11gR1 OWLPrime

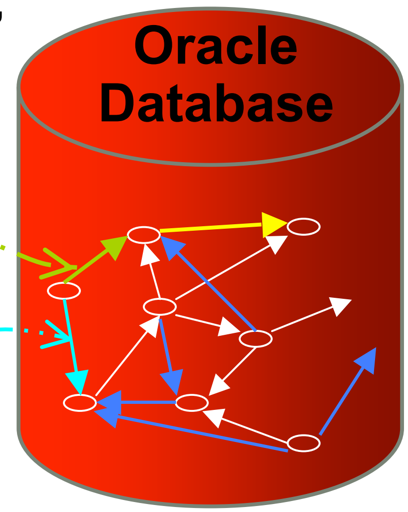
Oracle New England Development Center
Zhe Wu
alan.wu@oracle.com, Ph.D.
Consultant Member of Technical Staff
Dec 2007

Agenda

- Background
  - 10gR2 RDF
  - 11gR1 RDF/OWL
- 11gR1 OWL support
  - RDFS++, OWLSIF, OWLPrime
- Inference design & implementation in RDBMS
- Performance
- Completeness evaluation through queries

Oracle 10gR2 RDF

- **Storage**
  - Use DMLs to insert triples incrementally
    - insert into rdf_data values (…, sdo_rdf_triple_s(1, **‘<subject>’, ‘<predicate>’, ‘<object>’));**
  - Use Fast Batch Loader with a Java interface
- **Inference** (forward chaining based)
  - Support RDFS inference
  - Support User-Defined rules
  - PL/SQL API create_rules_index
- **Query** using SDO_RDF_MATCH
  - Select x, y from table(sdo_rdf_match(
    ‘(?x rdf:type :Protein) (?x :name ?y)’
    ….));
  - Seamless SQL integration
- **Shipped in 2005**



Oracle 11gR1 RDF/OWL

- **New features**
  - Bulk loader
  - Native OWL inference support (with optional proof generation)
  - Semantic operators
- **Performance improvement**
  - Much faster compared to 10gR2
    - Loading
    - Query
    - Inference
- **Shipped (Linux/Windows platform) in 2007**
- **Java API support**
  - Oracle Jena Adaptor (released on OTN) implemented HP Jena APIs.
  - Sesame (forthcoming)

Oracle 11gR1 OWL is a scalable, efficient, forward-chaining based reasoner that supports an expressive subset of OWL-DL

Why?

- **Why inside RDBMS?**
  - Size of semantic data grows really fast.
  - RDBMS has transaction, recovery, replication, security, …
  - RDBMS is ***efficient*** in processing queries.

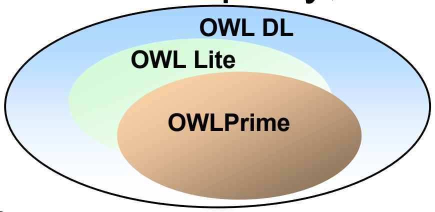
- **Why OWL-DL subset?**
  - Have to ***scale*** and support large ontologies (with large ABox)
    - Hundreds of millions of triples and beyond
    - No existing reasoner handles complete DL semantics at this scale
      - Neither Pellet nor KAON2 can handle LUBM10 or ST ontologies on a setup of 64 Bit machine, 4GB Heap[1]

- **Why forward chaining?**
    - Efficient query support
    - Can accommodate any graph query patterns

1 The summary Abox: Cutting Ontologies Down to Size. ISWC 2006

OWL Subsets Supported

- **Three subsets for different applications**
  - RDFS++
    - RDFS plus owl:sameAs and owl:InverseFunctionalProperty
  - OWLSIF
    - Based on Dr. Horst's pD* vocabulary[1]
  - OWLPrime
    - rdfs:subClassOf, subPropertyOf, domain, range
    - owl:TransitiveProperty, SymmetricProperty, FunctionalProperty, InverseFunctionalProperty,
    - owl:inverseOf, sameAs, differentFrom
    - owl:disjointWith, complementOf,
    - owl:hasValue, allValuesFrom, someValuesFrom
    - owl:equivalentClass, equivalentProperty



