Introduction to Knowledge Graphs and Semantic Web Technologies

SPARQL Endpoints and Triple Stores

Olaf Hartig

olaf.hartig@liu.se

Acknowledgement: Some slides in this slide set are adaptations of slides of Olivier Curé (University of Paris-Est Marne la Vallée, France)

LINKÖPING UNIVERSITY

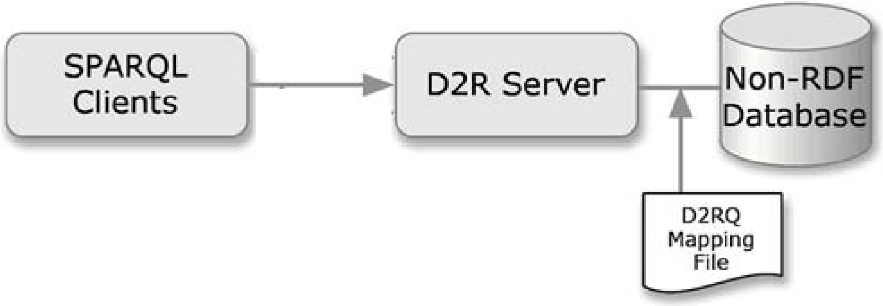
What is a SPARQL Endpoint?

- HTTP-based Web service that supports the SPARQL protocol (which is part of the family of W3C standards for SPARQL)
- Provides SPARQL-based access to a server-side dataset
  - send your SPARQL query → receive the query result
- Dataset may be
  - actual RDF data stored and maintained in a triple store*
  - a *virtual* RDF view of any other form of database (e.g., relational)

*DBMS for RDF data

RDF Virtualization Component

- Rewrites SPARQL queries to the source language (e.g., SQL)
- Relies on a mapping from the underlying database to RDF
  - R2RML is a W3C standard for mapping relational DBs to RDF
  - RML extends R2RML for other forms of source data (e.g., JSON)
- Example systems
  - Ontop https://ontop-vkg.org/
  - Morph-RDB https://morph.oeg.fi.upm.es/tool/morph-rdb
  - Sparqlify http://aksw.org/Projects/Sparqlify.html
  - D2R Server http://d2rq.org/



