Making the Most of your Triple Store:
Query Answering in OWL 2 Using an RL Reasoner

Yujiao Zhou [1] Bernardo Cuenca Grau [1] Ian Horrocks [1]
Zhe Wu [2] Jay Banerjee [2]

[1]University of Oxford

[2]Oracle Corporation

August 30, 2013

RDF & OWL

Resource Description Framework (RDF)

- The RDF is a family of W3C specifications.
- The RDF data model is based on making statements about resources in the form of subject-predicate-object expressions.

Web Ontology Language (OWL)

- OWL is endorsed by W3C as a family of knowledge representation languages.
- It allows to illustrate schema information upon RDF triples.

Datalog

We will use datalog languages as intermediate representations of ontologies. Datalog languages are subsets of first-order logic.

- Datalog

$$B_1(\vec{x}) \wedge \ldots \wedge B_m(\vec{x}) \rightarrow A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x})$$

- Datalog$^{\pm,\vee}$

$$B_1(\vec{x}) \wedge \ldots \wedge B_m(\vec{x}) \rightarrow \bigvee_i \exists \vec{y}_i \varphi(\vec{x}, \vec{y}_i)$$

where φ_i is a conjunction of atoms with free variables $\vec{x} \cup \vec{y}$.

Objective

Conjunctive Query Answering
in OWL 2 Ontologies and Large Data Sets
using scalable OWL 2 RL reasoners

Ontologies & Data Sets[1]

- An *ontology* $\mathcal{O}$ consists of a set of schema axioms.
- A *data set* $\mathcal{D}$ consists of a set of data assertions.
  We assume a data set only contains atomic class assertions $A(c)$ or atomic property assertions $r(a, b)$.

[1] Here we slightly deviated from W3C standard.

Running Example

A ontology $\mathcal{O}_{ex}$ consists of the following axioms.

$-$*EquivalentClasses*(Student *UnionOf*(GradStudent UndergradStudent))

$-$*SubClassOf*(

SomeValuesFrom(*InverseOf*(hasStudent) College)

UndergradStudent)

$-$*SubClassOf*(Student *SomeValueFrom*(takes Course))

A data set $\mathcal{D}_{ex}$ consists of the following facts.

GradStudent(Alice), College(New), hasStudent(New, Bob)

Query Languages

- SPARQL is the standard query language for RDF.
- The *conjunctive query* is a restricted form of first order logic.
- ...

The following an example query asks for all the graduate students that take a same course as Alice.

$$Q_{\mathrm{ex}}(x) := \ \mathrm{GradStudent}(x) \wedge \mathrm{takes}(x, y) \wedge \mathrm{takes}(\mathrm{Alice}, y)$$

Difference:

In SPARQL semantics, y should be binded to a named individual in $\mathcal{O}_{\mathrm{ex}}$ or $\mathcal{D}_{\mathrm{ex}}$, whereas y is existentially quantified in conjunctive queries.

Our goal is to deal with conjunctive queries.

Query Answers

Given an ontology $\mathcal{O}$ and a data set $\mathcal{D}$ and a conjunctive query $Q(\vec{x}) := \varphi(\vec{x}, \vec{y})$, $\mathsf{cert}(Q, \mathcal{O}, \mathcal{D})$ is defined as the *answer* to Q w.r.t. $\langle \mathcal{O}, \mathcal{D} \rangle$, which satisfies

$$\vec{a} \in \mathsf{cert}(Q, \mathcal{O}, \mathcal{D}) \text{ iff } \mathcal{O} \cup \mathcal{D} \models Q(\vec{a})$$

