SPARQL with Qualitative and Quantitative Preferences
Position Paper

Marina Gueroussova[1], Axel Polleres[2], and Sheila A. McIlraith[1]

[1] Department of Computer Science, University of Toronto, Toronto, Canada
[2] Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

Abstract. The volume and diversity of data that is queriable via SPARQL and its increasing integration motivate the desire to query SPARQL information sources via the specification of preferred query outcomes. Such preference-based queries support the ordering of query outcomes with respect to a user's measure of the quality of the response. In this position paper we argue for the incorporation of preference queries into SPARQL. We propose an extension to the SPARQL query language that supports the specification of qualitative and quantitative preferences over query outcomes and examine the realization of the resulting preference-based queries via off-the-shelf SPARQL engines.

1 Introduction

Once the sole purview of IT departments, today's data is produced, shared, and consumed by a diversity of stakeholders – corporate and consumer. It is stored in a variety of formats, dynamically integrated, and queried by a variety of users, many of whom are largely unfamiliar with the content of those sources. As such, querying today's data sources using standard query languages is evolving into a manual/iterative search process, in which repeated queries are devised to hone in on outcomes that meet some criteria. Consider looking for a car to buy on the web – one might prefer a car with a powerful engine, but only if it's a hybrid, or failing that an electric car if it's under a certain price, and so on. Such a query requires not only the specification of the information to be returned, but also specification of an ordering or preference over what is returned. With these fundamental changes in the nature of data management and querying comes the need for query languages and engines that are better suited to the specification of and search for high-quality outcomes.

Preferences have been a long studied subject across many fields including economics, philosophy, and artificial intelligence. Within the database community there is a growing literature on preferences (e.g., [19]). Indeed, both Chomicki [5, 3] and independently, Kießling, Endres, and Wenzel [10] have proposed extensions to SQL that support the specification of quantitative (e.g., top-k [9]) and qualitative (e.g., skyline [1]) SQL queries. Top-k queries use a scoring function to determine an ordering over query results. In contrast, skyline queries filter a dataset with respect to a set of preference relations, returning a set of undominated tuples.

Our concern in this paper is with SPARQL [8] and with the provision of a means of succinctly specifying queries that will enable a user to search for data sources (SPARQL

endpoints) and data content that is tailored to their individual preferences, and in turn for SPARQL query engines to return ordered outcomes that reflect those preferences. Within the semantic web community, there has been significant recent work on the computation of top-k queries (e.g., [14, 2, 13, 20]), but little on how to extend the expressiveness of SPARQL to address a broad spectrum of qualitative and quantitative preferences. As such, (qualitative) preference-based querying is often realized by multiple lengthy queries that stipulate different combinations of hard constraints, or via "standard tricks[3]" such as "stacking" OPTIONAL patterns, as illustrated in the following example which preferably returns the email address of my friends, and the homepage if there is no email.

```
SELECT ?Contact WHERE { me foaf:knows ?X
                        OPTIONAL {?X foaf:mbox ?Contact}
                        OPTIONAL {?X foaf:homepage ?Contact} }
```

In 2006 Siberski, Pan, and Thaden introduced the notion of qualitative preferences into SPARQL queries [18]. In that work they realized their preference queries through the development of solution modifiers. Interestingly, their work was done before the semantics of OPTIONAL was established, and it was our hypothesis that much of what they did in 2006 with solution modifiers could be done in native SPARQL 1.0 and SPARQL 1.1 through rewriting. Building on that work, we propose an extension of the SPARQL query language, PrefSPARQL that, like Siberski et. al.'s work, builds on the vetted work on SQL preference queries, extending it here to support the expression of conditional preferences. In Section 3 we focus on the realization of qualitative preferences, and in particular on skyline and conditional preferences, showing how they can be rewritten into SPARQL 1.1 (and also SPARQL 1.0) and thus realized by existing SPARQL query engines. The work presented here is only the first step of a larger endeavour that will see the extension of the presented query grammar with a number of interesting features, and the development of optimized query processing techniques that are tailored to the efficient computation of preference-based SPARQL queries.

