Tutorial 6: Large scale semantic graph data management and analytics

Olivier Curé
University of Paris-Est Marne la Vallée

olivier.cure@u-pem.fr
@oliviercure
Who am I?

- Associate professor in Computer Science at Université Paris-Est Marne la Vallée (France)
- Research on Data and Knowledge management, Reasoning, Big data
- Designed and maintains a self-medication web application targeting over 7 million clients of more than 35 insurance companies in France
Tutorial overview

• Semantic Web
• Storing and querying
• Automated reasoning
• Integrating data and knowledge
• Analytics
• Conclusion
Semantic is in the air

It is spring season for AlphaGo

IBM Watson

- Big trend is Machine learning but many discovered models will be handled by Semantic data management tools
- Many initiatives to represent semantic data on the Web, e.g., Schema.org
Plan

- Semantic Web technologies
- Storing and querying
- Automated reasoning
- Integrating data and knowledge
- Analytics
- Conclusion
Goals of this section

- Short presentation of
  - RDF, a data model
  - SPARQL, a query language for RDF
  - and ontology languages, to represent knowledge and support inferences
RDF (Resource Description Framework)

- The most prevalent data approach to represent semantic data and meta-data

- A key component of W3C’s Semantic Web stack together with
  - SPARQL
    - A query language for RDF
  - RDFS
    - A light ontology language
  - OWL
  - A family of expressive ontology languages
RDF (2)

- W3C recommendations in 1999 and 2014 (RDF 1.1)
- Supports the definition of directed labeled graphs

- A data model for the Web of data based on the notion of triples : (subject, predicate, object)
  - Omnipresence of IRIs
  - Triple signature \((s, p, o) \rightarrow (I \cup B) \times I \times (I \cup B \cup L)\) with I, B and L sets of resp. IRIs, Blank nodes, and Literals
  - Several syntaxes available: RDF/XML, JSON-LD, RDFa and Turtle family
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbp: <http://dbpedia.org/property/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
dbr:Kind_of_Blue rdf:type dbr:Album ;
    dbp:writer dbr:Miles_Davis ;
    foaf:name "Kind of Blue"@en ;
    dbo:genre dbr:Modal_jazz ;
    dbo:releaseDate "1959-08-17"^^xsd:date .

dbr:The_Big_Nowhere rdfs:subClassOf dbo:Book ;
    dbo:genre dbr:James_Ellroy ;
    dbp:language "English"^^xsd:string ;
    foaf:name "The Big Nowhere"@en ;
    dbo:releaseDate "September 1988"^^xsd:string .

dbr:Into_the_Wild_(film) rdf:type dbo:Film ;
    dbo:director dbr:Sean_Penn ;
    foaf:name "Into the Wild"@en ;
    dbp:language "English"^^xsd:string .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbp: <http://dbpedia.org/property/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

dbr:Kind_of_Blue rdf:type dbo:Album ;
    dbp:writer dbr:Miles_Davis ;
    foaf:name "Kind of Blue"@en ;
    dbo:genre dbr:Modal_jazz ;
    dbo:releaseDate "1959-08-17"^^xsd:date .

dbr:The_Big_Nowhere rdf:type dbo:Book ;
    dbo:author dbr:James_Ellroy ;
    dbp:language "English"^^rdf:langString ;
    foaf:name "The Big Nowhere"@en ;

dbr:Into_the_Wild_(film) rdf:type dbo:Film;
    dbo:director dbr:Sean_Penn ;
    foaf:name "Into the Wild"@en ;
A query language for RDF data

Conjunctive queries based on graph pattern matching

Triples that possibly contain variables (prefixed with ? Symbol)

Example:

```
PREFIX rdf : <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp : <http://dbpedia.org/property/>
SELECT ?s ?date ?lang
WHERE {
    ?s rdf:type dbo:Book ;
    dbp:releaseDate ?date ;
}
```
SPARQL

