# Towards Exploratory OLAP Over Linked Open Data – A Case Study

Dilshod Ibragimov[1,2](✉), Katja Hose[2], Torben Bach Pedersen[2], and Esteban Zimányi[1]

[1] Université Libre de Bruxelles, Brussels, Belgium
{dibragim,ezimanyi}@ulb.ac.be
[2] Aalborg University, Aalborg, Denmark
{diib,khose,tbp}@cs.aau.dk

**Abstract.** Business Intelligence (BI) tools provide fundamental support for analyzing large volumes of information. Data Warehouses (DW) and Online Analytical Processing (OLAP) tools are used to store and analyze data. Nowadays more and more information is available on the Web in the form of Resource Description Framework (RDF), and BI tools have a huge potential of achieving better results by integrating real-time data from web sources into the analysis process. In this paper, we describe a framework for so-called exploratory OLAP over RDF sources. We propose a system that uses a multidimensional schema of the OLAP cube expressed in RDF vocabularies. Based on this information the system is able to query data sources, extract and aggregate data, and build a cube. We also propose a computer-aided process for discovering previously unknown data sources and building a multidimensional schema of the cube. We present a use case to demonstrate the applicability of the approach.

**Keywords:** Exploratory OLAP · LOD · QB4OLAP

## 1 Introduction

In the business domain, there is a constant need to analyze big volumes of information for intelligent decision making. Business intelligence tools provide fundamental support in this direction. In general, companies use data warehouses to store big volumes of information and OLAP tools to analyze it. Data in such systems are generated by feeding operational data of enterprises into data warehouses. Then, OLAP queries are run over data to generate business reports. Multidimensional Expressions (MDX) query language is the de-facto standard for OLAP querying.

Traditionally, such analyses are performed in a "closed-world" scenario, based only on internal data. With the advent of the Web, more and more data became available online. These data may be related to, for example, the market, competitors, customer opinions (e.g., tweets, forum posts), etc. Initially, these data were not suitable for machine processing. Later, a framework that extends the

M. Castellanos et al. (Eds.): BIRTE 2013 and 2014, LNBIP 206, pp. 114–132, 2015.
DOI: 10.1007/978-3-662-46839-5_8

principles of the Web from documents to data converting the Web of Documents into the Web of Data was proposed. According to the standards, to facilitate a discovery of the published data, these data should comply with the Linked Data principles [32]. RDF was chosen as a standard model for data interchange on the Web [34]. With these principles in action, the whole Internet may be considered as one huge distributed dataspace.

With data being publicly available, businesses see the benefits of incorporating additional, real-time data into the context of information received from data warehouses or analyzing these data independently. Companies may explore new data opportunities and include new data sources into business analyses. A new type of OLAP that performs discovery, acquisition, integration, and analytical querying of new external data is necessary. This type of OLAP was termed *Exploratory* OLAP [2].

In the past years the scientific community has been working on bringing these new BI concepts to end-users. The main focus of research was providing an easy and flexible access to different data sources (internal and external) for non-skilled users so that the users can express their analytical needs and the system is able to produce data cubes on-demand. Optimally, the internal complexity of such systems should be transparent to end-users.

In [1] a vision to new generation BI and a framework to support self-service BI was proposed. The process, according to this framework, is divided into several steps and consists of query formulation, source discovery and selection, data acquisition, data integration, and cube presentation phases. Based on this framework, we propose our approach to performing exploratory OLAP over Linked Open Data (LOD). For the sake of simplicity, our scenario considers only data available in RDF format and accessible over SPARQL endpoints [35].

The novel contribution of this paper are:

- We define a multidimensional schema of an OLAP cube exclusively in RDF. This multidimensional schema allows to define remote data sources for querying during the OLAP analysis phase.
- We propose a computer-aided approach to deriving the schema of the OLAP cube from previously unknown sources.

The remainder of the paper is structured as follows: in Sect. 2, we introduce a case study for exploratory OLAP scenario where the multidimensional schema and sources of data are already known. We show how we can retrieve data and build an OLAP cube. In Sect. 3, we propose ideas for sources discovery and schema generation for such cases. In Sect. 4, we present a conceptual framework for achieving exploratory OLAP over LOD. In Sect. 5, we discuss the related work. Finally, in Sect. 6, we conclude this paper and identify future work.

## 2 A Movie Case Study

This scenario is based on the dataset originating from the Linked Movie Database[1] (LinkedMDB) website, which provides information about movies.

[1] http://data.linkedmdb.org.

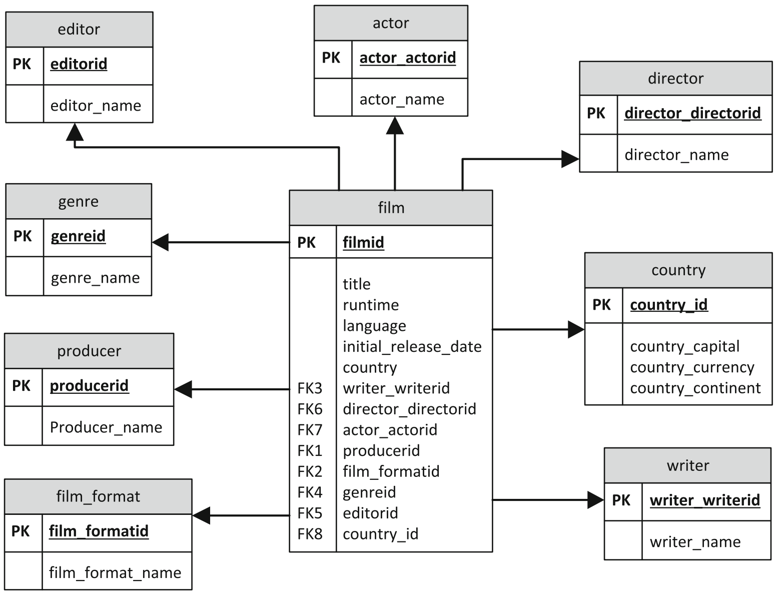


**Fig. 1.** Partial LinkedMDB logical schema

LinkedMDB publishes Linked Open Data for movies, including a large number of interlinks to several datasets on the LOD cloud and references to related webpages. Data can be queried using a SPARQL endpoint[2].

