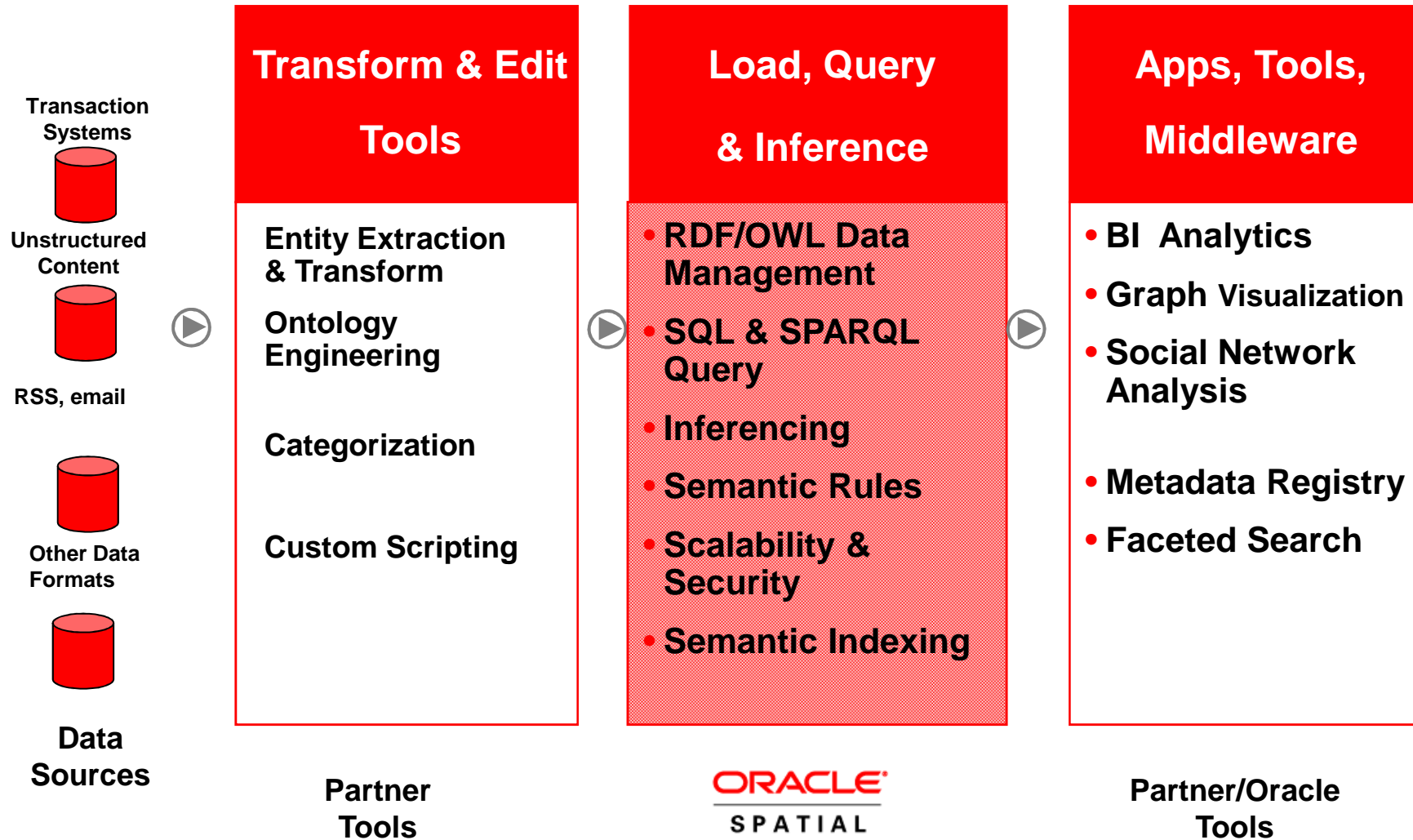


# Oracle Database 11g Semantic Technologies

Xavier Lopez, Ph.D., Director, Spatial & Semantic Technologies



# Extraction, Modeling, Reasoning & Discovery Workflow



# Semantic Technologies Customers

## In Production

### Life Sciences



11g Reference



11g Reference



11g Reference

### Defense/ Intelligence



### Education



### Telecomm

Hutchinson 3G  
Austria



### Clinical Medicine



THE UNIVERSITY of TEXAS  
HEALTH SCIENCE CENTER  
AT HOUSTON

11g Reference

### Publishing

Westlaw  
Thomson Reuters

11g Reference

ORACLE

# Semantic Technologies Customers Developing or Deploying

## Life Sciences & Clinical

 Cleveland Clinic  
Heart and Vascular Institute



Partner:  MONDECA

## Defense & Intel

**Raytheon**



## Telecomm



## Entertainment



# Oracle Database 11g RDF/OWL Capability

- Oracle 11g is the leading commercial database with native RDF/OWL data management
- Scalable & secure platform for wide-range of semantic applications
- Readily scales to ultra-large repositories (+10s billions)
- Choice of SQL or SPARQL query
- Leverages Oracle Partitioning. RAC supported
- Growing ecosystem of 3<sup>rd</sup> party tools partners

## Key Capabilities:

### Load / Storage

- Native RDF graph data store
- Manages billions of triples
- Fast batch, bulk and incremental load

### Query

- SQL: SEM\_Match
- SPARQL: via Jena plug-in
- Ontology assisted query of RDBMS data