- A query language for RDF data
- Conjunctive queries based on graph pattern matching
- Triples that possibly contain variables (prefixed with ? Symbol)
  - Example:

```sparql
PREFIX rdf : <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbp : <http://dbpedia.org/property/>
SELECT ?s ?date ?lang
WHERE {
    ?s rdf:type dbo:Book ;
    dbp:releaseDate ?date ;
}

?s dbr:The_Big_Nowhere
?date "English"^^...?
?lang "September 1988"^^...
Ontology languages

- For a given domain, an ontology describes a set of concepts and their relationships.
- RDF Schema (RDFS) is the language with the lowest expressiveness: sub-classes, sub-properties, domain and range of properties, instance typing.

Diagram:
- Person
  - Artist
  - Writer
    - SongWriter
    - ScreenWriter
- Work
  - WrittenBook
  - Book

Relationships:
- author
- rdfs:subClassOf
- object property
Ontology languages (2)

- OWL (Web Ontology Language) is a family of increasing expressive ontology languages

- Expressiveness comes at a computational cost justifying a trade-off

- OWL2 QL, RL and EL are tractable

- OWL2 DL is not tractable

- OWL2 Full is not decidable
Knowledge Base (KB)

- KB = a set of facts + an ontology

- In Description Logics terminology:
  - KB = <ABox, TBox>
    - With ABox = Assertional box
    - And TBox = Terminological box
Plan

- Semantic Web technologies
- **Storing and querying**
- Automated reasoning
- Integrating data and knowledge
- Analytics
- Conclusion
Storing and querying RDF triples

Goals of this section

- Achieve an initial understanding of the RDF database management ecosystem
- Understand differences between 7 identified production-ready stores
RDF storage

- Although most production-ready RDF stores support ACID characteristics, they are mostly loading data in batches.

- They are not used as OLTP databases (On line transaction processing) and do not possess OLAP (online analytical processing) functionalities.

- But they enable data integration at Web scale together with reasoning services.
RDF Storage (2)

- RDF is a logical data model and thus does not impose any physical storage solution

- Existing RDF stores are either
  - based on an existing DataBase Management System,
    - relational model, e.g., PostgreSQL
    - NoSQL, e.g., Cassandra
  - Designed from scratch, e.g., as a Graph store
RDF stores taxonomy

Native
- Main memory-based
- Disk-based

Non-native
- RDBMS
  - Schema-based
    - Vertical partitioning
    - Hierarchical property table
  - Property table
- NoSQL
  - Schema-free
    - Triple table
  - Key-value
    - Column family
    - Document store
  - Graph database
Running example

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbp: <http://dbpedia.org/property/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

dbr:Kind_of_Blue rdf:type             dbo:Album ;
       dbp:writer  dbr:Miles_Davis ;
       foaf:name   "Kind of Blue"@en;
       dbo:genre   dbr:Modal_jazz ;
       dbo:releaseDate "1959-08-17"^^xsd:date .

dbr:The_Big_Nowhere rdf:type             dbo:Book ;
       dbo:author  dbr:James_Ellroy ;
       dbp:language "English"^^rdf:langString ;
       foaf:name   "The Big Nowhere"@en ;

dbr:Into_the_Wild_(film) rdf:type             dbo:Film;
       dbo:director dbr:Sean_Penn;
       foaf:name   "Into the Wild"@en ;
## Storage layout – triple table

- One large relation with 3 columns (s,p,o)

<table>
<thead>
<tr>
<th>Subject</th>
<th>Property</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbr:Kind_of_Blue</td>
<td>rdf:type</td>
<td>dbo:Album</td>
</tr>
<tr>
<td>dbr:Kind_of_Blue</td>
<td>dbp:writer</td>
<td>dbr:Miles_Davis</td>
</tr>
<tr>
<td>dbr:Kind_of_Blue</td>
<td>foaf:name</td>
<td>&quot;Kind of Blue&quot;@en</td>
</tr>
<tr>
<td>dbr:Kind_of_Blue</td>
<td>dbo:genre</td>
<td>dbr:Modal_jazz</td>
</tr>
<tr>
<td>dbr:Kind_of_Blue</td>
<td>dbo:releaseDate</td>
<td>&quot;1959-08-17&quot;^^xsd:date</td>
</tr>
<tr>
<td>dbr:The_Big_Nowhere</td>
<td>rdf:type</td>
<td>dbo:Book</td>
</tr>
<tr>
<td>dbr:The_Big_Nowhere</td>
<td>dbo:author</td>
<td>dbr:James_Ellroy</td>
</tr>
<tr>
<td>dbr:The_Big_Nowhere</td>
