WaRG: Warehousing RDF Graphs
Dario Colazzo, Tushar I. Ghosh, François Goasdoué, Ioana Manolescu, Alexandra Roatis

To cite this version:
Dario Colazzo, Tushar I. Ghosh, François Goasdoué, Ioana Manolescu, Alexandra Roatis. WaRG: Warehousing RDF Graphs. Bases de Données Avancées, Oct 2013, Nantes, France. hal-00868670

HAL Id: hal-00868670
https://inria.hal.science/hal-00868670
Submitted on 1 Oct 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
WaRG: Warehousing RDF Graphs

Dario Colazzo† Tushar I. Ghosh§,* François Goasdoué† Ioana Manolescu†,* Alexandra Roatis†,*

§Inria Saclay, France †Université Paris-Sud, France ‡Université Paris Dauphine, France †Université Rennes 1, France
firstname.lastname †@inria.fr •@iri.fr ‡dauphine.fr †@univ-rennes1.fr

ABSTRACT
We propose to demonstrate WaRG, a system for performing warehouse-style analytics on RDF graphs. To our knowledge, our framework is the first to keep the warehousing process purely in the RDF format and take advantage of the heterogeneity and semantics inherent to this model.

1. INTRODUCTION
Databases of facts, each characterized by multiple dimensions, whose values are recorded in measures, are at the core of multidimensional data warehouses (DWs in short) [13]. The facts can then be analyzed by means of aggregating the measures, e.g., “what is the average sale price of item I every month in every store?”

Data warehouses are typically built to analyze (some aspects of) an enterprise’s business processes. First, analysts choose the data sources of interest for a given class business questions. Next, they describe the facts, dimensions, and measures to be analyzed. Then, for each relevant business question, an analytical query (commonly known as a cube) is formulated, by (i) classifying facts along a set of dimensions and (ii) reporting the aggregated values of their measures. Data warehousing has attracted enormous interest, both from practitioners, e.g., [14] and from the research community, e.g., [8, 12]; warehousing tools are now part of major relational database servers. Relational data warehousing can be thus considered a pretty mature area.

Recent years witnessed a steady and important interest in Semantic Web data, represented within the W3C’s Resource Description Framework (or RDF, in short) [18]. The RDF model allows describing resources, by specifying the values of their properties. The RDF language is used to export, share, and collaboratively author data, in many settings. It serves as a metadata language, e.g., to describe cultural artifacts in large digital libraries, and to encode protein sequence data (e.g., the Uniprot data set). RDF is a natural target for representing heterogeneous facts contributed by millions of Wikipedia users, gathered within the DBpedia data source, as well as for the Linked Open Data effort, aiming at connecting and sharing collectively produced data and knowledge.

The current popularity of RDF raises interest in models and tools for RDF data analytics. Consider applications seeking to harvest, aggregate and analyze user data from various sources (such as social networks, blog posts, comments on public Web sites). The data is heterogeneous (facts about the user age, gender, region, endorsements) and graph-structured (describes relationships between users, places, companies). It comes from multiple sources and may have attached semantics, based on ontologies for which RDF is an ideal format. Analyzing such data with warehouse-style tools enables understanding user behavior and needs, and proposing services better suited to the users.

Despite the perceived need, there is currently no satisfactory conceptual and practical solution for large-scale RDF analytics. Relational DW tools are not suitable, since loading RDF data in a relational analytical schema may lead to facts with unfilled or multiply-defined dimensions or measures; the latter does not comply with the relational multidimensional setting and DW tools. Further, to fully exploit RDF graphs, the heterogeneity and rich semantics of RDF data should be preserved through the warehouse processing chain and up to the analytical queries. In particular, RDF analytical queries should be allowed to jointly query the schema and the data, e.g., ask for most frequently specified properties of a CollegeStudent, or the top-three most represented groups of Inhabitants. Changes to the underlying database (e.g., a new subclass of Inhabitant) should not cause the warehouse schema to be re-designed; instead, the new resources (and their properties) should propagate smoothly to the analysis schema and cubes.

