

# Oracle Semantic Technologies Inference Best Practices with RDFS/OWL

ORACLE WHITE PAPER | SEPTEMBER 2014



## Introduction

This white paper provides inference best practices information for RDFS and OWL using Oracle Database 11g Semantic Technologies. The paper begins with recommendations for preparing to work with semantic data and Oracle Database tuning considerations. Then, it provides some best practices to inference using a RDFS rulebase, RDFS++, OWLSIF, and OWLPrime rulebases, considerations for user-defined rules with both, and an approach to handling semantics beyond OWLPrime. It concludes with LUBM8000 inference benchmark results for over 1 billion triple

## Preparing To Work With Semantic Data

The current Semantic Technologies patch set must be installed on Oracle Database before implementing step two below. The patch set is available on the OTN Semantic Technologies Software page. Please refer to Section 2 of Oracle Jena Adaptor Patch Installation Guide<sup>1</sup> for details.

It is recommended that Oracle 11gR1 RDF/OWL users use the SEM\_APIS.CREATE\_ENTAILMENT API to create the rules index, instead of SEM\_APIS.CREATE\_RULES\_INDEX. The latter exists mainly for backward compatibility.

The following simple example illustrates a flow of creating a BIGFILE tablespace with a maximum size of 300GB, initializing semantic network, creating an application table, adding one triple using DML statement, and performing inference using OWLPrime rulebase.

### Step 1:

```
SQL> -- login as SYS with DBA privilege
SQL> -- Note: you need to customize the following statement
SQL> -- based on your setup.
SQL> --
SQL> CREATE BIGFILE TABLESPACE SEMTS
DATAFILE '<path_to_datafile>/SEMTS.DBF' SIZE 512m REUSE
AUTOEXTEND ON NEXT 512M MAXSIZE 300G
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO;
```

---

<sup>1</sup> Server side patch installation guide for Oracle Jena Adaptor.

[http://www.oracle.com/technology/tech/semantic\\_technologies/documentation/jenadr\\_patch\\_ig.txt](http://www.oracle.com/technology/tech/semantic_technologies/documentation/jenadr_patch_ig.txt)

Step 2:

```
SQL> EXECUTE sem_apis.create_sem_network('SEMTS');
```

```
SQL> --
```

```
SQL> -- login as a regular user, SCOTT for example
```

```
SQL> --
```

```
SQL> CREATE TABLE owlst(id number, triple sdo_rdf_triple_s);
```

Table created.

```
SQL> EXECUTE sem_apis.create_sem_model('owlst','owlst','triple');
```

PL/SQL procedure successfully completed.

```
SQL> INSERT INTO owlst VALUES (1, sdo_rdf_triple_s('owlst',  
'http://foo.com/name/John', 'http://www.w3.org/2002/07/owl#sameAs',  
    'http://foo.com/name/JohnQ'));
```

1 row created.

```
SQL> commit;
```

```
SQL> EXECUTE sem_apis.create_entailment('owlst_idx', sem_models('owlst'),  
sem_rulebases('OWLPRIME'));
```

## Oracle Database Tuning Considerations

There is no single database configuration fits all applications. This section provides guidelines that have been found to be helpful for inference performance on a Linux PC test system with 2GB physical memory. However, your hardware, application configuration and data set may yield different results. Therefore, it is recommended to follow standard best practices for Oracle Database tuning, including testing and benchmarking in your environment. The Oracle® Database Performance Tuning Guide<sup>2</sup> is an excellent resource.

---

<sup>2</sup> Oracle® Database Performance Tuning Guide, 11g Release 1 (11.1), Part Number B28274-01.

**Maintain up-to-date statistics.** PL/SQL APIs SEM\_PERF.GATHER\_STATS or SEM\_APIS.ANALYZE\_MODEL should be invoked after a significant amount of data has been added to one of multiple semantic models that are involved in an inference. The first API collects statistics for the all models in the semantic network while the second API collects statistics on a per model basis.

**SGA/PGA.** In general, it is helpful to configure the database to have sufficient memory dedicated to SGA and PGA. One can modify the single parameter MEMORY\_TARGET to change the total memory allocated and let Oracle Database automatically set SGA and PGA separately.