<td>dbp:language</td>
<td>&quot;English&quot;^^rdf:langString</td>
</tr>
<tr>
<td>dbr:The_Big_Nowhere</td>
<td>foaf:name</td>
<td>&quot;The Big Nowhere&quot;@en</td>
</tr>
<tr>
<td>dbr:The_Big_Nowhere</td>
<td>dbp:releaseDate</td>
<td>&quot;September 1988&quot;^^rdf:langString</td>
</tr>
<tr>
<td>dbr:Into_the_Wild_(film)</td>
<td>rdf:type</td>
<td>dbo:Film</td>
</tr>
<tr>
<td>dbr:Into_the_Wild_(film)</td>
<td>dbo:director</td>
<td>dbr:Sean_Penn</td>
</tr>
<tr>
<td>dbr:Into_the_Wild_(film)</td>
<td>foaf:name</td>
<td>&quot;Into the Wild&quot;@en</td>
</tr>
<tr>
<td>dbr:Into_the_Wild_(film)</td>
<td>dbp:writer</td>
<td>dbr:Miles_Davis</td>
</tr>
<tr>
<td>dbr:Into_the_Wild_(film)</td>
<td>foaf:name</td>
<td>&quot;Into the Wild&quot;@en</td>
</tr>
</tbody>
</table>
## Storage layout – Clustered property table

- Clusters properties that tend to appear together

### Property table

<table>
<thead>
<tr>
<th>Subject</th>
<th>rdf:type</th>
<th>dbp:language</th>
<th>dbp:releaseDate</th>
<th>foaf:name</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbr:Kind_of_Blue</td>
<td>dbo:Album</td>
<td>NULL</td>
<td>&quot;1959-08-17&quot;</td>
<td>&quot;Kind of Blue&quot;</td>
</tr>
<tr>
<td>dbr:The_Big_Nowhere</td>
<td>dbo:Book</td>
<td>&quot;English&quot;</td>
<td>&quot;September 1988&quot;</td>
<td>&quot;The Big Nowhere&quot;</td>
</tr>
<tr>
<td>dbr:Into_the_Wild_(film)</td>
<td>dbo:Film</td>
<td>&quot;English&quot;</td>
<td>NULL</td>
<td>&quot;Into the Wild&quot;</td>
</tr>
</tbody>
</table>

### Left-over table

<table>
<thead>
<tr>
<th>Subject</th>
<th>Property</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbr:Kind_of_Blue</td>
<td>dbp:writer</td>
<td>dbr:Miles_Davis</td>
</tr>
<tr>
<td>dbr:Kind_of_Blue</td>
<td>dbo:genre</td>
<td>dbr:Modal_jazz</td>
</tr>
<tr>
<td>dbr:The_Big_Nowhere</td>
<td>dbo:author</td>
<td>dbr:James_Ellroy</td>
</tr>
<tr>
<td>dbr:Into_the_Wild_(film)</td>
<td>dbo:director</td>
<td>dbr:Sean_Penn</td>
</tr>
</tbody>
</table>
Storage layout – Property-class table

- One table for each class

**Class: Album**

<table>
<thead>
<tr>
<th>Subject</th>
<th>dbp:writer</th>
<th>foaf:name</th>
<th>dbo:genre</th>
<th>dbp:releaseDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbr:Kind_Of_Blue</td>
<td>dbr:Miles_Davis</td>
<td>&quot;Kind of Blue&quot;</td>
<td>dbr:Modal_jazz</td>
<td>&quot;1959-08-17&quot;</td>
</tr>
<tr>
<td>dbr:ESP</td>
<td>dbr:Miles_Davis</td>
<td>&quot;E.S.P.&quot;</td>
<td>dbr:Jazz</td>
<td>&quot;1965&quot;</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Class: Book**

<table>
<thead>
<tr>
<th>Subject</th>
<th>dbp:author</th>
<th>dbp:language</th>
<th>foaf:name</th>
<th>dbp:releaseDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbr:The_Big_Nowhere</td>
<td>dbr:James_Ellroy</td>
<td>&quot;English&quot;</td>
<td>&quot;The Big Nowhere&quot;</td>
<td>&quot;September 1988&quot;</td>
</tr>
<tr>
<td>dbr:The_Road</td>
<td>dbr:Cormac_McCarthy</td>
<td>&quot;English&quot;</td>
<td>&quot;The road&quot;</td>
<td>&quot;September 2006&quot;</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Left-over table**

<table>
<thead>
<tr>
<th>Subject</th>
<th>Property</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbr:Into_the_Wild_(film)</td>
<td>dbo:director</td>
<td>dbr:Sean_Penn</td>
</tr>
<tr>
<td>dbr:Into_the_Wild_(film)</td>
<td>foaf:name</td>
<td>&quot;Into the Wild&quot;</td>
</tr>
<tr>
<td>dbr:Into_the_Wild_(film)</td>
<td>dbp:language</td>
<td>&quot;English&quot;</td>
</tr>
<tr>
<td>dbr:Into_the_Wild_(film)</td>
<td>rdf:type</td>
<td>dbo:Film</td>
</tr>
</tbody>
</table>
Storage layout – Vertical partitioning

- One table two-column per property

<table>
<thead>
<tr>
<th>Subject</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbr:Kind_of_Blue</td>
<td>&quot;1959-08-17&quot;^^xsd:date</td>
</tr>
<tr>
<td>dbr:The_Big_Nowhere</td>
<td>&quot;September 1988&quot;^^rdf:langString</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbr:Kind_of_Blue</td>
<td>dbo:Album</td>
</tr>
<tr>
<td>dbr:The_Big_Nowhere</td>
<td>dbo:Book</td>
</tr>
<tr>
<td>dbr:Into_the_Wild_(film)</td>
<td>dbo:Film</td>