We propose to demonstrate WARG, a tool embodying our novel approach for warehouse-style analytics on Semantic Web data. WARG supports the first all-RDF formal model for data warehousing. Its main features are:

- a formal model for analytical schemas, capturing the data of interest for a given analysis of an RDF data set. Importantly, an analytical schema instance is an RDF graph itself, and as such, it preserves the heterogeneity, semantics, and flexible mix of schema and data present in the RDF model;
- analytical queries, RDF counter-parts of the relational analytical cubes, and supporting typical analytical operations (slice, dice, drill-down etc.)

In the sequel, Section 2 presents our analytical RDF model, the WARG platform architecture is discussed in Section 3 and we describe use scenarios in Section 4. Finally, we discuss related work in Section 5.

2. DATA MODEL
An RDF graph is a set of triples of the form s p o. A triple states that its subject s has the property p, and the value of that property is the object o. We only consider well-formed triples, s ∈ U ∪ B, p ∈ U and o ∈ U ∪ B ∪ L, where U is a set of URIs, L of literals (constants), and B are blank nodes (unknown URIs or literals), such that U, B and L are pairwise disjoint.
The RDF standard provides a set of built-in classes and properties, part of the rdf: and rdfs: pre-defined namespaces. We use these namespaces exactly for these classes and properties, e.g., rdf:type which specifies the class(es) to which a resource belongs.

Trips are used to describe resources, i.e., to express class a rdf:type o (unary relation – o(a)) and property s p o (binary relation – p(s, o)) assertions.

A valuable feature of RDF is RDF Schema (RDFS) which enriches the descriptions in RDF graphs, by declaratively states which a resource belongs.

2.1 RDF graph analysis

We model a schema for RDF graph analysis, called analytical schema, as a labeled directed graph. Each node represents a set of facts that may be analyzed, while

Figure 2: Sample Analytical Schema (AnS).

nodes reachable from its outgoing edges represent the possible dimensions and measures according to which it may be analyzed. Nodes (respectively edges) correspond to class (respectively property) assertions, defined using BGP queries.

Definition 1 (Analytical Schema). An analytical schema (AnS) is a labeled directed graph \( \langle N, E, \lambda, \delta \rangle \) in which:

- \( N \) is the set of nodes;
- \( E \subseteq N \times N \) is the set of directed edges;
- \( \lambda : N \cup E \rightarrow URI \) is an injective labeling function, mapping nodes and edges to URIs;
- \( \delta : N \cup E \rightarrow Q \) is a function assigning to each node \( n \in N \) a unary BGP query \( \delta(n) = q(x) \), and to every edge \( n \rightarrow n' \in E \) a binary BGP query \( \delta(n \rightarrow n') = q(x, y) \).

We assume that through the \( \lambda \) function, each node in the AnS defines a new class, while each edge defines a new property. Since \( \lambda \) is injective, the URIs assigned to the classes and properties defined by an AnS’ nodes and edges are pairwise disjoint. Just as an analytical schema defines the data available to the analyst in a typical relational data warehouse scenario, in our framework, the classes and properties created by AnS (and labeled by \( \lambda \)) are the only ones visible to further RDF analytics. Example 2 introduces an AnS for our running example.

Example 1 (Running Example). We consider an RDF graph comprising information about users and products (depicted in Figure 1). The graph features a resource user1 whose name is Bill and whose age is 28. Bill works with user2 and is a friend of user5; moreover, Bill is an active contributor to two different blogs, one shared with his co-worker user2. Bill bought a smartphone and rated it online. The database also contains information about the RDFS schema for this graph.

We consider the well-known subset of SPARQL consisting of (unions of) basic graph pattern (BGP) queries, also known as SPARQL conjunctive queries. A BGP is a set of triple patterns, or triples in short. Each triple has a subject, property, and object. Subjects and properties can be URIs, blank nodes or variables; objects can also be literals.

