Morph-Skyline: Skyline Queries for Virtual Knowledge Graph Access

Marlene Goncalves[1,2], David Chaves-Fraga[2], and Óscar Corcho[2]

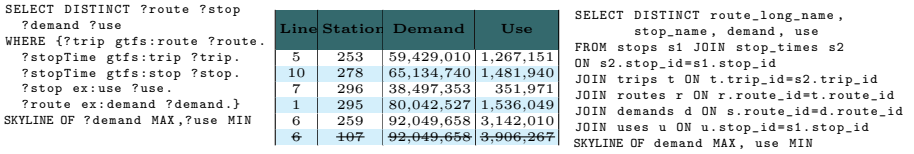
[1] Universidad Simón Bolivar, Venezuela
mgoncalves@usb.ve
[2] Ontology Engineering Group, Universidad Politécnica de Madrid
{mgoncalves,dchaves,ocorcho}@fi.upm.es

Abstract. A skyline set corresponds to the points that are non-dominated by any other points in terms of a multi-criteria function, i.e, there is no other point with values better than them in the criteria defined in a multi-criteria function. Particularly, skyline queries can be used to filter the points that best meet a multi-criteria function in decision making applications over large datasets. Most of these datasets are represented in relational databases under different schemes and data models. Integrating these datasets in a unified view may be useful for stakeholders to make more accurate decisions. As a proof of concept, we have developed Morph-Skyline, a virtual knowledge graph open source engine, which is able to automatically translate SPARQL skyline queries to equivalent ones in SQL. In this paper, we will demonstrate Morph-Skyline functionalities, and users will observe transport sector scenarios where Morph-Skyline allows specifying preferences by using skyline clause. We also evaluate the performance of Morph-Skyline against the state-of-the-art skyline methods for skyline processing over knowledge graphs. Evaluations were performed on Geonames and GTFS datasets of the subway network of Madrid with scale from 1 to 1000.

Keywords: Skyline Queries · OBDA · Query translation · R2RML.

1 Introduction

Skyline queries [1] have gained popularity in decision making applications to find a set of non-dominated data points (known as the skyline set) in a multi-dimensional dataset, i.e., a set of points such that none of them is better than the rest. Formally, a relational database is comprised of tables whose instances can be seen formally as a set of multi-dimensional points that describe each table in terms of their attributes. A skyline query consists of a list of numeric attributes or dimensions, together with the MIN or MAX directives that indicate whether the values of the corresponding dimension must be minimized or maximized. The



```
SELECT DISTINCT ?route ?stop
  ?demand ?use
WHERE {?trip gtfs:route ?route.
  ?stopTime gtfs:trip ?trip.
  ?stopTime 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

