Mixed-instance querying: a lightweight integration architecture for data journalism
Raphaël Bonaque, Tien Duc Cao, Bogdan Cautis, François Goasdoué, Javier Letelier, Ioana Manolescu, Oscar Mendoza, Swen Ribeiro, Xavier Tannier, Michaël Thomazo

To cite this version:
Raphaël Bonaque, Tien Duc Cao, Bogdan Cautis, François Goasdoué, Javier Letelier, et al.. Mixed-instance querying: a lightweight integration architecture for data journalism. VLDB, Sep 2016, New Delhi, India. hal-01321201v2

HAL Id: hal-01321201
https://inria.hal.science/hal-01321201v2
Submitted on 27 May 2016

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.
Mixed-instance querying: a lightweight integration architecture for data journalism

R. Bonaque\textsuperscript{1,2} T. D. Cao\textsuperscript{1,2} B. Cautis\textsuperscript{2,3,4} F. Goasdoué\textsuperscript{5} J. Letelier\textsuperscript{1,2} I. Manolescu\textsuperscript{1,2} O. Mendoza\textsuperscript{1,2} S. Ribeiro\textsuperscript{2,4,6} X. Tannier\textsuperscript{2,4,6} M. Thomazo\textsuperscript{1,2}

\textsuperscript{1}INRIA & LIX, École Polytechnique\textsuperscript{2}U. Paris-Saclay\textsuperscript{3}LRI\textsuperscript{4}U. Paris-Sud\textsuperscript{5}IRISA, U. Rennes 1 & INRIA\textsuperscript{6}LIMSI, CNRS

ABSTRACT

As the world’s affairs get increasingly more digital, timely production and consumption of news require to efficiently and quickly exploit heterogeneous data sources. Discussions with journalists revealed that content management tools currently at their disposal fall very short of expectations. We demonstrate TATTOOINE, a lightweight data integration prototype, which allows to quickly set up integration queries across (very) heterogeneous data sources, capitalizing on the many data links (joins) available in this application domain. Our demonstration is based on scenarios we study in collaboration with Le Monde, France’s major newspaper.

1. MOTIVATION AND OUTLINE

Data, social networks, and the media

As the world’s affairs get increasingly digital, the production and consumption of news are massively impacted. First, data that journalists may work on is more and more digitized: structured databases, such as government-gathered data (demographics, economics, taxes, elections, etc.), legal records, stock quotes for companies; un-structured and semi-structured datasets, all can be exploited for news analysis. For instance, large-scale RDF datasets, endowed with ontologies, such as DBPedia or Yago, incorporate significant amounts of manual and automatic effort that led to structuring information about a wide variety of places, people, major events, and concepts. These sources of open data can be readily queried through SPARQL endpoints, e.g., data sources in the Linked Open Data cloud (lod-cloud.net). High-quality content may also be organized in structured text form (HTML or XML), e.g., (interrelated) laws and regulations, public speeches etc. Second, the arena of interactions between organizations (countries, government, corporations, media) and/or individuals (from politicians to bloggers), which journalism follows and analyzes, is increasingly digital, with the Web and social networks as communication means of choice. The latter allows journalists to examine connections between public and unknown figures in social networks\footnote{A Twitter account has been correctly identified as belonging to French politician Marine Le Pen, by analyzing its posts and its social connections: \url{bit.ly/1Lxi27C}} and analyze online interaction traces between these actors.

The need for dynamic data journalism tools

A novel generation of data journalists has emerged, well aware of the potential of computerized databases, and eager to combine in innovative ways both static and dynamic information coming from structured, semi-structured, and un-structured databases and social feeds. Nevertheless, feedback collected during discussions with the data journalism and fact-checking team Les Décodeurs (lemonde.fr/les-decodeurs) of the major French journal “Le Monde”, as well as with journalists from “The Washington Post” and “The Financial Times” reveal that current content management tools at their disposal are disparate and limited, falling very short of expectations. Furthermore, the very short news cycles for many news stories, the frequent emergence of new sources, as well as the eagerness to be the first to exploit them, do not entice journalists nor media companies to invest time, effort, and funding into devising and filling a “standard” data warehouse comprising all types of information.

