Interacting with Statistical Linked Data via OLAP Operations

Benedikt Kämpgen[1], Sean O'Riain[2], and Andreas Harth[1]

[1] Institute AIFB, Karlsruhe Institute of Technology, Karlsruhe, Germany
{benedikt.kaempgen,harth}@kit.edu
[2] Digital Enterprise Research Institute, National University of Ireland, Galway
sean.oriain@deri.org

Abstract. Online Analytical Processing (OLAP) promises an interface to analyse Linked Data containing statistics going beyond other interaction paradigms such as follow-your-nose browsers, faceted-search interfaces and query builders. As a new way to interact with statistical Linked Data we define comon OLAP operations on data cubes modelled in RDF and show how a nested set of OLAP operations lead to an OLAP query. Then, we show how to transform an OLAP query to a SPARQL query which generates all required facts from the data cube. Both metadata and OLAP queries are issued directly on a triple store; therefore, if the RDF is modified or updated, changes are propagated directly to OLAP clients.

Keywords: OLAP, query, operation, Linked Data, statistics, XBRL

1 Introduction

Linked Data provides easy access to large amounts of interesting statistics from many organizations for information integration and decision support, including financial information from institutions such as the UK government[3] and the U.S. Securities and Exchange Commission.[4] However, interaction paradigms for Linked Data such as follow-your-nose browsers, faceted-search interfaces, and query builders [10, 12] do not allow users to analyse large amounts of numerical data in an exploratory fashion of "overview first, zoom and filter, then details-on-demand" [24]. Online Analytical Processing (OLAP) operations on data cubes

In: C. Unger, P. Cimiano, V. Lopez, E. Motta, P. Buitelaar, R. Cyganiak (eds.): Proceedings of Interacting with Linked Data (ILD 2012), Workshop co-located with the 9th Extended Semantic Web Conference, Heraklion, Greece, 28-05-2012, published at http://ceur-ws.org

[3] http://data.gov.uk/resources/coins
[4] http://edgarwrap.ontologycentral.com/

for viewing statistics from different angles and granularities, filtering for specific features, and comparing aggregated measures fulfil this information seeking mantra and provide interfaces for decision-support from statistics [2,4,25]. However OLAP on statistical Linked Data imposes two main challenges:

- OLAP requires a model of data cubes, dimensions, and measures. Automatically creating such a multidimensional schema from generic Linked Data is difficult, and only semi-automatic methods have proved applicable [16,18,21]. The RDF Data Cube vocabulary (QB)[5] is a Linked Data vocabulary to model statistics in RDF. Several publishers have already used the vocabulary for statistical datasets.[6]
- OLAP queries are complex and require specialised data models, e.g., star schemas in relational databases, to be executed efficiently [9]. The typical architecture of an OLAP system consists of an ETL pipeline that extracts, transforms and loads data from the data sources into a data warehouse, e.g., a relational or multidimensional database. OLAP clients such as JPivot allow users to built OLAP queries and display results in pivot tables. An OLAP engine, e.g., Mondrian, transforms OLAP queries into queries to the data warehouse, and deploys mechanisms for fast data cube computation and selection, under the additional complexity that data in the data warehouse may change dynamically [8, 14].

In this paper, we assume that statistical Linked Data has been modelled using QB and focus on efficiently executing OLAP queries on statistical Linked Data. In previous work [11] we have presented a proof-of-concept to interpret Linked Data reusing QB as a multidimensional model and to automatically load the data into a data warehouse used by common OLAP systems. The fact that OLAP queries are executed not on the RDF directly but by a common OLAP engine after automatically populating a data warehouse result in two drawbacks: first, although the relational star schema we adopted is a quasi-standard logical model for data warehouses, our approach requires a ROLAP engine to execute OLAP queries; second, if statistical Linked Data is updated, e.g., if a single new row is added, the entire ETL process has to be repeated, to have the changes propagated. Figure 1 shows our new data flow of having an OLAP engine issuing SPARQL queries directly to a triple store.

The current work presents a new way to interact with statistical Linked Data:

- We define comon OLAP operations on data cubes as Linked Data reusing QB and show how a nested set of OLAP operations lead to an OLAP query.
- We show how to transform an OLAP query to a SPARQL query which generates all required facts from the data cube.