- **Jointly determined with domain experts, customers and partners**

1 Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary

Semantics Characterized by Entailment Rules

- RDFS has 14 entailment rules defined in the SPEC.
  - E.g. rule :

```
aaa   rdfs:domain XXX .
uuu     aaa          yyy .          \ uuu rdf:type XXX .
```

- OWLPrime has 50+ entailment rules.
  - E.g. rule :

```
aaa  owl:inverseOf  bbb  .
bbb  rdfs:subPropertyOf  ccc .
ccc  owl:inverseOf  ddd .          \ aaa rdfs:subPropertyOf ddd .
```

```
xxx   owl:disjointWith yyy .
a       rdf:type          xxx .
b       rdf:type          yyy .     \ a owl:differentFrom b .
```

- These rules have efficient implementations in RDBMS

Applications of Partial DL Semantics

- Complexity distribution of existing ontologies [1]
  - Out of 1,200+ real-world OWL ontologies
    - Collected using Swoogle, Google, Protégé OWL Library, DAML ontology library …
  - 43.7% (or 556) ontologies are RDFS
  - 30.7% (or 391) ontologies are OWL Lite
  - 20.7% (or 264) ontologies are OWL DL.
  - Remaining OWL FULL

1 A Survey of the web ontology landscape. ISWC 2006

Support Semantics beyond OWLPrime (1)

- Option1: add user-defined rules
  - Both 10gR2 RDF and 11g RDF/OWL supports user-defined rules in this form (filter is supported)

Antecedents		Consequents
?x :parentOf ?y . ?z :brotherOf ?x .	⎝	?z :uncleOf ?y

- E.g. to support core semantics of owl:intersectionOf

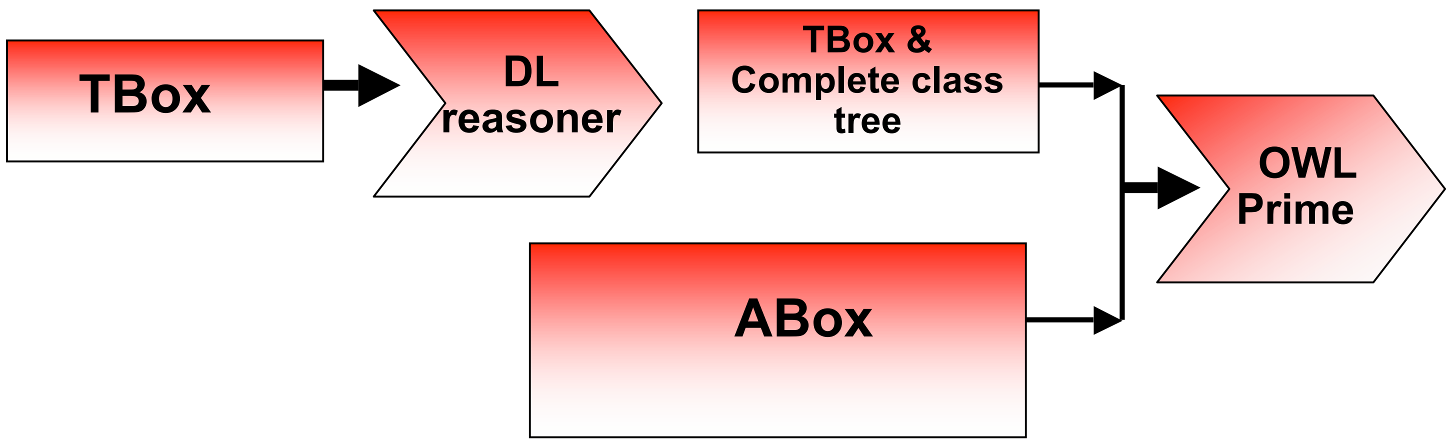
```
<owl:Class rdf:ID="FemaleAstronaut">
  <rdfs:label>chair</rdfs:label>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Female" />
    <owl:Class rdf:about="#Astronaut" />
  </owl:intersectionOf>
</owl:Class>
```

