YarcData™
A DIVISION OF CRAY INC.

uRiKA and Graph Analytics

uRiKA == universal RDF integration Knowledge Appliance

Agenda

- **A Quick Overview**
- Converting Data into RDF
- Ingesting Data into uRiKA
- Querying uRiKA
- Visualizing Results
- Summary

uRiKA – YarcData Hardware and Software

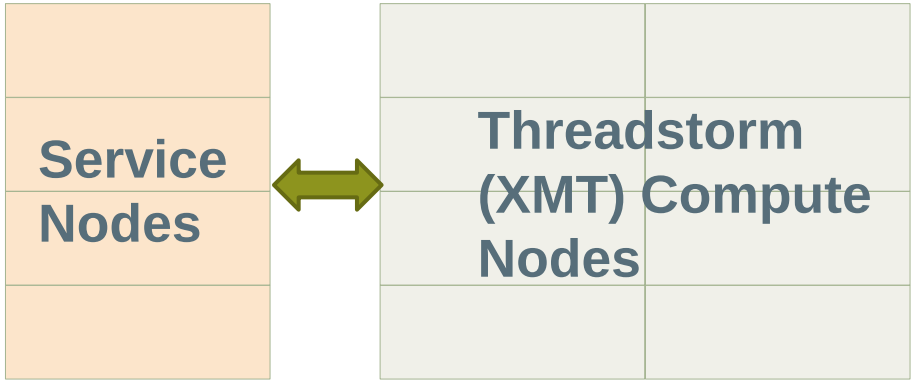
- **Cray Hardware Engine**
  - Originally designed for deep analysis of large datasets
  - Very large *scalable* shared memory
    - Architecture can support 512TB shared memory
    - Typical systems are 2 TB to 32 TB
  - Multithreading
    - Unique highly multithreaded architecture
    - 128 hardware threads per processor
    - Extreme parallelism, hides memory latency

- **Multithreaded Graph Database**
  - Highly parallel in-memory RDF quad store
  - High performance inference engine
  - High performance parallel I/O

- **Industry Standard Front End**
  - Based on Jena open source semantic DB
  - WS02 application framework
  - All standard SuSE Linux infrastructure and languages

Hybrid Graph Appliance: Ease of Use AND Performance!

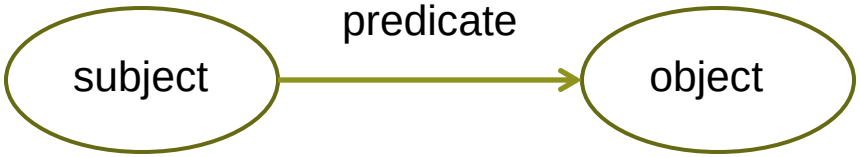
- Proven Cray infrastructure
- Cray XT5 3D Torus High Speed Interconnect



- **Service Nodes:**
  - AMD Opteron Processors, SuSe Linux
  - **Open environment based on Jena**
- **Threadstorm Processors:**
  - Multithreaded processors, MTK OS
  - **Pre-programmed by YarcData**

Emerging Web 3.0 Standards: RDF and SPARQL

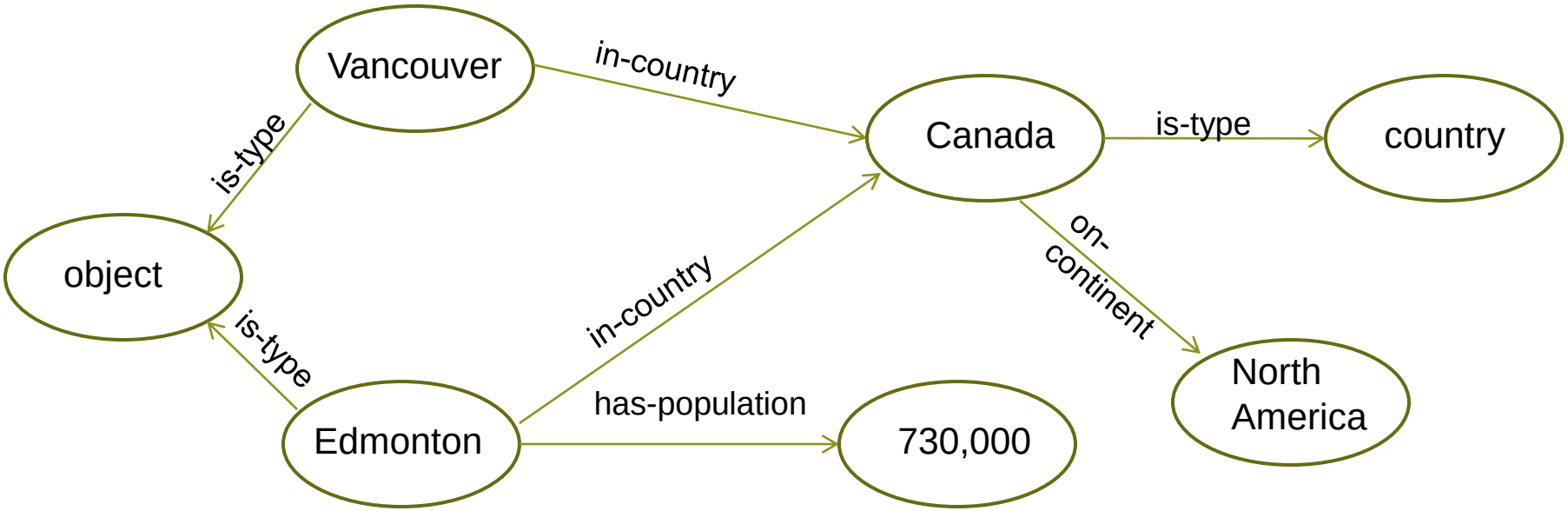
- **Resource Description Framework (RDF)**
  - Designed to enable semantic web searching and integration of disparate data sources
  - W3C standard formats
  - Every datum represented as subject/predicate/object
    - Ideally with each of those expressed with a URI
  - Standard ontologies in some domains
    - *e.g.*, Open Biological and Biomedical Ontologies (OBO)
  - Examples:




```
<ncbitax:NCBITaxon_840261>   rdf:type              owl:Class
<ncbitax:NCBITaxon_195644>   rdfs:subClassOf      <ncbitax:NCBITaxon_185881>
<ncbitax:NCBITaxon_816681>   rdfs:label           "Characiformes sp. BOLD:AAG5151"@en
```

Semantic Database of RDF Triples

- RDF triples databases are inherently graphical
- Some researchers call semantic databases "semantic graph databases"



Emerging Web 3.0 Standards: RDF and SPARQL

- **SPARQL Protocol and RDF Query Language (SPARQL)**
  - Enables matching of graph patterns in the semantic DB
  - Reminiscent of SQL

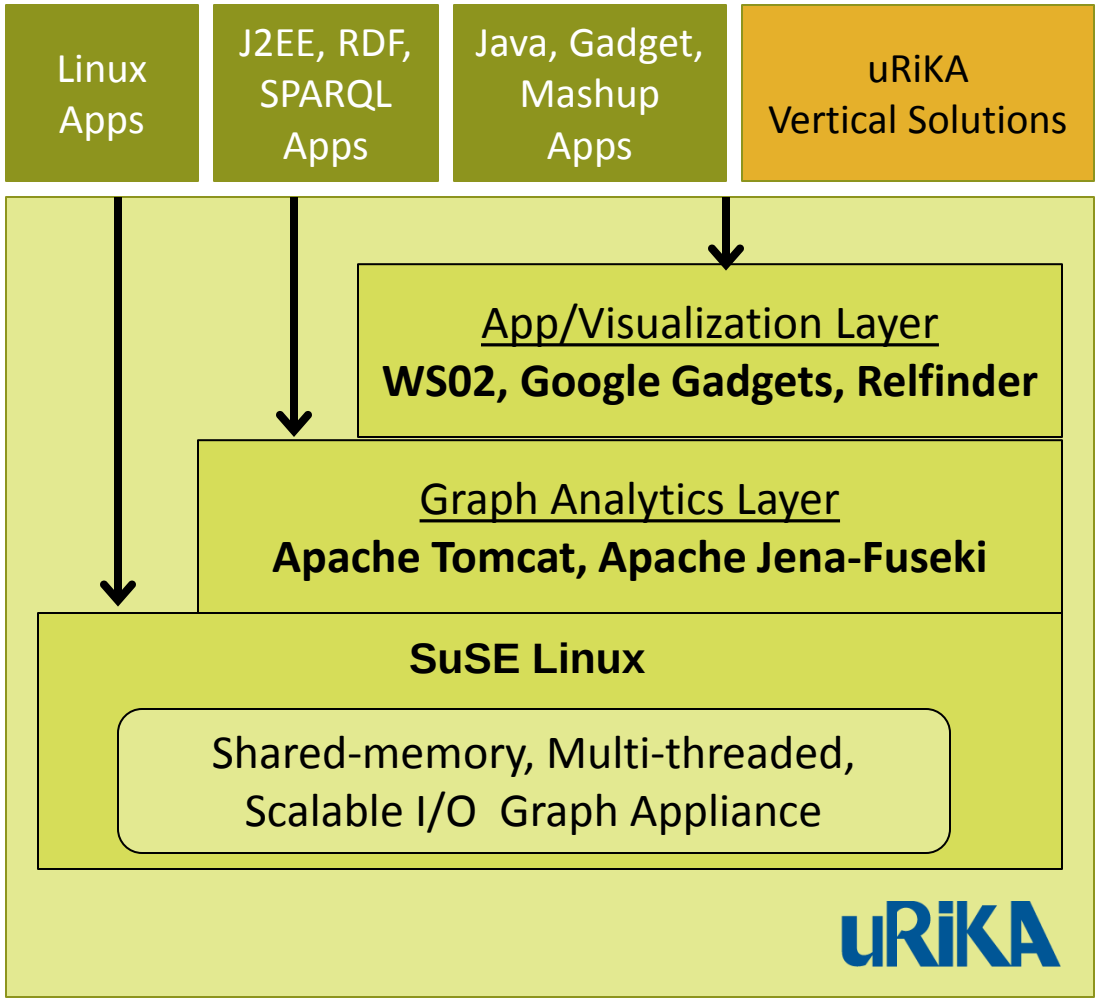
```
# Lehigh University BenchMark (LUBM) Query 9
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y, ?Z
WHERE
{?X rdf:type ub:Student .
 ?Y rdf:type ub:Faculty .
 ?Z rdf:type ub:Course .
 ?X ub:advisor ?Y .
 ?Y ub:teacherOf ?Z .
 ?X ub:takesCourse ?Z}
```