In the remainder of the paper, we first present a motivational scenario from the financial domain in Section 2. As a prerequisite for our contribution, in

[5] `http://www.w3.org/TR/2012/WD-vocab-data-cube-20120405/`
[6] `http://wiki.planet-data.eu/web/Datasets`

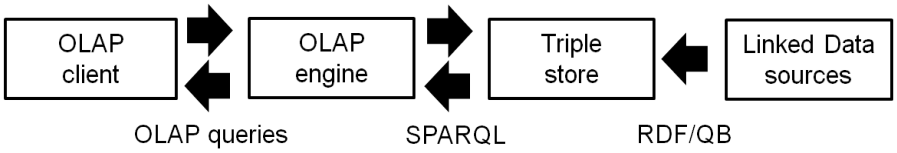

Fig. 1. Data flow for OLAP queries on statistical Linked Data in a triple store

Section 3, we formally define a multidimensional model of data cubes based on QB. Then, in Section 4, we introduce OLAP operations on data cubes and present a direct mapping of OLAP to SPARQL queries. We apply this mapping in a small experiment in Section 5 and discuss some lessons learned in Section 6. In Section 7, we describe related work, after which, in Section 5, we conclude and describe future research.

2 Scenario: Analysing Financial Linked Data

In this section we describe a scenario of analysing Linked Data containing financial information. The Edgar Linked Data Wrapper[7] provides access to XBRL filings[8] as Linked Data reusing QB. Those filings disclose balance sheets of a large number of US organizations, for instance that *RAYONIER INC* had a *sales revenue net* of 377,515,000 USD from 2010-07-01 to 2010-09-30.[9]

Using LDSpider, we crawled Linked Data from the Edgar wrapper and stored a data cube *SecCubeGrossProfitMargin* into an Open Virtuoso triple store. The data cube contains single disclosures from financial companies such as *RAYONIER INC*. Each disclosure either discloses cost of goods sold (CostOfGoodsSold) or sales revenue net (Sales) as measures. The two measures have the unit USD and an aggregation function that returns the number of disclosures, or - if only one - the actual number. Any disclosure is fully dependent on the following dimensions: the disclosing company (Issuer), the date a disclosure started (Dtstart) and ended (Dtend) to be valid, and additional segment information (Segment).

In our scenario, a business analyst wants to compare the number of disclosures of cost of goods sold for two companies. He requests a pivot table with issuers *RAYONIER INC* and *WEYERHAEUSER CO* on the columns, and the possible periods for which disclosures are valid on the rows, and in the cells showing the number of disclosed cost of goods sold, or - if only one - the actual number. Figure 2 shows the needed pivot table.

[7] `http://edgarwrap.ontologycentral.com/`
[8] `http://www.xbrl.org/Specification/XBRL-RECOMMENDATION-2003-12-31+Corrected-Errata-2008-07-02.htm`
[9] `http://edgarwrap.ontologycentral.com/archive/52827/0001193125-10-238973#ds`

Columns (issuer) Rows (dtstart, dtend)		RAYONIER INC	WEYERHAEUSER CO
2009-01-01	2009-3-31	*1,100,335 USD*	*0 values*
2009-04-01	2009-06-30	*2 values*	*2,300,800 USD*
...	...	...	...

Filters: CostOfGoodsSold

Fig. 2. Pivot table to be filled in our scenario

3 A Multidimensional Model Based on QB

In this section, as a precondition for OLAP queries on Linked Data, we formally define the notion of data cubes in terms of QB. The definition is based on the multidimensional models of Gómez et al. [7], Pedersen et al. [20] and the Open Java API for OLAP[10] as well as on the Linked Data vocabularies of QB, SKOS[11] and skosclass.[12]

Definition 1 (Linked Data store with terms and triples). *The set of RDF terms in a triple store consists of the set of IRIs $\mathcal{I}$, the set of blank nodes $\mathcal{B}$ and the set of literals $\mathcal{L}$. A triple $(s, p, o) \in \mathcal{T} = (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$ is called an RDF triple, where s is the subject, p is the predicate and o is the object.*

Given a triple store with statistical Linked Data, we use basic SPARQL triple patterns on the store to help us defining sets of multidimensional elements. Given a multidimensional element x $iri(x) \in (\mathcal{I} \cup \mathcal{B})$ returns its IRI or blank node:

Member defines the set of members as $Member = \{?x \in (\mathcal{I} \cup \mathcal{B}) | (?x \text{ a } skos{:}Concept)\}$. Let $\mathcal{V} = 2^{Member}$, $V \in \mathcal{V}$, $ROLLUPMEMBER \subseteq Member \times Member$, $rollupmember(V) = \{(v_1, v_2) \in V \times V | (iri(v_1)\ skos{:}broaderiri(v_2) \vee iri(v_2)\ skos{:}narrower\ iri(v_1))\}$

