UPPSALA UNIVERSITET

Spatio-Temporal Gridded Data Processing on the Semantic Web

Andrej Andrejev, Dimitar Misev*, Peter Baumann*, Tore Risch

Department of Information Technology, Uppsala University
** Computer Science & Electrical Engineering Department, Jacobs University, Bremen*

andrej.andrejev@it.uu.se

UPPSALA UNIVERSITET

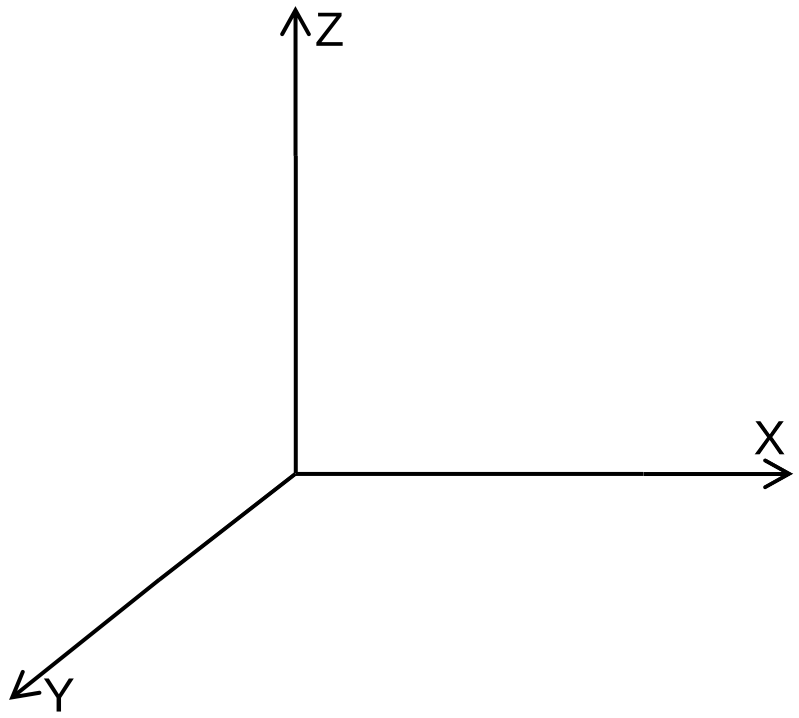
Introduction

They never come alone...

Introduction

Big Data..

Introduction



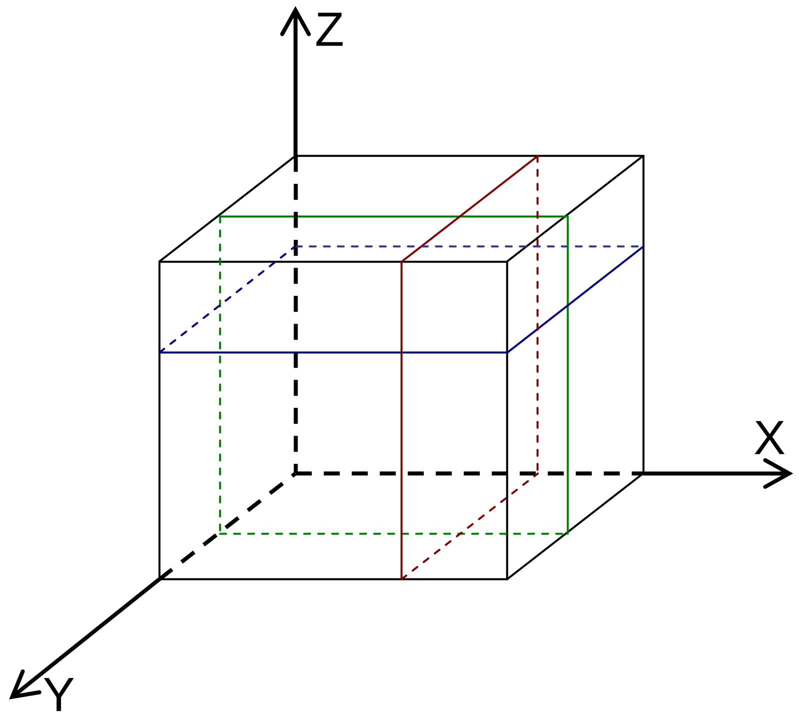
Big Data..

Namely: **Massive Numeric Datasets**

• typically ordered along a number of orthogonal axes

UPPSALA UNIVERSITET

Introduction



Big Data..

Namely: **Massive Numeric Datasets**

- **typically ordered along a number of orthogonal axes**

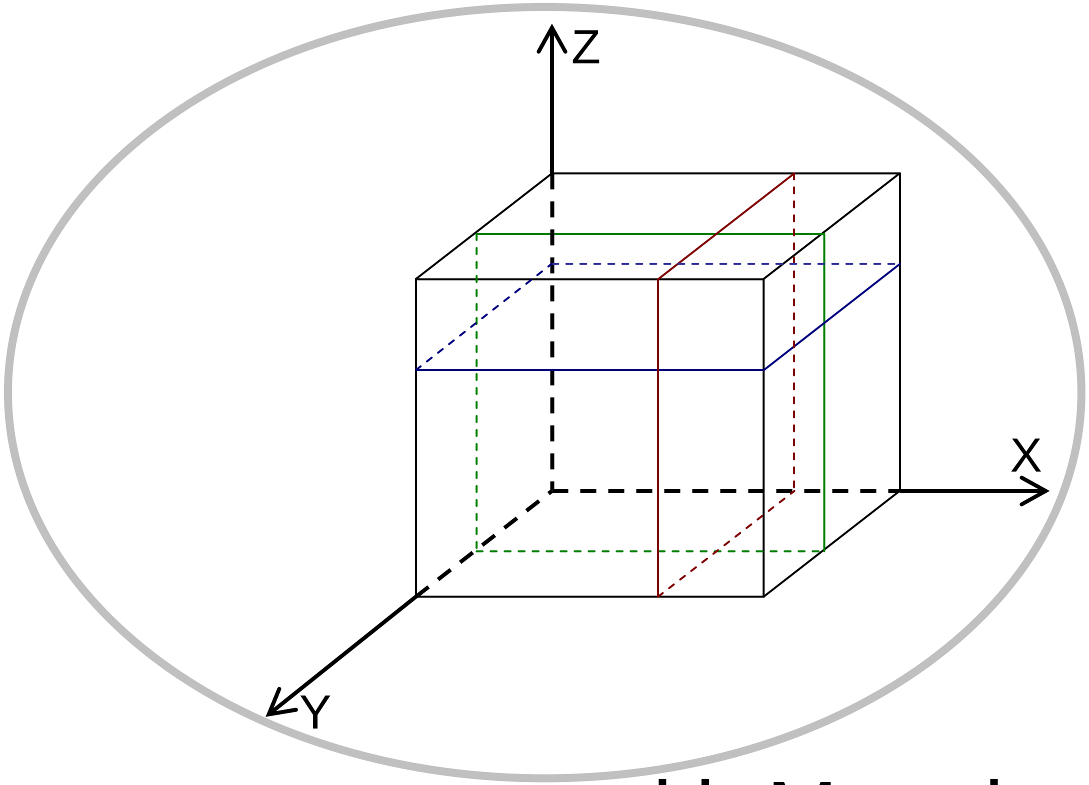
Introduction

They come together...

Introduction

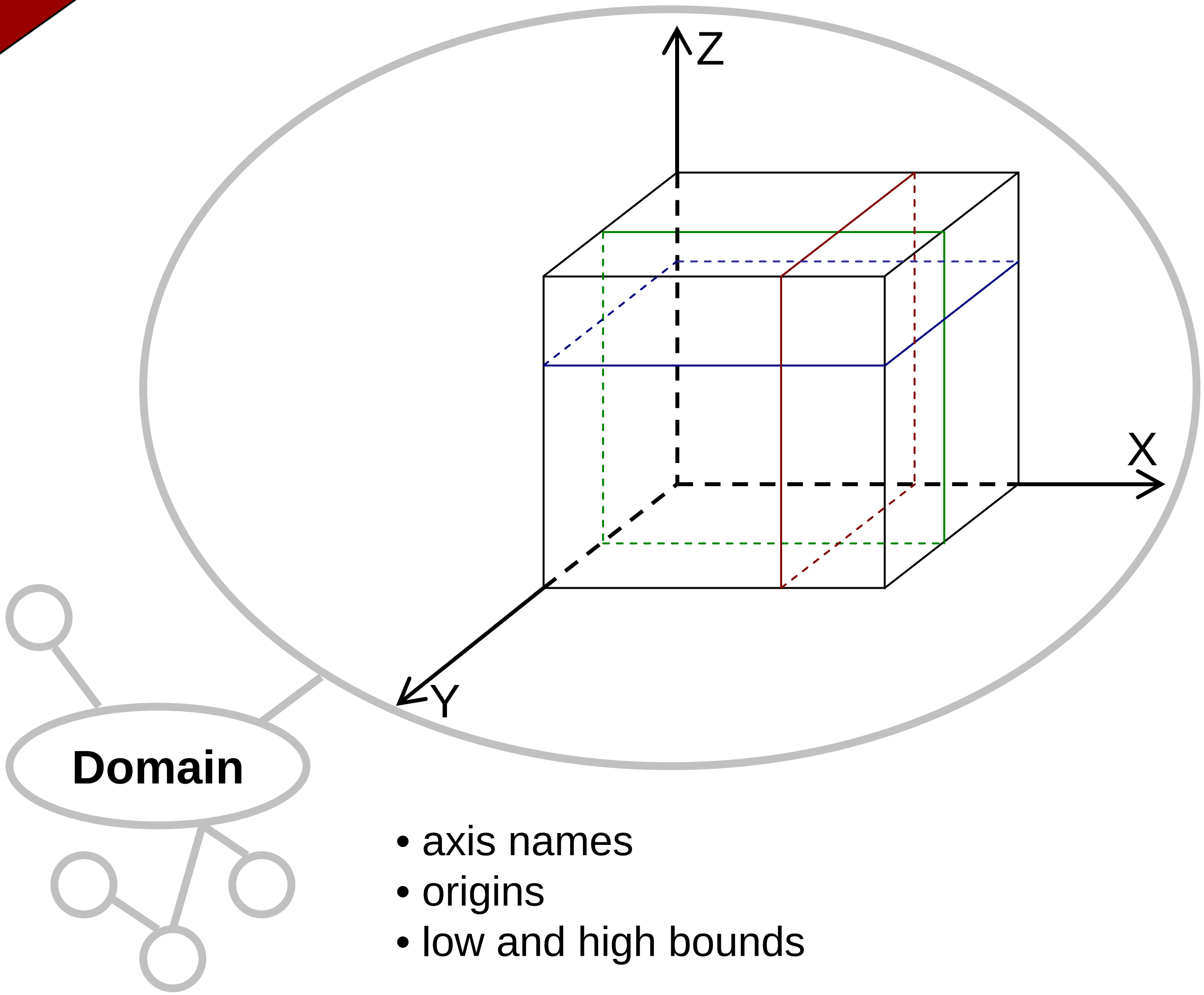
... with Metadata

Introduction



... with Metadata

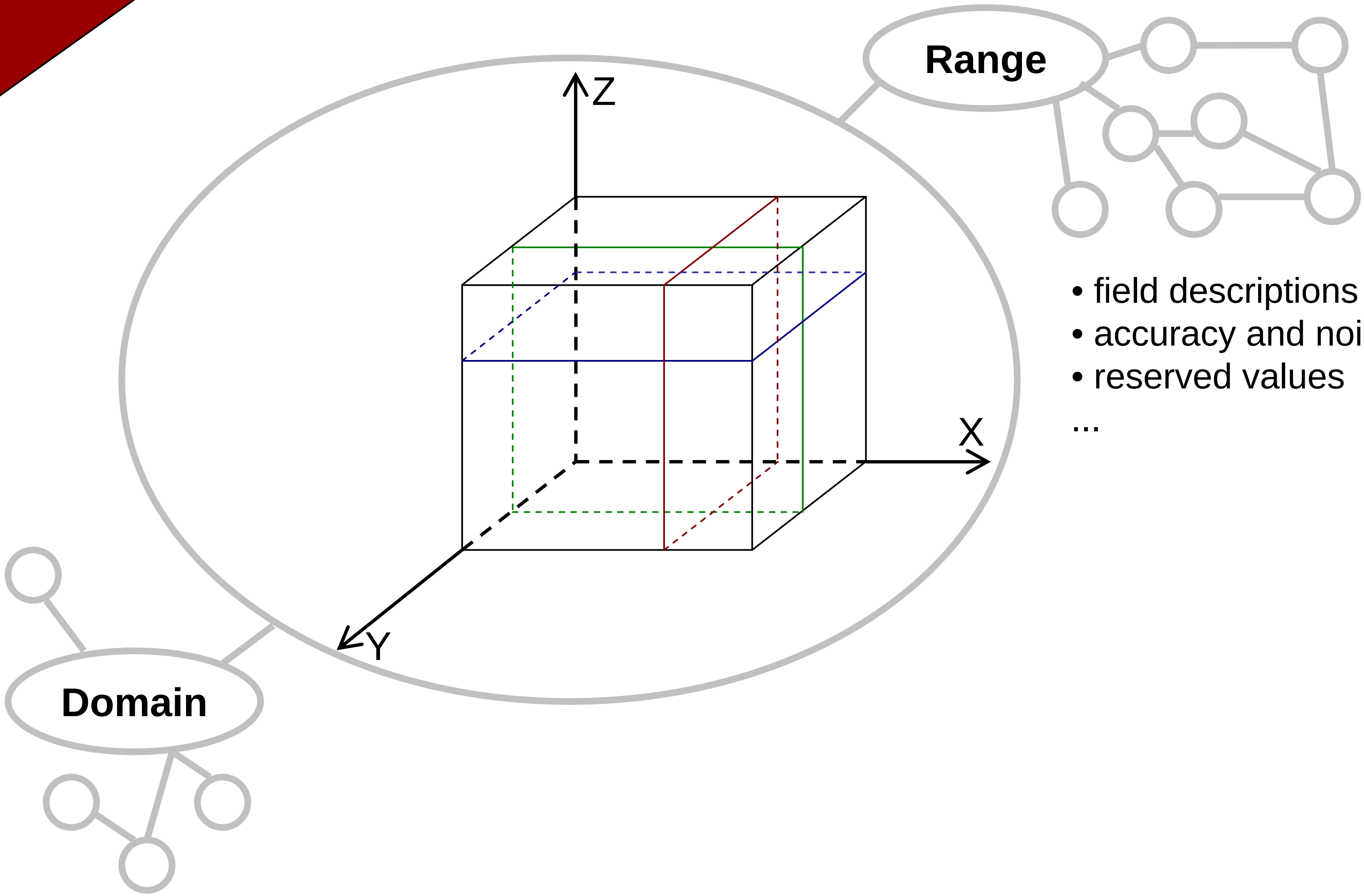
Introduction



• axis names
• origins
• low and high bounds
...

UPPSALA UNIVERSITET

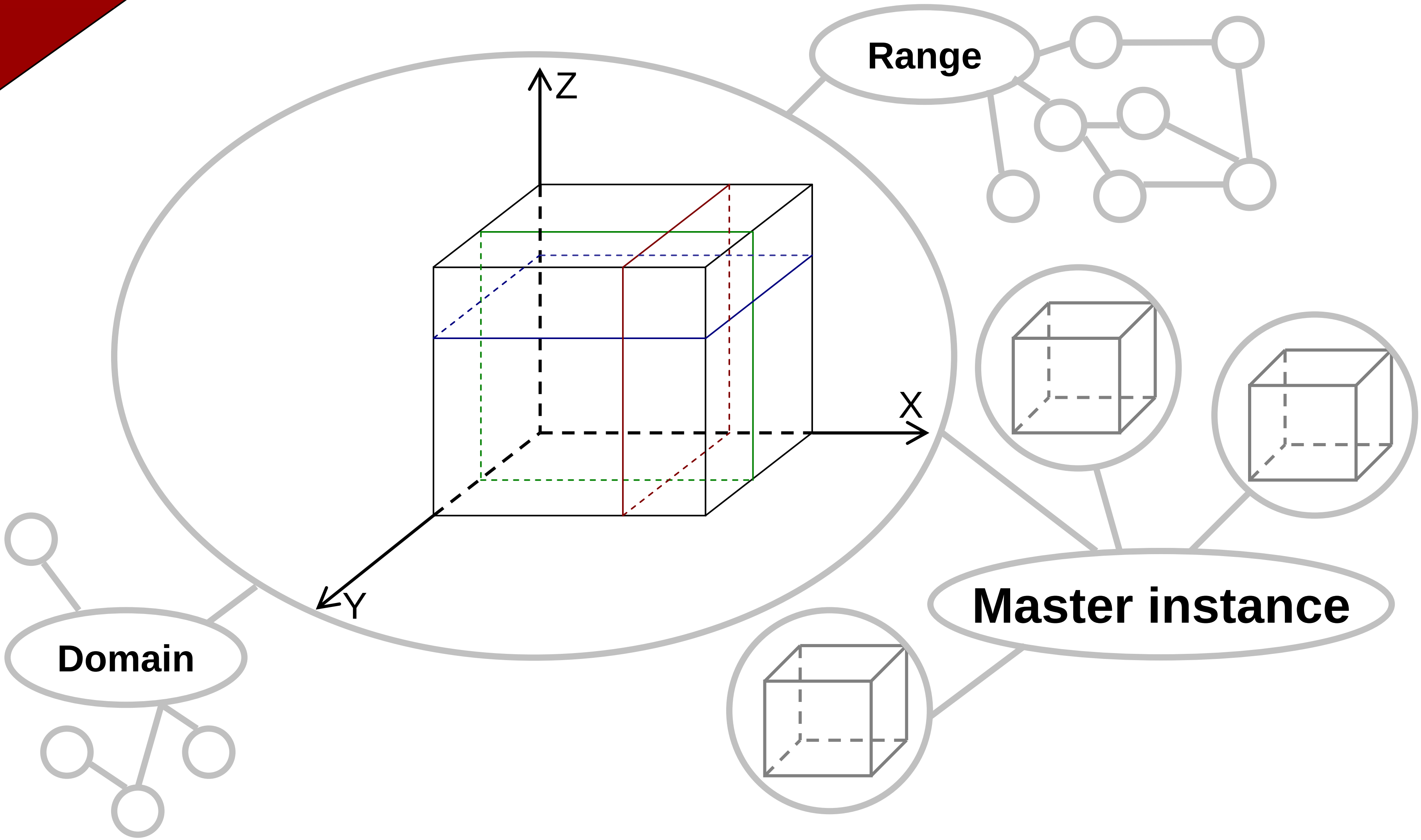
Introduction



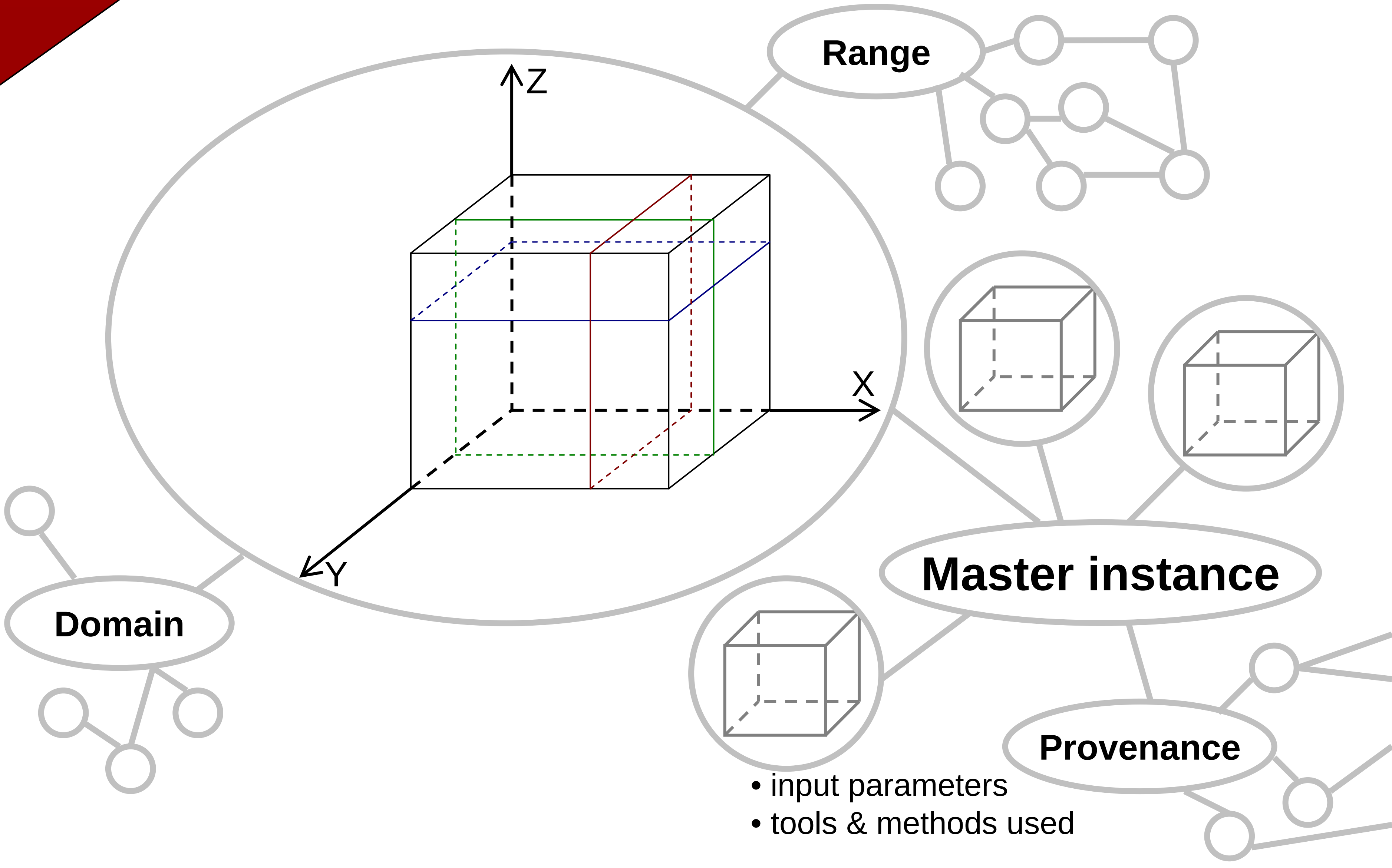
- field descriptions
- accuracy and noise
- reserved values

...

Introduction



Introduction



Introduction

They stay together...

Introduction

They stay together...

... since our data model is

RDF with Arrays

UPPSALA UNIVERSITET

RDF with Arrays databases

- RDF is very suitable for describing properties about scientific experiments (metadata) **but**:
    - Arrays are represented in a very inefficient way in RDF

- **Our approach**: Extend RDF with compact numerical array representation

Scientific SPARQL

- SPARQL is very suitable for searching scientific RDF-based metadata, **but**:
    - SPARQL has no support for queries involving array operations

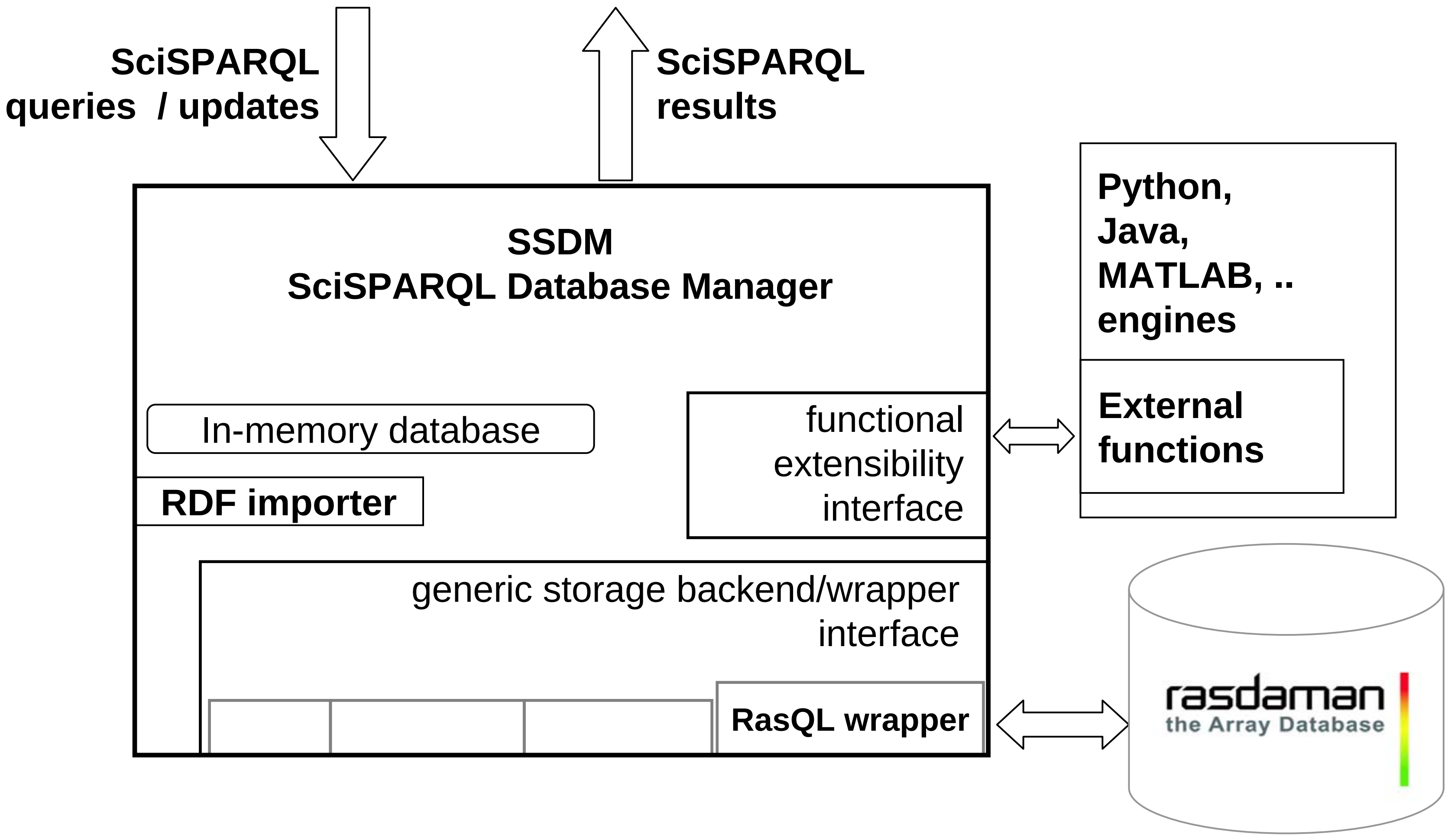
- **Our approach**: Extent SPARQL with common array operators => SciSPARQL