Classification of *RDF Triple Stores**

**Triple store* = DBMS for RDF data

RDF Storage

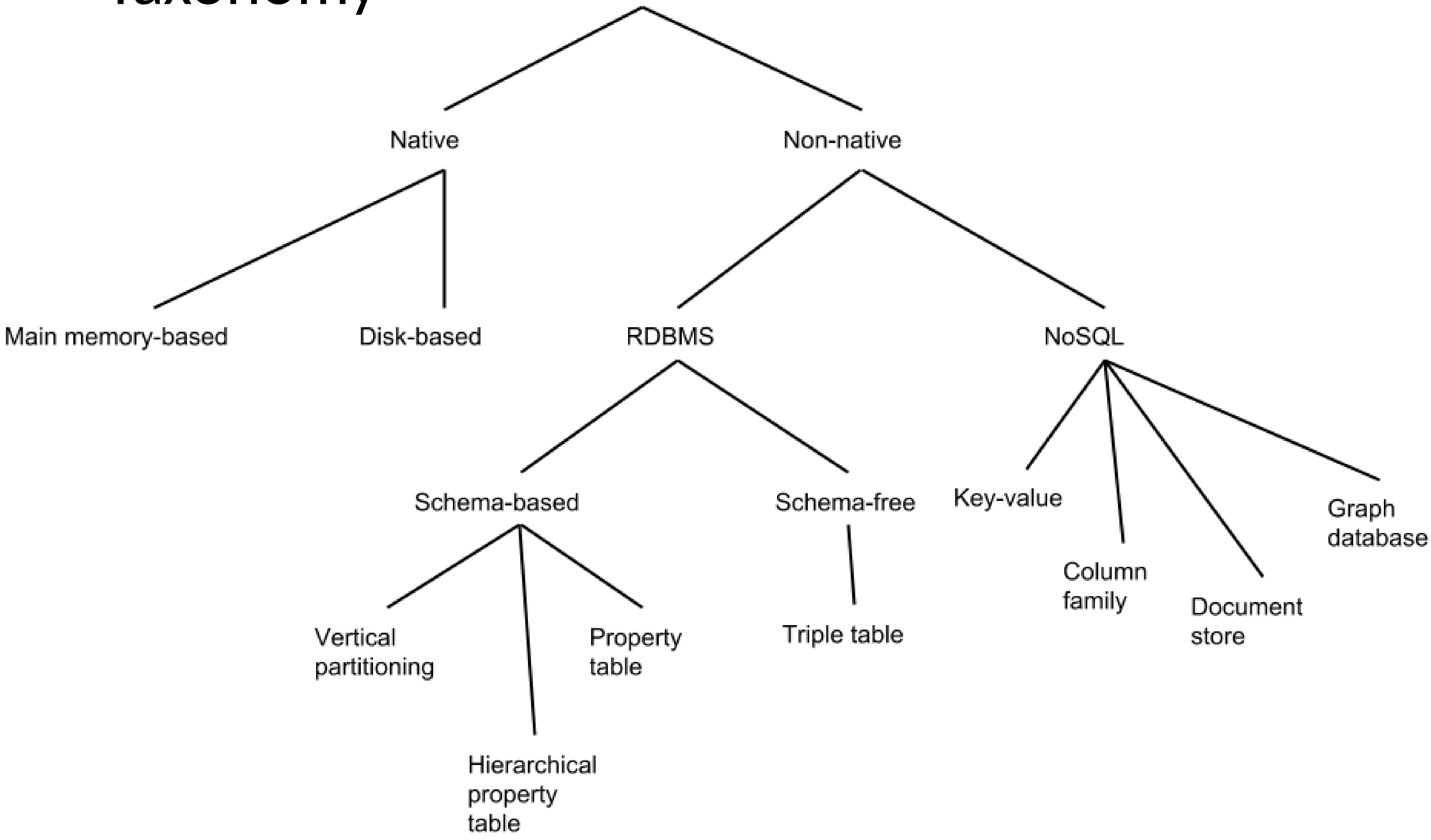
- RDF is a logical data model and, thus, does not impose any physical storage solution

- Existing triple stores are either
  - designed from scratch (“native”)

