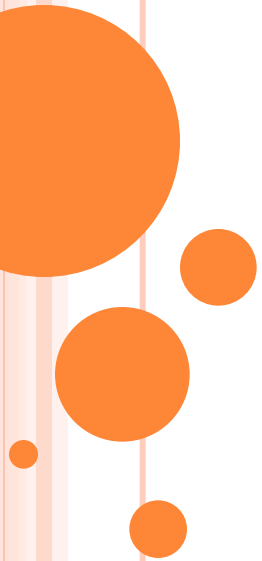
TripleProv: Efficient Processing of Lineage Queries in a Native RDF Store

Marcin Wylot, Philippe Cudre-Mauroux, Paul Groth



Presenter

Rudra Pratap Deb Nath

28th November 2014

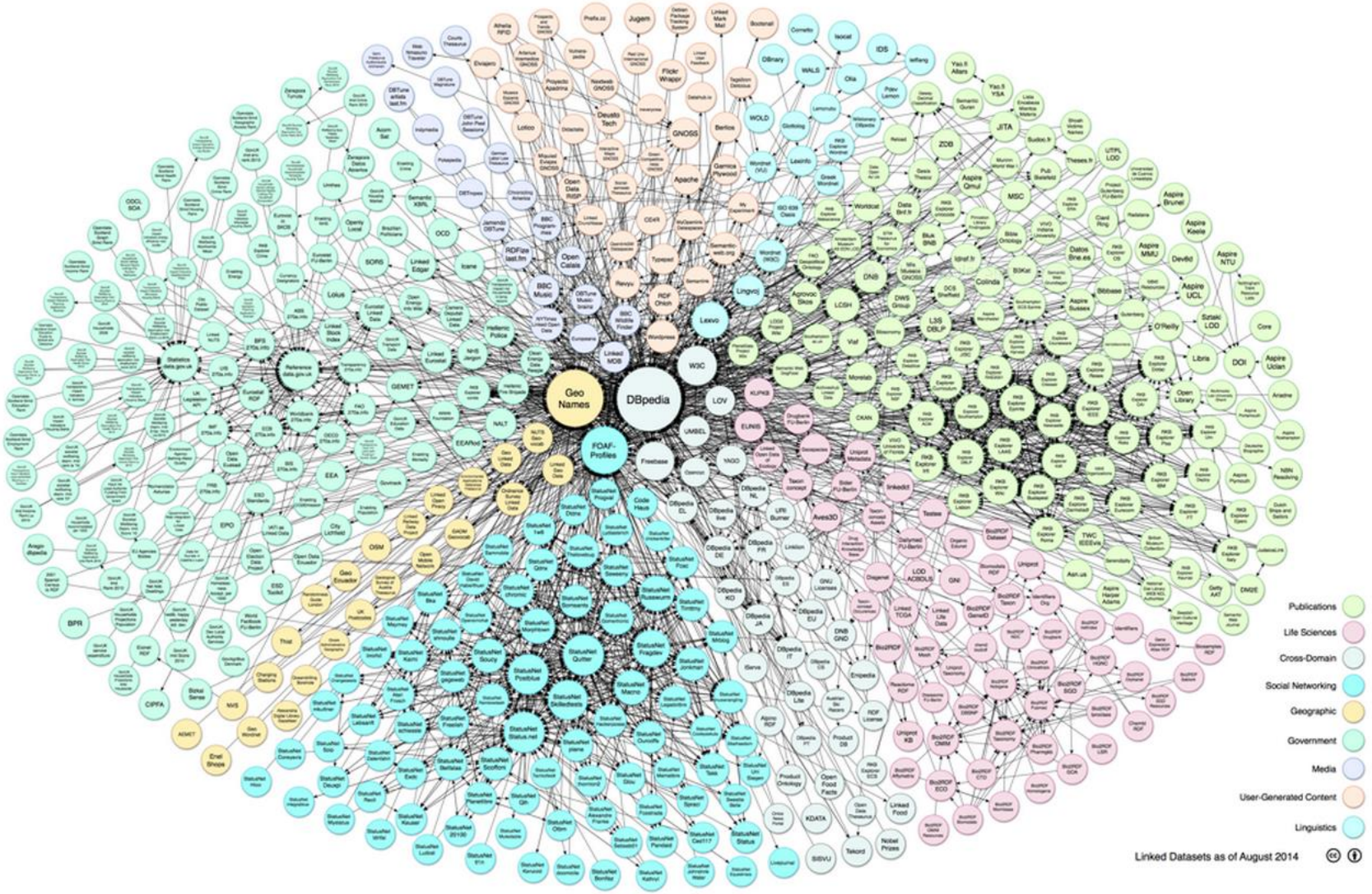
Presentation Outline

- Motivation
  - Querying Heterogeneous Data Sources
  - Provenance
  - Current State
  - Contribution
- Provenance Polynomials
- System Architecture
- Storage for Provenance
- Query Mechanism for Provenance
- Performance Evaluation
- Conclusion
- Future Direction

Presentation Outline

- Motivation
  - Querying Heterogeneous Data Sources
  - Provenance
  - Current State
  - Contribution
- Provenance Polynomials
- System Architecture
- Storage for Provenance
- Query Mechanism for Provenance
- Performance Evaluation
- Conclusion
- Future Direction

QUERYING HETEROGENEOUS DATA SOURCES



Linked Open Data Cloud

Querying heterogeneous data sources

- How is the result derived ?
- Are the sources trustable ?

Presentation Outline

- Motivation
  - Linked Open Data
  - Provenance
  - Current State
  - Contribution
- Provenance Polynomials
- System Architecture
- Storage for Provenance
- Query Mechanism for Provenance
- Performance Evaluation
- Conclusion
- Future Direction

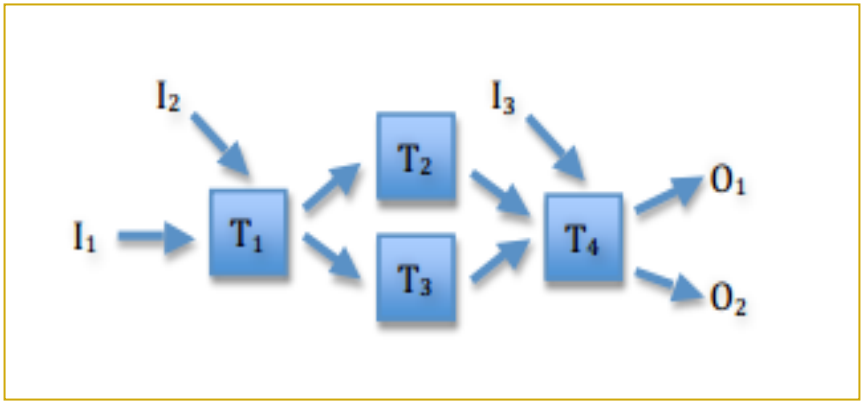
PROVENANCE

❖ Provenance describes from where and how the data is derived.

❖ To ensure data quality, security, trustworthyness, copyright issue, provenance plays a good role.

Two provenance

- ❖ *Where Provenance* : Which input sources are used to get the result ?
- ❖ *How Provenance* : How are the sources manipulated to produce the result ?