A typical movie record contains information about the movie, the actors who played in the movie, the director of the movie, the genre, the initial release date, the runtime, the country where it was produced, etc. An example record for the movie "The Order"[3] stored in LinkedMDB is as follows (all the prefixes used in the paper are listed in the appendix):

```
<http://data.linkedmdb.org/resource/film/1005> rdf:type movie:film ;
  movie:actor <http://data.linkedmdb.org/resource/actor/32063> ;
  movie:actor <http://data.linkedmdb.org/resource/actor/42288> ;
  foaf:based_near <http://sws.geonames.org/2921044/> ;
  movie:country <http://data.linkedmdb.org/resource/country/DE> ;
  dc:date ''2003,2003-09-05'' ;
  movie:director <http://data.linkedmdb.org/resource/director/9091> ;
  movie:film_cut <http://data.linkedmdb.org/resource/film_cut/15031> ;
  movie:filmid ''1005''^^xsd:int ;
  movie:genre <http://data.linkedmdb.org/resource/film_genre/28> ;
  movie:initial_release_date ''2003,2003-09-05'' ;
  rdfs:label ''The Order'' ;
  movie:language <http://www.lingvoj.org/lingvo/en> ;
  foaf:page <http://www.imdb.com/title/tt0304711> ;
  movie:runtime ''102'' ;
  dc:title ''The Order'' .
```

[2] http://data.linkedmdb.org/sparql.
[3] http://data.linkedmdb.org/resource/film/1005.

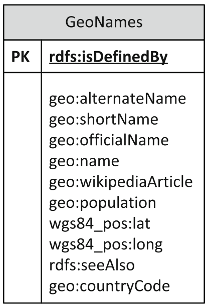
A partial logical schema of the LinkedMDB is given in Fig. 1. LinkedMDB also contains links to other datasets using the property `owl:sameAs`. For example, a country information is interlinked to GeoNames[4]. Based on the analysis of GeoNames, the partial logical schema of GeoNames is illustrated in Fig. 2.

Suppose a user wants to analyze data about movies. Examples of typical queries could be:

– Average runtime for movies by movie director and country
– Number of movies by continent and year

**N.B.:** She may want to do it in the context of information retrievable from GeoNames.

For this purpose, the user may want to construct a virtual data cube. Data will be retrieved from two sources: LinkedMDB and GeoNames. The data cube is considered virtual because data are not materialized in the local system. This data cube accepts user queries, queries the data sources, retrieves the information, processes it, and answers user queries. The multidimensional schema of such a data cube is given in Fig. 3. The schema describes the dimensions: `Country (Population, Country Name)`, `Release Date (Year, Quarter, Month)`, `Director`, `Actor`, `Script Writer` and the measure: `Runtime`.



| GeoNames | |
|---|---|
| PK | rdfs:isDefinedBy |
| | geo:alternateName<br>geo:shortName<br>geo:officialName<br>geo:name<br>geo:wikipediaArticle<br>geo:population<br>wgs84_pos:lat<br>wgs84_pos:long<br>rdfs:seeAlso<br>geo:countryCode |

**Fig. 2.** Partial GeoNames logical schema

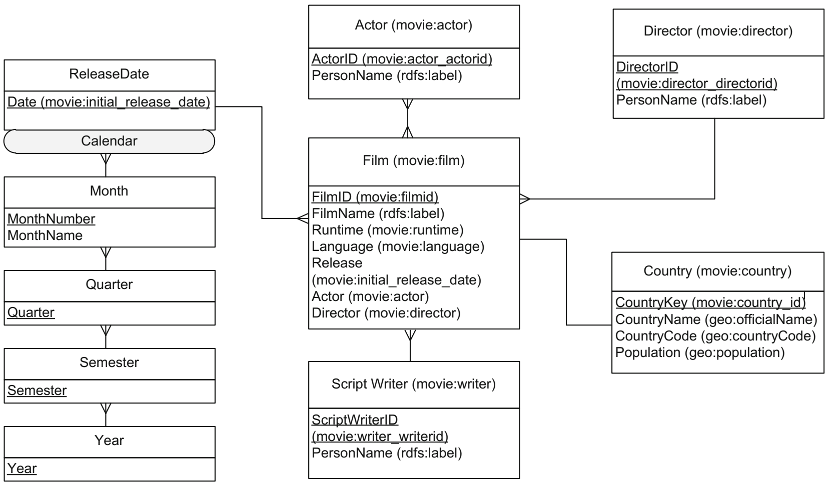

ReleaseDate
Date (movie:initial_release_date)
Calendar
Month
MonthNumber
MonthName
Quarter
Quarter
Semester
Semester
Year
Year
Actor (movie:actor)
ActorID (movie:actor_actorid)
PersonName (rdfs:label)
Film (movie:film)
FilmID (movie:filmid)
FilmName (rdfs:label)
Runtime (movie:runtime)
Language (movie:language)
Release (movie:initial_release_date)
Actor (movie:actor)
Director (movie:director)
Script Writer (movie:writer)
ScriptWriterID (movie:writer_writerid)
PersonName (rdfs:label)
Director (movie:director)
DirectorID (movie:director_directorid)
PersonName (rdfs:label)
Country (movie:country)
CountryKey (movie:country_id)
CountryName (geo:officialName)
CountryCode (geo:countryCode)
Population (geo:population)

**Fig. 3.** Conceptual schema of the data cube

4 http://www.geonames.org/.

Knowing the structure of the cube, a user wants to find the average runtime for movies by director and country. She issues an MDX query as shown in Listing 1.1:

```
WITH MEMBER Measures.AvgRuntime AS Avg(Film.Director.CurrentMember, Measures.Runtime)
SELECT NON EMPTY {Film.Director.Members} ON COLUMNS,
        NON EMPTY {Film.Country.Members} ON ROWS
FROM [MoviesDataWarehouse] WHERE (Measures.AvgRuntime);
```

Listing 1.1: MDX Query on the Data Cube

Data on the Web are mostly stored and retrieved as RDF and not as relational data. Therefore, we propose to use a fully RDF-based approach for exploratory OLAP over LOD sources and to analyze data without converting them to relational data and storing them in a local data warehouse. Additionally, loading and storing highly volatile, real-time data in a local system may not be practical.

