Graph Databases Lifecycle Methodology and Tool to Support Index/Store Versioning

Pierfrancesco Bellini, Ivan Bruno, Paolo Nesi, Nadia Rauch
DISIT Lab
Dipartimento di Ingegneria dell'Informazione, DINFO
Università degli Studi di Firenze
Via S. Marta 3, 50139, Firenze, Italy
Tel: +39-055-2758511, fax: +39-055-2758570
http://www.disit.dinfo.unifi.it ***alias*** http://www.disit.org
Paolo.nesi@unifi.it

Context and Motivations

- Graph database are taking place for systems exploiting knowledge base, KB
  - Include a set of ontologies and data instances: static data, reconciliation data, real time data, historical data, gelocated data, etc.
  - Smart city, smart cloud, smart learning, etc.
- **KB need complex and non consolidated methodologies for their implementation**
  - Many issues may lead to invalidate a building in favor of new version, e.g., looking for more inference, corrections in model ontology, changes in historical data, etc.
  - Methodologies present iterative processes that lead to restructure/rebuild the knowledge base

Context and Motivations (2)

- RDF stores (end they end points) are a way for Knowledge Base implementation
- RDF stores see Ontologies and Instances as triple (quadruple) s-p-o (context)
- RDF stores presents several problems in managing:
  - **Versioning** and thus deleting triples including the inferred triples… at real time performance ?
  - **Performance in db store building (inference)**
  - Performance in db store querying (inference)

