Malleability-Aware Skyline Computation on Linked Open Data

Christoph Lofi[1], Ulrich Güntzer[2], Wolf-Tilo Balke[1]

[1]Institut für Informationssysteme
Technische Universität Braunschweig, 38106 Braunschweig, Germany
{lofi, balke}@ifis.cs.tu-bs.de

[2]Institut für Informatik
Universität Tübingen, 72076 Tübingen, Germany
ulrich.guentzer@informatik.uni-tuebingen.de

Abstract. In recent years, the skyline query paradigm has been established as a reliable and efficient method for database query personalization. While early efficiency problems have been approached, new challenges in its effectiveness continuously arise. Especially, the rise of the Semantic Web and linked open data leads to personalization issues where skyline queries cannot be applied easily. In fact, the special challenges presented by linked open data establish the need for a new definition of object dominance that is able to cope with the lack of strict schema definitions. However, this new view on dominance in turn has serious implications on the efficiency of the actual skyline computation, since transitivity of the dominance relationships is no longer granted. Therefore, our contributions in this paper can be summarized as a) we design a novel, yet intuitive skyline query paradigm to deal with linked open data b) we provide an effective dominance definition and establish its theoretical properties c) we develop innovative skyline algorithms to deal with the resulting challenges and extensively evaluate the our new algorithms with respect to performance and the enriched skyline semantics.

Keywords: Query Processing; Personalization; Skyline Queries; Linked Open Data.

1 Introduction

The continuous efforts to put the Semantic Web vision into practice have led to two important insights: implementing a full-fledged machine understandable Web has largely failed, but focusing only on the 'reasonable' part already reveals a vast variety of valuable data [1]. This area of so-called linked open data (LOD) [2] has immediately spawned interesting efforts like for instance the DBPedia knowledge base[1] that currently describes more than 3.64 million things, out of which 1.83 million are classified in a consistent ontology. Moreover, the potential applications also promoted the development of innovative methods to make such data available to users in a struc-

[1] http://www.dbpedia.org/About

tured way. IR-style or rule-based extraction frameworks like ALICE [3], Xlog [4], or SOFIE [5] can already crawl the Web and extract structured relationships from unstructured data with largely sufficient accuracy.

However, when it comes to retrieval of the now structured information also the typical query paradigms have to be adapted. This is not only because extracted knowledge is usually represented in some form of knowledge representation language (with RDF triples as most prominent example), but also due to the semantic loss of focus that results from ambiguities in the extraction process. For instance when querying for a person's *place of birth*, the information where somebody *grew up* is generally heavily related, but definitely less focused regarding the original query intention. Still, whenever the exact place of birth is unknown, the information where a person grew up is still much more helpful than an empty result set. Thus, it should be retrieved as relevant, but of course should always get a penalty in the ranking. This desirable facet of retrieval is known as *schema malleability* [6, 7].

While current retrieval paradigms for example in SOFIE's retrieval engine NAGA [8] or Xlog's DBLife [9], only focus on SQL-style retrieval (usually SPARQL over RDF) and keyword search with top-k ranking, the problem of preference-based retrieval paradigms like skyline queries over linked open data has not yet been solved. In this paper we tackle the problem of *malleability-aware skyline queries* over linked open data. The problem is twofold: first a viable semantics has to be defined trading a user's value preferences against the extracted relationships' loss of focus with respect to the original query, then efficient algorithm(s) have to be designed to solve the retrieval task in practical runtimes.

In a nutshell the problem is the intuitive interleaving of each individual user's attribute value preferences with the generally applicable preferences on attribute semantics as specified in the query. Whereas skyline queries up to now only dealt with relaxing value preferences, the new additional relaxation in attribute semantics is owed to the linked open data. Let's extend our example from above:

Example: A user might be interested in famous Nobel laureates in physics that were *born* in Munich, Germany. Querying the DBPedia knowledge base retrieves only two entries: Rudolf Mössbauer and Arno Allan Penzias. However, a similar query for Nobel laureates in physics *growing up* in Munich also retrieves Werner Heisenberg (who went to school in Munich) and a further relaxation to Nobel laureates in physics *living in* Munich finally retrieves Wilhelm Conrad Röntgen. With a different degree of relevance (with respect to famousness and having a relationship with Munich) all these are possible answers that, however, with respect to the original query are getting less focused and thus should be displayed accordingly. That means the final result including schema malleability may be a trade-off between the famousness of the physicists and their relationship to Munich, which is best represented by a skyline query result.

To model this paradigm in databases (and schema malleability as such), each query attribute can be considered as a database column holding not only tuples based on the strict relationships given by the query, but also tuples from semantic similar relationships. However, to prepare for later retrieval each such malleable attribute has to be associated with a second attribute measuring the semantic loss of focus for each tuple.

This can be done by either automatically measuring semantic loss of focus by instance-based precision/recall tests like shown in [10], testing the relationships' semantic relatedness with externally available ontologies like in [11], or simply denoting possible relationships and allowing users to define a (partial) order over these relationships with respect to their queries.

In any case, the new associated attribute columns have to be considered by retrieval algorithms, but in contrast to the attribute value columns have a slightly different quality. This is because relaxations on preferred *values* for cooperative query processing might change a tuple's desirability, but larger relaxations in attribute *senses* might render tuples utterly useless. Consider the example above where a Nobel laureate's *place of birth* is relaxed in terms of the preferred *value*, e.g. from 'Munich' to 'Bavaria' or 'Germany', or in terms of the *relationship with Munich* e.g., from 'born in' to 'lived in'. Whether a broader relaxation of the sense like 'visited' is still of any use is doubtful. Thus, classical skyline query processing following Pareto-optimality cannot readily be applied. Moreover, by basically doubling the problem dimensionality also the well-known efficiency problems of skyline processing in terms of runtimes and result set manageability, see e.g., [12], are bound to be encountered.