2 PrefSPARQL Syntax

In this section we propose a core grammar for PrefSPARQL that addresses a selection of qualitative and quantitative preferences in support of specifying preferred SPARQL query outcomes. We illustrate our grammar with respect to skyline and conditional preference queries. Skyline queries for relational databases have been the subject of significant research (e.g., [5, 1]), and refer to a set of results that are no worse than any other result across all dimensions of a set of independent boolean or numerical preferences [1]. Skyline queries have been studied extensively in the SQL context by Chomicki [4, 5], and by Kießling, Endres, and Wenzel as an essential part of their *Preference SQL* language and system [10, 11].

We build upon an earlier approach to adopt features of *Preference SQL* in SPARQL, by Siberski, Pan, and Thaden [18]. In particular, we extend the SPARQL query language in a similar fashion to the proposal in [18]; we also use 'AND' to separate

[3] e.g., http://answers.semanticweb.com/questions/20682/preference-patterns-for-sparql-11

independent dimensions of skyline queries. The key differences in our proposal are that, firstly, we add preferences at the level of filters (production 68 of the SPARQL grammar [8, Section 19]) rather than as solution modifiers, with the justification that preferences semantically *filter* the solution set rather than 'ordering' or 'slicing' it. This approach makes preferences usable inside any patterns in a nested fashion as opposed to just at the end of queries.[4] Secondly, we follow the latest version of *Preference SQL* [10] in replacing 'CASCADE' with 'PRIOR TO'. Furthermore, we support additional atomic preference constructs such as 'AROUND', 'MORE THAN', 'LESS THAN', and 'BETWEEN'; 'x BETWEEN (Low,$High$)' differs from writing '$((x >= Low)$ && $(x <= High))$' in SPARQL in that – in the absence of a value in the chosen interval – 'BETWEEN' will return the closest value to Low (or $High$, resp.); AROUND(x), MORE THAN(x), and LESS THAN(x) are analogous to BETWEEN(x, x), BETWEEN(x, ∞) and, BETWEEN($-\infty, x$) respectively. Finally, we augment the grammar with conditional (IF-THEN-ELSE) preferences. We note that, given the availability of conditional preferences, BETWEEN (as well as AROUND, MORE THAN, and LESS THAN) come for free[5].

```
Filter ::= 'FILTER' Constraint
          | 'PREFERRING' '(' MultidimensionalPref ')'
MultidimensionalPref ::= PrioritizedPref ('AND' PrioritizedPref)*
PrioritizedPref ::= ConditionalOrAtomicPref
                    ('PRIOR TO' ConditionalOrAtomicPref)*
ConditionalOrAtomicPref ::= ConditionalPref | AtomicPref
ConditionalPref ::= 'IF' Expression
                    'THEN' ConditionalOrAtomicPref
                    'ELSE' ConditionalOrAtomicPref
AtomicPref  ::= Expression | HighestPref | LowestPref | AroundPref
              | BetweenPref | MoreThanPref | LessThanPref
HighestPref ::= 'HIGHEST' Expression
LowestPref  ::= 'LOWEST' Expression
AroundPref  ::= Expression 'AROUND' Expression
BetweenPref ::= Expression 'BETWEEN' '(' Expression ',' Expression ')'
MoreThanPref ::= Expression 'MORE THAN' Expression
LessThanPref ::= Expression 'LESS THAN' Expression
```

To illustrate this, consider a modification of the example from [18]. The knowledge base contains therapist ratings and their appointment offerings (in Turtle syntax).

```
@prefix : <http://www.example.org/>.
:mary a :therapist ; :rated :excellent ;
        :offers :appointment1, :appointment2 .
:appointment1 :day "Tuesday"; :starts 1500; :ends  1555 .
:appointment2 :day "Sunday"; :starts 1600; :ends 1655 .
```

[4] While with the addition of subqueries in SPARQL1.1 this does not add to language expressivity, it still allows one to write certain preference queries more concisely.