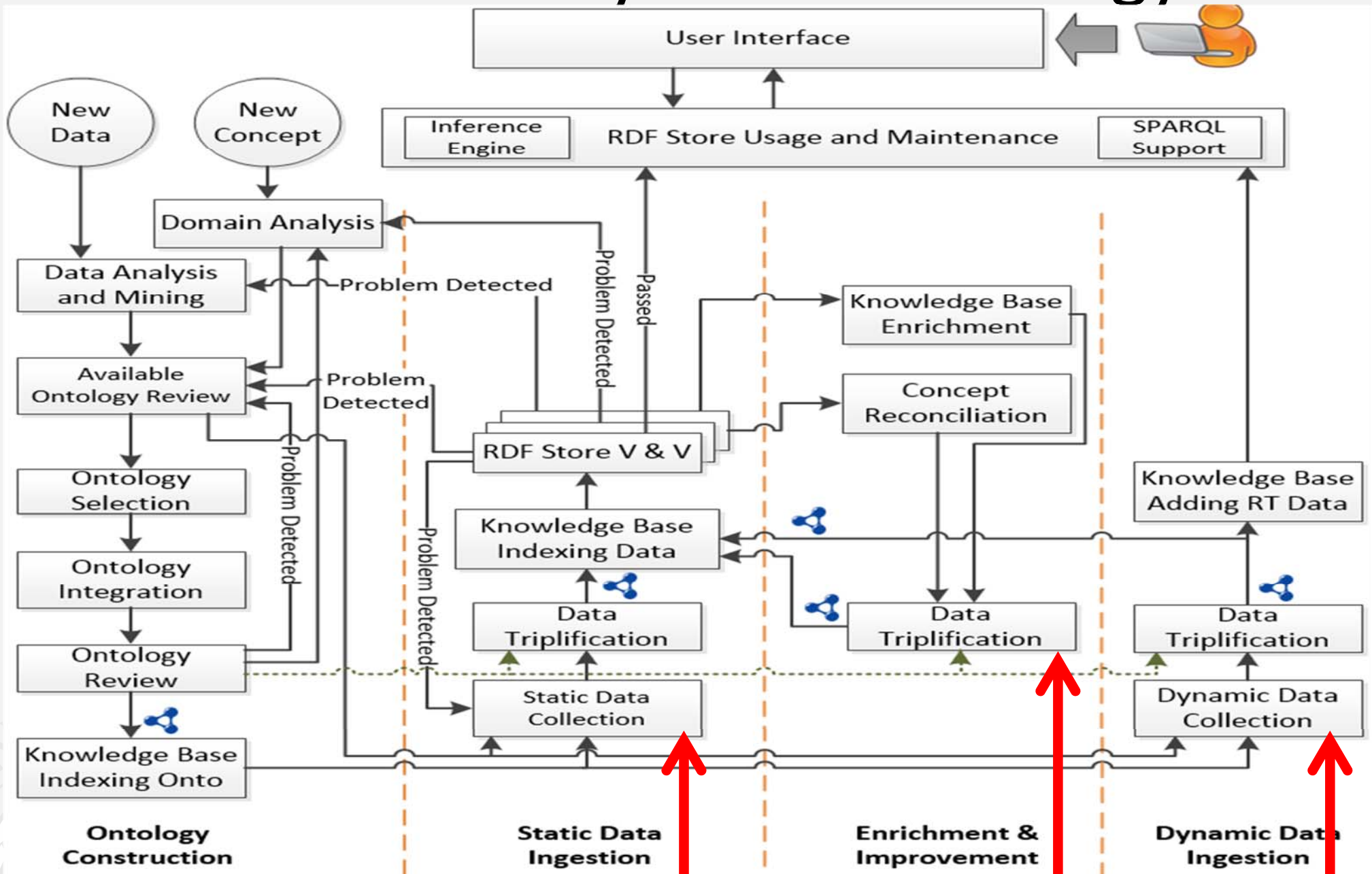
The paper contributions

- A methodology for knowledge base life cycle building and improvement
  - addressing more details and data kind and cases of the state of the art methodologies
- A versioning system for RDF store building supporting the methodology
  - Completely new tool that can save up to 90% of time in RDF rebuilding

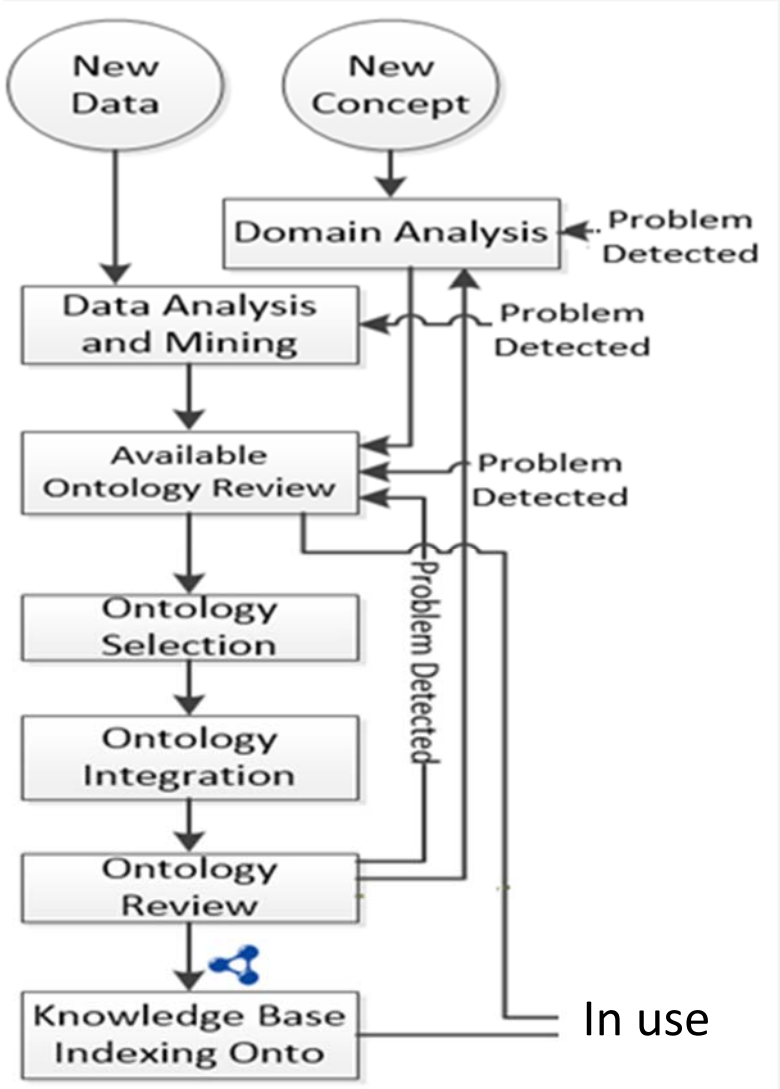
RDF store service: building process

- built incrementally via progressive refinements mediating among
  - reusing ontological models,
  - increasing the capability of making deductions and reasoning on the knowledge base concepts,
  - maintaining acceptable: query and rendering performances,
  - simplifying the design of the front-end services,
  - flexibility to the arrival of additional data and models and/or corrections,