The contribution of this paper is threefold: we design an intuitive notion of skyline dominance with respect to malleability in the form of semantically typed links in linked open data and discuss its characteristics. We develop innovative algorithms to efficiently process skyline queries even over large data repositories. And we extensively evaluate these algorithms with respect to runtime behavior and skyline manageability. In fact, our experiments show that in the general case, our algorithm can achieve significant performance improvements over the baseline. However, when slightly restricting general malleability, we can even show that performance indeed can increased by several orders of magnitude, even rivaling the runtime behavior of classical skyline algorithms over strictly transitive preferences.

This paper is structured as follows: after briefly surveying related work in section 2, we discuss the necessary foundations and theoretical characteristics of skylines over linked open data in section 3. Section 4 then presents and evaluates skyline algorithms over several malleability attributes, whereas section 5 deals with the special case of a single aggregated malleability attribute. We close with a short summary and outlook.

2 Related Work

Due to its potential usefulness linked open data has received a lot of attention and even inspired a taskforce[2] of the World Wide Web Consortium (W3C). Current research is often focused on the area of business intelligence, but also for the collection of common knowledge. The basic idea is using the Web to create typed links between data items from different sources. Once extracted, these links represent semantic relationships which can in turn be exploited for querying. However when querying (or reasoning over) such relationships the exact nature of the relationship and its semantic

[2] http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

correspondence to the query is often difficult. Therefore apart from typical exact match queries (usually performed in SPARQL[3] over RDF triples) many approaches for ranking the best matching information have been designed.

The first notable approach to rank queries on extracted entity properties was EntitySearch [13] proposing an elaborate ranking model combining keywords and structured attributes. When it comes to exploiting also semantic relationships NAGA [8] used a scoring model based on the principles of generative language models, from which measures such as confidence, informativeness, and compactness are derived, which are subsequently used to rank query results. Finally [14] develop a general model for supporting approximate queries on graph-modeled data, with respect to both attribute values and semantic relationships and derive a first top-k algorithm to implement the ranking efficiently. However, like in all top-k frameworks even far-fetched semantic relationships can be compensated for by good matching attribute values. Moreover, all these approaches directly work on graph-structured data relying on the path-based semantic relatedness e.g., defined by [15], whereas our approach works on relationship malleability quantifying the respective loss of focus.

To our knowledge the only algorithm similar to skyline queries on linked open data is given by [16]. However, the developed algorithm has been designed for optimizing skyline queries over RDF data stored using a vertically partitioned schema model and thus presents an efficient scheme to interleave the skyline operator with joins over multiple relational tables. Unfortunately it does not offer any techniques with respect to personalization and the problem of semantic linkage and thus is not really related to our work here. In brief, to our knowledge our approach features the first skyline algorithm respecting semantic malleability.

3 Theoretical Foundations of Malleability-Aware Skylines

In the following we will briefly revisit the notion of Pareto skylines as given by [17]. Assume a database relation $R \subseteq D_1 \times \ldots \times D_n$ on n attributes.

- A preference P_i on some attribute A_i with domain D_i is a strict partial order over D_i. If some attribute value $a \in D_i$ is *preferred over* another value $b \in D_i$, then $(a, b) \in P_i$, written as $a >_i b$ (read "a dominates b wrt. to P_i"). The set of all preferences is denoted as P.
- Analogously, also an equivalence relation on D_i compatible with P_i can be defined. Then, two attribute values attribute $a, b \in D_i$ can be defined as being equivalent with respect to the domain: $a \approx_i b$. Moreover, if some attribute value $a \in D_i$ is either *preferred over or equivalent to* some value $b \in D_i$, we write $a \gtrsim_i b$.

Assuming preferences $P_1, \ldots, P_n$ for each attribute in R, the concept of Pareto dominance between two tuples $\vec{x}, \vec{y} \in R$ with $\vec{x} = (x_1, \ldots, x_n)$ can be defined as:

Definition 1: Pareto Dominance

$$(\vec{x} >_P \vec{y}) := \forall_{i \in D}(x_i \gtrsim_i y_i) \wedge \exists_{i \in D}(x_i >_i y_i)$$

[3] http://www.w3.org/TR/rdf-sparql-query/

The classical *skyline set* [18] can now be defined as all those tuples in each database instance, which are not dominated by any other tuple:

Definition 2: Pareto Skyline for some relation ***R*** and preferences ***P***

$$sky(R,P) \coloneqq \{\vec{x} \in R \mid \neg\exists\, \vec{y} \in R : \vec{y} >_P \vec{x}\}$$

Now we are ready to extend the semantics by introducing the concept of *malleability-aware dominance*, which specifically respects the semantic challenges introduced by linked open data entities. Like motivated above the intuition is that regarding each queried attribute personalized skyline queries consist of a user-specific value preference and a certain meaning of the attribute that may more or less correspond to some number of extracted attribute types in the database instance. Thus, for getting acceptable results, not only the entities' attribute values, but also the loss of focus with respect to each attribute's semantics has to be taken into account. The baseline approach for this would be to simply compute skylines with the double dimensionality (one attribute value and a malleability score for each attribute).