```
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
```

(c) SQL skyline Query

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 and station uses. The station 107 is dominated by the station 259.

answer of a skyline query q corresponds to the points in the multi-dimensional dataset D that are non-dominated, i.e., all the points p, such that: *i*) there is no other point p' in D with values better or equal than p in all the dimensions of p, and *ii*) other points in the skyline are better than p in at least one dimension.

The problem of efficiently computing the skyline over a database was introduced by Börzsönyi and colleagues [1] and it has been extensively studied in the literature [4]. This problem has also been studied for Semantic Web. There are some works related to extensions of SPARQL with qualitative preferences [7], which are based on a more general operator than the skyline. Finally, [5] presented a set of client-based algorithms to evaluate skyline queries over knowledge graphs using standard query interfaces for RDF although they assumed no control over how the data is stored (e.g., indexes, internal structures, etc). To the best of our knowledge, existing OBDA engines do not support skyline queries. OBDA systems are usually focused on the transformation of the original sources into a global schema and its corresponding materialization but also on query translation techniques for highly dynamic data sources [8].

To illustrate the skyline approach, suppose a database containing data from the Madrid metro system[3] following the GTFS (General Transit Feed Specification) model[4], 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.

Consider a skyline query to identify the least used stations within the most demanded lines. In this scenario, a station can be chosen, if and only 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 minimizing station uses and maximizing demands per line. Following these criteria, a SPARQL skyline is expressed in Fig. 1a and translated into SQL according to Fig. 1c. The computed skyline is presented

[3] `https://www.metromadrid.es`

[4] `https://developers.google.com/transit/gtfs`

in Fig. 1b. The stations 253, 278, 296, 295, and 259 are the non-dominated ones, i.e., there is no other station s 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. Based on related work, we devised a solution where we have developed techniques able to identify the skyline set on relational databases using an Ontology-Based Data Access (OBDA) approach [2] and extending the Chebotko et al.'s translation method [2]. Particularly, we have implemented an extension Chebotko et al.'s method which translates the preference clause where each variable in the clause is mapped to an attribute in the database. Our tool, Morph-Skyline[5], receives a skyline SPARQL query, translates it into SQL according to the defined R2RML mappings, and then executes the translated SQL query directly into the relational database engine. Under this approach, our example skyline SPARQL query (Fig. 1a) is translated to SQL (Fig. 1c) and then executed by a relational database engine.

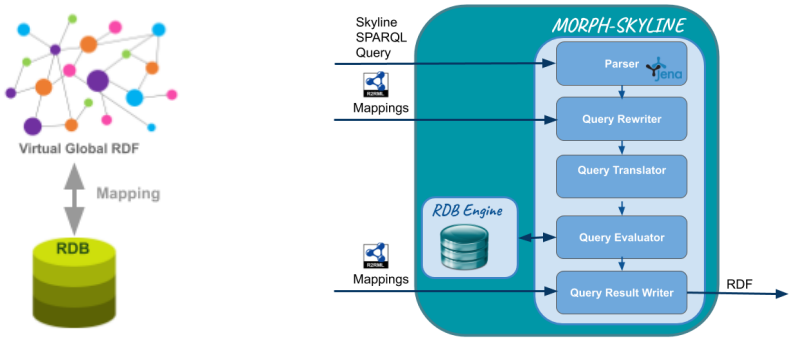
2 The Morph-Skyline Architecture

At data level, the Morph-Skyline architecture, shown in Fig. 2a, is constituted by a virtual knowledge graph, a relational schema and the mapping between them. In this architecture, the user formulates skyline SPARQL queries over the virtual knowledge graph, which are transformed into the underlying query languages of the relational database. Fig. 2b depicts the Morph-Skyline architecture at component-level. The Morph-Skyline receives a skyline SPARQL query and a set of R2RML mappings. Given a skyline SPARQL query, the parser component performs its lexical, syntactic and semantic analysis. If the skyline SPARQL query statement is correct, the Query Rewriter component prepares the query to facilitate any optimization by the Query Translator. Then, the Query Translator component uses the mappings to translate the skyline query posed over the ontology into a skyline SQL query to be executed by the underlying relational database engine. Subsequently, the query evaluator component receives the skyline query rewritten by the Query Translator and executes it over the relational database engine. Lastly, results obtained by the query evaluator are translated by the query result writer component to RDF using the rules provided in the mappings which define how ontological terms are related to terms occurring in the relational schema.

3 Demonstration of Use Cases

Within the transportation domain, the Madrid Metro web page provides its data in GTFS format together with statistics such as uses by stations and demands

[5] `https://doi.org/10.5281/zenodo.3974178`



(a) Data-level architecture (b) Component-level architecture