⎝

1. ⎝ :FemaleAstronaut rdfs:subClassOf :Female
2. ⎝ :FemaleAstronaut rdfs:subClassOf :Astronaut
3. ?x rdf:type :Female .
   ?x rdf:type :Astronaut . ⎝
   x rdf:type :FemaleAstronaut

Support Semantics beyond OWLPrime (2)

- **Option2: Separation in TBox and ABox reasoning**
  - TBox tends to be small in size
    - Generate a class subsumption tree using complete DL reasoners (like Pellet, KAON2, Fact++, Racer, etc)
  - ABox can be arbitrarily large
    - Use Oracle OWL to infer new knowledge based on the class subsumption tree from TBox



11g OWL Inference PL/SQL API

- **SEM_APIS.CREATE_ENTAILMENT**(
  - Index_name
  - sem_models('GraphTBox', 'GraphABox', …),
  - sem_rulebases('OWLPrime'),
  - passes,
  - Inf_components,
  - Options

)
  - Use "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('OWLPrime'),
  - Criteria,
  - Max_conflicts,
  - Options

)

Typical Usage:

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

- **Above APIs can be invoked from Java clients through JDBC**

Advanced Options

- Give users more control over inference process
  - Selective inference (component based)
    - Allows more focused inference.
    - E.g. give me only the subClassOf hierarchy.
  - Set number of passes
    - Normally, inference continue till no further new triples found
    - Users can set the number of inference passes to see if what they are interested has already been inferred
    - E.g. I want to know whether this person has more than 10 friends
  - Set tablespaces used, parallel index build
  - Change statistics collection scheme

11gR1 OWL Usage Example

- Create an application table
  - create table app_table(triple sdo_rdf_triple_s);
- Create a semantic model
  - exec sem_apis.create_sem_model('family', 'app_table','triple');
- Load data
  - Use DML, Bulk loader, or Batch loader
  - insert into app_table (triple) values(1, sdo_rdf_triple_s('family', '<http://www.example.org/family/Matt>', '<http://www.example.org/family/fatherOf>', '<http://www.example.org/family/Cindy>'));
  - …
- Run inference
  - exec sem_apis.create_entailment('family_idx',sem_models('family'), sem_rulebases('owlprime'));
- Query both original model and inferred data

```
select p, o
  from table(sem_match('(<http://www.example.org/family/Matt> ?p ?o)',
                       sem_models('family'),
                       sem_rulebases('owlprime'), null, null));
```

After inference is done, what will happen if

- *New assertions are added to the graph*

- Inferred data becomes incomplete. Existing inferred data ***will be reused*** if create_entailment API invoked again. Faster than rebuild.

- *Existing assertions are removed* from the graph

- Inferred data becomes invalid. Existing inferred data ***will not be reused*** if the create_entailment API is invoked again.

Separate TBox and ABox Reasoning

- Utilize Pellet and Oracle’s implementation of Jena Graph API
  - Create a Jena Graph with Oracle backend
  - Create a PelletInfGraph on top of it
  - PelletInfGraph.getDeductionsGraph
- Issues encountered: no subsumption for anonymous classes from Pellet inference.

```
<owl:Class rdf:ID=“Employee">
  <owl:union rdf:parseType="Collection">
    <owl:Restriction>
       <owl:onProperty rdf:resource="#reportsTo" />
       <owl:someValuesFrom>
           <owl:Class rdf:about="#Manager" />
        </owl:someValuesFrom>
    </owl:Restriction>
    <owl:Class rdf:about="#CEO" />
  </owl:union>
</owl:Class>
```

Solution: create intermediate named classes

- Similar approach applies to Racer Pro, KAON2, Fact, etc. through DIG

Soundness

- Soundness of 11g OWL verified through
  - Comparison with other well-tested reasoners
  - Proof generation
    - A proof of an assertion consists of a rule (name), and a set of assertions which together deduce that assertion.
    - Option “PROOF=T” instructs 11g OWL to generate proof

TripleID1	:emailAddress	rdf:type	owl:InverseFunctionaProperty .
TripleID2	:John	:emailAddress	:John_at_yahoo_dot_com .
TripleID3	:Johnny	:emailAddress	:John_at_yahoo_dot_com .