On the test machine with 2GB physical memory, the following setting was used (a database restart is required afterwards) as SYS with DBA privilege. No other applications were competing for memory resources with Oracle Database processes.

```
SQL> alter system set memory_target='1600M' scope=spfile;
```

It is sometimes useful to configure SGA and PGA manually by setting the MEMORY\_TARGET initialization parameter to 0<sup>3</sup> (refer to Oracle® Database Administrator's Guide for details), especially when memory is limited but the ontology is huge. For example, the following configuration was used on the test system with just 2GB physical memory when performing OWLPrime inference in a Lehigh University Benchmark ontology benchmark (LUBM)<sup>4</sup> with 8000 universities and more than 1 billion triples. DB\_CACHE\_SIZE, which controls a minimum value for the memory pool, was given a generous allocation as shown below.

```
db_cache_size=832M  
java_pool_size=32M  
large_pool_size=16M  
shared_pool_size=160M  
sga_target=1056M  
pga_aggregate_target=352M
```

**filesystemio\_options.** On the Linux test system, it was found that by setting filesystemio\_options='SETALL' the inference performance improved significantly. This has not been verified on other platforms. A database restart is necessary after this new setting.


```
SQL> alter system set filesystemio_options='SETALL' scope=spfile;
```

**Physical I/O.** Physical I/O can be a bottleneck for inference. Oracle Database 11g inference process performs many table joins and sorts. When dealing with a huge RDF/OWL data model the inference process will cause physical I/O operations for reading source triple data and the persistence of inferred triples, and storing and swapping intermediate results.

---

<sup>3</sup> Oracle® Database Administrator's Guide, 11g Release 1 (11.1), Part Number B28310-03,

<sup>4</sup> <http://swat.cse.lehigh.edu/projects/lubm/>



Given a fixed amount of memory, one simple strategy to reduce I/O wait time is to evenly distribute I/O across multiple hard disks. In the test system, one hard disk was dedicated to hold temporary tablespace used by the database. The other hard disk stored database log files and the tablespace for the semantic network. (Automatic Storage Management can also be used to configure this.) This configuration was appropriate for the test system hardware (see Benchmark results section below) because temporary tablespace is heavily used for joins and sorts of a large amount data, given a limited physical memory.

The following command line creates a BIGFILE temporary tablespace and then instructs the database to use it as the default. Note that it is necessary to customize the command based on machine configuration and hard disk capacity.

```
SQL> create bigfile temporary tablespace tmp_ts
      tempfile '<path_to_your_datafile>'
      size 512M reuse
      autoextend on next 512M maxsize 300G
      EXTENT MANAGEMENT LOCAL
      ;
SQL> ALTER DATABASE DEFAULT TEMPORARY TABLESPACE tmp_ts;
```

## Inference Best Practices

### RDFS Rulebase

Not all RDFS rules are equally informative. For example, rule RDFS12 does not provide any new information that an application cannot readily have. To skip or deselect this rule during inference, specify 'RDFS12-' as the fifth parameter.

```
BEGIN
SEM_APIS.CREATE_ENTAILMENT(
  'example_rdfsidx',
  SEM_Models('my_example'),
  SEM_Rulebases('RDFS'),
  SEM_APIS.REACH_CLOSURE,
  'RDFS12-');
END;
/
```

One can also deselect more components for faster inference using the following value for the INF\_COMPONENTS\_IN parameter. The RDF Semantics Web site<sup>5</sup> provides detailed descriptions of rules. Consider this information before deciding to skip a rule.

'RDFS12-, RDFS4A-, RDFS4B-, RDFS6-, RDFS8-, RDFS10-, RDFS13-'

#### RDFS++, OWLSIF, and OWLPrime Rulebase

The following information about supported RDFS/OWL vocabulary constructs supplements the information in the Semantic Technologies Developer's Guide<sup>6</sup>. The set of RDFS/OWL vocabulary constructs included in each rulebase is listed as follows.

Rulebase Name	RDFS/OWL constructs included
RDFS++	all RDFS vocabulary constructs, owl:InverseFunctionalProperty, owl:sameAs
OWLSIF	<b>all RDFS vocabulary constructs,</b> owl:FunctionalProperty, owl:InverseFunctionalProperty, owl:SymmetricProperty, owl:TransitiveProperty, owl:sameAs, owl:inverseOf, owl:equivalentClass, owl:equivalentProperty, owl:hasValue, owl:someValuesFrom, owl:allValuesFrom
OWLPRIME	rdfs:subClassOf, rdfs:subPropertyOf, rdfs:domain, rdfs:range, owl:FunctionalProperty, owl:InverseFunctionalProperty, owl:SymmetricProperty, owl:TransitiveProperty, owl:sameAs, owl:inverseOf, owl:equivalentClass, owl:equivalentProperty, owl:hasValue, owl:someValuesFrom, owl:allValuesFrom,

5 <http://www.w3.org/TR/2004/REC-rdf-mt-20040210/>

6 Oracle® Database Semantic Technologies Developer's Guide 11g Release

	owl:differentFrom, owl:disjointWith, owl:complementOf
--	---

Similar to inference with RDFS rulebase, one can deselect one or multiple components if there is a clear, expected outcome of inference based on your ontology. For example, if the only interest is in class and property hierarchies and transitive relationships, then it makes sense to deselect components for owl:sameAs and owl:differentFrom.

