generalization data structure $HSky$

$\wp'$ (Algorithm 1, lines from 21 to 24). For the sake of efficiency, the search only explores nodes associated with some generalization of preference $\wp'$ (Algorithm 1, line 21). This pruning insures that the path going from the node associated with $\wp^0$ to the node associated with $\wp'$ has a minimal length.

Example 6. $\wp^0 = (\{(BRN, EPT)\})^H \bigcup \wp^0_{Sn} \bigcup \wp^0_{Re}$. *The skyline related to* $\wp^0$ *is* $Sky(D,E)_{\wp^0} = \{a, b, e, c, d, f\}$. *Let* $\wp' = \wp^0 \bigcup \{(LR,GL), (Yar,VLN)\}$ *be a specialization of* $\wp^0$. *In Figure 3, the path from the start node* $\wp^0 \equiv \wp^{31}$ *to the target node* $\wp' \equiv \wp^{53}$, *whose skyline must be computed, is colored in red.*

$Sky(D,E)_{\wp'} = Sky(D,E)_{\wp^{31}}$ - $(HNSky(D,E)_{(\wp^{31},\wp^{44})} \bigcup HNSky(D,E)_{(\wp^{44},\wp^{53})}) = \{a, b, c, e, d, f\}$ -$(\{c\} \bigcup \{\}) = \{a, b, e, d, f\}$.

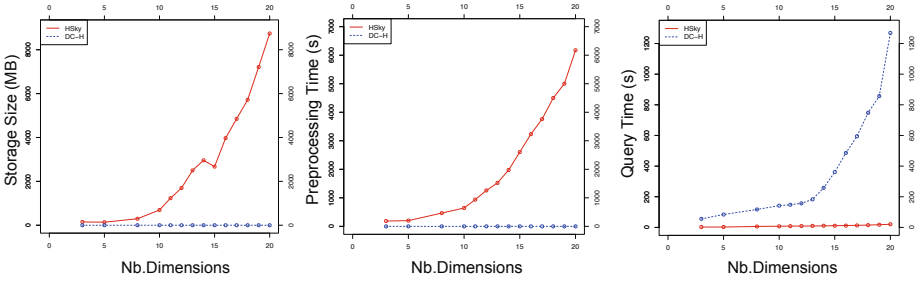
The query evaluation of a generalization is symmetric to the specialization.

4 Experiments

In this section, we present an empirical evaluation of algorithm *HSky* on synthetic data. *HSky* is implemented in *JAVA*. The experiments were performed on an Intel Xeon CPU 3GHz with 16 GB de RAM under Linux. Data related to dimensions with only one hierarchical level were produced by the generator presented in [1]. Three kinds of datasets were generated: independent data, correlated data, non-correlated data. A detailed description of these datasets can be found in [1]. We only give the results concerning non-correlated data. The results on other datasets were similar, whereas the results of pre-computing and query answering are much shorter for correlated data. Data for hierarchical dimensions were generated with respect to a Zipfian distribution [10]. By default, the Zipfian parameter θ was initialized to 1 (non-correlated data). This yielded 700.000 tuples for 6 dimensions with one hierarchical level. We imposed the number of hierarchical dimensions vary from 3 to 20 and the number of hierarchical levels of these dimensions vary from 3 to 7. The base preference was chosen to yield a balanced number of specializations and generalizations. The query preference template was such that the preference represents an indirect specialization / generalization of the base preference.

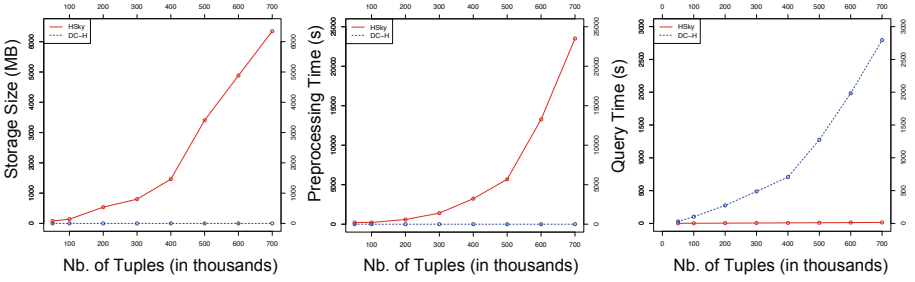
To the best of our knowledge, there does not exist a work on skyline extraction within hierarchical dimensions in the literature. Thus, we have implemented algorithm *DC-H* that computes skyline on the *Divide & Conquer* (*D&C*) principle [1]. *DC-H* does not materialize any partial result and re-computes all the skyline sets at every hierarchical level. As *D&C* can be encoded naturally in parallel, we have parallelized *DC-H* to improve its performance. Algorithm *HSky* is also based on the *DC-H* principle.

Varying the Number of Hierarchical Dimensions. In these experiments, the number of one level dimensions is set to 6 and the number of hierarchical dimensions varies from 3 to 20. Figure 4 shows that the memory size and the pre-computation runtime of *HSky* rises with the number of hierarchical dimensions. This is related to the complexity of the data structure *HSky* (quadratic in the number of hierarchical dimensions). Algorithm *DC-H* does not need any memory storage, nor pre-computation runtime since it does not store any partial result.