:John owl:sameAs :Johnny **(proof := TripleID1, TripleID2, TripleID3, “IFP”)**

Design & Implementation

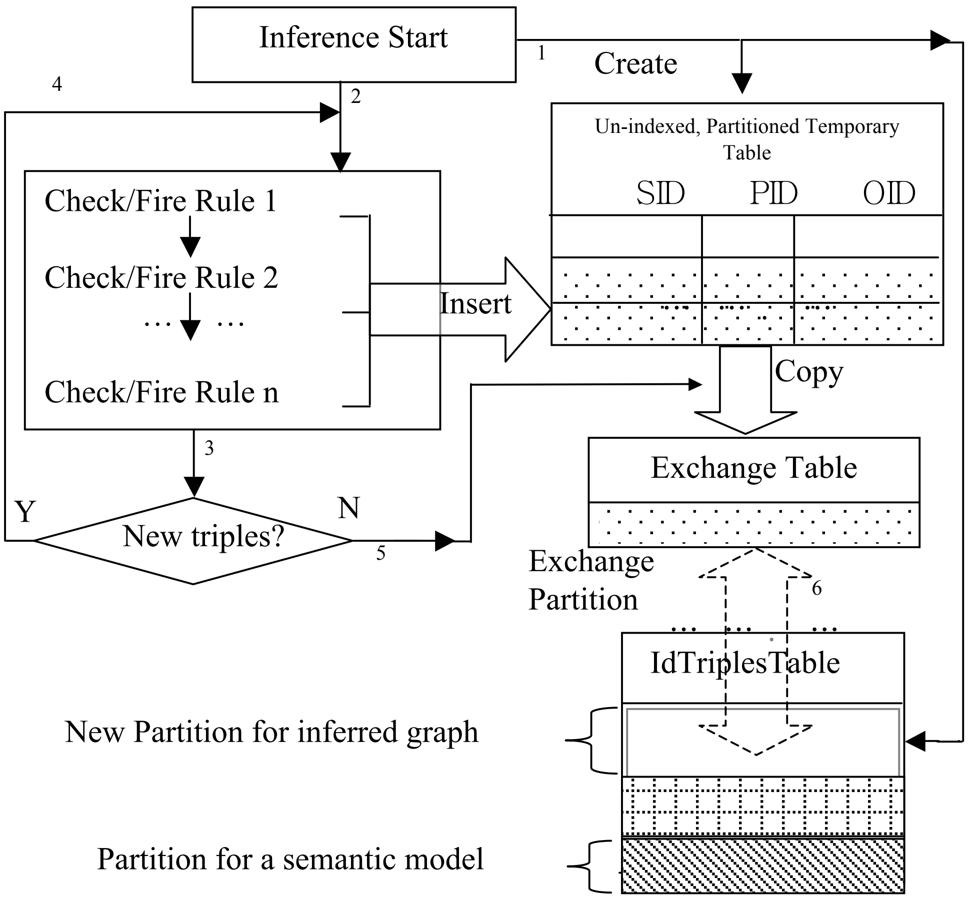
Design Flow

- Extract rules
- Each rule implemented individually using SQL
- Optimization
  - SQL Tuning
  - Rule dependency analysis
  - Dynamic statistics collection
- Benchmarking
  - LUBM
  - UniProt
  - Randomly generated test cases

TIP

- **Avoid incremental index maintenance**
- **Partition data to cut cost**
- **Maintain up-to-date statistics**

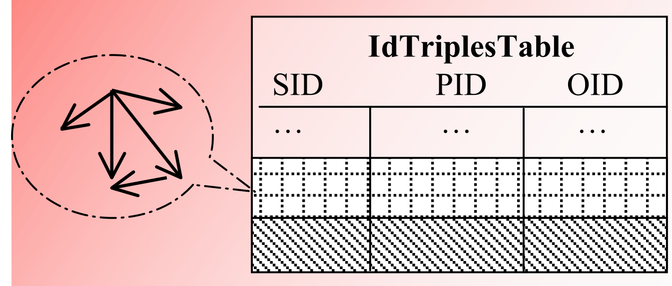
Execution Flow



Background- Storage scheme

- Two major tables for storing graph data
- VALUES table stores mapping from URI (etc) to integers
- IdTriplesTable stores basically SID, PID, OID

VALUE	ID
http://…/John	123
-·-·-·-·-·-	-·-·-·-·-·-



Entailment Rule Implementation In SQL

Example Rule

aaa owl:inverseOf bbb .
bbb rdfs:subPropertyOf ccc .
ccc owl:inverseOf ddd .

aaa rdfs:subPropertyOf ddd .