</tr>
</tbody>
</table>
Query processing and indexing

- Graph query processing relates to navigating from nodes to nodes. It requires a high rate of join operations.

- Query processing efficiency largely depends on the indexing the triples of the graph.

- Systems like RDF-3X stores up to 15 indexes (SPO, SOP, POS, PSO, .., SP, SO, .., S, P, O), Hexastore, Yars2 have 6 indexes.
RDF Store ecosystem

Non-native
- 3Store
- Jena2
- JenaDB
- RDFSuite
- DLDB
- RDFStore
- RDFBroker
- SWStore
- roStore
- RDFKB
- RDFJoin
- DB2 RDF

NoSQL
- Aweto
- CumulusRDF
- D-SPARQL
- PigSPARQL
- H2RDF
- TrinityRDF
- MarkLogic

Production ready
- Oracle
- Virtuoso
- v5
- v11
- v12
- v7
- v8

Native
- OWLIM
- JenaTDB
- bigdata
- v2
- v1.1
- v2
- v2.2
- v2.3
- v2.6
- v2.7
- now RDF4J
- v2.5
- AnzoGraph

Sesame
- v2
- v3
- v4
- v5.1
- v6.2
- v6.3
- v6.5

Compressed
- HDT
- TripleBit
- WaterFowl
- RDF
- TriAD
- dist-RDFox
- Wukong
- DiploCloud
- SPARQLGx
- Sempala
- S2RDF

System inheritance
System influence
RDF distributed data management

- RDF storage is part of Big data
- Distribution of RDF triples over a cluster of machines
Seven production-ready stores
Seven production-ready stores

- They all guarantee
  - ACID transactions
  - Replication (mostly Master-Slave, some Master-Master)
  - Partition (Range, Hashing)
Data models and querying

- Some of these systems support other data models
  - XML for MarkLogic and Virtuoso
  - Property graph for GraphDB, BlazeGraph and Stardog
Data models and querying

- Some of these systems support other data models
  - XML for MarkLogic and Virtuoso
  - Property graph for GraphDB, BlazeGraph and Stardog
  - Relational for Virtuoso and Oracle
  - Document for MarkLogic

- Hence other query languages than SPARQL (v1.1) can be supported
  - Gremlin for property graph, Xquery for XML, SQL for relational data model, Prolog
Some of these systems have free editions but with some feature or usage limitations (as of 2018):

- MarkLogic’s dev license is free for up to 1TB and 10 months max
- Stardog: community (10DB max with 25M triples/DB, 4 users), dev (no limits but 30 day trial)
- Allegrograph: free and dev have restrictions of 5M and 50M respectively
- Virtuoso and GraphDB: free but no clustering and no replication
- Blazegraph: free for a single machine (GPLv2)

All systems have commercial editions (Oracle is commercial only)
## Summary

<table>
<thead>
<tr>
<th>Triple store</th>
<th>Full-text search</th>
<th>Cloud-ready</th>
<th>Extra features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allegrograph</td>
<td>Integrated + solr</td>
<td>AMI</td>
<td></td>
</tr>
<tr>
<td>Blazegraph</td>
<td>Integrated + solr</td>
<td>AMI</td>
<td>Reification done right</td>
</tr>
<tr>
<td>GraphDB</td>
<td>Integrated + solr + elasticsearch (ent.)</td>
<td>AMI</td>
<td>RDF ranking</td>
</tr>
<tr>
<td>MarkLogic</td>
<td>Integrated</td>
<td>AMI</td>
<td>With Xquery, Javascript</td>
</tr>
<tr>
<td>Oracle</td>
<td>Integrated</td>
<td></td>
<td>Inline in SQL</td>
</tr>
<tr>
<td>Stardog</td>
<td>Integrated + Lucene</td>
<td>AMI</td>
<td>Integrity constraints, Explanations</td>
</tr>
<tr>
<td>Virtuoso</td>
<td>Integrated</td>
<td>AMI</td>
<td>Inline in SQL</td>
</tr>
</tbody>
</table>
Ontology-Based Data Access (OBDA)

- Relevant when you have an existing (relational) database and want to reason over it using an ontology

- The ontology models the domain, hides the structure of the data sources and enriches incomplete data

- The ontology is connected to the data sources via mappings that relate concepts and properties to SQL views over the sources

- Queries, expressed in SPARQL, are translated into the query language (usually SQL)

- State of the art is Ontop.

Some commercial solutions: Capsenta’s ultrawrap
APIs

- Two popular Java APIs to process and handle RDF data and SPARQL queries are:
  - RDF4J (formerly Sesame)
  - Apache Jena

- They both
  - provide a JDBC-like and REST-like APIs
  - storing, querying and reasoning capabilities
Native full-text search in Blazegraph

- Integration into SPARQL via the `bds:search` predicate