Provenance : Application

Post Query Application

- Result ranking
- Compute trust
- Information quality

Query Execution

- Changing query strategy on fly
- Restricting the result to certain subset of sources
- Access control
- Observe whether the result is valid after removing some sources

Presentation Outline

- Motivation
  - Querying Heterogeneous Data Sources
  - Provenance
  - Current State
  - Contribution
- Provenance Polynomials
- System Architecture
- Storage for Provenance
- Query Mechanism for Provenance
- Performance Evaluation
- Conclusion
- Future Direction

Current State

- W3C PROV recommendations for interchanging provenance
- Provenance is attached to RDF data using either RDF reification or named graphs
- It is difficult to track provenance using named graphs under updates and RDFS reasoning.
- Many of the tasks do not include how provenance
- No provenance polynomial for SPARQL query
- No current high-performance triple store (RDF database system) is able to automatically derive provenance data

Presentation Outline

- Motivation
  - Querying Heterogeneous Data Sources
  - Provenance
  - Current State
  - Contribution
- Provenance Polynomials
- System Architecture
- Storage for Provenance
- Query Mechanism for Provenance
- Performance Evaluation
- Conclusion
- Future Direction

Contribution

Unique Contributions

- Provenance of query result at different granularity levels
- New storage model to store provenance data in native RDF data storage model (dipLODocusrdf)
- Implementation of new system TripleProv and evaluation of the system

Presentation Outline

- Motivation
  - Querying Heterogeneous Data Sources
  - Provenance
  - Current State
  - Contribution
- Provenance Polynomials
- System Architecture
- Storage for Provenance
- Query Mechanism for Provenance
- Performance Evaluation
- Conclusion
- Future Direction

Requirement

- Capability to pinpoint the sources of the result. (*pProjection*)
- Capability to trace back the complete list of sources and how they are combined (*pConstraint*)

GRAPH-BASED QUERY

```
select ?lat ?long ?g1 ?g2 ?g3 ?g4
where {
    graph ?g1 {?a []     "Eiffel Tower" . }
    graph ?g2 {?a inCountry FR .          }
    graph ?g3 {?a lat       ?lat .        }
    graph ?g4 {?a long      ?long .       }
}
```

```
lat long 11 12 14 14,
lat long 11 12 14 15,
lat long 11 12 15 14,
lat long 11 12 15 15,
lat long 11 13 14 14,
lat long 11 13 14 15,
lat long 11 13 15 14,
lat long 11 13 15 15,
```

```
lat long 12 12 14 14,
lat long 12 12 14 15,
lat long 12 12 15 14,
lat long 12 12 15 15,
lat long 12 13 14 14,
lat long 12 13 14 15,
lat long 12 13 15 14,
lat long 12 13 15 15,
```

```
lat long 13 12 14 14,
lat long 13 12 14 15,
lat long 13 12 15 14,
lat long 13 12 15 15,
lat long 13 13 14 14,
lat long 13 13 14 15,
lat long 13 13 15 14,
lat long 13 13 15 15,
```

Provenance Polynomial

- Union ($\oplus$)
  - A constraint or projection satisfied with multiple sources

$$l1 \oplus l2 \oplus l3$$

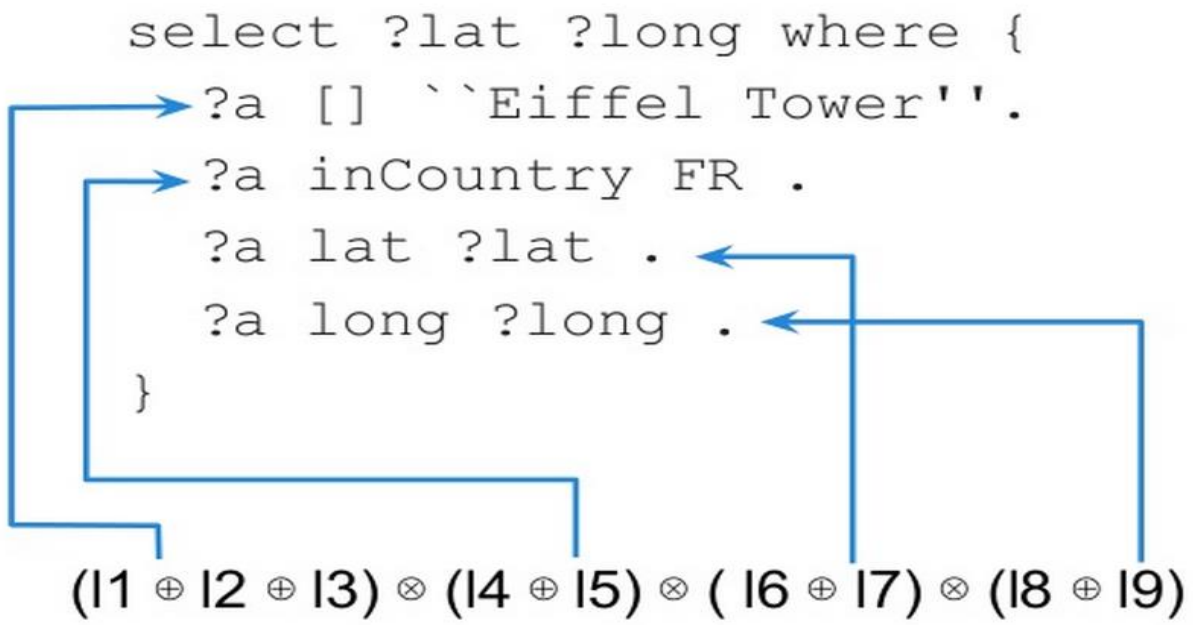
  - Multiple entities satisfy a set of constraints or projections
- Join ($\otimes$)
  - Sources were joined to satisfy a constraint or projection
  - OS and OO joins between a few sets of constraints.

$$(l1 \oplus l2) \otimes (l3 \oplus l4)$$

Granularity levels

- Source Level : Sources of triples
- Triple Level: Quadruples(triple + source) used to answer the query