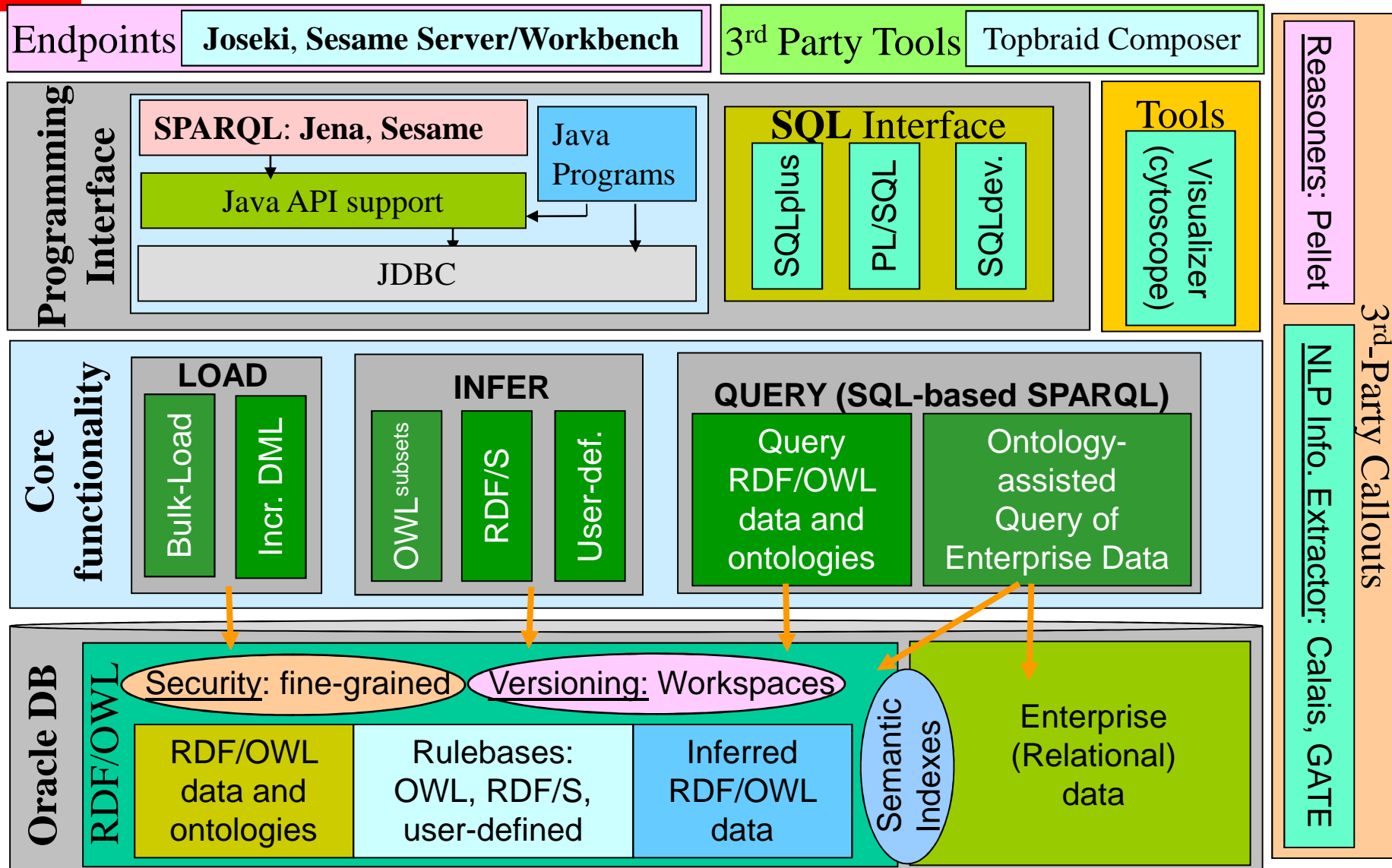
### Reasoning

- Forward chaining model
- OWL 2
- User defined rule base



ORACLE

# Architectural Overview





# Interfaces

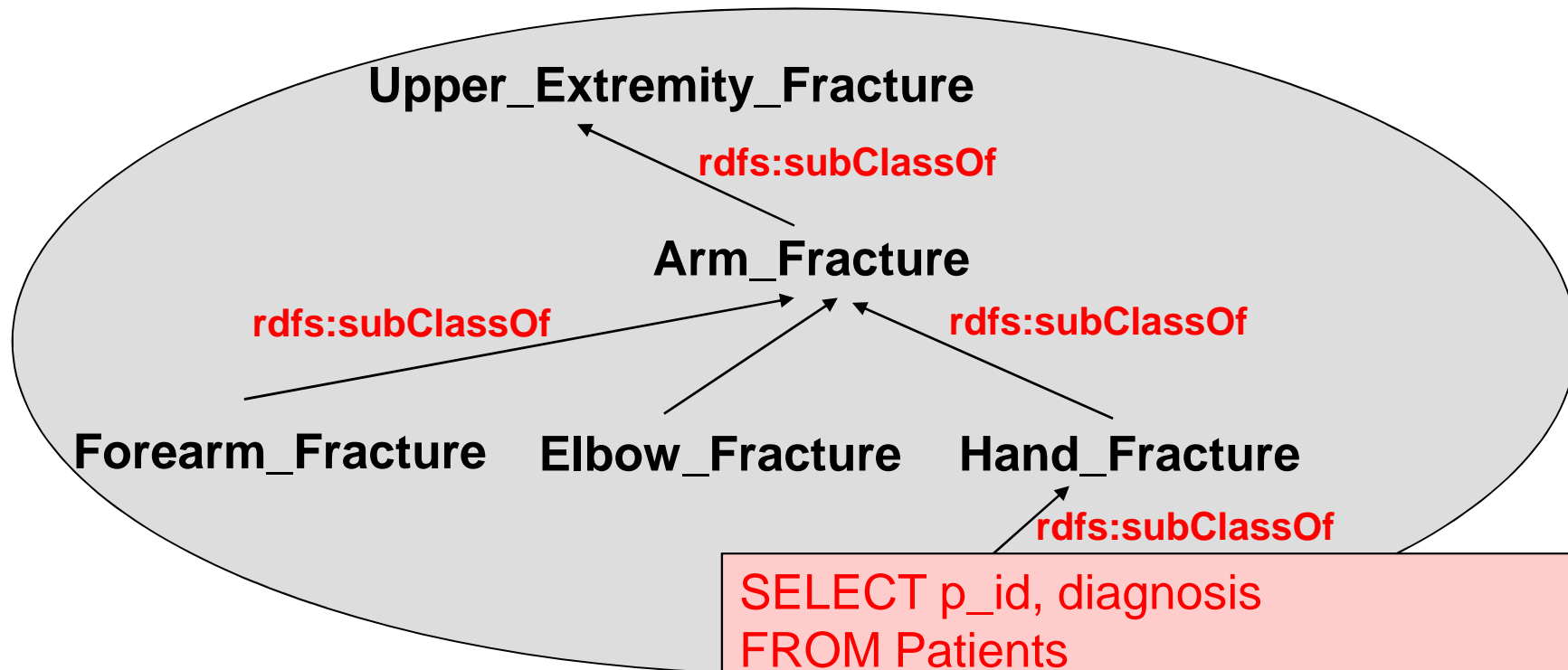
- SQL-based (SQL and PL/SQL)
- Java-based
  - Jena (using Jena Adapter from Oracle)
  - Sesame (using Sesame Adapter from Oracle)
- SPARQL Endpoints
  - Joseki
  - OpenRDF Workbench