In our case study we use RDF vocabularies such as QB4OLAP [5] and VoID [31] to describe the multidimensional schema. QB4OLAP is an RDF vocabulary that allows the publication of multidimensional data. QB4OLAP can represent dimension levels, level members, rollup relations between levels and level members, etc. QB4OLAP can also associate aggregate functions to measures. VoID is an RDF Schema vocabulary for expressing metadata about RDF datasets. The vocabulary may specify how RDF data can be accessed using various protocols. For example, the SPARQL endpoint location can be specified by the property `void:sparqlEndpoint`. Based on the information from the multidimensional schema, the system will be able to identify the sources and query them. An excerpt of the multidimensional schema for our running example, expressed in the QB4OLAP and VoID vocabularies, is given in Listing 1.2.

```
## Data structure definition and dimensions
exqb:FilmCube a qb:DataStructureDefinition ;
    void:sparqlEndpoint
        <http://data.linkedmdb.org/sparql> ;
## Dimensions
qb:component [qb:dimension exqb:Actor];
qb:component [qb:dimension exqb:ReleaseDate];
qb:component [qb:dimension exqb:Director];
qb:component [qb:dimension exqb:Country];
## Definition of measures
qb:component [qb:measure exqb:Runtime];
## Attributes
qb:component [qb:attribute exqb:FilmName] .
```

```
## Dimension Properties and Hierarchies
exqb:year a qb4o:LevelProperty ;
    skos:closeMatch db:Year ;
    rdfs:comment "Film release year"@en ;
    qb4o:inDimension exqb:ReleaseDate .
exqb:quarter a qb4o:LevelProperty ;
    rdfs:comment "Film release quarter"@en ;
    qb4o:inDimension exqb:ReleaseDate .
exqb:ReleaseDate a qb:DimensionProperty .
exqb:Actor a qb:DimensionProperty ;
    skos:mappingRelation movie:actor ;
    rdfs:seeAlso owl:sameAs ;
    qb4o:hasAttribute exqb:PersonName .
```

Listing 1.2: Multidimensional Schema Expressed in QB4OLAP

To answer the MDX query, the system needs to send SPARQL queries to remote data endpoints for data retrieval. To do this, it first finds appropriate information for the measures and the dimensions specified in the MDX query from the multidimensional schema. The system finds the sources of data for dimensions /measures (`void:sparqlEndpoint`), all the attributes (`qb4o:hasAttribute`), the mapping information to map these attributes to the source equivalents (`skos:mappingRelation`), etc. For instance, for the MDX query given in

Listing 1.1 the system needs to find the information about the `Runtime` measure and the `Director` and the `Country` dimensions. Then, the system sends SPARQL queries to the LinkedMBD and the GeoNames SPARQL endpoints. The query that is sent to LinkedMDB to retrieve the information regarding dimensions, attributes, and measures is given in Listing 1.3.

```
### Retrieving attributes, dimensions, and measures
CONSTRUCT {
    ?movieUrl exqb:Runtime ?runtime . ?movieUrl exqb:FilmName ?movieName .
    ?movieUrl exqb:Country ?country . ?country owl:sameAs ?owlCountry .
    ?movieUrl exqb:Director ?directorID . ?directorID exqb:PersonName ?directorName .
} WHERE {
    ?movieUrl rdf:type movie:film . ?movieUrl movie:country ?country .
    ?country owl:sameAs ?owlCountry . ?movieUrl rdfs:label ?movieName .
    ?movieUrl movie:runtime ?runtime . ?movieUrl movie:director ?directorID .
    ?directorID rdfs:label ?directorName .
}
```

Listing 1.3: SPARQL Query to LinkedMDB

This query uses the `CONSTRUCT` clause to automatically create triples. These triples specify the dimension attributes and therefore can easily be copied to the final QB4OLAP structure. An excerpt from the result returned to the system for the query is as follows:

```
<rdf:Description rdf:about="http://data.linkedmdb.org/resource/film/930">
  <exqb:FilmName>Godfather</exqb:FilmName>
  <exqb:Director rdf:resource="http://data.linkedmdb.org/resource/director/448"/>
  <exqb:Country rdf:resource="http://data.linkedmdb.org/resource/country/IN"/>
  <exqb:Runtime>158</exqb:Runtime>
</rdf:Description>
<rdf:Description rdf:about="http://data.linkedmdb.org/resource/film/2939">
  <exqb:Director rdf:resource="http://data.linkedmdb.org/resource/director/10494"/>
  <exqb:Runtime>120</exqb:Runtime>
  <exqb:FilmName>Raincoat</exqb:FilmName>
  <exqb:Country rdf:resource="http://data.linkedmdb.org/resource/country/IN"/>
</rdf:Description>
<rdf:Description rdf:about="http://data.linkedmdb.org/resource/director/448">
  <exqb:PersonName>K. S. Ravikumar (Director)</exqb:PersonName>
</rdf:Description>
<rdf:Description rdf:about="http://data.linkedmdb.org/resource/director/10494">
  <exqb:PersonName>Rituparno Ghosh (Director)</exqb:PersonName>
</rdf:Description>
<rdf:Description rdf:about="http://data.linkedmdb.org/resource/country/IN">
  <owl:sameAs rdf:resource="http://sws.geonames.org/1269750/"/>
</rdf:Description>
```

Then, the data from GeoNames may be downloaded. In our running example the system uses the URI received from the "`owlCountry`" property and use it in the `VALUES` statement of the SPARQL query. We use a `VALUES` statement to group several arguments together in one query. Our goal is to send as few queries as possible. Since GeoNames does not have an associated SPARQL endpoint, the query is sent to the mirrored endpoint (http://lod2.openlinksw.com/sparql):

```
CONSTRUCT {
    ?s geo:countryCode ?o1 . ?s geo:name ?o2 . ?s geo:population ?o3 .
} WHERE {
    ?s geo:countryCode ?o1 . ?s geo:name ?o2 . ?s geo:population ?o3 .
    VALUES (?s){ (<http://sws.geonames.org/1149361/>) ... (<http://sws.geonames.org/1269750/>) }
}
```

Listing 1.4: SPARQL Query to GeoNames

This query returns information about the country's population, name, and code. A sample answer may look as follows:

```
<http://sws.geonames.org/1149361/>  geo:countryCode "AF" ;
    geo:name  "Islamic Republic of Afghanistan" ;
    geo:population  "29121286" .
```