Example

$$\begin{aligned}
\mathcal{O}_{\text{ex}} = \{ & \textit{EquivalentClasses}(\text{Student } \textit{UnionOf}(\text{GradStudent UndergradStudent})), \\
& \textit{SubClassOf}(\\
& \quad \textit{SomeValuesFrom}(\textit{InverseOf}(\text{hasStudent}) \text{ College}) \\
& \quad \text{UndergradStudent}), \\
& \textit{SubClassOf}(\text{Student } \textit{SomeValueFrom}(\text{takes Course}))\} \\
\mathcal{D}_{\text{ex}} = \{ & \text{GradStudent}(\text{Alice}), \text{College}(\text{New}), \text{hasStudent}(\text{New}, \text{Bob})\} \\
Q_{\text{ex}}(x) = & \text{GradStudent}(x) \wedge \text{takes}(x, y) \wedge \text{takes}(\text{Alice}, y)
\end{aligned}$$

$$\text{cert}(Q_{\text{ex}}, \mathcal{O}_{\text{ex}}, \mathcal{D}_{\text{ex}}) = \{\text{Alice}\}$$

Since Alice is a graduate student, she is a student and thus takes some courses. So Alice herself is an answer the query Q_{ex}. From $\mathcal{O}_{\text{ex}}$ and $\mathcal{D}_{\text{ex}}$, we can not infer that Bob is graduate student. so Bob is not an answer to Q_{ex}.

Query Answering in Ontologies

- QA in full OWL 2 is of high computational complexity
- Lightweight OWL profiles

  Off-the-shelf scalable OWL 2 RL reasoners:
  Oracle's RDF Semantic Graph, OWLim . . .

OWL 2 RL

+ A large fragment of OWL 2 closely connected to datalog
+ Scalable query answering by materialisation (PTime)
– Restrictions on expressivity
  - *Disjunctive axioms*

    SubClassOf (Student *UnionOf* (GradStudent UndergradStudent))

  - *Existential axioms*

    SubClassOf (Student *SomeValuesFrom*(takes Course))

To Overcome the Expressivity Restrictions

– Full-fledged OWL 2 reasoner
    - High computation complexity.
    - The scalability of such systems falls far short of that exhibited by RL reasoners.

+ To approximate the query answers.

Lower Bounds

An bonus of OWL 2 RL reasoners is that in practice they are capable to process an arbitrary OWL 2 ontology as they ignore (parts of) axioms outside OWL 2 RL.

$$\mathsf{rl}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{cert}(Q, \mathcal{O}, \mathcal{D})$$

- Soundness guarantee
- Lower bound of the query answers.

Lower Bounds

An bonus of OWL 2 RL reasoners is that in practice they are capable to process an arbitrary OWL 2 ontology as they ignore (parts of) axioms outside OWL 2 RL.

$$\mathsf{rl}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{cert}(Q, \mathcal{O}, \mathcal{D})$$

- Soundness guarantee
- Lower bound of the query answers.

How INCOMPLETE are the answers computed by RL reasoners???

Lower Bounds

An bonus of OWL 2 RL reasoners is that in practice they are capable to process an arbitrary OWL 2 ontology as they ignore (parts of) axioms outside OWL 2 RL.

$$\mathsf{rl}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{cert}(Q, \mathcal{O}, \mathcal{D})$$

- Soundness guarantee
- Lower bound of the query answers.

How INCOMPLETE are the answers computed by RL reasoners???

$$\mathsf{rl}(Q_{\mathsf{ex}}, \mathcal{O}_{\mathsf{ex}}, \mathcal{D}_{\mathsf{ex}}) = \emptyset$$

Lower Bounds

An bonus of OWL 2 RL reasoners is that in practice they are capable to process an arbitrary OWL 2 ontology as they ignore (parts of) axioms outside OWL 2 RL.

$$\mathsf{rl}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{cert}(Q, \mathcal{O}, \mathcal{D})$$

- Soundn
- Lower bo   of the query answers.

Upper Bounds ???