# New 11g Semantics Features

- Strong security for Semantic Technologies
  - Security policies and data classification for RDF data
- Semantic indexing for documents
  - Semantic indexing of documents based on popular natural language tools
- Faster, more efficient reasoning to find new relationships
  - Parallel and incremental inference, owl:sameAs optimization
- Change management for collaboration
- Standards & open source support
  - SPARQL query support for Filter, Union in SEM\_MATCH table function
  - OWL: union, intersection, OWL 2 property chains, disjoint properties
  - Pellet OWL DL reasoner Integration
  - Jena Adapter (for V2.5 and above)
  - Sesame Adapter (for V2.3 and above)
  - Supports W3C SKOS & SNOMED ontologies

# Ontology-assisted Query using SQL Operators



Patients

ID	DIAGNOSIS
1	Hand_Fracture
2	Rheumatoid_Ar

```
SELECT p_id, diagnosis
FROM Patients
WHERE SEM_RELATED (
  diagnosis,
  'rdfs:subClassOf',
  'Upper_Extremity_Fracture',
  sem_models('Medical_ontology'),
  sem_rulebases('RDFS') ...
  123) = 1
AND SEM_DISTANCE(123) <= 2;
```



# Infer Semantic Data

- Native inferencing in the database for
  - RDF, RDFS, and a rich subset of OWL semantics (OWLSIF, OWLPRIME, RDFS++)
  - User-defined rules
- Forward chaining
  - New relationships/triples are inferred and stored ahead of query time
  - Removes on-the-fly reasoning and results in fast query times
- Proof generation
  - Show one deduction path



# New Release 11.2 Inference Features

- Richer semantics support including
  - owl:intersectionOf, unionOf, oneOf,
  - W3C SKOS
  - SNOMED
  - OWL2's owl:propertyChainAxiom, owl:NegativePropertyAssertion, owl:hasKey, owl:propertyDisjointWith,
  - Most OWL 2 RL/RDF rules
- Performance enhancement
  - Large scale owl:sameAs handling
  - Parallel inference
  - Incremental inference

# Using New Release 11.2 Inference Capabilities

- Enabling Parallel inference option

```
EXECUTE sem_apis.create_entailment('M_IDX',sem_models('M'),  
sem_rulebases('OWLPRIME'), sem_apis.REACH_CLOSURE, null, 'DOP=x');
```

- Where 'x' is the degree of parallelism (DOP)

- Enabling Incremental inference option

```
EXECUTE sem_apis.create_entailment ('M_IDX',sem_models('M'),  
sem_rulebases('OWLPRIME'),null,null, 'INC=T');
```

- Enabling owl:sameAs option to limit duplicates

```
EXECUTE Sem_apis.create _entailment('M_IDX',sem_models('M'),  
sem_rulebases('OWLPRIME'),null,null, 'OPT_SAMEAS=T');
```

- Enabling compact data structures

```
EXECUTE Sem_apis.create _entailment('M_IDX',sem_models('M'),  
sem_rulebases('OWLPRIME'),null,null, 'RAW8=T');
```

- Enabling SKOS inference

```
EXECUTE Sem_apis.create_entailment('M_IDX',sem_models('M'),  
sem_rulebases('SKOSCORE'),null,null...);
```

# Semantic Indexing of Documents

11.2

```
CREATE INDEX ArticleIndex
ON NewsFeed (Article)
INDEXTYPE IS SemContext
PARAMETERS ('gate_nlp')
```

Triples table with rowid references

Subject	Property	Object	rid
p:Marcus	rdf:type	rc::Person	r1
p:Marcus	:hasName	"Marcus"^^...	r1
p:Marcus	:hasAge	"38"^^xsd:...	r1
...	...	...	...

Analytical Queries  
On Graph Data

Newsfeed table


docId	Article	Source
1	Indiana authorities filed felony charges and a court issued an arrest warrant for a financial manager who apparently tried to fake his death by crashing his airplane in a Florida swamp. Marcus Schrenker, 38 ...	CNN
2	Major dealers and investors ...	NW
..		..

