



Data-Integration & Reasoning Middleware

ORDI, Rich-RDF, OWLIM, Light-weight Inference

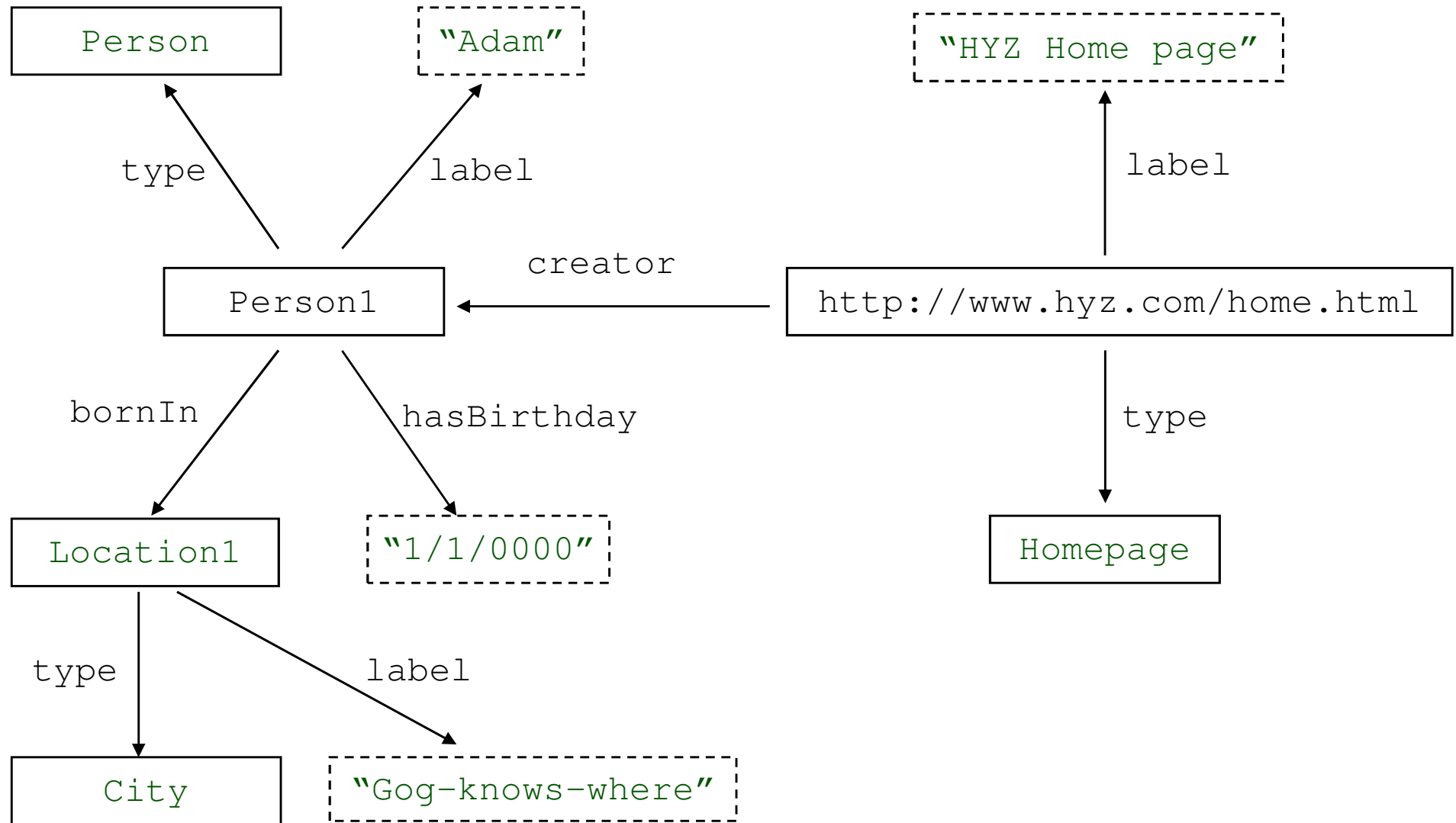
Atanas Kiryakov

Apr, 2008

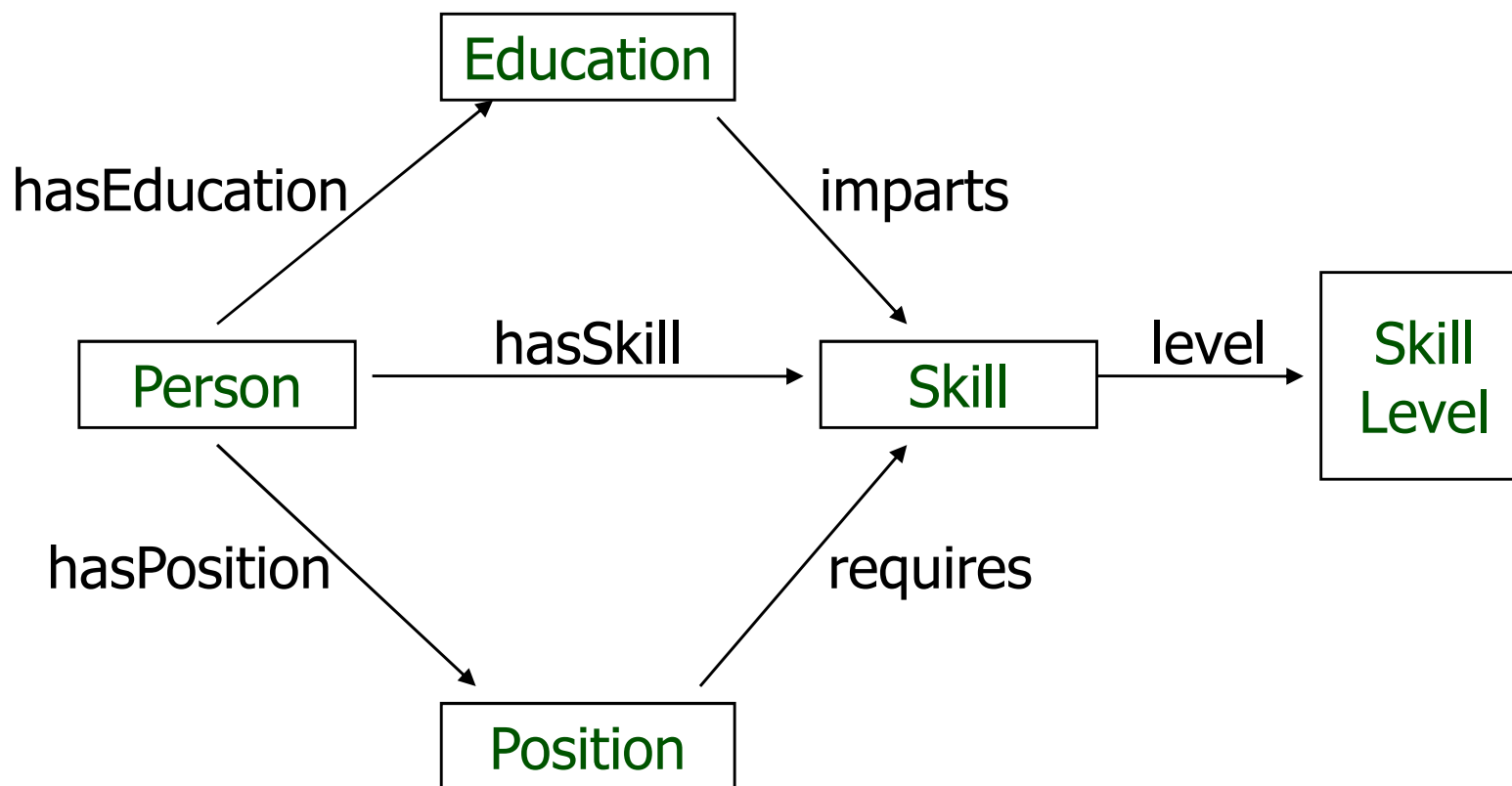
Outline

- RDF models
- Architecture – Sesame, TRREE, ORDI, OWLIM
- ORDI
- OWLIM
 - Reasoning support
 - Versions, Scalability & Performance
- WP5 notes