How INCOMPLETE are the answers computed by RL reasoners???

$$\mathsf{rl}(Q_{\mathsf{ex}}, \mathcal{O}_{\mathsf{ex}}, \mathcal{D}_{\mathsf{ex}}) = \emptyset$$

Over-approximation to OWL 2 RL (Core Approach)

$$\begin{array}{ccccccc} \mathcal{O} & \rightsquigarrow & \Sigma_{\mathcal{O}} & \rightsquigarrow & \Xi(\Sigma_{\mathcal{O}}) & \rightsquigarrow & \mathcal{O}' \\ \vdots & & \vdots & & \vdots & & \vdots \\ \text{OWL 2} & & \textsc{Datalog}^{\pm,\vee} & & \textsc{Datalog} & & \text{OWL 2 RL} \end{array}$$

Step 1 An OWL 2 ontology $\mathcal{O}$ is translated into a set $\Sigma_{\mathcal{O}}$ of datalog$^{\pm,\vee}$ rules equivalently;

Step 2 *To transform $\Sigma_{\mathcal{O}}$ into datalog rules, we replace disjunctions with conjunctions and replace existentially quantified variables with fresh individuals;

$$\Xi(\Sigma_{\mathcal{O}}) \models \Sigma_{\mathcal{O}}$$

Step 3 $\Xi(\Sigma_{\mathcal{O}})$ is translated back into OWL 2 RL ontology.

Examples

$\mathcal{O}$	$\rightsquigarrow$	$\Sigma_{\mathcal{O}}$	$\rightsquigarrow$	$\Xi(\Sigma_{\mathcal{O}})$	$\rightsquigarrow$	$\mathcal{O}'$
$\vdots$		$\vdots$		$\vdots$		$\vdots$
OWL 2		$\textsc{Datalog}^{\pm,\vee}$		$\textsc{Datalog}$		OWL 2 RL

$$\begin{array}{ll}
& \textit{SubClassOf}(\text{Student } \textit{UnionOf}(\text{GradStudent UndergradStudent})) \\
\rightarrow & \text{Student}(x) \rightarrow \text{GradStudent}(x) \vee \text{UndergradStudent}(x) \\
\rightsquigarrow & \text{Student}(x) \rightarrow \text{GradStudent}(x) \wedge \text{UndergradStudent}(x) \\
\rightarrow & \textit{SubClassOf}(\text{Student } \textit{IntersectionOf}(\text{GradStudent UndergradStudent}))
\end{array}$$

$$\begin{array}{ll}
& \textit{SubClassOf}(\text{Student } \textit{SomeValueFrom}(\text{takes Course})) \\
\rightarrow & \text{Student}(x) \rightarrow \exists y\, \text{takes}(x, y) \wedge \text{Course}(y) \\
\rightsquigarrow & \text{Student}(x) \rightarrow \text{takes}(x, c) \wedge \text{Course}(c) \\
& \textit{SubPropertyOf}(r \text{ takes}) \\
\rightarrow & \textit{SubClassOf}(\text{Student } \textit{hasValue}(r\ c)) \\
& \textit{PropertyRange}(r \text{ Course})
\end{array}$$

Lower and Upper Bounds

$$\begin{array}{ccccccc} \mathcal{O} & \rightsquigarrow & \Sigma_{\mathcal{O}} & \rightsquigarrow & \Xi(\Sigma_{\mathcal{O}}) & \rightsquigarrow & \mathcal{O}' \\ \vdots & & \vdots & & \vdots & & \vdots \\ \text{OWL 2} & & \textsc{Datalog}^{\pm,\vee} & & \textsc{Datalog} & & \text{OWL 2 RL} \end{array}$$

$$\mathsf{rl}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{cert}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{rl}(Q, \mathcal{O}', \mathcal{D})$$

- The *Lower Bound* is the computed answer of Q by an RL reasoner w.r.t. $\langle \mathcal{O}, \mathcal{D} \rangle$;