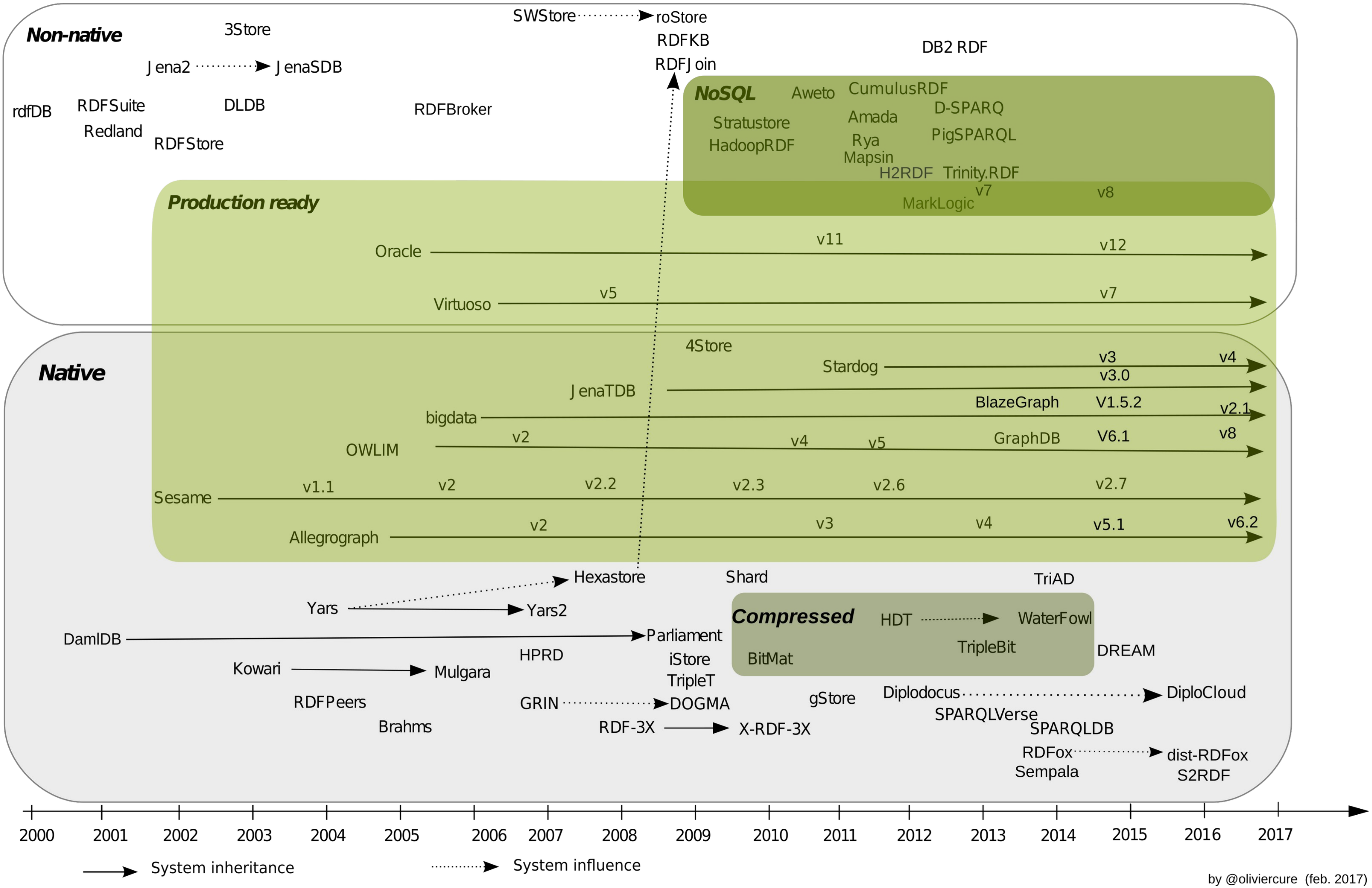
    or

  - based on an existing DBMS
    - Relational model, e.g., PostgreSQL
    - NoSQL, e.g., Cassandra

Taxonomy


Native
Non-native
Main memory-based
Disk-based
RDBMS
NoSQL
Schema-based
Schema-free
Key-value
Graph database
Column family
Document store
Vertical partitioning
Property table
Triple table
Hierarchical property table

