Morph-Skyline: Virtual Ontology-Based Data Access for Skyline Queries

1nd Marlene Goncalves
Computer Science and Information Technology
Universidad Simón Bolívar
Caracas, Venezuela
mgoncalves@usb.ve

2nd David Chaves-Fraga
Ontology Engineering Group
Universidad Politécnica de Madrid
Boadilla del Monte, Madrid
dchaves@fi.upm.es

3rd Oscar Corcho
Ontology Engineering Group
Universidad Politécnica de Madrid
Boadilla del Monte, Madrid
ocorcho@fi.upm.es

***Abstract*—Skyline queries are being used in decision-making applications to help stakeholders find the set of data that satisfies certain criteria, whose weight may not be assigned beforehand. Given the wide availability of heterogeneous datasets that are being published following Open Data initiatives, combining skyline queries with query processing approaches such as Ontology-Based Data Access (OBDA), may help stakeholders to improve their decisions exploiting and integrating multiple and heterogeneous data sources. In this paper, we address the problem of evaluating SPARQL skyline queries over an OBDA approach. Our approach implements two different techniques: rewriting skyline queries into SPARQL 1.0 and then translating to SQL, or translating them directly into queries that can be evaluated by the relational database. Our experimental results suggest that the execution time can be reduced by up two orders of magnitude in comparison to current approaches scaling up to larger datasets while identifying precisely the skyline set.**

***Index Terms*—Skyline, OBDA, Query translation, R2RML**

I. Introduction

In databases, a skyline is defined as a set of tuples which stand out among the others because they are of special interest for a specific type of problem [1]. For instance, in a database related to retail units in shopping centers, we may be interested in understanding which are the retail units where more people pass by and/or that have larger amounts of shopping transactions per day and/or while being among the smallest ones. From a more formal point of view, given a dominance relationship in a multidimensional dataset, a skyline is defined as the set of points that are not dominated by any other, where a point is considered to dominate another one if it is as good or better in all dimensions and better in at least one dimension [2]. These queries are relevant in many multi-criteria decision making applications [1]. They can help stakeholders to make better decisions when multiple criteria over data are expressed (e.g., footfall, shopping transactions). More specifically, they are really relevant when stakeholders can not predefine a score function based on these criteria, and all of them are equally important.

Skyline queries have been widely studied in relational databases (RDB) extending SQL with the skyline operator [2]. Exploiting the benefits of having a well designed RDB, including typical constraints over its schema, different algorithms [1] are proposed to efficiently obtain a high-quality skyline set from an input query. In the context of the knowledge graphs, extensions of the SPARQL query language to incorporate user qualitative preferences have been also studied [3]–[5]. Qualitative preferences allow arbitrary comparisons between the values in tuples while the skyline entails combinations of totally ordered comparisons between these values. These approaches do not provide a native operator over RDF but they rely on query rewriting techniques in order to be compliant with standard SPARQL and allowing its evaluation by any RDF triplestore. Also, [6] presents a set of client-based skyline algorithms over knowledge graphs using standard query interfaces, such as SPARQL endpoints and TPF (Triple Pattern Fragments), with no control over how the knowledge graph is stored. Although this means that skyline queries can be executed over SPARQL endpoints, the lack of techniques that exploit the data storage structures in triplestores to specifically deal with the complexity of these queries can have a negative impact over their evaluation. Thus, the performance of these queries over RDF knowledge graphs is still low.

In this work, we are interested in enabling the evaluation of SPARQL skyline queries over data that are not only available in RDF, as in the aforementioned works, but in relational databases. Ontology-Based Data Access (OBDA) has been proposed to allow access to data according to an existing ontology using a set of mapping rules [7], either by generating materialized views (RDF files) [8] or by translating SPARQL queries into queries that are supported by the underlying source, which is known as virtualization [9], [10]. The latter is specially relevant when a skyline query has to be evaluated because: i) it ensures up to date results at the moment of the execution, and ii) the skyline clause can be pushed down to the underlying data management systems (e.g., an RDBMS) exploiting the benefits of proposed skyline algorithms to improve query performance and the quality of the result sets.

Problem Statement and Research Objective: In this paper, we focus on the problem of evaluating SPARQL skyline queries over a virtual OBDA approach. We are interested in determining the feasibility of such type of query evaluation, understanding the correctness and completeness of our approach, and then, determining whether its performance is adequate in comparison with current skyline query approaches over RDF. **Approach:** We describe Morph-Skyline, a virtual OBDA approach for skyline queries. Based on the SPARQL-to-SQL

query translation approach defined in [11], Morph-Skyline translates and optimizes skyline queries from SPARQL to SQL by means of a set of R2RML mappings. The approach includes two different algorithms: i) skyQT, which pushes down the application of the skyline clause, delegating its treatment to a physical operator of the RDBMS; ii) QRT, that rewrites a SPARQL skyline query to the corresponding one in SPARQL 1.0 and then, translates it to SQL. **Evaluation:** We adapt the benchmark for skyline queries defined in [2] and the TPC-H benchmark presented in [12], with SPARQL queries of several dimensionalities. This evaluation allows understanding the impact that different data distributions, query dimensionalities, and dataset sizes have on a SPARQL-to-SQL skyline query evaluation. The results of the experiments suggest that all these variables impact on the total query execution time and OBDA approaches overcome native SPARQL methods up to two orders of magnitude. **Contributions:** Our contributions can be summarized as follows: *i*) a formal definition of the problem of evaluating SPARQL skyline queries in an OBDA context; *ii*) Morph-Skyline, an OBDA approach based on the SPARQL-to-SQL translation approach proposed by Chebotko et al. [11] that is able to evaluate SPARQL skyline queries over RDB; *iii*) the definition and implementation of two skyline algorithms, skyQT for RDBMS with specific physical operators for these queries and QRT for query rewriting techniques. *iv*) An empirical evaluation of the Morph-Skyline behavior over two benchmarks with queries of different dimensionalities.

The remainder of this article is structured as follows: Section II motivates the problem of evaluating skyline queries in an OBDA context. Section III presents related work in skyline queries and ontology-based data access (OBDA). Section IV describes our OBDA-based approach and two proposed solutions which are implemented in Morph-Skyline. Section V reports and discusses on the results of our empirical study, and finally, Section VI concludes and gives insights for future work.