Lower and Upper Bounds

$$\begin{array}{ccccccc} \mathcal{O} & \rightsquigarrow & \Sigma_{\mathcal{O}} & \rightsquigarrow & \Xi(\Sigma_{\mathcal{O}}) & \rightsquigarrow & \mathcal{O}' \\ \vdots & & \vdots & & \vdots & & \vdots \\ \text{OWL 2} & & \text{DATALOG}^{\pm,\vee} & & \text{DATALOG} & & \text{OWL 2 RL} \end{array}$$

$$\mathsf{rl}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{cert}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{rl}(Q, \mathcal{O}', \mathcal{D})$$

- The *Lower Bound* is the computed answer of Q by an RL reasoner w.r.t. $\langle \mathcal{O}, \mathcal{D} \rangle$;
- The *Upper Bound* is the computed answers of Q by an RL reasoner w.r.t. $\langle \mathcal{O}', \mathcal{D} \rangle$ if

Lower and Upper Bounds

$$
\begin{array}{ccccccc}
\mathcal{O} & \rightsquigarrow & \Sigma_{\mathcal{O}} & \rightsquigarrow & \Xi(\Sigma_{\mathcal{O}}) & \rightsquigarrow & \mathcal{O}' \\
\vdots & & \vdots & & \vdots & & \vdots \\
\text{OWL 2} & & \textsc{Datalog}^{\pm,\vee} & & \textsc{Datalog} & & \text{OWL 2 RL}
\end{array}
$$

$$\mathsf{rl}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{cert}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{rl}(Q, \mathcal{O}', \mathcal{D})$$

- The *Lower Bound* is the computed answer of Q by an RL reasoner w.r.t. $\langle \mathcal{O}, \mathcal{D} \rangle$;
- The *Upper Bound* is the computed answers of Q by an RL reasoner w.r.t. $\langle \mathcal{O}', \mathcal{D} \rangle$ if
  - $\mathcal{O}'$ is in OWL 2 RL profile;

Lower and Upper Bounds

$$\begin{array}{ccccccc} \mathcal{O} & \rightsquigarrow & \Sigma_{\mathcal{O}} & \rightsquigarrow & \Xi(\Sigma_{\mathcal{O}}) & \rightsquigarrow & \mathcal{O}' \\ \vdots & & \vdots & & \vdots & & \vdots \\ \text{OWL 2} & & \text{DATALOG}^{\pm,\vee} & & \text{DATALOG} & & \text{OWL 2 RL} \end{array}$$

$$\mathsf{rl}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{cert}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{rl}(Q, \mathcal{O}', \mathcal{D})$$

- The *Lower Bound* is the computed answer of Q by an RL reasoner w.r.t. $\langle \mathcal{O}, \mathcal{D} \rangle$;
- The *Upper Bound* is the computed answers of Q by an RL reasoner w.r.t. $\langle \mathcal{O}', \mathcal{D} \rangle$ if
  - $\mathcal{O}'$ is in OWL 2 RL profile;
  - $\mathsf{cert}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{rl}(Q, \mathcal{O}', \mathcal{D})$.

Example

The transformed rules $\Xi(\Sigma_{\text{ex}})$ should be as follows.

$$
\begin{aligned}
r_1 &: \quad \text{Student}(x) \to \text{GradStudent}(x) \wedge \text{UndergradStudent}(x) \\
r_2 &: \quad \text{GradStudent}(x) \to \text{Student}(x) \\
r_3 &: \quad \text{UndergradStudent}(x) \to \text{Student}(x) \\
r_4 &: \quad \text{hasStudent}(y, x) \wedge \text{College}(y) \to \text{UndergradStudent}(x) \\
r_5 &: \quad \text{Student}(x) \to \text{takes}(x, c) \wedge \text{Course}(c)
\end{aligned}
$$