SemContext index on  
Article column

```
SELECT docId FROM Newsfeed
WHERE SEM_CONTAINS (Article,
{'?x rdf:type rc:Person .
 ?x :hasAge ?age .
 FILTER(?age >= 35)})=1
AND Source = 'CNN'
```



# Change Mgmt./Versioning for Sem. Data

- Manage public and private versions of semantic data in database workspaces ([Workspace Manager](#)) 
- An RDF Model is version-enabled by version-enabling the corresponding application table.
  - `exec DBMS_WM.enableVersioning (table_name => 'contracts_rdf_data');`
- RDF data modified within a workspace is private to the workspace until it is merged.
- SEM\_MATCH queries on version-enabled models are version aware and only return relevant data.
- New versions created only for changed data
- Versioning is provisioned for inference



# Jena Adapter for Oracle Database 11g Release 1

- Implements Jena's Graph/Model/BulkUpdateHandler/... APIs
- "Proxy" like design
  - Data not cached in memory for scalability
  - SPARQL query converted into SQL and executed inside DB
    - A SPARQL with just conjunctive patterns is converted into a single SEM\_MATCH query
- Allows various data loading
  - Bulk/Batch/Incremental load RDF or OWL (in N3, RDF/XML, N-TRIPLE etc.) **with strict syntax verification and long literal support**
- Integrates Oracle Database release 11g RDF/OWL with tools
  - TopBraid Composer
  - External complete DL reasoners (e.g. Pellet)



## GeoSPARQL: Handling Spatial Data in RDF

- To develop best practices for managing spatial data in RDF (OGC Specification Proposal 2009)
- To define structured vocabulary and semantics for geographic features (metadata) and relationships.
  - E.g: **ogc:dimension** property on a Spatial Object can capture the dimension of the object
- To manage geographic data as RDF terms using standard serialization formats.
  - E.g: GML captured as text with appropriate RDF literal type.
- To add the ability to answer queries involving geographic features and relationships.
  - E.g: The **ogc:touches** relationship can link two Spatial Objects in a SPARQL triple pattern.



# Performance

# Query Performance

Ontology LUBM50 6.8 million & 5.4 million inferred		LUBM Benchmark Queries						
OWLPrime & new inference components	Query	<b>Q1</b>	<b>Q2</b>	<b>Q3</b>	<b>Q4</b>	<b>Q5</b>	<b>Q6</b>	<b>Q7</b>
	# answers	4	130	6	34	719	519842	67
	Complete?	Y	Y	Y	Y	Y	Y	Y
	Time (sec)	0.05	0.75	0.20	0.5	0.22	1.86	1.71
	Query	<b>Q8</b>	<b>Q9</b>	<b>Q10</b>	<b>Q11</b>	<b>Q12</b>	<b>Q13</b>	<b>Q14</b>
	# answers	7790	13639	4	224	15	228	393730
	Complete?	Y	Y	Y	Y	Y	Y	Y
	Time (sec)	1.07	1.65	0.01	0.02	0.03	0.01	1.47

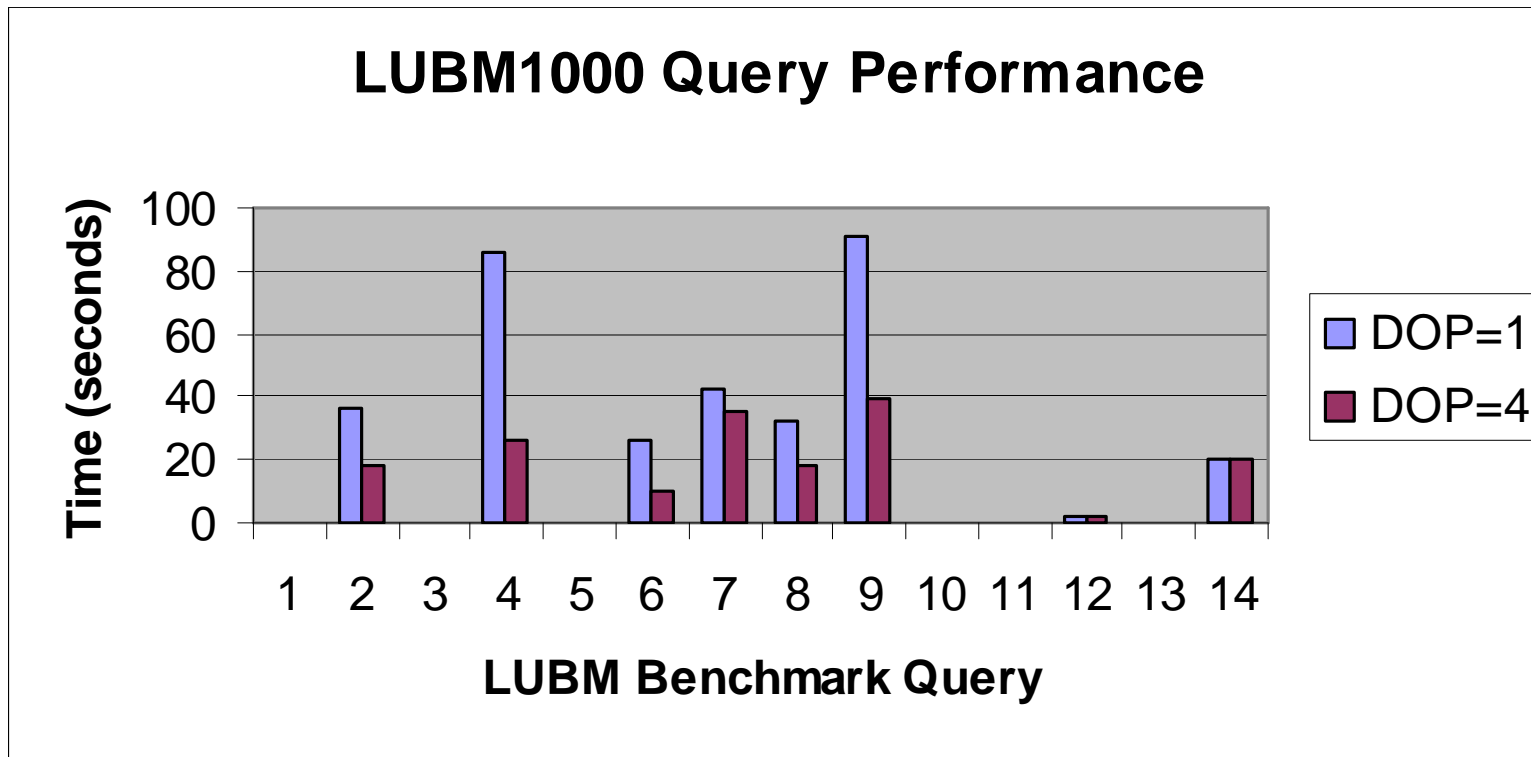
- Setup: Intel Q6600 quad-core, 3 7200RPM SATA disks, 8GB DDR2 PC6400 RAM, No RAID. 64-bit Linux 2.6.18. Average of 3 warm runs

