Building Database Infrastructure for Managing Semantic Data
Agenda

• Semantics support in the database
• Our model
  • Storage
  • Query
  • Inference
• Use cases: Enhancing database queries with semantics
Semantic Technology

- Facts are represented as triples
- Triple is the basic building block in the semantic representation of data
- Triples together form a graph, connecting pieces of data
- New triples can be inferred from existing triples
- RDF and OWL are W3C standards for representing such data
Using a Database for Semantic Applications

- Database queries can be enhanced using semantics
  - Syntactic comparisons can be enhanced with semantic comparisons
- All database characteristics become available for semantic applications
  - Scalability: Database type scale backed by decades of work difficult to match by specialized stores
  - Security, transaction control, availability, backup and recovery, lifecycle management, etc.
Using a Database for Semantic Applications (contd.)

• SQL (an open standard) interface is familiar to a large community of developers
  • Also SQL constructs can be used for operating on semantic data
  • Existing database users interested in exploring semantics to enhance their applications

• Databases are part of infrastructure in several categories of applications that use semantics
  • Biosurveillance, Social Networks, Telcos, Utilities, Text, Life Sciences, GeoSpatial
Our Approach

• Provide support for managing RDF data in the database for semantic applications
  • Storage Model
  • SQL-based RDF query interface
  • Query interface that enables combining with SQL queries on relational data
  • Inferencing in the database (based on RDFS and user-defined rules)
  • Support for large graphs (billion+ triples)
Technical Overview

**INFER**
- RDF/S
- User def. rules

**QUERY**
- Query RDF/OWL data and ontologies
- Combining relational queries with RDF/OWL queries

**STORE**
- Batch Load
- Incr. Load and DML

RDF/OWL data and ontologies
Enterprise (Relational) data
Semantic Technology Stack

- Self-desc. doc.
- Data
- Rules
- Proof
- Trust
- Digital Signature
- Universal Description
- Logic
- Ontology vocabulary
- RDF + rdfschema
- XML + NS + xsmlschema
- Unicode
- URI

Standards based
Semantic Technology
Storage
Storage: Schema Objects

RDF/OWL data and ontologies

- Appl. Tables
  - A1 -> Model 1
  - A2 -> Model 2
  - ... -> Model n

- Rulebase 1
- Rulebase 2
- ... -> Rulebase m

- Inferred Triple Set 1
- Inferred Triple Set 2
- ... -> Inferred Triple Set p
## Model Storage

### Application table 1

<table>
<thead>
<tr>
<th>ID (number)</th>
<th>TRIPLE (sdo_rdf_triple_s)</th>
<th>…</th>
<th>…</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Optional columns for related enterprise data

- Application table links to model in internal semantic store

### Application table 2

<table>
<thead>
<tr>
<th>Triple (SDO_RDF_TRIPLE_S)</th>
<th>……</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Internal Semantic Store
## Internal Semantic Store

### IdTriples

<table>
<thead>
<tr>
<th>Model</th>
<th>S_id</th>
<th>P_id</th>
<th>O_id</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition containing Data for <strong>Model 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition containing Data for <strong>Model n</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition containing Data for <strong>Inferred Triple Set 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition containing Data for <strong>Inferred Triple Set p</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### UriMap

<table>
<thead>
<tr>
<th>Value</th>
<th>Id</th>
<th>Type</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapping: Value 1:1 Id</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Rulebase

<table>
<thead>
<tr>
<th>Rb</th>
<th>rule</th>
<th>ante</th>
<th>filter</th>
<th>cons</th>
</tr>
</thead>
</table>
Storage: Highlights

- Generates **hash-based IDs** for values (handles collisions)
- Does **canonicalization** to handle **multiple lexical forms** of same value point
  - Ex: “0010”^^xsd:decimal and “010”^^xsd:decimal
- Maintains **fidelity** (user-specified lexical form)
- Allows **long literal** values (using CLOBs)
- Handles **duplicate** triples
- **No limits** on amount of data that can be stored
Semantic Technology
Query
RDF Querying Problem