PREFIX == shorthand for a URI

variables to be returned from the query

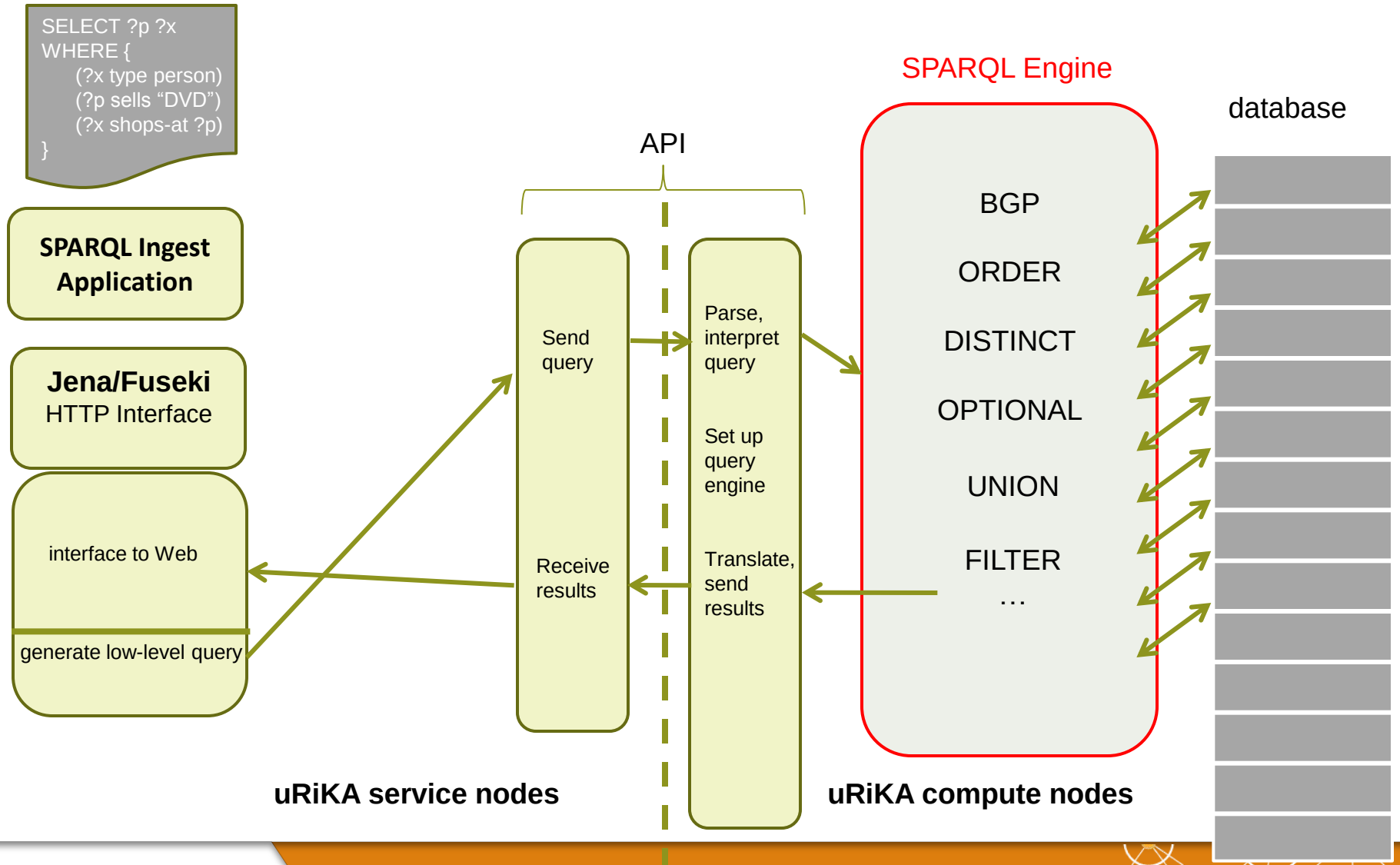
"find sets of (X, Y, Z) with a subject X of type Student, a subject Y of type Faculty, and a subject Z of type Course, where X is an advisee of Y, Y teaches course Z, and X takes course Z"

uRiKA Software Stack

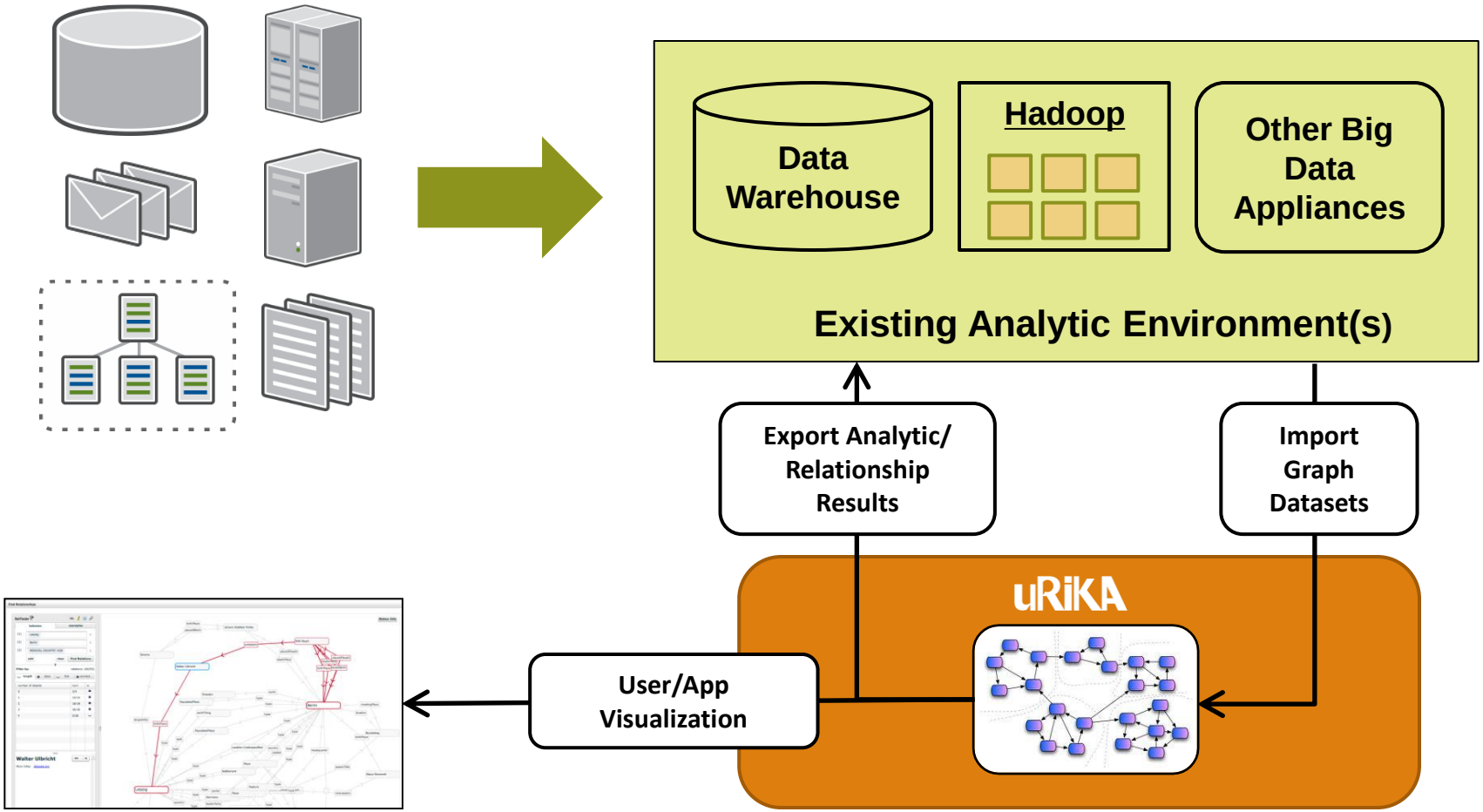


- **Industry-standard, Open-source Software Stack**
  - Linux, Java, Apache, WS02, Gadgets, Mashups…
- **Reusable Existing Skillsets**
  - OSGI, App Server, SOA, ESB, Web toolkit…
- **No Lock-in**
  - All applications and artifacts built on uRiKA can be run on other platforms