Our approach: lightweight integration architecture

We propose to demonstrate TATTOOINE, a lightweight data integration prototype we have devised based on our discussions with Les Décodeurs. TATTOOINE allows journalists to exploit heterogeneous data sources of different data models, which we view as a mixed data instance. Within the instance, we distinguish (Figure\textsuperscript{1}) a set of (structured, un-structured, or semi-structured) data sources, of various data models, each of which resides within a system providing some query capabilities over its data. Examples in the figure are: a set of Tweets stored within Apache Solr, a full-text search platform; a set of relational curated databases, such as those provided by INSEE, the French institute of economic and social statistics, or the Ministry of Interior, which compiles detailed results of national and regional elections; and a set of RDF data sources, such as French territory description data from the National Geographic Institute (IGN), and LOD sources, in particular DBPedia, etc.

The mixed instance also comprises a custom (application-dependent) RDF data graph depicted under the form of black round nodes connected by edges above the data sources. This graph comprises an ontology (set of classes and properties) and/or data triples specific to a certain data journalism domain. For instance, when working on an article about French politics, a journalist may load a specialized data source classifying French political parties into currents and giving the affiliation of each party in the European Parliament; as there are very few parties, such a data source is typically put together manually. Another data source may hold all the elected representatives of the people, as they can be extracted from...
we found out Les Décodeurs had scraped these as a one-time effort, and stored them in a simple tabular file. Whatever the sector, be it Middle East politics, EU education, or French agriculture, journalists do tend to have small-to-significant size data sources, typically under the form of a text or tabular file, which can be easily exported into RDF.

A crucial remark we made while looking at the journalists’ most commonly used data sources is that repeated values across sources are frequent. This is due to several reasons, among which: (i) link-richness: a public figure’s profile on, say, Twitter, also provides his Facebook ID, his web page, and possibly the web page of his main political party. This is because public figures want to attract attention, be easy to find, and easy to associate (in the reader’s mind) to their organizations, such as parties or NGOs; (ii) common naming for humans: media authors, whether public figures, communication managers, journalists, etc. have a strong interest in allowing readers to easily relate an article to a topic. Thus, staple terms such as “unemployment” or “migrants” are consistently used by publishers hoping to attract readers; the usage of Twitter hashtags is another striking example; (iii) common naming for machines (part of the so-called Search Engine Optimizations), where publishers pick non-ambiguous, name-rich titles and representative article snippets to facilitate indexing and lookup through search engines. For instance, “Sarkozy visits Libya” is systematically preferred to the more ambiguous “The President visits Country at War”. Common naming for machines also includes codes (such as “MSFT” for Microsoft in NASDAQ, or “75” for the French department enclosing Paris) defined by the administration or corporations, which are widely adopted and frequently used in databases and media.

RDF data often features repeated Uniform Resource Identifiers (URIs), since RDF advocates (re)using URIs to refer to resources. Moreover, RDF graphs may also include non-URI values, such as names or codes, appearing in other data sources. In Fig. 1, joins between the custom RDF graph and the sources are depicted by RDF edges having one end “in a data source”. When the custom graph joins with several data sources, it effectively provides a “bridge” across them. RDF is a model of choice for expressing such bridges, as its flexibility (lack of structural constraints) makes it easy to add to a graph any piece (no matter how small) of useful information.

**Mixed querying in TATOOINE** To exploit the join opportunities provided by these repeated values (URIs, names, keywords, codes), we devise our integration (mixed query) engine (Figure 1) as a mediator, running on top of the different data sources, oftentimes (but not necessarily) interconnected by means of custom application-dependent RDF data. The engine evaluates mixed queries, which combine sub-queries expressed in the query language(s) of several heterogeneous sources, and RDF querying on the custom data. A sample query is, for instance: “for a given hashtag and each political affiliation (left, right etc.), find the most prolific tweet authors of that affiliation having used that hashtag, and their Facebook accounts”. Mixed queries can be used also to query dynamically discovered datasets, e.g., the address of a relational database is found in an INSEE table and part of the mixed query is shipped there for evaluation. Writing such queries requires some database skill. To further simplify the interaction with the mixed data instance, we have also devised a keyword-based query engine, which exploits data source digests TATOOINE computes from the sources. Based on these digests, the keyword-based query engine identifies a set of mixed queries which, evaluated over the set of (joining) datasets, return the results users are interested in. Finally, data visualizations can be devised to interact with (the results of queries over) mixed instances; these are particularly effective data journalism tools.