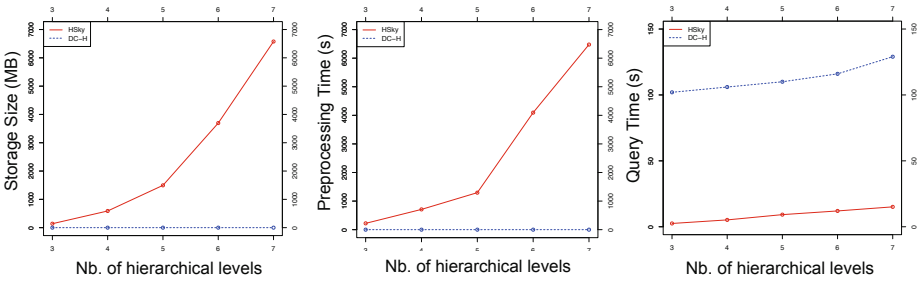
(a) (b) (c)

Fig. 4. Varying the number of hierarchical dimensions



(a) (b) (c)

Fig. 5. Varying the size of datasets



(a) (b) (c)

Fig. 6. Varying the number of levels in hierarchical dimension

Varying the Size of the Dataset. In these experiments, the number of tuples in the dataset varies from 50.000 to 700.000. Figure 5 shows that the memory size and the pre-computation runtime of *HSky* rises with the dataset size as well. This is due to the size of skyline sets which rises polynomially with the dataset size.

Varying the Number of Levels in Hierarchical Dimensions. In these experiments, the number of levels in hierarchical dimensions varies from 3 to 7. Figure 6 shows that the memory size and the pre-computation runtime of $HSky$ rises also with the number of hierarchical levels. This is due to the volume of hierarchical dimensions which rise exponentially with the number of hierarchical levels. However, one should note that experiments were notably complex since it is not common to have to analyze dimensions with 7 hierarchical levels in real applications.

However, for each of these experiments, $HSky$ outperforms *DC-H* for skyline query answering (Figures 4c, 5c and 6c). $HSky$ response times are quasi-instantaneous. In fact, for each new query, *DC-H* re-computes the whole skyline whereas $HSky$ deduces it using simple set operations. it should be noted that the query response time is an important criterion in the context of online analysis.

The construction of the data structure $HSky$ impacts the global runtime and memory storage. To obtain some benefit, on average, from 6 to 8 navigations from the same base preference must be performed.

5 Conclusion

This article proposes a new method, called $HSky$, which extends skyline points extraction to hierarchical dimensions. Coping with hierarchies on dimensions enables to specialize or generalize user preferences when computing skylines. Hierarchical relations between preferences on dimensions were formulated and some interesting properties were explicited. These properties, e.g. the hierarchical monotonicity property, are exploited in a data structure called $HSky$ for designing an efficient navigation tool along the preferences hierarchy while ensuring an online computation of skylines. The experiments underline the online performance of $HSky$ compared to *DC-H* (an algorithm for computing skylines with no materialization based on [1]) when the number of queries performed during navigation is greater than some threshold (6 to 8 queries on average).

This structure could evolve with respect to the application context. For example, if the size of the structure were too high it could be possible to restrict the specializations / generalizations to values explicitly ordered in the base preference. We plan to extend the hierarchical dimensions presented in this paper to the case of dynamic preferences, as introduced in [9, 11]. In this setting, the dynamical aspect of preferences may lead to an explosion of the preferences to be materialized. A potential solution would be to restrict the preferences to the class of so-called *nth-order* preferences [9].

References

[1] Borzsonyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proc of the 17th Int. Conf. on Data Engineering, pp. 421–430. IEEE Computer Society (2001)
[2] Raïssi, C., Pei, J., Kister, T.: Computing closed skycubes. Proc. VLDB Endow., 838–847 (2010)
[3] Golfarelli, M., Rizzi, S., Biondi, P.: myolap: An approach to express and evaluate olap preferences. IEEE Trans. Knowl. Data Eng. 23(7), 1050–1064 (2011)

[4] Golfarelli, M., Rizzi, S.: Expressing OLAP preferences. In: Winslett, M. (ed.) SSDBM 2009. LNCS, vol. 5566, pp. 83–91. Springer, Heidelberg (2009)
[5] Antony, S., Wu, P., Agrawal, D., Abbadi, A.E.: Aggregate skyline: Analysis for online users. In: Proceedings of the 2009 Ninth Annual International Symposium on Applications and the Internet, pp. 50–56. IEEE Computer Society (2009)
[6] Antony, S., Wu, P., Agrawal, D., El Abbadi, A.: Moolap: Towards multi-objective olap. In: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, pp. 1394–1396. IEEE Computer Society (2008)
[7] Jin, W., Ester, M., Hu, Z., Han, J.: The multi-relational skyline operator. In: ICDE, pp. 1276–1280 (2007)
[8] Magnani, M., Assent, I.: From stars to galaxies: skyline queries on aggregate data. In: EDBT, pp. 477–488 (2013)
[9] Bouadi, T., Cordier, M.O., Quiniou, R.: Computing skyline incrementally in response to online preference modification. T. Large-Scale Data- and Knowledge-Centered Systems 10, 34–59 (2013)
[10] Trenkler, G.: Univariate discrete distributions: N.L. Johnson, S. Kotz and A.W. Kemp, 2nd edn. John Wiley, New York (1992) ISBN 0-471-54897-9; Computational Statistics & Data Analysis, pp. 240–241 (1994)
[11] Bouadi, T., Cordier, M.O., Quiniou, R.: Incremental computation of skyline queries with dynamic preferences. In: Liddle, S.W., Schewe, K.-D., Tjoa, A.M., Zhou, X. (eds.) DEXA 2012, Part I. LNCS, vol. 7446, pp. 219–233. Springer, Heidelberg (2012)