Piece of RDF Graph

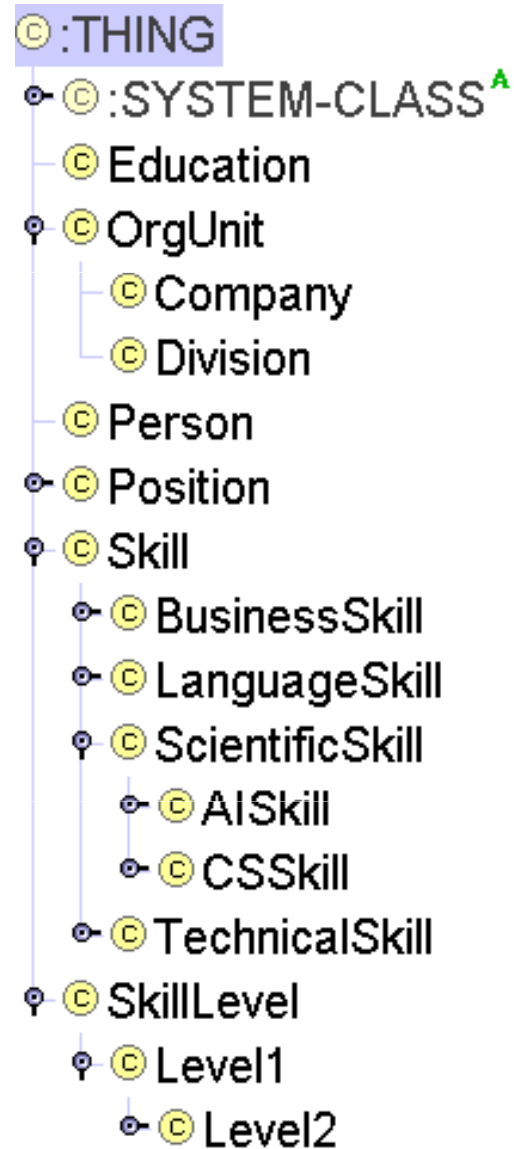


RDF: The schema as ER-graph

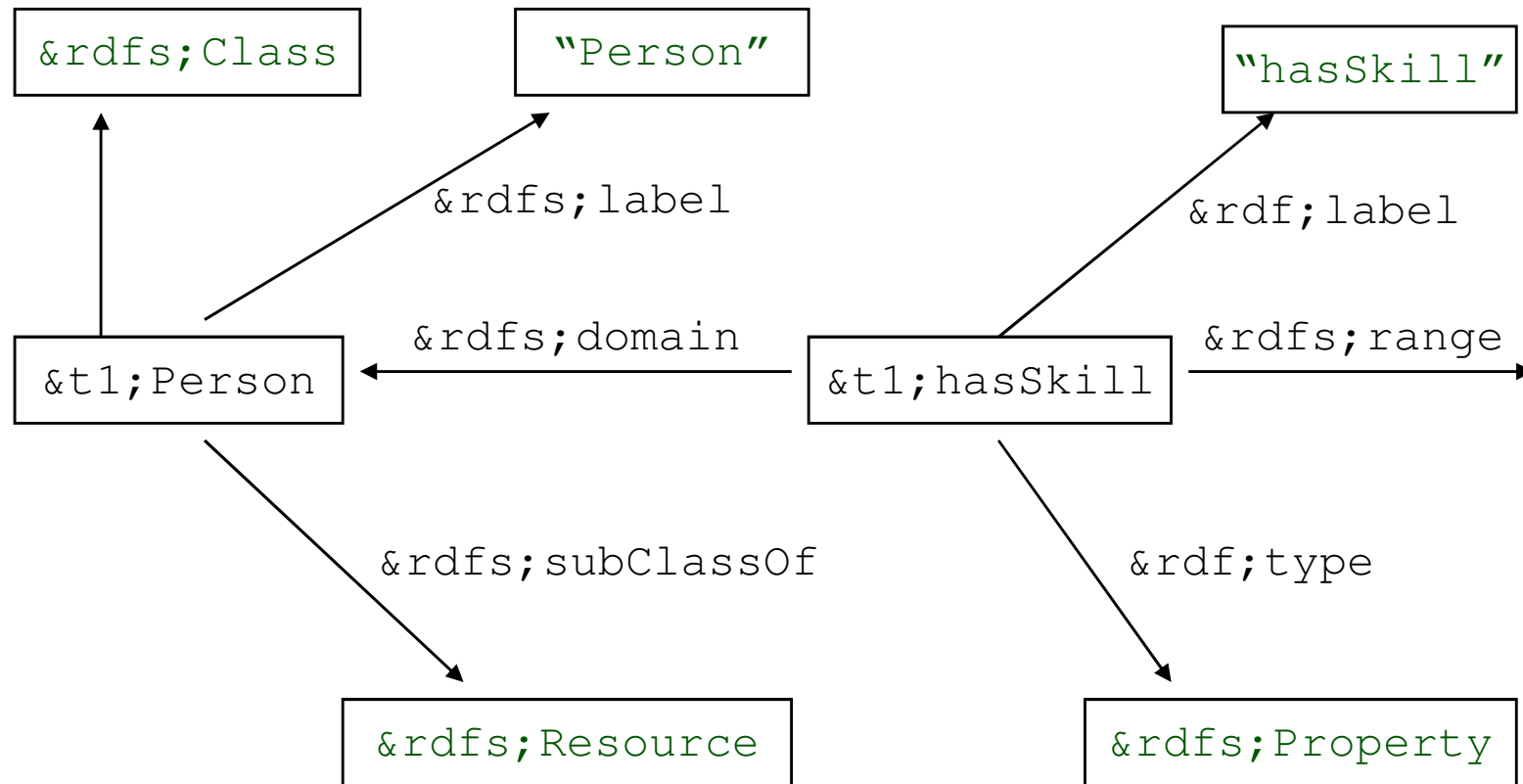


Simplified conceptual schema of a Skills Management Application

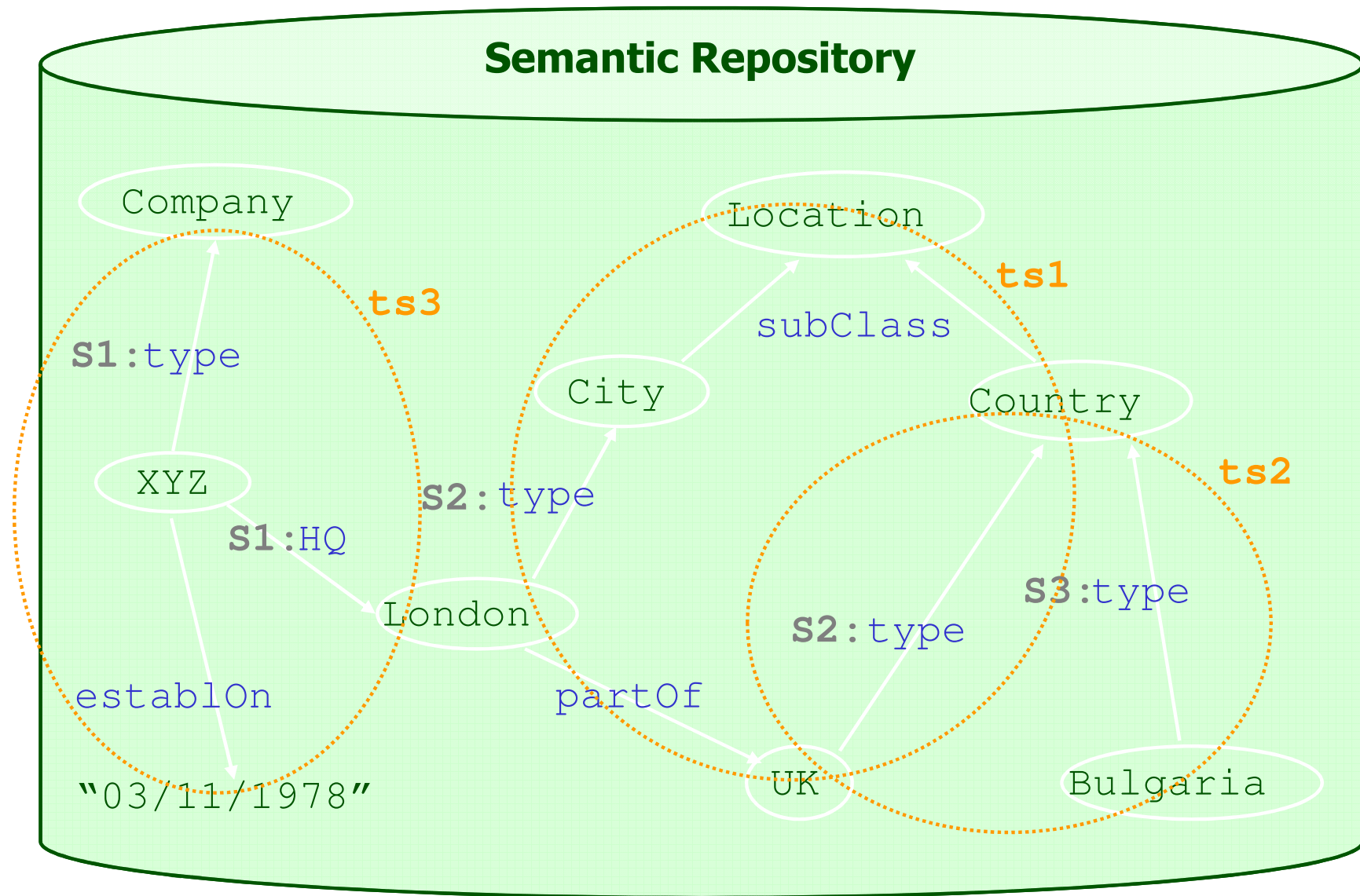
RDF: Sample Class Hierarchy



RDF: Piece of the real graph



ORDI v0.5 Data Model



ORDI Data Model

- ORDI Data Model
 - Directed multi-graph
 - Specialized support of meta-information
 - Fully compatible with:
 - RDF triple data model $\langle S, P, O \rangle$
 - Named Graphs $\langle S, P, O, NG \rangle$

- Triplesets are introduced

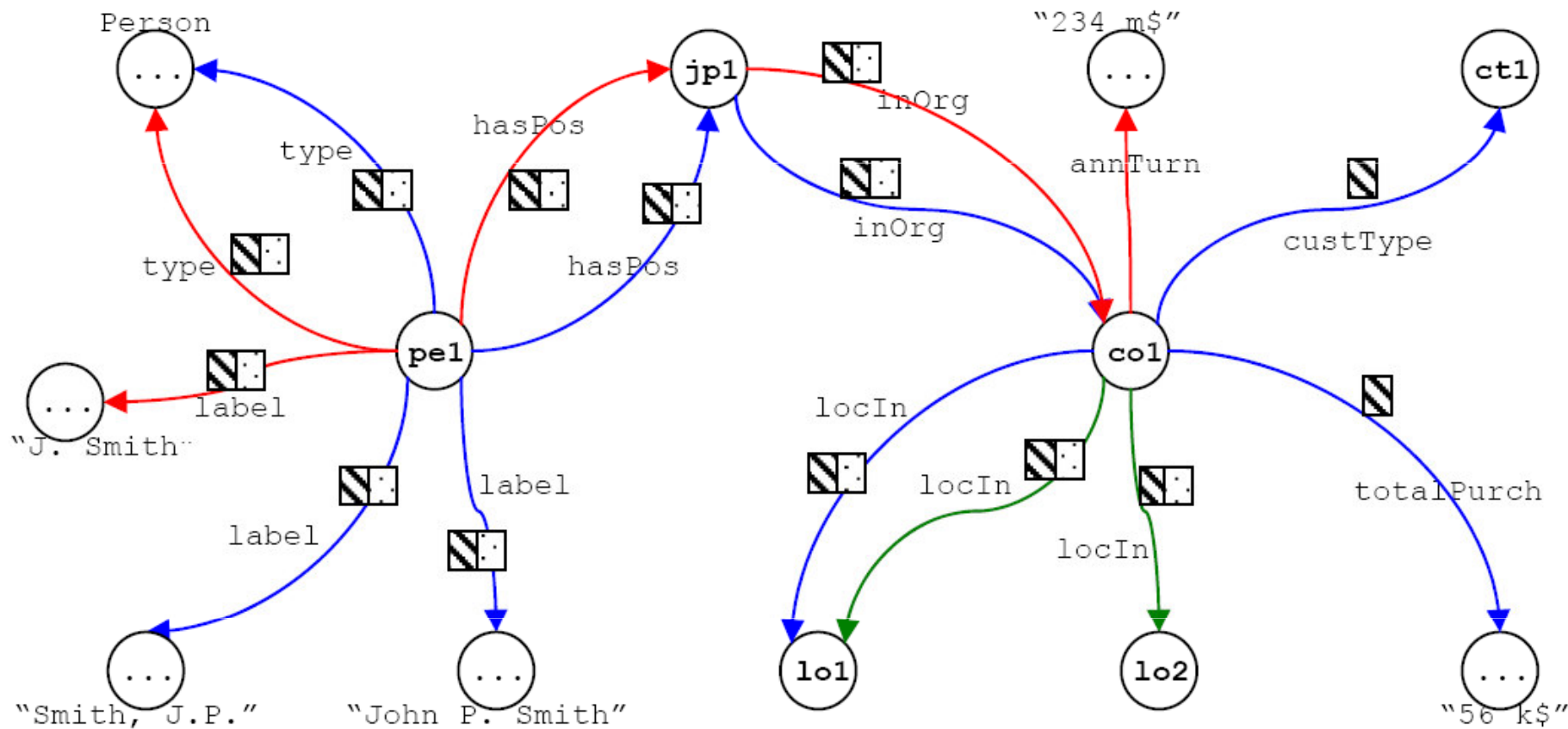
$\langle S, P, O, NG, \{TS_1, \dots, TS_N\} \rangle$

- A possible implementation

$\langle S, P, O, NG, ID \rangle$

$\langle ID, \text{associatedWithTripleSet}, TS_1 \rangle$

Is it the RDF enough?