RDF KB life cycle methodology



Ontology Construction

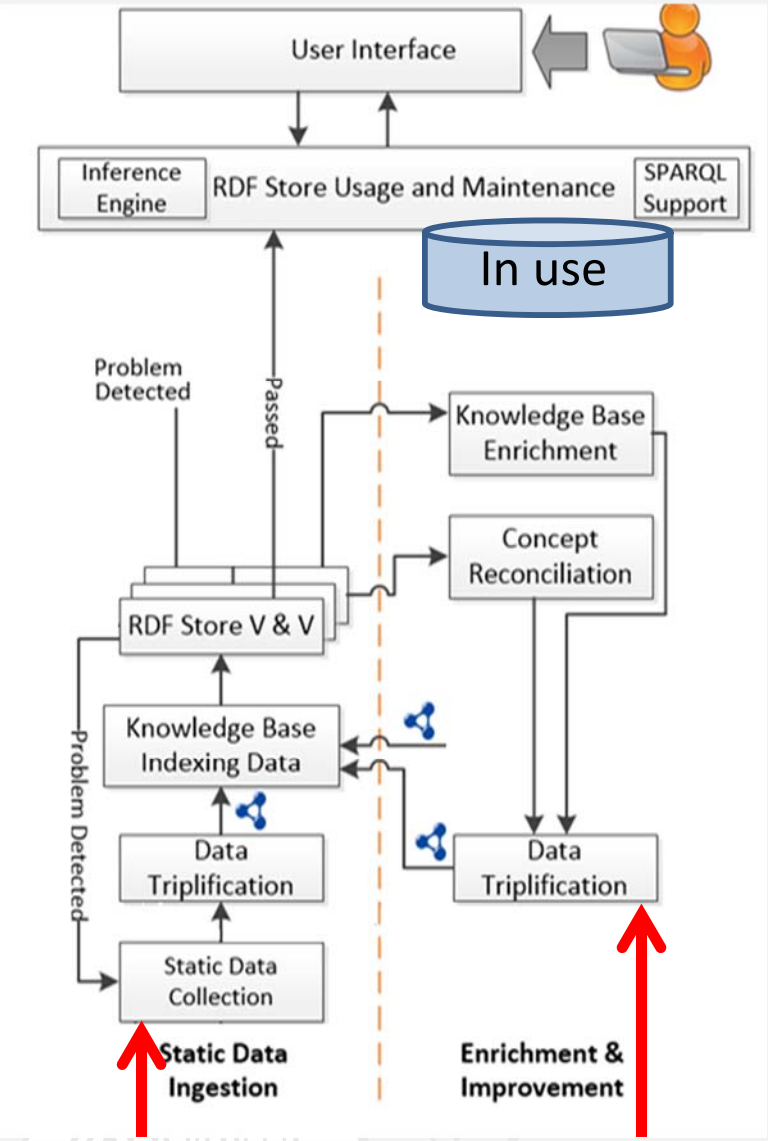


- Domain Analysis:
  - concepts, abstractions, aggregation, classes vs attributes, etc.
- Available Ontologies
  - Diffusion, Licensing, quality, inference, etc.
- Ontology Integration
  - Glued concepts, more inference
- Ontology review
  - Conceptual assessment
  - Formal verification with metrics and tools
- Points from which one has to restart, have to be saved
- Integrated version of the ontology goes IN USE

Km4city example

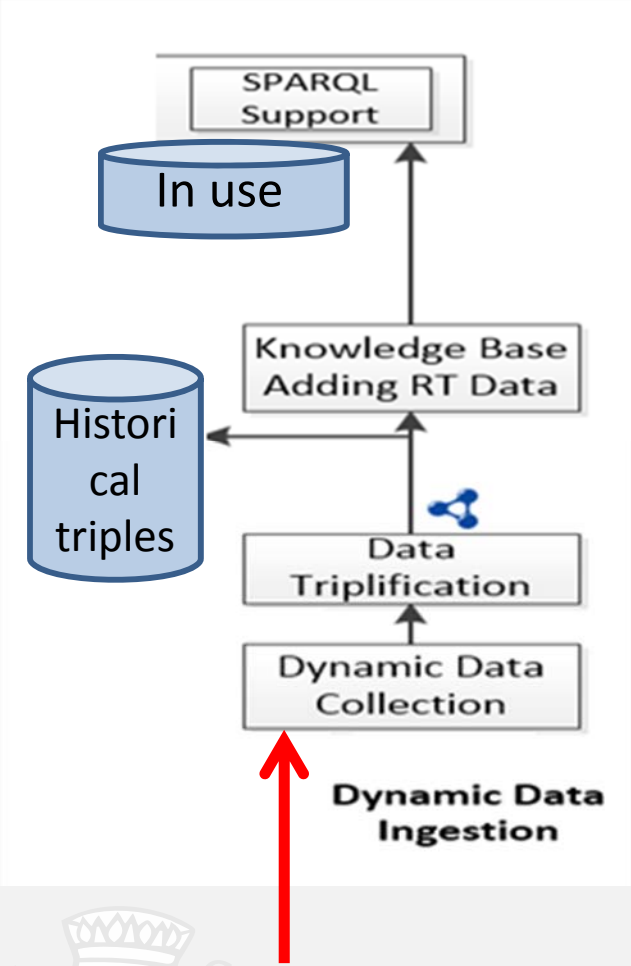
- reuses:
  - *dcterms* to set of properties and classes for modeling metadata;
  - *foaf* dedicated to relations among people or groups;
  - *schema.org* for a description of people and organizations;
  - *wgs84_pos* representing latitude and longitude;
  - *GoodRelations* for a description of business entities and their locations;
  - *OWL-Time* for temporal modeling;
  - *OTN* for transport aspects;
  - *GIS Dictionary*, to represent the spatial component of geographic features