Fig. 2: **The Morph-Skyline Architecture**. Morph-Skyline receives a Skyline SPARQL query and outputs the results of executing the SPARQL query over Morph-RDB. Morph-Skyline translate from SPARQL to SQL to identify the skyline set which is transformed to RDF as output of a query.

by lines[6] in XLS format. We have joined these datasets in conjunction with pre-calculated distances from Madrid Geonames dataset[7]. For each station, we have pre-computed the closest distance to a school, a market, a shopping center and a hospital. This dataset was scaled-up over the scale values (5, 10, 50, 100, 500 and 1000) by using VIG[8]. For these datasets, we have created an R2RML mapping document for accessing each relational table with 81 PredicateObjectMaps and 14 JoinConditions. In the context of this demo we will show how Morph-Skyline can be used in the transportation domain where stakeholders perform SPARQL skyline queries over relational databases, so that the usefulness of skyline SPARQL queries can be better understood and demonstrated specifying preferences as criteria in a SPARQL query. We will also show the impact of the dataset and skyline sizes on the performance of Morph-Skyline and the methods based on TPF, brTPF and SkyTPF [5] to evaluate skyline queries over knowledge graphs. To evaluate the methods implemented in [5], we have transformed the datasets into RDF by means of RDFizer [3] and then, we have converted them to HDT by using the HDT library [9]. We report the average execution time in seconds for 5 SPARQL skyline queries varying the number of skyline criteria from 2 to 6. Each query was executed 5 times in cold mode on a machine with the following characteristics: 2GHz CPU with 8 cores, 64 GB RAM with Ubuntu 18.04 as its operating system. It is important to highlight that we have had to eliminate the `DISTINCT` feature in the `SELECT` clause to execute the methods based on TPF, brTPF and SkyTPF since they do not parse `SELECT DISTINCT`

[6] `https://www.metromadrid.es/es/transparencia/proyectos-y-datos\#panel1`
[7] `https://download.geonames.org/export/dump/`
[8] `https://github.com/ontop/vig`
[9] `http://www.rdfhdt.org/manual-of-the-java-hdt-library/`

statement. Our results show that Morph-Skyline time is constant as the size of the dataset increases because the size of the skyline remains small for the different sizes of datasets and queries. It is noteworthy that the best case for a skyline method is when the skyline is small [1]. Unfortunately, the methods based on TPF and brTPF return empty answers and SkyTPF it outputs an execution error when they were executed on the original dataset. For the scaled datasets, the methods based on TPF, brTPF and SkyTPF were not able to execute the queries because of an execution error. Due to the lack of space, the resources and results obtained in this real use case are publicly available in the GitHub repository of the engine[10].

4 Conclusions

Identifying the points in a database that best meet a set of user criteria can be represented as a skyline-based ranking problem. In our case, we have gone further by moving into an OBDA context. We have developed a tool that provides an efficient solution to this problem, allowing data to reside in a relational database and skyline SPARQL queries to be used. As a proof of concept, we have developed Morph-Skyline and implemented an algorithm based the Chebotko et al.'s translation method on top of a relational database. We will demonstrate the capabilities of our tool as well as the performance of our solution with respect to state-of-the-art solutions.

References

1. Borzsonyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. In: Proceedings of the 17th International Conference on Data Engineering, pp. 421-430. IEEE Computer Society. Heidelberg, Germany (2001).
2. Chebotko, A., Lu, S., Fotouhi, F.: Semantics preserving SPARQL-to-SQL translation. Data Knowl. Eng., vol. 68(10), pp. 973-1000. Elsevier Science Publishers B. V. (2009).
3. Iglesias, E., Jozashoori, S., Chaves-Fraga, D., Collarana, D., Vidal, M.E., SDM-RDFizer: An RML Interpreter for the Efficient Creation of RDF Knowledge Graphs, ACM Intern. Confer. on Information and Knowledge Management, (2020).
4. Kalyvas, C., Tzouramanis, T. A Survey of Skyline Query Processing. Computing Research Repository Journal (2017).
5. Keles, I., Hose, K.: Skyline Queries over Knowledge Graphs. In: The Semantic Web, pp. 293-310. Auckland, New Zealand (2019).
6. Mandl, S., Kozachuk, O., Endres, M., Kießling, W.: Preference Analytics in EXASolution. In: Datenbanksysteme für Business, Technologie und Web (BTW), pp. 613-632. GI. Hamburg, Germany (2015).
7. Patel-Schneider, P., Polleres, A., Martin, D.: Comparative Preferences in SPARQL. Knowl. Eng. and Knowl. Management, pp. 289-305. Nancy, France (2018)
8. Sequeda, J., Arenas, M., Miranker, D.: OBDA: Query Rewriting or Materialization? In Practice, Both!. In: The Semantic Web - ISWC, pp. 535-551. Trentino, Italy (2014).

[10] `https://github.com/oeg-upm/morph-skyline`