A boolean BGP query is of the form \( \text{ASK WHERE } \{ t_1, \ldots, t_o \} \), while a non-boolean BGP query is of the form \( \text{SELECT x WHERE } \{ t_1, \ldots, t_o \} \), where \( \{ t_1, \ldots, t_o \} \) is a BGP; the variables \( x \) are a subset of the variables occurring in \( t_1, \ldots, t_o \). We use the conjunctive query notation \( q(x) \) for \( t_1, \ldots, t_o \) for both ASK and SELECT queries (for boolean queries, \( x \) is empty).

2.1 RDF graph analysis

We model a schema for RDF graph analysis, called analytical schema, as a labeled directed graph. Each node represents a set of facts that may be analyzed, while

Figure 1: Running example: RDF/S graph.
Table 1: Labels and queries of some nodes and edges of the analytical schema (AnS) shown in Figure 2.

Table 2: Partial instance of the AnS in Figure 2.

2.2 Analytical Queries

Data warehouse analysis summarizes facts according to relevant criteria into cubes, which analyze facts characterized by some dimensions, using a measure. We consider a set of dimensions \(d_1, d_2, \ldots, d_n\), such that each dimension \(d_i\) may range over the value set \(\{d^1_i, \ldots, d^n_i\}\); the Cartesian product of all dimensions \(d_1 \times \cdots \times d_n\) defines a multidimensional space \(M\). To each tuple \(t\) in this multidimensional space \(M\) corresponds a subset \(F_t\) of the analyzed facts, having for each dimension \(d_i\), \(1 \leq i \leq n\), the value of \(t\) along \(d_i\).

A measure is a set of values characterizing each analyzed fact \(f\). The facts in \(F_t\) are summarized by the cube cell \(M[t]\) by the result of an aggregation function \(\oplus\) (e.g., count, sum, average, etc.) applied to the union of the measures of the \(F_t\) facts: \(M[t] = \bigoplus (\bigcup_{f \in F_t} v)\).

An analytical query consists of two (rooted) queries and an aggregation function. The first query, known as a classifier in traditional data warehouse settings, defines the dimensions \(d_1, d_2, \ldots, d_n\) according to which the facts matching the query root will be analyzed. The second query defines the measure according to which these facts will be summarized. Finally, the aggregation function is used for summarizing the analyzed facts.

Definition 3 (Analytical Query). Given an analytical schema \(S = \langle N, E, \lambda, \delta \rangle\), an analytical query \((AnQ)\) rooted in the node \(r \in N\) is a triple

\[
Q = \langle c(x, d_1, \ldots, d_n), m(x, v), \oplus \rangle
\]

where:

- \(c(x, d_1, \ldots, d_n)\) is a query rooted in the node \(r_\mathfrak{c}\) of its graph \(G_r\). This query is called the classifier of \(x\) w.r.t. the \(n\) dimensions \(d_1, \ldots, d_n\).
- \(m(x, v)\) is a query rooted in the node \(r_m\) of its graph \(G_m\). This query is called the measure of \(x\).
- \(\oplus\) is a function computing a value (a literal) from an input set of values. This function is called the aggregator for the measure of \(x\) w.r.t. its classifier.
- For every homomorphism \(h_c\) from the classifier to \(S\) and every homomorphism \(h_m\) from the measure to \(S\), \(h_r(r_c) = h_m(r_m) = r\) holds.

The last item above guarantees the “well-formedness” of the analytical query, that is: the facts for which we aggregate the measure, are indeed those classified along the desired dimensions. It is worth noticing that, from a practical viewpoint, this condition can be easily and naturally guaranteed by giving explicitly in the classifier and the measure either the type of the facts to analyze, using \(x\) rdf:type \(\lambda(r)\), or a property describing those facts, using \(x\) \(\lambda(r \rightarrow n)\) \(\circ\) with \(r \rightarrow n \in E\). As a result, since the labels are unique in an AnS (its labeling function is injective), every homomorphism from the classifier (respectively the measure) to the AnS does map the query’s root node labeled with \(x\) to the AnS’s node \(r\).