However, apart from the obvious scalability problems, also the semantics are unclear. Whereas attribute values like dates, prices, or ratings are usually crisp and follow a certain preference order (users want the cheapest price for some product or the highest quality rating), labels for semantic relationships are usually fuzzy and to some degree ambiguous depending on their labels. Often *grew_up_in* may be used synonymously with *born_in*, but *lived_in* definitely is not. Thus, the *relative loss of focus* (or δ-distance) between semantic labels needs to be considered: if two labels are at most differing by δ they should be considered semantically equivalent, but once two labels are too far apart a different class of semantic relationship has to be assumed.

Definition 3: *δ*-preferences for modeling malleability over linked open data

A δ-preference δP_i on some attribute A_i with metric domain D_i and metric $dist_i(.,.)$ is a reflexive and transitive binary relation $\succ_i$ over D_i, together with an *intransitive* form of equivalence with the notion of indifference: for all $a, b \in D_i$: $a \approx_i b \Leftrightarrow dist_i(a,b) \leq \delta$ see e.g [19]. If some attribute value $a \in D_i$ is *preferred over* another value $b \in D_i$ and $dist_i(a,b) > \delta$ we write $a \succ_i b$ (read "a strictly δ-dominates b wrt. to δP_i"). The combination of several δP_i can easily be achieved using the normal Pareto product and will be denoted as $\succ_{\delta P}$. Likewise, we write $a \succcurlyeq_i b$ if either $a \approx_i b$ or $a \succ_i b$.

It is easy to mix δ-preferences and normal strict partial order preferences to create a product preference over some relation (which for ease of use we will again simply denote by '>') and we will define the respective domination relationships for malleability-aware skylines in section 4 and 5. But up to now such δ-preferences with relative distances have not been considered in skyline queries, because their use directly contradicts the generally assumed *transitivity* of domination relationships between tuples. Actually, since long results in psychology show that in contrast to common belief intransitivity often occurs in a person's system of values or preferences, potentially leading to unresolvable conflicts, see e.g., [20] or [21]. Analogously, in economics intransitivity may occur in a consumer's preferences. While this may lead to consumer behavior that does not conform to perfect economic rationality, in recent years economists have questioned whether violations of transitivity must necessarily lead to 'irrational' behavior, see for instance [22].

Indeed, from an order-theoretical point of view it is easy to show that whenever δ-distances are used in at least one preference and $(\vec{x} >_P \vec{y})$ and $(\vec{y} >_P \vec{z})$ are given, $(\vec{x} >_P \vec{z})$ does not necessarily follow:

***Lemma 1: Dominance relationships are not transitive using $\boldsymbol{\delta}$*-distances.**
Proof: Transitivity for dominance regarding any product preference P is violated, if three tuples $\vec{x}$, $\vec{y}$, and $\vec{z}$ can be constructed, for which holds: $(\vec{x} >_P \vec{y} \wedge \vec{y} >_P \vec{z})$, but $\vec{x} \ngtr_P \vec{z}$.
Assume a product preference P over some relation R and assume there is one attribute m for which a δ-preference δP_m is declared stating the equivalence of values within the relative distance of some fixed δ. Now define preference P^ by removing δP_m from P and construct three tuples $\vec{x}$, $\vec{y}$, and $\vec{z}$ such that $(\vec{x} >_{P^\wedge} \vec{y} >_{P^\wedge} \vec{z})$. Now assign values of $\vec{x}$, $\vec{y}$, and $\vec{z}$ for attribute m as follows $y_m := (\, x_m + \delta)$ and $z_m := (\, y_m + \delta)$. Then with respect to P holds $\vec{x} >_P \vec{y}$ because of $(\vec{x} >_{P^\wedge} \vec{y})$ and x_m and y_m are equivalent with respect to the chosen δ. Analogously holds $\vec{y} >_P \vec{z}$. However, $\vec{x} \ngtr_P \vec{z}$ because of $\vec{x} >_{P^\wedge} \vec{z}$, but $(z_m = (\, x_m + 2\delta) \succ_{\delta Pm} x_m)$. Hence $\vec{x}$ and $\vec{z}$ are incomparable with respect to P and the domination relationship is not transitive. □

While the resulting preference orders are not transitive, at the same time domination relationships within the intransitive product order are sensible, since there can never exist any cyclic base preferences. However, this is only the case when strict partial-order preferences and δ-preferences are used conjointly to build the product order, product orders built only from δ-preferences will inevitably lead to cycles. In order to guarantee acyclic product orders, some observations can be made: a) no cycles can ever emerge between tuples showing dominance with respect to any attributes, over which a strict partial-order preference is defined (due to their guaranteed transitivity), b) cycles can only occur, if tuples are equivalent with respect to all partial-order preferences. In this case, *strict* δ-dominance ($\succ$) must be enforced, and none of the tuples are allowed to dominate by simple δ-dominance alone ($\succcurlyeq$). This leads to our formal definition of malleability-aware dominance (Definition 4) in Section 4.1.

3.1 Implications for Algorithm Design

The danger on intransitivity of dominance relationships is that it may lead to *non-deterministic behavior* when computing skylines using standard skyline algorithms. According to Definition 2, the skyline contains all tuples of a given relation which are not dominated by any other tuple, assuming that preferences are partial orders. Naïvely, this would need an algorithm pairwise comparing all tuples with respect to the chosen dominance criterion. In practice however, most skyline algorithms increase efficiency by pruning large numbers of tuple comparisons (e.g. basic block-nested-loop (BNL) algorithms [18], branch-and-bound algorithms [23], distributed algorithms [24], or online algorithms [25]). These optimizations usually all rely on the transitivity of dominance.