# Inference Performance

<b>Parallel Inference</b> (LUBM8000 1.06 billion triples + 860M inferred)	<ul style="list-style-type: none"> <li>• Time to finish inference: 12 hrs.</li> <li>• <b>3.3x faster compared to serial inference in release 11.1</b></li> </ul>
<b>Parallel Inference</b> (LUBM25000 3.3 billion triples + 2.7 billion inferred)	<ul style="list-style-type: none"> <li>• Time to finish inference: 40 hrs.</li> <li>• <b>30% faster than nearest competitor</b></li> <li>• 1/5 cost of other hardware configurations</li> </ul>
<b>Incremental Inference</b> (LUBM8000 1.06 billion triples + 860M inferred)	<ul style="list-style-type: none"> <li>• Time to update inference: less than 30 seconds after adding 100 triples.</li> <li>• <b>At least 15x to 50x faster than a complete inference done with release 11.1</b></li> </ul>
<b>Large scale owl:sameAs Inference</b> (UniProt 1 Million sample)	<ul style="list-style-type: none"> <li>• 60% less disk space required</li> <li>• 10x faster inference compared to release 11.1</li> </ul>

- Setup: Intel Q6600 quad-core, 3 7200RPM SATA disks, 8GB DDR2 PC6400 RAM, No RAID. 64-bit Linux 2.6.18. **Assembly cost: less than USD 1,000**

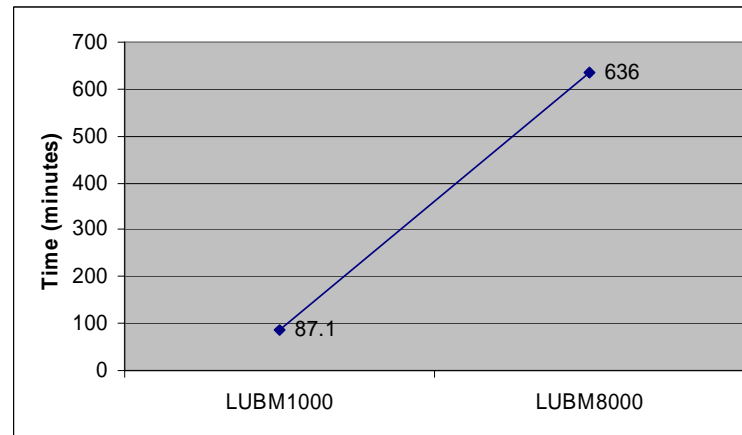
# Query Performance: Going Parallel



- **Setup: Server class machine with 16 cores, NAND based flash storage, 32GB RAM, Linux 64 bit, Average of 3 warm runs**

# Load Performance on Server

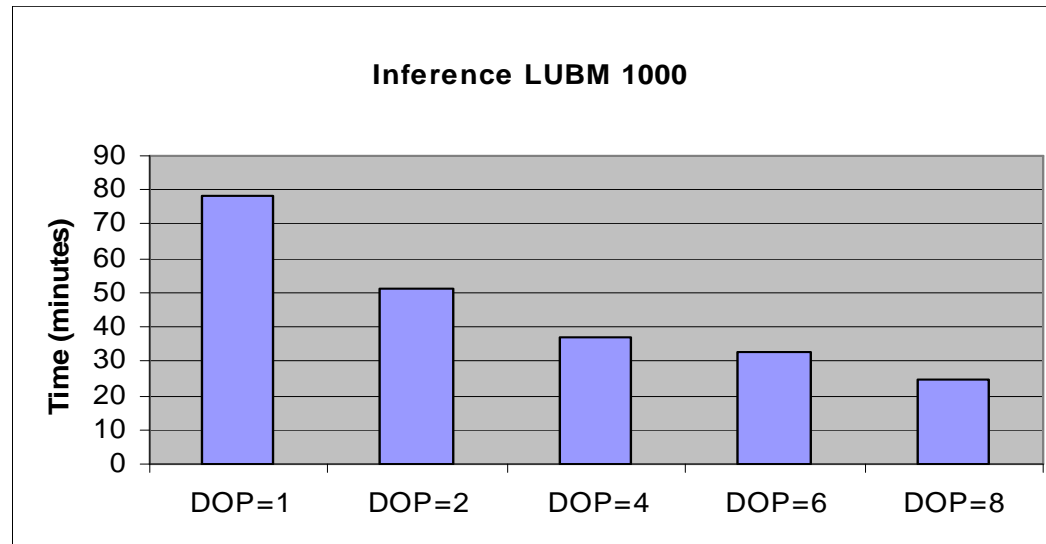
- LUBM1000 (138M triples)
  - 8.3 minutes to load data into staging table
  - 78.8 minutes to load data from staging table (DOP=8)