Definition 3 can be easily extended to more than one measure, by defining a set of measure queries and an associate set of aggregation functions.

Example 4 (Analytical Query). The next query asks for the number of blogs where the user posts, classified by the user age and city:

\[
(c(x, a, c), m(x, y), \text{count})
\]

where the classifier and measure queries are defined by:

\[
c(x, a, c) \text{:- } x \text{ personAge } a, x \text{ personCity } c
\]

\[
m(x, b) \text{:- } x \text{ wroteMessg } a, o \text{ messgInBlog } b
\]
The semantics of an analytical query is:

**Definition 4 (Answer Set of an AnQ).** Let $\mathcal{I}$ be the instance of an AnS with respect to some RDF graph. Let $Q = (c(x, d_1, \ldots, d_n), m(x, v), \oplus)$ be an AnQ against $\mathcal{I}$. The answer set of $Q$ against $\mathcal{I}$, denoted $\text{ans}(Q, \mathcal{I})$, is:

$$\text{ans}(Q, \mathcal{I}) = \{ (\hat{d}_1^{\prime}, \ldots, \hat{d}_n^{\prime}, \oplus(q^I(\mathcal{I}))) \mid (x_1, d_1^{\prime}, \ldots, d_n^{\prime}) \in c(\mathcal{I}) \text{ and } q^I \text{ is defined as } q^I(v) \mapsto m(x_1, v) \}$$

assuming that the type of each value returned by $q^I(\mathcal{I})$ belongs (or can be converted by the SPARQL rules [19]) to the input type of the aggregator $\oplus$. Otherwise, the answer set is undefined.

The analytical query returns each tuple of dimension values from the answer of the classifier, together with the aggregated result of the measure query. The answer set of an AnQ can thus be represented as a cube of $n$ dimensions, holding in each cube cell the corresponding aggregate measure.

**Example 5 (Analytical Query Answer).** Consider the query in Example 4 over the analytical schema in Figure 2. Some triples from the instance of this analytical schema were shown in Table 2. The classifier query returns:

$$(\text{user}_1, 28, \text{"Madrid"}), (\text{user}_3, 35, \text{"NY"})$$

while the measure query returns:

$$(\text{user}_1, \text{blog}_1), (\text{user}_1, \text{blog}_2), (\text{user}_3, \text{blog}_2)$$

Aggregating the blogs among the classification dimensions leads to the AnQ answer:

$$(28, \text{"Madrid"}, 2), (35, \text{"NY"}, 1)$$

3. **ARCHITECTURE**

We deploy our approach using kdb+ v3.0 (64 bits) [1], an in-memory column database used in decision support analytics applications. kdb+ provides arrays (tables), which can be manipulated through the interpreted programming language. We store in kdb+ the RDF graph $G$, the AnS definitions and AnS instance, when we choose to materialize it. We also translate BGP queries into q queries that kdb+ interprets.

Figure 3 outlines our data layout. As customary in RDF databases, we dictionary-encode all URIs into integers; these mappings are stored within the [d] table. The graph saturation is stored in [t]. Analytical schema definitions are stored as follows. The [a] table stores the $\lambda$ function encoded as triples $\lambda(n_1) \mapsto \lambda(n_2) \mapsto \lambda(n_3)$ for all the nodes $n_1$, $n_2$, and $n_3$. The [ie] table stores the $\delta$ function for AnS nodes, and similarly [ae] stores the $\delta$ function for AnS edges. Finally, we use either the [i] table to store the AnS instance $\mathcal{I}$, or several [in] and [ie] tables when considering partitioned tables.

The demo will rely on a subset of DBPedia.