Named graphs: ng1, ng2, ng3

Triplesets: ts1, ts2

ORDI data model is:

- Well-defined data-model semantics
- Neutral in terms of interpretation and semantics by not prescribing any sort of epistemological semantics
- Capable to efficiently support meta-data on statement level
- Backward compatible extension of RDF and Named Graphs
- Triplesets name avoid ambiguity with:
 - Contexts – in the way they are introduced in Sesame 2.0
 - Datasets – in the way they are introduced in SPARQL
 - Named-graphs, in the way they are introduced at <http://www.w3.org/2004/03/trix/>, impl. in Jena, used in SPARQL
- http://www.ontotext.com/ordi/ORDI_SG/ORDI_Datamodel.html

ORDI Data Model

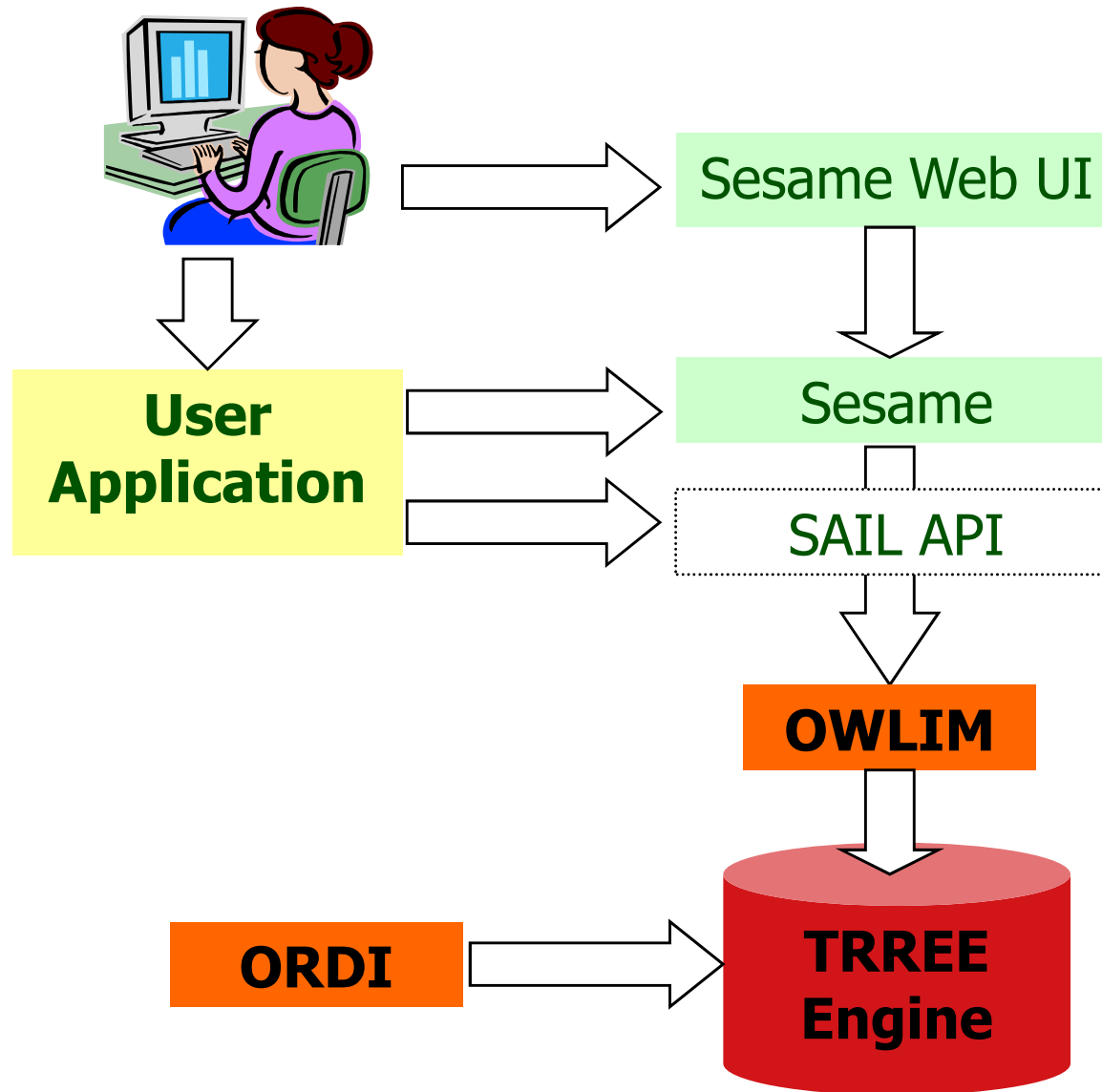
- Solution to WSMML to RDF translation problems, which is compatible with the existing infrastructure

| S | P | O | NG | TS |
|----------------|-----------------|--------------|------------|--|
| Boieng747 | dc:title | "A" | Location:1 | ts:updateDate; ts:concept wsml:NFP |
| Boieng747 | dc:title | "B" | Location:2 | ts:updateDate; ts:instance wsml:FP |
| ts:updateDate1 | rdf:type | LastUpdate | | |
| ts:updateDate1 | rdf:value of | "28/10/2006" | | |

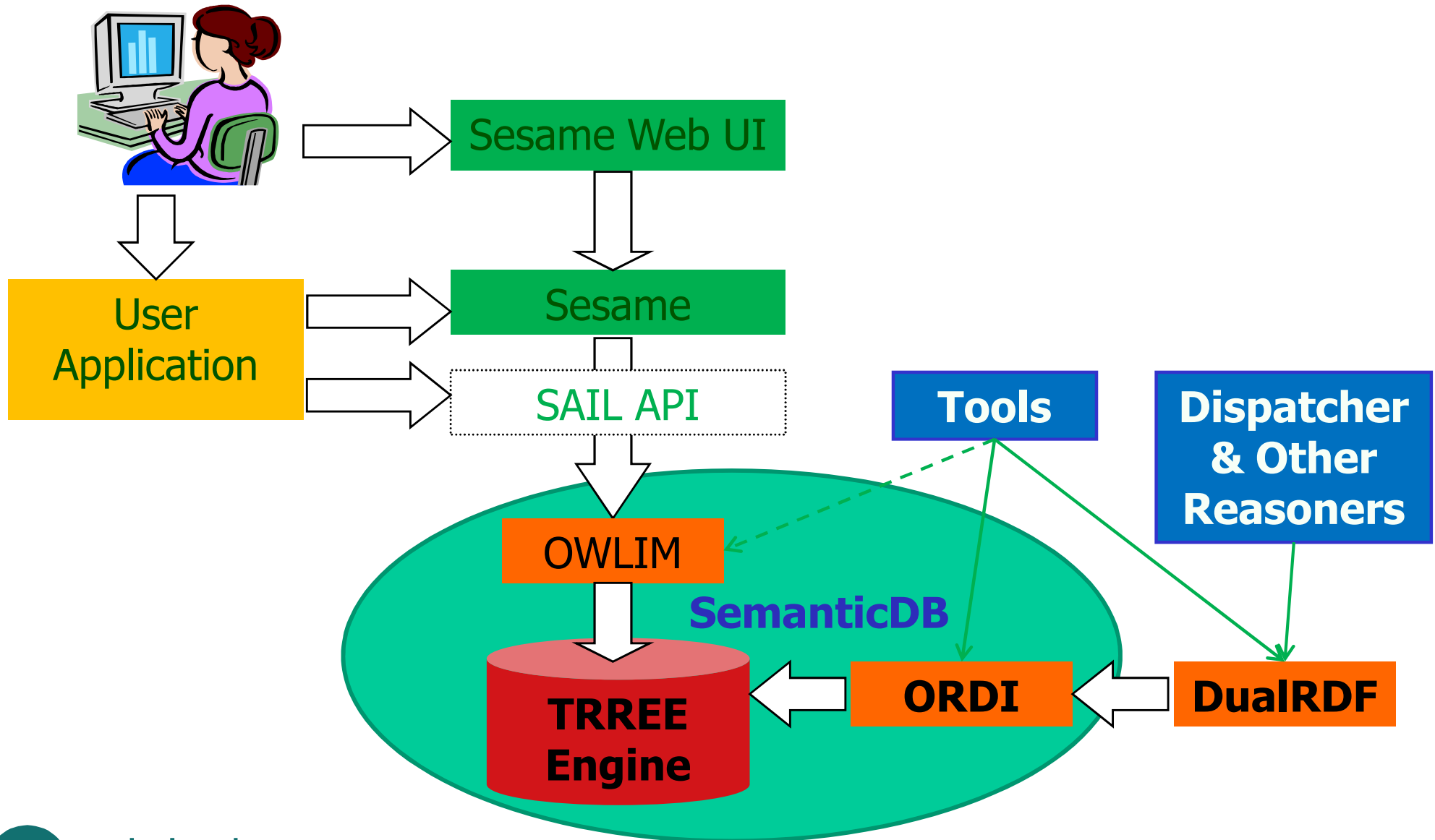
Outline

- RDF models
- **Architecture – Sesame, TRREE, ORDI, OWLIM**
- ORDI
- OWLIM
 - Reasoning support
 - Versions, Scalability & Performance
- WP5 notes