II. Motivating Example

Suppose a database containing data from the Madrid metro system[1] following the GTFS (General Transit Feed Specification) model[2], a table named $demands$ storing statistics on annual accumulated demand or the number of users by line, and a table called $uses$ containing the number of uses or movements within each station where uses are calculated as the number of entries and exits through the turnstile plus the number of transfers between lines if it is a station with access to more than one Metro line. Also, consider that the Madrid Metro Council wants to identify the least used stations within the most demanded lines so as to make some decisions on the types of services to be offered (e.g. the rental value of the business units inside of stations or the amount of personnel in these stations). According to the Madrid Metro Council, both the station uses and demands per line are equally important and relevant; hence, a predefined score function cannot be

[1]https://www.metromadrid.es

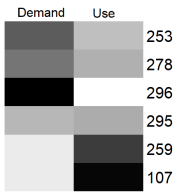
[2]https://developers.google.com/transit/gtfs

```
SELECT DISTINCT ?route ?stop ?demand ?use
WHERE {?trip gtfs:route ?route.
  ?stopTime gtfs:trip ?trip; gtfs:stop ?stop.
  ?stop ex:use ?use. ?route ex:demand ?demand.}
SKYLINE OF ?demand MAX,?use MIN
```

(a) SPARQL skyline Query

Line	Station	Demand	Use
5	253	59,429,010	1,267,151
10	278	65,134,740	1,481,940
7	296	38,497,353	351,971
1	295	80,042,527	1,536,049
6	259	92,049,658	3,142,010
~~6~~	~~107~~	~~92,049,658~~	~~3,906,267~~

(b) Skyline

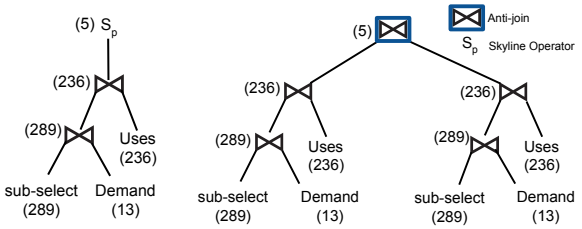


(c) Heatmap

Fig. 1: **A sample skyline based on the Madrid Metro network**. Metro stations whose services may need to be reorganized in terms of demands per line (Demand) and station uses (Use). We can observe that in line 6, the station 107 is dominated by the station 259.

assigned to be used in a query. A station can be chosen if there is no other station with a lower use belonging to a line with a higher demand. To select a station, we must identify the set of all the stations that are non-dominated by any other station in terms of two criteria: minimizing station uses and maximizing demands per line. Following these criteria, a SPARQL skyline query is expressed in Fig.1a, and the computed skyline is presented in Fig. 1b. The stations 253, 278, 296, 295, and 259 are the non-dominated ones, i.e., there is no other station with values better than them in these two attributes. Additionally, a station s_1 dominates a station s_2, if s_1 has better or equal values and at least one better in demand and use than s_2, e.g., the station 259 dominates the station 107. Moreover, for these stations, a heatmap is shown in Fig. 1c which represents the best values with lighter colors. Visually, a station s_1 dominates a station s_2 if s_1 has the same or lighter colours than s_2 and at least one lighter color. It can be noticed that the station 107 has the same color in Demand as the station 259 but a darker color in Use. In Fig. 2 we show two different query plans from a skyline query to obtain the results of the example. In our example, preferences are calculated by joining $trips$, $stop\_times$, $stops$, $routes$, $demands$ and $uses$ firstly, then performing the skyline operator on the join result (Fig. 2a). This solution can also be calculated with an equivalent translated query without the Skyline clause (Fig. 2b). We can observe that the results obtained (Fig. 2c) for the option that implements the skyline clause inside the DBMS obtains improved results.

In addition, the Madrid Transport Consortium (CTM) wants to create seamless multimodal mobility solutions, combining various travel services of its Transport Service Providers (TSP). However, most TSP systems are based on RDB technology and a change in their data model can be costly and time-consuming. Thus, CTM has decided to convert these data to another data model (RDF) under the OBDA approach [7] to make them available without renewing its entire software infrastructure. The RDF materialization is not



(a) Skyline Query Plan (b) Translated Query Plan

Approaches/Metrics	Execution Time (sec)	# Intermediate Results
Skyline Query	9.185	9,984
Translated Query	12.025	11,047

(c) Obtained results

Fig. 2: **Skyline and translated query plans and results**. Plans for the query that retrieves the skyline of the least used stations within the most demanded lines. Sub-select represents the join between $trips$, $stop\_times$, $stops$, and $routes$ which produces 8,916 intermediate results. The operator Anti-join corresponds to FILTER NOT EXISTS including pair-wise dominance checks on sub-select with itself.

an option because the enormous data size and to ensure the results are always up to date. Taking the advantage of an OBDA approach [7], skyline queries can be abstracted of schema-level details representing them using common and shared vocabularies while query evaluation benefits from the exploiting of skyline algorithms over an RDB instance.

III. State of the Art

The problem of efficiently computing the skyline for a dataset and a query has been extensively studied in the literature [2], [13]–[15]. Initially, Börzsönyi and colleagues [2] introduced two solutions to evaluate skyline queries in the context of databases: the first solution was based on the divide & conquer principle where data is partitioned to be processed and merged in main memory; the second solution was based on the Block Nested Loop (BNL) algorithm where each tuple is compared with the rest and the tuple is returned only if it is not dominated by any other. Subsequently, other solutions were introduced to improve the BNL performance by means of a monotone preference function that exploits data ordering properties [13]–[15]. In addition, several works benefit from the properties of index structures to progressively return more and more results until the full skyline is retrieved [16], [17]. All these skyline algorithms based on scanning or indexing can be integrated as physical implementations for the Skyline operator inside a database engine, showing substantial benefits [2], [12], [18]. In these works, the skyline operator is not executed on top of an RDBMS but it is incorporated during the query processing considering cost and cardinality estimation of the skyline operator. Bearing in mind that the skyline as a physical operator within a relational engine is an efficient implementation for computing the skyline, and the maturity of relational database systems, in this work, one of the two proposed techniques evaluates a skyline query using an OBDA system that provides such physical operator in the RDBMS.

There are some works related to extensions of SPARQL with qualitative preferences [3]–[5], which are based on a more general operator than skyline called winnow [19]. The authors in [3] propose to add preference-based querying capabilities to SPARQL. SPREFQL [4] is another extension of SPARQL for qualitative preferences. At the implementation level, they presented a query rewriting technique that maps from a SPREFQL query to an equivalent SPARQL query by means of the NOT EXISTS operator. Unfortunately, their solution based on query rewriting does not work correctly due to the fact that it is based on the SPARQL EXISTS, which has many known problems [20]. Thus, [5] identified and fixed the problem in the previous proposals for acyclic and transitive preference relations. Datalog +/- was extended in [21] to include preferences and the authors developed algorithms to answer more general preferences than skyline queries over Datalog +/- ontologies. Finally, [6] presented a set of client-based algorithms to evaluate skyline queries over knowledge graphs using standard query interfaces for RDF. They are focused on the optimization over client architectures so they assumed no control over how the data is stored (e.g., indexes, internal structures, etc), hence, they cannot exploit them to optimize the performance and quality of the queries at server side. They introduced methods on different interfaces: SPARQL endpoints (not implemented), Triple Pattern Fragments (TPF), Bindings-restricted Triple Pattern Fragments (brTPF) and a TPF-like interface (skyTPF).