- LUBM8000 (1B+)
  - 25 minutes to load data into staging table
  - 10hr 36 minutes to load data from staging table (DOP=8)
- Setup: Dual quad-core, Sun Storage F5100 Flash Array, 32 GB RAM

# Inference Performance on Server

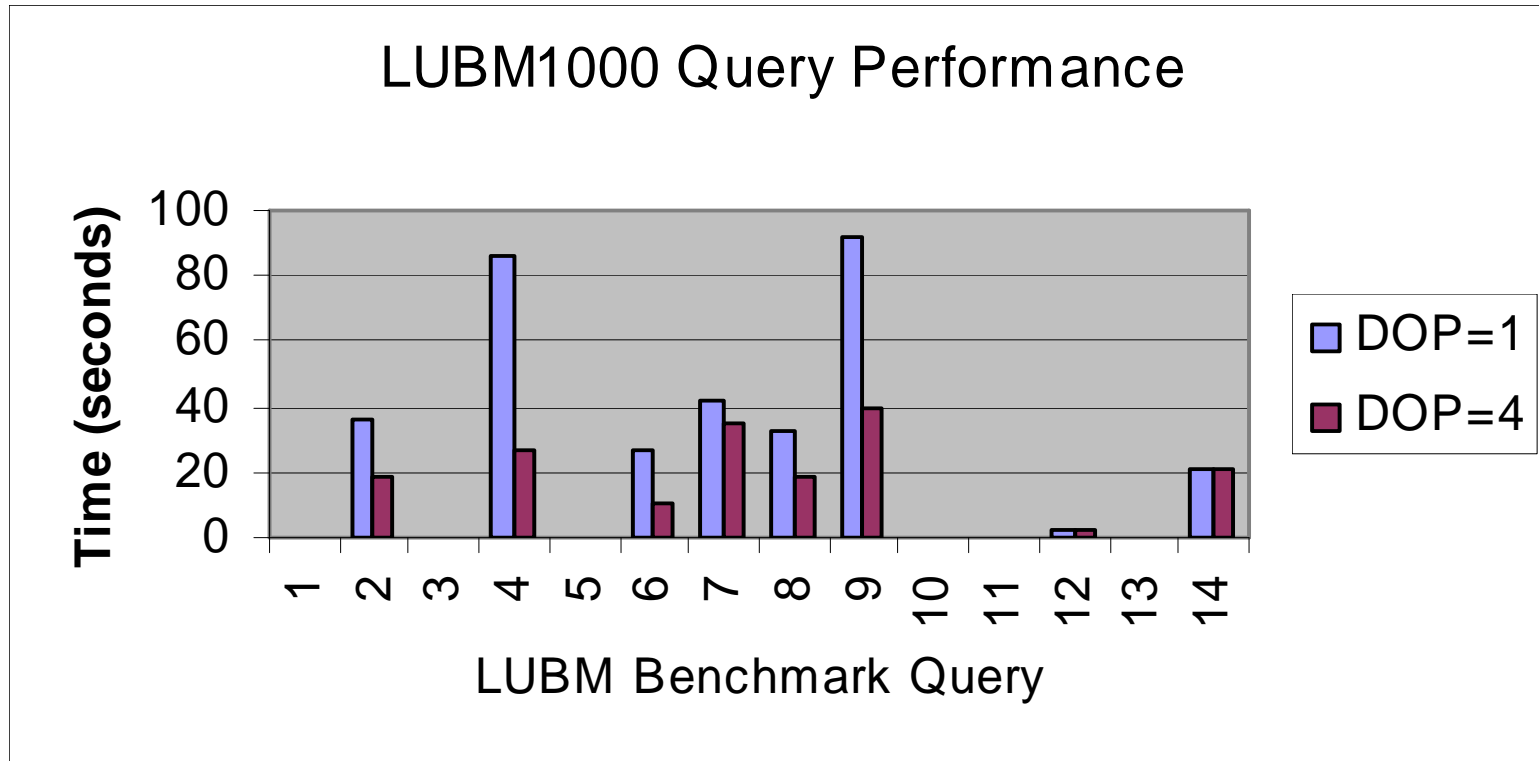
- **Inference performance for LUBM1000 (138M)**
  - 24.6 minutes to infer 108M+ new triples (DOP=8)



- **Inference performance for LUBM8000 (1B+)**
  - 226 minutes to infer 860M+ new triples (DOP=8)
- **Setup: Dual quad-core, Sun Storage F5100 Flash Array, 32 GB RAM**

# Query Performance on Server

- **Parallel query execution**



- **Setup: Server class machine with 16 cores, NAND based flash storage, 32GB RAM, Linux 64 bit, Average of 3 warm runs**



# Oracle RDF Database Benefits

- Native graph data store in Oracle database
  - Optimized for RDF triples
  - Native inferencing engine
- Scalable: Supports ultra-large repositories (+10 billion)
- Open: Growing ecosystem of 3<sup>rd</sup> party tools partners
- Secure: Triple level security
- Aligned with W3C Standards (OWL 2, RDF, SPARQL)
- Powerful Collaborative Change Management
- Interoperable with other business data
  - Business, XML, Spatial, Temporal
- Supported by leading 3<sup>rd</sup> party tools

## Contacts & Resources

- Online training demos, whitepapers, documentation, discussion forum, sample code
  - [http://www.oracle.com/technology/tech/semantic\\_technologies/index.html](http://www.oracle.com/technology/tech/semantic_technologies/index.html)