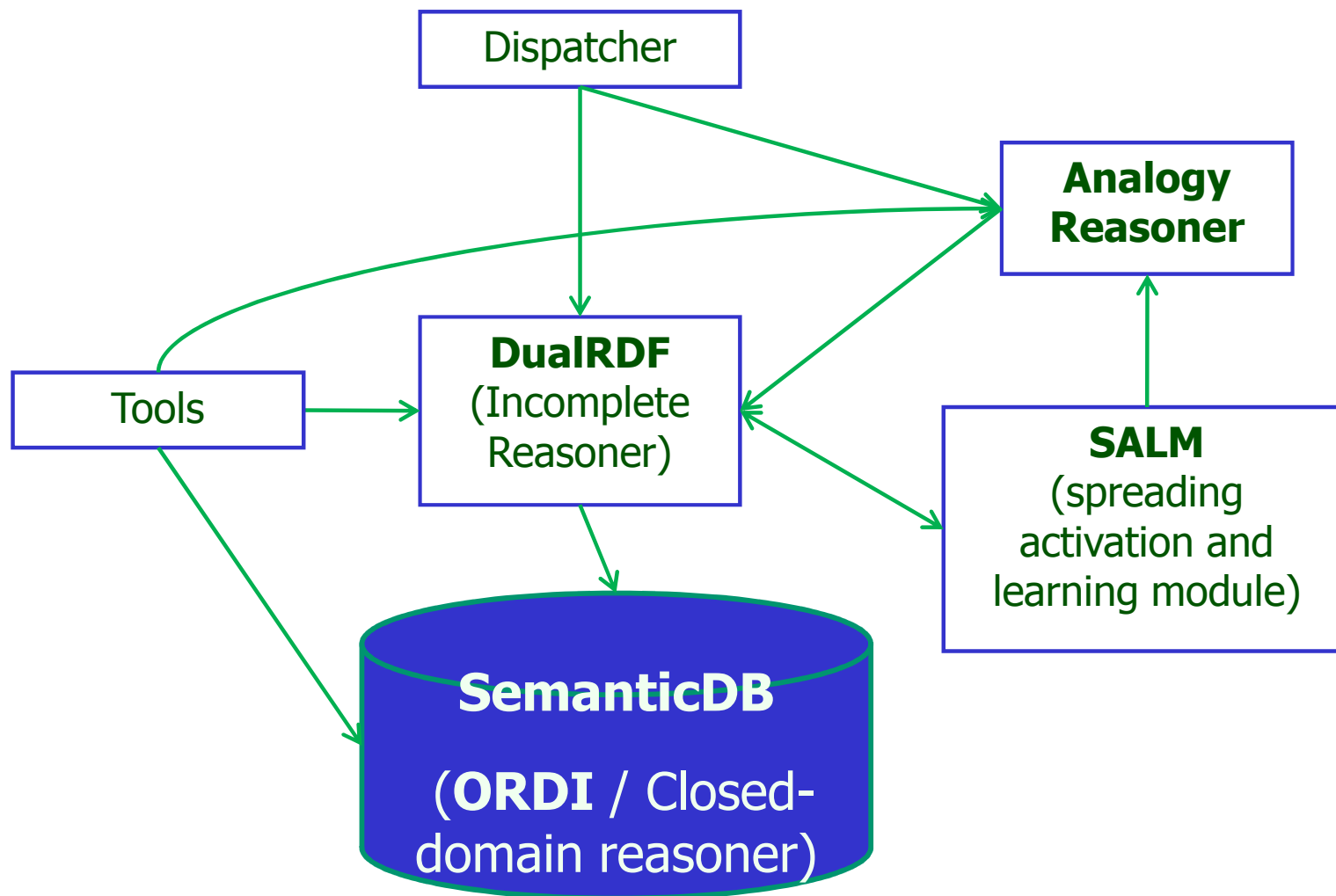
TRREE, ORDI, OWLIM



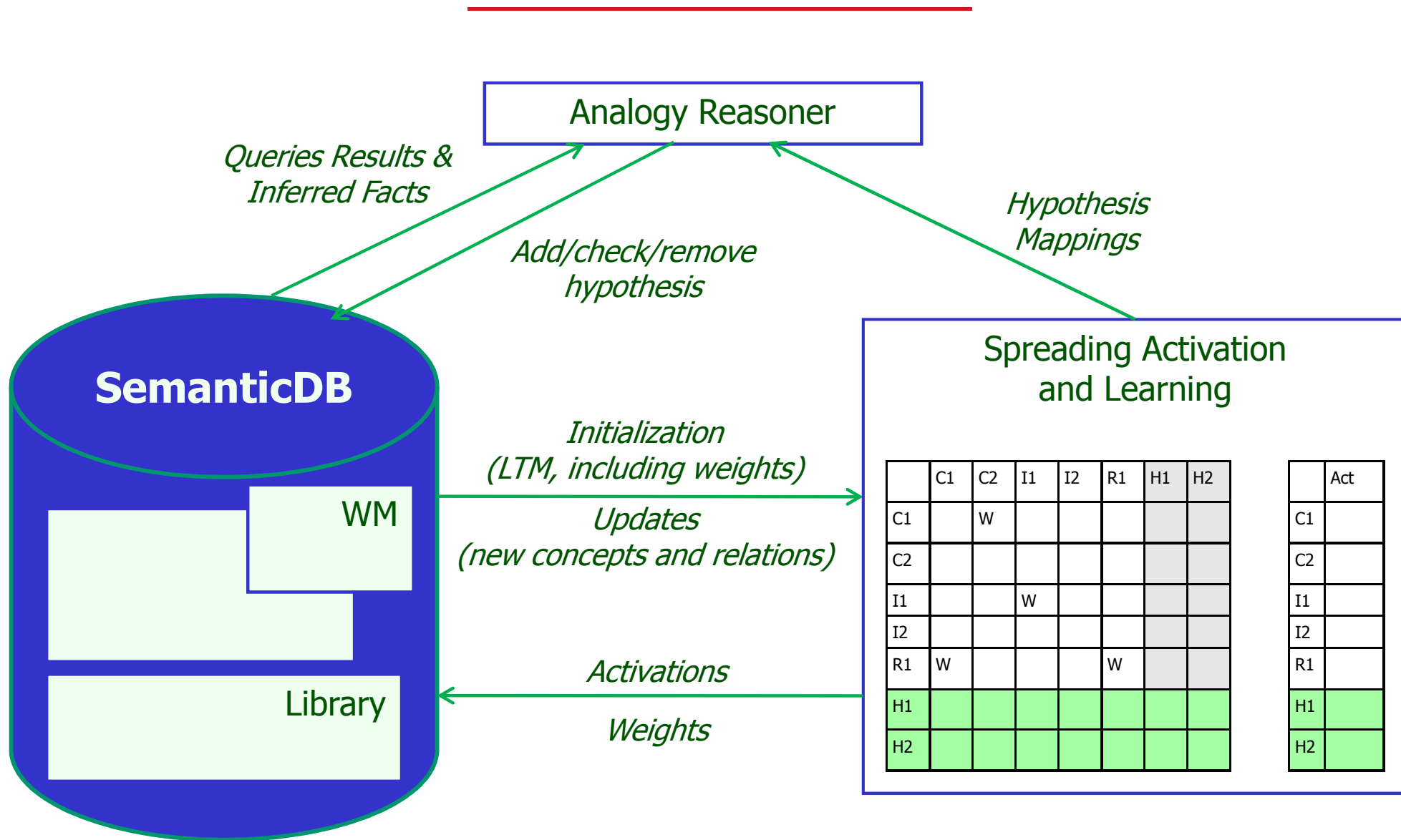
RASCALLI's SemanticDB Architecture



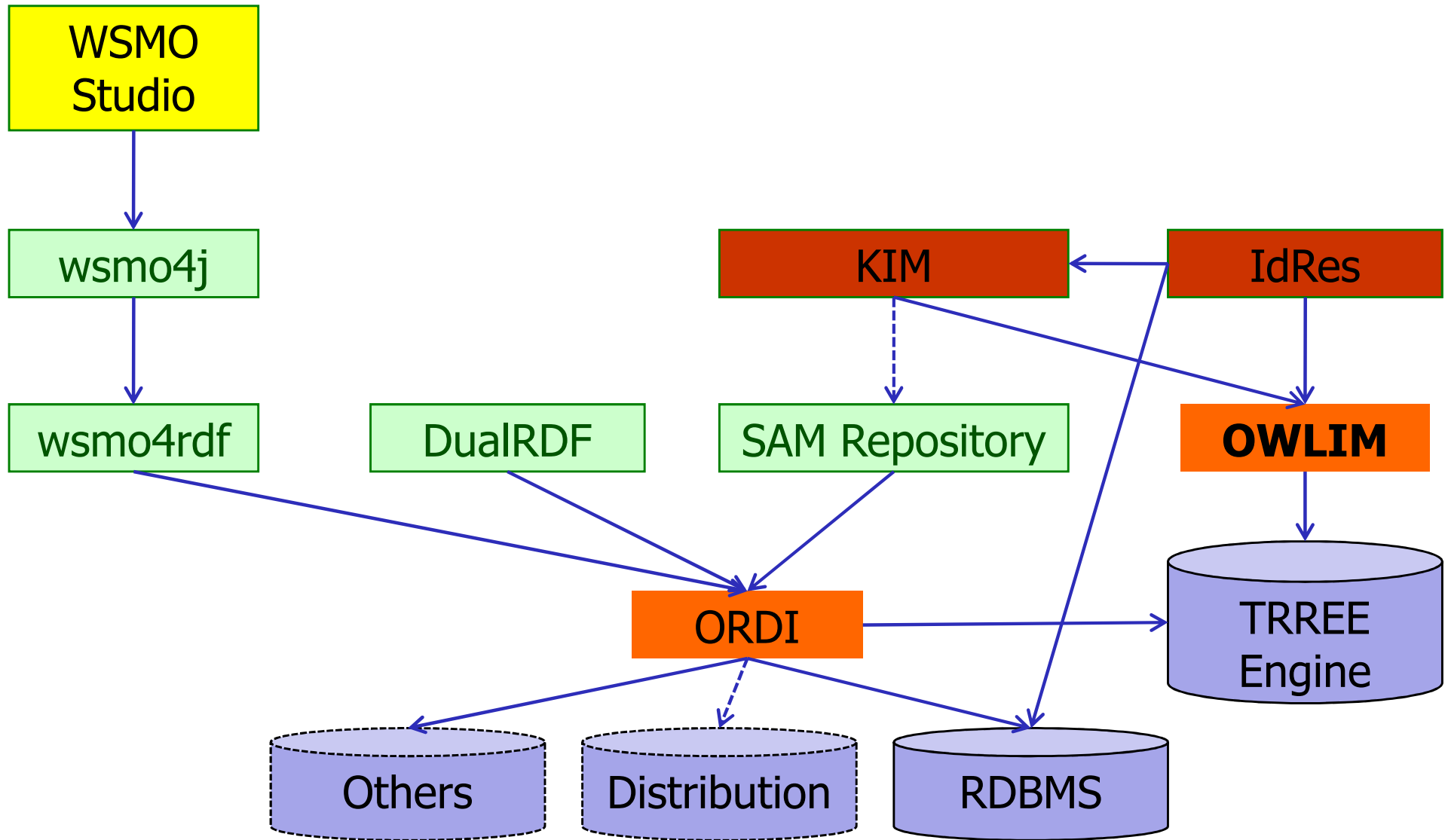
DualRDF and Reasoner integration



DUAL re-implementation architecture



TRREE, ORDI, OWLIM



Outline

- RDF models
- Architecture – Sesame, TRREE, ORDI, OWLIM
- **ORDI**
- OWLIM
 - Reasoning support
 - Versions, Scalability & Performance
- WP5 notes

ORDI Current Release

- <http://ordi.sourceforge.net/>
- Three back-ends supported
 - TRREE (efficient reasoning t with materialization)
 - RDBMS (efficient integration of JDBC sources)
 - YARS (efficient distribution of large RDF graphs)
- Three specialized data services (plug-ins)
 - WSMO4RDF: support for storage & reasoning with WSML
 - SAM: semantic annotation storage and management
 - DualRDF: RDF spreading activation and priming

