PROVENANCE MANAGEMENT FOR EVOLVING RDF DATASETS

Argyro Avgoustaki[1,2], Giorgos Flouris[1],
Irini Fundulaki[1] and Dimitris Plexousakis[1,2]

[1] Institute of Computer Science-FORTH, Greece
[2] Computer Science Department, University of Crete, Greece

FORTH
Institute of Computer Science

Linked Data

- publishing and interlinking **structured data** on the Web using Semantic Web technologies
- sharing structured data on the Web as easily as documents are shared today
- using the Web to create **typed links** between data from different sources

Semantic Web

a space where people and organizations can post and consume data about anything

© W3C

Provenance for Linked Data

- Great volume of Linked Data
- Open and unconstrained nature of data
- Different data sources
- Heterogeneity
- Varying quality

W3C Provenance Incubator Group

With the arrival of massive amounts of Semantic Web data information about the origin of that data, i.e., provenance, becomes an important factor in developing new Semantic Web applications.

RDF, SPARQL and SPARQL Update

- **RDF**
  - W3C standard for representing information in the Web
  - An **RDF triple** (s, p, o) asserts the fact that subject **(s)** is associated with object (**o)** through predicate (**p**)
  - A set of RDF triples is an **RDF graph**
  - An **RDF named graph** (n,G) is an RDF graph G with a name n
  - A triple along with a named graph forms an **RDF quadruple**
- **SPARQL/ SPARQL Update**
  - querying and updating RDF data
  - graph pattern matching languages
  - a **quad pattern** denotes the quadruples in an RDF dataset that are of a specific form
  - quad patterns can be combined using SPARQL operators (Union, Join) to form **graph patterns**

SPARQL INSERT Updates

INSERT { qp_{ins} } WHERE { gp }

where:

- qp_{ins} is a quad pattern
- gp is of the form gp^1 **UNION** gp^2 … **UNION** gp^k
- gp^i is of the form $qp^i_1 . qp^i_2 . \ldots . qp^i_m$
- i: the order of a graph pattern in the WHERE clause
- j: the order of a quad pattern in gp^i
- $qp^i_j.pos$, $qp_{ins}.pos$, where pos ϵ s,p,o, are **quad pattern position identifiers**

Challenges for capturing SPARQL Update Provenance

- Named graph component is user defined
- Triples with different origin may be added to the same graph
- Provenance for quadruples instead of triples
- New quadruples may be the result of combining different quadruples and operators
- Not adequate information provided

Need for a new fine-grained abstract provenance model

Related Work

- In the SQL context:
  - Why and where: A characterization of data provenance [Buneman et al., 2006] presents why and where provenance models
  - Provenance semirings.[Green et al.,2007] introduces the how provenance model
  - Provenance management in curated databases [Buneman et al.,2006 considers where provenance model
  - On the Expressiveness of Implicit Provenance in Query and Update Languages. [Buneman et al., 2007] considers where provenance

Related Work

- In the RDF context:
  - Algebraic Structures for Capturing the Provenance of SPARQL Queries [Geerts et al., 2013] extends semirings
  - Provenance for Linked Data [Karvounarakis et al. 2013] considers why and how provenance
  - Coloring RDF triples to capture provenance [Flouris et al., 2009] supports where provenance
  - Dynamic provenance for SPARQL updates [Halpin et al., 2014] introduces a new model for capturing graph provenance

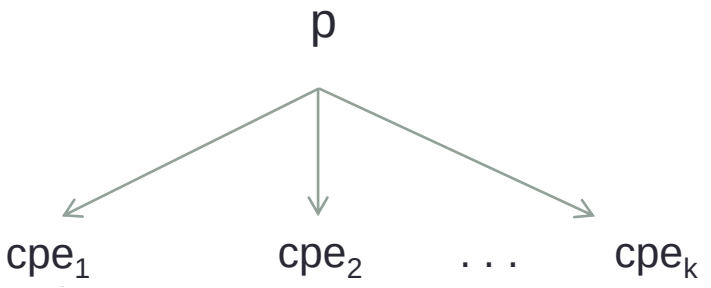
Our approach

- records attribute and triple level provenance
- based on **where** and **how (semirings)** provenance models
- comprised of **abstract quadruple identifiers (c_i), abstract operators (join $\odot$, union $\oplus$)** and **quad pattern position identifiers ($qp^i.pos$)**
- supports **reconstructability**

Reconstructability refers to the ability of using the provenance information for reconstructing an INSERT update that is compatible with the INSERT update that generated the result of the update.