Timeline of Triple Store Proposals



Non-native

3Store

Jena2 → JenaSDB

rdfDB

RDFSuite

Redland

RDFStore

DLDB

RDFBroker

SWStore → roStore

RDFKB

RDFJoin

DB2 RDF

NoSQL

Aweto

CumulusRDF

D-SPARQ

Stratustore

Amada

HadoopRDF

Rya

PigSPARQL

Mapsin

H2RDF

Trinity.RDF

Production ready

MarkLogic v7 v8

Oracle v11 v12

Virtuoso v5 v7

4Store

Stardog v3 v4

JenaTDB v3.0

bigdata BlazeGraph V1.5.2 v2.1

OWLIM v2 v4 v5 GraphDB V6.1 v8

Sesame v1.1 v2 v2.2 v2.3 v2.6 v2.7

Allegrograph v2 v3 v4 v5.1 v6.2

Native

Hexastore

Shard

TriAD

Yars → Yars2

DamlDB → Parliament

Compressed

HDT → WaterFowl

TripleBit

BitMat

DREAM

HPRD

Kowari → Mulgara

iStore

TripleT

RDFPeers

GRIN → DOGMA

gStore

Diplodocus → DiploCloud

SPARQLVerse

Brahms

RDF-3X → X-RDF-3X

SPARQLDB

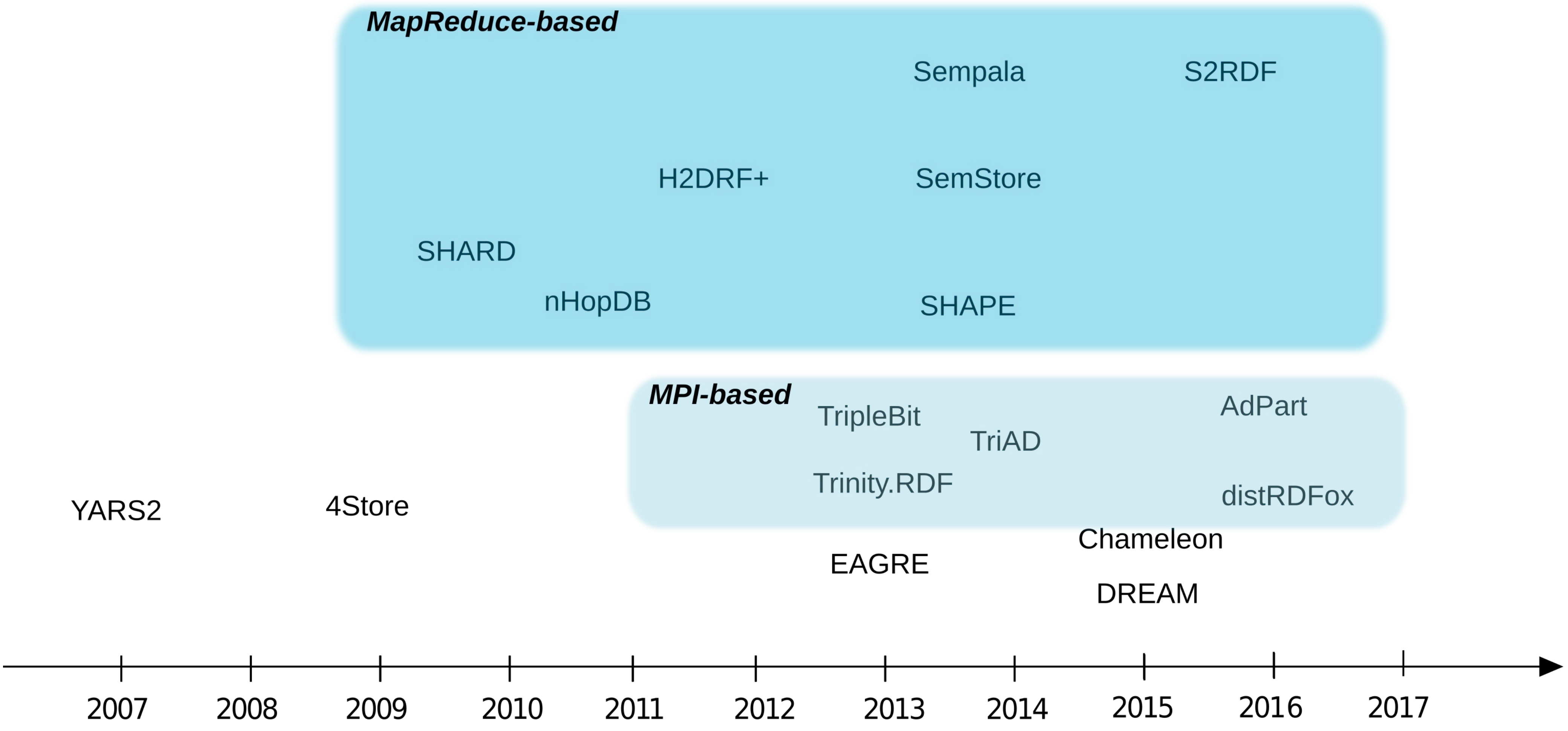
RDFox → dist-RDFox

Sempala

S2RDF

by @oliviercure (feb. 2017)

Prototypes of Distributed Triple Stores



MapReduce-based

Sempala

S2RDF

H2DRF+

SemStore

SHARD

nHopDB

SHAPE

MPI-based

TripleBit

TriAD

AdPart

Trinity.RDF

distRDFox

YARS2

4Store

Chameleon

EAGRE

DREAM

by @oliviercure (feb. 2017)

Production-Ready Triple Stores

Overview

Name
Allegrograph
Blazegraph
GraphDB
MarkLogic
Oracle
Stardog
Virtuoso

Transactions with ACID Properties

Name	
Allegrograph	✓
Blazegraph	✓
GraphDB	✓
MarkLogic	✓
Oracle	✓
Stardog	✓
Virtuoso	✓

- **A**tomicity: a transaction (TA) is an atomic unit of processing; it is either performed in its entirety or not performed at all
- **C**onsistency preservation: a correct execution of a TA must take the DB from one consistent state to another
- **I**solation: even if TAs are executing concurrently, they should appear to be executed in isolation; that is, their final effect should be as if each TA was executed alone from start to end
- **D**urability: once a TA is committed, its changes applied to the database must never be lost due to subsequent failure