Outline

- RDF models
- Architecture – Sesame, TRREE, ORDI, OWLIM
- ORDI
- **OWLIM**
 - Reasoning support
 - Versions, Scalability & Performance
- WP5 notes

Rule-Based Inference (Materialization)

Rules:

```
<C1, subclass, C2>  
<C2, subClass, C3>  
=> <C1, subClass, C3>
```

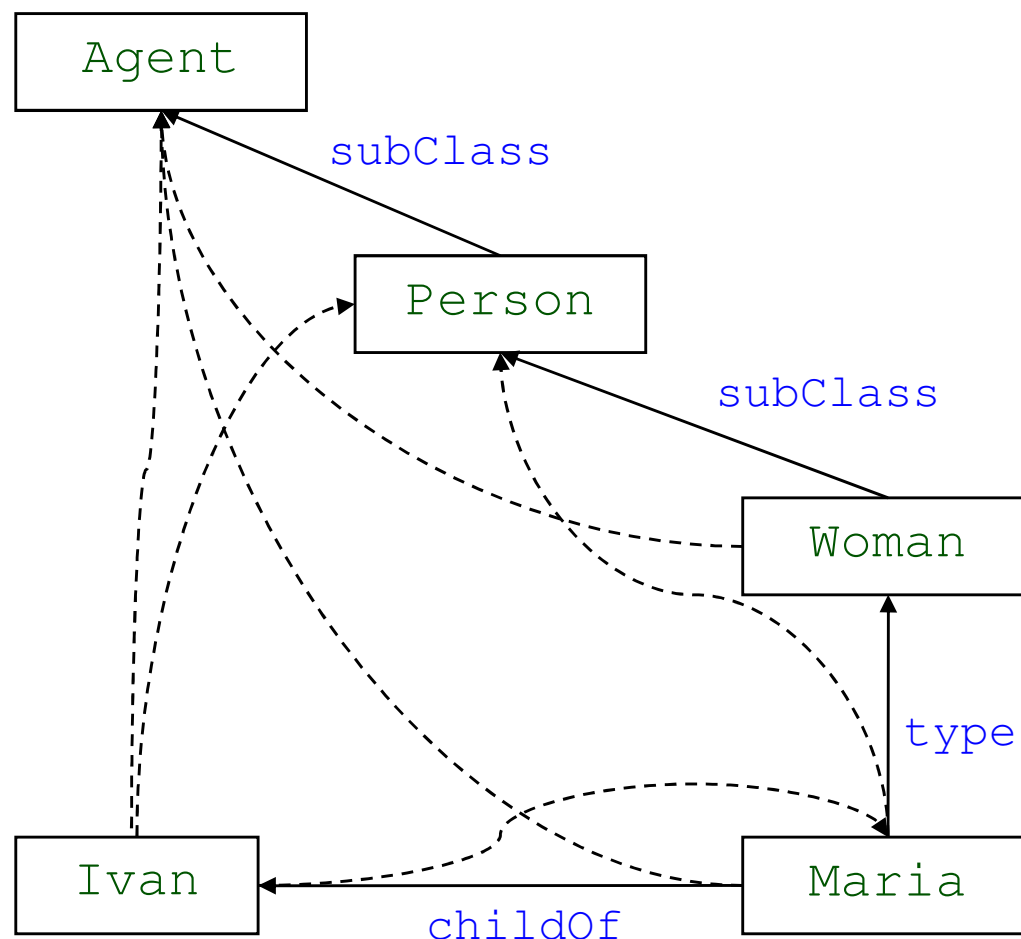
```
<I, type, C1>  
<C1, subclass, C2>  
=> <I, type, C2>
```

```
<I1, P1, I2>  
<P1, range, C2>  
=> <I2, type, C2>
```

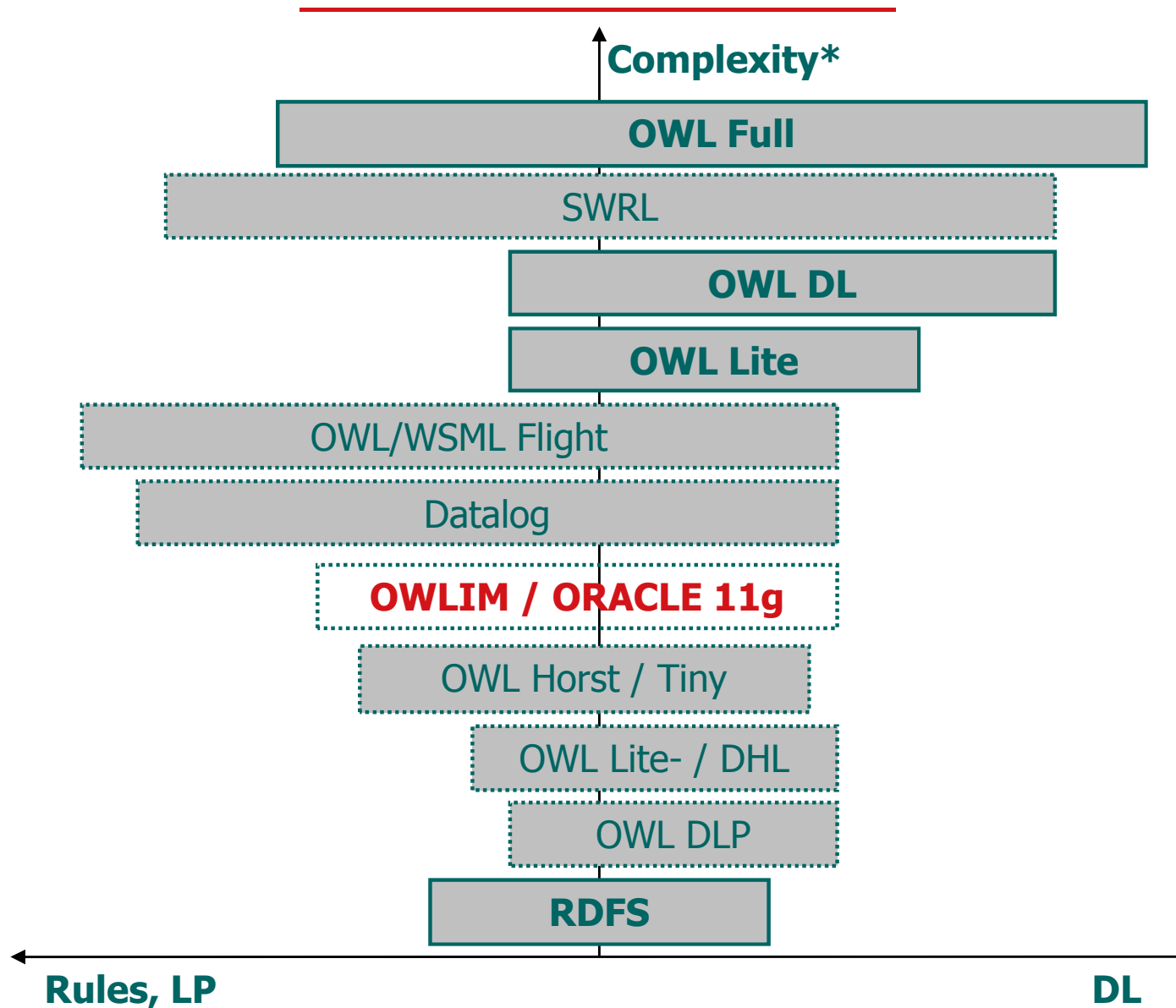
```
<P1, inverseOf, P2>  
<I1, P1, I2>  
=> <I2, P2, I1>
```

Property definition:

```
<childOf, range, Person>  
<childOf, inverseOf, parentOf>
```



Naïve OWL Fragments Map



Semantics Supported by OWLIM

- The **ruleset** parameter allows for switching between 4 predefined inference modes:
 - **owl-max** – the most expressive set (see the next slides);
 - **owl-horst** – a set similar to the one defined in [Horst05]:
 - It is sufficient to pass the LUBM benchmark correctly;
 - Similar to what was defined as OWL-Tiny at SWAD-Europe'03
 - **rdfs** – the standard RDF(S) semantics;
 - **empty** – as an RDF store without any inference.
- The **partialRDFS** parameter allows switching on/off an optimization in the RDFS and OWL support

Outline

- RDF models
- Architecture – Sesame, TRREE, ORDI, OWLIM
- ORDI
- **OWLIM**
 - Reasoning support
 - **Versions, Scalability & Performance**
- WP5 notes

Versions and Features

- There are two versions: **SwiftOWLIM** and **BigOWLIM**
 - Both using TRREE, but different versions
 - The same inference and semantics (rule-compiler, etc)
- SwiftOWLIM is good for experiments and **medium-sized data**
 - Extremely fast loading of data (incl. inference, storage, etc.)
- BigOWLIM is designed to handle **huge volumes of data** and **intensive querying**
 - Query optimizations ensure much faster query evaluation
 - Scales much better, having lower memory requirements

Versions and Features (II)

| | SwiftOWLIM | BigOWLIM |
|---|---|--|
| Scale (Mill. of explicit statem.) | 10 MSt, using 1.6 GB RAM 100 MSt , using 16 GB RAM | 130 MSt, using 1.6GB 1068 MSt , using 12GB |
| Processing speed (load+infer+store) | 30 KSt/s on notebook 200 KSt/s on server | 5 KSt/s on notebook 60 KSt/s on server |
| Query optimization | No | Yes; much faster |
| Persistence | Back-up in N-Triples | Binary files, allowing instant initialization |
| Efficient owl:sameAs | No | Yes |
| Licence and Availability | Open-source under LGPL; Uses SwiftTRREE that is free, but not open-source | Commercial Evaluation copies provided on request |