Oracle RDF

Product Manager: [xavier.lopez@oracle.com](mailto:xavier.lopez@oracle.com)



# **Jena Adaptor for Oracle Database 11g Release 2**



# Jena Adaptor for Oracle Database 11g Release 1

- Implements Jena's Graph/Model/BulkUpdateHandler/... APIs
- “Proxy” like design
  - Data not cached in memory for scalability
  - SPARQL query converted into SQL and executed inside DB
    - A SPARQL with just conjunctive patterns is converted into a single SEM\_MATCH query
- Allows various data loading
  - Bulk/Batch/Incremental load RDF or OWL (in N3, RDF/XML, N-TRIPLE etc.) **with strict syntax verification and long literal support**
- Integrates Oracle Database release 11g RDF/OWL with tools including
  - TopBraid Composer
  - External complete DL reasoners (e.g. Pellet)

# Programming Semantic Applications in Java

- Create an Oracle object
  - `oracle = new Oracle(oracleConnection);`
- Create a GraphOracleSem Object
  - `graph = new GraphOracleSem(oracle, model_name, attachment);`
- Load data
  - `graph.add(Triple.create(...));` // for incremental triple additions
- Collect statistics
  - `graph.analyze();`
- Run inference
  - `graph.performInference();`
- Collect statistics
  - `graph.analyzeInferredGraph();`
- Query
  - `QueryFactory.create(...);`
  - `queryExec = QueryExecutionFactory.create(query, model);`
  - `resultSet = queryExec.execSelect();`


No need to  
create model  
manually!

Important for  
performance!



# Jena Adaptor for Oracle Database 11g Release 2 (1)

- SPARQL service endpoint supporting full [SPARQL Protocol](#)
  - Integrated with Jena Joseki 3.4.0 (deployed in WLS 10.3)
  - Utilizing J2EE data source for DB connection specification
  - SPARQL Update (SPARUL) supported
- Tight integration with Jena ARQ 2.7.0 (2.8.0) for [faster](#) query performance
  - Translate OPTIONAL, UNION, FILTER based queries into a single SEM\_MATCH
  - Translate Property Path queries into plain SQL
  - Translate conjunctive pattern based queries into plain SQL
  - Translate BGP + parallel OPTIONALS (no nesting) into plain SQL ([result\\_cache enabled](#))
  - Fall back to the original Jena ARQ in case of failure



# Jena Adaptor for Oracle Database 11g Release 2 (2)

- Query management and execution control
  - Timeout
  - Query abort framework
    - Including monitoring threads and a management servlet
    - Designed for a J2EE cluster environment
  - Hints allowed in SPARQL query syntax
  - Parallel execution
- Support ARQ functions for projected variables
  - fn:lower-case, upper-case, substring, ...
- Native, system provided functions can be used in SPARQL
  - oext:lower-literal, oext:upper-literal, [oext:build-uri-for-id](#), ...



# Jena Adaptor for Oracle Database 11g Release 2 (3)

- Extensible user-defined functions in SPARQL

- Example

```
PREFIX ouext: <http://oracle.com/semtech/jena-adaptor/ext/user-  
def-function#>  
  
SELECT ?subject ?object (ouext:my_strlen(?object) as ?obj1)  
WHERE { ?subject dc:title ?object }
```

- User can implement the `my_strlen` functions in Oracle Database

- Connection Pooling through OraclePool

```
java.util.Properties prop = new java.util.Properties();  
prop.setProperty("InitialLimit", "2"); // create 2 connections  
prop.setProperty("InactivityTimeout", "1800"); // seconds  
  
....  
OraclePool op = new OraclePool(szJdbcURL, szUser, szPasswd, prop,  
"OracleSemConnPool");  
Oracle oracle = op.getOracle();
```