To the best of our knowledge, existing OBDA engines do not support skyline queries. The most related work is [22] which considers the problem of evaluating top-k queries in the context of OBDA over relational databases. OBDA systems are usually focused on the transformation of the original sources into a global schema and its corresponding materialization [8] but also on query translation techniques for highly dynamic data sources [9], [10]. To cover the features of different data formats, many different types of OBDA mapping languages have also been proposed in the last few decades, with a wide variety of syntax and formats. Since its W3C recommendation in 2012, R2RML has become the standard mapping specification for accessing relational databases. Many tools support these rules, some of them materialize the data into a knowledge graph (e.g. DB2Triples[3] and R2RMLParser[4]) and others provide virtual RDF views, focusing on formalizing the translation of SPARQL into SQL and optimizing the resulting SQL query (Morph-RDB [10] and Ontop [9]).

IV. Morph-Skyline: OBDA-based Skyline Queries

In this section we describe Morph-Skyline, an OBDA framework for translating and executing skyline queries from SPARQL-to-SQL. Our formalization is based on OBDA [7], which relies on conceptually representing a domain of interest over data stored in an underlying database system.

[3]https://github.com/antidot/db2triples
[4]https://github.com/nkons/r2rml-parser

Definition 1 (OBDA Specification [7])**.** *OBDA is defined as a triple $\sigma = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ where $\mathcal{O}$ is an ontology that describes the domain, $\mathcal{S}$ is a schema, and $\mathcal{M}$ is a mapping between $\mathcal{O}$ and $\mathcal{S}$. In addition, an OBDA instance is defined as a tuple $\mathcal{PI} = \langle \sigma, \mathcal{D} \rangle$ where σ is an OBDA specification and $\mathcal{D}$ is a data instance conforming to $\mathcal{S}$.*

Definition 2 (Skyline over a relational database)**.** *Let A_i be the set of attributes belonging to a relational table in $\mathcal{S}$. For $d \geq 1$, let $I = \{i_1, \ldots, i_n\}$ be a set of tuples characterized by a set of attributes $A = \{a_1, \ldots, a_d\} \subseteq \bigcup_{i=1}^{t} A_i$ where t is the number of tables in $\mathcal{S}$, and let $F = \{f_1, \ldots, f_d\}$ be a set of directives where each f_j specifies if the value in a_j is minimized or maximized. We assume that each $a_j \in A$ consists of numeric values only, and without loss of generality, we also assume minimization for each $f \in F$, i.e., a smaller value is preferred for each attribute $a_j \in A$. A tuple i_k dominates a tuple i_l, denoted by $i_k \prec i_l$, if $(\forall a_j \in A | i_k.a_j \leq i_l.a_j) \wedge (\exists a_j \in A | i_k.a_j < i_l.a_j)$. The Skyline set is a subset of I such that: $\{i_l \in I \mid (\neg \exists i_k \in I \mid i_k \prec i_l)\}$. A skyline query over a relational database is defined as a pair $q = (SQ, SA)$ where SQ is a non-skyline SQL query on $\mathcal{S}$ and SA is a set of pairs, each of which is a numeric attribute $a_j \in A$ belonging to tables included in SQ with its corresponding directive $f_j \in F$.*

Definition 3 (Skyline Query over an RDF Graph)**.** *Let $[\![P]\!]$ be the set of solutions to a SPARQL pattern P. Let SV be a set of d pairs, each of which is a numeric variable v_j that occurs in P with its corresponding directive f_j of minimization or maximization. A skyline query over an RDF Graph G is defined as a pair $q = (P, SV)$ which produces a subset of $[\![P]\!]$ such that: $\{p_l \in [\![P]\!] \mid (\neg \exists p_k \in [\![P]\!] \mid p_k \prec p_l)\}$ where $\prec$ is defined as Def 2 replacing I with $[\![P]\!]$, and A with the set of variables occurring in SV.*

Problem Definition. Given an OBDA Specification $\sigma = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ and an SPARQL skyline query $Q = (P, SV)$ over an RDF graph G resulting from the application of σ, the problem of skyline query translation in an OBDA setting is defined as finding a skyline query $Q' = (SQ, SA)$ over $\mathcal{S}$ that meets the following conditions:

- there is a function $\mu: P \rightarrow SQ$ that maps P to its corresponding SQL query over $\mathcal{S}$ according to the mappings $\mathcal{M}$. The function μ is defined as *trans* in [10].
- there is a mapping $\mu_i = (term(v), v') \in \mathcal{M}$ such that each pair $p = (v, r) \in SV$ matches $p' = (v', r) \in SA$ and there is no pair in SA that does not match a pair in SV where $term(v)$ is the ontology term corresponding to the variable v, e.g., the ontology term associated with the ?demand variable is mapped to the database attribute demand by using $\mathcal{M}$. Thus, in our motivating example, $SV = \{(?demand, MAX), (?use, MIN)\}$ is translated to $SA = \{(demand, MAX), (use, MIN)\}$.
- There is a method to translate Q to Q' where the results of Q' are equal to Q_{sql} being Q_{sql} a perfect SQL query skyline (gold standard), i.e., over any relational database, a SQL query written manually by an expert and the equivalent one in SPARQL (that then, is translated with our proposal) have the same results.

```
SELECT DISTINCT route_long_name, stop_name,demand,
                use
FROM stops s1
JOIN stop_times s2 ON s2.stop_id=s1.stop_id
JOIN trips t ON t.trip_id=s2.trip_id
JOIN routes r ON r.route_id=t.route_id
JOIN demands d ON s.route_id=d.route_id
JOIN uses u ON u.stop_id=s1.stop_id
SKYLINE OF demand MAX, use MIN
```

(a) A SQL skyline query.

```
SELECT DISTINCT ?route ?stop ?demand ?use
WHERE {?trip gtfs:route ?route.
       ?stopTime gtfs:trip ?trip; gtfs:stop ?stop.
       ?stop :use ?use. ?route :demand ?demand.
OPTIONAL {?trip2 gtfs:route ?route2.
       ?stopTime2 gtfs:trip ?trip2; gtfs:stop ?stop2.
       ?stop2 :use ?use2. ?route2 :demand ?demand2.
FILTER(?use2<=?use && ?demand2>=?demand &&
      (?use2<?use || ?demand2>?demand))}
FILTER(!BOUND(?stop2))}
```

(b) A SPARQL translated skyline query.