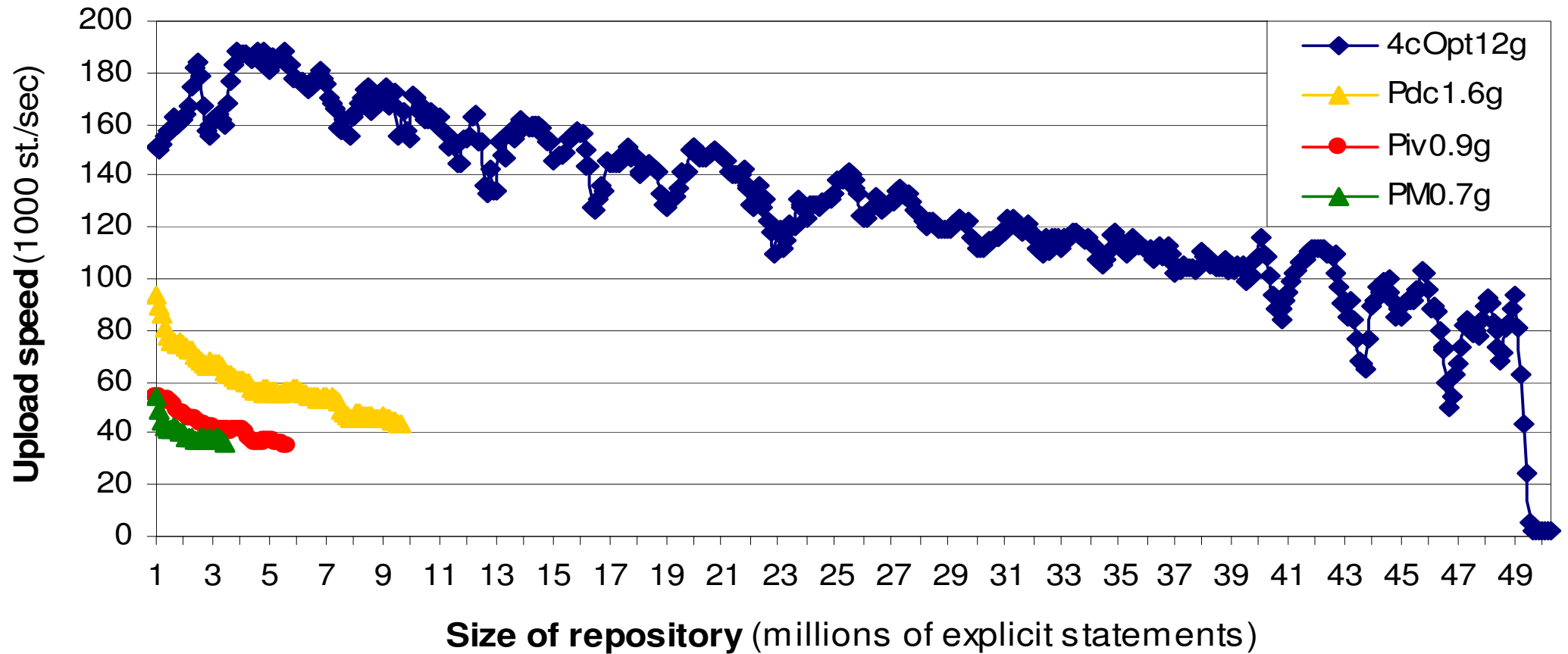
BigOWLIM and SwiftOWLIM versions 3.x

- **Version 3.0 is a new generation of OWLIM**
- **Based on the corresponding xTRREE v.3.0**
- **Supporting rich RDF model**
 - Native support for Named Graphs (contexts)
 - Native support for TripleSets – a mechanism for tagging and grouping quadruples (“contextualized” triples <s,p,o,ng>)
- **Compliant with Sesame 2.0**
- **Supporting SPARQL (thanks to Sesame)**

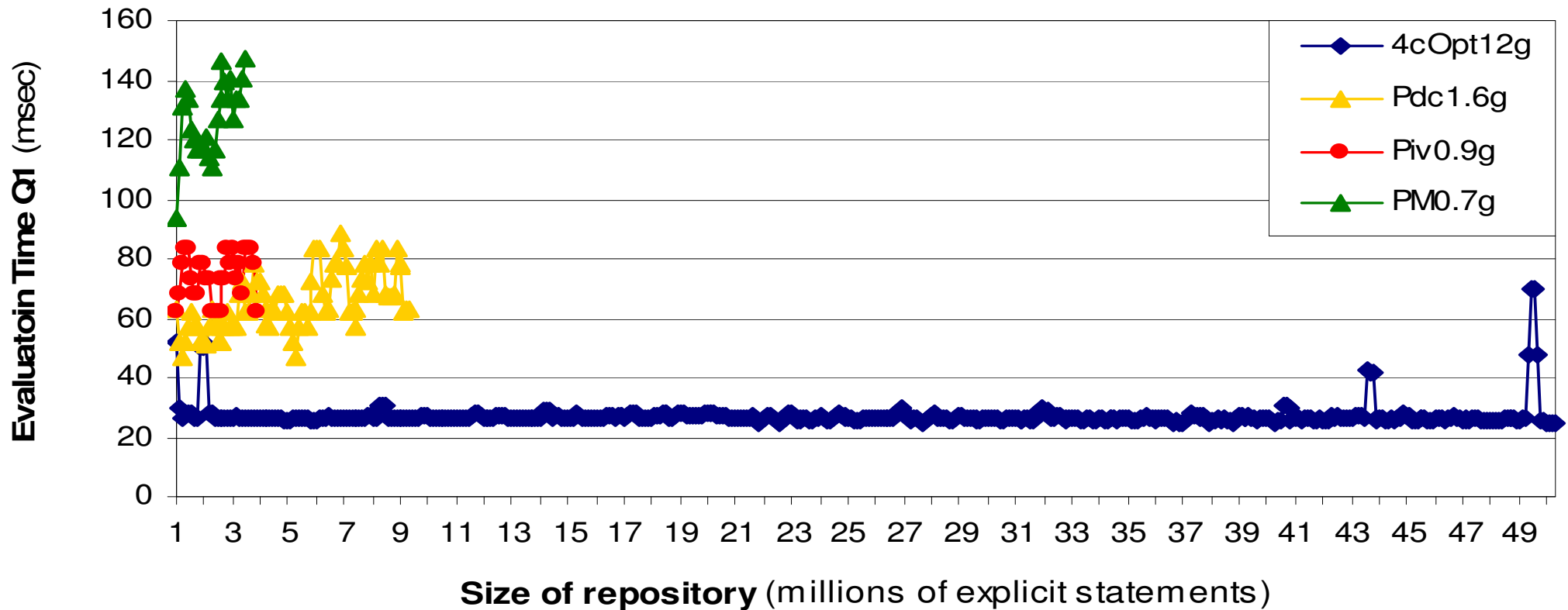
City Benchmark

- The repository is **pre-populated with about 500k explicit statements**:
 - a real ontology (PROTON) and
 - a knowledge base – the small version of KIM’s World Knowledge Base;
- **Synthetic descriptions of cities** are added incrementally,
 - each transaction adds one city described in about **10k statements**.
- **Interlinking to the non-synthetic part**:
 - the cities are linked to real provinces (randomly chosen from the WKB)
 - Ten synthetic organizations are created and “located” into each city;
 - 38 persons are created and settled within each of the organizations;
 - This way WKB is extended with LDAP-like data in realistic manner.
- A couple of test queries (in SeRQL) are evaluated after the addition of each 10 cities (i.e. after each new 100k statements).

OWLIM Performance: Upload and Inference

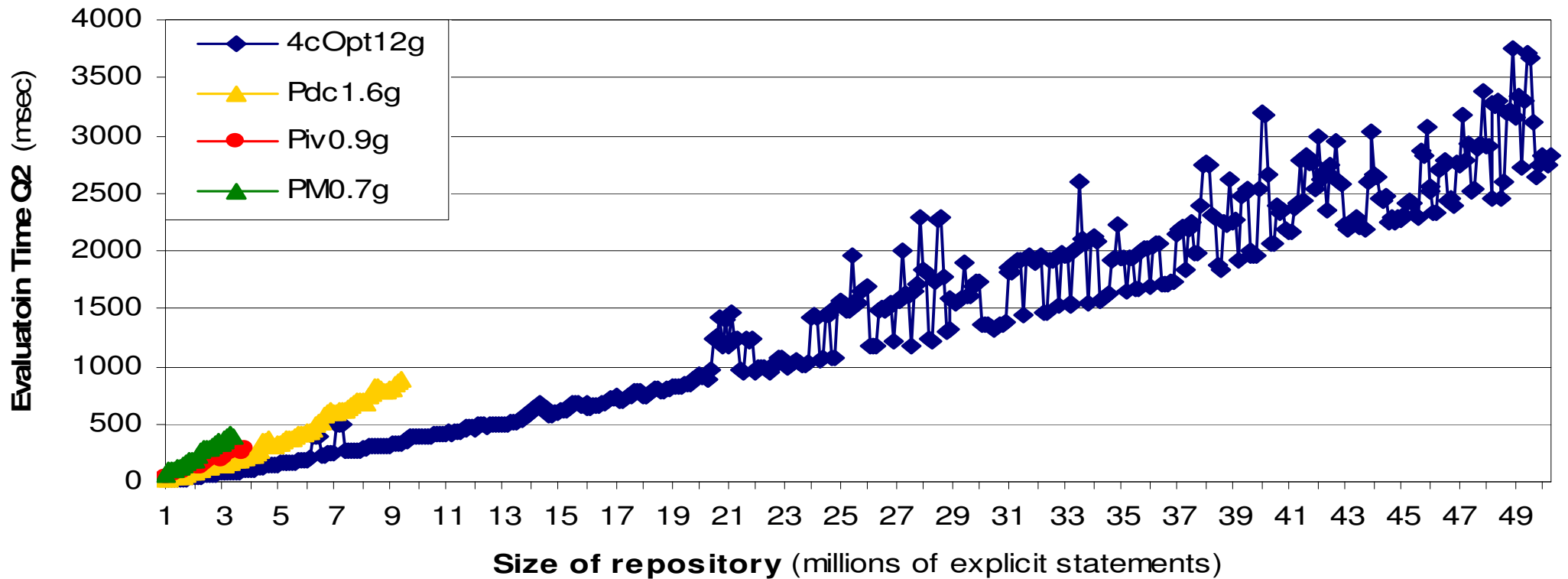


OWLIM Performance: Query Answering



- Q1: Pattern of 11 statement-joins
- Fixed small resultset – retrieval time close to 0
- The query evaluation time is almost constant

OWLIM Performance: Query Answering (II)



- Q2: Pattern of 12 statement-joins and LIKE “*xyz*” literal constraint
- Large result set which grows linearly with the repository

The LUBM Benchmark

- The **Lehigh University Benchmark (LUBM)** is an outstanding OWL repository benchmark:
 - <http://swat.cse.lehigh.edu/projects/lubm/>
- Synthetically generated datasets
 - On top of a fixed OWL ontology of “university organization”
 - The complexity is lower than OWL Horst; beyond OWL DLP
- There are 14 queries, checking inference and query evaluation speed
- The biggest set available is **LUBM(50,0) of size 6.8 Mst.**
 - About 600 MBytes in about 1000 RDF/XML files