Our approach

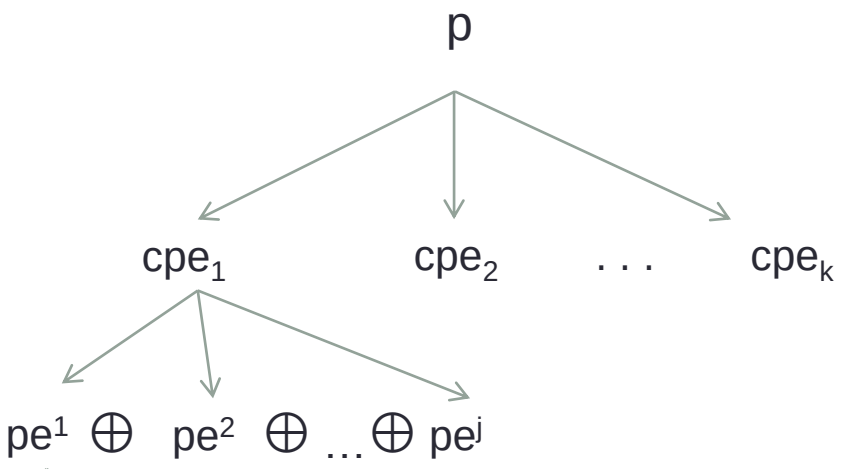
cpe represents each different way for a quadruple to be generated