Fig. 3: **SQL and SPARQL Skyline queries**. The skyline of the least used stations within the most demanded lines.

Proposed Solution

We propose Morph-Skyline as an OBDA skyline query translation engine for relational databases. Morph-Skyline is based on the Chebotko et al.'s translation approach [11] which formalizes query translation from SPARQL into SQL exploiting the use of a set of mapping rules and their corresponding functions. More specifically, Morph-Skyline relies on the formal definition of mappings and functions with R2RML provided by [10].

Query translation exploiting skyline operators in databases. skyQT is a technique based on a physical skyline operator. By using skyQT, a SPARQL skyline query $Q = (P, SV)$ is translated to a SQL skyline query skyline query $Q' = (SQ, SA)$. First, P is translated to SQ by means of the function μ defined as *trans* in [10]. Then, each pair $p = (v, r) \in SV$ is matched to $p' = (v', r) \in SA$ according to the mapping $\mathcal{M}$. Thus, the translated skyline query will be executed into the underlying database system. The SQL skyline query in Fig. 3a is the result of translating P to SQ, and $SV = \{(?demand, MAX), (?use, MIN)\}$ to $SA = \{(demand, MAX), (use, MIN)\}$. Algorithm 1 sketches the algorithm skyQT implemented by Morph-Skyline to process skyline queries in an OBDA context. skyQT receives a set of mappings M and a SPARQL skyline query Q. It firstly initializes SA, which will contain the skyline query translated to SQL, and it separates the skyline clause from the query Q and gets a non-skyline subquery (lines 1-3). In line 4, skyQT translates the non-skyline query nsQuery into SQL by means of the function *trans* [10]. Lines 5-6 build a hash structure from the pattern P belonging to the query nsQuery; this hash structure contains the attribute that maps each variable. Note that β returns the column/constant that corresponds to each

Algorithm 1 Skyline Query Translation, skyQT: $\mathcal{M}$ - Mappings, Q - Skyline query

1: $SA \leftarrow \varnothing$
2: sky $\leftarrow$ skyline clause from Q
3: nsQuery $\leftarrow Q \setminus$ sky
4: query $\leftarrow$ trans(nsQuery, $\mathcal{M}$)
5: pa $\leftarrow$ P from Q
6: $h \leftarrow \beta(pa, \mathcal{M})$
7: **for** each pair $(v_i, d_i) \in$ pairs(sky) **do**
8: $a_i \leftarrow$ get(h, v_i)
9: create a pair $p =(a_i,d_i)$
10: add p to SA
11: **return** (query, SA)

variable in a SPARQL query [10]. Subsequently, for each variable in the skyline clause, skyQT iteratively obtains its corresponding attribute by searching for it in the structure hash h, and creates a pair of a mapped attribute and a directive which is inserted into SA (lines 8-10). Finally, skyQT returns a pair with the translated query and SA.

Skyline query rewriting in SPARQL. The Query Rewriting Technique (QRT) proposed in [4] is an alternative to evaluate SPARQL skyline queries. It consists of rewriting a SPARQL skyline query by using "NOT EXISTS" (for SPARQL 1.1) or "OPTIONAL" and "FILTER" (for SPARQL 1.0) [4]. Since "FILTER NOT EXISTS" has known problems [5], QRT was analysed using the standard SPARQL 1.0. By using QRT, a SPARQL skyline query $Q = (P, SV)$ is translated to SPARQL 1.0 as: $Q' =$ P `OPTIONAL` { P' `FILTER` $(dominance(P, P', SV))$} `FILTER (!bound(?check))`, where P and P' represent SPARQL patterns, P' is the same graph pattern than P but with all variables renamed as fresh variables, `?check` is one of the fresh variables that is used to bind, SV is the preference criteria, and `dominance` is a function that checks dominance between solutions from P and P', and it is defined in Def. 4. `FILTER` within the `OPTIONAL` clause allows to perform dominance checks for each pair of instances and the `FILTER (!bound(?check))` verifies the instance is not dominated.

Definition 4 (Dominance)**.** *Let $P_{max} = \{p_1, \ldots, p_k\}$ and $P_{min} = \{p_{k+1}, \ldots, p_n\}$ be the set of variables belonging to SV to be maximized and minimized, respectively. Each variable p_i in $P_{max} \cup P_{min}$ is renamed as q_i according to P'. The $dominance(P, P', SV)$ function rewrites each criterion in SV as: i) $p_i \geq q_i$ for all $i = 1, \ldots, k$, and $p_i \leq q_i$ for all $i = k+1, \ldots, n$; and, ii) $p_i > q_i$ for some $i = 1, \ldots, k$ or $p_i < q_i$ for some $i = k+1, \ldots, n$.*

To illustrate the rewriting technique, consider again our motivating query shown in Fig. 1a. `?use,stop` of P are renamed as `?use2,stop2` for P' in Fig. 3b. `FILTER` within the `OPTIONAL` clause applies the dominance function of Def. 4 which allows to perform dominance checks for each pair of solutions and the last `FILTER` verifies the solution from P is not dominated by any solution from P'. If `?stop2` is bound means that it is dominated because the inner `FILTER` found a better solution than `?stop`. For a more detailed description of QRT, please refer to [4].

V. Experimental Evaluation

We study the efficiency and effectiveness of our two implementations of Morph-Skyline. First, we describe the hypotheses that we want to validate as well as the datasets and queries, mappings, metrics and implementation details for our experimental study. All the resources used are online[5,6].

Hypotheses: *i)* H_1: As the skyline query dimensionality increases, the skyline cardinality augments, hence, the OBDA skyline algorithms have to perform more dominance checks; therefore, they increase their execution time; *ii)* H_2: As the dataset size increases, the skyline size also enlarges [23] and the OBDA skyline algorithms have to perform more dominance checks; therefore, they increase their runtime; *iii)* H_3: A SPARQL-to-SQL skyline algorithm executed as a physical operator within a relational database engine has better performance than the operator executed on top of a database engine, even though the engine included techniques to optimize the query; *iv)* H_4: Morph-Skyline performs better than the current SPARQL-based approach on materialized RDF for skyline query evaluation.