Level defines the set of levels as $Level = \{(?x, V, rolluplevel(V)) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{V} \times ROLLUPLEVEL | (?x \text{ a } skosclass{:}ClassificationLevel \wedge \forall v \in V(iri(v)\ skos{:}member\ ?x)\}$. Let $\mathcal{L} = 2^{Level}$, $L \in \mathcal{L}$, $ROLLUPLEVEL \subseteq Level \times Level$, $rolluplevel(L) = \{(l_1, l_2) \in L \times L | (iri(l_1)\ skosclass{:}depth\ x) \wedge (iri(l_2)\ skosclass{:}depth\ y) \wedge x \leq y))\}$

Hierarchy defines the set of hierarchies as $Hierarchy = \{(?x, L, rolluplevel(L)) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{L} \times ROLLUPLEVEL | (?x \text{ a } skos{:}ConceptScheme\ ) \wedge \forall l \in L(iri(l)\ skos{:}inScheme\ ?x)\}$. Let $\mathcal{H} = 2^{Hierarchy}$.

[10] http://www.olap4j.org/
[11] http://www.w3.org/2004/02/skos/
[12] http://www.w3.org/2011/gld/wiki/ISO_Extensions_to_SKOS

Dimension defines the set of dimensions as $Dimension = \{(?x, H) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{H} | (?x \text{ a } qb{:}DimensionProperty\ ) \wedge \forall h \in H(?x\ qb{:}codeList\ iri(h))\}$. Let $\mathcal{D} = 2^{Dimension}$.

Measure defines the set of measures as $Measure = \{(?x, aggr) \in (\mathcal{I} \cup \mathcal{B}) \times \{UDF\} | (?x \text{ a } qb{:}MeasureProperty\ )\}$ with UDF a default aggregation function since QB so far does not provide a standard way to represent typical aggregation functions such as SUM, AVG and COUNT: if only one value is given, the value itself else the number of values is returned. Conceptually, measures are treated as members of a dimension-hierarchy-level combination labelled "Measures". Let $\mathcal{M} = 2^{Measure}$.

DataCubeSchema defines the set of data cube schemas as $\{(?x, D, M) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{D} \times \mathcal{M} | (?x \text{ a } qb{:}DataStructureDefinition \wedge \forall d \in D(?x\ qb{:}componentProperty\ ?comp \wedge ?comp\ qb{:}dimensionProperty\ iri(d)) \wedge \forall m \in M(?x\ qb{:}componentProperty\ ?comp \wedge ?comp\ qb{:}measureProperty\ iri(m)))\}$.

Fact defines the set of possible statistical facts as $Fact = \{(?x, ?c_0, \ldots, ?c_i, ?e_0, \ldots, ?e_j) \in (\mathcal{I} \cup \mathcal{B}) \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L}) \times \ldots \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L}) \times \mathcal{L} \times \ldots \times \mathcal{L} | (?x \text{ a } qb{:}Observation\ ) \wedge (?x ?d_0 c_0 \wedge ?d_0 \text{ a } qb{:}DimensionProperty) \wedge \ldots \wedge (?x ?d_i c_i \wedge ?d_i \text{ a } qb{:}DimensionProperty\ ) \wedge (?x ?m_0 e_0 \wedge ?m_0 \text{ a } qb{:}MeasureProperty) \wedge \ldots \wedge (?x ?m_j e_j \wedge ?m_j \text{ a } qb{:}MeasureProperty)\}$. Let $\mathcal{F} = 2^{Fact}$.

DataCube defines the set of data cubes as $DataCube = \{(cs, F) \in DataCubeSchema \times \mathcal{F} | cs = (?x, D, M) \wedge D = \{D_0, \ldots, D_{|D|}\} \wedge M = \{m_0, \ldots, m_{|M|}\} \wedge \forall c \in F(c = (?obs, c_0, \ldots, c_{|D|}, e_0, \ldots, e_{|M|}) : (?obs\ qb{:}dataSet\ ?ds \wedge ?ds\ qb{:}structure\ ?x) \wedge \forall D_i \in D(?x iri(D_i) c_i \wedge iri(D_i)\ qb{:}codeList\ ?h \wedge ?l\ skos{:}inSchme\ ?h \wedge ?v\ skos{:}member\ ?l \wedge (?v\ skos{:}notation\ ?c_i \vee ?v\ skos{:}exactMatch\ ?c_i)) \wedge \forall m_i \in M(?x iri(m_i) e_i \vee e_i = null)$.

We distinguish *metadata queries* and *OLAP queries* on data cubes. Whereas metadata queries return multidimensional objects such as the cube schema, the dimensions, and the measures, OLAP queries return facts directly contained in or derived from (e.g., by aggregation of several facts) the data cube.