Our approach

cpe represents each different way for a quadruple to be generated

pe corresponds to the provenance of one operand of a UNION operator

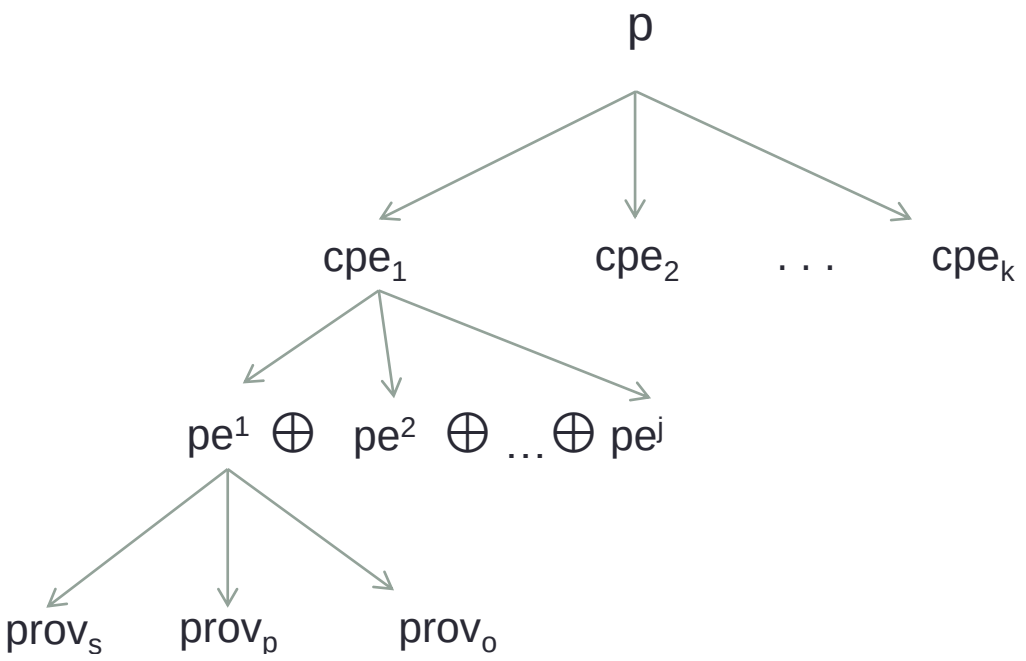


Our approach

cpe represents each different way for a quadruple to be generated

pe corresponds to the provenance of one operand of a UNION operator

$prov_{pos}$ allows the identification of the origin of each attribute



Our approach

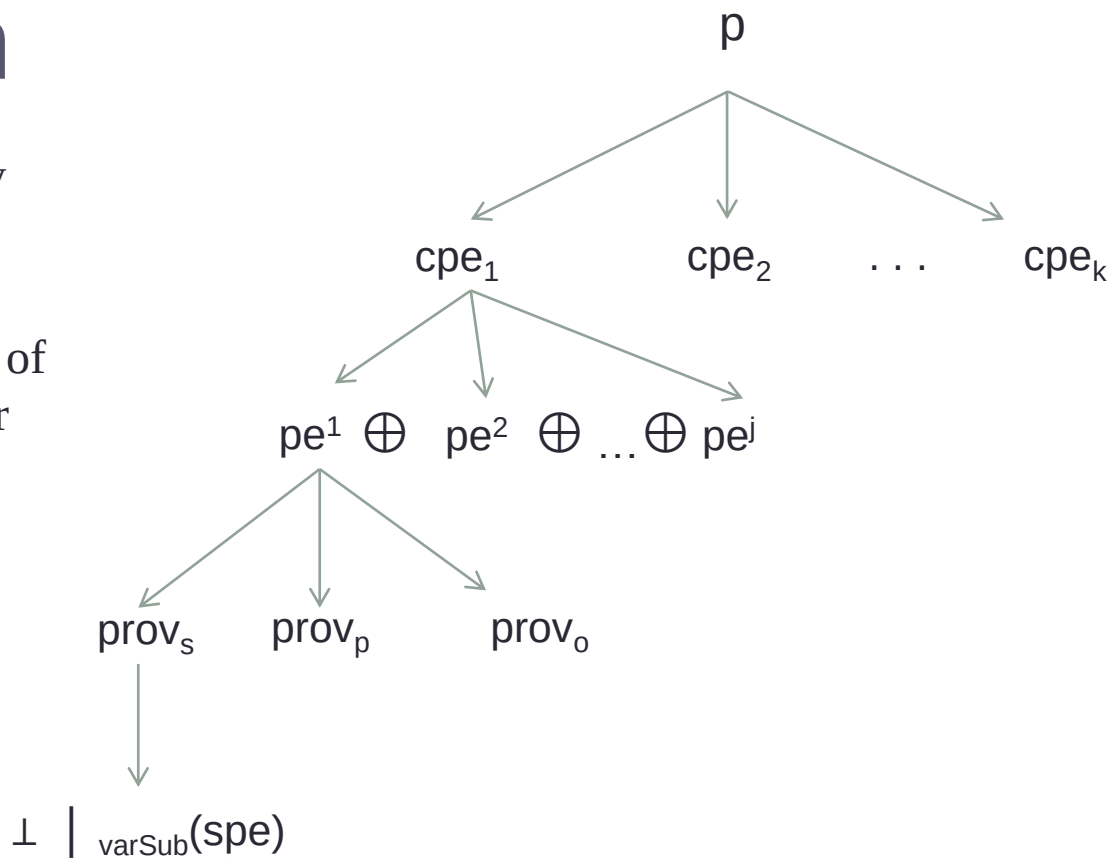
cpe represents each different way for a quadruple to be generated