Datasets and Queries: We have used two different benchmarks for the experimental evaluation of our proposal: *i*) We have generated synthetic datasets by using the benchmark generator implemented in [2]. We have generated data with three different distributions: correlated, anti-correlated and independent (uniform). For each distribution, five datasets were generated with 10K, 100K, 1M, 10M, and 100M tuples, respectively. Each dataset is characterized by an identifier and 10 dimensions. For each dataset, 9 queries with the MIN directive were evaluated varying dimensions from 2 to 10; *ii*) TPC-H: We use the Transaction Processing Council Ad-hoc/decision support benchmark consisting of 15 queries expanded by PREFERRING clauses in [12]. We adapt this benchmark with SPARQL queries. Only 6 of the queries presented in [12] are skyline with minimizing and maximizing criteria. We have generated an 1GB TPC-H dataset where the largest table has 6 million tuples. The skyline queries in TPC-H are $Q0, Q2, Q4, Q6, Q8$ and $Q9$ which include joins and vary dimensions from 1 to 4. The 1GB dataset was transformed into RDF by means of SDM-RDFizer [24] where the total number of triples is 111 million and the RDF file size is 20 GB. Finally, we have selected the benchmark in [2] because it is the commonly used benchmark for evaluating skyline queries which generates independent, correlated and anti-correlated data for a single table. Thus, for each relational schema, there is only one table. Also, since joins are costly operators in SPARQL-to-SQL engines [25], we have included the TPC-H benchmark that involves joins into their queries in order to execute more complex skyline queries.

Mappings: *i*) For the benchmark defined in [2], we have created an R2RML mapping document for accessing each relational dataset. For each dataset size and data distribution,

[5]https://github.com/oeg-upm/morph-skyline
[6]https://doi.org/10.5281/zenodo.3974178

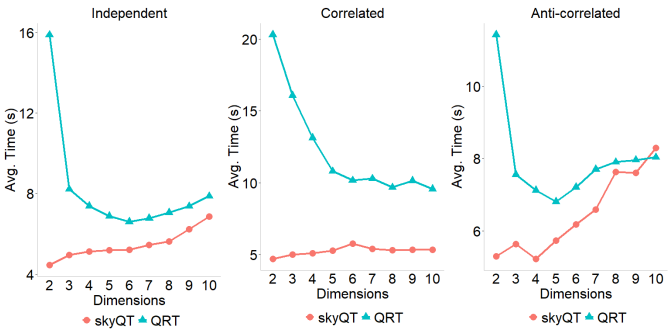
a table $T(id, d_1, \ldots, d_{10})$ is created and identified by the data distribution type and the dataset size. Each table T is mapped to a class Dataset with the attribute id as the identifier, using `rr:template` for generating the URI of the instances. Each attribute d_i is mapped to an ontology property d_i'; *ii*) For the TPC-H benchmark, we have created an R2RML mapping document for accessing each table with 53 PredicateObjectMaps, 53 Predicates, 53 ObjectsMaps, and 9 JoinConditions.

Evaluation Metrics: We measure performance as query execution time. It is computed as the elapsed time in seconds between the submission of a query to the engine and the delivery of the answers. Each query was executed 5 times in cold mode and timeouts are set up to 1 hour. The quality of skyline techniques are also measured in terms of precision, recall and F-measure. Precision measures the percentage of instances that should be in the skyline set computed in terms of the ground truth; recall measures the percentage of instances produced in terms of the ground truth. Ground truth corresponds to the skyline directly retrieved from the relational database engine and then materialized in RDF using the RML mappings and the SDM-RDFizer [24] engine to ensure their correctness.

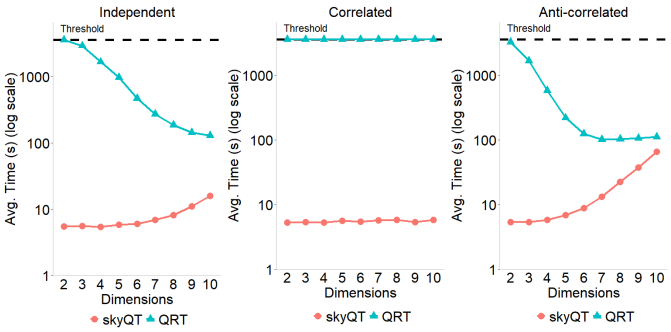
Engines: Morph-Skyline is an open source software under the Apache 2.0 licence written in Scala. The RDB engine used is EXASol because it includes a dedicated skyline operator implementing the BNL algorithm [12]. We have compared Morph-Skyline against SPREFQL [4], which is developed in Java and can query data loaded into a locally deployed SPARQL endpoint of Virtuoso Community Edition Version 7.1. We have also evaluated the multi-threaded version of the skyTPF and brTPF-based methods [6], a Java servlet implementation that uses RDF-HDT data sources to process skyline queries. We also wanted to study the multi-threaded version of skyTPF [6] for the benchmark defined in [2] but it outputs an execution error. We considered testing our work against NL [4], and TPF-based method [6], but the authors showed that these algorithms have worse performance w.r.t BNL, and the skyTPF and brTPF-based methods, respectively. Experiments are executed on a machine with the following characteristics: 2GHz CPU with 8 cores, 64 GB RAM with Ubuntu 18.04 as its operating system.

Discussion of the Observed Results.

In order to validate the proposed hypothesis we execute Morph-Skyline with the two implemented skyline OBDA algorithms (skyQT and QRT) using the queries defined by the range of dimensions (2-10) and the selected data sizes for all the distributions. Additionally, using the same resources, we run the skyTPF and brTPF-based methods [6], and SPREFQL [4] with two different algorithms: BNL where specific operators are implemented on the top of the triple store for each algorithm and RW, that implements a query rewriting technique. Although the methods, developed by SPREFQL, were designed for the winnow operator [19], certainly they can be adapted to evaluate the skyline operator as a particular