Reusing program libraries

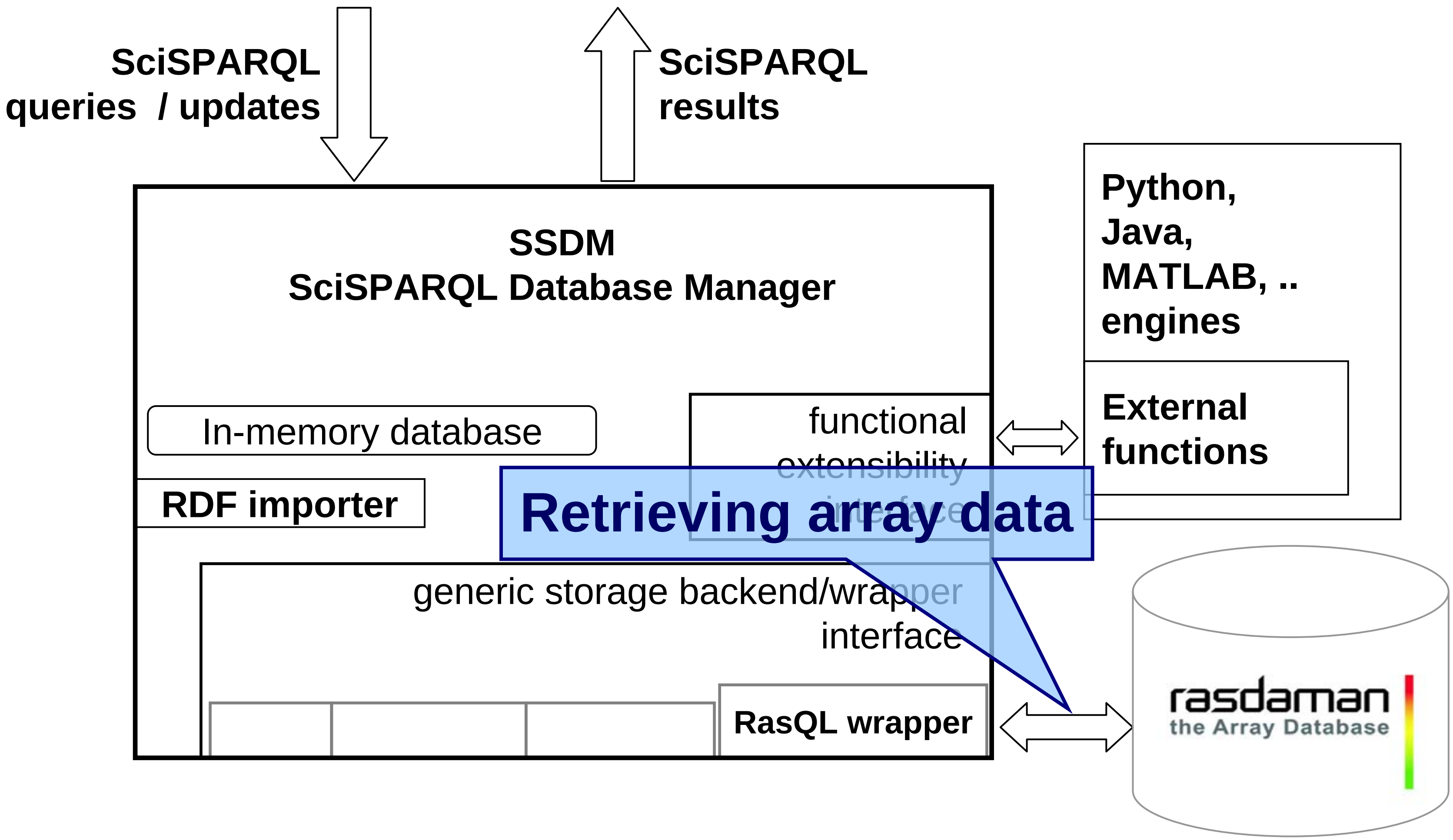
- Often need for using existing program libraries when processing experiments data, **but**:
    - SPARQL has no standard way of plugging in external program libraries and algorithms

- **Our approach**: SciSPARQL provides a general mechanism to call functions in C, Java, Python, or Matlab

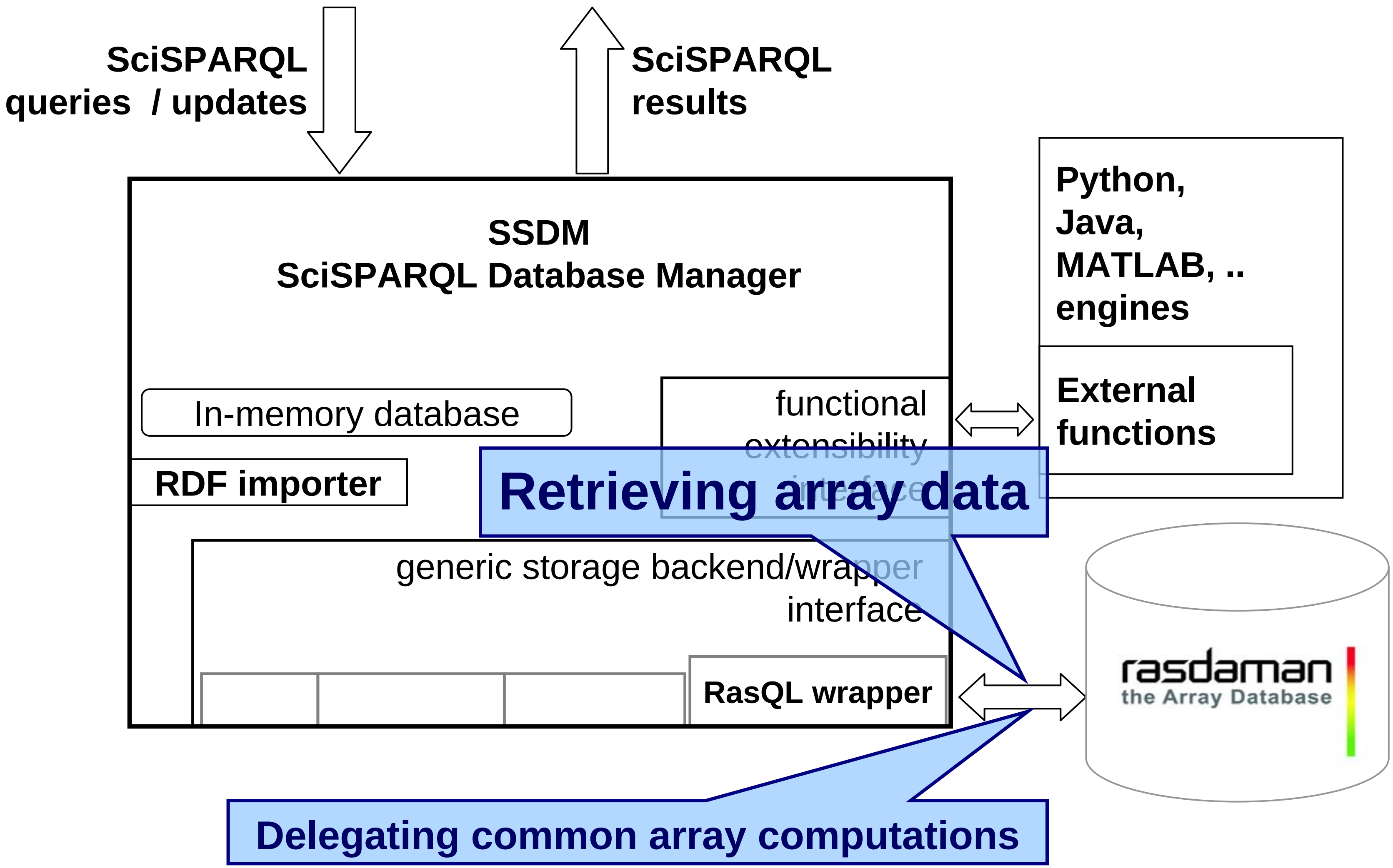
Our System Architecture



Our System Architecture



Our System Architecture



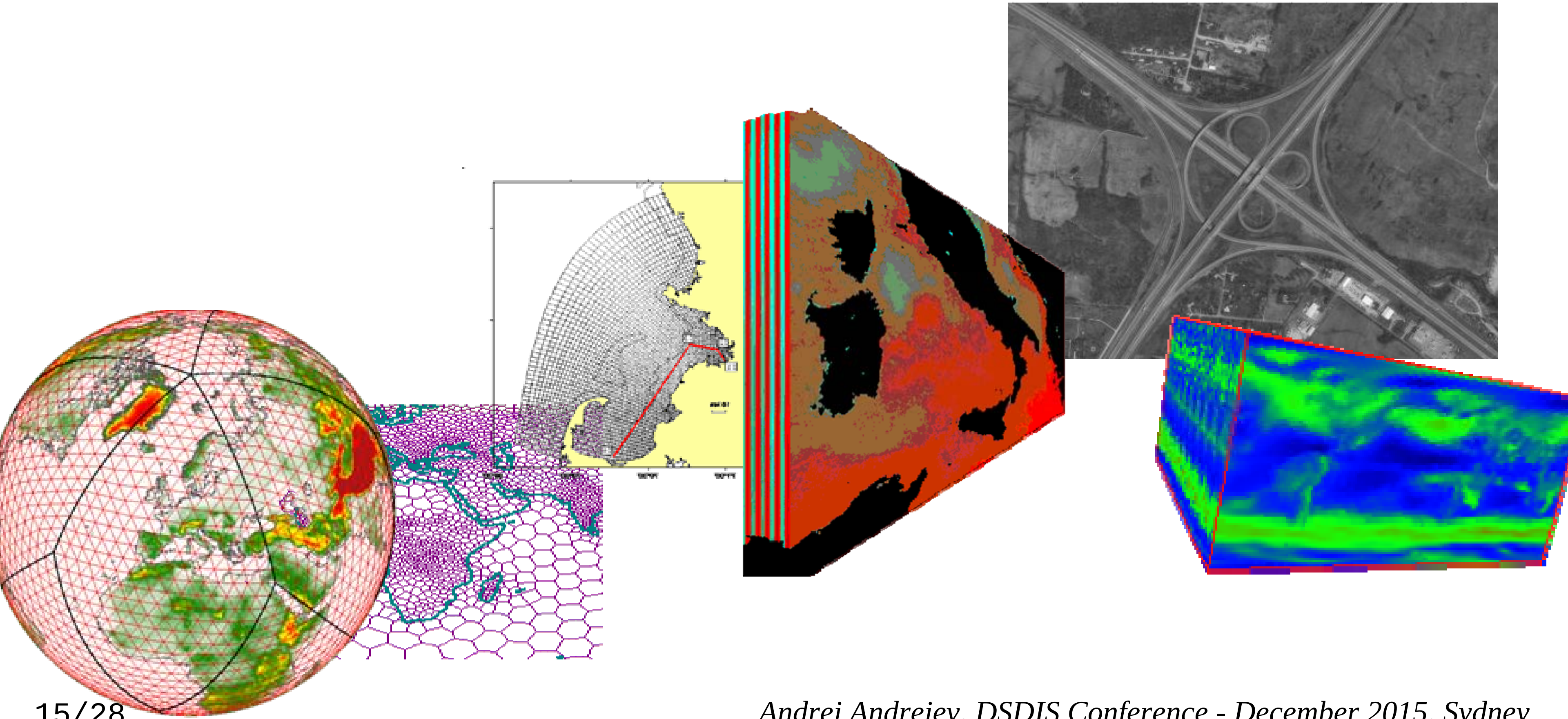
UPPSALA UNIVERSITET

Introducing Geo Coverages

A Coverage ...

A Coverage ...

... is an abstract data structure that represents some space/time varying phenomenon



Coverages

A Coverage ...

... is an abstract data structure that represents some space/time varying phenomenon

According to ISO 19123 and its implementation model by Open Geospatial Consortium:

Coverage subtypes are available for regular and irregular grids, point clouds and meshes

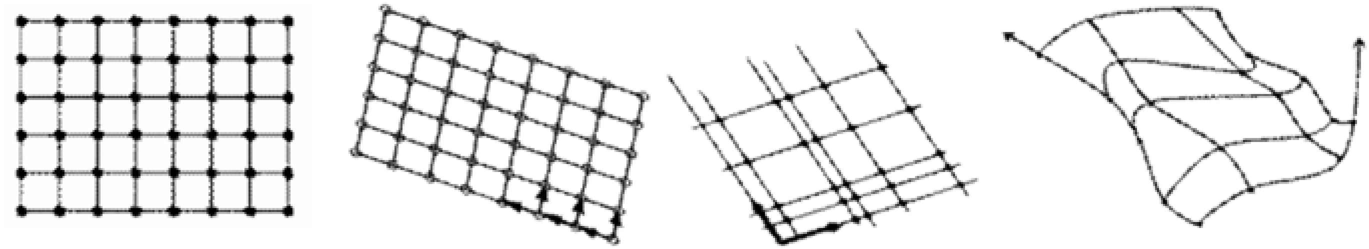
Coverages

A Coverage ...