• **Given**
  • RDF graphs: the data set to be searched
  • Graph Pattern: containing a set of variables

• **Find**
  • Matching Subgraphs

• **Return**
  • Sets of variable bindings: where each set corresponds to a Matching Subgraph
Query Example: Family Data

Data: :Tom :hasParent :Matt
    :Matt :hasFather :John
    :Matt :hasMother :Janice
    :Jack :hasParent :Suzie
    :Suzie :hasFather :John
    :Suzie :hasMother :Janice
    :John :hasName “John D”

Graph pattern ‘(:Tom :hasParent ?x)
    (?x :hasFather ?y)
    (?y :name ?name),’

Variable bindings: x = :Matt
    y = :John
    name = “John D”

Matching subgraph:
    ‘(:Tom :hasParent :Matt)
    (:Matt :hasFather :John)
    (:John :name “John D”),’
RDF Query Approaches

• General Approach
  • Create a new (declarative, SQL-like) query language
  • e.g.: RQL, SeRQL, TRIPLE, N3, Versa, SPARQL, RDQL, RDFQL, SquishQL, RSQl, etc.

• Our SQL-based Approach
  • Embedding a graph query in a SQL query
  • SPARQL-like graph pattern embedded in SQL query

• Benefits of SQL-based Approach
  • Leverages all the powerful constructs in SQL (e.g., SELECT / FROM / WHERE, ORDER BY, GROUP BY, aggregates, Join) to process graph query results
  • RDF queries can easily be combined with conventional queries on database tables thereby avoiding staging
SDO_RDF_MATCH Table Function

- **Input Parameters**
  
  SDO_RDF_MATCH (  
  Query, \(\leftarrow\) SPARQL-like graph-pattern (with vars)  
  Models, \(\leftarrow\) set of RDF/OWL models  
  Rulebases, \(\leftarrow\) set of rulebases (e.g., RDFS)  
  Aliases, \(\leftarrow\) aliases for namespaces  
  Filter \(\leftarrow\) additional selection criteria  
  )

- **Return type in definition is** AnyDataSet
- **Actual return type is determined at compile time based on the graph-pattern argument**
Query Example: SQL-based interface

```
select x, y, name from TABLE(SDO_RDF_MATCH(
    '(Tom :hasParent ?x)
    (?x :hasFather ?y)
    (?y :name ?name)',
    SDO_RDF_Models('family'),
    .., .., ..));
```

Returns the name of Tom’s grandfather

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matt</td>
<td>John</td>
<td>“John D”</td>
</tr>
</tbody>
</table>
Combining RDF Queries with Relational Queries

• Find salary and hiredate of Tom’s grandfather(s)
• SELECT emp.name, emp.salary, emp.hiredate
  FROM emp, TABLE(SDO_RDF_MATCH(
    ‘(:Tom :hasParent ?y)
    (?y :hasFather ?x)
    (?x :name ?name’),
    SDO_RDF_Models(‘family’),
    ...)) t
  WHERE emp.name=t.name;
RDF_MATCH Query Processing

- Substitute aliases with namespaces in search pattern
- Convert URIs and literals to internal IDs
- Generate Query
  - Generate self-join query based on matching variables
  - Generate SQL subqueries for rulebases component (if any)
  - Generate the join result by joining internal IDs with UriMap table
  - Use model IDs to restrict IdTriples table
- Compile and Execute the generated query
Table Columns returned by SDO_RDF_MATCH

Each returned row contains one (or more) of the following cols (of type VARCHAR2) for each variable `?x` in graph-pattern:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Value matched with <code>?x</code></td>
</tr>
<tr>
<td>x$rdfVTYP</td>
<td>Value <strong>TY</strong>pe: URI, Literal, or Blank Node</td>
</tr>
<tr>
<td>x$rdfLTYP</td>
<td>Literal <strong>TY</strong>pe: e.g., xsd:integer</td>
</tr>
<tr>
<td>x$rdfCLOB</td>
<td>CLOB value matched with <code>?x</code></td>
</tr>
<tr>
<td>x$rdfLANG</td>
<td><strong>LANG</strong>uage tag: e.g., “en-us”</td>
</tr>
</tbody>
</table>