### Our contributions
- (i) an RDF-centric lightweight integration approach for an arbitrarily varied set of data sources (with heterogeneous models and query languages which we exploit as much as possible) and
- (ii) a novel approach for keyword search on such a heterogeneous instance, based on source digests, making it easy for non-expert users to discover valuable connections in the data.

## 2. MIXED CONTENT QUERYING

In this section, we introduce now mixed content instances (Section 2.1) and tools for querying them (Section 2.2).

### 2.1 Mixed content

RDF data and queries RDF is a W3C standard to describe Web data and knowledge. It allows expressing triples, of the form $s$, $p$, $o$, meaning that subject $s$ has the property $p$ whose value is $o$. Sample data description using RDF triples are

$$LeMonde rdf:type Organization,$$

and

$$LeMonde foundedIn "1944".$$  

we denote by $G^s$, $G^o$, $G^p$ $s$, $o$, and $p$, respectively the subjects or objects of a property. E.g., journalists are employees translates to

$$Journalist rdfs:subclass Employee.$$  

worksFor rdf:type Journalist. RDF Schema can express domain specific knowledge about the classes and properties used in data descriptions notably using four popular properties: rdfs:subclass and rdfs:subproperty state that a class or property is a specialization (particular case) of another; rdfs:domain and rdfs:range type respectively the subjects or objects of a property. E.g., journalists are employees translates to

$$Journalist rdfs:subclass Employee.$$  

worksFor rdf:type Journalist. RDF Schema can express domain specific knowledge about the classes and properties used in data descriptions notably using four popular properties: rdfs:subclass and rdfs:subproperty state that a class or property is a specialization (particular case) of another; rdfs:domain and rdfs:range type respectively the subjects or objects of a property. E.g., journalists are employees translates to

$$Journalist rdfs:subclass Employee.$$  

worksFor rdf:type Journalist. RDF Schema can express domain specific knowledge about the classes and properties used in data descriptions notably using four popular properties: rdfs:subclass and rdfs:subproperty state that a class or property is a specialization (particular case) of another; rdfs:domain and rdfs:range type respectively the subjects or objects of a property. E.g., journalists are employees translates to

$$Journalist rdfs:subclass Employee.$$  

worksFor rdf:type Journalist.

For a given RDF graph $G$, we denote by $G^s$, $G^o$, $G^p$ $s$, $o$, and $p$, respectively the subjects or objects of a property. E.g., journalists are employees translates to

$$Journalist rdfs:subclass Employee.$$  

worksFor rdf:type Journalist.

For a given RDF graph $G$, we denote by $G^s$, $G^o$, $G^p$ $s$, $o$, and $p$, respectively the subjects or objects of a property. E.g., journalists are employees translates to

$$Journalist rdfs:subclass Employee.$$  

worksFor rdf:type Journalist.

For a given RDF graph $G$, we denote by $G^s$, $G^o$, $G^p$ $s$, $o$, and $p$, respectively the subjects or objects of a property. E.g., journalists are employees translates to

$$Journalist rdfs:subclass Employee.$$  

worksFor rdf:type Journalist.

A prominent language for querying RDF graphs is that of basic graph pattern queries (BGPs, in short), the SPARQL subset corresponding to conjunctive queries). A BGP is of the form:

$$\phi(\vec{x}) := t_1, \ldots, t_n\)$$

where $\phi(\vec{x})$ is the query head, with $q$ the query name and $\vec{x}$ its output variables projected from the query body $t_1, \ldots, t_n$, which is a conjunction of triple patterns, i.e., triples whose subject, property and object positions can also hold variables.