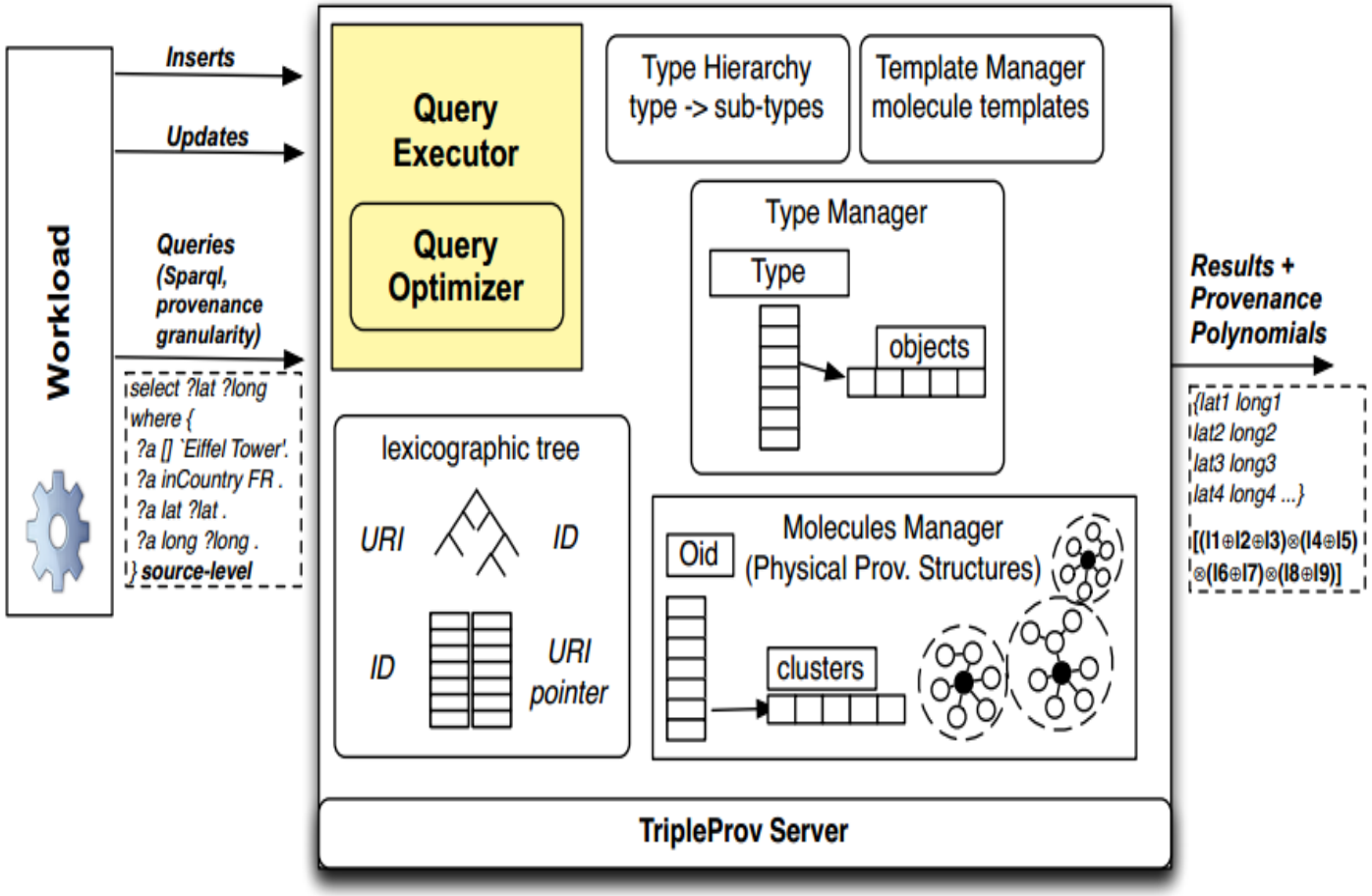
Evaluating the Groups



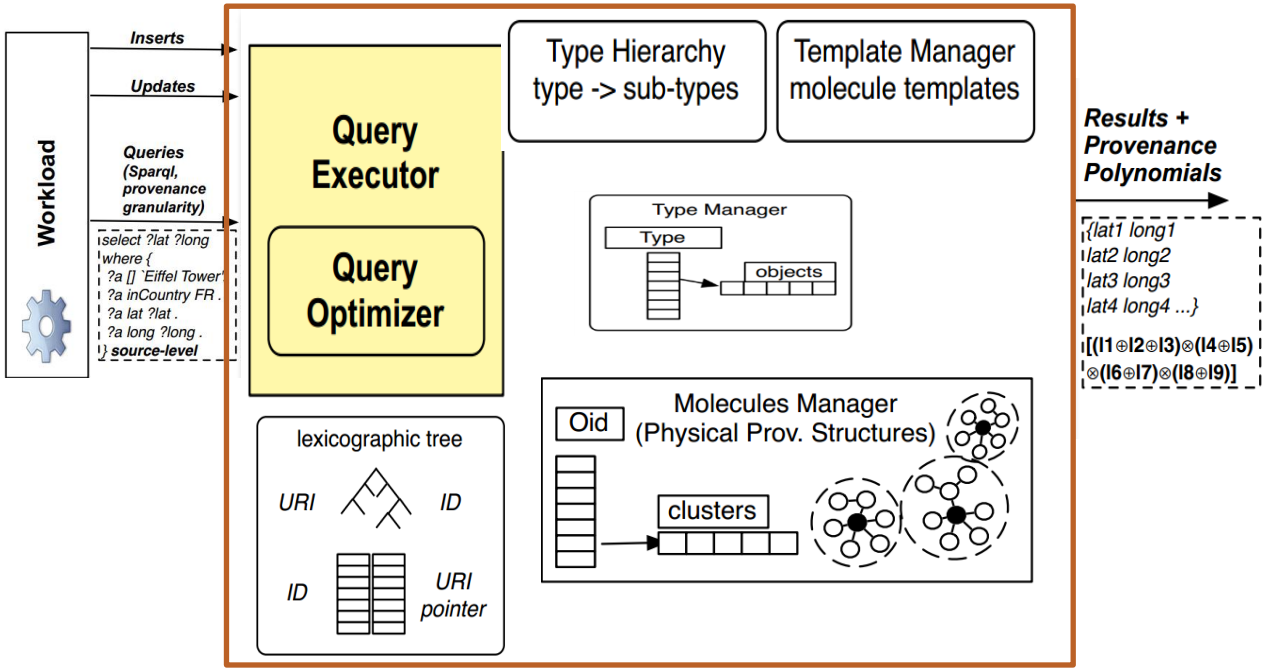
Presentation Outline

- Motivation
  - Querying Heterogeneous Data Sources
  - Provenance
  - Current State
  - Contribution
- Provenance Polynomials
- System Architecture
- Storage for Provenance
- Query Mechanism for Provenance
- Performance Evaluation
- Conclusion
- Future Direction