```sparql
prefix bds: <http://www.bigdata.com/rdf/search#>
SELECT ?s ?p ?o WHERE {
    ?o bds:search "dog" .
}
```
Allegrograph with Solr services

- Storage strategy for an RDF triple such as:
  \texttt{ex:someSubj ex:somePred "text to index"}
  - Tell Solr to associate "text to index" with a new \texttt{id}
  - Then, add a new triple into AllegroGraph:
    \texttt{ex:someSubj <http://www.franz.com/solrDocId> id}

- Now, you may write a SPARQL query such as:
  \begin{verbatim}
  PREFIX solr: <http://www.franz.com/ns/allegrograph/4.5/solr/>
  PREFIX franz: <http://franz.com/ns/allegrograph/4.5/>
  SELECT * WHERE {
    ?s solr:match 'medicate disastrous' .
    ?s franz:text ?text .
    ?s otherProperty ?other . }
  \end{verbatim}

- Solr can also be used from the API and the CLI
Plan

- Semantic Web technologies
- Storing and querying
- **Automated reasoning**
- Integrating data and knowledge
- Analytics
- Conclusion
Automated reasoning

Goals of this section

• Apprehend reasoning in a Knowledge Graph management context

• Understand the pros and cons of materialization and query reformulation

• Select the proper production-ready RDF store based on your reasoning needs
Reasoning

- The act of deriving implicit data from explicit one

- In a semantic context, deriving implies the use of logic-based processing

- This is based on a set of ontologies which are representing knowledge on the domain of discourse
Reasoning in data and knowledge bases

- Database management systems natively do not support reasoning
  - Some external solutions are possible but rare in production, e.g., Datalog, XSB, a Logic Programming systems that can connect to most RDBM systems

- Knowledge bases natively support inferences
  - All our 7 production-ready systems support reasoning
Supported ontology languages

- KB are concerned with a trade-off between ontology expressiveness and computational difficulty
- Intuitively, ontology languages with a tractable complexity are more amenable to reasoning in a KB system than ones with an exponential complexity
- The ‘tractables’ are:
  - RDFS
  - RDFS++ (RDFS with some OWL constructs)
  - OWLQL
  - OWLRL
  - OWL Horst
- Non tractable but decidable:
RDF Schema (RDFS)

- Concept hierarchy (using from the Semantic Sensor Network (SSN) ontology):

  - Explicit fact:
    QBE04 rdf:type SensingDevice

  - Implicit facts
    QBE04 rdf:type Sensor
    QBE04 rdf:type PhysicalObject
Property hierarchy (using SSN):

- Explicit fact:
  \[ \text{QBE04 ssn:onPlatform Platform\#2} \]

- Implicit fact
  \[ \text{QBE04 ssn:hasLocation Platform\#2} \]
RDFS++

- RDFS + sameAs + transitive and inverseOf properties

- Consider
  hasLocation inverseOf attachedSystem

- Explicit fact
  QBE04 ssn:onLocation Platform#2

- Implicit fact
  Platform#2 ssn:attachedSystem QBE04
RDFS++ (2)

- RDFS + sameAs + transitive and inverseOf properties

- Consider
  
  QBE04 sameAs QBS06

- Explicit facts
  
  QBE04 ssn:onLocation Platform#2
  QBE04 rdf:type SensingDevice

- Implicit facts
  
  QBS06 ssn:onLocation Platform#2
  QBS06 rdf:type SensingDevice
RDFS++ (3)

- RDFS + sameAs + transitive and inverseOf properties

Consider

QBE04 sameAs QBS06

Explicit facts

QBE04 ssn:onLocation Platform#2
QBE04 rdf:type SensingDevice

Implicit facts

QBS06 ssn:onLocation Platform#2
QBS06 rdf:type SensingDevice
QBS06 rdf:type Sensor
QBS06 rdf:type PhysicalObject
Platform#2 ssn:attachedSystem QBE04
RDFS++ (4)

- RDFS + sameAs + transitive and inverseOf properties

- Consider
  hasPart rdf:type TransitiveProperty

- Explicit facts
  Entity1 hasPart Entity2
  Entity2 hasPart Entity3

- Implicit fact
  Entity1 hasPart Entity3
Reasoning methods

- Two main reasoning methods
  - Materialization
  - Query rewriting
Materialization

- Make explicit all inferences in the store

- Pros:
  - Efficient query processing (no reasoning at query runtime)

- Cons:
  - Slow data loading
  - Data volume expansion
  - Tricky update management

- aka forward reasoning or saturation or closure
Materialization example

- **Explicit facts:**
  
  QBE04 ssn:onPlatform Platform#2  
  QBE04 rdf:type SensingDevice

- **The following facts would be added:**
  
  QBE04 rdf:type Sensor  
  QBE04 rdf:type PhysicalObject  
  QBE04 ssn:hasLocation Platform#2
Query rewriting

- Reformulate the original query such that all answers can be retrieved

- Pros:
  - No preprocessing overhead
  - No expansion of stored data volume
  - Easy update management

- Cons:
  - Slow query processing due to cost of reasoning at query runtime

- aka Backward reasoning or query reformulation
Query rewriting example

- Original query:
  SELECT ?x WHERE { ?x rdf:type PhysicalObject}

- Reformulated query:
  SELECT ?x WHERE {{ ?x rdf:type PhysicalObject}
  UNION
  SELECT ?x WHERE { ?x rdf:type Sensor}
  UNION
  SELECT ?x WHERE { ?x rdf:type SensingDevice}
### Production-ready and reasoning

<table>
<thead>
<tr>
<th>Triple store</th>
<th>Materialization</th>
<th>Query rewriting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allegrograph</td>
<td>OWLRL</td>
<td>RDFS++, Prolog</td>
</tr>
<tr>
<td>Blazegraph</td>
<td></td>
<td>RDFS, OWL Lite</td>
</tr>
<tr>
<td>GraphDB</td>
<td>RDFS, OWL Horst, OWLRL, OWLQL</td>
<td></td>
</tr>
<tr>
<td>MarkLogic</td>
<td></td>
<td>RDFS, RDFS++, OWL Horst</td>