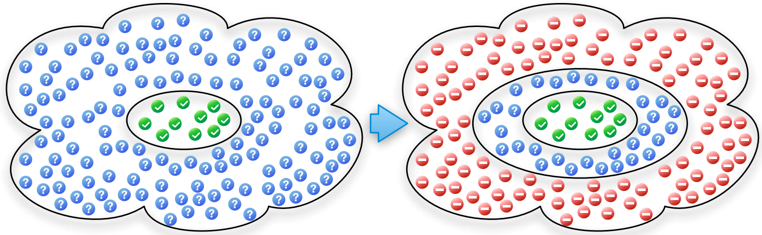
Recall that

$$
\begin{aligned}
\mathcal{D}_{\text{ex}} &= \{\text{GradStudent}(\text{Alice}), \text{College}(\text{New}), \text{hasStudent}(\text{New}, \text{Bob})\} \\
Q_{\text{ex}} &= \text{GradStudent}(x) \wedge \text{takes}(x, y) \wedge \text{takes}(\text{Alice}, y)
\end{aligned}
$$

The upper bound $\text{rl}(Q_{\text{ex}}, \mathcal{O}'_{\text{ex}}, \mathcal{D}_{\text{ex}}) = \{\text{Alice}, \text{Bob}\}$.

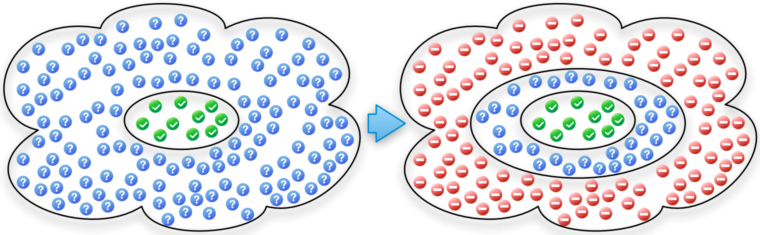
$$
\begin{aligned}
\text{hasStudent}(\text{New}, \text{Bob}), \text{College}(\text{New}) &\overset{r_4}{\rightsquigarrow} \text{UndergradStudent}(\text{Bob}) \\
\text{UndergradStudent}(\text{Bob}) &\overset{r_3}{\rightsquigarrow} \text{Student}(\text{Bob}) \\
\text{Student}(\text{Bob}) &\overset{r_1}{\rightsquigarrow} \text{GradStudent}(\text{Bob}) \\
\text{Student}(\text{Bob}) &\overset{r_5}{\rightsquigarrow} \text{takes}(\text{Bob}, c)
\end{aligned}
$$

Usages of Lower & Upper Bounds



- Lower and upper bounds coincide;

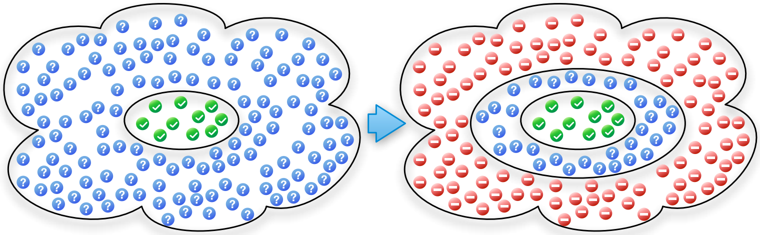
Usages of Lower & Upper Bounds



- Lower and upper bounds coincide;

  Most cases in the evaluation!

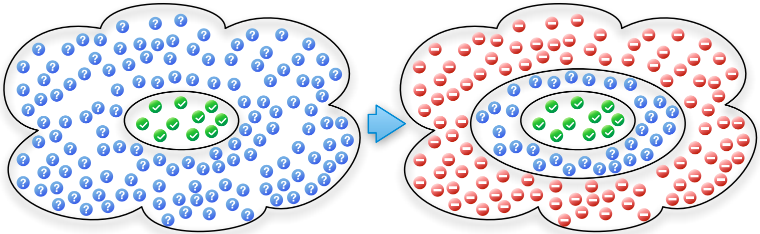
Usages of Lower & Upper Bounds



- Lower and upper bounds coincide;

  Most cases in the evaluation!
- To bound the incompleteness of an OWL 2 RL reasoner;

Usages of Lower & Upper Bounds



- Lower and upper bounds coincide;

  Most cases in the evaluation!
- To bound the incompleteness of an OWL 2 RL reasoner;
- To optimise the query answering process of an OWL 2 reasoner by checking only the answers in the gap between lower and upper bounds.