The data obtained from the two sources are merged into a QB4OLAP structure: all received facts may be stored as `qb:Observation` instances (in OLAP terminology this corresponds to facts indexed by dimensions), all dimension instances are stored as triples. The aggregated values for measures are computed based on the `qb4o:AggregateFunction` function type. A sample QB4OLAP structure is given in Listing 1.5:

```
<http://data.linkedmdb.org/resource/film/810> a qb:Observation;
  qb:dataSet exqb:MoviesDataWarehouse ;
  exqb:Director < http://data.linkedmdb.org/resource/director/8629> ;
  exqb:Runtime 188;
  exqb:Country < http://data.linkedmdb.org/resource/country/IN> .
http://data.linkedmdb.org/resource/film/930> a qb:Observation;
  qb:dataSet exqb:MoviesDataWarehouse ;
  exqb:Director < http://data.linkedmdb.org/resource/director/448> ;
  exqb:Runtime 158;
  exqb:Country < http://data.linkedmdb.org/resource/country/IN> .
<http://data.linkedmdb.org/resource/country/IN>
        exqb:CountryName "India" ;
        exqb:CountryCode "IN" ;
        exqb:Population "1173108018" .
<http://data.linkedmdb.org/resource/director/448>
        exqb:PersonName "K. S. Ravikumar (Director)" .
```

Listing 1.5: Observations in QB4OLAP

In case the number of returned triples is large and cannot be handled by a SPARQL endpoint or transferred over the Internet, the system can send aggregate subqueries to the sources. The aggregation can be performed on the graph patterns used for joining several federated SPARQL subqueries. This will help to reduce the number of records for which the values from the endpoints will be transferred. For example, the following subqueries return aggregate values (left) and additional information (right) on the runtime of the movies by director and country. The results can be connected via the values of the `?owlCountry`.

```
SELECT AVG(?runtime) ?dirName ?owlCountry
WHERE {
  ?movUrl exqb:Runtime ?runtime .
  ?movUrl exqb:Country ?country .
  ?cntr owl:sameAs ?owlCountry .
  ?movUrl exqb:Director ?dirID .
  ?dirID exqb:PersonName ?dirName .
} GROUP BY ?dirName ?owlCountry
```

```
SELECT ?owlCountry ?code ?c_name ?pop
WHERE {
  ?owlCountry geo:countryCode ?code .
  ?owlCountry geo:name ?c_name .
  ?owlCountry geo:population ?pop
}
```

Listing 1.6. Aggregate and Informational Subqueries

The intermediate results of the execution of the subqueries may be stored in an in-memory table. Then, the results of the execution of the subqueries will be merged into the QB4OLAP structure.

Based on these data, the computed aggregated values are returned back to a user of the system. A sample answer to the previous MDX query may look as shown in Table 1:

**Table 1.** Aggregated Values

| | Great Britain | India | United states | Venezuela | Pakistan | Russia | Netherlands |
|---|---|---|---|---|---|---|---|
| Sally Potter (Director) | 86 | | | | | | 96 |
| Robert Aldrich (Director) | | | 88 | | | | |
| Román Chalbaud (Director) | | 93 | | | | | |
| Roland Joffé (Director) | | | | 87 | | | |
| Gerald Thomas (Director) | 78 | | 83 | | | | |

## 3 Source Discovery and Schema Building for Exploratory OLAP

In the case study introduced in Sect. 2 we assume that the data sources and the multidimensional schema of the OLAP cube are known. However, in reality the discovery of essential data sources is not a trivial task. Despite the fact that the publication of Linked Data has gained momentum in recent years, there is still no single approach on how these data should be published to be easily discoverable. We identified three potentially interesting data source discovery approaches for further investigation. In all three approaches described below we show how we can derive a schema of the OLAP cube for the scenario discussed in Sect. 2.

### 3.1 Querying Knowledge Bases

The first approach is querying large knowledge bases such as DBpedia[5], Yago[6], or Freebase[7] to find relevant information. Data from such knowledge bases are usually freely accessible over SPARQL endpoints. Querying these endpoints for the term of interest may lead to the discovery of useful sources of data or the necessary information itself. Since the number of answers that come from these sources may be extremely large and not always relevant, there is a need for filtering the answers. Also, since the user entry may be ambiguous due to the ambiguity and complexity of natural languages, the end user needs to guide the process of source discovery by selecting most appropriate alternatives for further investigation.

To find some relevant information about the term "Film", we can send the following SPARQL query to the Freebase SPARQL endpoint:

[5] http://dbpedia.org/About.
[6] http://www.mpi-inf.mpg.de/yago-naga/yago/.
[7] http://www.freebase.com/.

**Table 2.** Freebase Query Partial Results

| ?s | ?l | ?count |
|---|---|---|
| http://rdf.freebase.com/ns/m.02nsjl9 | Film character | 2001832 |
| http://rdf.freebase.com/ns/film.film/character | Film character | 1384754 |
| http://rdf.freebase.com/ns/film.actor | Film actor | 874840 |
| http://rdf.basekb.com/ns/m.0jsg30 | Film performance | 673398 |
| http://rdf.freebase.com/ns/film.film | Film | 557505 |
| http://rdf.freebase.com/ns/m.0jsg4j | Film actor | 492249 |
| http://rdf.freebase.com/ns/film.film_crew_gig | Film crew gig | 456500 |
| http://rdf.basekb.com/ns/m.02nsjl9 | Film character | 410669 |
| http://rdf.basekb.com/ns/m.0jsg4j | Film actor | 215777 |
| http://rdf.freebase.com/ns/m.02_6zn1 | Film crewmember | 205938 |

**Table 3.** Freebase Movie Instances

| ?s | ?p | ?o |
|---|---|---|
| http://rdf.freebase.com/ns/m.0pj5t | rdfs:label | Falling Down |
| http://rdf.freebase.com/ns/m.0swhj | rdfs:label | A Charlie Brown Christmas |
| http://rdf.freebase.com/ns/m.0m2kd | rdfs:label | Stand by Me |
| http://rdf.freebase.com/ns/m.07cz2 | rdfs:label | The Matrix |
| http://rdf.freebase.com/ns/m.0c296 | rdfs:label | Amélie |
| http://rdf.freebase.com/ns/m.0prk8 | rdfs:label | Hamlet |
| http://rdf.freebase.com/ns/m.0j90s | rdfs:label | Guess Who's Coming to Dinner |
| http://rdf.freebase.com/ns/m.02yxx | rdfs:label | Fearless |
| http://rdf.freebase.com/ns/m.0p9rz | rdfs:label | Romeo and Juliet |
| http://rdf.freebase.com/ns/m.0symg | rdfs:label | Dead Man |

```
SELECT ?s ?l COUNT(?s) as ?count
    WHERE { ?someobj ?p ?s .  ?s rdfs:label ?l .
    FILTER(CONTAINS(?l,"Film") && (lang(?l) = 'en') && (!isLiteral(?someobj))) .
} ORDER BY DESC(?count) LIMIT 20
```

This query is optimized to allow sorting by relevance using the COUNT function so that the user sees the most relevant answers first. The partial result of the query is given in Table 2.

