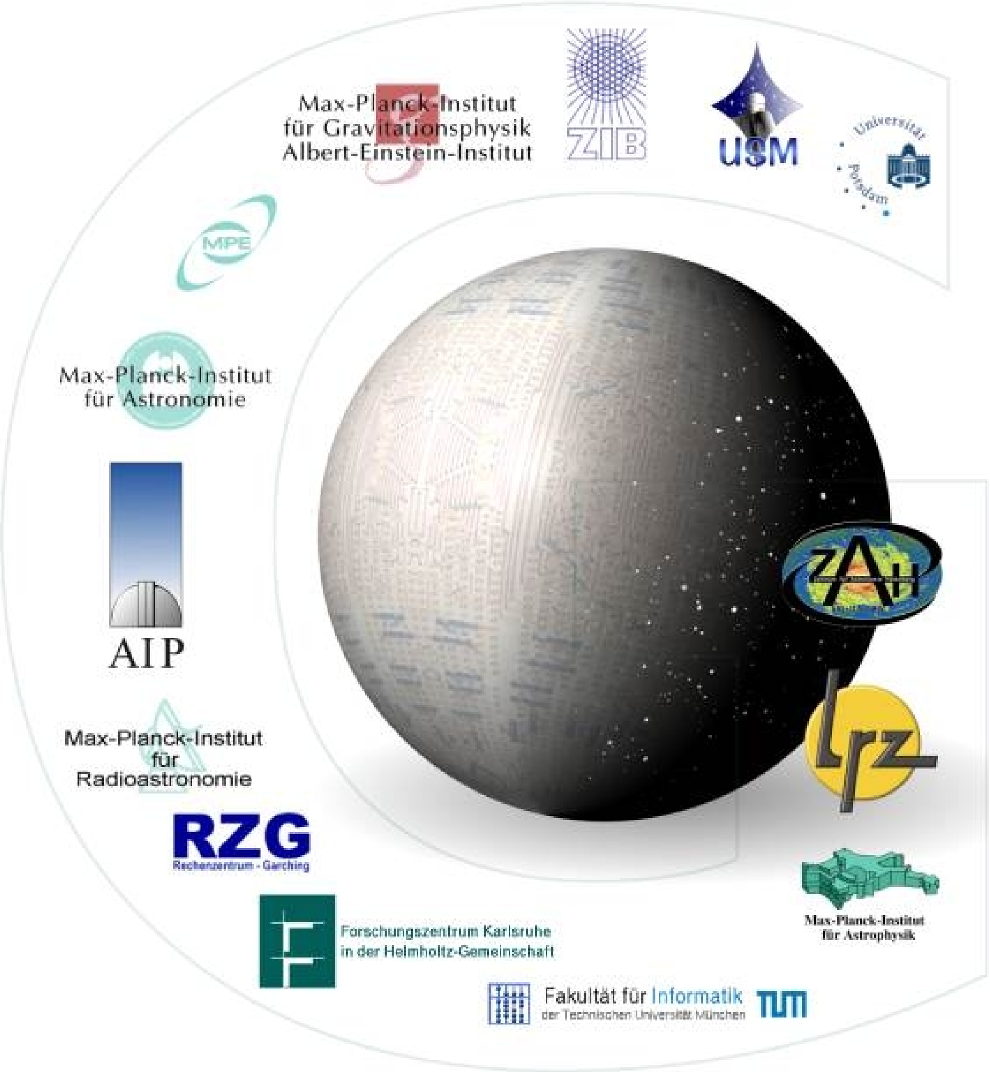
D2.1 Information Service requirements and architecture

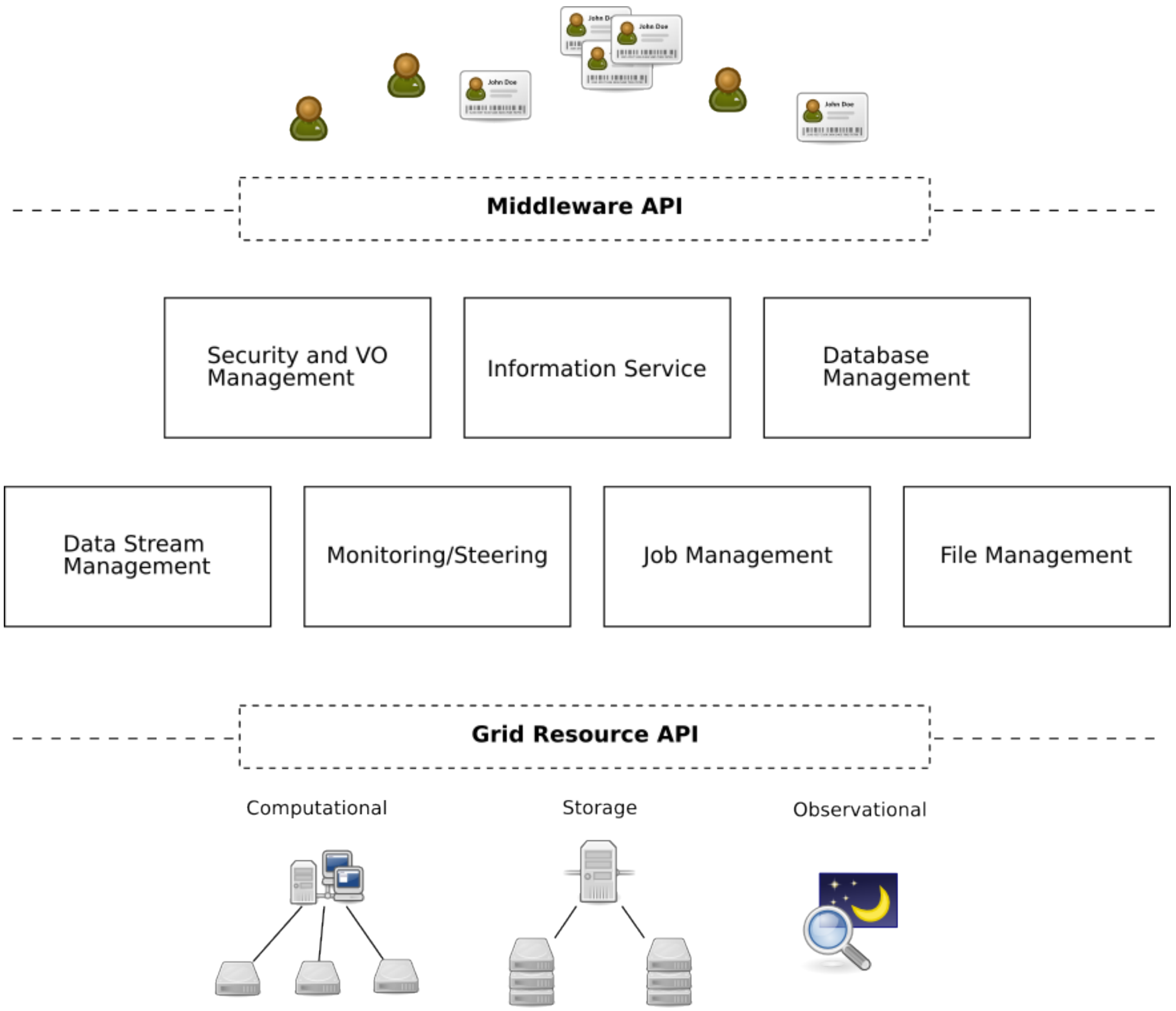
Mikael Högqvist
hoegqvist@zib.de


Max-Planck-Institut für Gravitationsphysik Albert-Einstein-Institut
ZIB
USM
Universität Potsdam
MPE
Max-Planck-Institut für Astronomie
AIP
Max-Planck-Institut für Radioastronomie
RZG
Rechenzentrum - Garching
ZAH
lrz
Max-Planck-Institut für Astrophysik
Forschungszentrum Karlsruhe in der Helmholtz-Gemeinschaft
Fakultät für Informatik der Technischen Universität München
TUM

Introduction

- Overview
  - AstroGrid-D introduction
  - Metadata and requirements
  - Architecture approach
  - Demo

AstroGrid-D introduction



Middleware API

Security and VO Management

Information Service

Database Management

Data Stream Management

Monitoring/Steering

Job Management

File Management

Grid Resource API

Computational

Storage

Observational

Metadata overview

- Purpose of metadata
  - Tag information with information to make it easier to find
- Information Service objectives
  - Provide methods to store and find information

AstroGrid-D metadata

- Resources (computational, storage, robotic telescopes, network, software)
  - Well-defined schemes
    - GLUE schema (computational, storage, software)
    - RTML (robotic telescopes)
- State of grid services (jobs, files, data stream, ...)

Metadata (cont.)

- Scientific metadata (domain-specific description of data sets, provenance)
- Application-specific metadata (job history, job progress, ...)
- Schemes will be decided later by the community

Requirements

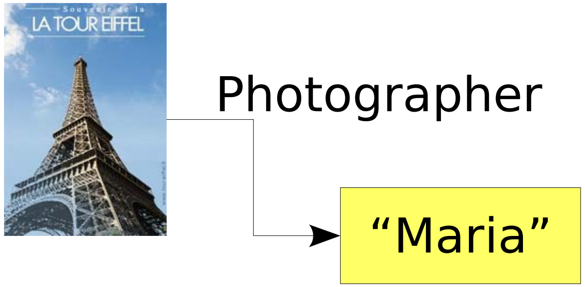
- Extensible/flexible schemes
- Easy to extract and export metadata
- Protect from unauthorized access
- Support exact match and range queries
- Handle different metadata characteristics

Approach

- Metadata representation with RDF
  - An RDF entry is a (subject, predicate, object)-tuple
  - A set of triples/statements form a graph
- Metadata access via SPARQL
  - Query language for RDF
  - Graph pattern matching
- Simple interface including add, update, remove and query

RDF Example

“A picture of the Eiffel tower has a photographer with value Maria”



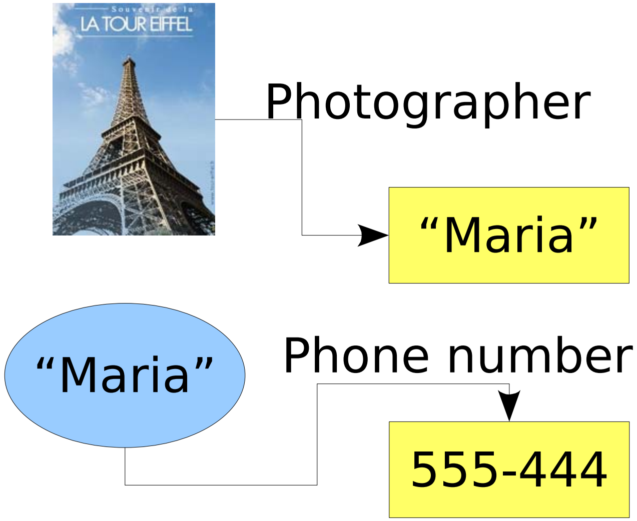
Photographer

“Maria”

RDF Example

“A picture of the Eiffel tower has a photographer with value Maria”

“Maria has a phone number with value 555-444”



Photographer

“Maria”

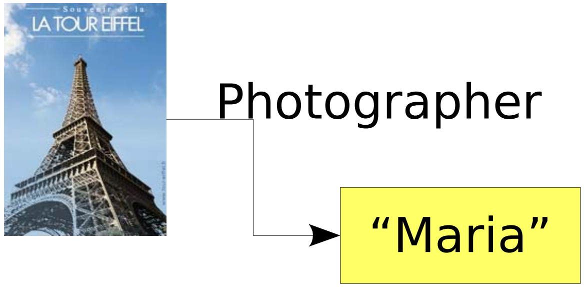
“Maria”

Phone number

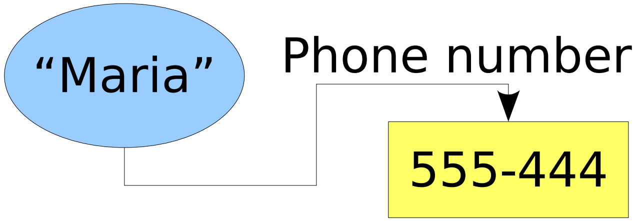
555-444

RDF Example

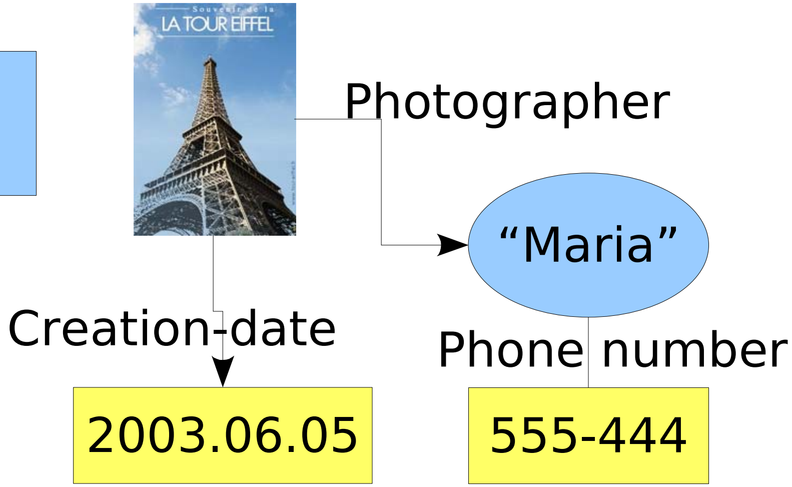
“A picture of the Eiffel tower has a photographer with value Maria”



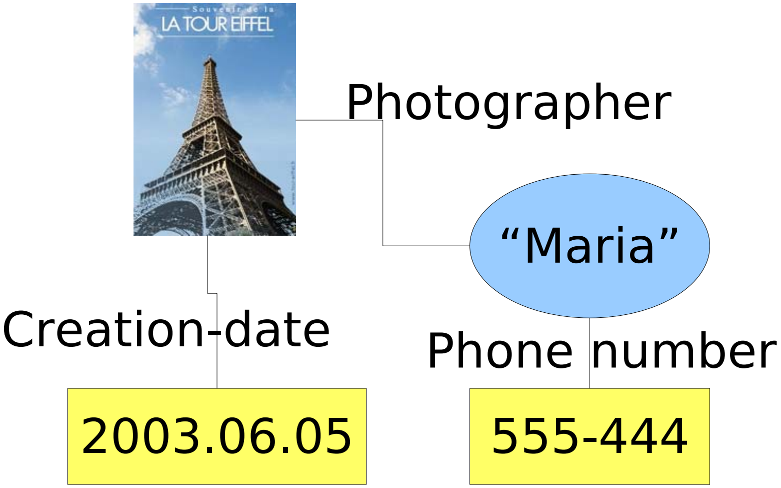
“Maria has a phone number with value 555-444”



“A picture of the Eiffel tower has creation-date with value 2003.06.05”



SPARQL Example



"Get the name and phone number of the photographer who took the picture of the Eiffel tower"

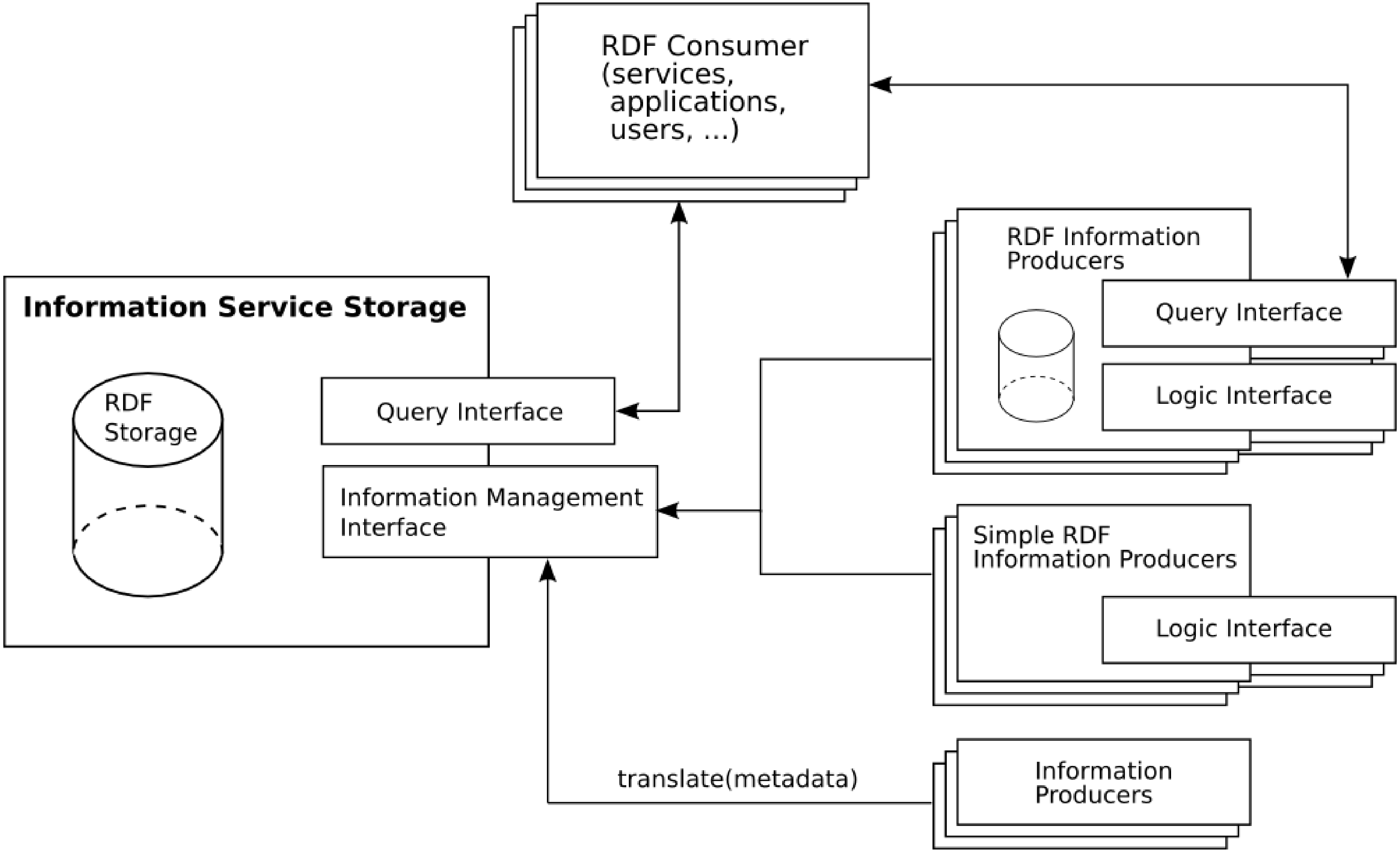
Input graph

SELECT ?name, ?number WHERE
{"Picture of Eiffel tower" "Photographer" ?name .
?name "Phone number" ?number }

Number	Name
555-444	Maria

Output results

Architecture overview


RDF Consumer (services, applications, users, ...)
RDF Information Producers
Query Interface
Logic Interface
Information Service Storage
RDF Storage
Query Interface
Information Management Interface
Simple RDF Information Producers
Logic Interface
translate(metadata)
Information Producers

Storage interface

- Add(String RDF, String context)
- Update(String RDF, String context)
  - Overwrite matching statements
- Remove([statements], String context)
  - Delete existing metadata part of the information service storage
- http://infoservice.gac-grid.de/?query=...
  - Extract metadata from the information service or RDF information producers

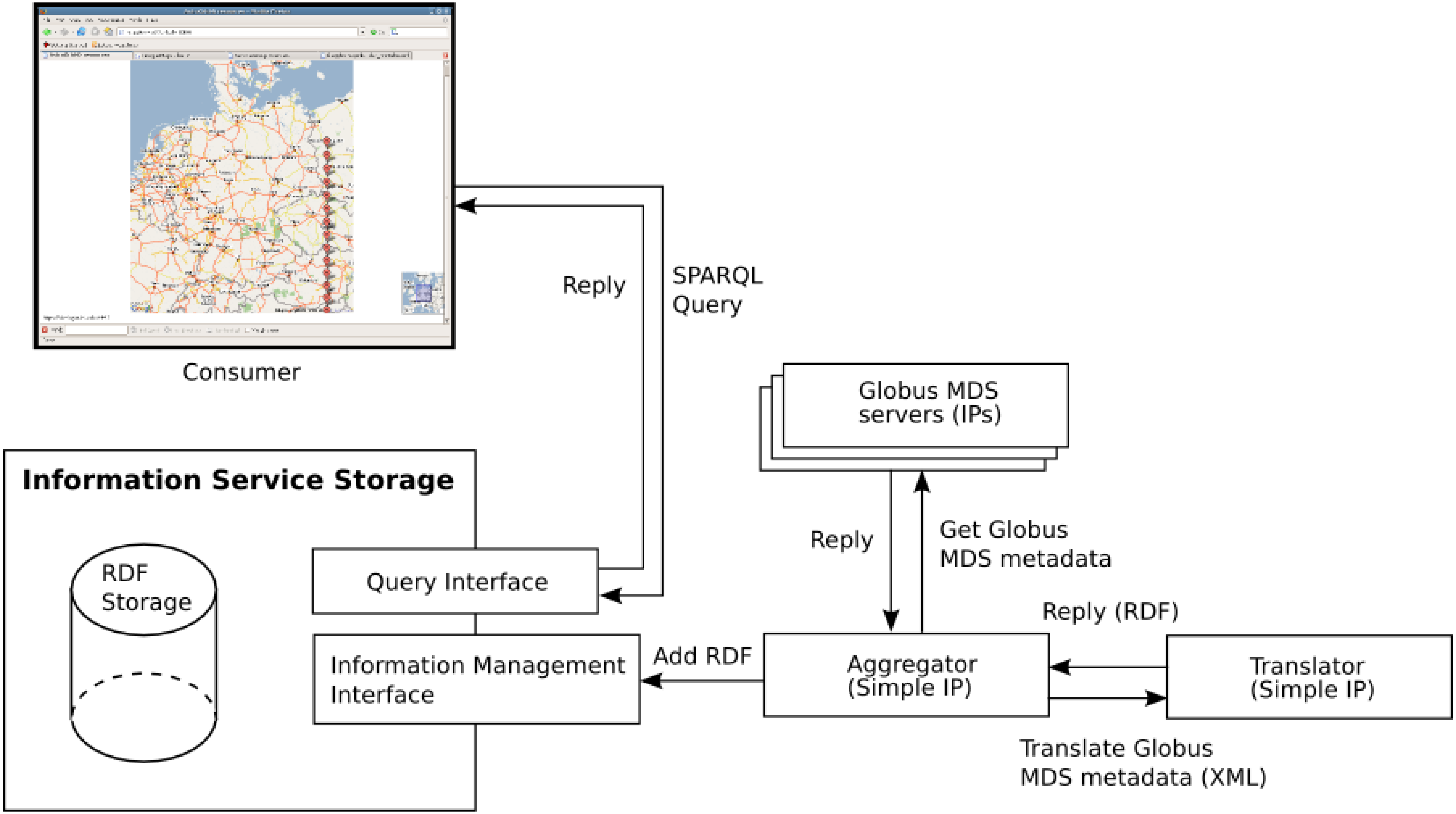
Storage internals

- Context
- Security
  - ACLs and levels
- System statements
  - Time-to-live, owner, ACL, Time stamp, ...

Demo: Overview

- Idea: use Google's map API to present grid resources using RDF metadata provided by an information services
- Tools
  - MDS4 WebMDS
  - Template language (KID) for translating to RDF
  - An RDF store (rdflib.net)
  - Add(String RDF_data) via XMLRPC
  - Query using XMLRPC returning results in JSON

Demo: Component interaction



Consumer

Reply

SPARQL Query

Globus MDS servers (IPs)

Reply

Get Globus MDS metadata

Information Service Storage

RDF Storage

Query Interface

Information Management Interface

Add RDF

Aggregator (Simple IP)

Reply (RDF)

Translator (Simple IP)

Translate Globus MDS metadata (XML)

Outlook

- Implementation
  - Wrap existing RDF tools
  - First version in November 2006 (M18)
- Distributed backend
  - Partitioning
  - Replication

Distributed file management

- Goal
  - Store, find and access
- Requirements
  - Remote partial file access
  - Automated staging of input and output data
  - Monitoring of log-files
  - Provide files to collaborations both within and outside the AstroGrid-D community
  - Restrict access to sensitive data

RDF Resources

- Storage/toolkits
  - Sesame 2 (www.openrdf.org) >70 million statements
  - Jena (http://jena.sf.net/)
  - rdflib.net (Python implementation)
  - Redland (http://librdf.org porting from C using SWIG)
- http://www.w3.org/RDF (start here!)
- http://www.thefigtrees.net/lee/blog/2006/03/sparql_calendar_demo_overview.html (practical sparql introduction)

References

- F. Manola and E. Miller. RDF primer. http://www.w3.org/TR/rdf-primer/, February 2004.
- E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF http://www.w3.org/TR/rdf-sparql-query/, April 2006.
- J. J. Carroll, C. Bizer, P. Hayes and P. Stickler. Named graphs, provenance and trust. In Proc. of WWW 2005.
- F.V. Hessman, C. Pennypacker, E. Romero-Colmenero and G. Tuparev. Telescope networking and user support via remote telescope markup language. In Proc. of SPIE 2004.
- S. Andreozzi, S. Burke, L. Field, S. Fisher, B. Konya, M. Mambelli, J. M. Schopf, M. Viljoen, and A. Wilson. Glue schema specification version 1.2. Technical report, December 2005.
- M. Högqvist and T. Röblitz. AstroGrid-D Metadata Management and Discovery, Deliverable D2.1

Demo: SPARQL queries

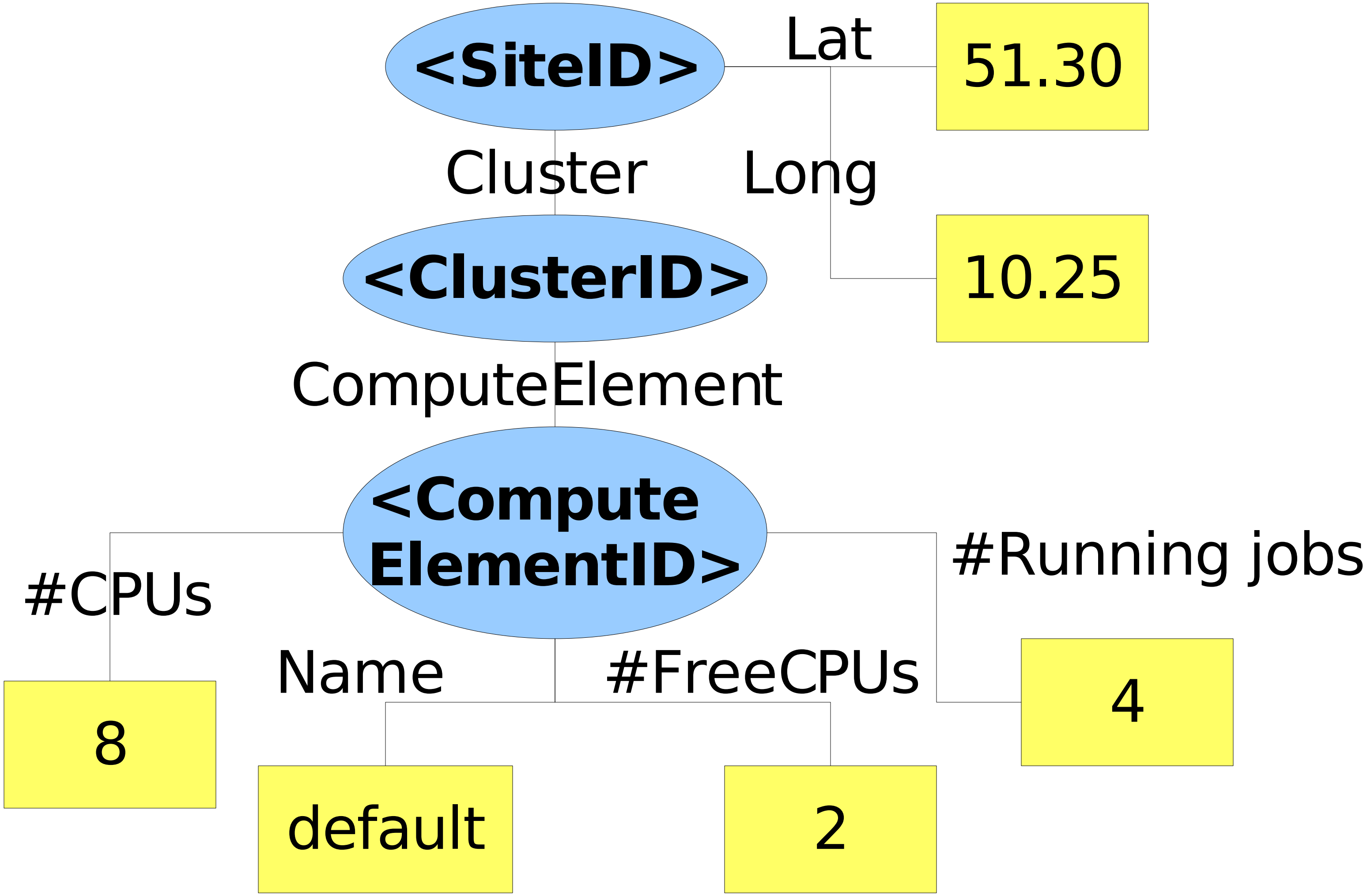
“Get all computing elements from site S”

```
SELECT ?ce WHERE {S “ComputeElement” ?ce}
```

“Get all sites and their longitude and latitude if available”

```
SELECT ?site, ?lat, ?long WHERE
{?site rdf:type “Site” .
 OPTIONAL {?site geo:lat ?lat .
           ?site geo:long ?long }
}
```

Demo: RDF graph (example)


<SiteID>
Lat
51.30
Cluster
Long
<ClusterID>
10.25
ComputeElement
<Compute ElementID>
#CPUs
#Running jobs
Name
#FreeCPUs
8
default
2
4