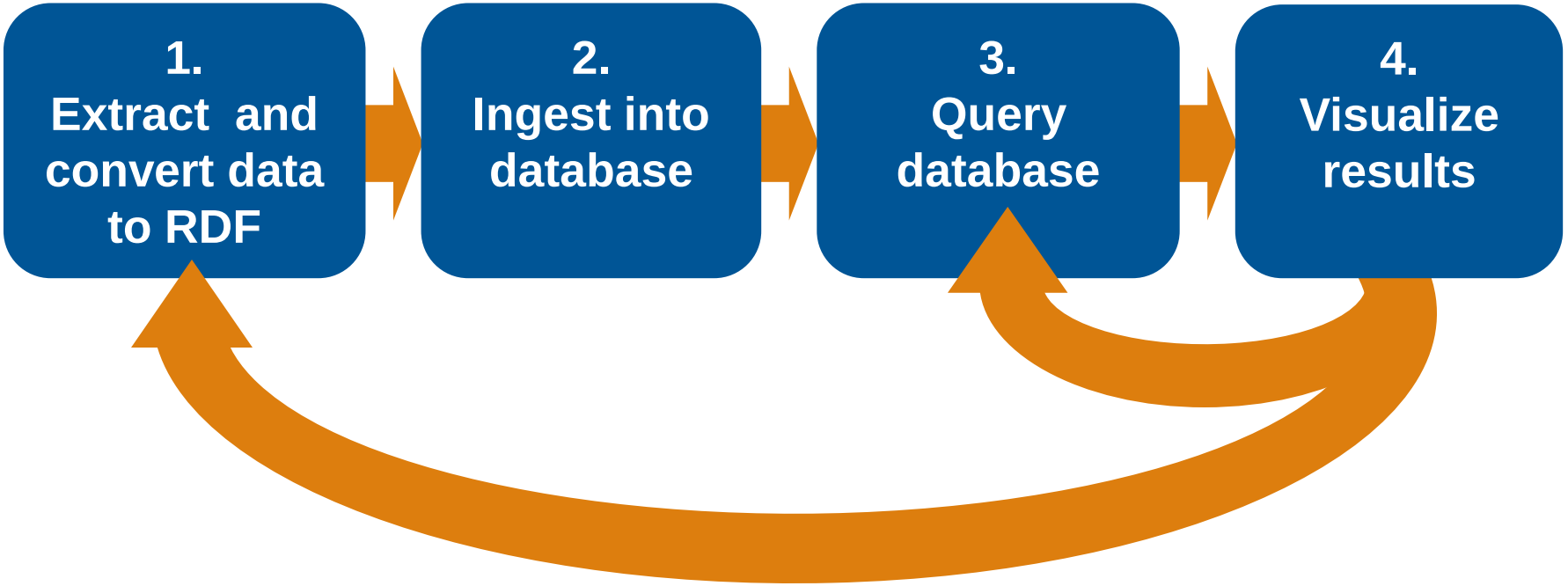
uRiKA High Level Functional View



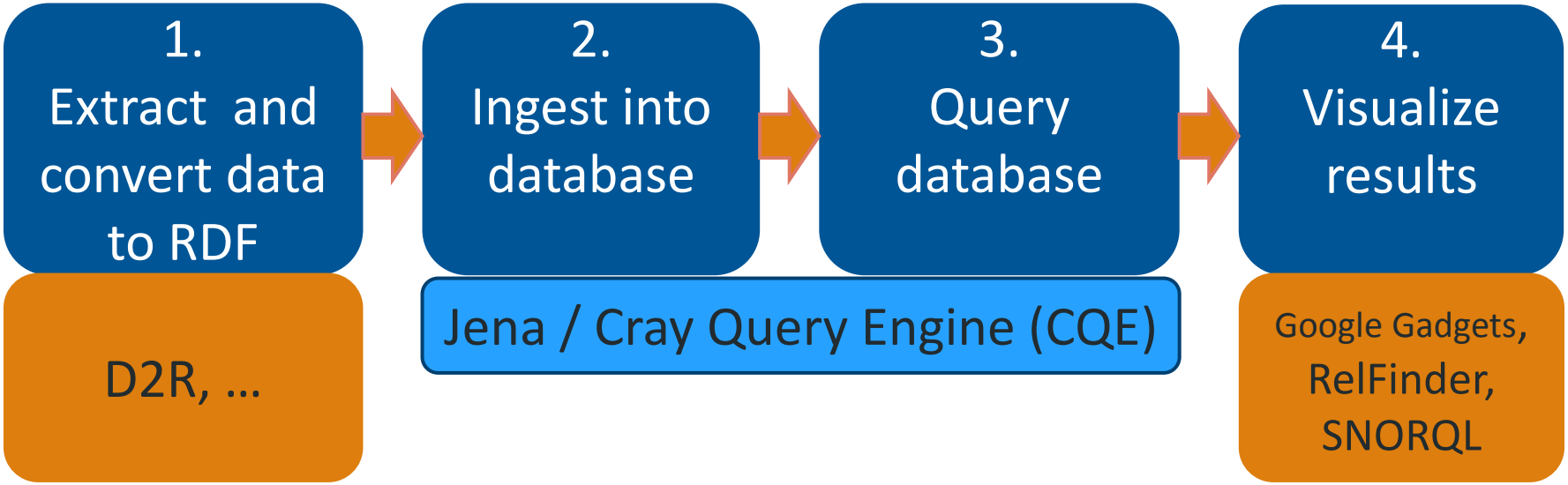
uRiKA complements existing Data Warehouse/Hadoop environment by offloading Graph Analytics



Workflow


1.
Extract and convert data to RDF
2.
Ingest into database
3.
Query database
4.
Visualize results

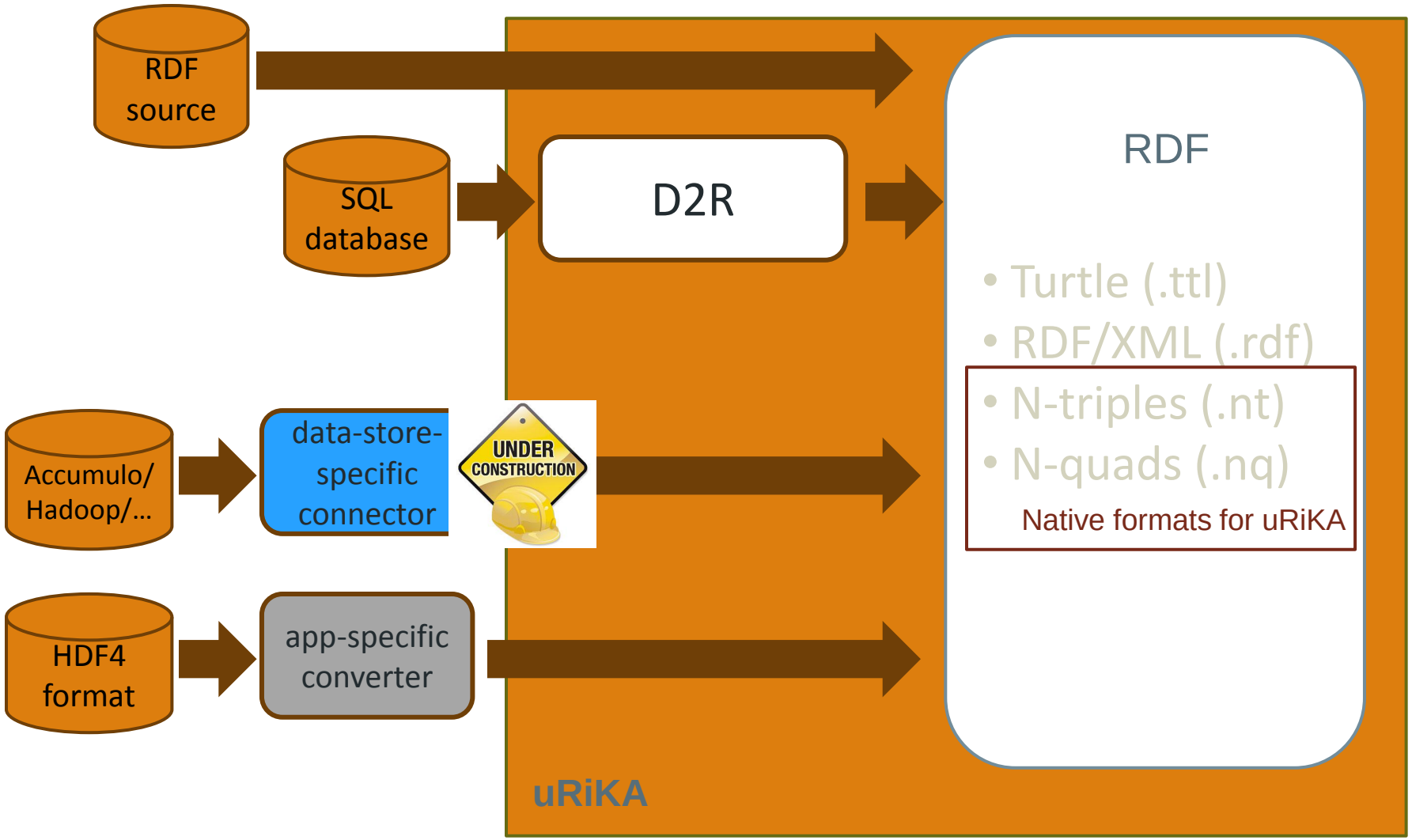
Implementation



Agenda

- A Quick Overview
- **Converting Data into RDF**
- Ingesting Data into uRiKA
- Querying uRiKA
- Visualizing Results
- Summary