Definition 1. *According to Gray et al. [8] a materialised data cube* (cs, F) *with* $cs = (x, D, M)$ *contains a set of facts* $MF = (?x, ?c_0, \ldots, ?c_{|D|}, ?e_0, \ldots, ?e_{|M|})$ *with* $?c_i \in C_i$ *with* $C_i : 0 \leq i \leq |D|$ *all possible members of a dimension* $D_i \in D$ *or the special* ALL *value, and with* $e_j \in T : 0 \leq j \leq |M|$ *with* T *a numeric domain including the special* null *value in cases of cube sparsity. A data cube can be materialised by a union of* $2^{|D|}$ *sub-queries, each grouping the result on the members of a subset of dimensions, replacing the values of the other dimensions by a special* ALL *value, and computing each measure value by the measure's aggregation function. The number of facts from a materialised data cube is given by* $|MF| = (|C_0| + 1) \cdot \ldots \cdot (|C_{|D|}| + 1)$.

Subqueries and aggregation functions in SPARQL 1.1 make easily possible the concept of Gray et al. [8] to fully materialise a data cube represented as Linked Data reusing QB. However, due to its exponential size w.r.t. the number of dimensions, such a SPARQL query is inefficient. OLAP queries may require only a small subset of all possible facts of a data cube; therefore, in the next section, we show how to evaluate OLAP queries using a single SPARQL query.

4 Mapping OLAP Operations to SPARQL on QB

In this section we show how to issue OLAP queries on a multidimensional model. We define common OLAP operations on single data cubes [19,20,22,23]. A nested set of OLAP operations lead to an OLAP query. We describe how to evaluate such an OLAP query using SPARQL on QB. Figure 3 illustrates the effect of common OLAP operations, with inputs and outputs.

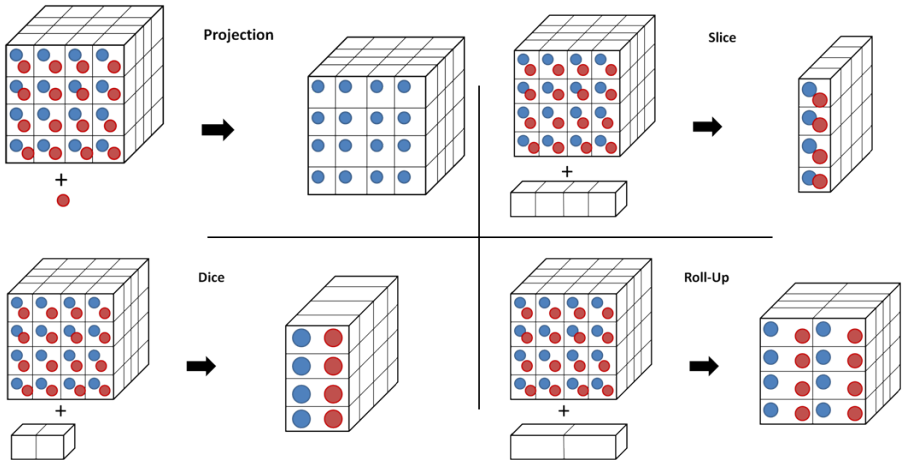

Fig. 3. Illustration of common OLAP operations with inputs and outputs (adapted from [23])

Note, this paper focuses on direct querying of single data cubes, the integration of several data cubes through *Drill-Across* or set-oriented operations such as union, intersection, difference is out-of-scope. Multiple datasets can already be queried together if they are covered by one *qb:DataStructureDefinition*.

Each OLAP operation has as input and output a data cube. Therefore, operations can be nested. A nested set of OLAP operations lead to an OLAP query. For interpreting a set of OLAP operations as an OLAP query and evaluate it using a SPARQL query on QB, we adopt the notion of *subcube queries* [13].

Definition 2. *We define an OLAP query as a subcube query [13] on a certain cube* (cs, C) *with* $cs = (?uri, D, M)$*, represented as a tuple with as many elements as dimensions and measures:* $(q_0, ..., q_{|D|}, m_0, ..., m_{|M|})$ *with* $dom(q_i) = \{?, ALL, x\}$ *with* ? *for an* inquired *dimension, with* ALL *for an aggregated dimension, with* x *for one or more members to fix a dimension, and with* m_i *a measure to query for.*

As examples, we describe three distinguishable subcube queries:

– A *full-cube query* returns exactly the tuples from a *DataCubeInstance* and inquires all dimensions: $(?, ?, ?, ..., m_0, ..., m_{|M|})$.

- A *point query* asks for a data cube comprising one specific instance tuple: $(a_0, a_1, ..., a_{|D|}, m_0, ..., m_{|M|})$ with a_i a member of a dimension D_i.
- A *fully-aggregated query* asks for the measures aggregated over all dimensions: $(ALL, ALL, \ldots, ALL, m_0, \ldots, m_{|M|})$.