System Architecture

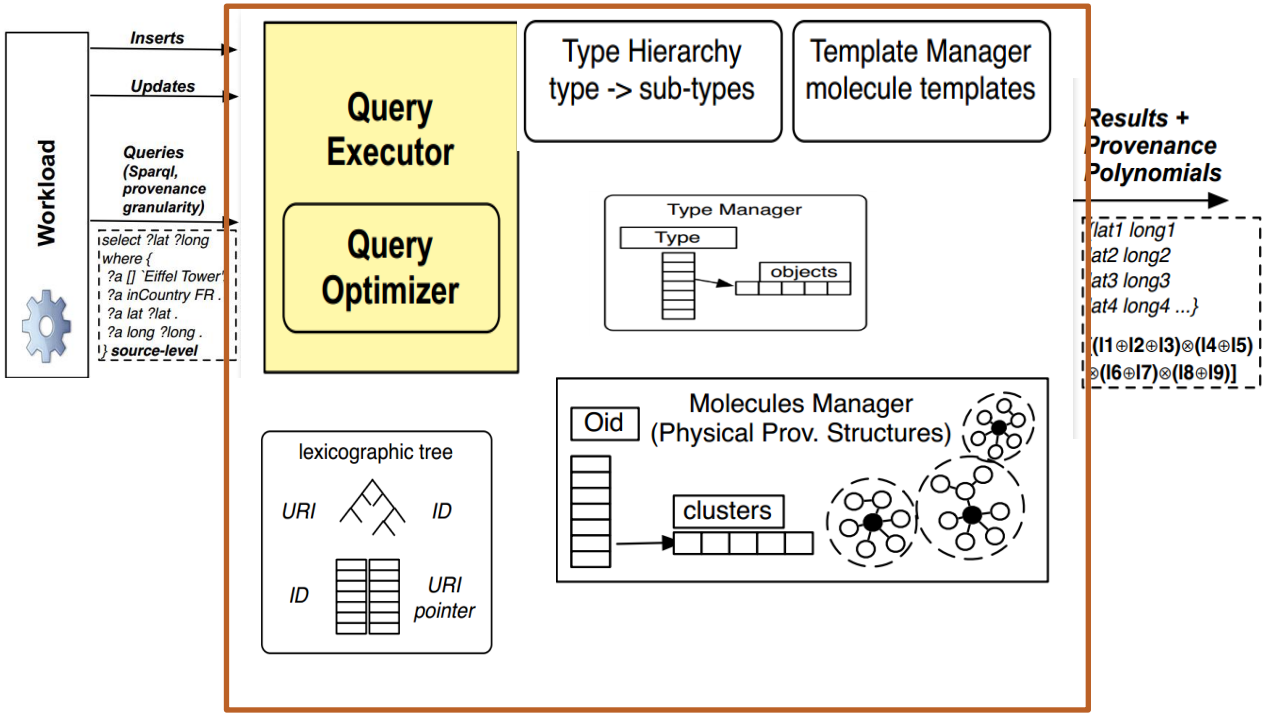


Query Executor



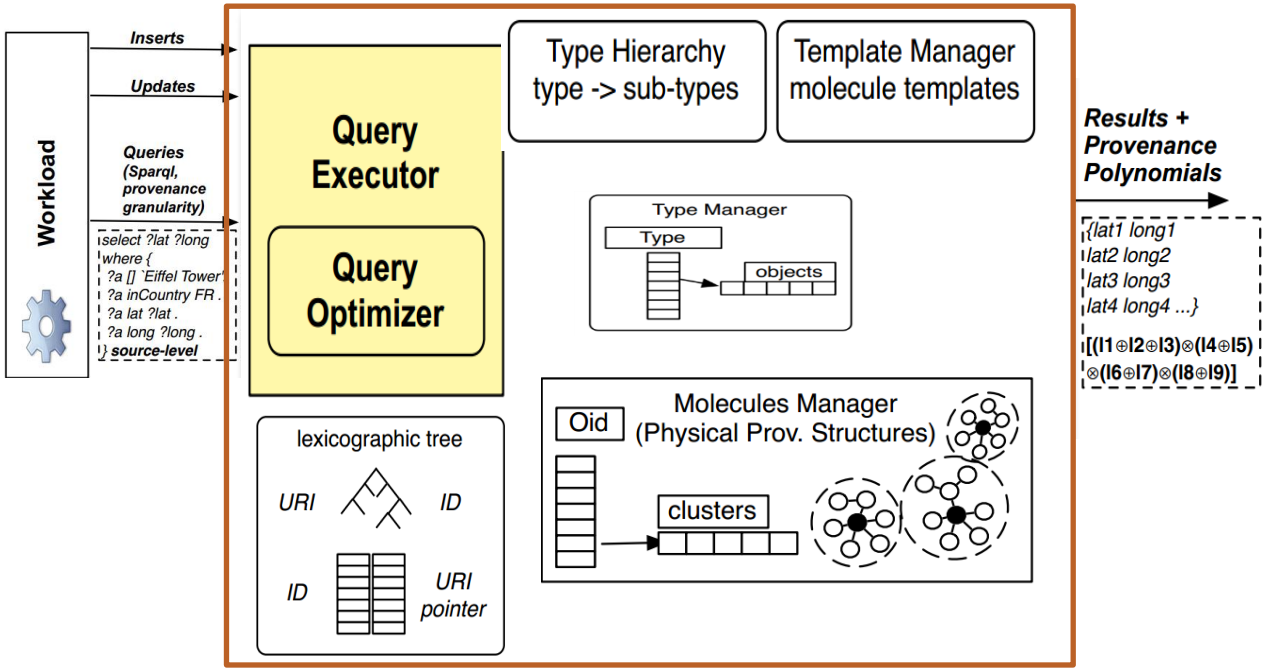
- Parse incoming query
- Rewrite the query plan
- Execute the query
- Return the result with provenance

KEY INDEX

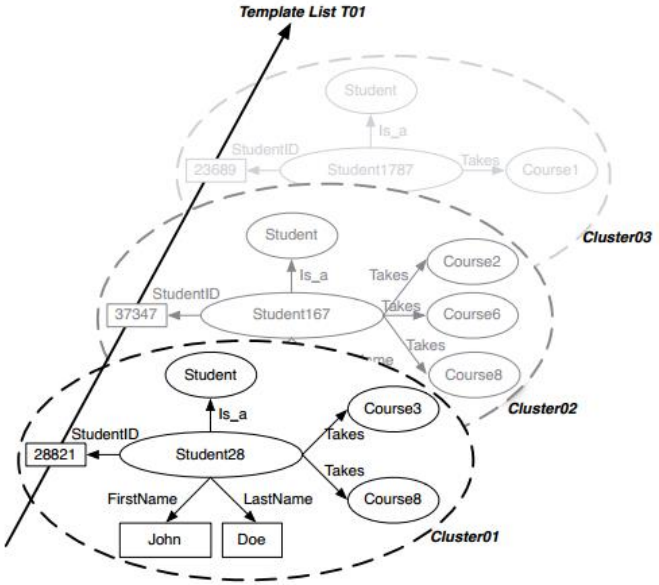


- Assign a unique key ID to each URI and literal
- Lexicographic tree is used to parse
  - Split based on common prifix
  - Every leaf represents a key ID composed of type prefix and auto-incrementat instance id.
- To relate instance ID to corresponding URI/literal, inverted index is used.

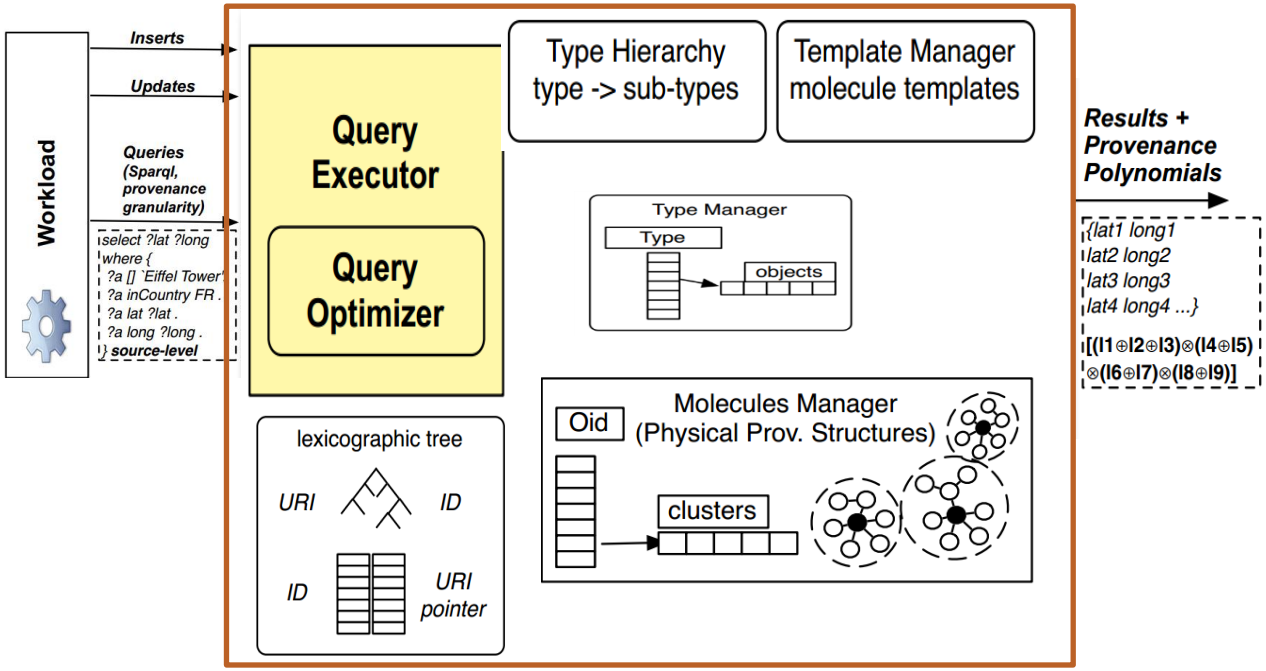
TYPE INDEX



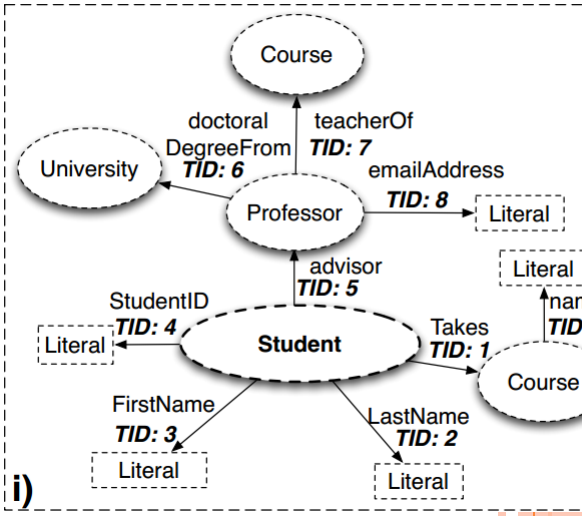
- Clustering all keys based on their types.
- Store in columnar fashion.
- Used for aggregrate and analytics queries.