By examining the returned answer, the user may find some interesting triples and may want to explore these triples further. The system at this stage helps the user to do so. For example, several triples should be retrieved for further exploration. In our case, one of the triples has a subject equal to <http://rdf.freebase.com/ns/film.film>. The following query returns several instances related to the triple pattern of interest:

```
SELECT ?s ?p ?o
    WHERE {
    ?s ?p ?o . ?s ns:type.object.type ns:film.film . FILTER (lang(?o) = 'en').
} LIMIT 10
```

The result of the execution of the query is given in Table 3.

If the user decides that the selected samples satisfy the needs, the user is aided in building a multidimensional model of the OLAP cube. Our proposition for building a graph representation of the source is based on characteristic sets (CS) (Neumann and Moerkotte [19]), which contain the properties of RDF data triples for triple subjects. The system should also offer possible candidates for measures, dimensions, and dimensional attributes, identifying all triples related to the instances, their data types, etc. Then the user chooses the schema that most closely reflects the needs or directs the system for further search. In our example, the user may encounter properties of interest such as runtime, director, actors, and country by exploring the instance properties of the class `ns:film.film`:

```
ns:m.0c296  ns:film.film.country    ns:m.0345h;
ns:film.film.directed_by    ns:m.0k181;
ns:film.film.edited_by ns:m.07nw1y6;
ns:film.film.genre    ns:m.05p553;
ns:film.film.initial_release_date    "2001-04-25"^^xsd:datetime;
ns:film.film.runtime..film.film_cut.runtime    ns:122.0;
ns:film.film.starring..film.performance.actor    ns:m.01y9t4;
ns:film.film.starring..film.performance.actor    ns:m.0jtcpc;
```

The whole process of discovering the sources and building the multidimensional schema needs to be guided by a user.

## 3.2 Querying Data Management Platforms

The second approach for source discovery is querying so-called data management platforms. One such platform is the Datahub[8]–the platform based on the CKAN[9] registry system. CKAN is an open source registry system that allows storing, distributing, and searching of the contents for spreadsheets and datasets. Search and faceting features allow users to browse and find the data they need. CKAN provides an API that can be used for searching the data by applications. For instance, CKAN's Action API provides functions for searching for packages or resources matching a user query. Using the Action API, we can list all the datasets (packages) residing in the system (http://datahub.io/api/3/action/package_list), view the dataset descriptions (http://datahub.io/api/3/action/package_show?id=linkedmdb), or search for datasets matching the search query (http://datahub.io/api/3/action/package_search?q=Film). The answer is returned in JSON format.

Querying the Datahub for a "Film" string returns 99 results, where 5 results have SPARQL endpoints: Prelinger Archives (http://api.kasabi.com/dataset/prelinger-archives/apis/sparql), Linked Movie Database (http://data.linkedmdb.

[8] http://datahub.io.
[9] http://ckan.org/.

org/sparql), DBpedia-Live (http://live.dbpedia.org/sparql), Europeana Linked Open Data (http://europeana.ontotext.com/sparql), and DBpedia (http://dbpedia.org/sparql). By retrieving several instances of triple patterns and identifying corresponding properties (the same process as proposed for querying knowledge bases), we may define the multidimensional schema needed for the OLAP cube.

### 3.3 Querying Semantic Web Search Engines

The third approach for sources discovery is querying semantic web search engines. An example of such search engines is Sindice[10], which also provides a Search API (http://sindice.com/developers/searchapiv3) using a query language (http://sindice.com/developers/queryLanguage). The Search API provides programmatic access to search capabilities of the search engine and returns the result in one of three formats: JSON, RDF, or ATOM. This API supports a keyword search to facilitate the discovery of relevant documents that contain either a keyword or a URI. The query language supports filtering the search results by URL, domain, class, predicate, ontology, etc. and grouping the search results by datasets.

Querying Sindice for the "Film" string returns many results (582,883), mostly individual triples, but grouping the results by datasets allows identifying the datasets for further exploration. The following query to Sindice reveals the Linked Movie Database dataset (http://data.linkedmdb.org) for further exploration among others:http://api.sindice.com/v3/search?q=Film&format=json&fq=format%3ARDF&page=6&facet.field=domain.

After discovering a proper source of information, we should apply the process of building the multidimensional schema of the OLAP cube.

## 4 Conceptual Framework

The main functionality of an exploratory OLAP system is illustrated in Fig. 4. Here we assume that there may (optionally) exist some internal data depicted as a cube with dotted lines. These data may serve as a foundation for further exploration. A user may want to enrich/supplement these data by external data from the Web. Ideally, the system should be able to retrieve data stored in any format (HTML, XML, CSV, RDF, etc.). In Fig. 4 these data are depicted as small colored cubes which extend the internal cube. This requirement imposes additional complexity over the system, so the part of the system that is responsible for exploratory OLAP can be further subdivided into several subparts, each handling another data format. In this paper we concentrate on Linked Open Data and we describe our vision on how to achieve exploratory OLAP over Linked Open Data.

The envisioned architecture for the exploratory OLAP over Linked Open Data system is sketched in Fig. 6. The system consists of four main modules. The Global Conceptual Schema module contains information about the schema

[10] http://sindice.com/.