[5] x BETWEEN(Low,$High$) can be viewed as syntactic sugar for LOWEST (IF $x >= Low$ && $x <= High$ THEN 0 ELSE IF $x < Low$ THEN $Abs(Low-x)$ ELSE $Abs(High-x)$).

For the case of the skyline preference, query Q1 prefers excellent therapists, appointments around lunchtime (between 12:00 and 13:00) over those outside lunchtime (written as a BETWEEN preference), and later appointments over earlier ones provided that both are equal with respect to lunchtime.

```
SELECT ?A WHERE {
   ?T :rated ?R; :offers ?A. ?A :starts ?S; :ends ?E .
   PREFERRING ( ?R = excellent AND
                ( ?S BETWEEN(1200,1300) AND ?E BETWEEN(1200,1300)
                  PRIOR TO HIGHEST ?E ) ) }
```

For the case of conditional preferences, in query Q2 we prefer appointments before 6PM on the weekends, and appointments after 6PM on the weekdays.

```
SELECT ?A WHERE { ?A :day ?D; :starts ?S.
                PREFERRING ( IF (?D = "Saturday" || ?D = "Sunday")
                THEN ?S < 1800 ELSE ?S >= 1800 ) }
```

3 Realizing PrefSPARQL Through Query Rewriting

As opposed to a contrary conjecture in [18], we argue that preferences such as the ones presented in Section 2 *can* be expressed in SPARQL itself by the following high-level translation schema, where P is a SPARQL pattern and $Pref$ is a 'PREFERRING' clause as defined in the grammar above.

$$\frac{P \texttt{ PREFERRING } Pref}{P \texttt{ FILTER NOT EXISTS } \{P' \texttt{ FILTER } (tr(P, P', Pref))\}}$$

Here, P' is a copy of P where all variables are renamed with fresh variables. The high-level idea here is that we reformulate PREFERRING clauses as FILTERs, asking for the non-existence of a solution to the pattern P' which dominates the solution of P according to $Pref$. This simple recipe, along with a translation of the domination relation as a FILTER expression, suffices to express the intended semantics of returning only dominating solutions.

While we do not provide the full translation function $tr(P, P', Pref)$ of PREFERRING clauses in this short position paper (see our technical report [7] for the translation), let us illustrate the feasibility by means of some examples. As for skyline queries, we take a simplified version of Q1 from above without BETWEEN,[6] where we prefer appointments outside rush hour, i.e., either starting after 18:00 or ending before 16:00, instead of over lunchtime.

```
SELECT ?A
WHERE {
   ?T :rated ?R; :offers ?A. ?A :starts ?S; :ends ?E .
   PREFERRING ( (?R = excellent) AND
                ((?S >= 1800) || ?E <= 1600)) PRIOR TO
                HIGHEST ?S )}
```

[6] As mentioned before BETWEEN can be translated using conditional preferences, which we will illustrate with the second example.

This query translates to the following SPARQL1.1 query:

```
1 SELECT ?A WHERE { ?T :rated ?R; :offers ?A. ?A :starts ?S; :ends ?E .
2   BIND ( (?R = pt:excellent)        AS ?Pref1 )
3   BIND ( (?S >= 1800 || ?E <= 1600) AS ?Pref2 )
4   BIND ( ?S                         AS ?Pref3 )
5   NOT EXISTS {
6      ?T_ :rated ?R_; :offers ?A_. ?A_ :starts ?S_; :ends ?E_ .
7      BIND ( (?R_ = :excellent)           AS ?Pref1_ )
8      BIND ( (?S_ >= 1600 || ?E_ <= 1600) AS ?Pref2_ )
9      BIND ( ?S_                          AS ?Pref3_ )
10     FILTER(
11        (  (?Pref1_ > ?Pref1) &&
12          !((?Pref2_ < ?Pref2) || (?Pref3_ < ?Pref3 && ?Pref2 = ?Pref2_ )))
13        ||
14        ( !(?Pref1_ < ?Pref1) &&
15           ((?Pref2_ > ?Pref2) || (?Pref3_ > ?Pref3 && ?Pref2 = ?Pref2_ )))) }}
```

In the copied pattern P' we replace every variable by a 'copy' appending '_' to the variable name. In both P and P' we bind every atomic expression in $Pref$ to a new variable (lines 2-4 and 7-9); then we use these variables to build up a FILTER expression that filters out 'dominating' witnesses for P; if no such witness exists, P survives. Let us explain briefly the rationale behind the translation of preference dominance in lines 11-15: the two dimensions (AND) of the skyline preference – i.e., that a dominating solution is better in at least one dimension and no worse in the others – are encoded in the two branches in lines 11-12 and lines 14-15, whereas the nested cascading (PRIOR TO) preference between $Pref2$ and $Pref3$ is encoded in the boolean expressions in lines 12 and 15, respectively.

Let us emphasize that the translation would also work without BINDs; the respective expressions could just be copied, although this would make the translated query more confusing. Similarly, `NOT EXISTS` could be replaced by a well-known combination of `OPTIONAL` and `FILTER(!bound)` (cf. [17, Section 11.4.1]) to emulate set difference, which would make the whole query also expressible in SPARQL1.0.

Conditional preferences can be encoded in SPARQL1.1 conveniently using the `IF`$(\cdot,\cdot,\cdot)$ function in filters, as shown in the translation of query `Q2` from above.

```
1 SELECT ?A WHERE {
2   ?T :rated ?R; :offers ?A. ?A :day ?D; :starts ?S.
3   BIND ( IF( (?D = "Sunday" || ?D = "Saturday"), ?S < 1800, ?S >= 1800)
4          AS ?Pref1)
5   NOT EXISTS { ?T_ :rated ?R_; :offers ?A_. ?A_ :day ?D_; :starts ?S_.
6      BIND ( IF( (?D_ = "Sunday" || ?D_ = "Saturday"), ?S_ < 1800, ?S_ >= 1800)
7           AS ?Pref1_)
8      FILTER (?Pref1_ > ?Pref1)}}
```

We note that the `IF`$(\cdot,\cdot,\cdot)$ function is – in principle – again syntactic sugar that can be emulated in SPARQL1.0 as well, such that in summary our presented preferences can be both implemented in rewritings to SPARQL1.1 and SPARQL1.0.

4 Summary and Discussion

In this position paper we argued for (re-)considering preferences in SPARQL queries. Given the vast amount of data being subjected to SPARQL queries, we can conceive many examples where complex preferences would be needed to find the "needle in the

haystack", with the user specifying preferences not only over query results but also over the sources (SPARQL endpoints) and provenance of data used to produce those results. Here we proposed a core grammar for PrefSPARQL, an extension to SPARQL 1.1 that supports the expression of preferred query results. Our language builds on established work on SQL preferences and on an earlier effort by Siberski et al. [18], extending it with conditional preferences. We further argued, contrary to the conjecture of Siberski et al., that these preference queries can be directly expressed in both SPARQL1.0 and SPARQL1.1 using OPTIONAL queries or novel features of SPARQL1.1, such as NOT EXISTS. We illustrated such a realization with respect to skyline and conditional preference queries. We acknowledge that, at the time Siberski and colleagues' work was performed, the semantics of SPARQL1.0 was not yet fully defined and SPARQL1.1 was still far off on the horizon.

Nevertheless, we argue that this topic needs further attention, since preference queries implemented naively by rewriting in SPARQL might become prohibitively costly. In particular, we emphasize that all the examples we gave in this paper, even those expressing simple preferences by "stacking OPTIONALs", as mentioned in Section 1, or, respectively all our example translations would produce so called non-well-designed patterns [15, 12] in SPARQL. We therefore plan to further investigate relaxations of the well-designedness restriction, which still enable efficiently evaluable preference queries.

The specification and efficient realization of preference-based SPARQL queries is an important topic that is worthy of further exploration. As this work is ongoing, considerations for expanding it include the addition of quantitative preferences in the form of top-k queries, ranking within a skyline, preferences over endpoints in the context of the SPARQL1.1 Federation extension [16], and SPARQL endpoint discovery (e.g. by preferences over Service descriptions [21]) as well as interaction of preferences with Entailment Regimes [6]. Further details relating to this paper can be found in [7].

Acknowledgements

We wish to thank Wolf Siberski, Jeff Pan, and Florian Wenzel for their assistance. This work has been partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), an Ontario Graduate Scholarship (OGS), and by the Vienna Science and Technology Fund (WWTF) through project ICT12-015.

References

[1] Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proc. of the 17th Int'l Conference on Data Engineering (ICDE). pp. 421–430 (2001)

[2] Bozzon, A., Valle, E.D., Magliacane, S.: Extending SPARQL algebra to support efficient evaluation of top-k SPARQL queries. In: Ceri, S., Brambilla, M. (eds.) Search Computing – Broadening Web Search, pp. 143–156. Springer Lecture Notes in Computer Science (2012)

[3] Chomicki, J.: Querying with intrinsic preferences. In: Proc. of the 8th Int'l Conference on Extending Database Technology (EDBT). pp. 34–51 (2002)

[4] Chomicki, J.: Preference formulas in relational queries. ACM Trans. on Database Systems (TODS) 28(4), 427–466 (2003)

[5] Chomicki, J.: Logical foundations of preference queries. IEEE Data Engineering Bulletin 34(2), 3–10 (2011)
[6] Glimm, B., Ogbuji, C.: SPARQL 1.1 Entailment Regimes (2013), W3C Recommendation
[7] Gueroussova, M., Polleres, A., McIlraith, S.A.: SPARQL with qualitative and quantitative preferences (extended report). Tech. Rep. CSRG-619, Department of Computer Science, University of Toronto (October 2013)
[8] Harris, S., Seaborne, A.: SPARQL 1.1 Query Language (2013), W3C Recommendation
[9] Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top-k query processing techniques in relational database systems. ACM Computing Surveys 40(4), 11:1–11:58 (Oct 2008)
[10] Kießling, W., Endres, M., Wenzel, F.: The preference SQL system - an overview. IEEE Data Engineering Bulletin 34(2), 11–18 (2011)
[11] Kießling, W., Köstler, G.: Preference SQL - design, implementation, experiences. In: Proc. of 28th Int'l Conference on Very Large Data Bases (VLDB). pp. 990–1001 (2002)
[12] Letelier, A., Pérez, J., Pichler, R., Skritek, S.: Static analysis and optimization of semantic web queries. In: Proc. of the 31st Symposium on Principles of Database Systems (PODS). pp. 89–100 (2012)
[13] Magliacane, S., Bozzon, A., Valle, E.D.: Efficient execution of top-k SPARQL queries. In: Proc. of the 11th Int'l Semantic Web Conference (ISWC). pp. 344–360 (2012)
[14] Montoya, G., Vidal, M.E., Corcho, Ó., Ruckhaus, E., Aranda, C.B.: Benchmarking federated SPARQL query engines: Are existing testbeds enough? In: Proc. of the 11th Int'l Semantic Web Conference (ISWC). pp. 313–324 (2012)
[15] Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Trans. on Database Systems (TODS) 34(3) (2009)
[16] Prud'hommeaux, E., Buil-Aranda, C.: SPARQL 1.1 Federated Query (2013), W3C Recommendation
[17] Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF (2008), W3C Recommendation
[18] Siberski, W., Pan, J.Z., Thaden, U.: Querying the semantic web with preferences. In: International Semantic Web Conference. pp. 612–624 (2006)
[19] Stefanidis, K., Koutrika, G., Pitoura, E.: A survey on representation, composition and application of preferences in database systems. ACM Trans. on Database Systems (TODS) 36(3), 19 (2011)
[20] Wagner, A., Tran, D.T., Ladwig, G., Harth, A., Studer, R.: Top-k linked data query processing. In: Proceedings of the 9th Extended Semantic Web Conference (ESWC). pp. 56–71 (2012)
[21] Williams, G.: SPARQL 1.1 Service Description (2013), W3C Recommendation