In the following we describe how to evaluate each OLAP operation in terms of this query model and how a nested set of OLAP operations results in one specific OLAP subcube query. We assume that names and schemas of input and output data cubes are implicitly given and we focus on the data cube facts which will be queried for. An input data cube is represented as a full-cube query $(?, ?, ?, ..., m_0, ..., m_{|M|})$.

Projection is defined as $Projection : DataCube \times Measure \rightarrow DataCube$ and removes a measure from the input cube and allows to query only for specific measures. We evaluate $Projection$ by removing a measure from the subcube query tuple.

Slice is defined as $Slice : DataCube \times Dimension \rightarrow DataCube$ and removes a dimension from the input cube and aggregates over the members of a dimension. We evaluate *Slice* by setting the tuple element of that dimension to ALL.

Dice is defined as $Dice : DataCube \times Dimension \times \mathcal{V} \rightarrow DataCube$ and allows to filter for and aggregate over certain dimension members. We evaluate $Dice$ by setting the tuple element of that dimension to this particular member or set of members and aggregate over the set. Note, dice is not a selection operation but a combined filter and slice operation.

Roll-Up is defined as $Roll-Up : DataCube \times Dimension \times Level \rightarrow DataCube$ and allows to create a cube that contains instance data on a higher aggregation level. We evaluate roll-up by replacing the inquired members of a dimension with members of a higher level. Note, we do not define $Drill - Down$, since it can be seen as an inverse operation to $Roll - Up$.

As an example, consider an OLAP query on our *SecCubeGrossProfitMargin* cube for the cost of goods sold (CostOfGoodsSold) for each issuer and each date until when each disclosure is valid (dtend), filtering by disclosures from two specific segments. A nested set of OLAP operations that queries the requested facts can be composed as follows. In all our queries, we use prefixes to make URIs more readable:

```
Slice(
        Dice(
                Projection(
                edgar:SecCubeGrossProfitMargin,
                edgar:CostOfGoodsSold),
        edgar:segment,
        {edgar:segmentAHealthCareInsuranceCompany,
        edgar:
            segmentAResidentialRealEstateDeveloperMember})
            ,
```

```
edgar:dtstart)
```

This query can then be represented as a subcube query with dimensions Issuer, Dtstart, Dtend, Segment:

```
(?, *, ?, {edgar:segmentAHealthCareInsuranceCompany,
edgar:segmentAResidentialRealEstateDeveloperMember},
    edgar:CostOfGoodsSold)
```

Next, we describe how to evaluate such an OLAP query using a SPARQL query on QB. Since OLAP hierarchies add considerable complexity to QB and since a $Roll-Up$ has a similar effect to a *Slice* operation, in this paper, we assume data cubes with only one hierarchy and level per dimension. A subcube query $Q = (q_0, \ldots, q_{|D|}, m_0, ..., m_{|M|})$ can be translated into a SPARQL query using the following steps:

1. We initialise the SPARQL query using the URI of the data cube. We query for all instance data from the data cube, i.e., observations linking to datasets which link to the data structure definition.
2. For each selected measure, we incorporate it in the SPARQL query by selecting additional variables for each measure and by aggregating them using the aggregation function of that measure, using OPTIONAL patterns in case we query for several measures.
3. For each inquired dimension, we query for all the instances of *skos:Concept* in a level of a hierarchy of the dimension and for the represented values used (from members either linked via *skos:notation* or *skos:exactMatch*) in the observations. We query for the observations showing property-value pairs for each of these variables. To display inquired dimensions in the result and correctly aggregating the measures, we *group by* each dimension variable.
4. For each fixed dimension, we filter for those observations that exhibit for each dimension one of the listed members.

We transform our example from above to the following SPARQL query. Note, "UDF" represents the standard aggregation function from our scenario:

```
select ?dimMem0 ?dimMem1 UDF(?measureValues0) where {
?obs qb:dataSet ?ds.
?ds qb:structure edgar:SecCubeGrossProfitMargin.
?obs edgar:issuer ?values0.
        ?dimMem0 skos:member ?level0.
        ?level0 skos:inScheme ?hierarchy0.
        edgar:issuer qb:codeList ?hierarchy0.
        ?dimMem0 skos:exactMatch ?values0.
?obs edgar:dtend ?values1.
        ?dimMem1 skos:member ?level1.
        ?level1 skos:inScheme ?hierarchy1.
        edgar:dtend qb:codeList ?hierarchy1.
        ?dimMem1 skos:notation ?values1.
```

```
?obs edgar:segment ?values2.
        ?slicerMem0 skos:member ?level2.
        ?level2 skos:inScheme ?hierarchy2.
        edgar:segment qb:codeList ?hierarchy2.
        ?slicerMem0 skos:exactMatch ?values2. Filter(?
            slicerMem0 = edgar:
            segmentAHealthCareInsuranceCompany
        OR ?slicerMem0 = edgar:
            segmentAResidentialRealEstateDeveloperMember)
?obs egar:CostOfGoodsSold ?measureValue0.
} group by ?dimMem0 ?dimMem1
```