Data ingestion and enrichment



- Static Data can be:
  - Open Data,
  - Historical real time data (from – to)
  - Enrichment data (to be identified)
  - Reconciliation Data (to be identified)
- Problems may derive from:
  - Inconsistencies instance level ?
  - Incompleteness: missing onto concepts, missing links, …

Real Time Data



- Need to be already verified and reconciled
  - No changes in their structure
  - No or known variability on instances
- May produce large volume of Cumulated data
  - can become the substantial part of the KB
  - cannot be deleted
  - Cannot be easily extracted from the RDF store, thus historical data may be saved for rebuilding

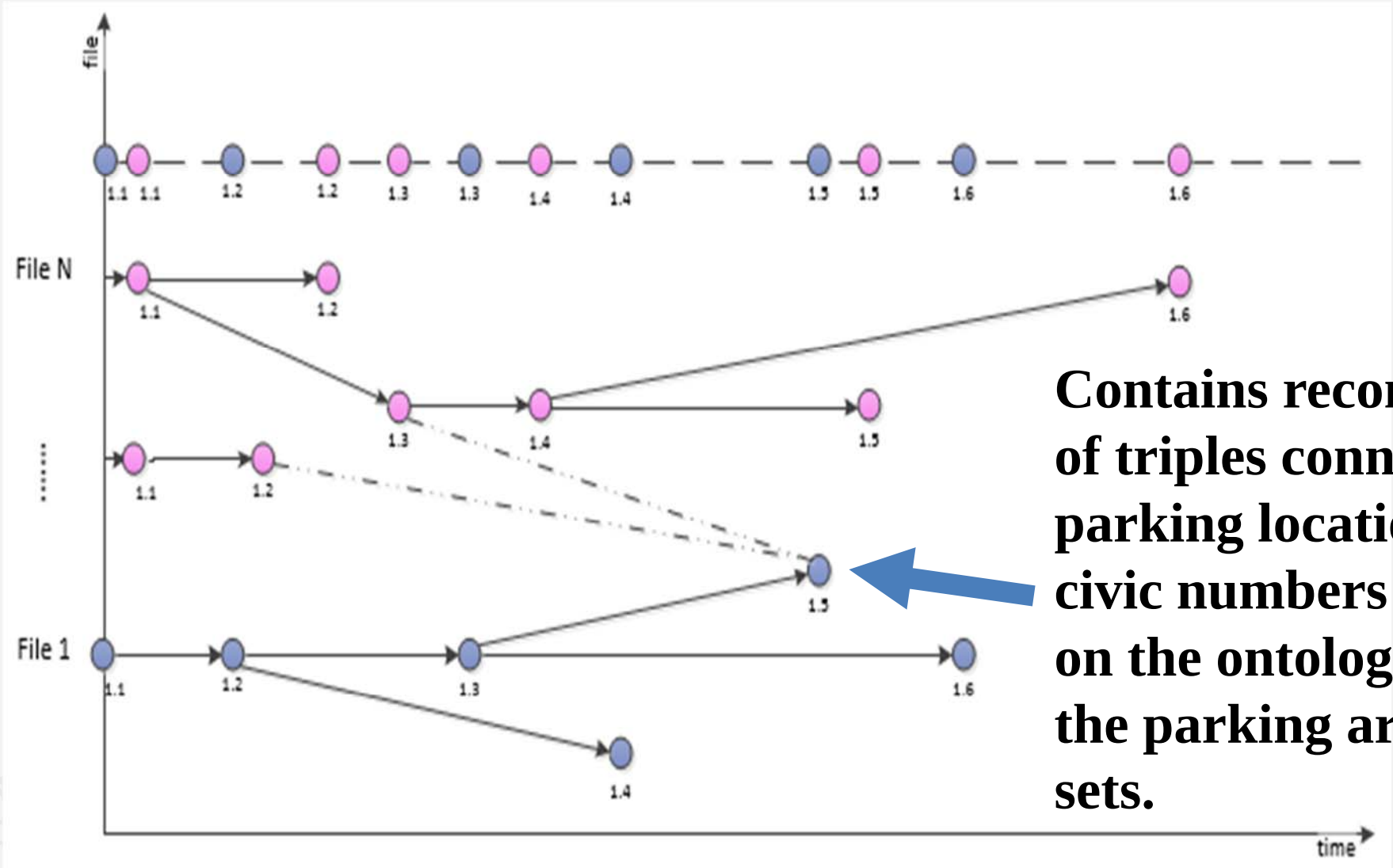
RDF Indexing Flow and Requirements

- There are several reasons to revise/change the RDF Indexing and thus the **RDF Store itself**
  - the ontology and thus the:
    - data mapping and triplification invalidating the indexing and the materialization of triples
  - the data triples coming from ingestion, historical, reconciliation, as:
    - a new data mapping, quality improvement,
    - changes in performing reconciliation, enrichment and triplification processes.

RDF Index and RDF Index Descriptor

- The RDF Index is substantially the RDF Store containing the triple (ontology, static data, etc…)
- The RDF Index Descriptor is:
  - the recipe to create the RDF Index and
  - the set of triples adopted to build it
- **Since the version of the RDF Store is not an a viable task (without redesigning the RDF store) we performed the versioning of the RDF Index Descriptor**