TEMPLATE MANAGER



- Analyzing new triples, it builds molecule templates.
- Template acts as data prototype for RDF molecule.
- Using the molecule root, molecule identifiers are created.
- Subsumption relationships are stored in *type hierarchy*



RDF MOLECULE

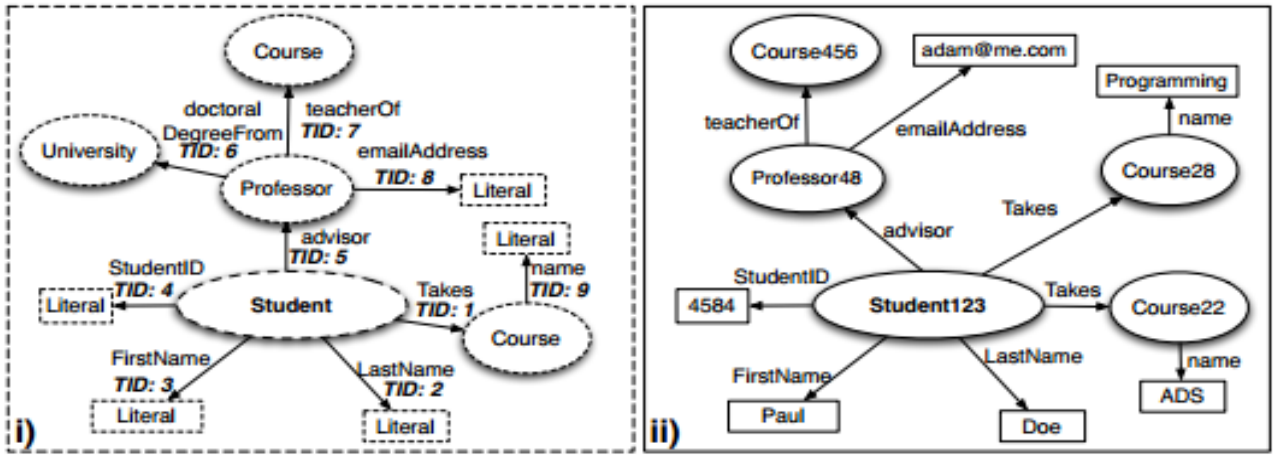
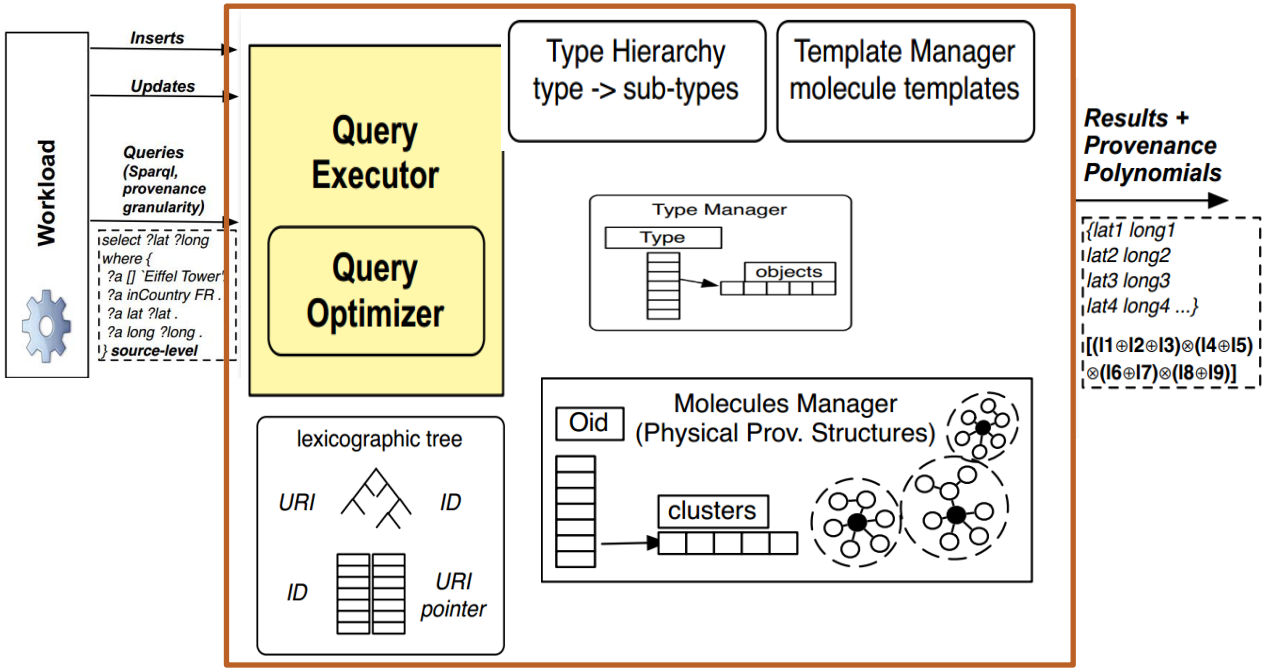


Figure 2: A molecule template (i) along with one of its RDF molecules (ii).

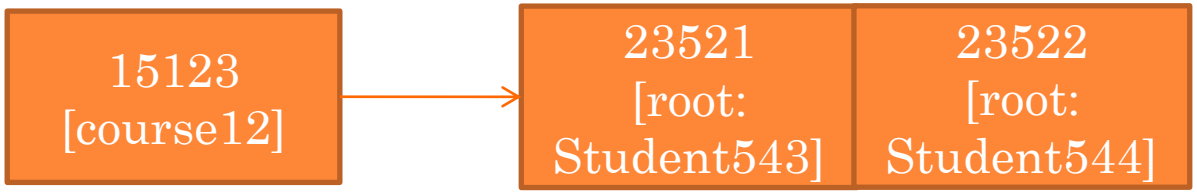
subj	obj1	obj2		objn

- RDF data is stored in TripleProv as RDF molecule.
- Hybrid structure of property table and RDF subgraphs.
- Based on the template definced by tamplete manager, a compact list of objects with root molecule is stored. Hence, TripleProv ensures storage compression.
- Logically groups set of related URIs and literals.
- Pysically co-locates information related to a given object into memory or disk.

Molecule Index



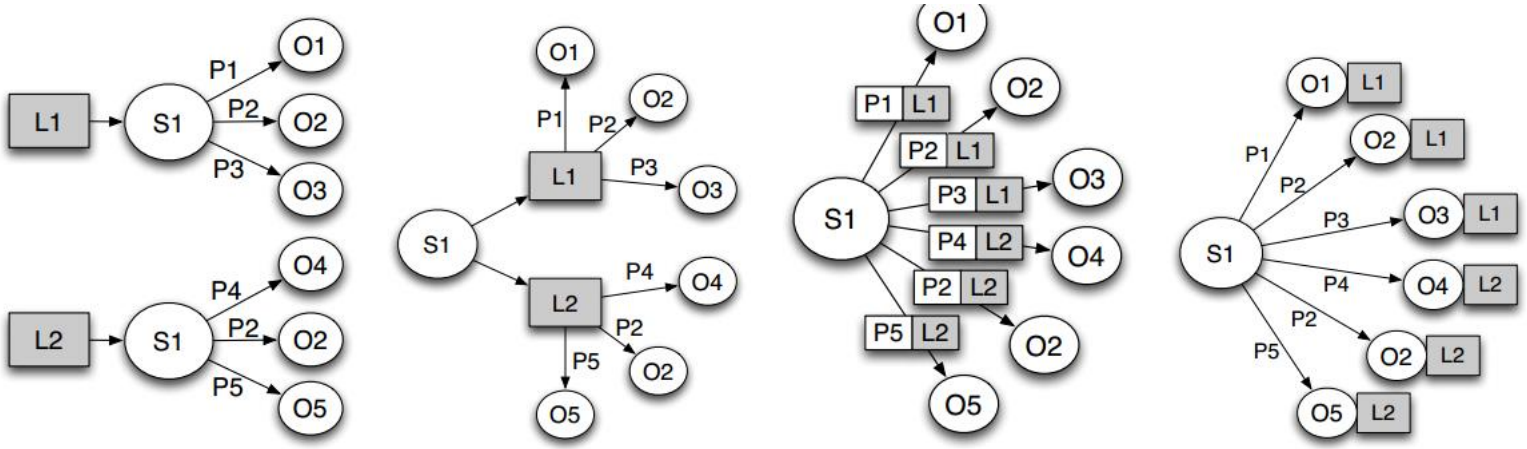
- For each key, store the list of local molecules containing that key
- For example