pe corresponds to the provenance of one operand of a UNION operator

$prov_{pos}$ allows the identification of the origin of each attribute

⊥ is used for constants

$varSub(spe)$ records the provenance in "copy" and join cases



Our approach

cpe represents each different way for a quadruple to be generated

pe corresponds to the provenance of one operand of a UNION operator

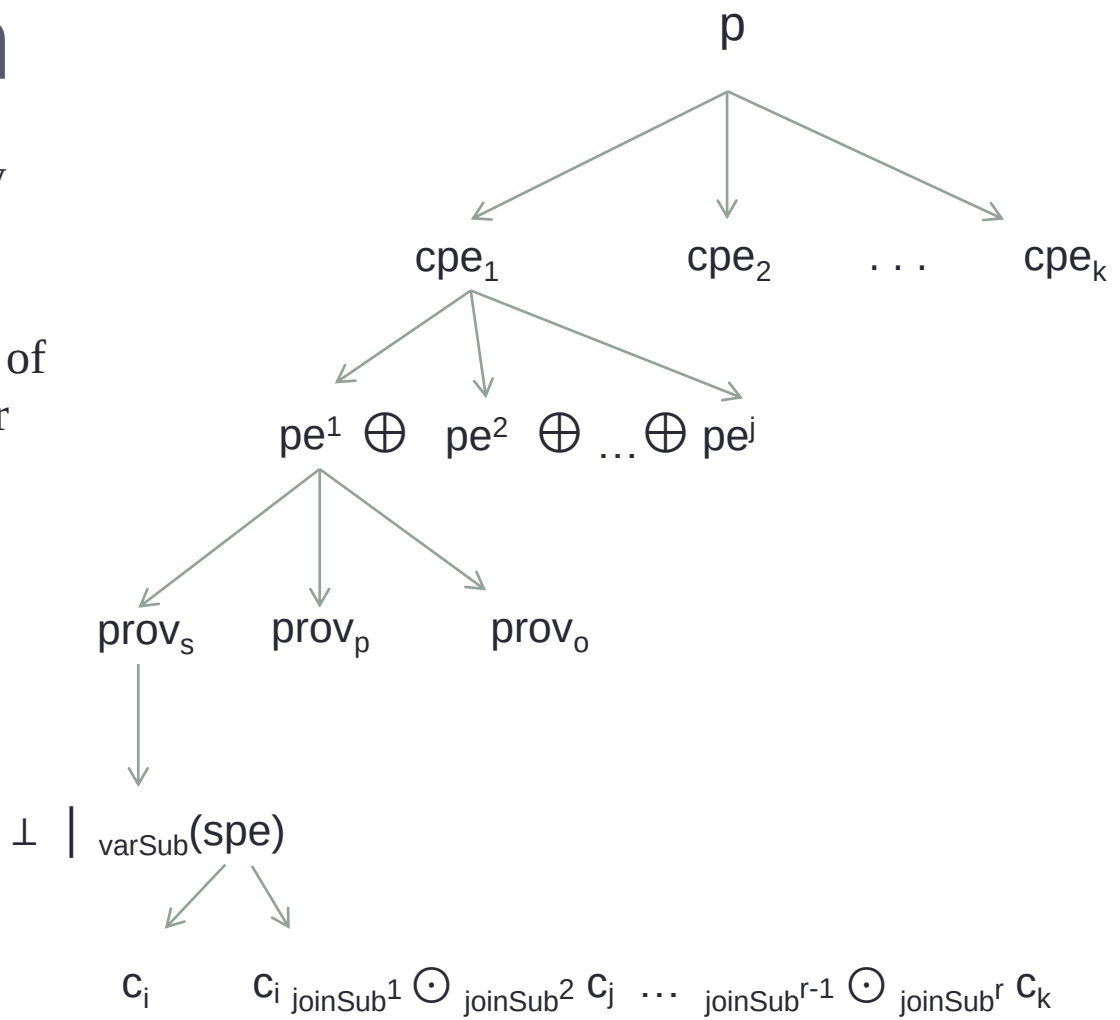
$prov_{pos}$ allows the identification of the origin of each attribute