Versioning: File and RDF Index

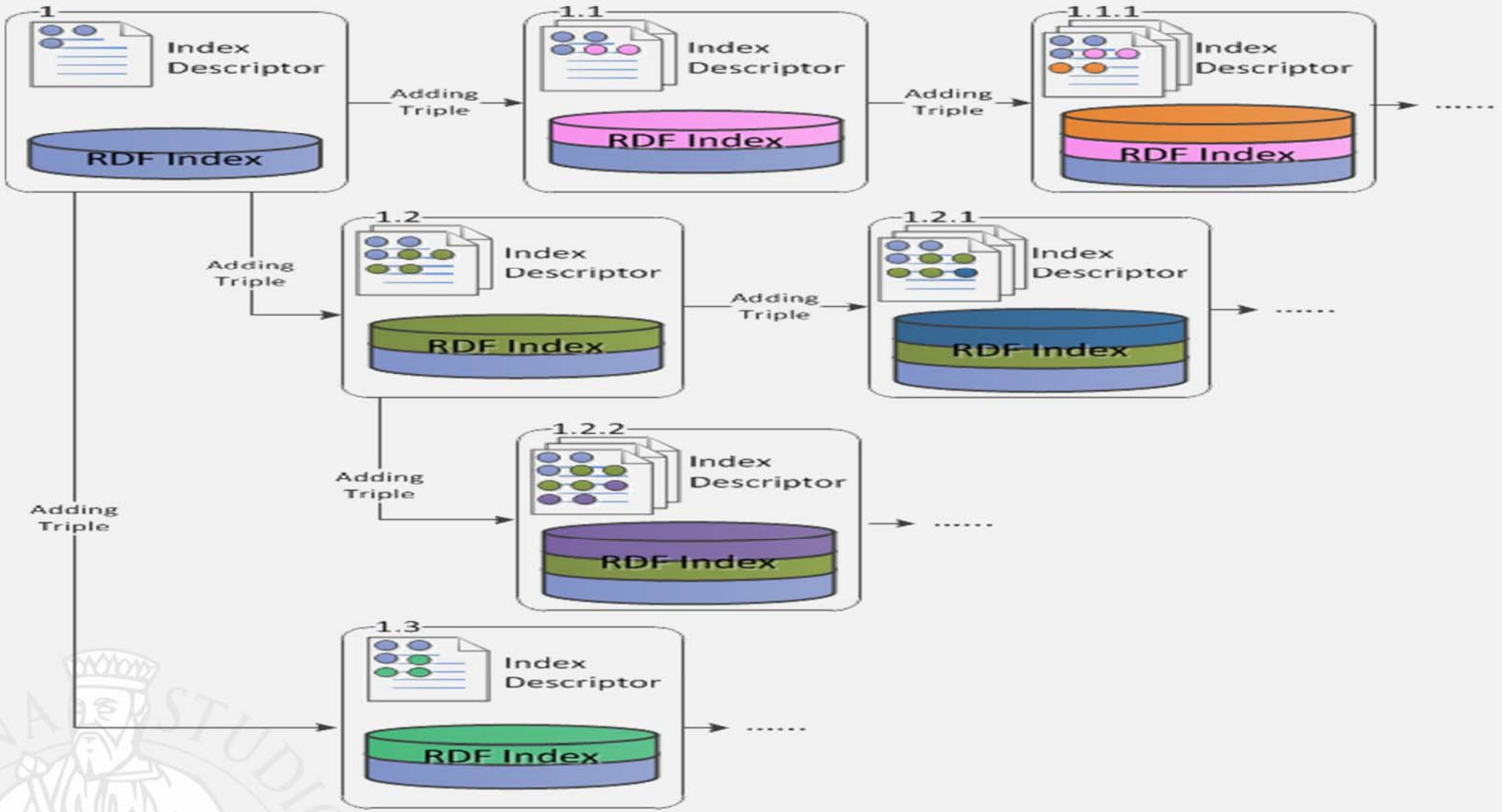


Contains reconciliation of triples connecting parking locations wrt civic numbers depends on the ontology and on the parking area data sets.

Verification and Validation

- In a real production of big data RDF store,
  - hundreds of files containing triples are produced
  - The restarting from scratch is time consuming, may be error prone, may lose the versioning / evolution value
- Not all changes can produce consistent RDF KB Store.
- The saving of intermediate consistent version can lead t o save time in exploiting the saved version as restarting point.