# Jena Adaptor for Oracle Database 11g Release 2 (4)

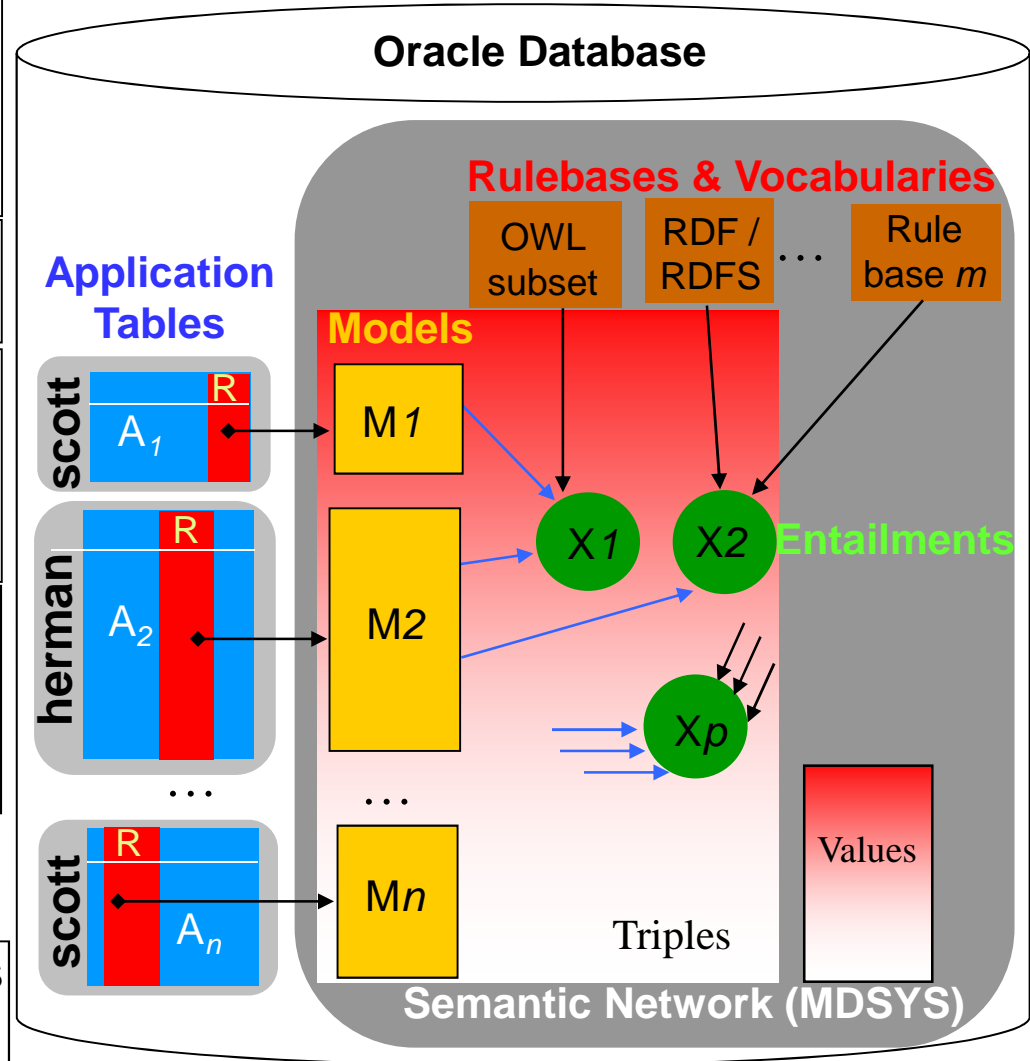
- API enhancements
  - Parallel statistics collection
  - Parallel and incremental inference
  - Parallel application table index building
  - Query **just** inferred data
  - Transparent Virtual model support
  - Shutdown hook logic disabled
  - Basic compression enabled by default for application tables
  - OracleUserRule now accepts filter parameter
  - More utility functions...

# Customer Needs / Oracle Value Propositions

Requirement	Oracle Value Proposition	Oracle Provides
<b>Reasoning and Discovery:</b> <ul style="list-style-type: none"> <li>Organize &amp; relate data with standard vocabularies, taxonomies, ontologies</li> </ul>	<ul style="list-style-type: none"> <li>Accurate, persistent, scalable discovery of new knowledge w/ inferencing and support for well known ontologies e.g., Dublin Core, NCI, SKOS, SNOMED, GENE, PO, FOAF, GeoRSS, SIOC, GoodRelations(ecommerce)</li> </ul>	<ul style="list-style-type: none"> <li>Persistent RDFS/ OWL inferencing</li> <li>User-defined rules for inferencing</li> <li>Plug-in architecture for inference engines such as Pellet, OntoBroker</li> <li>Inferencing proofs and explanations</li> <li>SPARQL &amp; mixed SQL DB queries</li> </ul>
<b>Scalability:</b> <ul style="list-style-type: none"> <li>Evolve schema dynamically</li> <li>Grow to 100's billions of triples</li> </ul>	<ul style="list-style-type: none"> <li>Supports use cases where alternative RDF data stores fail: inferencing beyond memory capacity, data sets 100's m to billions triples, billions of updates &amp; inferences per month</li> </ul>	<ul style="list-style-type: none"> <li>Efficient RDBMS storage of RDF data</li> <li>Support RAC, Exadata platform, partitioning, compression, versioning</li> <li>In DB fast incremental inferencing</li> <li>SQL*Loader direct-path bulk loading</li> <li>Supports concurrent users, dist. apps</li> </ul>
<b>Data Integration:</b> <ul style="list-style-type: none"> <li>Link structured &amp; unstructured content</li> <li>Loosely couple business silos</li> </ul>	<ul style="list-style-type: none"> <li>Save money, increase revenue through content reuse; enables apps to perform where standard integration ontologies are well-defined</li> </ul>	<ul style="list-style-type: none"> <li>RDF URI linking across data sources</li> <li>Jena &amp; Sesame distributed SPARQL</li> <li>Ontologically-assisted SQL query</li> <li>Integration w/ top 3rd party NLP entity extraction engines: e.g., OpenCalais</li> <li>Semantic Indexing for documents</li> </ul>
<b>Security</b> <ul style="list-style-type: none"> <li>Restrict access to data on a "need to know" basis</li> </ul>	<ul style="list-style-type: none"> <li>Save money and time, and protect RDF data in one database w/ fine-grained ACLs</li> </ul>	<ul style="list-style-type: none"> <li>VPD declarative constraints based on RDF data char. &amp; app. / user context</li> <li>OLS restricts RDF data access to users having compatible access labels</li> </ul>

# Core Entities in Oracle Database Semantic

- **Sem. Network Store** → Dictionary and data tables. New entities: **models**, **rule bases**, and **rules indexes (entailments)**. **OWL** and **RDFS** rule bases are preloaded.
- **SDO\_RDF\_TRIPLE\_S** → A new object type for RDF.
- **Application Table** → Contains col of object type `sdo_rdf_triple_s` to allow loading and accessing RDF triples, and storing ancillary values.
- **Model** → A model holds an RDF graph (set of S-P-O triples) and is associated with an `sdo_rdf_triple_s` column in an **application table**.
- **Rulebase** → A rulebase is a set of rules used for inferencing.
- **Entailments** → An entailment stores triples derived via inferencing.



# Load / Query / Inference

- **Load** →

- Bulk load
- Incremental load

- **Query** →

- SPARQL (direct or SQL-based)
- simple data access

- **Inference** →

- using OWL 2 RL, RDFS, etc.
- using User-defined rules

