RDFBroker
A Signature-Based High-Performance RDF Store

Michael Sintek, Malte Kiesel
DFKI GmbH, Kaiserslautern
\{sintek,kiesel\}@dfki.uni-kl.de

ESWC 2006, Budva

June 13, 2006
Outline

- Motivation
- RDFBroker Concepts
- Implementation
- Evaluation
- Conclusions and Future Work
Motivation

- RDF is lowest common denominator of knowledge representation formalisms, meant for interoperability between systems
- Not suited for
  - Direct use in general applications (applications’ data structures are typically object-oriented and not triple/graph-oriented)
  - Efficient handling in established databases (which are optimized for n-tuples, not binary predicates)
    - Accessing multiple properties of one resource involves accessing distributed parts ("pages")
    - Naive handling of triples leads to inefficient memory usage
Goals of Proposed Solution

- efficient import and export of RDF data
- adequate and efficient access from applications (like actors on the Semantic Web)
- schema independent
  - there is no “one-size-fits-all” schema/ontology language
  - schema might be incomplete or even non-existent
  - data is, at least at intermediate stages, simply not schema-compliant
use signatures (where a signature of a resource is the set of properties used on that resource)

allows Semantic Web data to be represented as normal (database) relations, with signatures being the database column headings

benefits:
- much more application-adequate
- much more efficient wrt. space and time
- queries that access multiple properties of a resource simultaneously benefit most
- benefit from database technology that has been optimized for performing queries on normal database tables

RDFBroker prototype:
http://rdfbroker.opendfki.de/
RDFBroker Concepts
Def. 1 The signature $\Sigma_G(s)$ of a resource $s$ wrt. an RDF graph $G$ is the set of properties that are used on $s$ in $G$:

$$\Sigma_G(s) = \{p \mid \exists o : \langle s, p, o \rangle \in G\}$$

Def. 2 The signature set $\Sigma_G$ for an RDF graph $G$ is the set of all signatures occurring in it, i.e.,

$$\Sigma_G = \{\Sigma_G(s) \mid \exists p, o : \langle s, p, o \rangle \in G\}$$
Def. 3 A signature $\Sigma(s_1)$ subsumes a signature $\Sigma(s_2)$ iff

$$\Sigma(s_1) \subseteq \Sigma(s_2)$$

Def. 4 The signature subsumption graph $G_G$ for an RDF graph $G$ is the directed acyclic graph with vertices $\Sigma_G$ and edges according to the subsumes relation between signatures, i.e., $G_G = (\Sigma_G, \subseteq)$.

The simplified signature subsumption graph $G'_G$ for an RDF graph $G$ is the graph that results from the signature subsumption graph by deleting all edges that can be reconstructed from the transitivity and reflexivity of $\subseteq$.
Sample graph $P$

Sample graph $P$

Signatures for the four subjects Person, $p1$, $p2$, $p3$:

$\Sigma_P(\text{Person}) = \{\text{rdf : type}\}$

$\Sigma_P(p1) = \{\text{rdf : type}, \text{rdfs : label}, \text{firstName}, \text{lastName}\}$

$\Sigma_P(p2) = \{\text{rdf : type}, \text{firstName}, \text{lastName}, \text{email}, \text{homepage}\}$

$\Sigma_P(p3) = \{\text{rdf : type}, \text{firstName}, \text{lastName}\}$
Signature set

\[ \Sigma_P = \{ \{ \text{rdf} : \text{type} \}, \{ \text{rdf} : \text{type}, \text{rdfs} : \text{label}, \text{firstName}, \text{lastName} \}, \{ \text{rdf} : \text{type}, \text{firstName}, \text{lastName}, \text{email}, \text{homepage} \}, \{ \text{rdf} : \text{type}, \text{firstName}, \text{lastName} \} \} \]

Simplified signature subsumption graph
Def. 5 The signature table $T_G(\{p_1, \ldots, p_n\})$ for a signature $\{p_1, \ldots, p_n\} \in \Sigma_G$ for an RDF graph $G$ is a two dimensional table with headings (rdf:about, $p_1, \ldots, p_n$) (where the $p_i$ are canonically ordered) and entries as follows: for each subject $s$ in $G$ with $\Sigma_G(s) = \{p_1, \ldots, p_n\}$, there is exactly one row in the table, where the rdf:about column contains $s$ and column $p_i$ contains the set of values for this property on $s$, i.e., $\{v \mid \langle s, p_i, v \rangle \in G\}$.