⊥ is used for constants

$_{varSub}$(spe) records the provenance in "copy" and join cases

c_i is a quadruple identifier

joinSub^x is a set of join quad pattern positions



Our approach

cpe represents each different way for a quadruple to be generated

pe corresponds to the provenance of one operand of a UNION operator

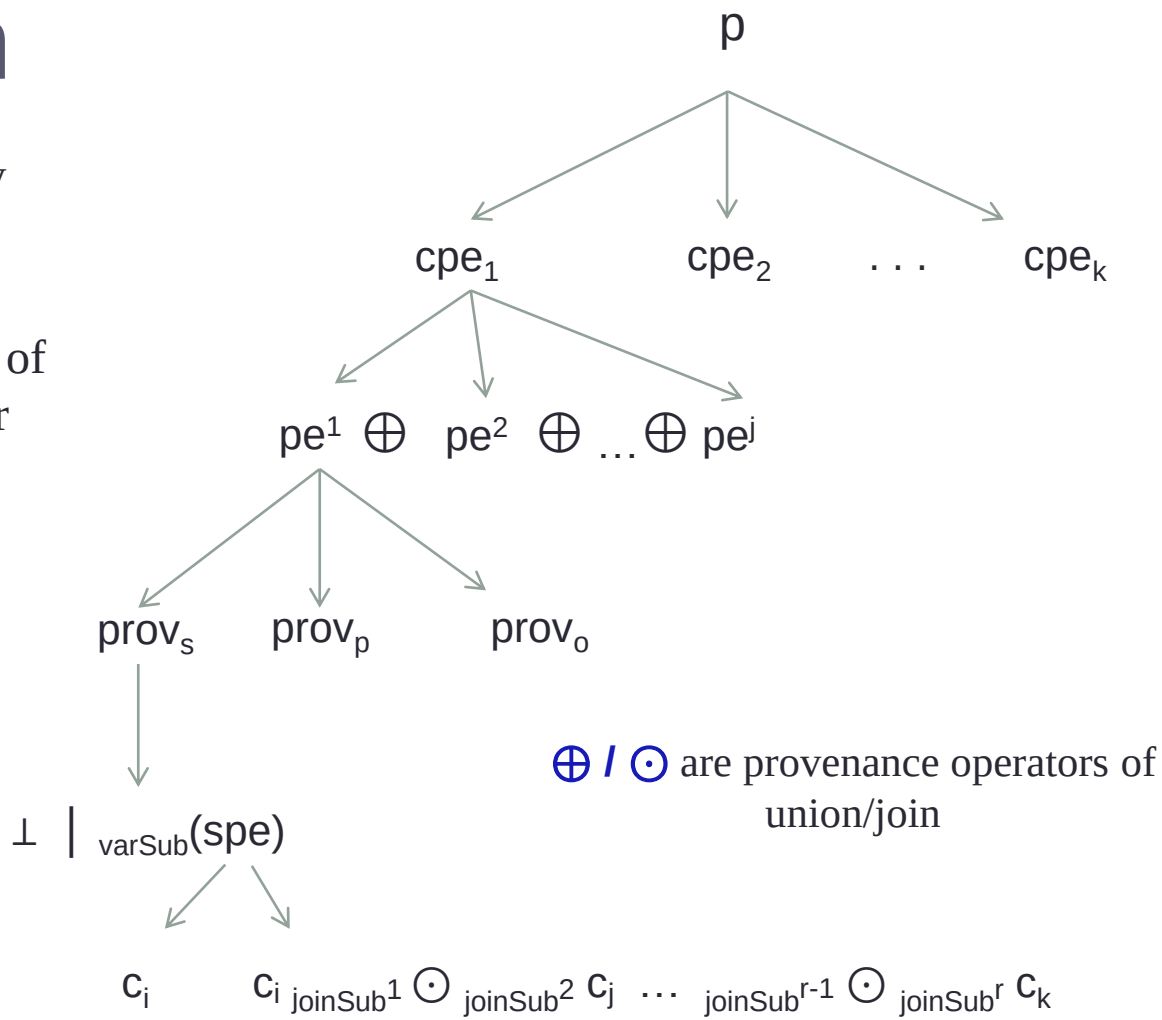
$prov_{pos}$ allows the identification of the origin of each attribute