(a) Performance of Morph-Skyline for 10K datasets

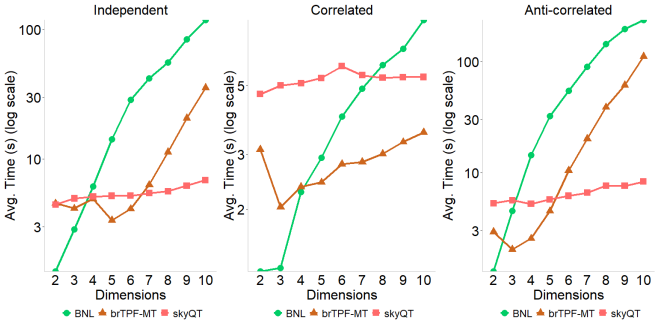


(b) Performance of Morph-Skyline for 100K datasets

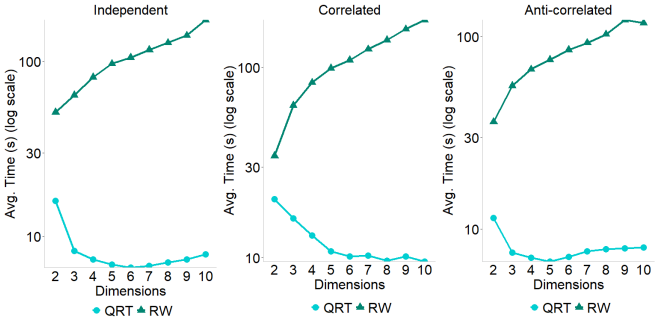
Fig. 4: **Performance of Morph-Skyline algorithms**. skyQT clearly overcomes QRT in all the cases.

case. With this comparison, we want to demonstrate the importance of skyline physical operators, and that translations to equivalent ones are not a good solution neither to native knowledge graphs nor virtual knowledge graphs. We analyze the results from the experimental setup more in the detail in the following sections.

Query rewriting vs physical skyline operators in OBDA. Fig. 4 reports the average execution time in seconds varying the number of dimensions from 2 to 10 for skyQT and QRT. The first observed result is that we cannot compare the two algorithms over all the generated data sizes due to the fact that QRT does not answer the queries of 1M, 10M and 100M on time, which means that data size clearly impacts over this algorithm. For the data sizes where we can test both algorithms the skyQT performance is better in almost all cases than for QRT, the average execution time for skyQT is up to two orders of magnitude less than for QRT. Thus, our hypothesis H_3 holds: the physical operator of skyQT executed within a relational database engine has better performance in view of the fact that the engine optimizes the query. Nevertheless, QRT is a bit better for the 10-dimensional query on the anti-correlated dataset for 10K size although the query takes on average about 0.26 ms less than skyQT. The main reason is because QRT performs a join among a set of dominated instances and the rest of instances but, in this case, the number of dominated instances are very small, which means that the size of the skyline is almost as big as the dataset size and the number of performed joins is reduced to the minimum. This is why QRT approximates skyQT with each increase in the



(a) Physical Operators for 10K datasets



(b) Rewriting Techniques for 10K datasets

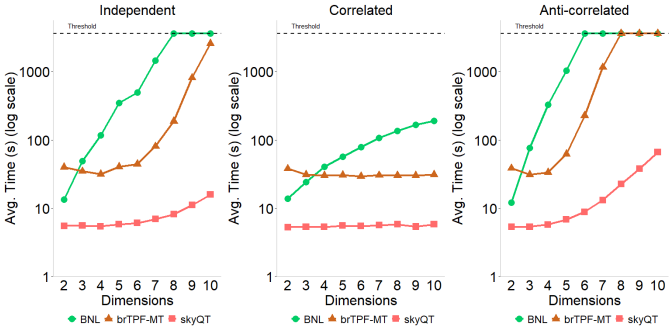
Fig. 5: **Comparison of skyline algorithms over SPARQL and OBDA approaches for 10K dataset size**. We can observe that Morph-Skyline performs better than RDF-native approaches when rewriting techniques are used.

number of dimensions. Additionally, we analyze the behaviour of the QRT algorithm in order to answer: why it does not scale-up (impact of the data size), contradicting hypothesis H_2, and why the performance worsens as the dimensionality increases, contradicting hypothesis H_1. We observed that both explanations are related and not dependent of the algorithm but on the RDBMS that has been used. QRT compares each instance t against the others to verify that there is no better one than t, which means that H_1 and H_2 should be held. Although we had assumed in our hypothesis H_1 that the performance worsens as the dimensionality increases, QRT contradictorily improves its performance. Thus, we have double checked QRT with up to 6-dimensional queries and a MySQL database that stores an independent 10K dataset, and the results obtained are presented in Table I. However, unlike QRT with EXASol, the performance does become worse with MySQL as we had anticipated. The execution time in MySQL for the same queries ranging from 13 min to 8 hr. Also, we have analyzed the execution plans of this kind of queries on EXASol to understand why the performance improves. The main complex part of the query plan is a left outer join between the input table R and the result produced by a theta self-join of the input table R: $R \bowtie (R \bowtie R)$. The engine, after performing the theta self-join operator, builds an on-the-fly index for the left outer join to optimize the execution. This index can be huge, e.g., the index contains approximately 2 billions of entries for 100K tuples and a 2-dimensional query, while the index comprises approximately 200 million entries for 100K tuples and a 4-dimensional query. For this reason, QRT performance improves in EXASol due to the fact that less data is indexed as dimensionality increases. We performed a similar experiment over another RDBMS and we observed that both hypotehesis hold. Finally, this solution is not scalable due to the amount of space it needs and it is the reason why QRT does not provide results for the rest of data sizes on time. Also, we can observe that although QRT includes specific optimization techniques for these types of queries, it is not able to overcome algorithms that exploit the use of pyhsical operators.

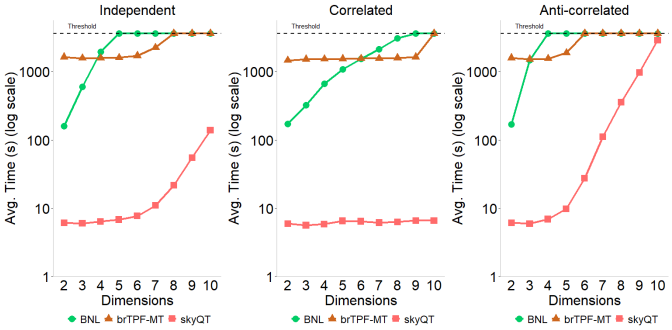
Dimensions	Avg. Time (s)
2	789.98
3	2378.97
4	6680.01
5	14436.96
6	29057.03

TABLE I: **Average execution time in seconds using Morph-Skyline with MySQL and QRT**. Performance degrades with query dimensionality for the rewriting technique over MySQL. Unlike EXASol, MySQL does not run any additional optimization and like skyQT, as many dominance checks are performed as the the square of skyline size.