</tr>
<tr>
<td>Oracle</td>
<td>RDFS, OWLRL, OWLQL</td>
<td></td>
</tr>
<tr>
<td>Stardog</td>
<td></td>
<td>All OWL2 (up to OWLDDL)</td>
</tr>
<tr>
<td>Virtuoso</td>
<td></td>
<td>RDFS++</td>
</tr>
</tbody>
</table>
Hybrid approaches

- Dynamic materialization used in AllegroGraph
  - RDFS++
  - Triples are generated on the fly and not saved in the triple store
- LiteMat
  - RDFS++
  - Semantic-aware encoding of the TBox which reduces the cost of query rewriting
Plan

- Semantic Web technologies
- Storing and querying
- Automated reasoning
- Integrating data and knowledge
- Analytics
- Conclusion
Goals

- Achieve global understanding of semantic integration:
  What are the main problems?
  Which approaches are in use?

- Understand the main features of some commercial and open source systems
Data integration

- is a core component of information technology

- enables the combination of data contained in multiple data sources

- has to deal with
  - discovering and representing mapping assertions between source schemata, e.g., database tables
  - answering queries using multiple data sources, e.g., using SQL
Semantic integration

- Bringing together diverse, possibly heterogeneous, sources of information and interrelating them by leveraging the semantic information that is embedded inside them

- Interrelation occur at the ontology/vocabulary level
  - Recall that ontologies aim for knowledge sharing

- For example, integrate data across DBpedia, Wikidata, or any other Linked Data sources
Why is it important?

- Combine data and knowledge from multiple sources to:
  - answer queries using multiple sources of data, e.g., in SPARQL
  - support interoperability between different systems and thus sharing knowledge
  - reason over multiple data sources using knowledge sources
Mapping ontologies is a hard problem

- Too many large ontologies to consider manual mappings

- Semantic integration has to deal with different levels of mismatches:
  - Ontology: syntax, expressiveness
  - Linguistic: terms used in ontology
  - Modeling: conventions, granularity
  - Domain: coverage
Two main approaches

- Existence of a shared ontology which is extended to relate external ontologies via some mappings

- No shared ontology is available:
  - Heuristics-based or machine learning techniques are used to relate ontologies
Shared ontologies

- Several types of ontologies:
  - **Top-level** ontologies formalize general notions (e.g., processes, events, time, space, physical objects, etc.). An example is DOLCE which aims at capturing “ontological categories underlying natural language and human common-sense”
  
  - **Domain** ontologies describe a specific domain in terms of concepts and properties
  
  - **Application** ontologies specify terms for a given application. They depend on a domain ontology
Shared ontologies (2)

- Top-level ontologies are designed to support ontology matching
  - If two ontologies extend the same top-level ontology then it is easier to find correspondences between them. The top-level ontology serves as a bridge.
Heuristics and ML approaches

- Heuristic approaches are usually based on one or a combination of structure, element or instance analysis. An example is the PROMPT Suite (developed by the Protégé team)

- Machine learning approaches can combine different learners using a probabilistic model to discover correspondences between ontologies. Examples are GLUE, FCAMerge
Systems

- Commercial products
  - Semaflora systems
  - TopQuadrant’s TopBraid
  - Cambridge Semantics

- Free, open-source solutions
  - RDF Refine
  - Silk
  - Limes
  - Karma
Commercial products

- Ontoprise was one of the early software suite created with Semantic Web standards in view.
  - Some components, including semantic integration tools, were acquired in 2012 by Semaflora systems.

- Since the early 2000s, TopQuadrant proposes a Semantic ecosystem around the TopBraid suite. It is composed of an advanced ontology editor, TopBraid Live to integrate data.

- Cambridge Semantics was founded in 2007. It proposes the Anzo suite which supports integration, cleaning and management of metadata.
LIMES

- LInk discovery framework for MEtrix Spaces
- Academic project with software maintenance
- Composed of several discovery approaches
  - Link discovery for approximation of similarity between instances
  - Machine learning (supervised and unsupervised)
- Easily configurable through files or a Graphical User Interface
A Linked data integration framework
With commercial support by the Eccenca start-up
Active since 2010, latest version is 2.7.1
Main features
- Generate links between related data items
- Apply data transformations to structured data sources (e.g., generate RDF triples from csv files)
- Link RDF triples to data sources on the Web (e.g., LOD)
KARMA

- An information integration tool
- Developed and maintained at University of South California (USC)
- Karma learns to recognize mapping of data to ontologies
- Provides a Graphical User Interface to interact with data sets and ontologies
Plan

- Semantic Web technologies
- Storing and querying
- Automated reasoning
- Integrating data and knowledge
- Analytics
- Conclusion
Goals

- Provide an overview of analytical jobs, including on Graphs