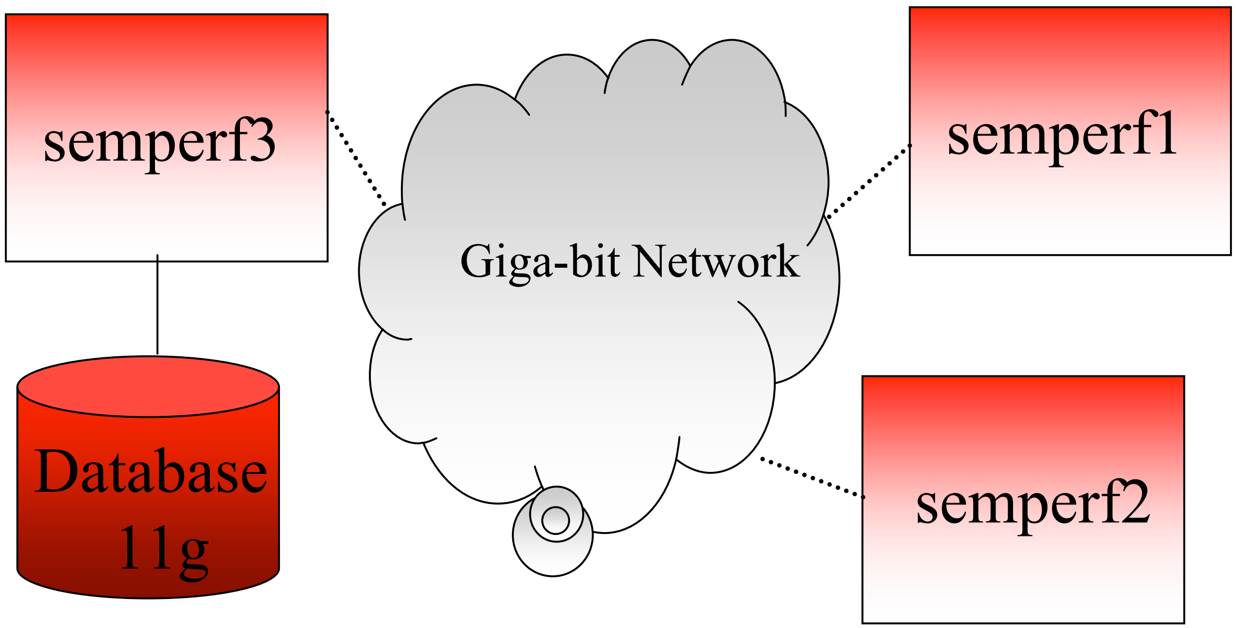
SQL Implementation

```
select distinct T1.SID sid,
                ID(rdfs:subPropertyOf) pid,
                T3.OID oid
 from <IVIEW> T1,
      <IVIEW> T2,
      <IVIEW> T3
 where T1.PID=ID(owl:inverseOf)
   and T2.PID=ID(rdfs:subPropertyOf)
   and T3.PID=ID(owl:inverseOf)
   and T1.OID=T2.SID
   and T2.OID=T3.SID
   and NOT EXISTS (
     select 1 from <IVIEW> m
     where m.SID=T1.SID
       and m.PID=ID(rdfs:subPropertyOf)
       and m.OID=T3.OID)
```

Performance Evaluation

Database Setup

- Linux based **commodity** PC (1 CPU, 3GHz, 2GB RAM)
- Database installed on machine “semperf3”



- Two other PCs are just serving storage over network

Machine/Database Configuration

- NFS configuration
    - *rw,noatime,bg,intr,hard,timeo=600,wsize=32768,rsize=32768,tcp*
- Hard disks: 320GB SATA 7200RPM (much slower than RAID). Two on each PC
- Database (11g release on Linux 32bit platform)