Presentation Outline

- Motivation
  - Querying Heterogeneous Data Sources
  - Provenance
  - Current State
  - Contribution
- Provenance Polynomials
- System Architecture
- Storage for Provenance
- Query Mechanism for Provenance
- Performance Evaluation
- Conclusion
- Future Direction

Different Types of Storage Model

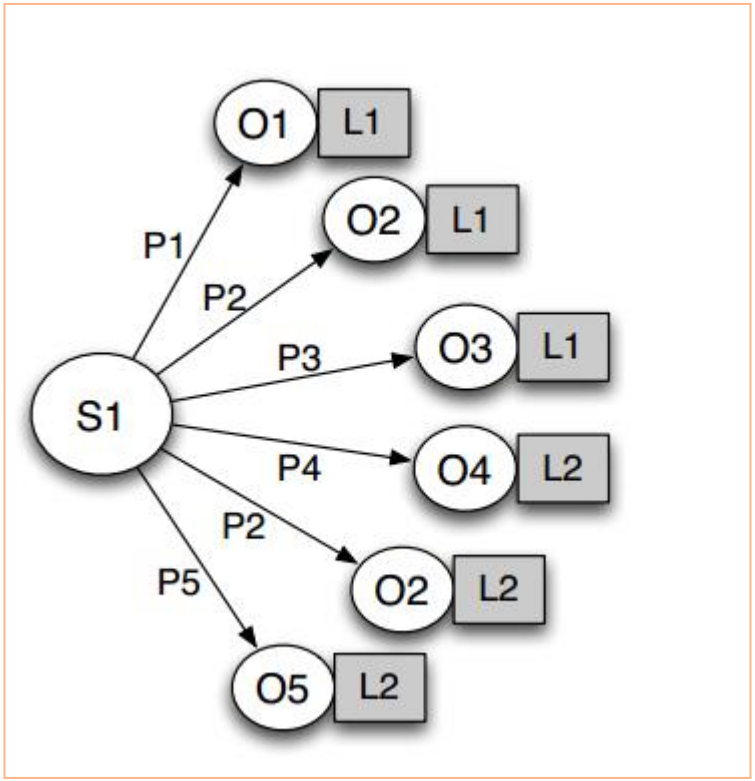


(a) LSPO Storage (b) SLPO Storage (c) SPLO Storage (d) SPOL Storage

- Ease of implementation
- Memory Consumption
- Violates the original molecule structure
- Query Execution

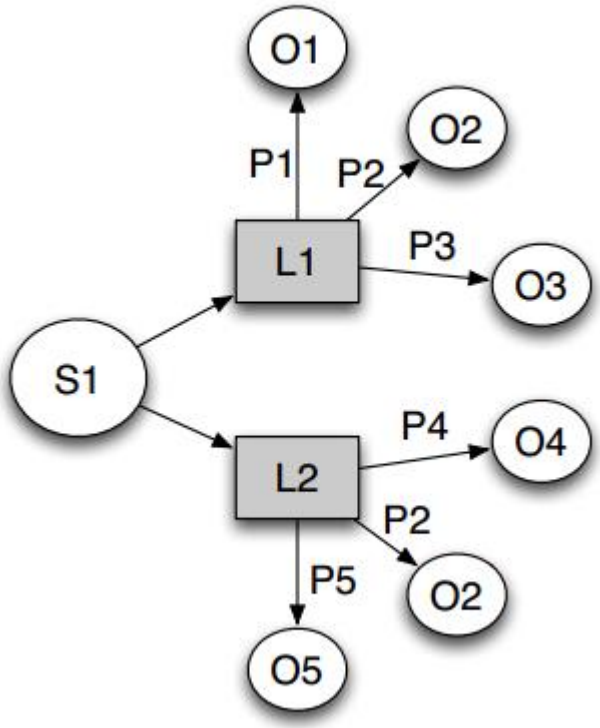
SPOL: Annotated Triples

- Annotated Provenance
- Easy to implement
- Quadraples
- Source data will be repeated for each triple



SLPO: Co-located storage

- Data is grouped by source
- Physically co-located
- Data related to a given subject located in one molecule
- No repetition of source in a molecule



Presentation Outline

- Motivation
  - Querying Heterogeneous Data Sources
  - Provenance
  - Current State
  - Contribution
- Provenance Polynomials
- System Architecture
- Storage for Provenance
- Query Mechanism for Provenance
- Performance Evaluation
- Conclusion
- Future Direction

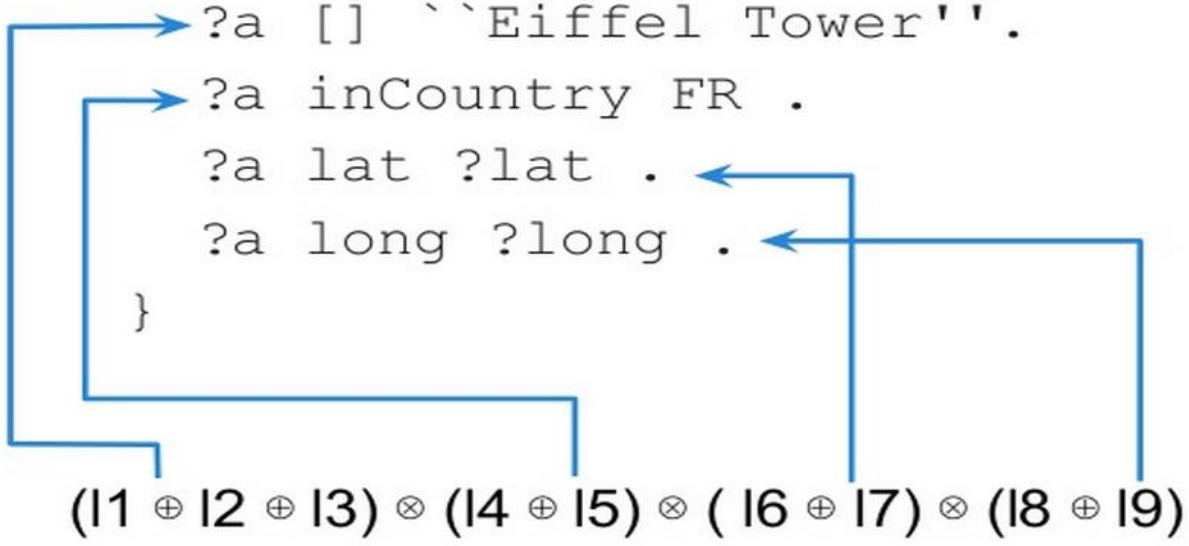
QUERYING

```
select ?lat ?long
where {
 ?a [] ''Eiffel Tower''. (<- 1st constraint)
 ?a inCountry FR .     (<- 2nd constraint)
 ?a lat ?lat .         (<- 1st projection)
 ?a long ?long .       (<- 2nd projection)
}
```

- From a given SPARQL query, a physical query plan is generated.
- The query executor chooses the most selective pattern to look-up molecules.
- Translate the bounded variable into key (for example 'Eiffel Tower') and retrieves all molecule containing that key.
- Each molecule is checked to determine whether
  - they met various constraints
  - the projections are correctly processed
- TripleProv keeps track of the provenance of each result.