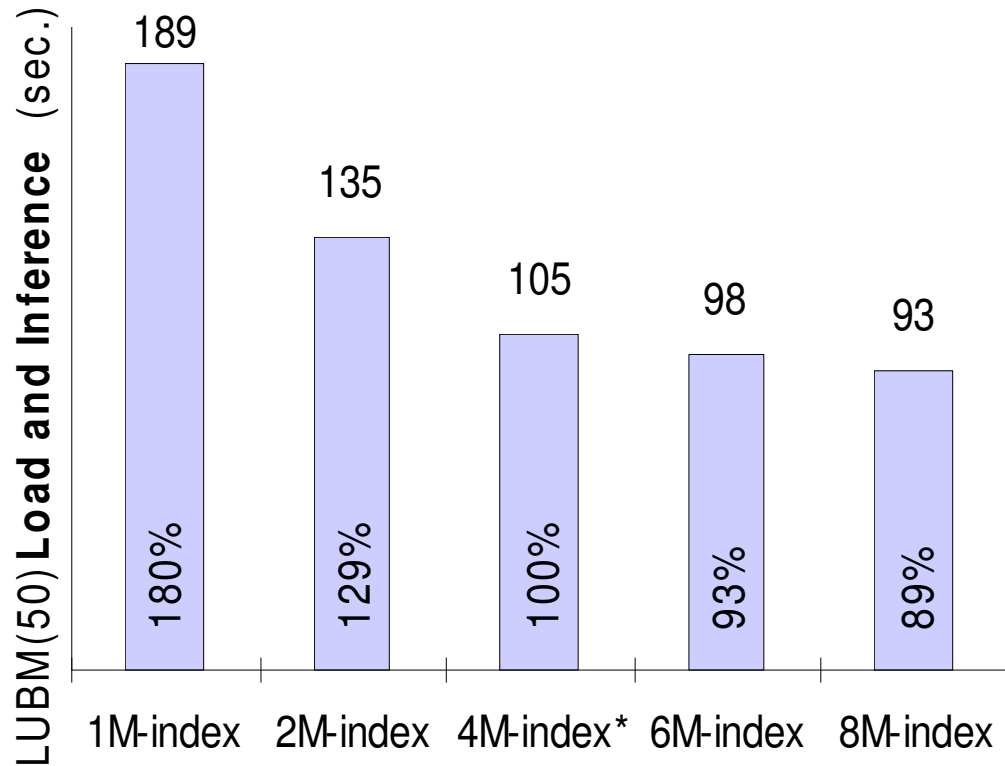
SwiftOWLIM under LUBM Benchmark

- **SwiftOWLIM loads LUBM(50,0) in 2 min. on a notebook:**
 - ORACLE 11g has the second best published results, for a system which can load it and answer the queries correctly; it does so in 8 min.
- **SwiftOWLIM can load LUBM(600,0) on server (machine 4cOpt12g) in < 65 min.**
 - About 80 Mst, 6GB input files, 12GB in N-Triples persistency
- OWLIM also passes a number of other benchmarks which require more complex reasoning: UOBM, ExQuant, etc.

OWLIM Performance Analysis

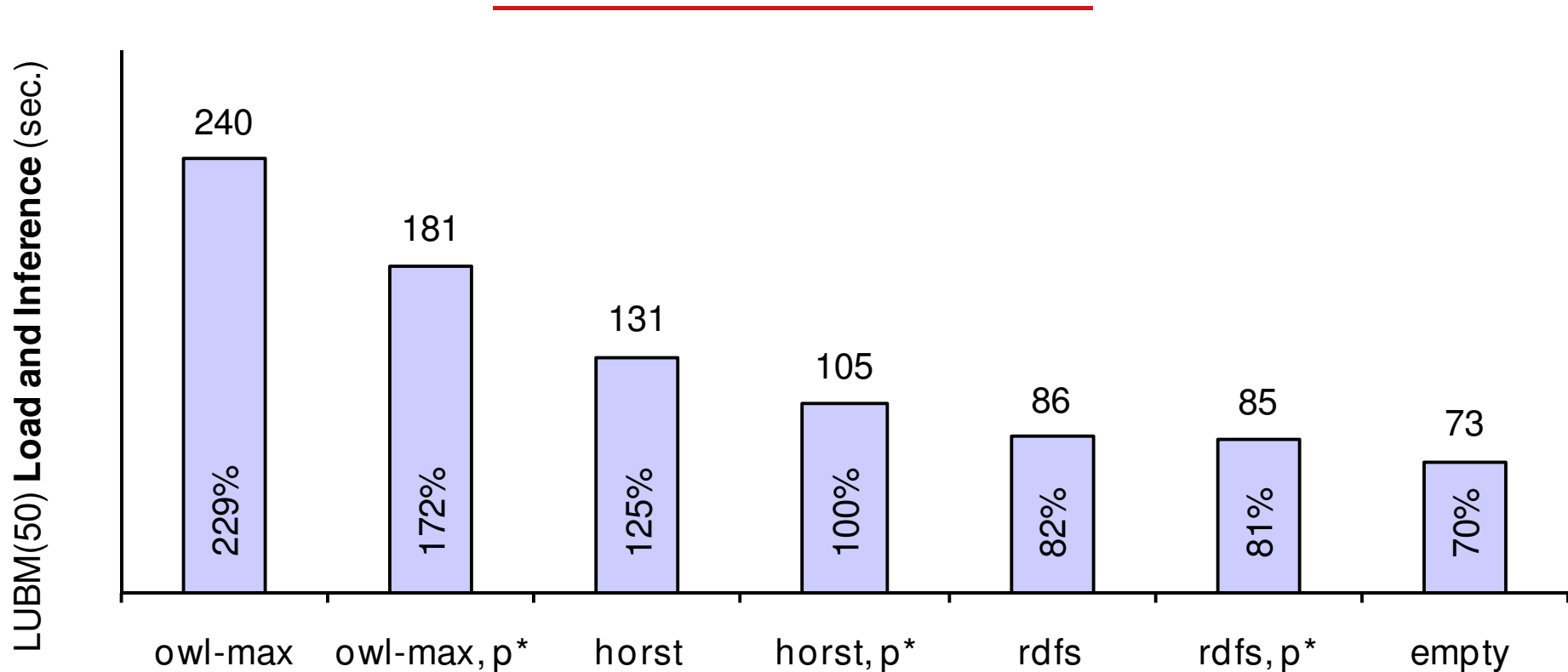
- Several tests of different configurations and parameters
- **Loading LUBM(50,0)**, which includes:
 - Parsing of the input RDF/XML files;
 - Inference – the inferred closure is calculated through forward-chaining and total materialization. This operation is not performed only when the **empty** rule-set is selected in order to force OWLIM to act as a plain RDF store;
 - Persistence (unless it is switched off) of all the data
- The **basic configuration (100% relative score)**:
 - Machine 4cOpt4g: 2xOpteron 270; 12GB of RAM;
 - 64-bit JDK 1.5.0;
 - 64-bit Suse Linux, ver. 10;
 - OWLIM with its **default settings**: noPersist=false; ruleset=owl-horst;partialRDFS=true; Index-Size: 4M entries; stackSafe=false

LUBM(50,0): The Optimal Index Size Analysis



- As expected, larger index sizes lead to better performance.
- Critical for the performance on LUBM(50,0) is the border line between 1 and 2 millions of index entries.
- Index sizes larger than the default setting (4 million entries, 64MB of RAM) seem to deliver very little improvement.

LUBM(50,0): Rule-set and Inference Mode

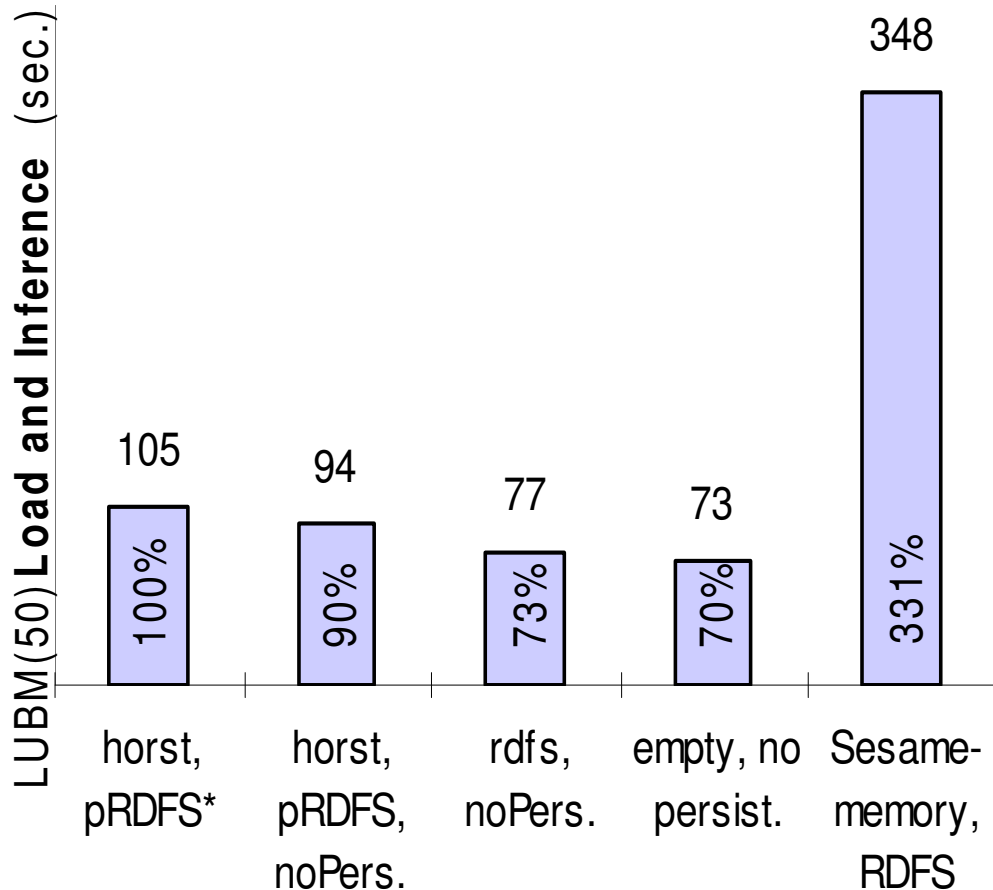


- p* above means that the partialRDFS optimizations are switched on
- Since SwiftOWLIM 2.9.1, there are few “optimizations” which partialRDFS triggers in the OWL support in rule-sets owl-horst and owl-max

LUBM(50,0): Rule-set and Inference Mode (II)

