Dynamic provenance for SPARQL Updates using Named Graphs

Harry Halpin (W3C)
James Cheney* (UoE)

* supported by Royal Society University Research Fellowship
"Linked Data"
Buzzword detox

- **Linked Data** = exporting RDF using stable ids
  - = the part of the “semantic web” that works
- **RDF** = `<subject, predicate, object>` triples (graphs)
- **Named graphs** = RDF graphs with a (URI) name
  - duh
- **SPARQL** = query language for RDF
  - think conjunctive queries with a few wrinkles
- **SPARQL Updates** = update language for RDF
  - Work in progress (part of SPARQL 1.1)
Dynamic provenance

• By this, I basically just mean version history
  – this would cover large fraction of things people seem to want right off the bat
• Plus minimal ability to track sources of data copied from other places
  – i.e., copy-paste provenance for RDF
• Use of URIs makes this possible.
• This paper: translate updates to self-maintain provenance.
DELETE WHERE {
g {?x s ?y .
 ?y t ?z }
}

INSERT {
g {?x u ?y }
} WHERE {
g {?x t ?y}
}
Update language

\[
U ::= \text{INSERT } \{C\} \text{ WHERE } P \\
| \text{DELETE } \{C\} \text{ WHERE } P \\
| \text{LOAD } g \text{ INTO } g' \quad | \text{CLEAR } g \\
| \text{CREATE } g \quad | \text{DROP } g
\]

where \(C\) is a SPARQL graph expression (possibly with variables) and \(P\) is a SPARQL pattern that queries a graph and binds the variables to URIs or literals.
Graph creation (CREATE  g):

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\_1.  ... (other metadata)
  }
}
Graph deletion (DROP g):

DROP g;
DELETE WHERE {
    GRAPH prov { g current g_{vi} }
};
INSERT DATA {GRAPH prov {
    u_i type drop.  u_i input g_{vi}.
    u_i meta m_i.  ... (other metadata)
}}
Graph load (LOAD  g  INTO  g')

LOAD h INTO g;
DELETE WHERE {GRAPH prov {
  g current g_{vi}
}
};
INSERT DATA {GRAPH prov {
  g version g_{vi+1}.  g current g_{vi+1}.  
  u_i type load.  u_i input g_{vi}.  
  u_i output g_{vi+1}.  u_i source h_j.  
  u_i meta m_i.  ... (other metadata) 
}
}
**Insertion (INSERT \{C\} WHERE P)**

CREATE g_{ui};
INSERT \{GRAPH g_{ui} \{C\}\} WHERE P;
INSERT \{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}.
  u_i input g_{vi}. u_i output g_{vi+1}.
  u_i type insert. u_i data g_{ui}.
  u_i source S_1. ... u_i source S_m.
  u_i meta m_i. ... (other metadata)
\}
\}
Also in paper

• Provenance querying via SPARQL
  – implicitly
  – SPARQL is not really enough: no recursion.
  – Not my problem.

• Strawman for provenance retrieval over HTTP
Next steps

• Implementation 😊
  – should be straightforward to implement slow version

• Mapping dynamic provenance from other data models / DBMSs
  – copy-paste DBs/DBWiki
  – export from Oracle with metadata??

• Reconcile with OPM / W3C PIL?