5 Experiment

In this section we demonstrate in a small experiment the applicability of our OLAP-to-SPARQL mapping to our scenario from the financial domain. The *SecCubeGrossProfitMargin* cube contains 17,448 disclosures that either disclose cost of goods sold or sales revenue net. The values of the measures fully depend on one of 625 different issuers, the date a disclosure started (27 members of dimension Dtstart) and ended (20 members of Dtend) to be valid, and additional information (21,227 members of Segment). The two measures have the unit USD and an aggregation function that returns the number of disclosures, or - if only one - the actual number. If fully materialised according to Definition 1, the cube contains $626 \cdot 28 \cdot 21 \cdot 21,228 = 7,813,772,064$ facts. To compute all of its facts, $2^4 = 16$ SPARQL subqueries would be needed.

OLAP interface allow users to interactively combine OLAP operations into an expression of an OLAP query language. Results of the query shall be visualised using a pivot table, a compact format to display multidimensional data [5]. As far as we know, MDX is the most widely used OLAP query language, adopted by OLAP engines such as Microsoft SQL Server, the Open Java API for OLAP, XML for Analysis (XMLA), and Mondrian. Therefore, we show that an MDX query can be transformed into an OLAP subcube query according to Definition 2 and evaluate the subcube query using a SPARQL query. The result is a subset of all possible facts from a data cube. The pivot table determines what dimensions to display on its columns and rows.

In order to answer the OLAP question of our scenario, we created the following MDX query. Multidimensional elements are described there using URIs.[13] For an introduction to MDX, see its website.[14] A more detailed description of how to transform an MDX query into an OLAP query due to space constraints we leave for future work when we evaluate our OLAP-to-SPARQL mapping more thoroughly.

[13] Note URIs need to be translated to an MDX-compliant format that does not use reserved MDX-specific characters.

[14] `http://msdn.microsoft.com/en-us/library/aa216770%28v=sql.80%29.aspx`

```
SELECT
{edgar:cik1417907idConcept, edgar:cik106535idConcept} ON
    COLUMNS,
CrossJoin(edgar:dtstartRootLevel.Members, edgar:
    dtendRootLevel.Members) ON ROWS
FROM [edgar:SecCubeGrossProfitMargin]
WHERE {edgar:CostOfGoodsSold}
```

A nested set of OLAP operations to compose our OLAP query is as follows:

```
Slice(Projection(
        edgar:SecCubeGrossProfitMargin,
        edgar:CostOfGoodsSold),
edgar:segment)
```

This query can then be represented as a subcube query with dimensions Issuer, Dtstart, Dtend, Segment: $(?, ?, ?, *, CostOfGoodsSold)$. The resulting SPARQL query is as follows:

```
select ?dimMem0 ?dimMem1 ?dimMem2 count(xsd:decimal(?
    measureValue0)) sum(xsd:decimal(?measureValue0))
where {
?obs qb:dataSet ?ds.
?ds qb:structure edgar:SecCubeGrossProfitMargin.
?obs edgar:issuer ?values0.
        ?dimMem0 skos:member ?level0.
        ?level0 skos:inScheme ?hierarchy0.
        edgar:issuer qb:codeList ?hierarchy0.
        ?dimMem0 skos:exactMatch ?values0.
?obs edgar:dtstart ?values1.
        ?dimMem1 skos:member ?level1.
        ?level1 skos:inScheme ?hierarchy1.
        edgar:dtstart qb:codeList ?hierarchy1.
        ?dimMem1 skos:notation ?values1.
?obs edgar:dtend ?values2.
        ?dimMem2 skos:member ?level2.
        ?level2 skos:inScheme ?hierarchy2.
        edgar:dtend qb:codeList ?hierarchy2.
        ?dimMem2 skos:notation ?values2.
?obs edgar:CostOfGoodsSold ?measureValue0.
} group by ?dimMem0 ?dimMem1 ?dimMem2
```

The aggregation function used is a non-standard one, therefore, we had to compute the SUM and COUNT for the measure. We run the query after a reboot of the triple store. The query took 18sec and returned 58 facts to be filled into the requested pivot table. The number of $7,813,772,064$ potential facts in the cube does not have a strong influence on the query since the cube is very sparse, for instance, the triple store contains observations only for a fraction of segment members.

6 Discussion

Data from the data cube is queried on demand, and no materialization is done. We correctly aggregate data on one specific granularity, defined by the mentioned inquired and fixed dimensions. Dimensions that are not mentioned will be automatically handled as having an *ALL* value [8], representing all possible values of the dimension. The aggregation results in correct calculations, since we assume only one hierarchy-level per dimension in this work. Only if observations would be defined on different granularities, e.g., gender *male*, *female*, and *total*, aggregating over them would result in incorrect numbers.

Filling the pivot table with measure values from the SPARQL result requires matching of the member values for each fact for the following reasons: first, if the data cube is sparse, i.e., not for every possible combination of members a value is given, then, for non-occurring combinations the SPARQL query does not return a value; second, all member combinations of inquired dimensions are calculated, even though only specific combinations might be selected, as in the case of the two issuers in the OLAP query of our scenario. Indexing of either the pivot table or the SPARQL result table may allow faster population of the pivot table.

In summary, though our OLAP algebra to SPARQL mapping may not result in the most efficient SPARQL query and require additional efforts for populating the pivot table, it correctly computes all required facts from the data cube without the need for explicitly introducing the non-relational *ALL* member or using sub-queries [8].

7 Related Work

Kobilarov and Dickinson [12] have combined browsing, faceted-search, and query-building capabilities for more powerful Linked Data exploration, similar to OLAP, but not focusing on statistical data. Though years have passed since then, current literature on Linked Data interaction paradigms does not seem to expand on analysing large amounts of statistics.

OLAP query processing in general has long been a topic of research [27]. OLAP operations have been defined on a logical level [1, 20, 26] or on a conceptual level [3, 22, 25]. Execution of OLAP operations mainly is concerned with the computation of the data cube and with storing parts of the results of that computation to efficiently return the results, to require few disk or memory space, and to remain easy to update if data sources change [14]. Approaches mainly depend on the type of data structure on which to perform the computations and in which to store the results. Data structures can roughly be grouped into ROLAP, using relational tables and star or snowflake schemas, and MOLAP, using multidimensional arrays for direct storing and querying of data cubes.

Specific approaches regarding OLAP on Linked Data seem to have concentrated so far on multidimensional modelling from ontologies [6, 15–17]. For instance, Nebot et al. [16] recognise the potential of OLAP to analyse RDF data,

but do not provide a dedicated query engine and require a multidimensional database that needs to be updated if RDF data changes. Entity-centric object databases [20] show some resemblance to OLAP querying, however, have so far not been applied to Linked Data.

In this work we use the graph-based RDF data model for querying and storing of multidimensional data reusing QB. Here, both schema information and actual data is accessed using Linked Data principles and managed using an RDF store. Our approach focuses on OLAP queries that can be composed by common OLAP operations and can be represented as a subcube query. Our mapping allows translating OLAP queries into one SPARQL query to be run on the RDF without storage of intermediate results. In our small experiment the produced SPARQL query showed sufficiently fast, but queries are expected to become insufficient for larger datasets. Since no materialization is done, only few extra space is required for a hashmap to fill the pivot table with the SPARQL result set, and updates to the RDF are propagated directly to OLAP clients. Although there may be more efficient querying approaches such as special indexing and caching, to the best of our knowledge, this is the first work on computing and querying of data cubes represented in RDF.

8 Conclusions and Future Work

We have presented an approach to interact with statistical Linked Data using common Online Analytical Processing operations of "overview first, zoom and filter, then details-on-demand". For that, we define common OLAP operations on single data cubes in RDF reusing the RDF Data Cube vocabulary, map nested sets of OLAP operations to OLAP subcube queries, and evaluate those OLAP queries using SPARQL. Both metadata and OLAP queries are issued directly on a triple store; therefore, if the RDF is modified or updated, changes are propagated directly to OLAP clients. Though, our OLAP-to-SPARQL mapping may not result in the most efficient SPARQL query and require additional effort in populating resulting pivot tables, we correctly calculate requested facts of a data cube without the need for explicitly introducing the non-relational *ALL* member or using subqueries.

Future work may be conducted in three areas: 1) extending our current approach with OLAP hierarchies and Drill-Across queries; 2) implementing an OLAP engine to more thoroughly evaluate our current approach and to investigate more efficient OLAP query execution plans; 3) investigating possible OLAP clients that map OLAP operations to intuitive user interactions.

Acknowledgements

This work was supported by the German Ministry of Education and Research (BMBF) within the SMART project (Ref. 02WM0800) and the European Community's Seventh Framework Programme FP7/2007-2013 (PlanetData, Grant 257641).

References

1. Agrawal, R., Gupta, A., Sarawagi, S.: Modeling Multidimensional Databases. In: Proc. of the Thirteenth International Conference on Data Engineering (1997)
2. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. ACM SIGMOD Record **26** (1997) 65–74
3. Chen, L., Ramakrishnan, R., Barford, P., Chen, B., Yegneswaran, V.: Composite Subset Measures. In: Proc. of the 32nd International Conference on Very Large Databases (2006)
4. Codd, E. F., Codd, S. B., Salley, C. T.: Providing OLAP to User-Analysts: An IT Mandate. (1993)
5. Cunningham, C., Galindo-Legaria, C. A., Graefe, G.: PIVOT and UNPIVOT: optimization and execution strategies in an RDBMS. In: Proc. of the Thirtieth International Conference on Very Large Databases (2004)
6. Diamantini, C., Potena, D.: Semantic enrichment of strategic datacubes. In: Proc. of the ACM 11th international workshop on Data warehousing and OLAP (2008)
7. Gómez, L. I., Gómez, S. A., Vaisman, A. A.: A Generic Data Model and Query Language for Spatiotemporal OLAP Cube Analysis Categories and Subject Descriptors. In: Proc. of EDBT 2012
8. Gray, J., Bosworth, A., Lyaman, A., Pirahesh, H.: Data cube: a relational aggregation operator generalizing GROUP-BY, CROSS-TAB, and SUB-TOTALS. In: Proc. of the Twelfth International Conference on Data Engineering (1995) 152–159
9. Harinarayan, V., Rajaraman, A.: Implementing data cubes efficiently. ACM SIGMOD Record (1996)
10. Harth, A.: VisiNav: A system for visual search and navigation on web data. Journal of Web Semantics **8**(4) (2010) 348–354
11. Kämpgen, B., Harth, A.: Transforming Statistical Linked Data for Use in OLAP Systems. In: Proc. of I-Semantics 2011
12. Kobilarov, G., Dickinson, I.: Humboldt: Exploring Linked Data. In: Proc. of Linked Data on the Web Workshop (LDOW 2008) at WWW 2008
13. Li, X., Han, J., Gonzalez, H.: High-dimensional OLAP: a minimal cubing approach. In: Proc. of the Thirtieth International Conference on Very Large Databases (2004)
14. Morfonios, K., Konakas, S., Ioannidis, Y., Kotsis, N.: ROLAP implementations of the data cube. ACM Computing Surveys **39** (2007) 12–es
15. Nebot, V., Berlanga, R.: Building data warehouses with semantic web data. Decision Support Systems **52** (2012) 853–868
16. Nebot, V., Berlanga, R., Pérez, J. M., Aramburu, M. J., Vej, S. L.: Multidimensional Integrated Ontologies : A Framework for Designing Semantic Data Warehouses. Journal on Data Semantics (2009) 1–36
17. Niinimäki, M., Niemi, T.: An ETL Process for OLAP Using RDF/OWL Ontologies. Journal on Data Semantics XIII **5530** (2009) 97–119
18. Pardillo, J., Mazón, J.-N.: Using Ontologies for the Design of Data Warehouses. International Journal of Database Management Systems **3** (2011) 73–87
19. Pardillo, J., Mazón, J.-N., Trujillo, J.: Bridging the semantic gap in OLAP models: platform-independent queries. In: Proc. of the ACM 11th international workshop on Data warehousing and OLAP (2008)
20. Pedersen, T. B., Gu, J., Shoshani, A., Jensen, C. S.: Object-extended OLAP querying. Data Knowl. Eng. **68** (2009) 453–480
21. Romero, O., Abelló, A.: Automating multidimensional design from ontologies. In: Proc. of the ACM tenth international workshop on Data warehousing and OLAP (2007)

22. Romero, O., Abelló, A.: On the Need of a Reference Algebra for OLAP. In: Proc. of DaWaK 2007
23. Romero, O., Marcel, P., Abelló, A., Peralta, V., Bellatreche, L.: Describing analytical sessions using a multidimensional algebra. In: Proc. of the 13th international conference on Data warehousing and knowledge discovery (2011)
24. Shneiderman, B.: The Eyes Have It : A Task by Data Type Taxonomy for Information Visualizations. Information Visualization (1996) 336–343
25. Trujillo, J.: Bridging the Semantic Gap in OLAP Models : Platform-independent Queries Categories and Subject Descriptors. Computing Systems (2008) 89–96
26. Vassiliadis, P.: Modeling Multidimensional Databases, Cubes and Cube Operations. In: Proc. of the 10th International Conference on Scientific and Statistical Database Management (1998)
27. Vassiliadis, P., Sellis, T.: A survey of logical models for OLAP databases. ACM Sigmod Record **28** (1999) 64–69