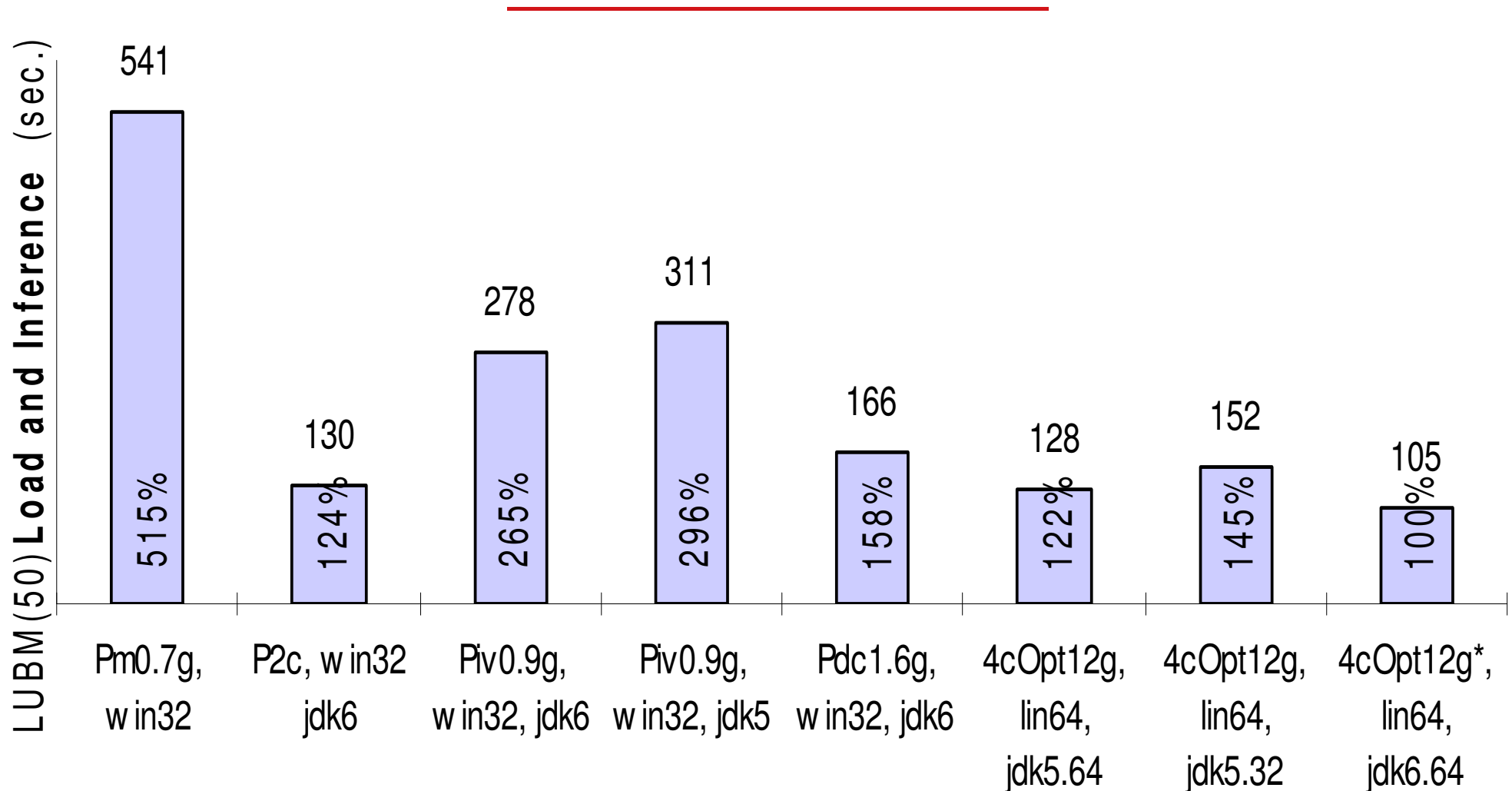
- The partialRDFS optimizations provide very little speed-up for RDFS entailment, however they speed up the OWL entailment by 25-30%
- **Inference takes about 30% of the overall processing time**
 - with the default configuration: owl-horst with partialRDFS
 - This is due to the highly optimized multi-threaded reasoning engine
- **OWL-horst reasoning is not much slower than RDFS**
- **Working with owl-max is twice slower** than owl-horst
 - This rule-set slows down OWLIM 3-4 times, compared to its “plain RDF store” setup (with the empty rule-set)

LUBM(50,0): The Impact of the Persistence



- The parsing and building of in-memory representation takes about half of the load time (see "empty, noPers.")
- The persistence takes 10% on top of the time for loading (see "horst, pRDFS, noPers.");
 - Or flat 10 sec., 10% on top of the time for parsing;
- "Sesame-memory" is four times slower than the "rdfs, noPers." setup of OWLIM.
 - It performs faster on 32-bit JDK 1.5 (305 s.); the time on 64-bit JDK was even higher.

LUBM(50,0): Different Hardware, OS, JDK



Refer to OWLIM's system documentation for analysis and comments.

RDF/OWL Representation of Wordnet

- Wordnet is the most popular **lexical knowledge base**
 - Encodes the meanings of about 100k English words
 - The meanings of the words are defined by word-senses, which related to word to a lexical concept
 - Lexical concepts are synsets, i.e. synonym sets
- Numerous **lexical semantic relations** are formally modelled:
 - Hyponymy (subsumption from a more-general term)
 - Antonymy (negation, a term with the opposite meaning)
- Standard **RDF/OWL representation of Wordnet** is available at <http://www.w3.org/2001/sw/BestPractices/WNET/wn-conversion>
 - It contains about **1.9M explicit statements** (the Full variant)
 - It is expressed in a fragment of OWL Lite
 - Total materialization derives another **6.3M implicit statements**

OWLIM loading Wordnet

- SwiftOWLIM can **load Wordnet in 123 sec. on a notebook!**
 - Loading includes: parsing, total materialization, indexing, storage;
 - Speed: **15 000 explicit st./sec.**; 65 000 st./sec. total.
 - The rule-set used is owl-max with partialRDFS=true;
- Loading with **RDFS inference is much faster**, 41 sec.:
 - Speed: **46 000 explicit st./sec.**; 200 000 st./sec. total.
- BigOWLIM needs **1 min. to load it with RDFS reasoning** on a notebook
 - It takes it 5 min. to load it with owl-max
- The best other result published is for AllegroGraph 64-bit RDFStore™
 - Loading it in 2.3 min.(at 12 970 expl. st./sec.) on a server machine
 - Much more limited inference (RDFS+/-) is performed
 - About twice slower than comparable BigOWLIM configuration on a notebook

Query Evaluation: 7 million statements

- Query evaluation data for LUBM(50) are available only for OWLIM and AllegroGraph
 - ORACLE publishes no query evaluation times for LUBM(50)
- AllegroGraph 2.0 provides incomplete results for most of the queries
- ORACLE 11g is the only engine, apart from OWLIM, which publishes correct query evaluation results for LUBM(50)
 - However, no query evaluation times are provided
- QTPS for BigOWLIM is below 12 msec. for all queries in LUBM(8000,0!)
 - It is 3 times faster than SwiftOWLIM

Query Evaluation: 7 million st. (2)

| Query No | LUBM(50): 7 MSt. | | | | | | |
|----------------|-------------------|----------------|----------|----------------|----------|------------------|-------------|
| | Number of results | BigOWLIM 0.9.6 | | SwiftOWLIM 2.9 | | AllegroGraph 2.0 | |
| | | Time (ms) | QTPR | Time (ms) | QTPR | Time (ms) | QTPR |
| 1 | 4 | 48 | 12.00 | 337 | 84.25 | 22 | 5.50 |
| 2 | 130 | 1 140 | 8.77 | 675 | 5.19 | 19410 | 19 410.00 |
| 3 | 6 | 16 | 2.67 | 1 | 0.08 | 9 | 1.50 |
| 4 | 34 | 1 | 0.01 | 1 | 0.01 | 26 | 0.74 |
| 5 | 719 | 1 | 0.00 | 1 | 0.00 | 217 | 0.32 |
| 6 | 519 842 | 531 | 0.00 | 254 | 0.00 | 14707 | 0.04 |
| 7 | 67 | 1 | 0.01 | 1 | 0.01 | 21 | 0.42 |
| 8 | 7 790 | 47 | 0.01 | 806 | 0.10 | 3057 | 0.52 |
| 9 | 13 639 | 1 969 | 0.14 | 2 610 | 0.19 | 75574 | 11.59 |
| 10 | 4 | 1 | 0.13 | 1 | 0.13 | 8 | 2.00 |
| 11 | 224 | 1 | 0.00 | 1 | 0.00 | 8 | 0.80 |
| 12 | 15 | 16 | 1.07 | 60 | 4.00 | 754 | 25.13 |
| 13 | 228 | 15 | 0.07 | 1 | 0.00 | 16 | 0.50 |
| 14 | 393 730 | 360 | 0.00 | 189 | 0.00 | 5521 | 0.01 |
| Average | | | 2 | | 7 | | 1390 |

Managing UNIPROT

- UNIPROT is one of the largest and most popular biological databases
 - It is result of merging several other databases; updated on regularly
- OWL representation of UNIPROT
 - The size of the most recent version is approx. **384 M.St.**
 - The semantics of can be handled by the **owl-max** rule-set
 - Still, it is highly interconnected, which complicates the inference
- **BigOWLIM is the first engine to reason against full UNIPROT**
 - The first load in Aug'07 uncovered modelling inconsistencies, which the “producers” of UNIPROT were happy to fix
 - Loading with OWL-Max takes 23 hours, 5000 st./sec
 - Loading with RDFS reasoning took 7 hours; 14 000 st./sec

BigOWLIM Facts (based on published results)

- **BigOWLIM is the only engine which can reason with 1B st.**
 - The record was set back in May 2006
 - It is still the only one which can handle the semantics of LUBM(8000)
 - It is the only engine which have results for OWL-Horst with more than 200M St.
- **BigOWLIM is the first engine to reason against full UNIPROT**
- **BigOWLIM's query performance is at least twice better than this of any other engine**

OWLIM in Use

- OWLIM is bundled as ontology service in **GATE 4.0**
 - GATE is the most popular text-mining platform
- OWLIM is used as a semantic repository in **KIM**
 - KIM is the most popular semantic annotation and search platform
- OWLIM is used in for large scale data-integration in several **life science projects** (<http://www.ontotext.com/lifeskim>)
- **TopBraid Composer** bundles OWLIM as a reasoner
- OWLIM is currently used in more than 5 European research projects in areas ranging from media monitoring to financial data analysis and from e-Government to space computing

Outline

- RDF models
- Architecture – Sesame, TRREE, ORDI, OWLIM
- ORDI
- OWLIM
 - Reasoning support
 - Versions, Scalability & Performance
- WP5 notes

Storage as Integration

- Think of “blackboard systems” and “space computing”
- Common storage can be used for communication
- This concept was already developed in project TripCom
 - RDF-based space (triples vs. tuples)
- ORDI is the storage layer of TripCom
- The triple-space modeling and management approach can be adapted

WP5 Notes

- We should have a first version of the architecture soon
 - But “D5.3.2 Overall LarKC architecture and design v1, M18”
- **Storage & retrieval** (relevant to WP2 and WP3)
 - Data-models, knowledge models, data-integration
- **Reasoning** (most relevant to WP4)
 - Method selection
 - Problem de-composition, workflow
- **Execution environment**
 - Parallelization
 - Mapping and deployment of components to hardware nodes

WP5 Notes (II)

- Immediate tasks:
 - Determine languages / data formats / data models
 - Determine APIs
 - Storage
 - Reasoning plug-ins
- ORDI can serve as storage layer
 - Data integration
 - Query evaluation
 - Light-weight reasoning
 - There is already a simple and generic API