QUERYING

```
select ?lat ?long where {
  ?a [] ``Eiffel Tower''.
  ?a inCountry FR .
  ?a lat ?lat .
  ?a long ?long .
}
```



Presentation Outline

- Motivation
  - Querying Heterogeneous Data Sources
  - Provenance
  - Current State
  - Contribution
- Provenance Polynomials
- System Architecture
- Storage for Provenance
- Query Mechanism for Provenance
- Performance Evaluation
- Conclusion
- Future Direction

Performance Evaluation

Hardware Platform

- HP ProLiant DL385 G7 server with ADM Opteron Processor (24 cores, 2 chips, 12 cores/chip)
- 64GB of DDR3 RAM

Datasets

- BTC: Billion Triple Challenge

  -Crawled from LOD

  -8 queries plus 2 queries based on UNION and OPTIONAL clause.

- WDC: Web Data Commons

  -Extracted from common Crawl Web corpus.

  -7 queries based on O-S joins, O-O joins, triangular joins and UNION and OPTIONAL clause.

- 115 million triples for each dataset.

Variant Considered

- V: Vanilla version (does not consider provenance)
- SG: Source-level granularity (provenance data grouped by source)
- SA: Source-level granularity (Annotated provenance data)
- TG: Triple-level granularity (provenance data grouped by source)
- TA: Triple-level granularity (Annotated provenance data)

Execution Time

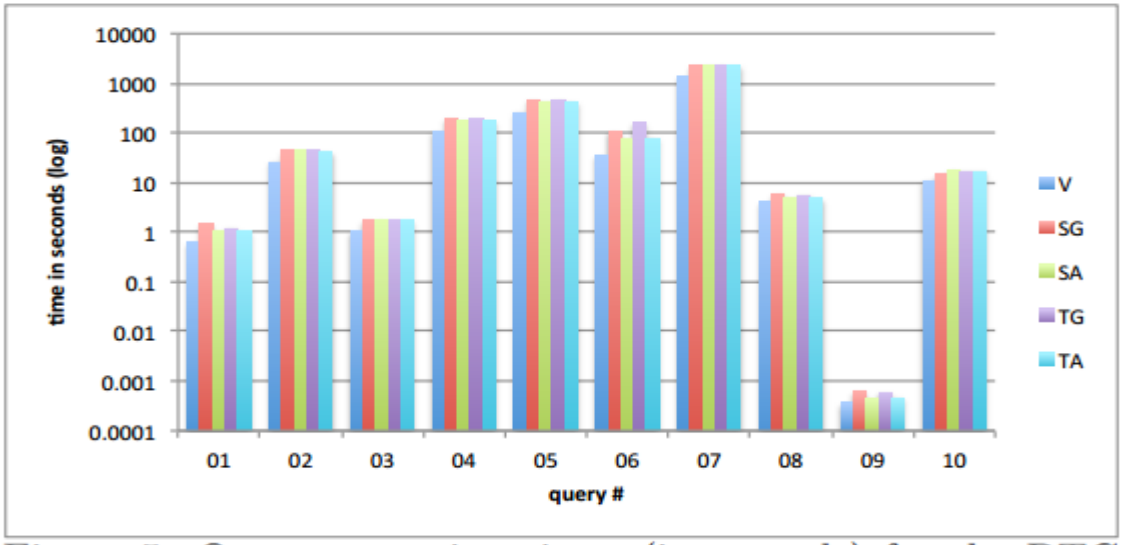


Figure 5: Query execution times (in seconds) for the BTC

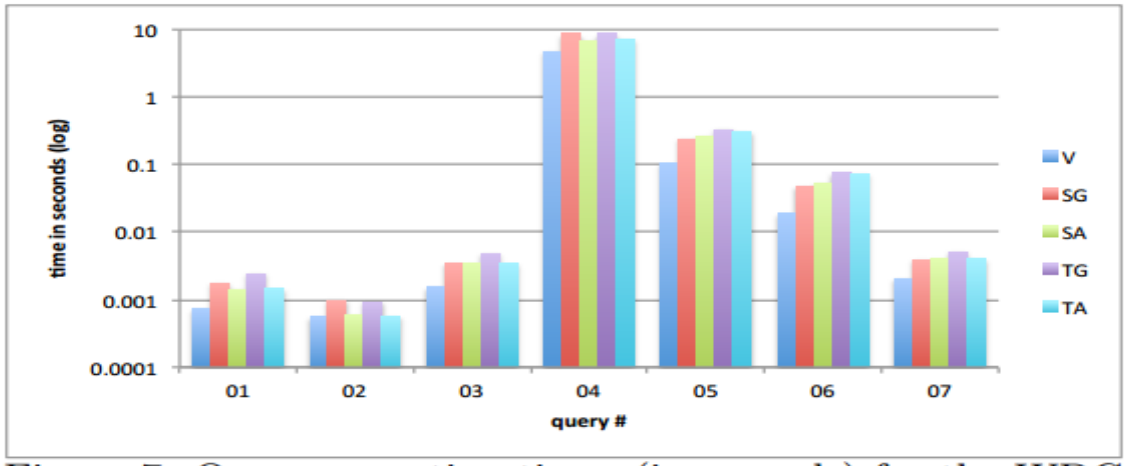


Figure 7: Query execution times (in seconds) for the WDC dataset (logarithmic scale)

Overhead for Provenance

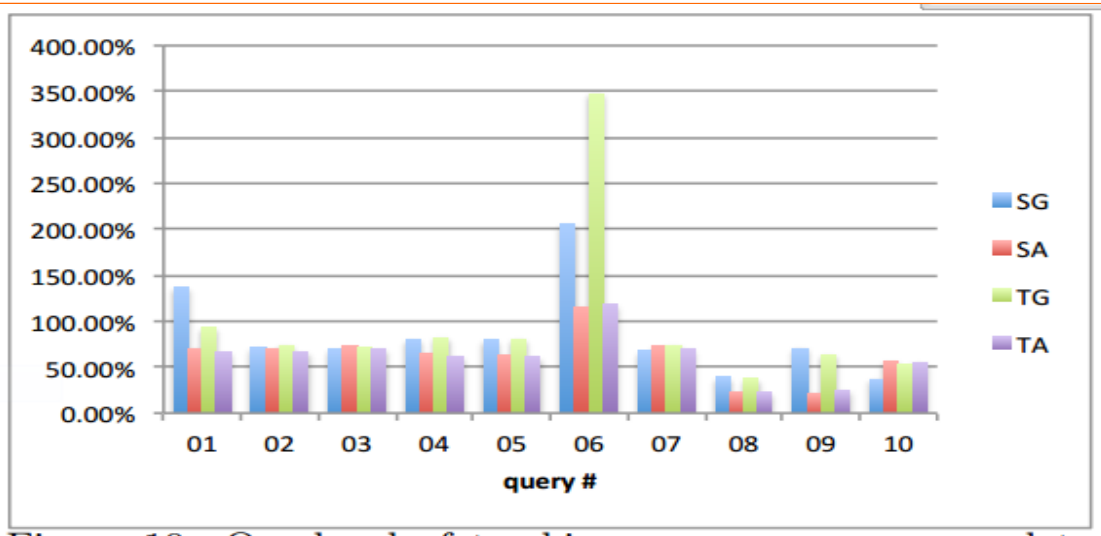


Figure 10: Overhead of tracking provenance compared to the vanilla version of the system for the BTC dataset