**Projection Optimization**: Only the columns referred to by the containing query are returned.
Optimization: Table Function Rewrite

- **TableRewriteSQL( )**
  - Takes RDF Query (specified via arguments) as input
  - generates a SQL string
- Substitute the table function call with the generated SQL string
- Reparse and execute the resulting query
- Advantages
  - Avoid execution-time overhead (linear in number of result rows) associated with table function infrastructure
  - Leverage SQL optimizer capabilities to optimize the resulting query (including filter condition pushdown)
Semantic Technology
Inference
Inference: Overview

• Native inferencing in the database for
  • RDF, RDFS
  • User-defined rules
• Rules are stored in rulebases in the database
• RDF graph is entailed (new triples are inferred) by applying rules in rulebase/s to model/s
• Infererencing is based on forward chaining: new triples are inferred and stored ahead of query time
  • Minimizes on-the-fly computation and results in fast query times
Inferencing

• RDFS Example:

\[
A \text{ rdf:type } B, \; B \text{ rdfs:subClassOf } C
\]
\[
\Rightarrow \; A \text{ rdf:type } C
\]
Ex: Matt rdf:type Father, Father rdfs:subClassOf Parent
\[
\Rightarrow \; Matt \text{ rdf:type Parent}
\]

• User-defined Rules Example:

\[
A :\text{hasParent} B, \; B :\text{hasParent} C
\]
\[
\Rightarrow \; A :\text{hasGrandParent} C
\]
Ex: Tom :hasParent Matt, Matt :hasParent John
\[
\Rightarrow \; Tom :\text{hasGrandParent} John
\]
Creating a rulebase and rules index (SQL based)

• Creating a rule base
  • `create_rulebase('family_rb');`
  • `insert into mdsys.RDFR_family_rb values('grandParent_rule',
    '(?x :hasParent ?y) (?y :hasParent ?z)',
    NULL,
    '(?x :hasGrandParent ?z)',
    .....);

• Creating a rules index
  • `create_rules_index('family_idx',sdo_rdf_models('family'),sdo_rdf_rulebases('rdfs','family_rb'))`
Query Example: Family Data

```
select y, name from TABLE(SDO_RDF_MATCH(
  ‘(:Tom :hasGrandParent ?y)
  (?y :name     ?name)’
  (?y rdf:type   :Male),
  SDO_RDF_Models('family'),
  SDO_RDF_Rulebases('family_rb'),
  .., ..));
```

Returns the name of Tom’s grandfather

<table>
<thead>
<tr>
<th></th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>‘John D’</td>
</tr>
</tbody>
</table>
Semantic Technology
Enhancing Database Queries with Semantics
Semantics Enhanced Search

**Medical Information Repositories**

- Multiple users might use multiple sets of terms to annotate medical images
  - Difficult to search across multiple medical image repositories

```
<table>
<thead>
<tr>
<th>Id</th>
<th>Image</th>
<th>Metadata</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>....Maxilla.......</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>....Mandible.....</td>
</tr>
<tr>
<td></td>
<td></td>
<td>........</td>
</tr>
</tbody>
</table>
```

Find me all images containing ‘Jaw’

- Consult Ontology
  - Ontology for SNOMED terms
  - Jaw
  - Maxilla
  - Mandible
Semantics Enhanced Search

**Geo-Semantics**

- Enhance geo-spatial search with semantics
  - Create an ontology using business categorizations (from the NAICS taxonomy) and use that to enhance yellow pages type search

Find me a Drug store near where I am

<table>
<thead>
<tr>
<th>Id</th>
<th>Business</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pharmacies and Drug Stores</td>
<td>...Health &amp; Personal care stores....</td>
</tr>
<tr>
<td>2</td>
<td>Cosmetics, Beauty Supplies, and Perfume Stores</td>
<td>....Pharmacies and drug stores....</td>
</tr>
</tbody>
</table>