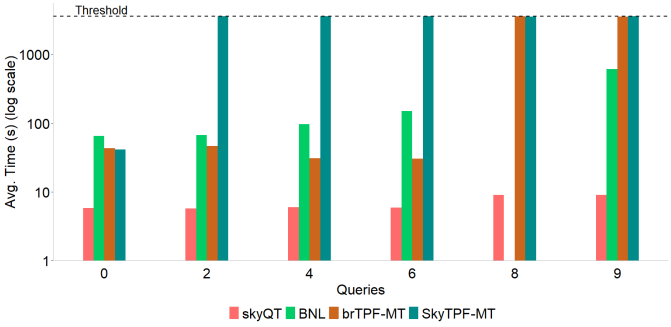
Morph-Skyline vs native SPARQL. We compare Morph-Skyline against SPREFQL and the brTPF-based method in order to analyze their performance with respect to the number of dimensions for three data distributions under 10K datasets as a case to show what happens. Fig. 5 reports the average execution time in seconds varying the number of dimensions from 2 to 10 for Morph-Skyline and the state-of-the-art tools. Particularly, Fig. 5a depicts the average execution time of skyQT against the BNL operator of SPREFQL and the brTPF-based method and Fig. 5b compares the average execution time of the rewriting techniques. It is noteworthy that skyTPF is not reported because it returned a runtime error, and the execution time of SPREFQL and brTPF-based method does not include the materialization time for generating RDF triples. We can observe in Figure 5a that BNL performs better than Morph-Skyline for low-dimensional queries with 3-6 dimensions depending on the data distribution, possibly because of the small size of the skyline set and the efficient data access mechanisms by Virtuoso. For the three data distributions where BNL is better, the cardinality of the skyline is lower than 520. On the other hand, skyQT has a dedicated physical skyline operator inside the database engine which allows the engine to directly apply optimizations during the skyline query processing and thus, it scales-up with the query dimensionality, reflecting in runtimes always below 10 seconds, while the execution time for BNL is up to two orders of magnitude more than skyQT. More in detail, skyQT outperforms BNL in medium and high dimensions (> 4) because of the growth of the intermediate results of the self-joins in Virtuoso. We have also observed in Figure 5a that the brTPF-based method performs better than Morph-Skyline for correlated data and with queries of up to 5-6 dimensions in the case of independent and anti-correlated



(a) Physical Operators for 100K datasets



(b) Physical Operators for 1M datasets



(c) Physical Operators for TPC-H

Fig. 6: **Scalability of OBDA vs SPARQL**. We compare skyline algorithms over OBDA and SPARQL approaches for larger datasets. We can observe that Morph-Skyline performs better than RDF-native approaches for larger datasets.

data, possibly because the data is processed by groups of triple patterns. In addition, Figure 5b shows that the rewriting technique on SPREFQL (RW) has the worst performance w.r.t QRT. Thus, our results also validate our hypothesis H_4.

On the other hand, Fig 6 depicts that Morph-Skyline overcomes SPREFQL and the brTPF-based method for larger datasets ($\geq 100K$) when we increase the size of the dataset. For the TPC-H benchmark, the skyTPF-based method successfully ran and it was included in Fig. 6c. It is also important to highlight that SPREFQL was unable to execute the query 8 of the TPC-H benchmark because the preferring clause includes functions over the criteria. We can observe that SPREFQL, and skyTPF and brTPF-based methods are worse than Morph-Skyline showing the benefits of a physical operator inside the

Q	Independent			Correlated			Anti-correlated		
	P	R	F	P	R	F	P	R	F
2	0.70	0.64	0.45	0.75	0.50	0.38	0.60	0.67	0.40
3	0.79	0.79	0.62	0.86	0.67	0.57	0.44	0.21	0.09
4	0.78	0.76	0.59	0.93	1.00	0.93	0.57	0.22	0.12
5	0.77	0.78	0.60	0.90	1.00	0.90	0.65	0.28	0.18
6	0.75	0.80	0.61	0.88	1.00	0.88	0.75	0.34	0.25
7	0.76	0.81	0.62	0.89	1.00	0.89	0.82	0.46	0.38
8	0.78	0.86	0.66	0.90	1.00	0.90	0.89	0.57	0.51
9	0.81	0.80	0.65	0.90	1.00	0.90	0.94	0.70	0.65
10	0.84	0.85	0.71	0.91	1.00	0.91	0.98	0.79	0.78

TABLE II: **Skyline Resultset Quality**. SPREFQL produces incorrect and incomplete answers. For the sake of simplicity, Morph-Skyline values are omitted since they are always 1.0

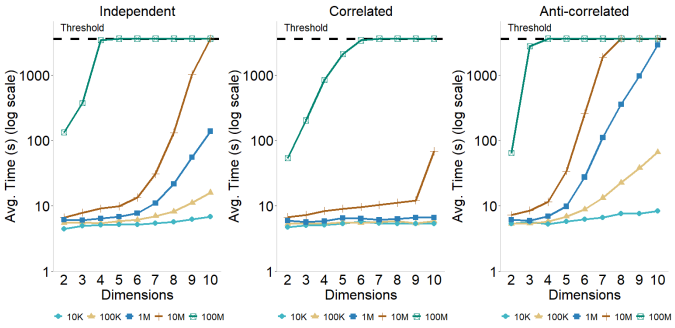


Fig. 7: **Morph-Skyline execution time over skyQT**. Performance worsens exponentially with the query dimensionality and the dataset size.

database engine. Thus, these results validate our hypothesis H_4: a SPARQL-to-SQL skyline query has better performance than the current SPARQL-based approach on materialized RDF.

Additionally, we report the SPREFQL quality for 10K datasets under the three data distributions. SPREFQL can produce incomplete and incorrect responses because it was developed for preferences more general than skyline. Table II reports precision, recall and F-measure for SPREFQL varying the number of dimensions and, in general, the three measures are improved as dimensionality increases. Incorrect skyline tuples identified by SPREFQL can be those dominated tuples that dominate another tuple and the missing ones by SPREFQL can be those skyline tuples that do not dominate another. We omit the Morph-Skyline quality because its precision and recall is 1.0, as evaluated by us.

Impact of the skyline parameters over Morph-Skyline. We also empirically study how the dataset size and dimensionality affects the total execution time of Morph-Skyline. Regarding the aforementioned scalability problems of the QRT approach, we show the results for skyQT over all the dataset sizes. More in detail, Fig. 7 shows how the skyQT performance is affected varying the number of dimensions from 2 to 10 for dataset sizes from 10K to 100M. We can observe that for each dimension, the execution time grows exponentially as the number of dimensions increases, thus corroborating our hypothesis H_1 for all the data distributions. Additionally, skyQT is run using BNL, where its cost directly depends on

the dataset size. In the worst-case, BNL is $O(n^2)$, where n is the data size, which means, that skyQT is clearly affected by the dataset size. This also holds our hypothesis H_2.