Example: When using non transitive dominance with for instance a BNL algorithm the result will vary *non-deterministically* depending on the order of the tuples in the database instance (and therefore, also the order of the tests for dominance). For example, when assuming $(\vec{x} >_P \vec{y})$, $(\vec{y} >_P \vec{z})$, but $(\vec{x} \not>_P \vec{z})$, then a skyline computed by some BNL algorithm just contains $\{\vec{x}\}$, if the test for $(\vec{y} >_P \vec{z})$ is performed first and thus $\vec{z}$ is immediately pruned from the database. Otherwise, if $(\vec{x} >_P \vec{y})$ is tested first, the resulting skyline contains $\{\vec{x}, \vec{z}\}$, because $\vec{y}$ is removed prematurely before also $\vec{z}$ could be removed by testing $(\vec{y} >_P \vec{z})$; and due to $(\vec{x} \not>_P \vec{z})$, $\vec{z}$ incorrectly remains in the skyline set.

However, the idea of skylines is still sensible since as we will prove in lemma 2, cyclic preferences cannot occur and thus a skyline based on the notion of containing all non-dominated objects can be computed. Since pruning may cause difficulties, the obvious way is by simply comparing all tuples in the database instance pairwise (with quadratic runtime). But as we will see in the next section, far more efficient algorithms can be designed and thus skylines over linked open data are indeed practical.

4 Malleability-aware Skylines

Before delving into designing skyline algorithms capable of dealing with intransitivity as described above, we have to formalize our concept of product orders also built from δ-preferences in form of a dominance criterion usable in skyline algorithms.

4.1 Malleability-aware Skylines with Individual Attribute Malleability

Assuming preferences P that can be decomposed into strict partial-order preferences $P^{\wedge}$, and δ-preferences δP, the concept of malleability-aware dominance between two tuples $\vec{x}, \vec{y}$ can be defined as

Definition 4: Malleability-aware dominance over individual attributes

$$(\vec{x} >_P \vec{y}) :\Leftrightarrow \big((\vec{x} >_{P^{\wedge}} \vec{y}) \wedge (\vec{x} \succcurlyeq_{\delta P} \vec{y})\big) \vee \big((\vec{x} \approx_{P^{\wedge}} \vec{y}) \wedge (\vec{x} \succ_{\delta P} \vec{y})\big)$$

In this definition, there is a malleability-aware dominance *a)* if all non-malleable attribute values of $\vec{x}$ show Pareto dominance over $\vec{y}$ and all malleable attributes of $\vec{x}$ are at least equivalent to those of $\vec{y}$ with respect to the δ-preferences (i.e. all malleable attributes encoding the tuple's loss-of-focus are tested for "soft" dominance here, allowing a certain δ of flexibility) or *b)* if all data attributes are equivalent with respect to the Pareto preferences, but show strict dominance with respect to the malleable attributes for the δ-preferences (this means: all malleable attributes encoding loss-of-focus have to show real Pareto dominance, i.e. no δ-distances are considered. This important property is required to prevent cycles to form in P:

***Lemma 2:* Product orders of strict partial order preferences and δ-preferences following Definition 4 cannot contain cyclic preferences.**

Proof: We have to show that the dominance relation of the product order does not induce cycles, more precisely, if $\overrightarrow{x_1} >_P \ldots >_P \overrightarrow{x_k}$ with $k > 1$, then neither $\overrightarrow{x_k} >_P \overrightarrow{x_1}$ nor $\overrightarrow{x_k} \approx_P \overrightarrow{x_1}$ is possible. Please note that $\vec{x} \approx_P \vec{y}$ means $\overrightarrow{x_\iota} \approx_{Pi} \overrightarrow{y_\iota}$ for all non-malleable attributes and $\overrightarrow{x_J} = \overrightarrow{y_J}$ for all malleable attributes, i.e. no malleability is allowed for equivalence.

For $1 \leq t \leq k$ let $\overrightarrow{x_t} \coloneqq (x_{t,1}, \ldots, x_{t,n}, x_{t,n+1}, \ldots, x_{t,n+m})$ where the first n attributes are non-malleable and the following m attributes are malleable. We distinguish two cases:

a) There is a strict preference in the non-malleable part between two objects in an assumed cycle, i.e. there are $1 \leq t < k$ and $1 \leq i \leq n$ such that $x_{t,i} >_{Pi} x_{t+1,i}$. Then within the cycle we have $x_{1,i} \gtrsim_{Pi} \ldots \gtrsim_{Pi} x_{t,i} >_{Pi} x_{t+1,i} \gtrsim_{Pi} \ldots \gtrsim_{Pi} x_{k,i}$ and therefore $x_{1,i} >_{Pi} x_{k,i}$, rendering both $\overrightarrow{x_k} >_P x_1$ and $\overrightarrow{x_k} \approx_P x_1$ impossible.

b) If there is no strict preference in the non-malleable part, for all $1 \leq t < k$ and $1 \leq i \leq n$ we have $x_{t,i} \approx_{Pi} x_{t+1,i}$. Thus following Definition 4 for the malleable attributes for all $1 \leq t < k$ holds: $(x_{t,n+1}, \ldots, x_{t,m}) \succ_{\delta P} (x_{t+1,n+1}, \ldots, x_{t+1,m})$ which means $(x_{1,n+1}, \ldots, x_{1,m}) \succ_{\delta P} (x_{k,n+1}, \ldots, x_{k,m})$ due to the strictness of $\succ_{\delta P}$. Hence $\overrightarrow{x_k} >_P x_1$ is impossible.
In the same way it is easy to see that also $\overrightarrow{x_k} \approx_P x_1$ is impossible. □

Now, the respective malleability-aware skyline can be computed analogously to definition 2 by

$$sky(R, P) \coloneqq \{\vec{x} \in R \mid \neg\exists\, \vec{y} \in R : \vec{y} >_P \vec{x}\}$$

Unfortunately, actually implementing such malleability-aware skyline computations algorithmically poses several challenges. Therefore, in the following we demonstrate how such algorithms can be designed. For the sake of cleaner notions and without loss of generality, we will assume that all our preferences are encoded in the database tuples by normalized scores in $[0,1]$, where 1 represents the most preferable attribute values, and 0 the least preferable ones. Any database tuple $\vec{x}$ is given by $\vec{x} = (x_1, \ldots, x_m)$, and the individual attributes can be separated into non-malleable data attributes x_i with $i \in D$ (corresponding to $P^\wedge$), and malleable attributes x_i with $i \in M$ (corresponding to δP). Then, the dominance criterion of definition 4 can be reformulated as:

$$(\vec{x} >_P \vec{y}) :\Leftrightarrow \left[(\forall_{i\in D}\, x_i \geq y_i \land \exists_{i\in D} x_i > y_i) \land \left(\forall_{i\in M} x_i \geq (y_i - \delta_i)\right)\right]$$
$$\lor\, [(\forall_{i\in D} x_i = y_i) \land (\forall_{i\in M} x_i \geq y_i \land \exists_{i\in D} x_i > y_i)]$$

It is easy to see that this definition is equivalent to the Pareto dominance as given by definition 1 for the cases of $M = \emptyset$ or $\delta = 0$. If $M \neq \emptyset$ and $\delta > 0$, then malleability-aware dominance allows for additional tuples being dominated compared to Pareto dominance, hence the resulting Skyline is a subset of the Pareto skyline.

4.2 Computing Non-Transitive Skylines

As already indicated in section 3.1, modern Skyline algorithms have come to rely on the transitivity of dominance criteria: For sake of improved performance, many tuple comparisons are avoided by pruning objects early, relying on transitivity for the computational correctness, i.e. a tuple shown to be dominated can be fully excluded from the further execution of the algorithm. However, without guaranteed transitivity, even basic algorithms like the well-known Block-Nested Loop Algorithm [18] fail. Therefore, the need arises to develop new algorithms being able to cope with these new requirements. In this section, we will therefore present a general purpose algorithm designed for use with any non-transitive dominance criteria, including dominance for malleability-aware skylines.

The naïve solution to the given problem is relying on exhaustive pair-wise comparison, i.e. each possible tuple pair has to be tested for dominance. However, this algorithm shows prohibitive practical performance requiring $½\big(n\,(n-1)\big)$ expensive tests for dominance, with n being the size of the database (and assuming that each test for dominance is bi-directional, i.e. by testing $a >_P b$, we can test $b >_P a$ at the same time).

Hence, we propose a novel algorithm which is capable of dealing with any transitive or non-transitive preferences P. Our algorithm is derived from this naïve implementation by carefully avoiding any tuples comparisons which are guaranteed to show no effect. This can be formalized as follows:
Given is a database relation R with n tuples and preferences P. Furthermore, we need the set T of all tuples which need further testing for *a)* if any $t \in T$ is dominated by any other tuple and *b)* if any $t \in T$ dominates any tuples itself; T is initialized with $T = R$. Furthermore, we use the set S of all tuples which are the final skyline, and the set L (i.e. losers) of those tuples which have already been shown to be dominated by any other tuple. In contrast Skyline algorithms with transitive dominance, we cannot exclude tuples in L from further computation without additional guarantees. This results in following algorithm:

Algorithm 1: Non-Transitive Skyline Algorithm

```
$T := R$;  $L := \emptyset$;  $S := \emptyset$;
while ($T \setminus L \neq \emptyset$) do
        choose $t \in (T \setminus L)$;
        $C := T \setminus \{t\}$;
        $failed := false$;
        while ($C \neq \emptyset$) do
                choose $c \in C$;
                if $t >_P c$ then $L := L \cup \{c\}$
                elseif $c >_P t$ then
                        $L := L \cup \{t\}$;  $failed = true$;
                $C := C \setminus \{c\}$;
                if $failed$ then $C := C \setminus L$
        If not($failed$) then $S := S \cup \{t\}$;
        $T := T \setminus \{t\}$;
```

The algorithm contains two loops, the outer one iterating t over all objects to be tested which have not already been shown to be dominated. For finding new dominance relationships, the second loop iterates c over the set C (C is initialized in each run with $T \setminus \{t\}$.) By testing t and each c for dominance, objects can be marked to be dominated by adding them the set L of all losers. As soon as t is dominated, any subsequent comparisons of t with any other tuple which has been shown to be dominated can be avoided as those yield no new information. If t was not dominated within the inner loop, it can safely be added to the skyline. Compared to the naïve approach, this algorithm saves a significant number of superfluous tuple comparisons (see evaluation in the next section).

Furthermore, this algorithm can be implemented efficiently by representing the membership of a tuple in the different sets by simple flags attached to the tuples in R, thus minimizing the overhead of additional book-keeping.

5 Evaluations

5.1 Evaluating general malleability-aware Skylines

In this section, we evaluate the effects of malleability-aware dominance respecting any number of malleable attributes on the properties of skylines. Furthermore, we will also measure the performance of respective skyline algorithms.

Skyline Size: For the first set of experiments, we examined the impact malleability-aware dominance (represented by varying values δ) on the Skyline size. For this purpose, we relied on syntetic data, and in each experimental run generated new database tuples with 12 independently-distributed numeric attributes. Six of these attrib-

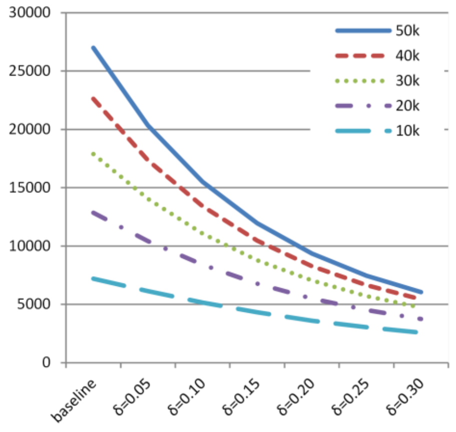

Fig. 1. Skyline Size wrt. to δ
using 6 malleable and 6 non-malleable attributes and varying database sizes; y-axis shows skyline size

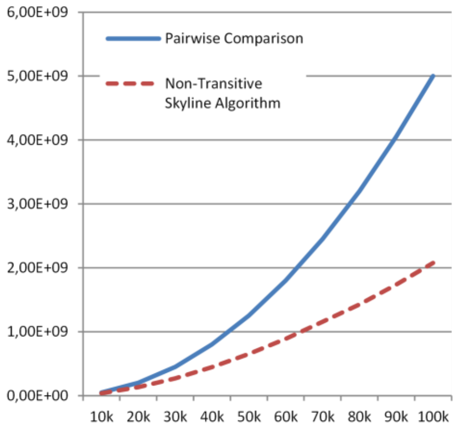

Fig. 2. Performance
using 6 malleable and 6 non-malleable attributes; x-axis shows #tuples in database; y-axis shows number of required tests for dominance; $\boldsymbol{\delta = 0.15}$

utes represent non-malleable (data) attributes, while the other six attributes are malleable ones representing loss-of-focus. Using the operationalized dominance criterion of Section 4.1, skylines are computed for δ-values ranging from $\delta = 0$ (the baseline; equivalent to Pareto skylines as in definition 2) to $\delta = 0.3$. For each value of δ, the experiment is repeated 50 times with newly generated tuples (to ensure comparability, the same random seed is used for each δ, resulting in the same sequence of generated tuples). The averaged results are shown in Fig. 1. It is clearly obvious that the skyline resulting from the baseline ($\delta = 0$, identical to Pareto skyline of the same data) is not practically manageable: from the 50,000 database tuples, 26,981 are contained in the skyline (53%). This can be attributed to high dimensionally of $d = 12$. But with growing δ, the skyline sizes dramatically decrease: already with $\delta = 0.15$, the skyline is reduced to 11,959 tuples in average - a clearly more manageable result. Similar behavior can also be observed for smaller database sizes. Therefore, we can conclude that malleability-aware skyline indeed efficiently address the issue of overly large skylines when considering on malleable loss-of-focus attribute per data attribute.

Performance of Algorithms: In the second set of experiments, we examined the performance of the naïve baseline and our non-transitive skyline algorithm (measured in the required number of tests for dominance). Similar to the last experiment, we again relied on synthetic data with 12 independent-attributes (6 malleable, 6 non-malleable), and incrementally increased the size n of the database from 10,000 tuples up to 100,000 tuples. The results are shown in Fig. 2: Clearly, our non-transitive algorithm shows significantly better performance than the baseline using pairwise comparisons. Furthermore, this performance advantage increases with growing database sizes. But still, the total time required by both algorithms is quite high (272 seconds with n= $100k$ using our non-transitive skyline algorithm vs. 637 seconds for pairwise comparisons; tests performed on a 1.86 GHz Dual-Core CPU, using Java 6 and just a single core.) Therefore, additional optimizations must be found for application domains with tighter time constraints.

5.2 Malleability-aware Skylines with a single malleable attribute

As demonstrated in the last section, the runtime of general non-transitive skyline algorithms with one malleable loss-of-focus attribute for each non-malleable data attribute can be quite high. Thus, for time-critical applications, we suggest reducing the number of malleable attributes to using just a single attribute. This single attribute then represents the overall loss-of-focus of a given database tuple with respect to the query in an aggregated form. This reduction can be implemented by different methods: a) by combining multiple malleable attributes by some combining function or b) by directly eliciting just a single attribute representing loss-of-focus using one of the established frameworks for this task (e.g.[10] or [11]).