Optmisation$_1$

No matter if $\mathcal{O}' \cup \mathcal{D}$ is consistent or not,

$$\mathsf{cert}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{cert}(Q, \mathcal{O}', \mathcal{D});$$

However, when $\mathcal{O}' \cup \mathcal{D}$ is inconsistent, the upper bound is the trivial one, namely, all the tuples with appropriate arity.
In this case, if $\mathcal{O} \cup \mathcal{D}$ is consistent, removing all the rules $\Sigma_\perp$ of the form $A_1 \wedge \ldots \wedge A_m \to \perp$ from $\Xi(\Sigma_\mathcal{O})$ doesn't lose any answers, i.e.

$$\mathsf{cert}(Q, \mathcal{O}, \mathcal{D}) \subseteq \mathsf{cert}(Q, \Xi(\Sigma_\mathcal{O}) \setminus \Sigma_\perp, \mathcal{D}).$$

Optimisation$_2$

When $\mathcal{O}' \cup \mathcal{D}$ is consistent, sophisticated strategies can be applied to approximate disjunctions.

- $-$ replace disjunctions by conjunctions;
- $+$ choose one disjunct by heuristics
  - randomly choose one;
  - the simplest one;
  - the one that appear least in the body of rules.

$$
\begin{array}{rl}
 & \mathsf{Student}(x) \rightarrow \mathsf{GradStudent}(x) \vee \mathsf{UndergradStudent}(x) \\
\rightsquigarrow & \mathsf{Student}(x) \rightarrow \mathsf{UndergradStudent}(x)
\end{array}
$$

Then Bob is not in the upper bound any more since it can be inferred to be a instance of GradStudent.

Safety Harbor

"THE FOLLOWING IS INTENDED TO OUTLINE OUR GENERAL PRODUCT DIRECTION. IT IS INTENDED FOR INFORMATION PURPOSES ONLY, AND MAY NOT BE INCORPORATED INTO ANY CONTRACT. IT IS NOT A COMMITMENT TO DELIVER ANY MATERIAL, CODE, OR FUNCTIONALITY, AND SHOULD NOT BE RELIED UPON IN MAKING PURCHASING DECISION. THE DEVELOPMENT, RELEASE, AND TIMING OF ANY FEATURES OR FUNCTIONALITY DESCRIBED FOR ORACLE'S PRODUCTS REMAINS AT THE SOLE DISCRETION OF ORACLE."

RDF Semantic Graph in Oracle Database

RAC Exadata scalability

Compression partitioning

SQL Loader direct path load

Parallel load, inference, query

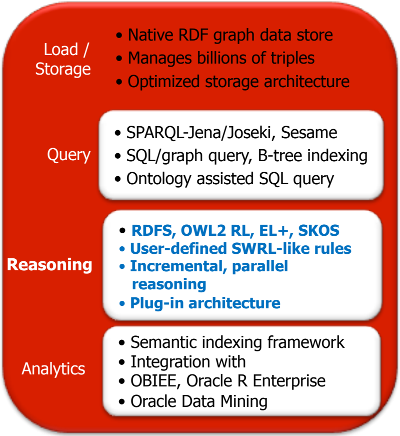
High Availability

Triple-level label security

Choice of SPARQL, SQL, or Java

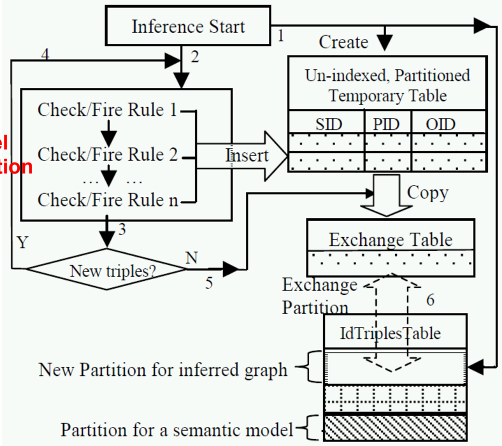
Native inference engine

Enterprise Manager



Native Inference Engine in Oracle

Parallel Execution



Leverage SQL and relational technologies (partitioning, compression)

Setup for Performance

Use a **balanced** hardware system for databases and mid-tier servers

– A single, huge physical disk for everything is **not** recommended. Multiple hard disks tied together through ASM is a good practice

– A virtual machine for multiple databases and applications is **not** recommended

– Make sure throughput of hardware components **matches** up

Component	Hardware spec	Sustained throughput
CPU core	-	100 - 200 MB/s
1/2 Gbit HBA	1/2 Gbit/s	100/200 MB/s
16 port switch	8 * 2 Gbit/s	1,200 MB/s
Fiber channel	2 Gbit/s	200 MB/s
Disk controller	2 Gbit/s	200 MB/s
GigE NIC (interconnect)	2 Gbit/s	80 MB/s*
Disk (spindle)		30 - 50 MB/s
MEM		2k-7k MB/s

Hardware specification: a dual quad core (Intel Xeon E5620) CPU, 5 SATA disks, and 40GB RAM with the operating system Linux 2.6.18.

Tips for Best Inference Performance in Oracle

- Analyze models before running inference
  – SQL: sem_apis.analyze_model(), JAVA: analyze()
- Use the right API
  – sem_apis.create_entailment()
- Pick RAW8=T for compact intermediate data storage
- Pick DOP $= \langle n \rangle$ for parallel inference
  – Require a balanced setup with multi CPU cores
- Pick INC=T for incremental inference
- Dynamic Sampling level 1 can improve inference performance
- Additional optimizations
  – Separate Tbox inference from Abox inference may reduce # of inference rounds required
  – Dynamic incremental inference: off by default, could be turned on by DYN_INC_INF=T option

Data Sets

- Lehigh University Benchmark (LUBM)
  - The LUBM describes the organisation of universities and academic departments;
  - It is outside OWL 2 RL since the appearance of existential axioms;
- University Ontology Benchmark (UOBM)
  - The UOBM is a extension of LUBM with a more complex ontology containing disjunctive axioms and negations.
- Fly Anatomy (FLY): this is a realistic and complex ontology describing the anatomy of flies and it is rich in existential axioms.

Evaluation

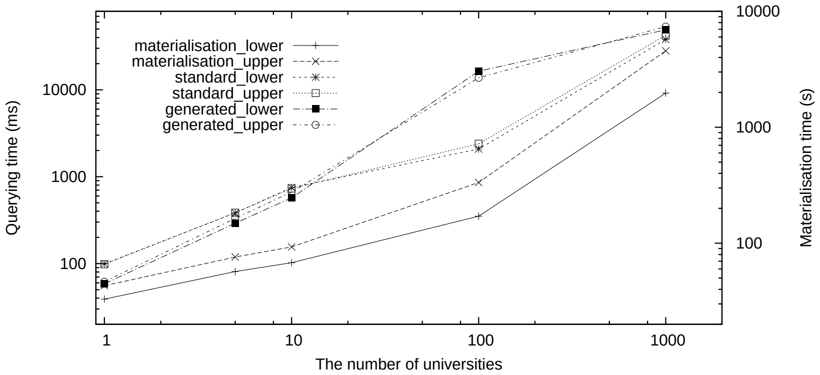
Tightness of the lower and upper bounds:

- LUBM
  - 14/14 standard queries and
    74/78 generated queries with matching bounds;
- UOBM
  - 8/14 modified LUBM queries,
    4/15 standard queries, and
    101/198 generated queries with matching bounds;
- FLY
  - 1/5 queries with matching bounds;
  - We managed to verify that the upper bounds are all tight in this case.