### User-defined Rules

As documented in the Semantic Technologies Developer's Guide, it is required to specify 'USER\_RULES=T' if an inference involves user-defined rules on top of those rulebases, including RDFS, RDFS++, OWLSIF, and OWLPRIME, provided by Oracle. The following command shows an example of such an inference.

SQL> -- In the following statement, 'USER\_RULES=T' is required

```
SQL> EXECUTE sem_apis.create_entailment('owlst2_idx',
    sem_models('owlst'),
    sem_rulebases('OWLPRIME','USER_RULEBASE'),
    SEM_APIS.REACH_CLOSURE, null, 'USER_RULES=T');
```

Most user-defined rules have a single pattern in the consequent, that is, there is only one pattern in the rule head. In rare cases, a user-defined rule has more than one pattern in the consequent. However, it is recommended as more efficient to break the rule into multiple rules with single pattern in the consequent.

### Defining semantics beyond OWLPrime

Although OWLPrime has provided a very expressive subset of OWL DL constructs, there are still a few applications that require more semantics. There are two ways to accomplish this as described in the referenced white paper<sup>7</sup>:

- » Adding user-defined rules
- » Integrating with complete DL reasoners through the Oracle Jena Adaptor.

<sup>7</sup> A scalable RDBMS-based inference engine for RDFS/OWL, Oct. 2007, [http://ontolog.cim3.net/file/work/DatabaseAndOntology/2007-10-18\\_AlanWu/RDBMS-RDFS-OWL-InferenceEngine--AlanWu\\_20071018.pdf](http://ontolog.cim3.net/file/work/DatabaseAndOntology/2007-10-18_AlanWu/RDBMS-RDFS-OWL-InferenceEngine--AlanWu_20071018.pdf)

The following example illustrates how to support core semantics of owl:intersectionOf. The left column shows a simple definition of OWL concept in RDF/XML format. The right column lists three corresponding user-defined rules. Note that only the last rule is truly necessary. The first two rules can simply be added to the ontology as two schema triples.

OWL in RDF/XML format	User-defined Rules
<pre>&lt;owl:Class rdf:ID="FemaleAstronaut"&gt;   &lt;rdfs:label&gt;female astronaut&lt;/rdfs:label&gt;   &lt;owl:intersectionOf     rdf:parseType="Collection"&gt;     &lt;owl:Class rdf:about="#Female" /&gt;     &lt;owl:Class rdf:about="#Astronaut" /&gt;   &lt;/owl:intersectionOf&gt; &lt;/owl:Class&gt;</pre>	<pre>1. → :FemaleAstronaut rdfs:subClassOf :Female 2. → :FemaleAstronaut rdfs:subClassOf :Astronaut 3. ?x rdf:type :Female .    ?x rdf:type :Astronaut .    → x rdf:type :FemaleAstronaut</pre>

**Table 1. User-defined Rules for owl:intersectionOf**

It is possible to use complete DL reasoners<sup>8</sup> to compute a class subsumption tree (assume TBox size is reasonable) and apply Oracle's OWL inference on the combination of the complete class subsumption tree and ABox data. The following code snippet, based on the Oracle Jena Adaptor API, illustrates the idea.

```
Model modelOracle = ModelOracleSem.createOracleSemModel(
oracle, modelName);

// Creates an in-memory Jena model based on an Oracle RDF/OWL // model
Model model = ModelFactory.createDefaultModel();
model.add(modelOracle);
modelOracle.getGraph().close();

PelletInfGraph pelletInfGraph = new PelletInfGraph(
model.getGraph(),new PelletReasoner());
```

The above pelletInfGraph can be added back to the original Oracle OWL model. You can also refer to the Oracle Jena Adaptor examples Java code on the OTN Semantic Technologies Web page<sup>9</sup> for more details.

<sup>8</sup> Pellet: The Open Source OWL DL Reasoner. <http://pellet.owldl.com/>



## LUBM8000 (1 billion+ triples ) inference benchmark Results

Oracle has already reported scalable inference performance for Oracle Database 11g Release 1 OWLPrime and RDFS using four well-known benchmarks: UniProt, LUBM50, LUBM500, and LUBM1000 on a configuration comprising three PC systems, connected using a Giga-bit network.<sup>7,10,11</sup> The LUBM1000 benchmark ontology has over 133 million triples. The goal of this performance testing was to further test the scalability of the Oracle Database 11g inference engine by running the OWLPrime inference against the benchmark LUBM8000 ontology with over 1 billion triples.

The configuration for this performance testing was chosen to be easy for customers to replicate. It comprised one PC with a single CPU (3.0GHz) running in hyper threading mode. It had two 3.0Gb/s physical SATA hard disks. Each can hold up to 500GB of data. The memory type is DDR2 533 (PC2 4200) dual channel memory. The PC supports up to 4GB physical memory. Linux 2.6.9-34.Elsmp was the operating system used and Oracle Database 11g (11.1.0.6) for Linux was running on the PC.

Two tests were run using 2GB and 4GB physical memory configurations. In both cases, 521.7 million new triples were generated after the inference finished. The inference time-memory tradeoff is plotted in Figure 1 below. It is clear that more physical memory improves significantly the inference performance. Simply by adding 2GB more main memory, the inference time dropped from 88.5 hours to 56.7 hours<sup>12</sup>. Scalable performance is expected for future testing with more powerful hardware. Note that PGA\_AGGREGATE\_TARGET = 2000M, SGA\_TARGET = 1256M, DB\_CACHE\_SIZE = 832M, SHARED\_POOL\_SIZE = 160M, LARGE\_POOL\_SIZE = 16M, JAVA\_POOL\_SIZE = 32M, are set when memory size is 4GB.

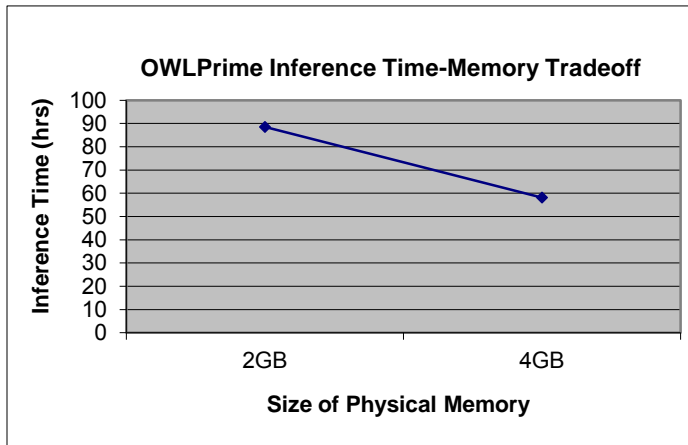
---

9 [http://www.oracle.com/technology/tech/semantic\\_technologies/sample\\_code/index.html](http://www.oracle.com/technology/tech/semantic_technologies/sample_code/index.html)

10 The semantic web for application developers, Oracle Open World, 2007  
[http://download.oracle.com/otndocs/tech/semantic\\_web/pdf/oow2007\\_semantics\\_forappdev\\_k.pdf](http://download.oracle.com/otndocs/tech/semantic_web/pdf/oow2007_semantics_forappdev_k.pdf)

11 Why, When, and How to Use Oracle Database 11g Semantic Technologies,  
[http://download.oracle.com/otndocs/tech/semantic\\_web/pdf/oow2007\\_semantics\\_techtalk\\_k.pdf](http://download.oracle.com/otndocs/tech/semantic_web/pdf/oow2007_semantics_techtalk_k.pdf)

12 On Linux platform, a bug (6687381) fix is required to avoid Oracle database reporting "Oracle process running out of OS kernel I/O resources" errors. Solaris and NT platforms should not be affected.



**Figure 1. OWLPrime Inference Performance on benchmark LUBM8000 (1billion+ triples)**





A final test was run using the 14 LUBM benchmark queries to determine the quality of query results. The number of matches resulting from these query executions is consistent with reported results by other vendors.



Oracle Corporation, World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065, USA

Worldwide Inquiries  
Phone: +1.650.506.7000  
Fax: +1.650.506.7200

CONNECT WITH US

-  [blogs.oracle.com/oracle](http://blogs.oracle.com/oracle)
-  [facebook.com/oracle](http://facebook.com/oracle)
-  [twitter.com/oracle](http://twitter.com/oracle)
-  [oracle.com](http://oracle.com)

**Hardware and Software, Engineered to Work Together**

Copyright © 2014, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0914