As an immediate effect, the number of dimensions to be respected during skyline computation is reduced drastically, leading to direct performance advantages due to respectively reduced skyline sizes. However, there is a less obvious and significantly more crucial advantage resulting from this reduction which allows us to build vastly more efficient skyline algorithms. The basic considerations leading to these algorithms are as follows:

When using established skyline algorithms like BNL, the only problem which is encountered when dealing with malleability-aware dominance is that tuples are eliminated early which are required to dominate another non-skyline tuple, and due to non-transitivity, none of the remaining tuples can lead to the same dominance; thus an incorrect skyline is computed (e.g. see example in Section 3.1). Therefore, we could use a more efficient standard algorithm like BNL if it could be made "safe", i.e. if this situation can be prevented. In the general case with multiple malleable attributes, this is unfortunately not possible. But when using just one malleable attribute, the correctness of BNL depends only on the order in which the tuples are inserted into the window: for example, consider three tuples with preferences already encoded in scores $\vec{x} = (0.8, 0.8, 0.4)$, $\vec{y} = (0.7, 0.7, 0.6)$, and $\vec{z} = (0.6, 0.6, 0.8)$; the grey score represents the single malleable attribute. When computing a malleability-aware skyline with $\delta = 0.1$, then $\vec{x} >_P \vec{y} >_P \vec{z}$, and the resulting skyline is just $\{\vec{x}\}$. But due to $\vec{x} \not>_P \vec{z}$, the BNL algorithm could first test $\vec{x} >_P \vec{y}$, removing $\vec{y}$, and resulting in the skyline $\{\vec{x}, \vec{z}\}$ (because $\vec{z}$ cannot be dominated anymore). Obviously, the skyline result would be correct if the tested order was $\left(\vec{x} >_P (\vec{y} >_P \vec{z})\right)$. It is easy to see that this observation can be generalized, i.e. problems in BNL can only occur if tuples with a lower malleability score are removed before they have been tested for dominance against all tuples with a higher malleability score. Therefore, for the case that there is only one malleable attribute, we can use established algorithms like BNL if all tuples are processed in descending order with respect to the malleability attribute (preventing the situation leading to incorrect skylines described above), i.e. the skyline algorithm is therefore *stratified* with respect to the malleability attribute. This can be implemented by pre-sorting the data before executing e.g. a BNL algorithm. The effectiveness of this approach is tested later in this section.

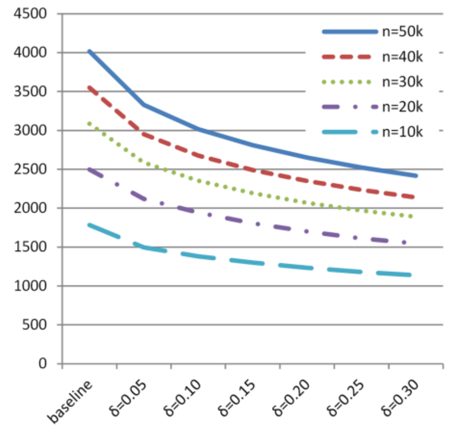

Fig. 3. Skyline Size wrt. to δ
using one malleable and 6 non-malleable attributes and varying database sizes; y-axis shows skyline size

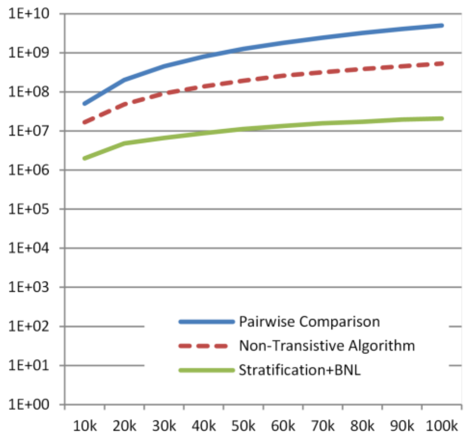

Fig. 4. Performance
using one malleable and 6 non-malleable attributes; x-axis shows #tuples in database; y-axis shows number of required tests for dominance on a logarithmic scale. $\boldsymbol{\delta = 0.15}$

Skyline Size: Before dealing with performance issues, similar to the last section, we also measured the skyline sizes for varying δ and database sizes n. Again, we generate tuples using 6 non-malleable independently distributed attributes, but just one single malleable attribute. As now the number of overall dimensions is reduced from $d = 12$ down to $d = 7$, the respective skyline sizes are also reduced dramatically to only 4,017 tuples (8% of database) for the baseline $\delta = 0$ with $n = 50{,}000$ (see Fig. 4). But still, by slightly increasing δ, the skyline can be furthermore decreased to more manageable levels (e.g. 2,809 for $\delta = 0.15$ and $n = 50{,}000$).

Performance of Algorithms: In this last set of experiments, we examined the performance of the naïve baseline, our non-transitive skyline algorithm, and the stratified BNL algorithm as described above. Again, performance is measured by the required number of tests for dominance. We also relied on synthetic data with 7 independent-attributes (one malleable, 6 non-malleable), and incrementally increased the size n of the database from 10,000 tuples up to 100,000 tuples. The results are shown in Fig. 4 (using a logarithmic y-axis). Here, we can see that the stratified BNL-algorithm needs roughly two orders of magnitudes fewer dominance tests than the naïve baseline, and is also one order of magnitude more efficient than our general non-transitive skyline algorithm. In terms of absolute runtime, the general non-transitive algorithm needed 218 seconds for $n = 100$ and $\delta = 0.15$, which is still quite long. In contrast, the stratified BNL algorithm could be executed in less than 1.4 seconds using the same hardware (the time needed for sorting the 50,000 tuples before executing the algorithm is negligible). This significant result clearly shows that malleability-aware skylines can even be used in interactive environments having tight constraints with respect to response time like for example web applications.