Getting Data into RDF Format



… But Conversion is Not Enough

- Primary RDF goal is to be able to fuse data from different sources
- Ontologies (definitions of entities and relationships) must either be the same or be mappable onto each other
- Same: Use common ontologies
  - Generic: *e.g.*, RDF, OWL, Dublin Core
  - Social Networks: FOAF
  - Biology: *e.g.*, OBO, NCBI
- Map: Use ontology-mapping tools (*e.g.*, Top Braid Composer)

Converting Data among Different RDF Formats

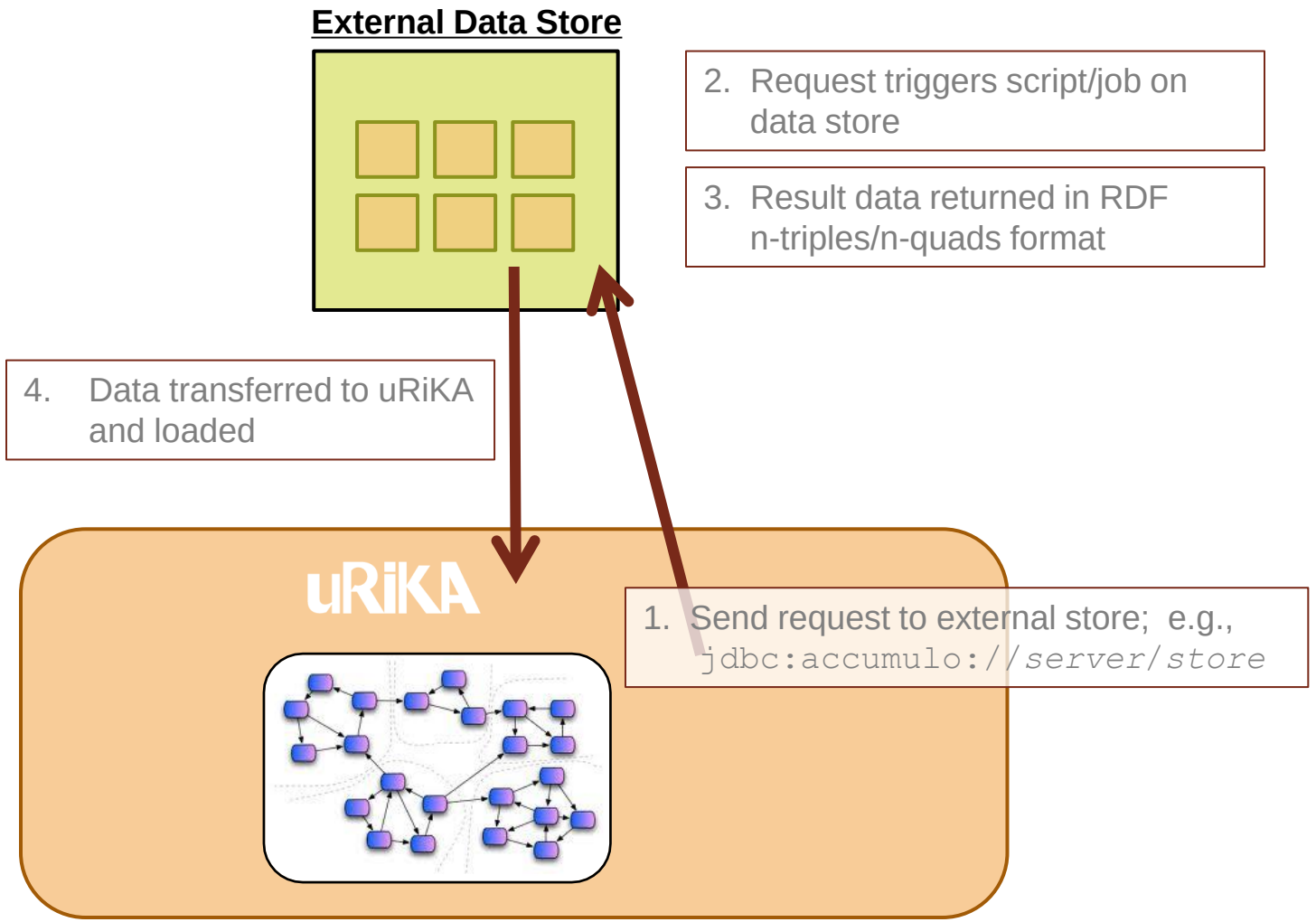
- All files must be converted into .nt/.nq format before being Loaded into CQE
- In the fullness of time, the admin UI will convert from other formats as part of making the SDB
- Today, we use various conversion tools from the command line
  - rdf2rdf: Java tool, see http://www.l3s.de/~minack/rdf2rdf/
  - RIOT: runnable via bash scripts, see http://incubator.apache.org/jena/documentation/io/riot.html

Extracting Data from an SQL Database

- **Use the D2R Server***
  - D2R can be used for an RDB in place; we extract the data once for ingestion into uRiKA
  - See D2R Server Data Extraction Configuration Guide for details
  - Two-step process
    - Generate a mapping from the target database
      - e.g., `generate-mapping -d jdbc:mysql://server/database -o map.n3`
      - Many SQL DB types supported by D2R; MySQL, Oracle, and Postgres used so far with uRiKA
      - Edit the mapping file if (as typical) not all tables/fields are needed
    - Extract the data into an RDF file
      - e.g., `dump-rdf -m map.n3`
      - Move the result file into Lustre

* http://www4.wiwiss.fu-berlin.de/bizer/d2rq/spec/#specification

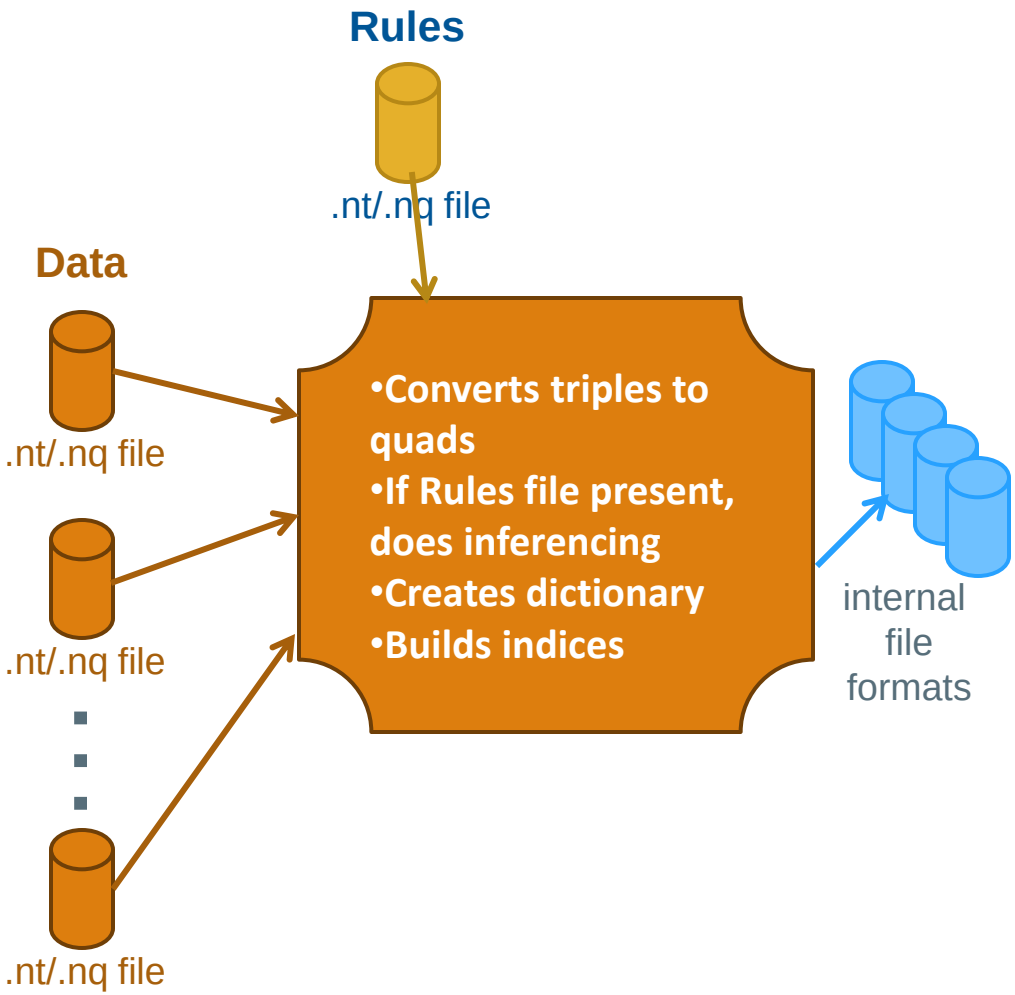
Importing Data from Data Stores
e.g., Hadoop, Accumulo



Agenda

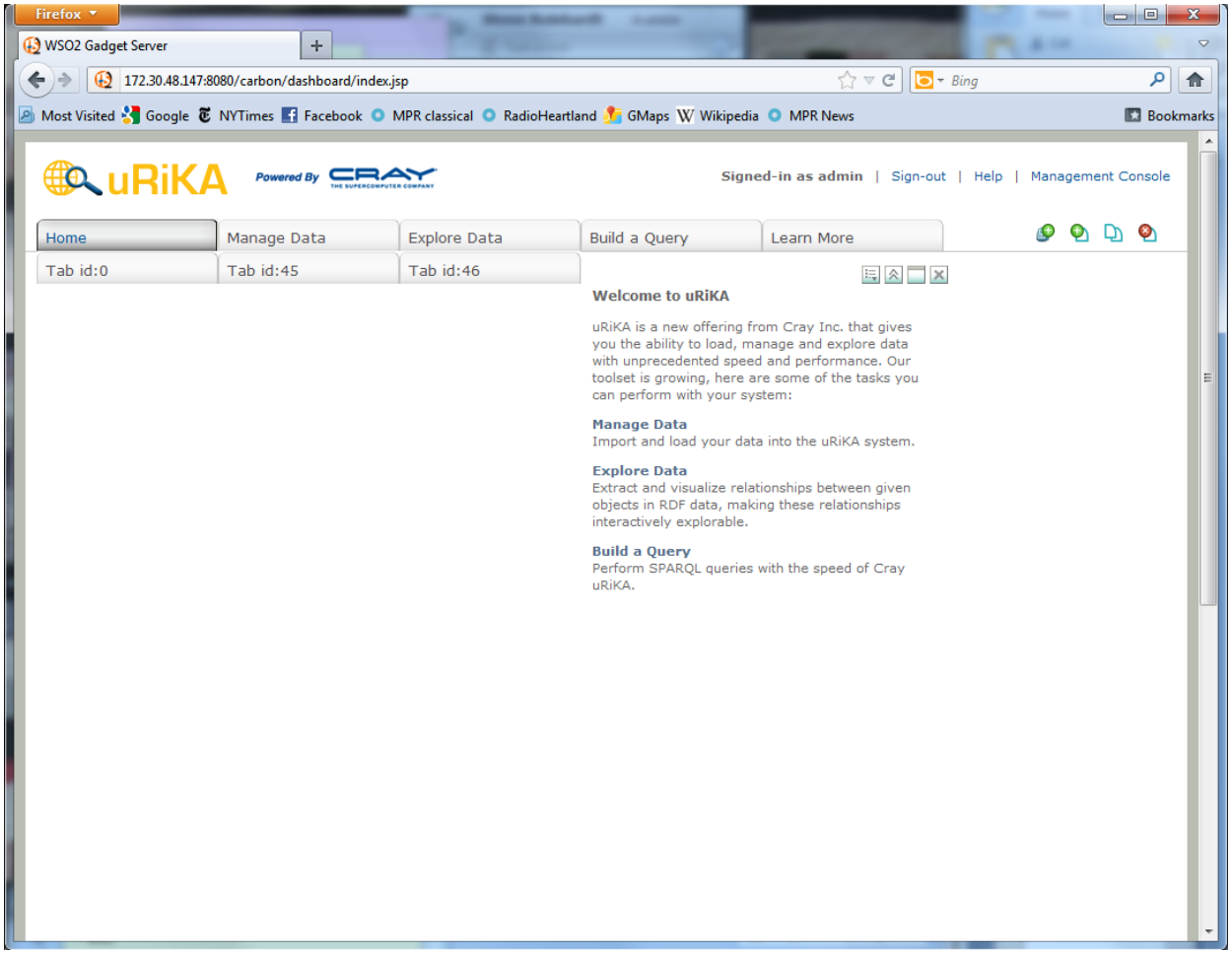
- A Quick Overview
- Converting Data into RDF
- **Ingesting Data into uRiKA**
- Querying uRiKA
- Visualizing Results
- Summary

Ingestion Data Flow

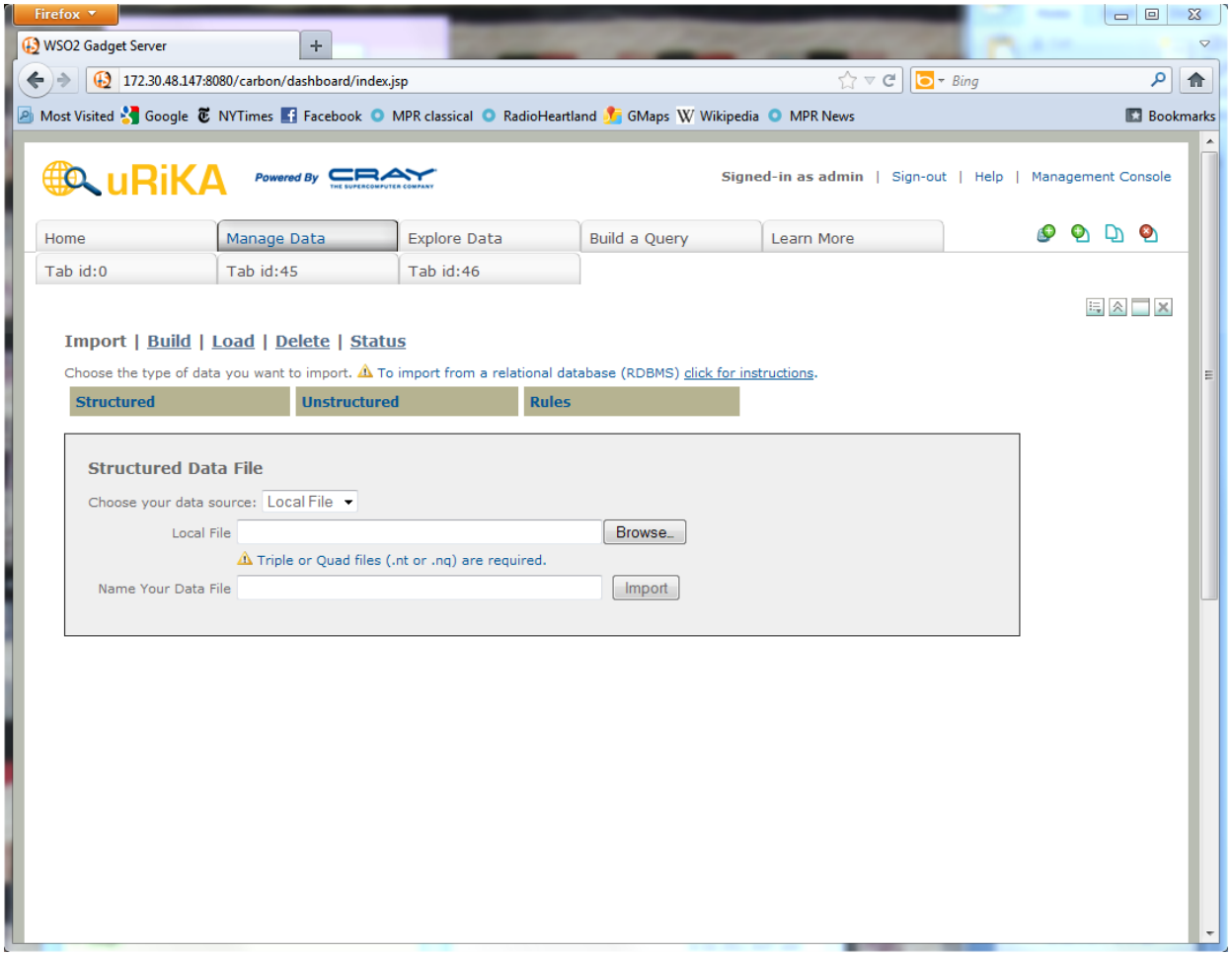


- One or more .nt files (already resident on Lustre) can be combined into a single SDB
- A rules file may also be provided, which causes the ingestion step to run inferencing
- The output of the ingestion step is ~10 files in internal binary formats
- Those files are loaded directly into a uRiKA instance when it's initiated

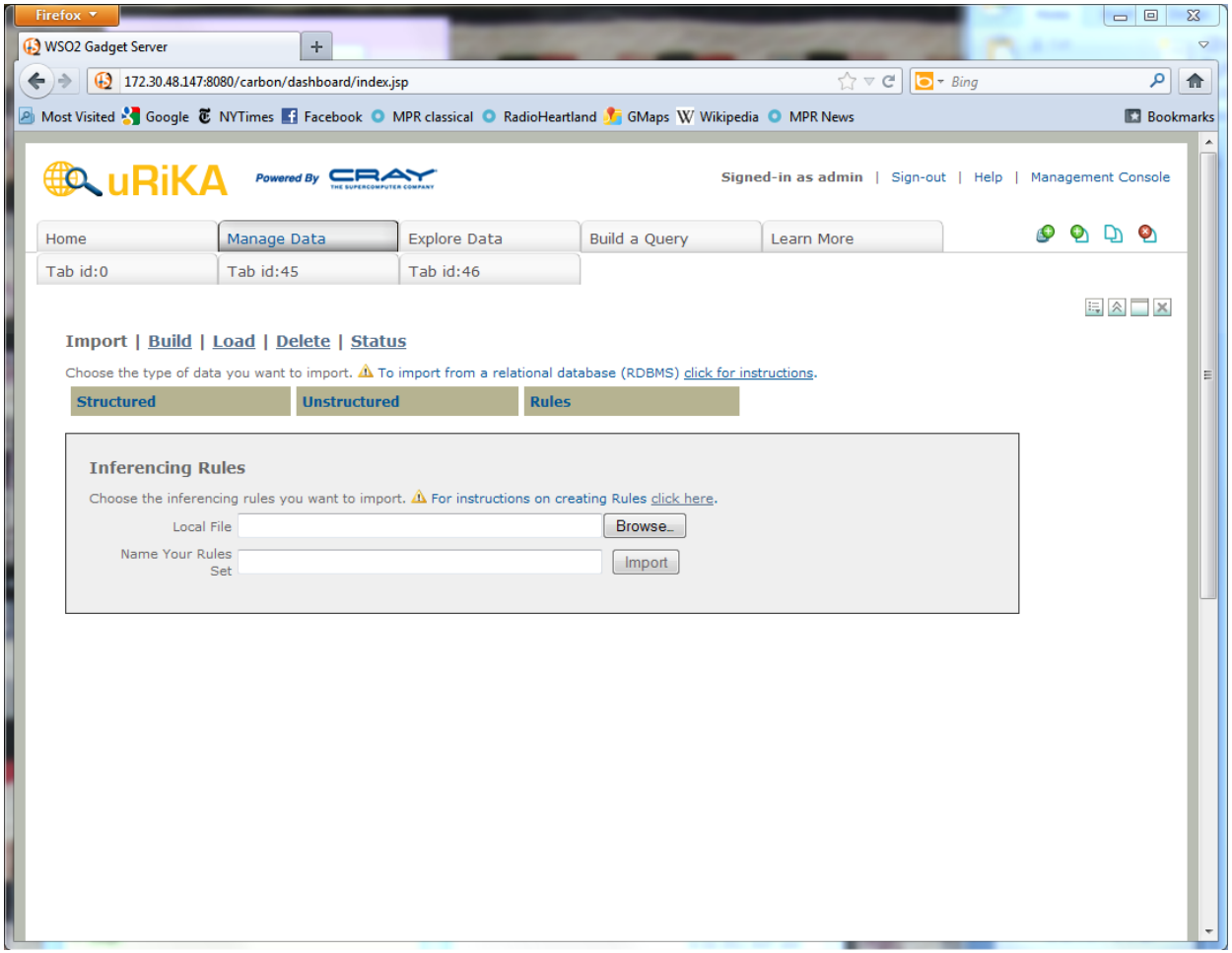
UI – Home


Firefox
WSO2 Gadget Server
172.30.48.147:8080/carbon/dashboard/index.jsp
Bing
Most Visited
Google
NYTimes
Facebook
MPR classical
RadioHeartland
GMaps
Wikipedia
MPR News
Bookmarks
uRiKA
Powered By
CRAY
THE SUPERCOMPUTER COMPANY
Signed-in as admin | Sign-out | Help | Management Console
Home
Manage Data
Explore Data
Build a Query
Learn More
Tab id:0
Tab id:45
Tab id:46
Welcome to uRiKA
uRiKA is a new offering from Cray Inc. that gives you the ability to load, manage and explore data with unprecedented speed and performance. Our toolset is growing, here are some of the tasks you can perform with your system:
Manage Data
Import and load your data into the uRiKA system.
Explore Data
Extract and visualize relationships between given objects in RDF data, making these relationships interactively explorable.
Build a Query
Perform SPARQL queries with the speed of Cray uRiKA.

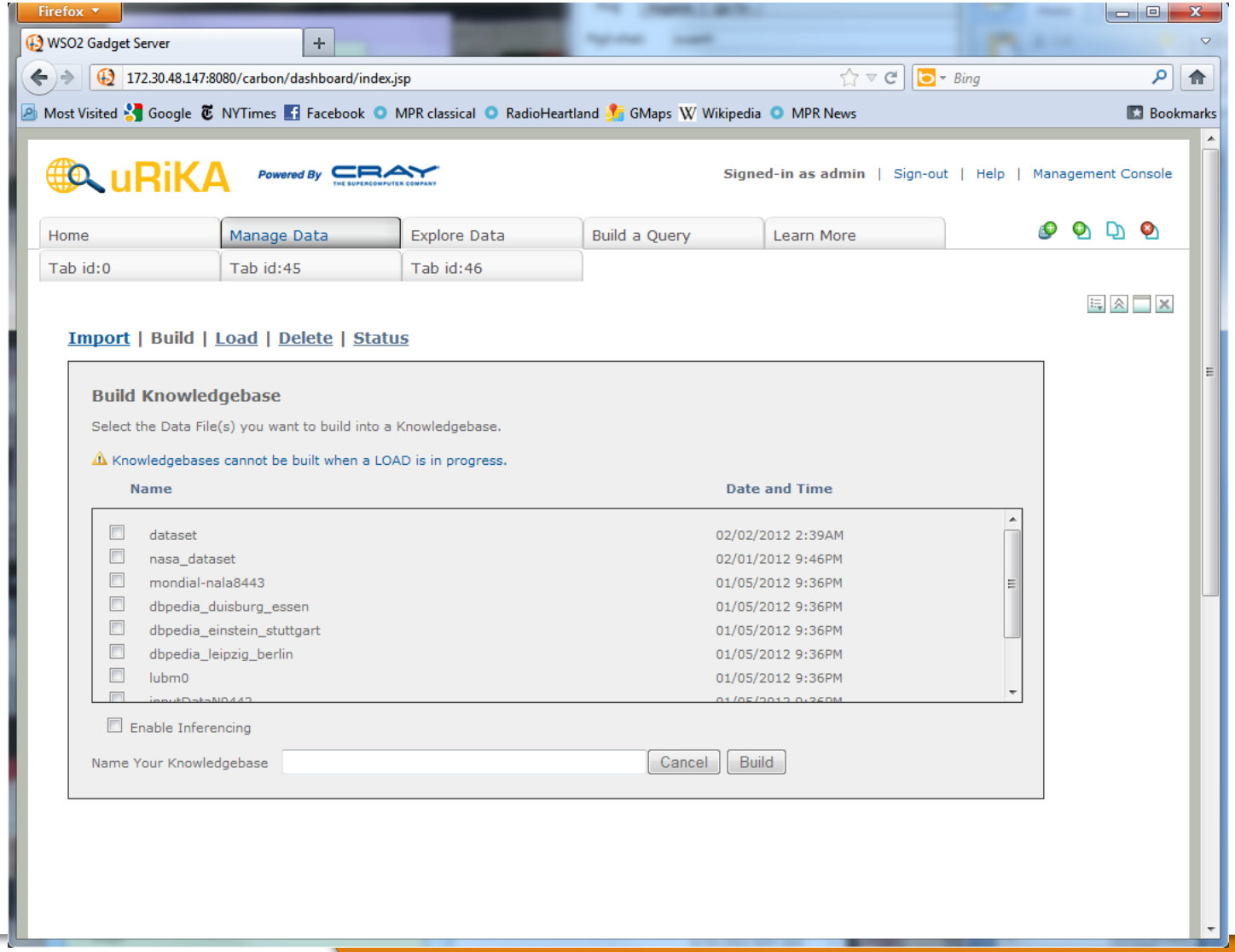
Ingestion UI – Import Data



Ingestion UI – Import Rules

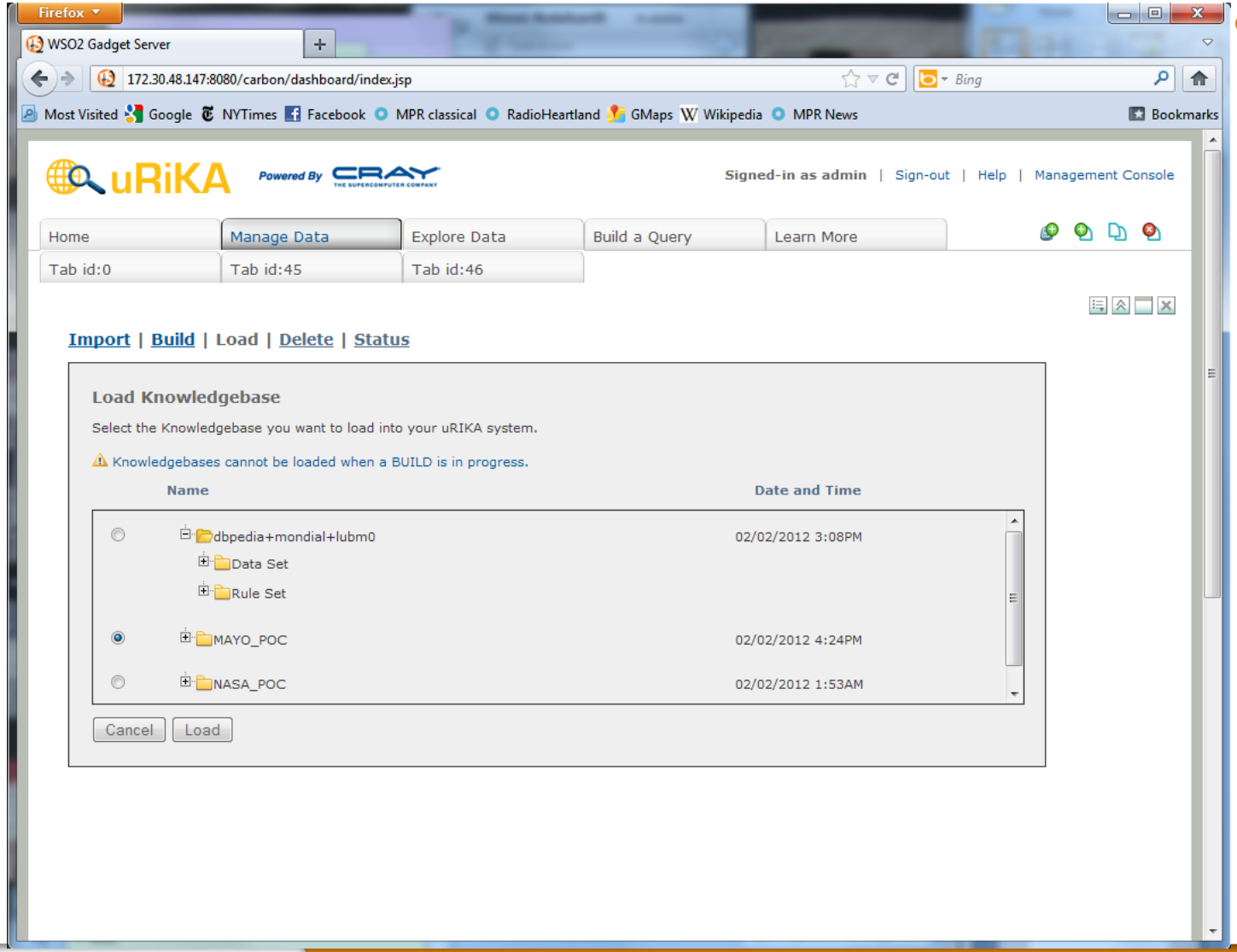


Ingestion UI – Build



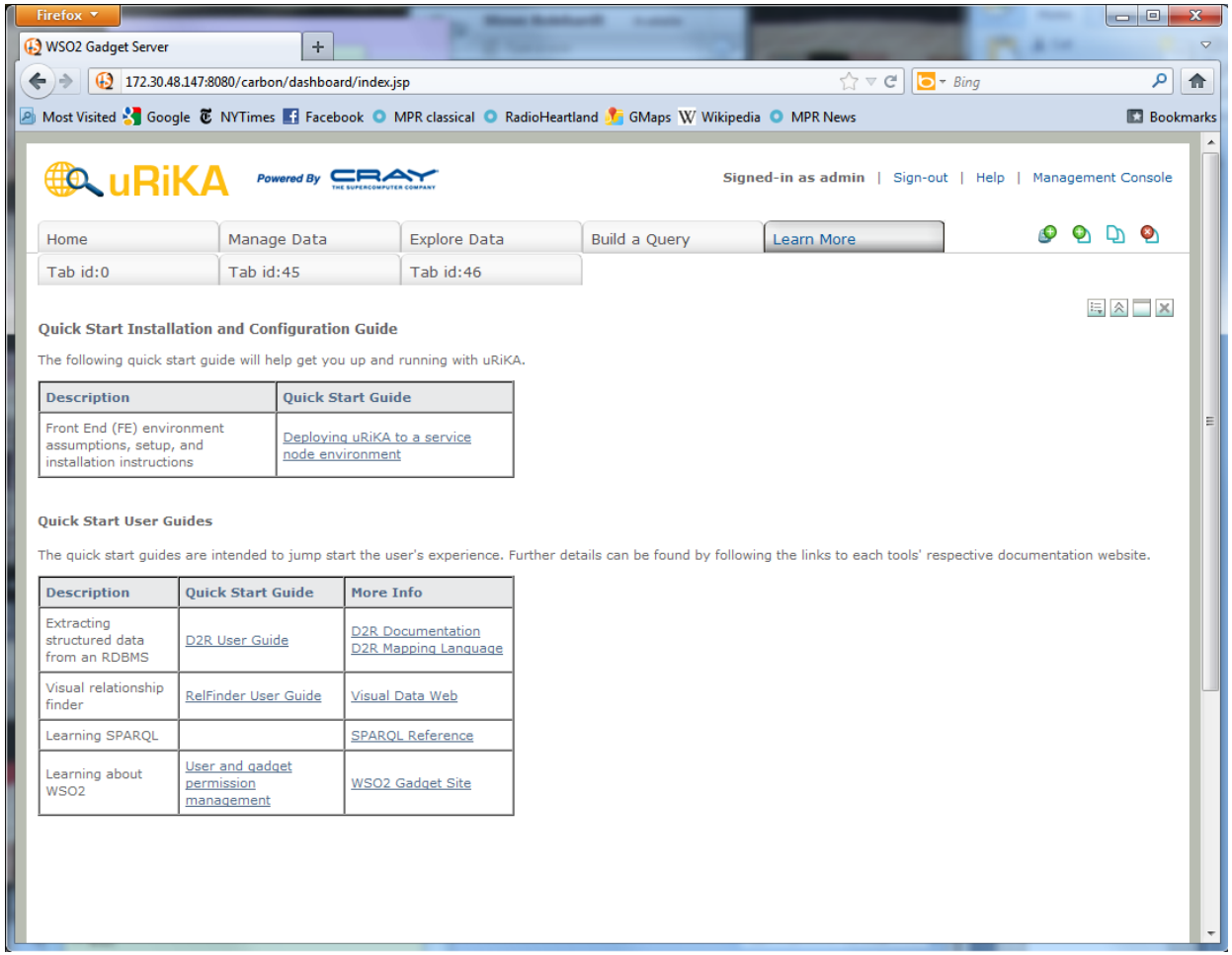
- Can select which (Lustre-resident) files to combine into a knowledge base

Ingestion UI – Load



- Load starts the SDB instance (after stopping a current instance)

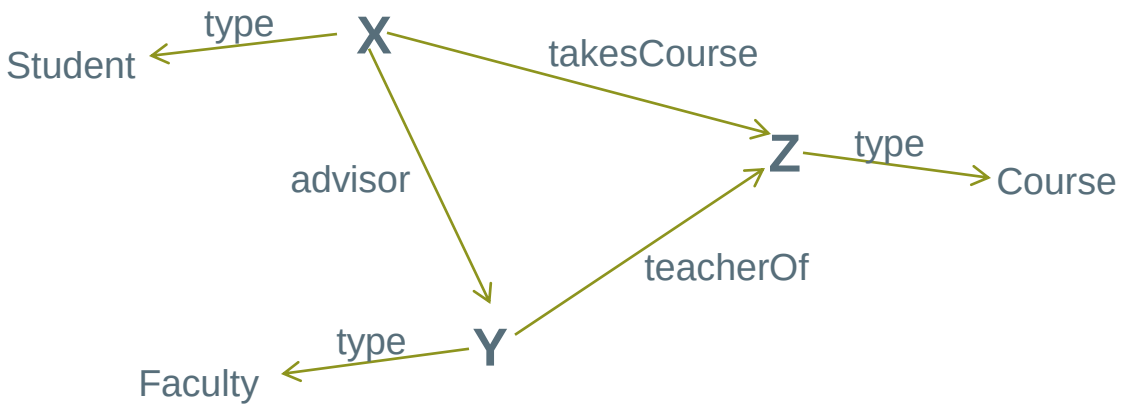
UI – Learn More



Agenda

- A Quick Overview
- Converting Data into RDF
- Ingesting Data into uRiKA
- **Querying uRiKA**
- Visualizing Results
- Summary

SPARQL Query



Lehigh University Benchmark (LUBM) Query 9

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X, ?Y, ?Z
WHERE
{?X rdf:type ub:Student .
 ?Y rdf:type ub:Faculty .
 ?Z rdf:type ub:Course .
 ?X ub:advisor ?Y .
 ?Y ub:teacherOf ?Z .
 ?X ub:takesCourse ?Z}
```

Use of Y as both a subject and an object requires a join

SPARQL roadmap

- **Current uRiKA release (0.9) covers SPARQL 1.0**
  - Released 2008
  - Read-only database
  - Basic pattern matching, filters, unions
- **Next release will cover SPARQL 1.1**
  - Specification released 2011
  - Property paths not included (spec under revision)
  - Improved Database Administrator interface
  - More input file formats supported
- **SPARQL 1.1 Update specification partially supported**
  - In-memory INSERT DATA, DELETE DATA

Inferencing

schema.ttl

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <http://example.org/vehicles/> .

:Vehicle a rdfs:Class .
:Car rdfs:subClassOf :Vehicle .
:SportsCar rdfs:subClassOf :Car .
```

data.ttl

```
@prefix ex: <http://example.org/vehicles/> .
@prefix : <http://myvehicledata.com/> .

:FordFiesta a ex:Car .
:AudiA8 a ex:Car .
:FerrariEnzo a ex:SportsCar .
```

query.rq

```
PREFIX ex: <http://example.org/vehicles/> .
PREFIX : <http://myvehicledata.com/> .

SELECT ?car
WHERE { ?car a ex:Car }
```

- **Without inferencing, query will return**
  - FordFiesta, AudiA8
- **With inferencing, query will return**
  - FordFiesta, AudiA8, FerrariEnzo
- **We expect inferencing to be a uRiKA strength**

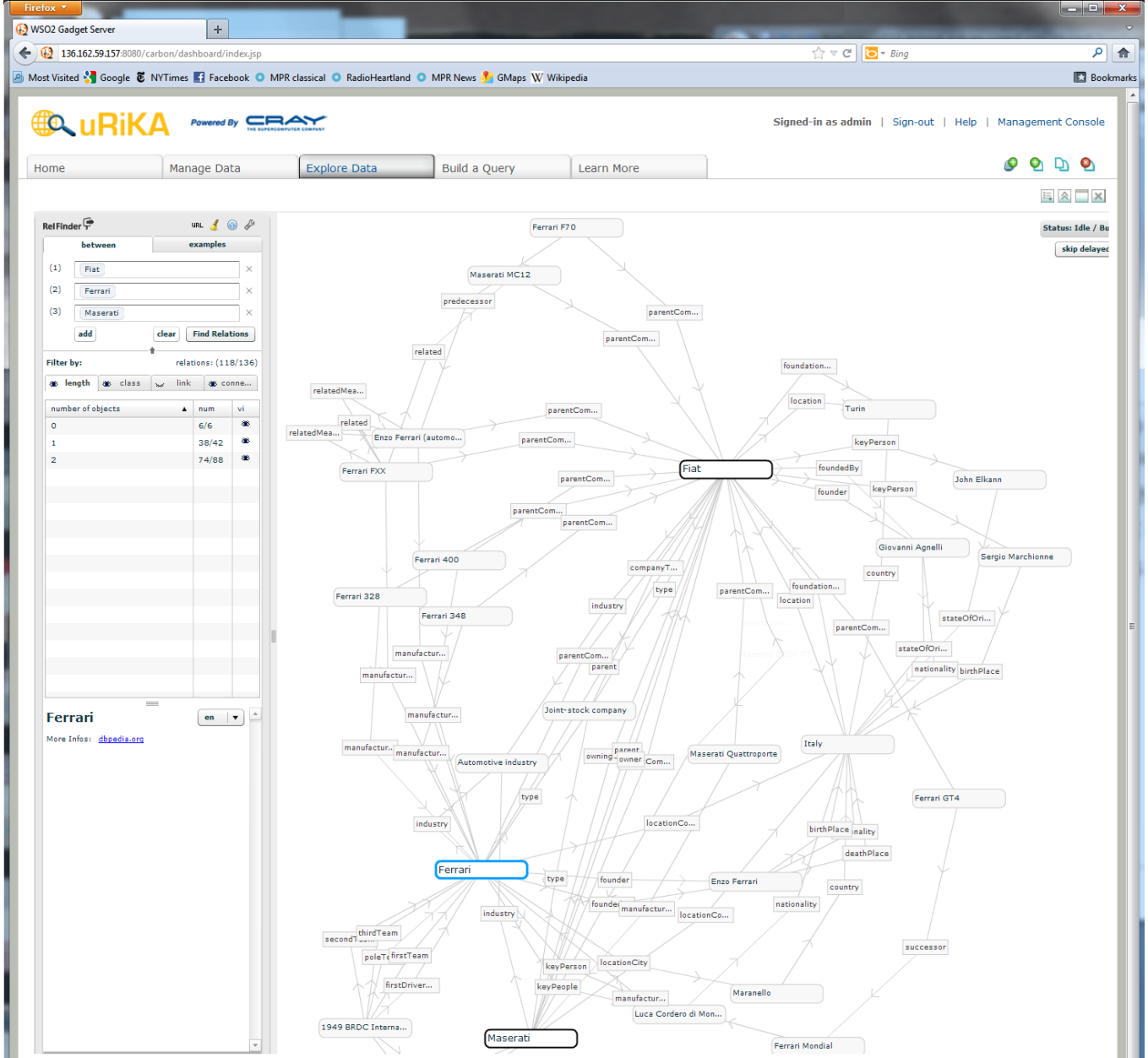
Example from http://www.dotnetrdf.org/content.asp?pageID=Inference%20and%20Reasoning

Agenda

- A Quick Overview
- Converting Data into RDF
- Ingesting Data into uRiKA
- Querying uRiKA
- **Visualizing Results**
- Summary

Visualizing Results

- RelFinder (built into the UI as Explore Data) does this with independently of the relationships in the data
- New visualization packages are being investigated for the next release
- Google Gadgets within the WSO2 framework are the primary interface for gadget development
  - These interfaces are not fully documented or exposed in 0.9


Firefox
WSO2 Gadget Server
136.162.59.157:8080/carbon/dashboard/index.jsp
Most Visited Google NYTimes Facebook MPR classical RadioHeartland MPR News GMaps Wikipedia
Bookmarks
uRiKA
Powered By CRAY
Signed-in as admin | Sign-out | Help | Management Console
Home
Manage Data
Explore Data
Build a Query
Learn More
RelFinder
between
examples
(1) Fiat
(2) Ferrari
(3) Maserati
add
clear
Find Relations
Filter by:
relations: (118/136)
length
class
link
conne...
number of objects
num
0 6/6
1 38/42
2 74/88
Ferrari
en
More Infos: dbpedia.org
Ferrari F70
Maserati MC12
predecessor
parentCom...
related
relatedMea...
Enzo Ferrari (automo...
Ferrari FXX
Ferrari 400
Ferrari 328
Ferrari 348
manufactur...
Joint-stock company
Automotive industry
industry
type
owning...
parent
owner
Com...
Maserati Quattroporte
Italy
Ferrari GT4
locationCo...
Ferrari
founder
Enzo Ferrari
manufactur...
locationCo...
nationality
industry
thirdTeam
firstTeam
keyPerson
locationCity
keyPeople
firstDriver...
manufactur...
Maranello
Luca Cordero di Mon...
1949 BRDC Interna...
Maserati
Ferrari Mondial
successor
deathPlace
country
birthPlace
nality
foundation...
location
Turin
keyPerson
John Elkann
foundedBy
founder
Giovanni Agnelli
Sergio Marchionne
companyT...
stateOfOri...
Fiat
Status: Idle / Bu
skip delayed

Agenda

- A Quick Overview
- Converting Data into RDF
- Ingesting Data into uRiKA
- Querying uRiKA
- Visualizing Results
- **Summary**

Summary

- uRiKA is targeted at large-scale semantic-graph processing
- Some end-to-end ETL workflows need components beyond what uRiKA has been tested with today
- We expect to test and qualify third-party components
- Most customers have their own favorite ETL conponents- we will learn a lot by working with early customers
- … and modify uRiKA based on those lessons

Learn More

- SPARQL by Example tutorial, by Lee Feigenbaum, http://www.cambridgesemantics.com/2008/09/sparql-by-example/
- Search RDF data with SPARQL, by Philip McCarthy http://www.ibm.com/developerworks/xml/library/j-sparql/
- Semantic Web for the Working Ontologist, by Dean Allemang and James Hendler, ISBN 978-0123859655
- Learning SPARQL, by Bob DuCharme, O'Reilly, ISBN 978-1-449-30659-5

- RDF: www.w3.org/**RDF**/
- SPARQL: www.w3.org/TR/rdf-**sparql**-query/
- D2R: www4.wiwiss.fu-berlin.de/bizer/**d2r**-server
- *WSO2:* **wso2**.com/products/**application**-server
- Google Gadgets: www.**google**.com/webmasters/**gadgets**/

Thank you!

James D. Maltby, Ph.D
jmaltby@yarcdata.com

Backup Slides

Data Sizes

- Today, typically the input data should consume no more than 50-60% of the memory of the system, to leave space for temporary storage during queries
- Estimating memory consumption from the number of triples is inexact, but 100B:triple is a reasonable rule of thumb

New SPARQL 1.1 features

- **Aggregates and GROUP BY**
  - Divide solution set into subgroups, can apply operator to subgroup
- **Subqueries**
  - Like RDBMS “view” => query against a previous result
- **Negation**
  - NOT EXISTS – does a pattern exist
- **SPARQL 1.1: New FILTER expressions**
  - EXISTS, NOT EXISTS
  - IN, NOT IN
  - COALESCE – return RDF term value of the first term in the expression that evaluates without error
  - IF – evaluates the Effective Boolean Value of the first argument, returns 2nd or 3rd depending on whether EBV was true or fals

More new features in SPARQL 1.1

- **SERVICE**
  - Query remote SPARQL endpoint
- **AVG(), MIN(), MAX(), COUNT()**
  - Functions over returned values
- **MINUS – eliminate a specific match**
- **BIND**
  - Assign a value to a new variable in a group graph pattern

Dynamic in-memory Updates

- **SPARQL 1.1 Update requests**
  - Subset of full specification
  - Submitted over HTTP like a SPARQL query
- **INSERT DATA *QuadData***
  - Performed in-memory on compute nodes
  - Forward static inferencing on inserted triples
  - Copy of update stored on disk
- **DELETE DATA *QuadData***
  - Also in-memory, logged to disk
  - Forward static inferencing on deleted triples
  - No backward inferencing