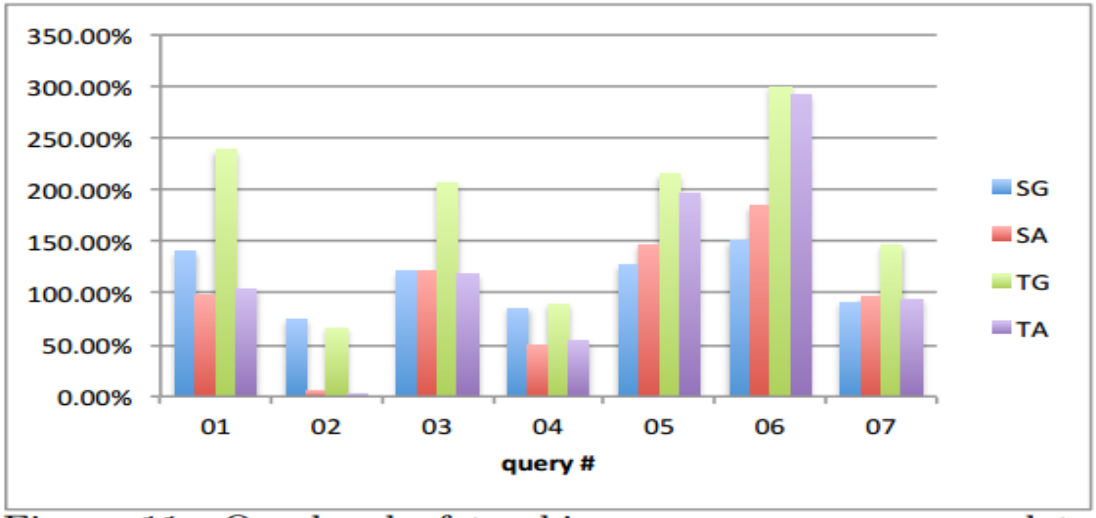


Figure 11: Overhead of tracking provenance compared to the vanilla version of the system for the WDC dataset

Loading Time and Memory Consumption

	V	G	A
Loading Time (min)	23.32	27.9	26.8
Memory Consumprion (GB)	36.26	53.62	39.54

Figure 12: Loading times and memory consumption for the BTC dataset

	V	G	A
Loading Time (min)	27.46	67.78	30.56
Memory Consumprion (GB)	42.53	66.22	50.29

Figure 13: Loading times and memory consumption for the WDC dataset

Presentation Outline

- Motivation
  - Querying Heterogeneous Data Sources
  - Provenance
  - Current State
  - Contribution
- Provenance Polynomials
- System Architecture
- Storage for Provenance
- Query Mechanism for Provenance
- Performance Evaluation
- Conclusion
- Future Direction

Conclusion

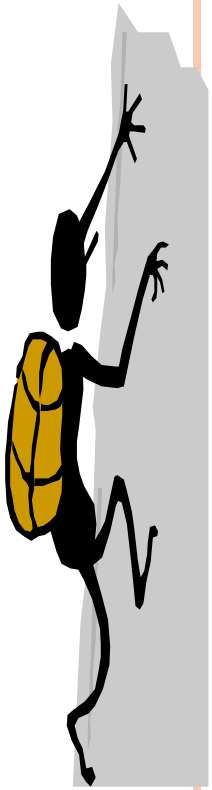
- Provenance overload is satisfactory. (on average 60-70%)
- Appropriate Storage model depends on the structure of dataset.
- Annotate model is better for heterogenous datasets.
- Co-located is better for homogenous data.

Presentation Outline

- Motivation
  - Querying Heterogeneous Data Sources
  - Provenance
  - Current State
  - Contribution
- Provenance Polynomials
- System Architecture
- Storage for Provenance
- Query Mechanism for Provenance
- Performance Evaluation
- Conclusion
- Future Direction

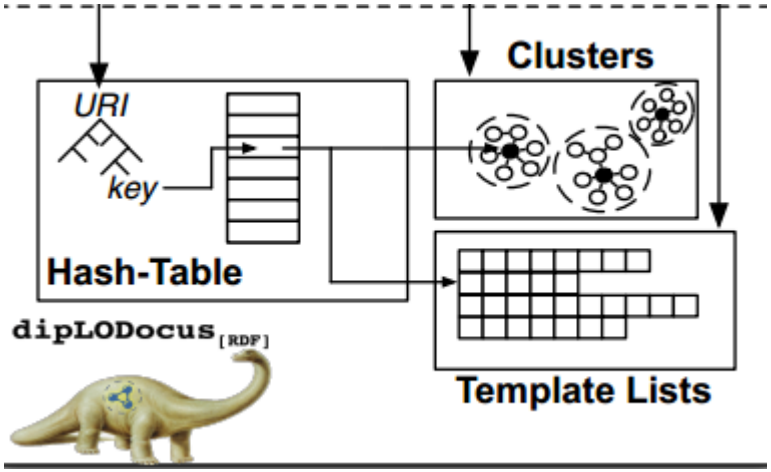
Future Direction

- Distributed version
- Optimization between query execution time
- and memory consumption
- Query over provenance of query result.



Thank you

System Architecture



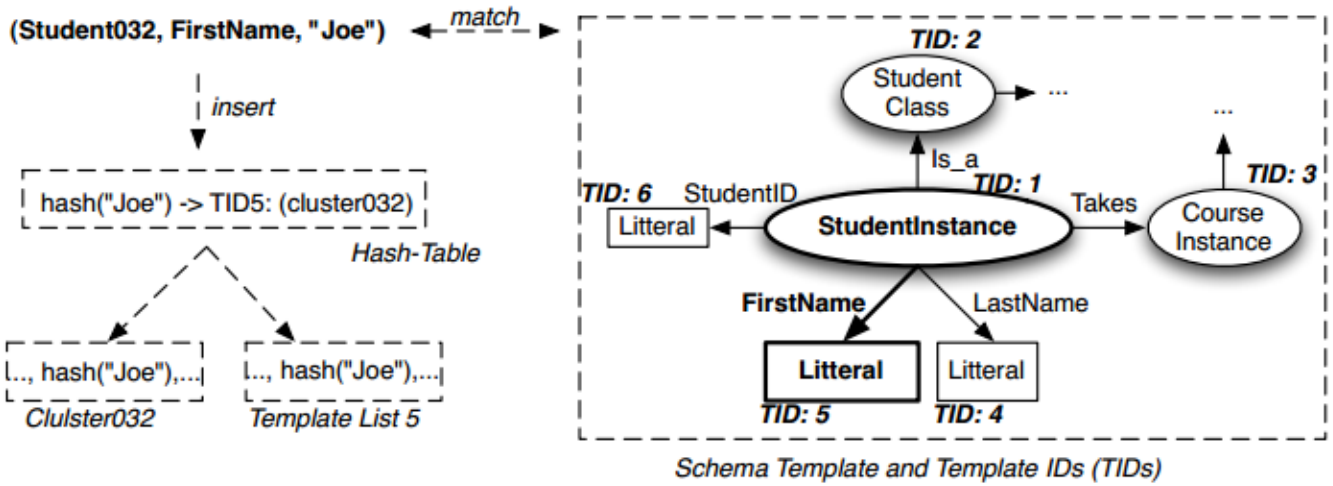


Fig. 3. An insert using templates: an incoming triple (upper left) is matched to the current RDF template of the database (right), and inserted into the hash-table, a cluster, and a template list.

BACKUP SLIDES