Ontology for business categorizations

Oracle
Faceted Geo-Semantic Search powered by Oracle Semantic Technologies

Name: %
Health and Personal Care Stores

use semantics

Map of the San Francisco Bay area with cities and roads labeled.
Faceted Geo-Semantic Search powered by Oracle Semantic Technologies

Name: %
Health and Personal Care Stores

WALGREENS
PHONE: 510-2614552
BUSINESS_ID: 12835
CATEGORY_ID: 446110
CATEGORY_NAME: Pharmacies and Drug Stores
NAICS.nt

http://www.oracle.com/sw/taxonomy.owl#naics2002_code448111

http://www....taxonomy.owl#naics20...
Biosurveillance

- Biosurveillance application: Track patterns in health data
- Data from 8 emergency rooms in Houston at 10 minute intervals
- Data converted into RDF/OWL and loaded into the database
- 8 months data is 600M+ triples
- Automated analysis of data to track patterns:
  - Spike in flu-like symptoms (RDF/OWL inferencing to identify a flu-like symptom)
  - Spike in children under age 5 coming in
Data Integration in the Life Sciences

“Find all pieces of information associated with a specific target”

• Data integration of multiple datasets
  • Across multiple representation formats, granularity of representation, and access mechanisms
  • Across In-house and public sets (Gene Ontology, UniProt, NCI thesaurus, etc.).
• Standardized and machine-understandable data format with an open data access model is necessary to enable integration
  • Data-warehousing approach represents all data to be integrated in RDF/OWL
  • Semantic metadata layer approach links metadata from various sources and maps data access tool to relevant source
• Ability to combine RDF/OWL queries with relational queries is a big benefit
• Lilly and Pfizer are using semantic technology to solve data integration problems
Use Case: SenseLab Overview

The SenseLab Project is a long term effort to build integrated, multidisciplinary models of neurons and neural systems, using the olfactory pathway as a model. This is one of a number of projects funded as part of the Human Brain Project whose aim is to develop neuroinformatics tools in support of neuroscience research. The project involves novel informatics approaches to constructing databases and database tools for collecting and analyzing neuroscience information, and providing for efficient interoperability with other neuroscience databases.

- Overview
- Membrane Properties Resource

Brain Database Research

<table>
<thead>
<tr>
<th>Neuronal Databases</th>
<th>CellPropDB</th>
<th>NeuronDB</th>
<th>ModelDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Olfactory Databases</td>
<td>ORDB</td>
<td>odorDB</td>
<td>OdorMapDB</td>
</tr>
<tr>
<td>Disease Databases</td>
<td>BrainPharm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Part of this work published in the Workshop on Semantic e-Science

Courtesy: SenseLab, Yale University
Relational to Ontological Mapping

<table>
<thead>
<tr>
<th>Compartmen</th>
<th>Cell: NeuronDB</th>
<th>Receptor</th>
<th>Channel</th>
<th>Pathological Agent (PA)</th>
<th>PA Action</th>
<th>Drug</th>
<th>Drug Action</th>
<th>Stage</th>
<th>Note</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soma</td>
<td>CA1 pyramidal neuron</td>
<td>I A</td>
<td></td>
<td>beta Amyloid</td>
<td>Inhibits</td>
<td></td>
<td></td>
<td>Early</td>
<td>View</td>
<td>66240</td>
</tr>
<tr>
<td></td>
<td>Olfactory bulb mitral cell</td>
<td>GabaA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Early</td>
<td>View</td>
<td>66750</td>
</tr>
<tr>
<td>Dendrite</td>
<td>CA1 pyramidal neuron</td>
<td>I A</td>
<td></td>
<td>beta Amyloid</td>
<td>Inhibits</td>
<td></td>
<td></td>
<td>Early</td>
<td>View</td>
<td>66240</td>
</tr>
<tr>
<td></td>
<td>Olfactory bulb mitral cell</td>
<td>GabaA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Early</td>
<td>View</td>
<td>66750</td>
</tr>
<tr>
<td>Unspecified</td>
<td>Oocyte</td>
<td>I L high threshold</td>
<td>beauty Amyloid</td>
<td>Inhibits</td>
<td></td>
<td></td>
<td></td>
<td>Early</td>
<td>View</td>
<td>66252</td>
</tr>
<tr>
<td></td>
<td>CA1 pyramidal neuron</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Early</td>
<td>View</td>
<td>66753</td>
</tr>
<tr>
<td></td>
<td>CA1 pyramidal neuron</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Early</td>
<td>View</td>
<td>66758</td>
</tr>
<tr>
<td></td>
<td>CA1 pyramidal neuron</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>View</td>
<td></td>
<td>66250</td>
</tr>
</tbody>
</table>