$\perp$ is used for constants

$_{varSub}$(spe) records the provenance in "copy" and join cases

c_i is a quadruple identifier

joinSub^x is a set of join quad pattern positions



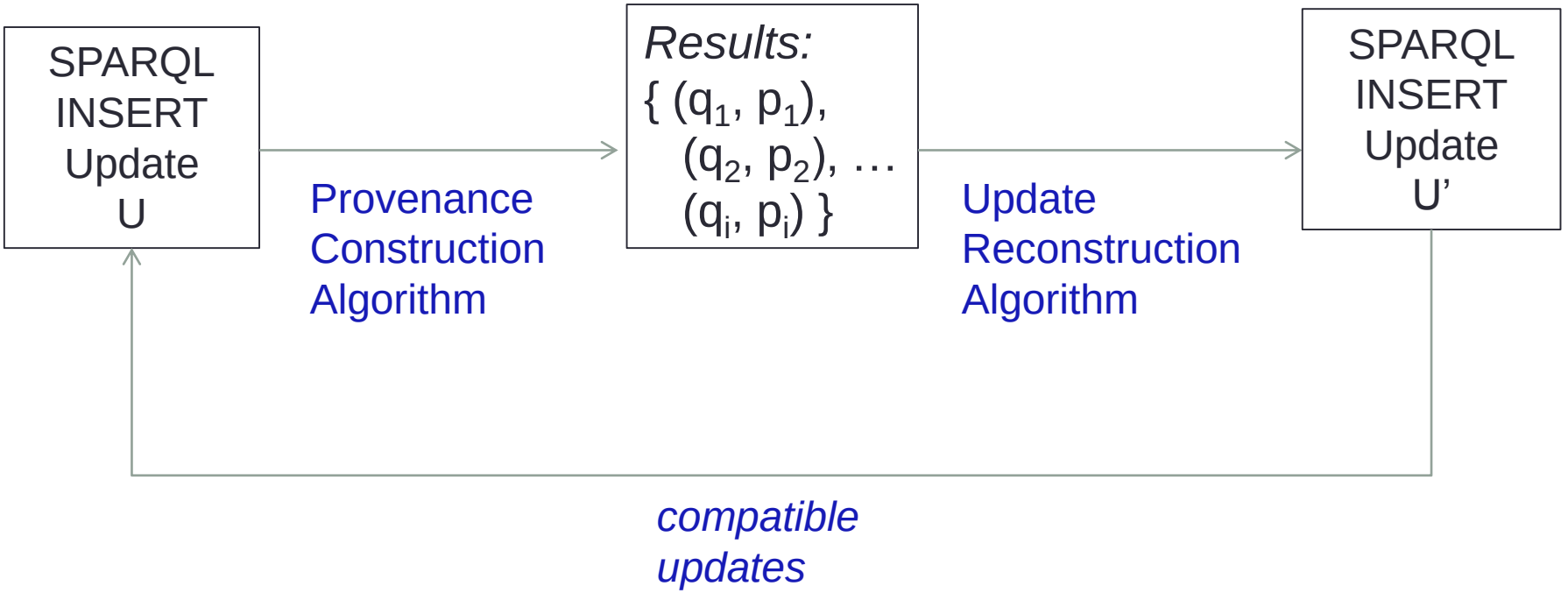
Our approach

INSERT
{?s a ?o <ex:g> }
WHERE
{?s a ?o <ex:g1> }

q_i: (<ex:dog> ,a, <ex:animal>)