Cluster Setups

Name
Allegrograph ✓
Blazegraph ✓
GraphDB ✓
MarkLogic ✓
Oracle ✓
Stardog ✓
Virtuoso ✓

- Replication: mostly master-slave,
  some master-master
- Partitioning: range, hash

Support for other Data Models (besides RDF)

Name	
Allegrograph	
Blazegraph	✓
GraphDB	✓
MarkLogic	✓
Oracle	✓
Stardog	✓
Virtuoso	✓

- Relational Model (with SQL)
  - Virtuoso, Oracle
- XML (with XQuery)
  - MarkLogic, Virtuoso
- Document Model
  - MarkLogic
- Property Graphs (with Gremlin)
  - Blazegraph, GraphDB, Stardog

Licenses

Name
Allegrograph
Blazegraph
GraphDB
MarkLogic
Oracle
Stardog
Virtuoso

- Most of these systems have a free-to-use edition, some even have a feature-limited free software version (open source)
- All have commercial editions

Full-Text Search Support

Name	Full-text search
Allegrograph	Integrated + Solr
Blazegraph	Integrated + Solr
GraphDB	Integrated + Solr + ElasticSearch (enterp.)
MarkLogic	Integrated
Oracle	Integrated
Stardog	Integrated + Lucene
Virtuoso	Integrated

Cloud Readyness

Name	Full-text search	Cloud-ready
Allegrograph	Integrated + Solr	AMI
Blazegraph	Integrated + Solr	AMI
GraphDB	Integrated + Solr + ElasticSearch (enterp.)	AMI
MarkLogic	Integrated	AMI
Oracle	Integrated	
Stardog	Integrated + Lucene	AMI
Virtuoso	Integrated	AMI

AMI: Amazon Machine Image

Other, cloud-native options:

Amazon Neptune

DYDRA

Automated Reasoning in Triple Stores

Atomated Reasoning?

- Definition of properties and classes in RDF vocabularies based on an ontology language such as RDFS or OWL (covered later)
- Allows for inferring additional data from existing data that uses these properties and classes
- Example:
  - assume foaf:knows is defined as a symmetric relationship
  - then, given the RDF triple (ex:olaf, foaf:knows, ex:eva), we can infer the triple (ex:eva, foaf:knows, ex:olaf)
- Example:
  - assume ex:Man is defined as a subclass of foaf:Person
  - then, given the triple (ex:olaf, rdf:type, ex:Man), we can infer (ex:olaf, rdf:type, foaf:Person)

Approach 1: *Materialization*

a.k.a. *forward reasoning* or *closure*

- Idea: make explicit all inferences in the triple store
- Pros:
  - efficient query processing (no reasoning at query runtime)
- Cons:
  - slow data loading
  - data volume expansion
  - tricky update management

Approach 2: *Query Rewriting*

a.k.a. *backward reasoning* or *query reformulation*

- Idea: 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:
  - slower query processing due to cost of reasoning at runtime