Def. 6 The signature table set $T_G$ for an RDF graph $G$ is defined as $T_G = \{T_G(s) \mid s \in \Sigma_G\}$. 
### Signature table set for $P$

<table>
<thead>
<tr>
<th>rdf:about</th>
<th>rdf:type</th>
<th>rdfs:label</th>
<th>firstName</th>
<th>lastName</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Person</td>
<td>“Michael”</td>
<td>“Michael”</td>
<td>“Sintek”</td>
</tr>
<tr>
<td>p2</td>
<td>Person</td>
<td>“Frank”</td>
<td>“Smith”</td>
<td>email</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rdf:about</th>
<th>rdf:type</th>
<th>firstName</th>
<th>lastName</th>
<th>email</th>
<th>homepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>Person</td>
<td>“Michael”</td>
<td>“Sintek”</td>
<td>http://</td>
<td></td>
</tr>
</tbody>
</table>
foundation for queries (and rules) by defining the algebraic operations used in (relational) databases

two sets of database operators

- directly operating on RDF graphs: two operators: projection $\pi$ and (for efficiency reasons) combined projection and selection $[\pi\sigma]$
- operating on the resulting tables: usual set of algebraic operators known from relational databases, i.e., $\pi$, $\sigma$, $\times$, $\Join$, $\cup$, $\cap$, $\setminus$, ...
Def. 7  The projection \( \pi \) on an RDF graph \( G \) for a property tuple \((p_1, \ldots, p_n)\) is defined as follows:

\[
\pi_{(p_1, \ldots, p_n)}(G) = \bigcup \pi_{(p_1, \ldots, p_n)}(t)
\]

for all \( t = T_G(s) \) with \( s \in \Sigma_G \) and \( \{p_1, \ldots, p_n\} \subseteq s \)

where \( \pi \) is the normal database projection operator slightly modified to work on non-normalized tables (since the entries are set-valued).

- essential part: subsumption restriction
  \( \{p_1, \ldots, p_n\} \subseteq s \)

- i.e., consider the signature tables with signatures that are subsumed by the properties occurring in the operator
**Def. 8** The projection-selection $[\pi \sigma]$ on an RDF graph $G$ for a property tuple $p = (p_1, \ldots, p_n)$ and a condition $C$ is defined as follows:

$$[\pi \sigma]^C_p(G) = \pi_p \bigcup (\sigma_C \circ \pi_{p'})(t)$$

for all $t = T_G(s)$ with $s \in \Sigma_G$ and $p' \subseteq s$

and $p' = \{p_1, \ldots, p_n\} \cup \text{properties}(C)$

where $\pi$ and $\sigma$ are the normal database selection operators (modified as $\pi$ above), and $\text{properties}(C)$ is the set of properties that occur in $C$. 
Sample queries

**Exa. 1** ‘Return first name and last name for all persons.’

\[
  r = \left[ \pi \sigma \right]_{\text{rdf:type=Person}} (\text{firstName,lastName}) (P)
\]

**Exa. 2** ‘Find first name, email address, and homepage for the person with last name “Smith”’:

\[
  r = \left[ \pi \sigma \right]_{\text{rdf:type=Person} \land \text{lastName=”Smith”}} (\text{firstName,Email,homepage}) (P)
\]

*Without complex selection conditions:*

\[
  r = \pi_{\{1,3,4\}} (\sigma_{2=”Smith”} (\left[ \pi \sigma \right]_{\text{rdf:type=Person}} (\text{firstName,lastName,Email,homepage}) (P)))
\]
Implementation
- prototype: in-memory variant, JDK 1.5 plus Sesame’s RIO parser
- concepts have directly corresponding implementations plus appropriate index structures (for each column in a signature table, currently realized as a hash table)
- implementation of \( \pi \) and \([\pi\sigma]\) require efficient lookup of the signature tables
- merging of signature tables to reduce number of tables
- natural querying
- RDF Schema
Signature lookup for properties $p = \{p_1, \ldots, p_n\}$:

(a) add $\emptyset$ as an artificial root to the simplified signature subsumption graph $G'_G$

(b) starting at $\emptyset$, find all minimal signatures $s$ which are subsumed by $p$, i.e., for which $p \subseteq s$ holds

(c) add all signatures which are subsumed by these minimal signatures (simply by collecting all signatures reachable from the minimal ones using a depth-first walk)
merge small adjacent signatures tables (i.e., which share many properties) or small tables with subsumed big ones
- greedy algorithm, tries to minimize the number of NULL values to be added and the number of merge operations
- resulting signature subsumption graphs very similar to user defined schemas (co-occurrence of properties is the basis for (manually) defining classes in an schema/ontology)
- approach expected to perform similar to mapping an RDF Schema to an (object-) relational database directly
Exa. 3  *Using Java as query language.*

```java
p.projectAndSelect(
    p.properties(FIRSTNAME, LASTNAME, EMAIL, HOMEPAGE),
    p.C_equals(p.RDF_TYPE, PERSON))
.select(p.C_equals(1, p.literal("Smith")))
.project(0,2,3);
```

- higher-level query languages like SPARQL have the disadvantage compared to using Java that they do not nicely cooperate with Java data
- e.g., query parameters have to be translated into textual representations matching the query language syntax
- simpler debugging, manual optimizations possible
- but: declarative query languages are easier to read, automatic optimizations
Def. 9 The (simplified) RDFS immediate consequence operator $T_{\text{RDFS}}$ for an RDF graph $G$ is defined as follows:

$$T_{\text{RDFS}}(G) = G \cup \{\langle p, s\text{PO}, q \rangle \mid \{\langle p, s\text{PO}, r \rangle, \langle r, s\text{PO}, q \rangle \} \subseteq G\}$$

$$\cup \{\langle p, s\text{CO}, q \rangle \mid \{\langle p, s\text{CO}, r \rangle, \langle r, s\text{CO}, q \rangle \} \subseteq G\}$$

$$\cup \{\langle s, \text{type}, c \rangle \mid \{\langle c', s\text{CO}, c \rangle, \langle s, \text{type}, c' \rangle \} \subseteq G\}$$

$$\cup \{\langle s, p, o \rangle \mid \{\langle p', s\text{PO}, p \rangle, \langle s, p', o \rangle \} \subseteq G\}$$

Thm. 1 The (simplified) RDF Schema semantics of an RDF graph $G$ is the fixpoint of $T_{\text{RDFS}}$:

$$G^{\text{RDFS}} = \bigcup_{n \in \mathbb{N}_0} T^n_{\text{RDFS}}(G \cup AP)$$

- semi-naive bottom-up evaluation of $T_{\text{RDFS}}$ to materialize $G^{\text{RDFS}}$
- materialization of the propagation for rdf:type is not necessary and can be handled by query rewriting; drastically reduces the number of additionally created data
Evaluation
we evaluated distribution of signature tables and load times, memory consumption, and query execution times

comparison with other freely available RDF stores (Sesame and Jena)

data sets from TAP (http://sp11.stanford.edu/), mainly swirl-SiteMoviesIMDB.rdf (298 MB and 3,587,064 triples)
### Signature Table Distribution

<table>
<thead>
<tr>
<th>tuples per table</th>
<th>tuples</th>
<th>tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–10</td>
<td>8941</td>
<td>4137</td>
</tr>
<tr>
<td>11–100</td>
<td>15375</td>
<td>506</td>
</tr>
<tr>
<td>101–1000</td>
<td>13730</td>
<td>62</td>
</tr>
<tr>
<td>1001–10000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10001–100000</td>
<td>117288</td>
<td>2</td>
</tr>
<tr>
<td>100001–1000000</td>
<td>130939</td>
<td>1</td>
</tr>
</tbody>
</table>

- **swirl-SiteMoviesIMDB.rdf**: 298 MB, 3,587,064 triples, 286,273 resources in subject position, 4,708 signature tables
- three tables hold 87% of all tuples
- many small tables, merging will improve performance
### Query Times

<table>
<thead>
<tr>
<th></th>
<th>Query 1</th>
<th></th>
<th>Query 2</th>
<th></th>
<th>Query 3</th>
<th></th>
<th>Query 4</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>mem</td>
<td>time</td>
<td>mem</td>
<td>time</td>
<td>mem</td>
<td>time</td>
<td>mem</td>
</tr>
<tr>
<td>RDFBroker</td>
<td>70ms</td>
<td>4MB</td>
<td>1200ms</td>
<td>63MB</td>
<td>260ms</td>
<td>4MB</td>
<td>160ms</td>
<td>10MB</td>
</tr>
<tr>
<td>Jena</td>
<td>4300ms</td>
<td>82MB</td>
<td>8700ms</td>
<td>26MB</td>
<td>70ms</td>
<td>3MB</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sesame</td>
<td>1400ms</td>
<td>24MB</td>
<td>2200ms</td>
<td>46MB</td>
<td>50ms</td>
<td>2MB</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **Query 1:** ‘return some interesting properties of all movies’
  \[
  \pi \sigma_{\text{rdf:type=imdb:Movie}}(\text{rdf:about}, \text{rdfs:label}, \text{imdb:PropertyCountry}, \text{PropertySound}, \text{Mix})
  \]

- **Query 2:** ‘find names for persons casted in movies’
  \[
  \pi \sigma_{\text{rdf:type=imdb:Movie}}(\text{rdf:about}, \text{rdfs:label}, \text{imdb:creditedCast}) \land \text{\textbf{3} = \textbf{1}}
  \]
  \[
  \pi \sigma_{\text{rdf:type=imdb:Person}}(\text{rdf:about}, \text{rdfs:label})
  \]

- **Query 3:** ‘find persons playing in movies three cast hops separated from Kevin Bacon’

- **Query 4:** ‘find movies with same title and return some useful properties on them, like release year, cast, genre, …’
  \[
  \pi \sigma_{\text{rdf:type=imdb:Movie}}(\text{rdf:about}, \text{rdfs:label}, \text{imdb:creditedCast}, \ldots) \land \text{\textbf{2} = \textbf{1}, \textbf{1} \neq \textbf{1}}
  \]
  \[
  \pi \sigma_{\text{rdf:type=imdb:Movie}}(\text{rdf:about}, \text{rdfs:label}, \ldots)
  \]
Conclusions and Future Work
signatures as basis for storing and querying arbitrary RDF data
- with table merging, approximates user-defined schemas/ontologies
- is thus expected to perform similar to hand-coded (object-)relational databases
- approach profits from well-investigated deductive database technologies
- even for in-memory case often outperforms other RDF stores
- for mass data, we will make use of on-disk databases
Future Work

- intend to support query and also rule language standards like SPARQL and the result of the W3C Rule Interchange Format (RIF) Working Group
- planned: P2P network (or grid) to improve performance by using in-memory (instead of on-disk) stores in peers and using the signature subsumption graph for distributing the data, routing queries, and developing appropriate peer leave/join algorithms

- enabled by open source community for RDFBroker:
  http://rdfbroker.opendfki.de/