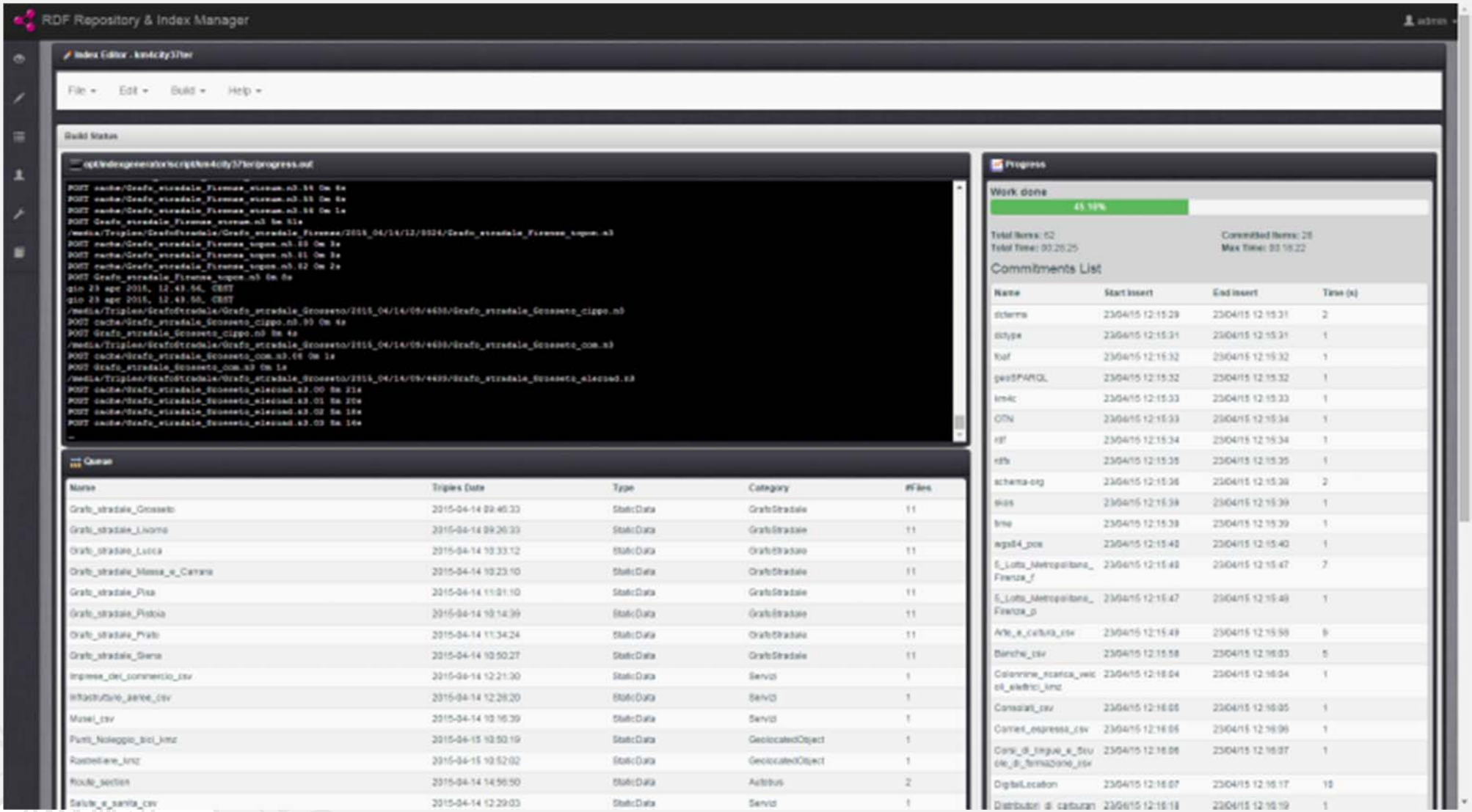
Example: 4 versions on the same core



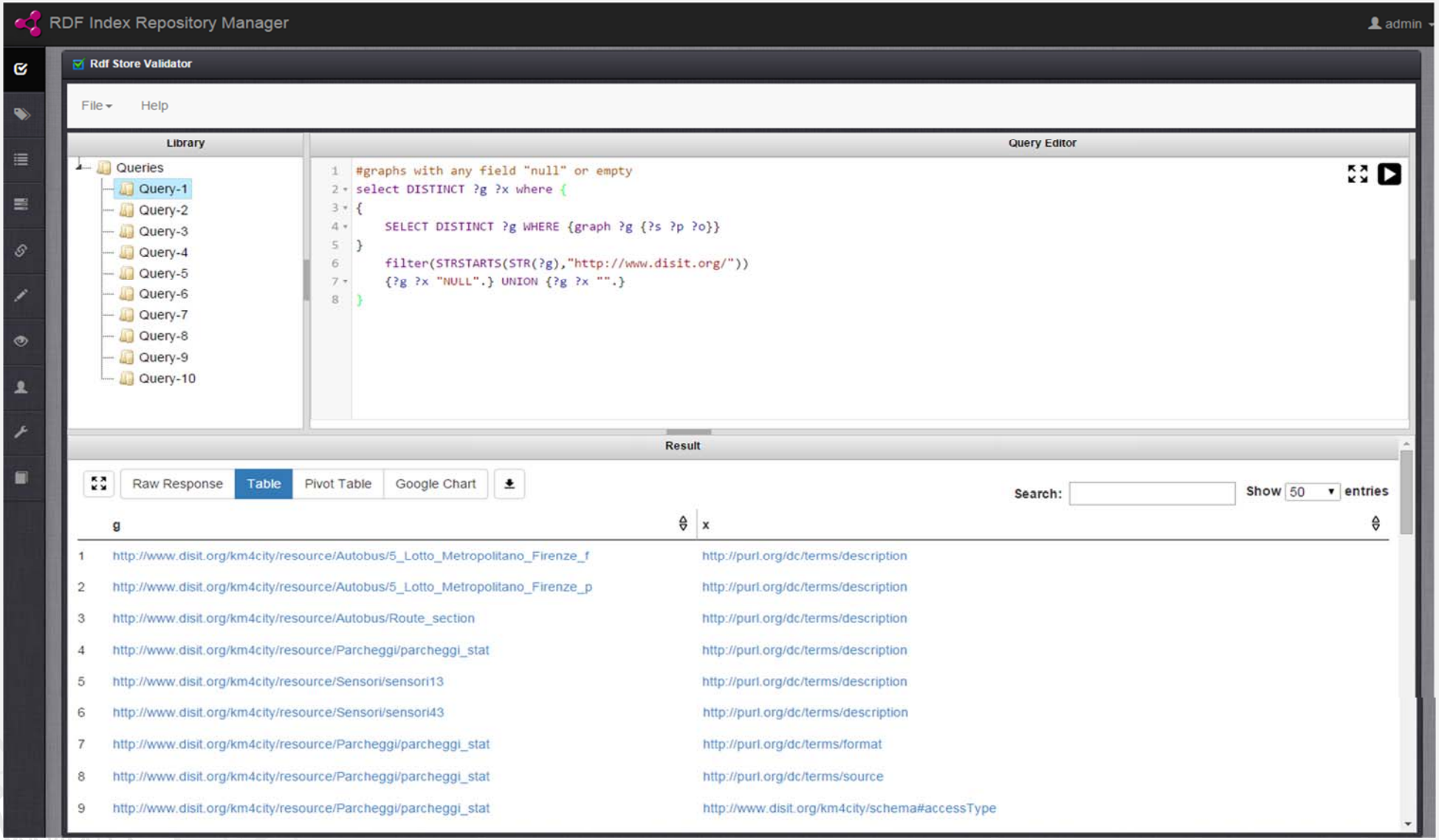
RDF Index Manager tool

- **Keep tracing** *RDF KB Store Versions, RKBSV,* in terms of *files of triples*, *index-description*, and RDF *Index*;
- **Maintaining a repository** of *RKBSVs* where they could be stored and retrieved;
- Selecting a *RKBSV* from the repository for modification, to examine changes and the history version, to be used as base for building a new version;
- Managing the *index descriptor* as a list of files containing triples;
- **Generating a RDF KB index** on the basis of an *RKBSV* independently from the RDF store kind automatically, and in particular for SESAME OWLIM and Virtuoso;
- Monitoring the RDF KB index generation and the feeding state;
- **Suggest the closest version** of the *RKBSV* with respect to the demanded new index in terms of files of triples;
- **Avoiding manually managing the script file of indexing**, since it is time consuming and an error prone process.

RIM tool: building monitor during production



RIM tool: during RDF assessment



Experimental Results

	Ontologies	+ street graphs	+ smart city Services	+Enrich& Reconciliations	+Historical data 1 month
Indexing process					
Final number of triples	**15.809**	**33.547.501**	**34.462.930**	**34.557.142**	**44.218.719**
Final number of Files	12	137	178	185	27794
Added triples with respect to previous version	15809	33.531.692	915.429	94.212	9.661.577
Added Files with respect to previous version	12	125	41	7	27609
OWLIM SE 4.3					
Indexing Time without RIM (s)	**18**	**6536**	**6198**	**7516**	**12093**
Indexing Time with RIM (s)	**11**	**6029**	**514**	**343**	**5745**