The evaluation of a BGP $\phi$ on a graph $G$ is classically defined as the set of $\phi(\vec{x})$ tuples obtained by all possible embeddings $\phi$ of $q$ into $G$. The answer of $q$ over $G$ is defined as the evaluation $q$ against $G^q$, i.e., all the tuples, explicit or implicit, holding in $G$.

SPARQL queries also allow disjunction, optional pattern matching, construction of RDF triple results, aggregation etc.  

**Mixed instances** We assume a given set of data models, and one or several query languages for each. For example, an RDF source may accept SPARQL, an XML one may accept XPath or XQuery.

**Definition 2.1** (Mixed instance). A mixed instance $I = (G, D)$ is an RDF graph $G$ together with a set of data sources $D$.  

---

*Some data journalists are very tech-savvy, but many others clearly prefer just using (and imagining!) data management tools.*
Figure 2: Sample tweet from the mixed data instance $I$.

As a running example, we consider a mixed instance $I$ as sketched in Figure 1: its RDF graph $G$ contains information on entities related to the French political landscape such as:

- POL01140 rdf:type politician
- POL01140 position headOfState
- POL01140 foaf:name "François Hollande"
- POL01140 twitterAccount "fhollande"

$G$ also contains metadata on other RDF or relational data sources from $I$, such as the INSEE data source $d_{INSEE}$ containing for instance an SQL table for the Production and value-added of the agriculture in 2015 (en.fao.org). Finally, $I$ contains a Solr database of tweets, exemplified (in JSON) in Figure 2.

2.2 Mixed instance querying

To query mixed instances, we first propose Conjunctive Mixed Queries (CMQ, in short) of the form:

$$q(x) = q(x_1, q(x_2), q(x_3), \ldots, q_n(x_n) | d_{SA})$$

with, similar to BGPs, $q(x)$ being the query head, $q$ the query name, and $\vec{x}$ its output variables projected from the body. However, the body is a conjunction of a query $q_G$ over the custom application dependent RDF graph $G$ of a mixed instance $I = (G, D)$ (recall Fig. 1) and, optionally, of $q_1, \ldots, q_n$ queries over $D$ sources. Importantly, each of $d_1, \ldots, d_n$ is either a source URI or a variable (i.e., in $\bigcup_{i=0}^{n-1} \vec{x}_i$). CMQs generalize BGPs against RDF graphs.

The answers to $q$ are the set of $\phi(\vec{x})$ tuples obtained through all possible embeddings $\phi$ of $q$ into $I$, i.e., for which $(i)$ $\phi(\vec{x}_0)$ is an answer to $q_G$ against $G$ and $(ii)$ for every $i$, $\phi(\vec{x}_i)$ is an answer to $q_i$ against $d_i$, if $d_i$ is a source URI, or against $q(d_i)$ if $d_i$ a variable. Observe that this allows to identify at runtime, for each embedding $\phi$, a potentially different data source.

A sample CMQ looking for tweets from head of states about “SIA2016” (Salon International de L’Agriculture) is:

$$q_{SIA}(t, id) = q(t, id, tweetContains(t, id, SIA2016) | d_{Solr})$$

where $q(t, id):$ position headOfState, $t$ twitterAccount id and the Solr query $tweetContains$ yields pairs $(t, id)$ such that the tweet $t$ with the hashtag “SIA2016” was posted on the account id. If $d_{Solr}$ is a URI, then $tweetContains$ is only evaluated on the source identified by this URI, otherwise it is evaluated on every data source of the mixed instance that accepts it.

Keyword-based querying

Mixed queries are powerful and flexible, but writing them requires query language skills and knowledge of the custom RDF graph. This may be hard for journalists, thus we devise an approach to query over mixed instances through keyword queries. From each source, we build a digest, comprising:

1. its schema (if it has one; otherwise we use data-derived structural summaries, i.e., XML or JSON Dataguides, RDF summaries [2] etc.) and
2. a representation of the set of atomic values (constant URIs, etc.) associated to each position in the schema.