- Understand the main features of some commercial and open source systems
Classification of database applications

- Two kinds of processing:
  - Transactional
    - Ability to collect and manage data in a concurrent and fault tolerant manner
  - Analytical
    - Ability to create information from data
    - Analyze operational data to create reports and support decision making
Kinds of analytics

- Descriptive analytics: what happened?
  - *Did we run out of beer cans in store S last month?*

- Diagnostic analytics: why did it happen?
  - *Get contextual information that last month was the beginning of the soccer world cup*

- Predictive analytics: what will happen?
  - *Use external knowledge of coming sport events to know when to stock up beer cans in some stores*

- Prescriptive analytics: how can we make it happen?
  - *Sponsor sport events to sell more beer cans*
Classification of database applications

- Two kinds of processing:
  - Transactional -> OnLine Transactional Processing (OLTP)
    - Ability to collect and manage data
  - Analytical -> OnLine Analytical Processing (OLAP)
    - Ability to create information from data
    - Analyze operational data to create reports and support decision making

- Online means end-users expect fast answers to their queries
## OLTP vs OLAP

<table>
<thead>
<tr>
<th></th>
<th>OLTP</th>
<th>OLAP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Read pattern</strong></td>
<td>Few tuples fetched by a key (index)</td>
<td>Large number of records are accessed</td>
</tr>
<tr>
<td><strong>Write pattern</strong></td>
<td>High frequency of small transactions</td>
<td>Low frequency of very large transactions, Bulk import, Data streams</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td><strong>Dataset sizes</strong></td>
<td>GB to few TB</td>
<td>TB to several PB</td>
</tr>
<tr>
<td><strong>End-users</strong></td>
<td>Many non specialized</td>
<td>Few specialized (Business analysts, Managers)</td>
</tr>
</tbody>
</table>
Data storage

- “A Data Warehouse stores and manages data. OLAP transforms Data Warehouse data into strategic information” (OLAP Council)
Analytical processing

- Performed by OLAP and/or data mining components
- There are several kinds of OLAP systems
  - Multidimensional OLAP (MOLAP)
  - Relational OLAP (ROLAP)
  - Hybrid OLAP (HOLAP)
  - And the emerging Graph OLAP (GOLAP)
ROLAP - Schema

- Distinction between fact (e.g., item sales) and dimension (e.g., store, customer) tables
ROLAP - Schema

- Distinction between **fact** (e.g., item sales) and **dimension** (e.g., store, customer) tables

- Schemata
  - Star schema
  - Snowflake schema (multiple levels of dimension tables)
  - Constellation schema (multiple fact tables)
ROLAP – SQL extensions
## OLAP - Views

<table>
<thead>
<tr>
<th></th>
<th>Virtual views</th>
<th>Materialized views</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB system</strong></td>
<td>OLTP</td>
<td>Data Warehouse</td>
</tr>
<tr>
<td><strong>Structure</strong></td>
<td>Logical table</td>
<td>Logical table</td>
</tr>
<tr>
<td><strong>Persisted in DBMS</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Cost of updates</strong></td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>
OLAP Storage

- Column-oriented storage
  - Stores tuples column-wise (not row-wise)
  - More efficient for select queries retrieving a subset of the tuples of some tuples
- Limited on writing operations (have to access several files to insert/delete a tuple)
OLAP – Data storage

- Compression
  - More possibilities to compress data on columnar storage than row storage
  - Solutions
    - Run-length encoding
    - Delta encoding
    - Bit encoding

Original data

<table>
<thead>
<tr>
<th>Time</th>
<th>Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00</td>
<td>24.5</td>
</tr>
<tr>
<td>12:05</td>
<td>24.6</td>
</tr>
<tr>
<td>12:10</td>
<td>24.5</td>
</tr>
<tr>
<td>12:15</td>
<td>24.4</td>
</tr>
<tr>
<td>12:20</td>
<td>24.3</td>
</tr>
</tbody>
</table>

Compressed data (value, offset, length)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>M</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>M</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>M</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>M</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>M</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>M</td>
<td>8</td>
</tr>
</tbody>
</table>

Compressed data

<table>
<thead>
<tr>
<th>Time</th>
<th>Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00</td>
<td>+5</td>
</tr>
<tr>
<td></td>
<td>+5</td>
</tr>
<tr>
<td></td>
<td>+5</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Compression
6 reasons why RDF data management is OLAP

1. Use cases emphasized by prod-ready systems

- Curating data for advanced analytics
- ERP Integration
- Mainframe migration
- Managing regulated data
- Smart metadata management
- Advanced search and discovery
- Analytics/BI
- Fraud detection
- Recommendations
6 reasons why RDF data management is OLAP

1. Use cases emphasized by prod-ready systems

- Curating data for advanced analytics
- ERP Integration
- Mainframe migration
- Managing regulated data
- Smart metadata management
- Advanced search and discovery
- Analytics/BI
- Fraud detection
- Recommendations
6 reasons why RDF data management is OLAP

2. Study in large graph processing [1]

- Analytics is the task where end-users are spending the most hours (more than testing, debugging, maintenance, ETL and cleaning).

- Top graph computations performed: finding connected components, neighborhood queries, finding shortest paths, subgraph matching (i.e., SPARQL), ranking and centrality scores, reachability queries.

3. Frequency of Linked Data Updates

- In practice, Linked Data publishing is periodic bulk loading, e.g., DBPedia, Yago, health science

- Periodically processing very large transactions makes more sense than supporting (very) frequently small transactions
- Example: load a new version of DBPedia (billions of assertions every six months)
6 reasons why RDF data management is OLAP

4. Features of the SPARQL query language

- SPARQL 1.0 (Jan 2008) -> SPARQL 1.1 (March 2013)
  
  - 5 years to get update operations (INSERT and DELETE) in W3C recommendation!
  
  - It did not get that long to get update support in SQL, Cypher, Gremlin
6 reasons why RDF data management is OLAP

5. Implementation aspects

- RDF Stores are mostly OLTP because it is easier to deal with ACIDity than efficient data partitioning
  - ACID is an engineering/programming challenge
  - Graph partitioning is a “theoretical” (NP-hard) problem
6 reasons why RDF data management is OLAP

6. Reasoning issues

- It is hard to efficiently maintain the consistency of an RDF store in the face of a high transaction rate and support for reasoning.
- With materialization, it may be impossible to support a high rate of transactions.
- With query rewriting, query answering will be very slow.
Anzo Graph

- Originally Sparql City, acquired By Cambridge Semantics in 2016
  - Designed by an expert of OLAP systems (Applix, Netezza, Paracel)
- Part of Anzo smart data lake and also available standalone
- Master-slave, shared-nothing
- RDFS+ inferences
- Rich library for SQL and excel-like functions, C++ API for UDF
- Extends SPARQL 1.1 with
  - OLAP ops: CUBE, ROLLUP, GROUPING SET
  - Named queries, views
- Graph algorithms: centrality, community detection, path finding
Anzo Graph

- Defined as both a Data Warehouse and a GOLAP system

- AnzoGraph can retrieve data from
  - Different external sources including OLTP RDF stores (partnership with MarkLogic)
  - Anzo Smart Data Lake

- Commercial with 60 days trial version
- Sufficient to support descriptive and diagnostic analytics
- What about predictive and prescriptive analytics?
Plan

- Semantic Web technologies
- Storing and querying
- Automated reasoning
- Integrating data and knowledge
- Analytics
- Conclusion
Summary

- RDF database management is more an OLAP than an OLTP market
- But most RDF Stores are OLTP. Only AnzoGraph can be considered GOLAP
- More research and implementation are needed on issues pertaining to:
  - Query processing
    - Integrating SQL-like OLAP operations and graph algorithms in SPARQL
  - Reasoning
  - Allowing prediction
Trends in semantic data management system

- Data and processing distribution, fault tolerance
- Handling both batch and stream processing
- Supporting reasoning services
- Performing description, diagnostic, predictive and prescriptive analytics:
Design of such a system

- The Data Warehouse and MapReduce worlds are merging
  - Teradata, Cloudera/Hortonworks, MapR + Arcadia Data

- Frameworks such as **Apache Spark** or Apache Flink are good candidates to design an OLAP RDF store
  - Libraries available for
    - query processing (SparkSQL / Table API)
    - stream processing (Structure streaming / Datastream API)
    - Graph processing (GraphX / Gelly)
    - Machine learning (MLib / FlinkML)
Spark’s GraphFrame

- Equivalent of Spark’s DataFrame for distributed, fault tolerant graph processing
- Equipped with triplets API and graph pattern matching (Cypher-like)
- Supports view materialization, query optimization (via Catalyst)
- Able to manipulate graph algorithms (pagerank, connected components, triangle count, etc.)
- Easy to design an app that manipulates MLlib, SparkSQL, Structured Streaming
- Limits: no index (brute force), no incremental graph updates
Moving forward

- HTAP: Hybrid Transactional Analytical processing
  - Breaking the wall between OLTP and OLAP
  - Allows advanced analytics on real time transaction data
- Challenges:
  - Being efficient for both OLTP and OLAP operations
  - Lack of experience on these systems
- Technical solutions
  - One copy of the data
  - In-memory
Plan

- Semantic Web technologies
- Storing and querying
- Automated reasoning
- Integrating data and knowledge
- Analytics
- Conclusion
Tutorial wrap-up

- Mature tools in the Semantic data management ecosystem
- High quality open source systems are available (Apache Jena, Protégé, RDF stores, HermiT, etc.)
- Large companies are already present or entering the market (Oracle, IBM, Microsoft, MarkLogic, etc.)
Tutorial wrap-up (2)

- Still many open issues to address
  - Efficient partitioning and RDF storage
    - Also an issue for LPG stores (Neo4J)
  - Reasoning and high performance query answering
  - SPARQL query processing and analytics
  - Understanding, visualizing very large graphs
  - Data cleansing
  - Streaming RDF data
    - C-SPARQL, integration with Apache Spark, Flink