**Visualizing analytical schemas** Figure 4 shows an example of a small analytical schema shown in our GUI. Our visualization tool allows selecting a node, which results in highlighting its connected nodes within the schema. This feature is useful when building an analytical query, since such a query must be a connected component from the schema.
example the edge between dbpo:PopulatedPlace and dbpo:Language in Figure 3. Selecting an edge in a graph leads to obtaining information about the edge itself, and the same holds for edges. When a multiple edge is selected, all the edges are displayed (Table 3).

**Evaluating queries** Users may build and evaluate analytical queries over a defined schema. Building an analytical query consists of defining each of its three components: classifier query, measure query and aggregation function. The classifier query is built starting from a selected node. The system then suggests (highlights) the nodes connected to it. As more nodes are selected, their neighbors are highlighted as potential options. Only nodes that have underlying data are considered at any time, because the intention is to build queries that can have answers. A selected node is interpreted as a variable in the query. The user may also add constants by selecting values from the node’s underlying data. The process goes on until the classifier is complete and the dimensions (distinguished variables) are selected. The measure query is built similarly. Finally, an aggregation function is chosen from a drop-down list of functions applicable to the chosen measure.

### 5. RELATED WORKS

Relational DWs have been thoroughly studied, and many efficient tools exist. In the context of the Web, “Web Data Warehouses” have been proposed, however, their focus is on distribution, data exchange or enrichment, rather than RDF and analytics. Some works propose new vocabularies for describing relational cubes in RDF; in contrast, WARG is a generic approach for defining and using pure RDF cubes. Data management techniques for RDF have been a hot topic, with recent works tackling storage, query processing and updates. Materialized views and Map-Reduce based RDF processing are examples. Oracle Hg provides a “Semantic Graph” extension etc. Ours is the first work to formalize RDF analytics and propose analytic schema and queries with well-defined semantics. Recent works have focused on graph warehousing. While we share certain general principles, these approaches are not meant for heterogeneous graphs and thus cannot handle multi-valued attributes. More importantly, they do not consider the semantically rich RDF model.

The separation between grouping and aggregation present in our analytical queries is similar to the MD-join operator introduced in relational DWs.

Finally, the recent SPARQL 1.1 language includes group-by and aggregation constructs closely inspired from SQL. While this bears similarities to our AnQs, our proposal is the first to outline a full-RDF data analytics chain, including a general framework for specifying analytical schemas which are RDF graphs themselves, and therefore carry all the rich features of the data model, most importantly structure heterogeneity and rich semantics.

### 6. REFERENCES


<table>
<thead>
<tr>
<th>Short URI</th>
<th>dbpo:Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full URI</td>
<td><a href="http://dbpedia.org/ontology/Person">http://dbpedia.org/ontology/Person</a></td>
</tr>
<tr>
<td>Query</td>
<td>q(x): x rdf:type dbpo:Person</td>
</tr>
<tr>
<td>Is empty</td>
<td>FALSE</td>
</tr>
<tr>
<td>Data</td>
<td>dbpr: Aristotle, dbpr: NeilArmstrong, dbpr: TimBerners-Lee</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Available edges</th>
<th>dbpo:language, dbpo:officialLanguage, dbpo:regionallanguage</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Short URI</th>
<th>dbpo:language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full URI</td>
<td><a href="http://dbpedia.org/ontology/language">http://dbpedia.org/ontology/language</a></td>
</tr>
<tr>
<td>Query</td>
<td>q((x/y): x dbpo:language)</td>
</tr>
<tr>
<td>Is empty</td>
<td>FALSE</td>
</tr>
<tr>
<td>Data</td>
<td>dbpr: France-dbpr: FrenchLanguage, dbpr: Germany-dbpr: GermanLanguage, dbpr: United_Kingdom-dbpr: EnglishLanguage</td>
</tr>
</tbody>
</table>

Table 3: Node (top) and edge (bottom) information.