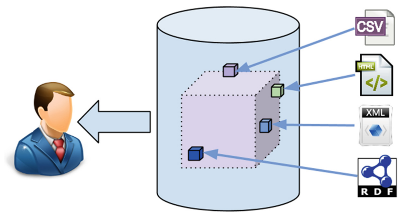


**Fig. 4.** Functional View

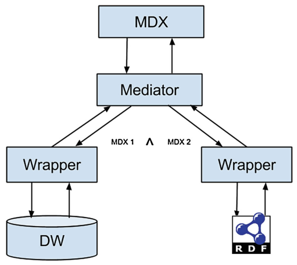


**Fig. 5.** Data Integration

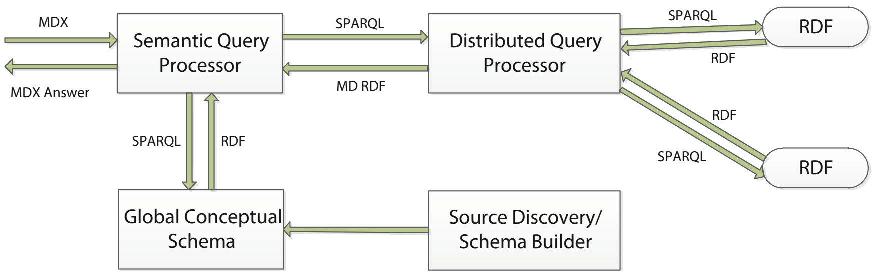


**Fig. 6.** System Architecture

of the specified data cube. In particular, it contains information about the measures, the dimensions and hierarchies in the dimensions, the potential aggregation functions over the measures, and pointers to data sources where the data are located. To represent this information, we propose to use the combination of QB4OLAP and VoID vocabularies. QB4OLAP allows defining dimensions, measures, and aggregations. The access and linkset metadata sections of the VoID vocabulary allow to describe data sources. An example of the multidimensional schema expressed in QB4OLAP that is part of the Global Conceptual Schema module can be found in Listing 1.2.

This combination of vocabularies is robust w.r.t. the schema complexity, the number of data sources, and the data volume. The schema complexity is handled by the QB4OLAP vocabulary as demonstrated in [30]. Recent changes to the QB4OLAP vocabulary [4] aid in defining complex multidimensional schemas with different hierarchies of levels in dimensions (balanced, recursive, ragged, many-to-many), different cardinalities between level members (one-to-many, many-to-many, etc.), levels belonging to different hierarchies, etc. A number of data sources can be referenced in a multidimensional schema of a data cube with the help of the VoID vocabulary. Regarding the data volume, recent experiments show that triple stores per se are not worse for analytical queries than RDBMS [15], so we expect our approach to be sufficiently scalable.

The Semantic Query Processor is a module of the system that accepts an MDX query as input and produces a multidimensional SPARQL query using the QB4OLAP vocabulary for further processing. For this purpose, it queries the Global Conceptual Schema to find appropriate information – the measures and the dimensions specified in the MDX query. After having received the requested information, the Semantic Query Processor will formulate SPARQL queries to all data endpoints and send these queries to the Distributed Query Processing module for data retrieval. Examples of such SPARQL queries can be found in Listings 1.3 and 1.4. The Distributed Query Processor in turn queries all data endpoints, collects and merges all data, and returns the result back to the Semantic Query Processor (Listing 1.5). The returned answer is then either displayed to the user or passed to the calling module for the integration with data from the internal data warehouse.

The integration of external dimensional data with an internal data warehouse has been studied before. For instance, Pedersen et al. [21] present an approach to the logical federation of OLAP and XML data sources. Following the same pattern, we envision that the system will have the mediator/wrappers to split and translate initial MDX query to other query languages. This is a common approach in distributed database systems [27]. The results received from the wrappers will then be merged by the mediator and shown to the user. The data integration architecture is depicted in Fig. 5.

The Source Discovery/Schema Builder module is responsible for deriving a schema of the OLAP cube based on the user requirements. This module interacts with the user during the schema construction phase. The user specifies the domain/key concept of interest; the module searches for appropriate data sources and proposes the most relevant of them to the user. The module uses the approaches described in Sect. 3 to find interesting data sources. We propose to use all three approaches because none of these approaches alone guarantees full reliability. After identifying data sources, the system proposes a list of potential facts, dimensions, and measures, constructs possible multidimensional schemas, and presents them to the user for confirmation. This multidimensional schema is then used in the Global Conceptual Schema module.

## 5 Related Work

In the following, we review previous research in semantic web warehousing, source discovery, and distributed SPARQL query processing.

### 5.1 Semantic Web Data Warehousing

Related work for semantic web data warehousing can be divided into two categories. In the first category of approaches, the data is loaded into a local data warehouse that is built over a relational database management system. The schema of the data warehouse is generally determined by an administrator of the system and the data from the Linked Data sources are loaded into the defined tables. Then, the OLAP queries are run against the data stored in a star or snowflake schema. In the second category of approaches the OLAP operations are executed directly over RDF stores via SPARQL.

Determining schema information for a discovered data source helps in building a multidimensional model of a data cube. In an RDF dataset, the subjects that share the same properties can be grouped together. The result is a list of property sets with associated subjects. These property sets are called Characteristic Sets. Neumann et al. [19] used the knowledge about these sets for the estimation of the result cardinality for join operations in triple stores. In comparison, we instead employ characteristic sets as a basis for building a multidimensional data cube schema.

Romero et al. [25] defined a semi-automatic general-purpose method of building a multidimensional schema for a data warehouse from a domain ontology. The method aims to propose meaningful multidimensional schema candidates. The method defines main steps that lead to identifying facts, dimensions, and dimension hierarchies. The system is semi-automatic in the sense that it expects a user confirmation for suggested concepts proposed as potential facts. Once the user selects a concept as a fact concept, it will give rise to a multidimensional schema. The disadvantage of this approach is the requirement to have a corresponding domain ontology. This may not be the case for all data sources.

Similarly, a semi-automatic method for the identification and extraction of data expressed in OWL is defined in [18]. OWL/DL is used to transfer valid data into fact tables and to build dimensions. According to the proposed method, an analyst defines a multidimensional star schema based on the known ontology of the source of data. Then, the data from the sources are loaded into the data warehouse. Overall, this method does not allow populating a multidimensional schema with semantic web data from the newly discovered sources with previously unknown structures.