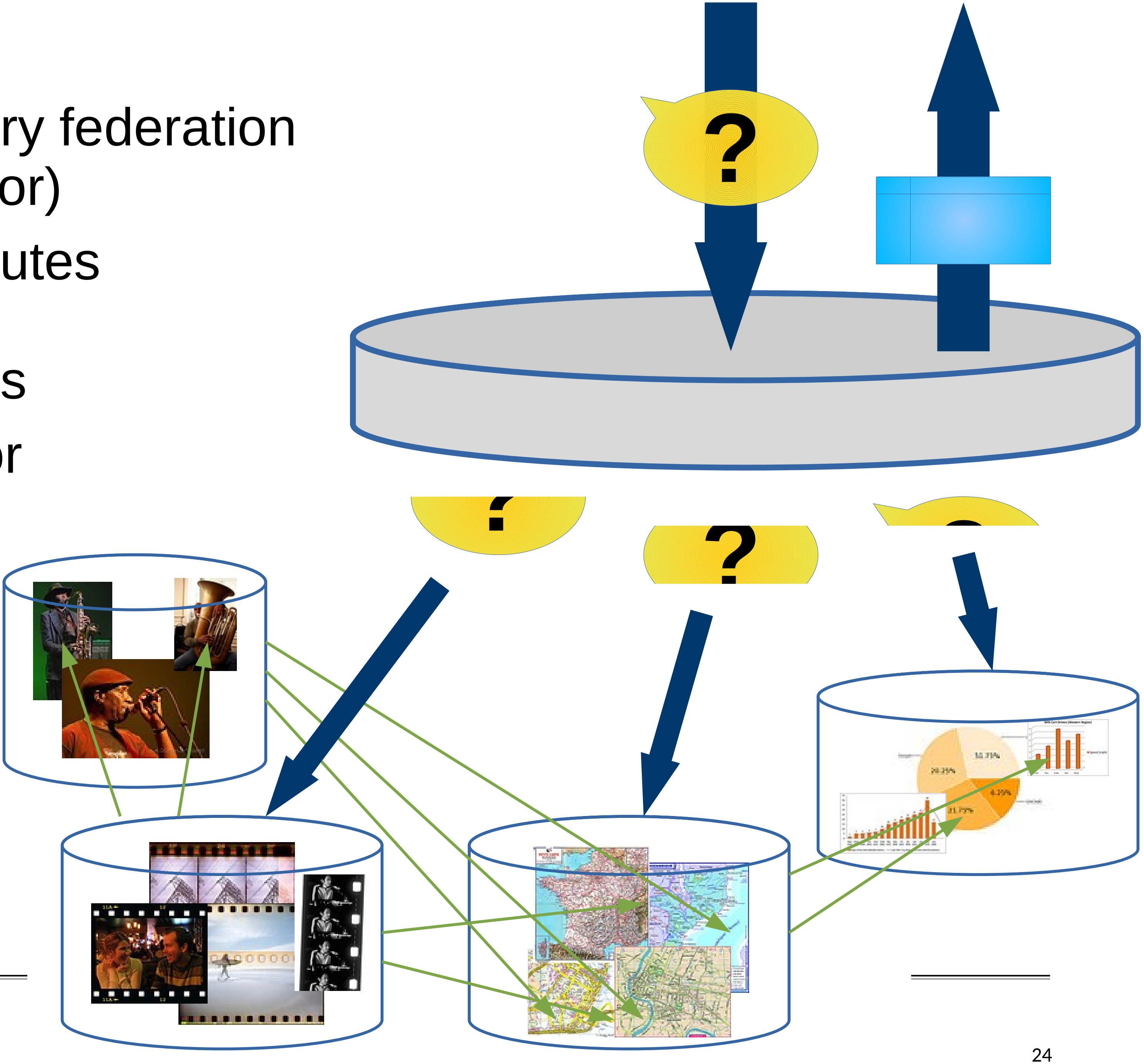
Reasoning in the Production-Ready Systems

<table>
<tr><th>Triple store</th><th>Materialization</th><th>Query rewriting</th></tr>
<tr><td><b>Allegrograph</b></td><td>OWLRL</td><td>RDFS++, Prolog</td></tr>
<tr><td><b>Blazegraph</b></td><td colspan="2">RDFS, OWL Lite</td></tr>
<tr><td><b>GraphDB</b></td><td>RDFS, OWL Horst, OWLRL, OWLQL</td><td></td></tr>
<tr><td><b>MarkLogic</b></td><td></td><td>RDFS, RDFS++, OWL Horst</td></tr>
<tr><td><b>Oracle</b></td><td>RDFS, OWLRL, OWLQL</td><td></td></tr>
<tr><td><b>Stardog</b></td><td colspan="2">All OWL2</td></tr>
<tr><td><b>Virtuoso</b></td><td></td><td>RDFS++</td></tr>
</table>