% of saved time, RIM versioning	**38,9**	**7,8**	**91,7**	**95,4**	**52,5**
Final Number of triples (including geo + inferred)	16062	57.486.956	59.395.432	59.486.748	73.441.126
disk space in Mbyte	310	8669	8936	9039	13110
VIRTUOSO 7.2					
Indexing Time without RIM (s)	**146**	**806**	**964**	**1000**	**2487**
Indexing Time with RIM (s)	**156**	**833**	**421**	**296**	**1932**
% of saved time, RIM versioning	**-6,8**	**-3,3**	**56,3**	**70,4**	**22,3**
Final Number of triples (including geo, no inferred)	21.628	35.452.613	36.301.322	36.420.445	46.232.510
disk space in Mbyte	68	1450	1632	1631	2294
GraphDB 6.1					
Indexing Time without RIM (s)	**9**	**7818**	**7929**	**7671**	**12915**
Indexing Time with RIM (s)	**2**	**6791**	**454**	**214**	**4849**
% of saved time, RIM versioning	**77,8**	**13,1**	**94,3**	**97,2**	**62,45**
Final Number of triples (including geo + inferred)	15.809	57.486.415	59.394.891	59.487.551	73.441.929
disk space in Mbyte	96	4276	4466	4643	5714

New Version

- Support Ontology Licensing
  - To take into account in ontology building and in KB RDF store usage, querying
- Support Data Licensing
  - To help selecting ontologies
  - To take into account in KB RDF store usage, querying
- Support Data ingestion process with
  - integrated Data Ingestion Manager
  - Maintaining under control the data sets to be included, their licenses, triplification, etc.

Conclusions

- The RIM model and tool allow:
  - Keeping under control and trace the RDF life cycle from construction of the ontology to the indexing and validation.
  - Reducing the time for indexing (RDF store construction) up to the 97% in some cases.

The benefits have been demonstrated for the most diffused RDF stores: OWLIM, GraphDB and Virtuoso.