This could mean: the values of the attribute $\alpha$ in a relation $R$; the values found in the user.screenname attributes of all the tweets; the values of property position etc. If value set representations are provided by the data sources (e.g., an index on $R.\alpha$), TATOONIE exploits them; if not, TATOONIE builds and stores them externally.

The precision level of the value set representations is controlled by parameters dividing up the available space; histograms and Bloom filters are used. We view all digests as directed graphs (e.g., for a relational database, there is one node per attribute, one edge per key-foreign key constraint, etc.), and to each node we attach the representation of the set of data values corresponding to it.

Given the search keywords “head of state” and “SIA2016”, the engine looks up the keywords in (value set representations from) the digests; in our example, “head of state” appears in the custom RDF RDF part, while “SIA2016” appears as a hashtag. Following the approach of [7], the shortest join paths from “head of state” to “SIA2016” are then identified, which may be through the id of a head of state, then his Twitter id (these are found in the custom RDF), to the tweet tagged “SIA2016” as hashtag (found in Solr). Thus, the structured query $q_{kwa}$ is generated from the keywords.

2.3 Evaluating mixed instance queries

A mixed query is decomposed into a query over the custom RDF and one query per data source. Subqueries are then ordered for evaluation such that (i) bindings for data sources must be obtained before the source can be queried, (ii) parallelism is exploited when possible, and (iii) the most selective subqueries are executed first, in classical mediator style, and extending our experience with mixed XML-RDF querying [7]. The remaining processing (joins etc.) on subquery results takes place within our in-house iterator-based execution engine (Java, approx. 10K lines).

3. Demonstration Scenario

Dataset We will use a mixed instance we built over the last year as part of our collaboration with Les Décodeurs. It comprises:

(i) tweets from about 4,500 French politicians and political groups, updated daily since June 2015 (1.6M tweets at the time of writing). Author, retweet number, timestamp, and stemmed text are continuously indexed by an Apache Solr engine; (ii) 10K Facebook posts collected from verified pages of the most important French politicians since June 2015. Author, creation, and collection timestamp, stemmed text, number of likes, shares, and comments associated with each post are also indexed by a Solr instance; (iii) an RDF graph containing basic (name, gender, date and place of birth, nationality, spouse, etc.) and detailed (DBpedia URI, personal website, Twitter ID, Facebook ID, current political position, party affiliations, parliament and senate group affiliations etc.) information of top French politicians, as well as political parties and currents.

Scenarios demonstrated through mixed queries prepared before the presentation include:

1. Identify factual sources of information that relate to the claims made by a personality on Twitter, for instance the French President.
2. Compare vocabulary used by different parties on a user-defined topic, and show the most influential tweets on this topic (illustrated by tag clouds). And all terms $w$ used by each party $P$ in a set of tweets $Q$ (result of a mixed query) are ranked by their exponentiated point-wise mutual information (PMI, in short), comparing the probability of $w$ in the party to its global probability in the entire corpus. Probabilities are calculated with Maximum Likelihood Estimation [10]:

$$PMI(w, Q) = \frac{\sum_{t \in Q} \frac{n_{tw}}{n_{t}} \cdot \frac{n_{Qw}}{n_{Q}}}{\sum_{t \in P} \frac{n_{tw}}{n_{t}} \cdot \frac{n_{Qw}}{n_{Q}}}$$

where $t$ is a tweet from party $P$, $n_{tw}$ the count of word $w$ in tweet $t$, $n_{t}$ the number of words in tweet $t$, $N_{Qw}$ the total number of words in $Q$, and $n_{Qw}$ the count of word $w$ in $Q$. 


connection), and then (if the search is successful) of refining the consisting first of searching (for specific items and for data inter-
keyword-based query interface. This fits their data usage patterns, combining several sources in order to get results, and with a simpler urgent to provide the journalists with query tools capable of com-
mapping nor an integrated global schema, since it appeared more 
tation is also an important area. TATOOINE
is queried through an ontology, the setting resembles 

6. REFERENCES