6 Summary & Outlook

In this paper we discussed the case of query processing over linked open data. Whereas traditional query processing algorithms are usually graph-based and use exact matches on typed links between data items in SQL-like languages like SPARQL, the fuzzy nature of semantic links calls for approximate query processing algorithms. In particular, the exact labels of links cannot always be taken at face value, because information extraction techniques, the use of different concept ontologies, and slight variations in the links' semantics introduce quite a bit of fuzzyness that algorithms have to deal with. Relying on techniques to estimate different labels' loss of focus regarding each other, in this paper we presented the first skyline query algorithm that can efficiently deal with semantically typed links in linked open data. Modeling the semantic malleability of attributes by δ-preferences, we proved that the resulting product order is indeed well-defined and can be used effectively as the basis for a sensible definition of malleability-aware skylines over linked open data.

Moreover, in our experiments we show that our innovative algorithms can efficiently evaluate such skylines and when restricting the type of malleability will even result in runtime improvements of several orders of magnitude against the baseline. Therefore, even interactive applications with tight response time requirements are possible. While we performed the algorithmic considerations here on synthetic data to test our algorithms in an unbiased environment, our future work will focus on the integration of our algorithmic framework into practical linked open data sets. Our aim is to use potential bias in the data for a tighter integration of the attribute malleability respective to each individual query. It seems that different query intensions might need different degrees of admissible malleability to stay semantically meaningful.

References

1. Hitzler, P., van Harmelen, F.: A reasonable Semantic Web. Semantic Web. 1, 39–44 (2010).
2. Heath, T., Hepp, M., Bizer, C. eds: Special Issue on Linked Data. International Journal on Semantic Web and Information Systems (IJSWIS). 5, (2009).
3. Banko, M., Etzioni, O.: Strategies for lifelong knowledge extraction from the web. Int. Conf. on Knowledge Capture (K-CAP). ACM Press, Whistler, Canada (2007).
4. Shen, W., Doan, A.H., Naughton, J.F., Ramakrishnan, R.: Declarative information extraction using datalog with embedded extraction predicates. Int. Conf. on Very Large Data Bases (VLDB). , Vienna, Austria (2007).
5. Suchanek, F.M., Sozio, M., Weikum, G.: SOFIE: a self-organizing framework for information extraction. Int. World Wide Web Conf. (WWW). , Madrid, Spain (2009).
6. Dong, X., Halevy, A.Y.: Malleable schemas: A preliminary report. Int. Workshop on Web Databases (WebDB). , Baltimore, Maryland, USA (2005).
7. Dong, X., Halevy, A.: A platform for personal information management and integration. Conf. on Innovative Data Systems Research (CIDR). , Asilomar, California, USA (2005).
8. Kasneci, G., Suchanek, F.M., Ifrim, G., Ramanath, M., Weikum, G.: Naga: Searching and Ranking Knowledge. Int. Conf. on Data Engineering (ICDE). , Cancún, México (2008).
9. DeRose, P., Shen, W., Chen, F., Lee, Y., Burdick, D., Doan, A.H., Ramakrishnan, R.: DBLife: A community information management platform for the database research community. Conf. on Innovative Data Systems Research (CIDR). Citeseer, Asilomar, California, USA (2007).
10. Mena, E., Kashyap, V., Illarramendi, A., Sheth, A.: Imprecise Answers in Distributed Environments: Estimation of Information Loss for Multi-Ontology Based Query Processing. International Journal on Cooperative Information Systems. 9, (2000).
11. Gracia, J., Mena, E.: Web-based measure of semantic relatedness. Conf. on Web Information Systems Engineering (WISE). pp. 136–150. , Auckland, New Zealand (2008).
12. Godfrey, P., Shipley, R., Gryz, J.: Algorithms and analyses for maximal vector computation. The VLDB Journal. 16, 5-28 (2007).
13. Cheng, T., Chang, K.C.-C.: Entity search engine: Towards agile best effort information integration over the web. Conf. on Innovative Data Systems Research (CIDR). , Asilomar, California, USA (2007).
14. Mandreoli, F., Martoglia, R., Villani, G., Penzo, W.: Flexible Query Answering on Graph-modeled Data. Int. Conf. on Extending Database Technology (EDBT). , St. Petersburg, Russia (2009).
15. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: A Semantic Search Engine for XML. Int. Conf. on Very Large Data Bases (VLDB). , Berlin, Germany (2003).
16. Ling Chen, Gao, S., Anyanwu, K.: Efficiently Evaluating Skyline Queries on RDF Databases. 8th Extended Semantic Web Conference. , Heraklion, Greece (2011).

17. Balke, W.-T., Güntzer, U., Lofi, C.: Eliciting Matters - Controlling Skyline Sizes by Incremental Integration of User Preferences. 12th International Conference on Database Systems for Advanced Applications (DASFAA). , Bangkok, Thailand (2007).
18. Börzsönyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. Int. Conf. on Data Engineering (ICDE). , Heidelberg, Germany (2001).
19. Fishburn, P.C.: Intransitive indifference in preference theory: A survey. Operations Research. 18, (1970).
20. Tversky, A.: Intransitivity of preferences. Psychological Review. 76, (1969).
21. Fishburn, P.C.: The irrationality of transitivity in social choice. Behavioral Science. 15, (1970).
22. Anand, P.: Foundations of Rational Choice Under Risk. Oxford University Press (1995).
23. Papadias, D., Y. Tao, G.F., Seeger, B.: Progressive skyline computation in database systems. ACM Transactions on Database Systems. 30, 41-82 (2005).
24. Balke, W.-T., Güntzer, U., Zheng, J.: Efficient Distributed Skylining for Web Information Systems. Int. Conf. on Extending Database Technology (EDBT). , Heraklion, Greece (2004).
25. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: an online algorithm for skyline queries. Int. Conf. on Very Large Data Bases (VLDB). , Hongkong, China (2002).