Courtesy: SenseLab, Yale University
Use Case: Integrated Bioinformatics Data
Use Case: Knowledge Mining Solutions

Web Resources → Information Extraction → RDF/OWL

Information Extraction
- Categorization, Feature/term Extraction

Processed Document Collection → RDF/OWL

Knowledge Mining & Analysis
- Text Indexing using Oracle Text
- Non-Obvious Relationship Discovery
- Pattern Discovery
- Text Mining
- Faceted Search

Ontology Engineering Modeling Process → OWL Ontologies

Domain Specific Knowledge Base

SQL/SPARQL Query

Explore → Analyst

Browsing, Presentation, Reporting, Visualization, Query
Safe Harbor Statement & Confidentiality

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 decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Semantic Operators in SQL

- Two new first class SQL operators to semantically query relational data by consulting an ontology
  - SEM_RELATED (<col>,<pred>, <ontologyTerm>, <ontologyName> [,<invoc_id>])
  - SEM_DISTANCE (<invoc_id>) \textbf{\textless{} Ancillary Oper.}
- Can be used in any SQL construct (ORDER BY, GROUP BY, SUM, etc.)

- Semantic indextype
  - An index of type semantic indextype introduced for efficient execution of queries using the semantic operators
Ontology-assisted Query

```
Patients

<table>
<thead>
<tr>
<th>ID</th>
<th>DIAGNOSIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hand_Fracture</td>
</tr>
<tr>
<td>2</td>
<td>Rheumatoid_Arthritis</td>
</tr>
</tbody>
</table>
```

“Find all entries in diagnosis column that are related to ‘Upper_Extremity_Fracture’”

Syntactic query will not work:
```
SELECT p_id, diagnosis FROM Patients WHERE diagnosis = ‘Upper_Extremity_Disorder’;
```
Ontology-assisted Query

SELECT p_id, diagnosis
FROM Patients
WHERE SEM_RELATED (diagnosis, ‘rdfs:subClassOf’, ‘Upper_Extremity_Fracture’, ‘Medical_ontology’) = 1;

Patients

<table>
<thead>
<tr>
<th>ID</th>
<th>DIAGNOSIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hand_Fracture</td>
</tr>
<tr>
<td>2</td>
<td>Rheumatoid_Arthritis</td>
</tr>
</tbody>
</table>
Ontology-assisted Query

SELECT p_id, diagnosis FROM Patients
WHERE
    diagnosis, 'rdfs:subClassOf', 'Upper_Extremity_Fracture', 'Medical_ontology' = 1
    AND SEM_DISTANCE() <= 2;

Patients

<table>
<thead>
<tr>
<th>ID</th>
<th>DIAGNOSIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hand_Fracture</td>
</tr>
<tr>
<td>2</td>
<td>Rheumatoid_Arthritis</td>
</tr>
</tbody>
</table>
Summary

- Semantic Technology support in the database
  - Store RDF/OWL data and ontologies
  - Infer new RDF/OWL triples via native inferencing
  - Query RDF/OWL data and ontologies
  - Ontology-Assisted Query of relational data

- More information at: