Dynamic Provenance for SPARQL Updates using Named Graphs

James Cheney and Harry Halpin

W3C/MIT (@harryhalpin harry@w3.org)

October 23, 2014
1. What’s the point?
While the (Semantic) Web currently does have a way to exhibit static provenance information in the W3C PROV standards, the Web does not have a way to describe dynamic changes to data.

Our hypothesis is that a simple vocabulary, composed of insert, delete, and copy operations as introduced by Buneman et al., along with explicit identifiers for update steps, versioning relationships, and metadata about updates provides a flexible format for dynamic provenance on the Semantic Web.

A primary advantage of our methodology is it keeps the changes to raw data separate from the changes in provenance metadata, so legacy applications will continue to work and the cost of storing and providing access to provenance can be isolated from that of the raw data.
Dynamic Provenance for SPARQL Updates using Named Graphs

James Cheney and Harry Halpin

What’s the point?

An Example

```
DELETE WHERE {
  g {?X s ?Y . ?Y u ?Z }
}
```

```
INSERT {
  g {?X u ?Y }
} WHERE {
  g {?X t ?Y}
}
```

```
DELETES WHERE {
  g {?X s ?Y . ?Y u ?Z }
}
```

```
INSERT {
  g {?X u ?Y }
} WHERE {
  g {?X t ?Y}
}
```

```
DELETES WHERE {
  g {?X s ?Y . ?Y u ?Z }
}
```

```
INSERT {
  g {?X u ?Y }
} WHERE {
  g {?X t ?Y}
}
```

```
DELETES WHERE {
  g {?X s ?Y . ?Y u ?Z }
}
```

```
INSERT {
  g {?X u ?Y }
} WHERE {
  g {?X t ?Y}
}
```

```
DELETES WHERE {
  g {?X s ?Y . ?Y u ?Z }
}
```

```
INSERT {
  g {?X u ?Y }
} WHERE {
  g {?X t ?Y}
}
```
Teminology

- Any graph that records the insertion and deletion of triples from a given graph is considered a *provenance graph* (prov) for the given graph, which keeps track of auxiliary named graphs.

- $C$ denotes basic graph (or dataset) patterns that may contain variables; $R$ denotes conditions; $P$ denotes patterns, and $Q$ denotes queries. A graph store $\mathcal{D} = (G, \{g_i \mapsto G_1 \ldots, g_n \mapsto G_n\})$ consists of a default graph $G_0$ together with a mapping from names $g_i$ to graphs $G_i$. 
For queries, we consider a simple form of provenance which calculates a set of named graphs “consulted” by the query.

\[
S[C]^G = \bigcup \{ \text{names}(\mu(C)) \mid \mu \in [C]^G \}
\]

\[
S[P_1 \cdot P_2]^G = S[P_1]^G \cup S[P_2]^G
\]

\[
S[P_1 \text{ UNION } P_2]^G = S[P_1]^G \cup S[P_2]^G
\]

\[
S[P_1 \text{ OPT } P_2]^G = S[P_1]^G \cup S[P_2]^G
\]

\[
\]

\[
S[\text{SELECT } \vec{X} \text{ WHERE } P]^G = S[P]^G
\]

\[
S[\text{CONSTRUCT } C \text{ WHERE } P]^G = S[P]^G
\]
A graph creation `CREATE g` is translated to

```
CREATE g;
CREATE g_v0;
INSERT DATA {GRAPH prov {
  ⟨g version g_v0⟩,⟨g current g_v0⟩,
  ⟨u_1 type create⟩,⟨u_1 output g_v0⟩,
  ⟨u_1 meta m_i⟩,(metadata)
}}
```

where `g_v_i` is the current version of `g`. Note that since this operation deletes `g`, after this step the URI `g` no longer names a graph in the store; it is possible to create a new graph named `g`, which will result in a new sequence of versions being created for it.
A clear graph operation CLEAR $g$ is handled as follows:

```
CLEAR $g$;
DELETE WHERE {GRAPH prov \{g current g_{v_i}\}};
INSERT DATA {GRAPH prov {
    \{g version g_{v_{i+1}}\}, \{g current g_{v_{i+1}}\},
    \{u_i type clear\}, \{u_i input g_{v_i}\},
    \{u_i output g_{v_{i+1}}\}, \{u_i meta m_i\},
    (metadata)
}}
```
A clear graph operation `CLEAR g` is handled as follows:

```
CLEAR g;
DELETE WHERE {GRAPH prov \{\langle g \ current \ g_{-v_i} \rangle\} \};
INSERT DATA {GRAPH prov \{
    \langle g \ version \ g_{-v_{i+1}} \rangle,\langle g \ current \ g_{-v_{i+1}} \rangle,
    \langle u_i \ type \ clear \rangle,\langle u_i \ input \ g_{-v_i} \rangle,
    \langle u_i \ output \ g_{-v_{i+1}} \rangle,\langle u_i \ meta \ m_i \rangle,
    (metadata)
}}
```
A load graph operation LOAD $h$ INTO $g$ is handled as follows:

\[
\begin{align*}
  &\text{LOAD } h \text{ INTO } g; \\
  &\text{DELETE WHERE } \{ \text{GRAPH } prov \ \{ \langle g \text{ current } g_{\cdot vi} \rangle \} \}; \\
  &\text{INSERT DATA } \{ \text{GRAPH } prov \ \\
  &\quad \{ \langle g \text{ version } g_{\cdot vi+1} \rangle, \langle g \text{ current } g_{\cdot vi+1} \rangle, \\
  &\quad \langle u_i \text{ type load} \rangle, \langle u_i \text{ input } g_{\cdot vi} \rangle, \\
  &\quad \langle u_i \text{ output } g_{\cdot vi+1} \rangle, \langle u_i \text{ source } h_j \rangle, \\
  &\quad \langle u_i \text{ meta } m_i \rangle, (\text{metadata}) \} \}
\end{align*}
\]

where $h_j$ is the current version of $h$. 
An insertion \texttt{INSERT \{GRAPH g \{C\}\} WHERE P} is translated to a sequence of updates that creates a new version and links it to URIs representing the update, as well as links to the source graphs identified by the query provenance semantics and a named graph containing the inserted triples:

\begin{verbatim}
CREATE g_u_i;
INSERT \{GRAPH g_u_i \{C\}\} WHERE P;
INSERT \{GRAPH g \{C\}\} WHERE P;
CREATE g_v_i+1;
LOAD g INTO g_v_i+1;
DELETE DATA \{GRAPH prov \{\langle g \text{ current} g_v_i\rangle\}\};
INSERT DATA \{GRAPH prov \{
    \langle g \text{ version} g_v_i+1\rangle, \langle g \text{ current} g_v_i+1\rangle,
    \langle u_i \text{ input} g_v_i\rangle, \langle u_i \text{ output} g_v_i+1\rangle,
    \langle u_i \text{ type insert} \rangle, \langle u_i \text{ data} g_u_i\rangle,
    \langle u_i \text{ source} s_1\rangle, ..., \langle u_i \text{ source} s_m\rangle,
\}
\};
\end{verbatim}
Delete Graph

A deletion DELETE {GRAPH g {C}} WHERE P is handled similarly to an insert, except for the update type annotation.

CREATE g_{ui};
INSERT {GRAPH g_{ui} {C}} WHERE P;
DELETE {GRAPH g {C}} WHERE P;
CREATE g_{vi+1};
LOAD g INTO g_{vi+1};
DELETE DATA {GRAPH prov {⟨g current g_{vi}⟩}};
INSERT DATA {GRAPH prov {
  ⟨g version g_{vi+1}⟩,⟨g current g_{vi+1}⟩,
  ⟨ui input g_{vi}⟩,⟨ui output g_{vi+1}⟩,
  ⟨ui type delete⟩,⟨ui data g_{ui}⟩
  ⟨ui source s_1⟩,...,⟨ui source s_m⟩,
  ⟨ui meta m_i⟩,(metadata)}}
## Dynamic Provenance Vocabulary

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>upd:insert</td>
<td>Triples inserted into a graph.</td>
</tr>
<tr>
<td>upd:delete</td>
<td>Triples deleted from a graph.</td>
</tr>
<tr>
<td>upd:load</td>
<td>Triples copied from another graph.</td>
</tr>
<tr>
<td>upd:clear</td>
<td>All triples in graph deleted</td>
</tr>
<tr>
<td>upd:create</td>
<td>New graph initialized.</td>
</tr>
<tr>
<td>upd:drop</td>
<td>A graph is deleted.</td>
</tr>
<tr>
<td>upd:input</td>
<td>Linked to graph used as input in update operation.</td>
</tr>
<tr>
<td>upd:output</td>
<td>Link to graph of output of the operation.</td>
</tr>
<tr>
<td>upd:data</td>
<td>Changed data in insert/delete operation.</td>
</tr>
<tr>
<td>upd:version</td>
<td>Link between two versions of a graph.</td>
</tr>
<tr>
<td>upd:type</td>
<td>Type of update operation.</td>
</tr>
<tr>
<td>upd:current</td>
<td>Link to most current state of graph.</td>
</tr>
<tr>
<td>upd:source</td>
<td>Any other graph that was consulted by the update.</td>
</tr>
<tr>
<td>upd:meta</td>
<td>Link to any metadata about the graph.</td>
</tr>
</tbody>
</table>

Lightweight Dynamic Provenance Vocabulary
Provenance is a challenging problem for RDF. While some progress has been made on provenance and annotation for RDFS inferences and SPARQL queries, so far there has not been work on provenance for SPARQL Update. We have outlined an approach to the problem drawing on similar work in database archiving and copy-paste provenance in relational databases. In particular, the metadata carried by our technique can use the PROV data model already developed by the W3C Provenance Interchange Working Group.