$p_i : (qp^1_1 s(c_k) \ , \perp \ , qp^1_1 o(c_k))$

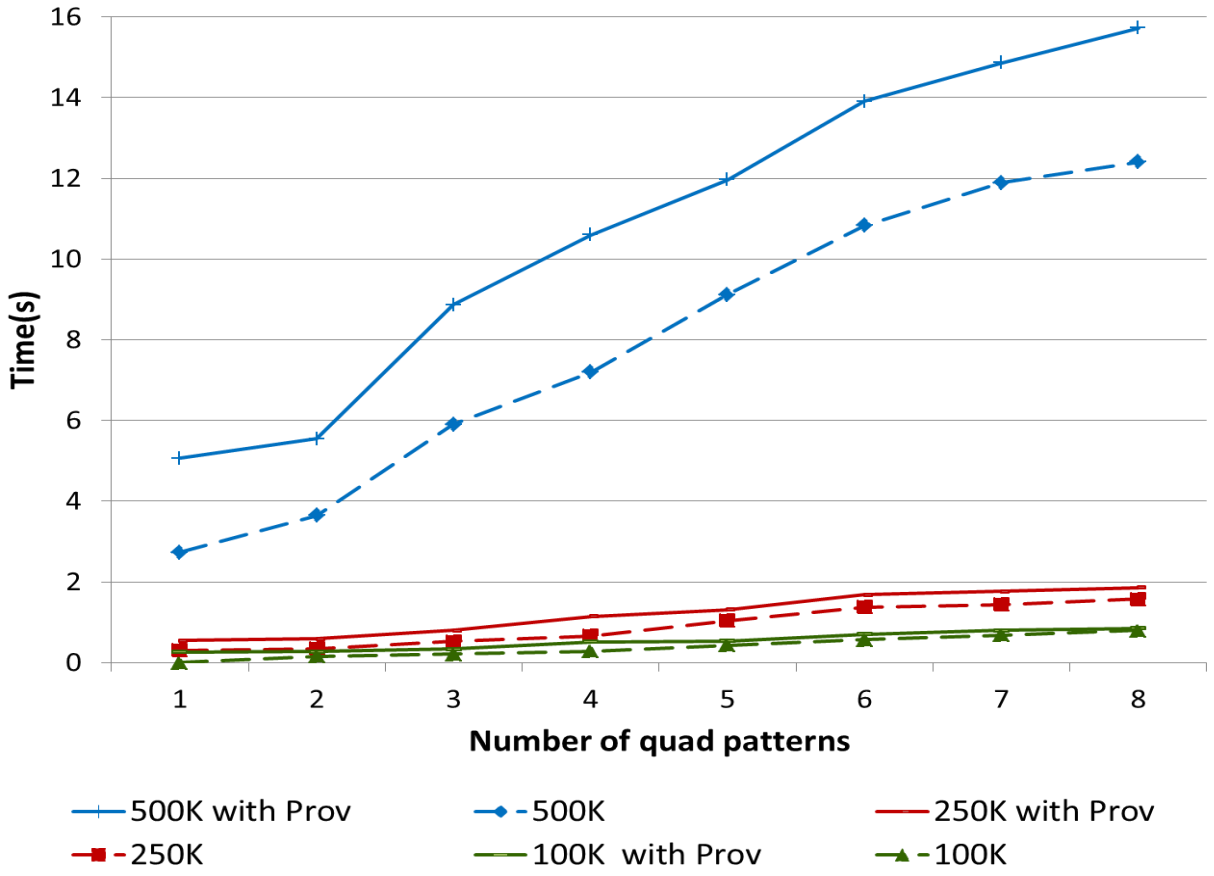
INSERT
{?v1 a ?v2 <ex:g> }
WHERE
{?v1 ?v3 ?v2 <ex:g1> }