VI. Conclusions and Future Work

In this paper, we describe Morph-Skyline, an OBDA-based engine that identifies a subset of data among the whole virtual knowledge graph that best meets criteria of a stakeholder request expressed as a SPARQL skyline query. We designed and implemented two algorithms: skyQT and QRT, algorithms based on a SPARQL-to-SQL query translation and able to compute the skyline set in an OBDA context. We studied the performance of skyQT with respect to the query rewriting technique, QRT, on synthetic data. The experimental results suggest that skyQT outperforms QRT and is capable of scaling to larger datasets. We also reported experimental results with state-of-the-art tools to evaluate skyline queries over RDF data. These results showed that our approach outperforms state-of-the-art tools for larger datasets. In the future, we plan to consider skyline queries including the DIFF directive, which allows grouping the skyline by the attribute that comes before the directive. We also want to evaluate skyline queries over data sources in other data formats such as CSV or JSON adapting, to skyline queries, general existing benchmark for virtual knowledge graph construction [26]. We also plan to empirically study the performance and scalability of our tool in real use cases from the transport domain.

Acknowledgments

The work presented in this paper is supported by the project Semantics for PerfoRmant and scalable INteroperability of multimodal Transport (SPRINT H2020-826172) and by the Spanish Ministerio de Economía, Industria y Competitividad and EU FEDER funds under the DATOS 4.0: RETOS Y SOLUCIONES - UPM Spanish national project (TIN2016-78011-C4-4-R) and by an FPI grant (BES-2017-082511).

References

[1] C. Kalyvas and T. Tzouramanis, "A survey of skyline query processing," Computing Research Repository Journal, vol. abs/1704.01788, 2017.

[2] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," in Proceedings of the 17th International Conference on Data Engineering. IEEE Computer Society, 2001, pp. 421–430.

[3] W. Siberski, J. Z. Pan, and U. Thaden, "Querying the Semantic Web with preferences," in The Semantic Web - ISWC 2006. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 612–624.

[4] A. Troumpoukis, S. Konstantopoulos, and A. Charalambidis, "An extension of SPARQL for expressing qualitative preferences," in The Semantic Web. Cham: Springer International Publishing, 2017, pp. 711–727.

[5] P. F. Patel-Schneider, A. Polleres, and D. Martin, "Comparative preferences in SPARQL," in Knowl. Eng. and Knowl. Management. Cham: Springer International Publishing, 2018, pp. 289–305.

[6] I. Keles and K. Hose, "Skyline queries over knowledge graphs," in The Semantic Web. Cham: Springer International Publishing, 2019, pp. 293–310.

[7] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyaschev, "Ontology-Based Data Access: A survey," in IJCAI. ijcai.org, 2018, pp. 5511–5519.

[8] A. Dimou, M. V. Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. V. de Walle, "RML: A generic language for integrated RDF mappings of heterogeneous data," in LDOW, ser. CEUR Workshop Proceedings, vol. 1184. CEUR-WS.org, 2014.

[9] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao, "Ontop: Answering SPARQL queries over relational databases," Semantic Web, vol. 8, no. 3, pp. 471–487, 2017.

[10] F. Priyatna, Ó. Corcho, and J. F. Sequeda, "Formalisation and experiences of R2RML-based SPARQL to SQL query translation using morph," in 23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014. ACM, 2014, pp. 479–490.

[11] A. Chebotko, S. Lu, and F. Fotouhi, "Semantics preserving SPARQL-to-SQL translation," Data Knowl. Eng., vol. 68, no. 10, pp. 973–1000, 2009.

[12] S. Mandl, O. Kozachuk, M. Endres, and W. Kießling, "Preference analytics in exasolution," in Datenbanksysteme für Business, Technologie und Web (BTW), ser. LNI, vol. P-241. GI, 2015, pp. 613–632. [Online]. Available: https://dl.gi.de/20.500.12116/2434

[13] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in 19th International Conference on Data Engineering, 2003.

[14] P. Godfrey, R. Shipley, and J. Gryz, "Maximal vector computation in large data sets," in VLDB '05: Proceedings of the 31st international conference on Very large data bases, 2005, pp. 229–240.

[15] I. Bartolini, P. Ciaccia, and M. Patella, "Efficient sort-based skyline evaluation," ACM Trans. Database Syst., vol. 33, no. 4, pp. 1–49, 11 2008.

[16] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in The 28th international conference on Very Large Data Bases, 2002, pp. 275–286.

[17] M. Endres and E. Glaser, "Indexing for skyline computation," in Flexible Query Answering Systems. Cham: Springer International Publishing, 2019, pp. 31–42.

[18] S. Chaudhuri, N. Dalvi, and R. Kaushik, "Robust cardinality and cost estimation for skyline operator," in Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference on. Los Alamitos, CA, USA: IEEE Computer Society, 2006, p. 64.

[19] J. Chomicki, "Querying with intrinsic preferences," in Advances in Database Technology — EDBT 2002. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 34–51.

[20] P. F. Patel-Schneider and D. Martin, "EXISTStential aspects of SPARQL," in International Semantic Web Conference, 2016.

[21] T. Lukasiewicz, M. V. Martinez, and G. I. Simari, "Preference-based query answering in datalog+/- ontologies," in IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013, 2013, pp. 1017–1023.

[22] U. Straccia, On the Top-k Retrieval Problem for Ontology-Based Access to Databases. Cham: Springer International Publishing, 2014, pp. 95–114.

[23] P. Godfrey, "Skyline cardinality for relational processing," in Foundations of Information and Knowledge Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 78–97.

[24] E. Iglesias, S. Jozashoori, D. Chaves-Fraga, D. Collarana, and M.-E. Vidal, "SDM-RDFizer: An RML interpreter for the efficient creation of rdf knowledge graphs," in ACM Intern. Confer. on Information and Knowledge Management, CIKM, 2020.

[25] D. Chaves-Fraga, K. M. Endris, E. Iglesias, O. Corcho, and M.-E. Vidal, "What are the parameters that affect the construction of a knowledge graph?" in On the Move to Meaningful Internet Systems: OTM 2019 Conferences, H. Panetto, C. Debruyne, M. Hepp, D. Lewis, C. A. Ardagna, and R. Meersman, Eds. Cham: Springer International Publishing, 2019, pp. 695–713.

[26] D. Chaves-Fraga, F. Priyatna, A. Cimmino, J. Toledo, E. Ruckhaus, and O. Corcho, "GTFS-Madrid-Bench: A Benchmark for Virtual Knowledge Graph Access in the Transport Domain," Journal of Web Semantics, vol. 65, 2020.