Federated Query Processing

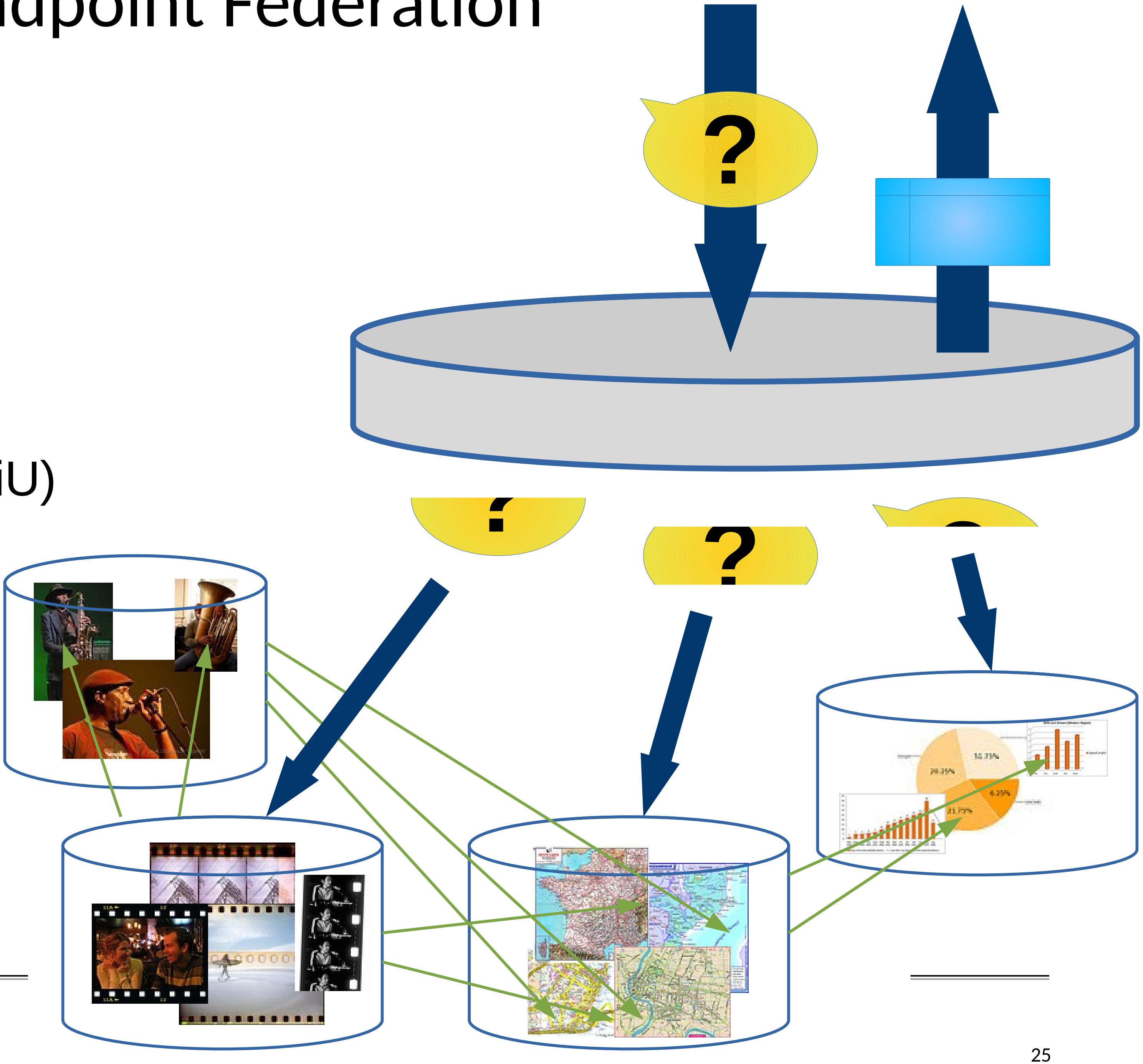
Idea

- Querying a query federation service (mediator)
- Mediator distributes sub-queries to relevant sources
- Finally, mediator combines sub-results



SPARQL Endpoint Federation

- Prototypes:
  - FedX
  - SPLENDID
  - ANAPSID
  - CostFed
  - HeFQUIN (@LiU)
  - etc.



SPARQL 1.1 Federation Extension

- SERVICE pattern in SPARQL 1.1
  - Explicitly specify query patterns whose execution must be distributed to a remote SPARQL endpoint

```
SELECT ?v ?ve WHERE
{
  SERVICE <http://volcanos.example.org/query> {
     ?v rdf:type umbel-sc:Volcano ;
        p:location dbpedia:Italy .
  }
  SERVICE <http://volcano-eruptions.org/sparql> {
     ?v p:lastEruption ?ve .
  }
}
```

www.liu.se

LINKÖPING
UNIVERSITY