Implementation and Evaluation

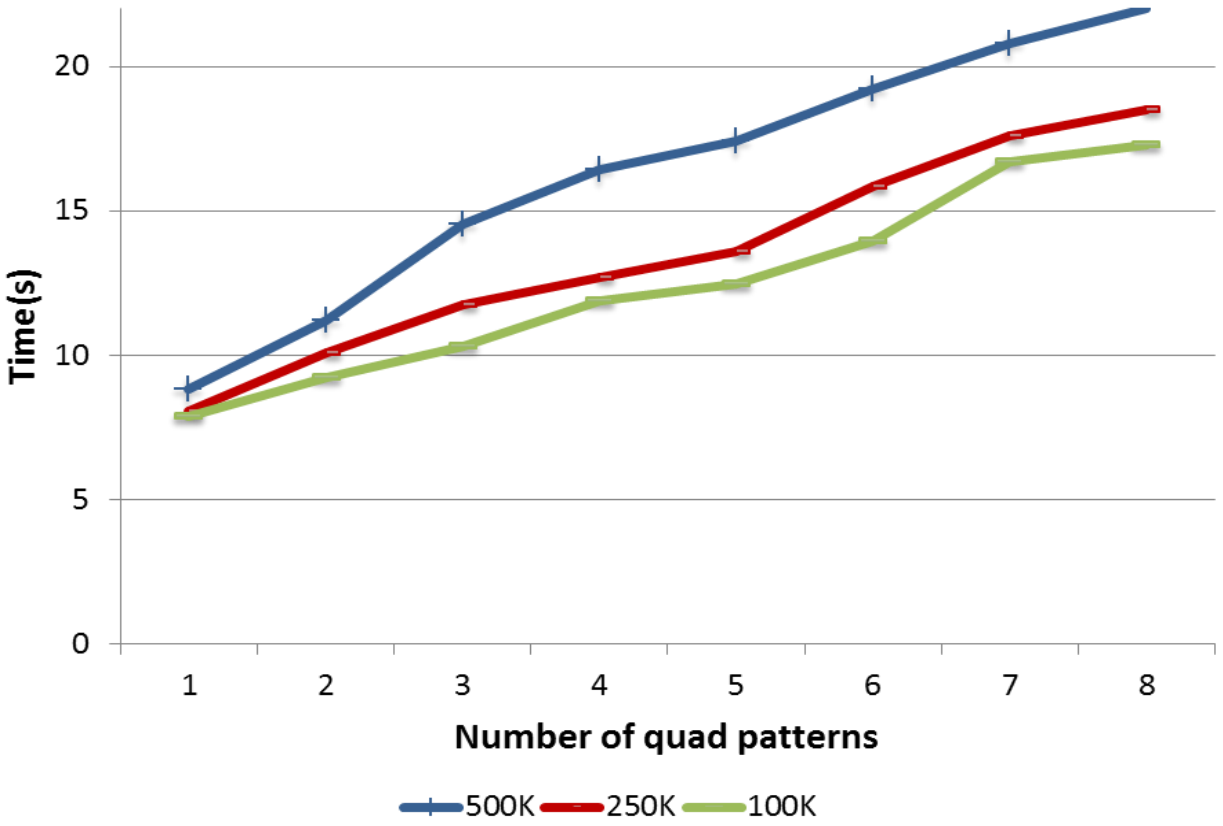
- Quadruples and provenance expressions are stored in a relational schema:
  - Quads(qid,s,p,o,n)
  - Prov(qid, cpeNo, peNo, $prov_s$, $prov_p$, $prov_o$)
- Virtuoso Database Engine as triple store
- Excerpts of BTC dataset containing 100K, 250K and 500K unique quadruples
- **Experiment 1** measures the time required to compute the results of an INSERT update along with their provenance
- **Experiment 2** considers the time required to compute only the result quadruples
- **Experiment 3** computes the time needed for reconstructing a compatible INSERT update based on a quadruple's provenance.

Implementation and Evaluation



Experiment 1,2

Implementation and Evaluation



Experiment 3

Conclusions

- A novel fine-grained provenance model for the quadruple results of SPARQL INSERT updates
  - ✓captures triple and attribute level provenance
  - ✓forms algebraic provenance expressions to annotate quadruples
  - ✓supports reconstructability
- Algorithmic support
  - ✓Provenance Construction Algorithm
  - ✓Update Reconstruction Algorithm
- Implemented and evaluated for a preliminary set of experiments

Future Work

- Extend our model to support *FILTER* and *OPTIONAL* operators and SPARQL functions
- Study the provenance of DELETE, CREATE and DROP updates
- Consider benchmarks supporting update operations
- Explore the use of PROV approach

Detailed presentation of our approach including source code can be found in http://www.ics.forth.gr/isl/provenance