A framework to streamline the ETL process from Linked Open Data to a multidimensional data model is proposed in [13]. In contrast to [18], this work does not require previous knowledge and an ontology to collect the data. The data that are retrieved from Linked Open Data sources are first stored in an intermediate storage, where these data are partitioned based on the type. Then, the analyst investigates the tables and chooses measures and dimensions for the multidimensional data model. Afterwards, the system generates the schema for the fact table, selects dimensions, and dumps data into relational tables for performing OLAP analysis. The disadvantage of this method is the requirement to have a high-level analyst for intermediate result investigation and multidimensional schema construction.

The approach proposed in [14] uses an ETL pipeline to convert statistical Linked Open Data into a format suitable for loading into an open-source OLAP system. The data are presented using the RDF Data Cube (QB) vocabulary [33] suitable for statistical data. The data that are stored in a QB file are loaded, via an ETL process, into the data warehouse. Then, the OLAP queries can be executed over the data. The advantage of the data stored as QB is that the measures and dimensions are already partly defined, so the transformation of data into the multidimensional model is easier. However, the method is not suitable for data expressed in other RDF vocabularies.

The execution of OLAP queries directly over an RDF store is explained in [16]. Statistical data defined with the help of on RDF Data Cube (QB) vocabulary are used. These data are loaded to a triple store. OLAP queries are translated to SPARQL queries and are run over the triple store. However, the proposed approach is applicable only to the data presented in QB. Moreover, the observations in the data should not include any aggregated values, otherwise the computation is incorrect.

In the majority of the current approaches [13,14,18] Linked Open Data are loaded into the relational tables of a data warehouse for further analysis. Our approach does not require a relational database for the OLAP analysis of web data. Additionally, our approach handles all types of RDF data unlike the proposal of [16], where only data stored as RDF Data Cubes (QB) are processed. Furthermore, our approach retrieves data from multiple sources whereas other approaches work with a single source of information at a time.

### 5.2 RDF Source Discovery

Heim et al. [10] propose an approach that automatically reveals relationships between two known objects in a large knowledge base such as DBpedia and displays them as a graph. They use properties in semantically annotated data to automatically find relationships between any pair of user-defined objects and visualize them. Although this approach is not relevant to source discovery the idea of searching through knowledge bases may be applicable to it.

Exploring Linked Data principles for finding data sources is proposed in [9]. One of these principles includes the usage of HTTP-based URIs as identifiers, which may be understood as a data link that enables the retrieval of data by looking up the URI on the Web. Hence, by exploring data during the query execution process one can obtain potentially relevant data for the system. However, this technique is less suitable for bulk retrieval of RDF data, which is needed for OLAP processing.

The publication of Linked Data as services is investigated in [22,23]. The use of Web Services and Service Oriented Architecture (SOA) is explored in this work. SOA facilitates easier data exchange between parties. A key component of SOA is the service repository, which serves the purpose of publishing and discovering services for future use. Research on service repositories for Web Services were extensive but the approach did not receive widespread adoption and was discontinued later. The main problem was the lack of support for expressive queries to identify and automate the discovery and consumption of services [23]. To address this problem, researchers propose to semantically annotate service descriptions to aid automatic discovery. Unfortunately, this technology did not receive widespread adoption either. If such a universal registry for services that publish Linked Data is created, a discovery and consumption of Linked Data from previously unknown sources will become easier.

An architecture of creating an up-to-date database of RDF documents by involving user participation in discovery of semantic web documents is described in [3]. This database can be used by search engines and semantic web applications to provide search and user-friendly services over the discovered documents. However, the service does not support discovery of SPARQL endpoints – this part of the process is left for future work. Scalability issues are not considered and are left for future as well.

Search engines for the semantic web [11,20] index the semantic web by crawling RDF documents and offer a search API over these documents. Different search engines use different index types: some index triples/quads, some index RDF documents. These search engines create an infrastructure to support application developers in discovering relevant data by performing lookup using, for example, full-text search over the literals. In this paper we propose to use semantic web search engines to support the discovery of SPARQL endpoints.

In this paper we further enhance existing approaches. We elaborate on ideas from [13,19,25] to build a multidimensional schema from previously unknown RDF data sources. Moreover, we extend principles from [10,20] for SPARQL endpoint discovery by grouping related results by datasets. For increased reliability in source discovery, we propose to employ a combination of approaches. Additionally, we target our approach to non-professional data analysts.

### 5.3 Indexing and Distributed Query Processing

As Linked Data are scattered over the Web, efficient techniques for distributed query processing become an important part of the system. Regarding distributed query processing over multiple SPARQL endpoints, several approaches and frameworks were proposed in the past years. In contrast to the systems for source discovery mentioned above, most systems for distributed query processing over SPARQL endpoints rely on the presence of pre-computed indexes or statistics to identify the relevance of sources [6–8,24,28] and only a few frameworks can avoid the need of pre-computed information [26]. Whereas most systems specialize in one type of data access, exploratory data access or SPARQL endpoints, hybrid systems propose handling different types of native access [17], often in combination with local caching [29].

In addition to determining the relevance of sources for a given SPARQL query based on the binary decision whether a source provides data that is relevant to answer any part of a query, sources can be selected based on their benefit [12]. In doing so, additional aspects are considered such as the overlap of the data provided by available sources. As a result, the minimum number of sources that still produce the complete answer to the query can be selected.

## 6 Conclusions and Future Work

In this paper, we presented a framework for exploratory OLAP over LOD sources. We introduced a system that uses a multidimensional schema of the data cube expressed in QB4OLAP and VoID. Based on this multidimensional schema, the system is able to query data sources, extract and aggregate data, and build an

OLAP cube. We proposed to store multidimensional information retrieved from external sources in a QB4OLAP structure. We also introduced a computer-aided process for discovering previously unknown data sources necessary for the given data cube and building a multidimensional schema. We presented a use case to demonstrate the applicability of the proposed framework. In the future, we plan to finish the prototype of the proposed framework and test the solution on large-scale case studies.

**Acknowledgment.** This research is partially funded by the Erasmus Mundus Joint Doctorate in "Information Technologies for Business Intelligence – Doctoral College (IT4BI-DC)".

## Appendix

## A Prefixes Used in the Paper

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX exqb: <http://example.org/exqb#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
PREFIX lmdbres: <http://data.linkedmdb.org/resource/>
PREFIX geo: <http://www.geonames.org/ontology#>
PREFIX wgs84_pos: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX qb: <http://purl.org/linked-data/cube#> .
PREFIX qb4o: <http://purl.org/olap#> .
PREFIX xml: <http://www.w3.org/XML/1998/namespace> .
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> .
PREFIX skos: <http://www.w3.org/2004/02/skos/core#> .
PREFIX foaf: <http://xmlns.com/foaf/0.1/> .
PREFIX dc: <http://purl.org/dc/elements/1.1/> .
PREFIX db: <http://dbpedia.org/resource/> .
PREFIX ns: <http://rdf.freebase.com/ns/> .
```

## References

1. Abelló, A., Darmont, J., Etcheverry, L., Golfarelli, M., Mazón, J., Naumann, F., Pedersen, T.B., Rizzi, S., Trujillo, J., Vassiliadis, P., Vossen, G.: Fusion cubes: towards self-service business intelligence. IJDWM **9**(2), 66–88 (2013)
2. Abelló, A., Romero, O., Pedersen, T.B., Berlanga, R., Nebot, V., Aramburu, M.J., Simitsis, A.: Using semantic web technologies for exploratory OLAP: a survey. TKDE **99** (2014)
3. Bojars, U., Passant, A., Giasson, F., Breslin, J.G.: An architecture to discover and query decentralized RDF data. In: SFSW (2007)
4. Etcheverry, L., Vaisman, A., Zimányi, E.: Modeling and querying data warehouses on the semantic web using QB4OLAP. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 45–56. Springer, Heidelberg (2014)
5. Etcheverry, L., Vaisman, A.A.: QB4OLAP: a vocabulary for OLAP cubes on the semantic web. In: COLD (2012)
6. Görlitz, O., Staab, S.: SPLENDID: SPARQL endpoint federation exploiting VOID descriptions. In: COLD (2011)

7. Hagedorn, S., Hose, K., Sattler, K., Umbrich, J.: Resource planning for SPARQL query execution on data sharing platforms. In: COLD (2014)
8. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K., Umbrich, J.: Data summaries for on-demand queries over linked data. In: WWW, pp. 411–420 (2010)
9. Hartig, O.: Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 154–169. Springer, Heidelberg (2011)
10. Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., Stegemann, T.: RelFinder: revealing relationships in RDF knowledge bases. In: Chua, T.-S., Kompatsiaris, Y., Mérialdo, B., Haas, W., Thallinger, G., Bailer, W. (eds.) SAMT 2009. LNCS, vol. 5887, pp. 182–187. Springer, Heidelberg (2009)
11. Hogan, A., Harth, A., Umbrich, J., Kinsella, S., Polleres, A., Decker, S.: Searching and browsing linked data with SWSE: the semantic web search engine. J. Web Semant. **9**(4), 365–401 (2011)
12. Hose, K., Schenkel, R.: Towards benefit-based RDF source selection for SPARQL queries. In: SWIM, pp. 2:1–2:86 (2012)
13. Inoue, H., Amagasa, T., Kitagawa, H.: An ETL framework for online analytical processing of linked open data. In: Wang, J., Xiong, H., Ishikawa, Y., Xu, J., Zhou, J. (eds.) WAIM 2013. LNCS, vol. 7923, pp. 111–117. Springer, Heidelberg (2013)
14. Kämpgen, B., Harth, A.: Transforming statistical linked data for use in OLAP systems. In: I-SEMANTICS, pp. 33–40 (2011)
15. Kämpgen, B., Harth, A.: No size fits all - running the star schema benchmark with SPARQL and RDF aggregate views. In: ESWC, pp. 290–304 (2013)
16. Kämpgen, B., O'Riain, S., Harth, A.: Interacting with statistical linked data via OLAP operations. In: ILD, pp. 336–353 (2012)
17. Ladwig, G., Tran, T.: Linked data query processing strategies. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 453–469. Springer, Heidelberg (2010)
18. Nebot, V., Berlanga, R.: Building data warehouses with semantic web data. Decis. Support Syst. **52**(4), 853–868 (2012)
19. Neumann, T., Moerkotte, G.: Characteristic sets: accurate cardinality estimation for RDF queries with multiple joins. In: ICDE, pp. 984–994 (2011)
20. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com: a document-oriented lookup index for open linked data. IJMSO **3**(1), 37–52 (2008)
21. Pedersen, D., Riis, K., Pedersen, T.B.: XML-extended OLAP querying. In: SSDBM, pp. 195–206 (2002)
22. Pedrinaci, C., Domingue, J.: Toward the next wave of services: linked services for the web of data. J.UCS **16**, 1694–1719 (2010)
23. Pedrinaci, C., Liu, D., Maleshkova, M., Lambert, D., Kopecky, J., Domingue, J.: iServe: a linked services publishing platform. In: ORES (2010)
24. Prasser, F., Kemper, A., Kuhn, K.: Efficient distributed query processing for autonomous RDF databases. In: EDBT, pp. 372–383 (2012)
25. Romero, O., Abelló, A.: Automating multidimensional design from ontologies. In: DOLAP, pp. 1–8. ACM (2007)
26. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: optimization techniques for federated query processing on linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 601–616. Springer, Heidelberg (2011)

27. Sheth, A., Larson, J.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput. Surv. **22**(3), 183–236 (1990)
28. Umbrich, J., Hose, K., Karnstedt, M., Harth, A., Polleres, A.: Comparing data summaries for processing live queries over linked data. WWWJ **14**(5–6), 495–544 (2011)
29. Umbrich, J., Karnstedt, M., Hogan, A., Parreira, J.X.: Hybrid SPARQL queries: fresh vs. fast results. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 608–624. Springer, Heidelberg (2012)
30. Vaisman, A., Zimányi, E.: Data Warehouse Systems: Design and Implementation. Springer, New York (2014)
31. W3C. Describing linked datasets with the VoID vocabulary (2010). http://www.w3.org/TR/void/
32. W3C. Data W3C (2013). http://www.w3.org/standards/semanticweb/data
33. W3C. The RDF data cube vocabulary (2013). http://www.w3.org/TR/2013/CR-vocab-data-cube-20130625/
34. W3C. W3C semantic web activity homepage (2013). http://www.w3.org/2001/sw
35. Semantic Web. SPARQL endpoint (2013). http://semanticweb.org/wiki/SPARQL_endpoint