Transformation time: The time to over-approximate the ontologies are negligible.

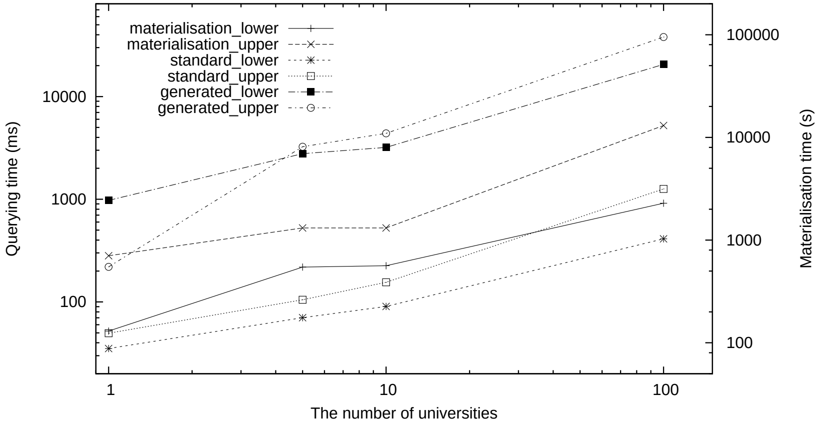
Evaluation

Materialisation and query time of LUBM:



Evaluation

Materialisation and query time of UOBM:



Thanks!

For more information: yujiao.zhou@cs.ox.ac.uk
Oracle Spatial and Graph: alan.wu@oracle.com

Appendix

Jena and Sesame Adapters

Jena and Sesame Adapters provide the following features:

– A set of easy-to-use and performant Java APIs to access Oracle database

– A standard-compliant SPARQL web service endpoint

  - SPARQL Protocol, Federated SPARQL, SPARQL update

– Data loading (RDF/XML, N-TRIPLES, N-QUADS, TriG ,Turtle) w/ long literals

– JSON output

– Oracle-specific extensions for query execution control and management

  - Timeout, abort, S2S, hints in SPARQL syntax, property path, result cache, mid-tier cache, user-defined functions

– Runs in Oracle WebLogic Server and Apache Tomcat

Native Inference Engine in Oracle: APIs

SEM_APIS.CREATE_ENTAILMENT(
entailment_name
sem_models(GraphTBox, GraphABox,),
sem_rulebases(OWL2RL),
passes,
inf_components,
Options,
)

PROOF=T to generate inference proof

Typical Usage:
- First load RDF/OWL data
- Call create_entailment to generate inferred graph
- Query both original graph and inferred data

Inferred graph contains only new triples
Saves time & resources

SEM_APIS.VALIDATE_ENTAILMENT(
sem_models((GraphTBox, GraphABox,),
sem_rulebases(OWL2RL),
Criteria,
Max_conflicts,
Options
)

Typical Usage:
- First load RDF/OWL data
- Call create_entailment to generate inferred graph
- Call validate_entailment to find inconsistencies

API: performInference, deleteInference, setInferenceOption, analyze methods in
- GraphOracleSem, DatasetGraphOracleSem (Jena Adapter)

Configure/Tune OS, Network, and Database

- Network configuration is important to performance
  – Network MTU (TCP, Infiniband) , net core rmem_max, wmem_max
- Linux OS Kernel parameters
  – shmmax, shmall, aio-max-nr, sem,
- Database parameters
  – SGA, PGA, filesystemio_options, db_cache_size, auto dop,
- Calibrate I/O performance
  – DBMS_RESOURCE_MANAGER.CALIBRATE_IO
- Gather statistics
- Run a typical workload on a typical data set
  – Check AWR report to see top waits
  – Check SQL Monitor report to find bottlenecks in SQL executions