Parameter	Value	Description
db_block_size	8192	size of a database block
memory_target	**1408M**	memory area for a server process + memory area for storing data and control information for the database instance
workarea_size_policy	auto	enables automatic sizing of areas used by memory intensive processes
statistics_level	TYPICAL	enables collection of statistics for database self management

Tablespace Configuration

- Created bigfile (temporary) tablespaces
- LOG files located on semperf3 diskA

Tablespace	Machine	Disk	Comment
USER_TBS	semperf2	diskA	for storing user's application table. It is only used during data loading. Not relevant for inference.
Temporary Tablespace	semperf1	diskB	Oracle's temporary tablespace is for intermediate stages of SQL execution.
UNDO	semperf2	diskB	for undo segment storage
SEM_TS	semperf3	diskB	for storing graph triples

Inference Performance

Ontology (size) (after duplicate elimination)	RDFS		OWLPrime		OWLPrime + Pellet on TBox	
	#Triples inferred (millions)	Time	#Triples inferred (millions)	Time	#Triples inferred (millions)	Time
LUBM50 6.6 million	2.75	12min 14s	3.05	8m 01s	3.25	8min 21s
LUBM1000 133.6 million	55.09	7h 19min	61.25	7hr 0min	65.25	7h 12m
UniProt 20 million	3.4	24min 06s	50.8	3hr 1min	NA	NA

2.52k triples/s

6.49k triples/s

As a reference (not a comparison)

BigOWLIM ***loads, inferences, and stores*** (2GB RAM, P4 3.0GHz,)
- LUBM50 in 11 minutes (JAVA 6, -Xmx192) [1]
- LUBM1000 in 11h 20min (JAVA 5, -Xmx1600) [1]

Note: Our inference time **does not** include loading time! Also, set of rules is different.

- **Results collected on a single CPU PC (3GHz), 2GB RAM (1.4G dedicate to DB), Multiple Disks over NFS**

1 http://www.ontotext.com/owlim/OWLIMPres.pdf, **Oct 2007**

Query Answering After Inference

Ontology LUBM50 6.8 million & 3+ million inferred		LUBM Benchmark Queries						
		Q1	Q2	Q3	Q4	Q5	Q6	Q7
OWLPrime	# answers	4	130	6	34	719	393730	59
	Complete?	Y	Y	Y	Y	Y	N	N
OWLPrime + Pellet on TBox	# answers	4	130	6	34	719	519842	67
	Complete?	Y	Y	Y	Y	Y	Y	Y

Query Answering After Inference (2)

Ontology LUBM50 6.8 million & 3+ million inferred		LUBM Benchmark Queries						
		Q8	Q9	Q10	Q11	Q12	Q13	Q14
OWLPrime	# answers	5916	6538	0	224	0	228	393730
	Complete?	N	N	N	Y	N	Y	Y
OWLPrime + Pellet on TBox	# answers	7790	13639	4	224	15	228	393730
	Complete?	Y	Y	Y	Y	Y	Y	Y

Query Answering After Inference (3)

Ontology LUBM1000 133 million & 60+ million inferred		LUBM Benchmark Queries						
		Q1	Q2	Q3	Q4	Q5	Q6	Q7
OWLPrime	# answers	4	2528	6	34	719	7924765	59
	Complete?	Y	**Unknown**	Y	Y	Y	N	N
OWLPrime + Pellet on TBox	# answers	4	2528	6	34	719	10447381	67
	Complete?	Y	**Unknown**	Y	Y	Y	**Unknown**	Y

Query Answering After Inference (4)

Ontology LUBM1000 133 million & 60+ million inferred		LUBM Benchmark Queries						
		Q8	Q9	Q10	Q11	Q12	Q13	Q14
OWLPrime	# answers	5916	131969	0	224	0	4760	7924765
	Complete?	N	N	N	Y	N	**Unknown**	**Unknown**
OWLPrime + Pellet on TBox	# answers	7790	272982	4	224	15	4760	7924765
	Complete?	Y	**Unknown**	Y	Y	Y	**Unknown**	**Unknown**

For More Information

http://search.oracle.com

semantic technologies

or

http://www.oracle.com/