... is an abstract data structure that represents some space/time varying phenomenon

According to ISO 19123 and its implementation model by Open Geospatial Consortium:

Coverage subtypes are available for regular and irregular grids, point clouds and meshes



A Grid Coverage

Grid Coverages

A Grid Coverage

is identified by

- **coverage id**

- **domain set**, specifying spatio-temporal region of interest
  - **geometric grid:** integer coordinates
  - **rectified grid:** grid origin and offset vector in some CRS
  - **referenceable grid:** CRS coordinates for each position

- **range type:** structural description and technical metadata, required of appropriate understanding of a coverage

Grid Coverages

A Grid Coverage

is identified by

- **coverage id**

- **domain set**, specifying spatio-temporal region of interest
  - **geometric grid:** integer coordinates
  - **rectified grid:** grid origin and offset vector in some CRS
  - **referenceable grid:** CRS coordinates for each position

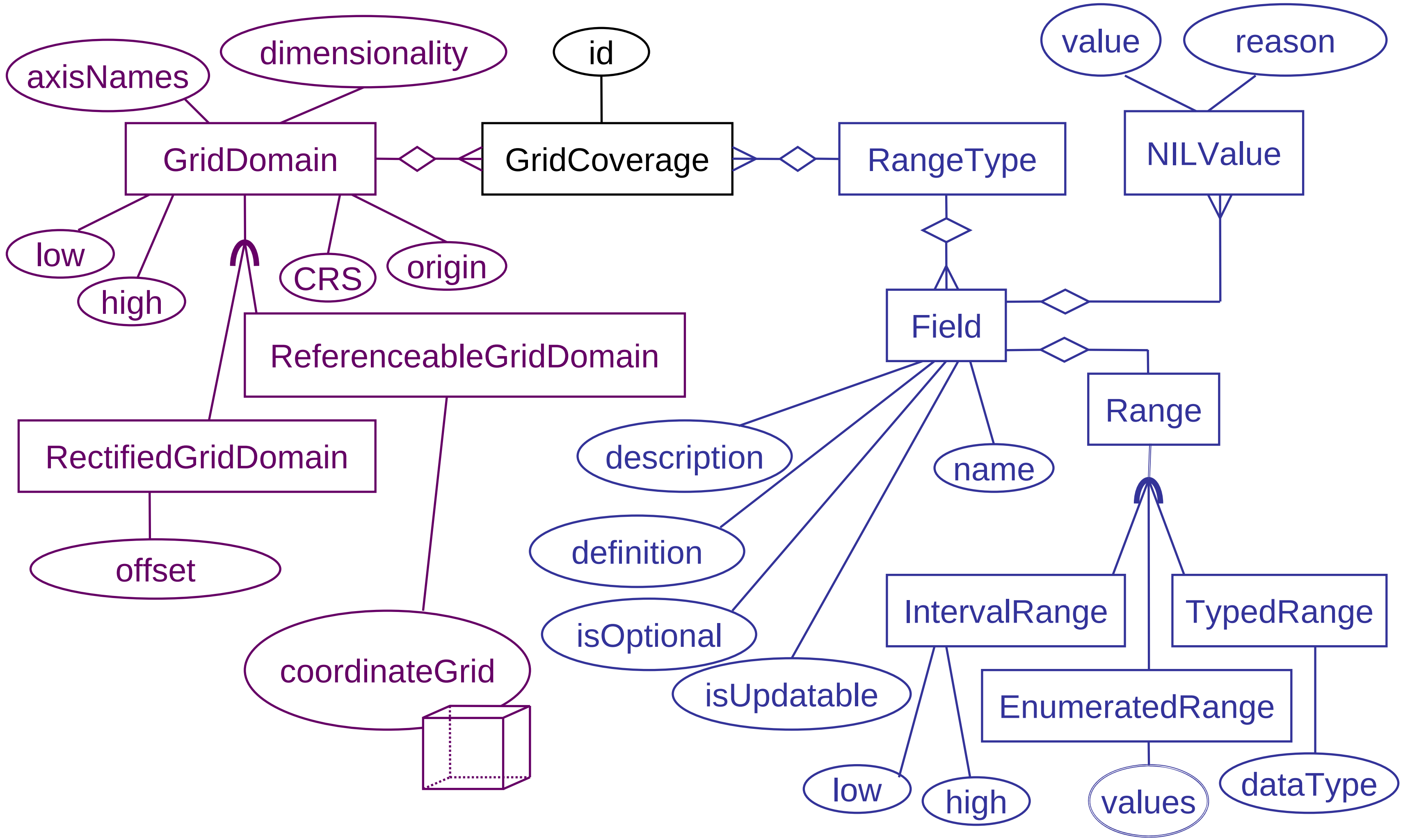
- **range type:** structural description and technical metadata, required of appropriate understanding of a coverage

A Range Type

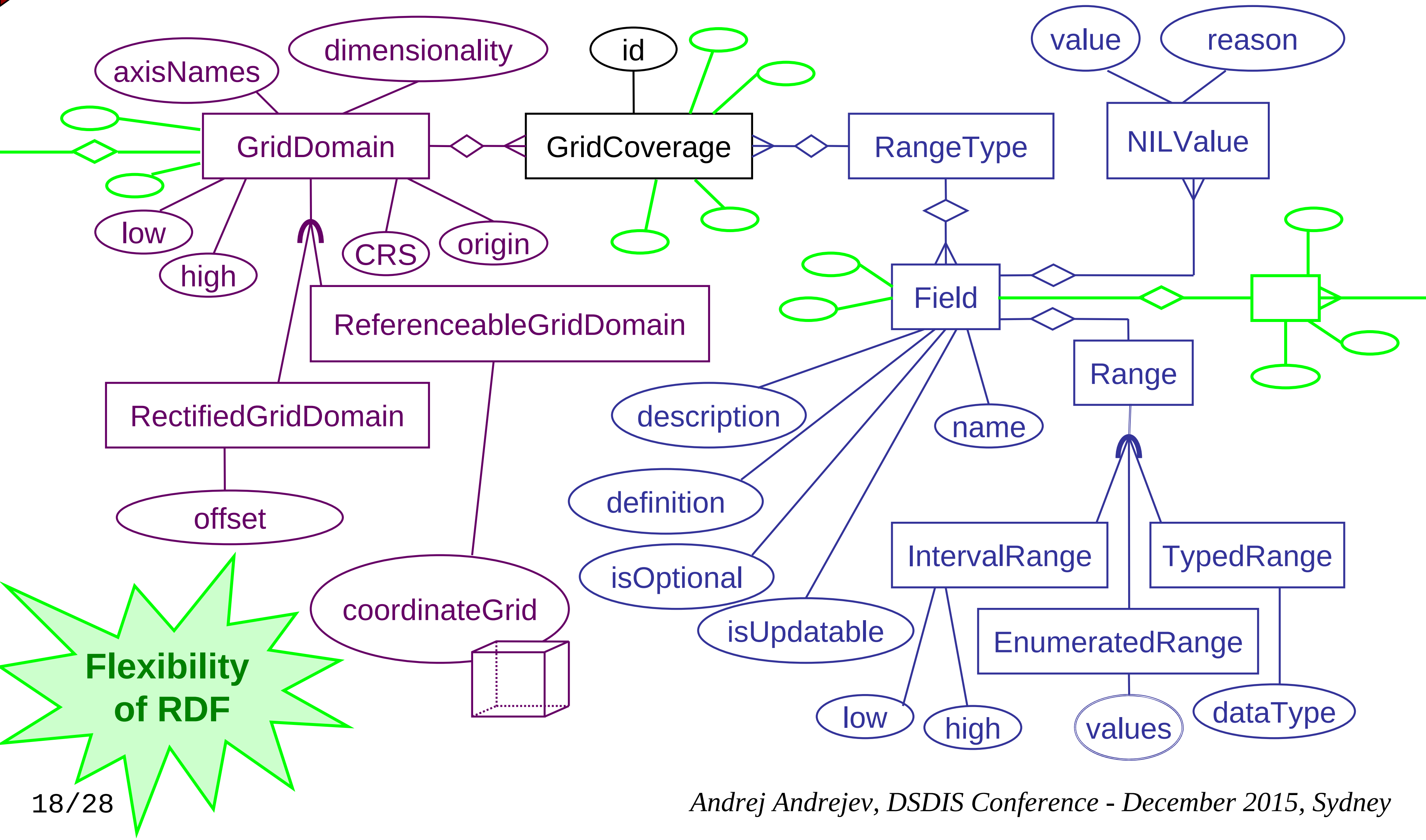
is composed of one or more fields, each having its own:

- **name** identifier
- human-readable **description**
- type definition
- unit of measure
- list of reserved values
- ...

Grid Coverage Ontology



Grid Coverage Ontology



SciSPARQL query example

SciSPARQL Example

Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
        AS ?result)
```

Create 0.5-threshold NDVI map of the coverage identified as "mycoverage", using its :nir and :red properties

```
SELECT (MAP(xsd:integer, NDVI(*, *, 0.5), ?nir_array, ?red_array)
        AS ?result)
 WHERE { ?c :id "mycoverage" ;
            :nir ?nir_array ;
            :red ?red_array }
```

SciSPARQL Example

Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
        AS ?result)
```

:real

Create 0.5-threshold NDVI map of the coverage identified as "mycoverage", using its :nir and :red properties

```
SELECT (MAP(xsd:integer, NDVI(*, *, 0.5), ?nir_array, ?red_array)
        AS ?result)
 WHERE { ?c :id "mycoverage" ;
            :nir ?nir_array ;
            :red ?red_array }
```

Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
        AS ?result)
```

:Boolean

Create 0.5-threshold NDVI map of the coverage identified as "`mycoverage`", using its `:nir` and `:red` properties

```
SELECT (MAP(xsd:integer, NDVI(*, *, 0.5), ?nir_array, ?red_array)
       AS ?result)
 WHERE { ?c :id "mycoverage" ;
            :nir ?nir_array ;
            :red ?red_array }
```

SciSPARQL Example

Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
        AS ?result)
```

→ either 0 or 255

:integer

Create 0.5-threshold NDVI map of the coverage identified as "`mycoverage`", using its `:nir` and `:red` properties

```
SELECT (MAP(xsd:integer, NDVI(*, *, 0.5), ?nir_array, ?red_array)
        AS ?result)
 WHERE { ?c :id "mycoverage" ;
            :nir ?nir_array ;
            :red ?red_array }
```

SciSPARQL Example

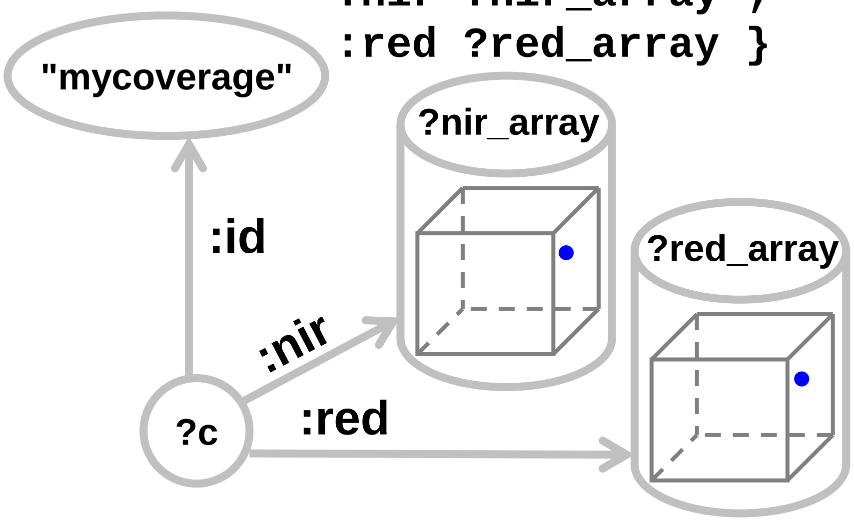
Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
        AS ?result)
```

either 0 or 255 :integer

Create 0.5-threshold NDVI map of the coverage identified as "`mycoverage`", using its `:nir` and `:red` properties

```
SELECT (MAP(xsd:integer, NDVI(*, *, 0.5), ?nir_array, ?red_array)
        AS ?result)
 WHERE { ?c :id "mycoverage" ;
            :nir ?nir_array ;
            :red ?red_array }
```



SciSPARQL Example

Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
        AS ?result)
```

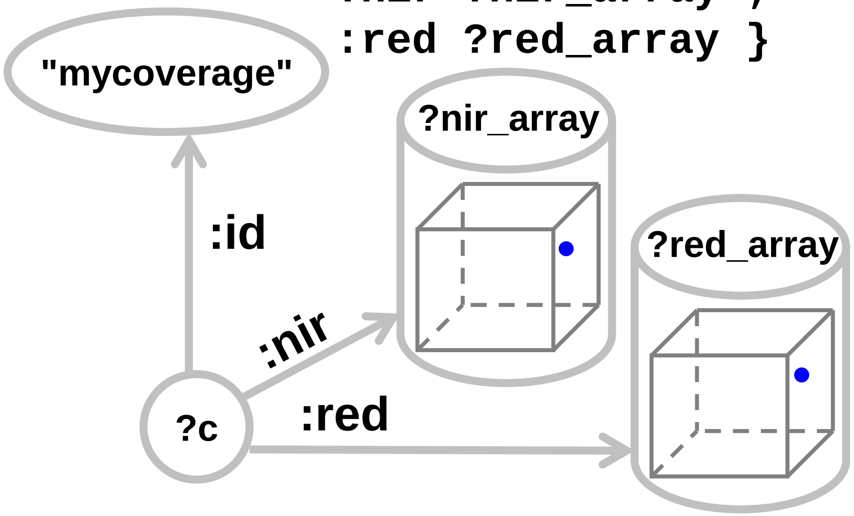
:integer

either 0 or 255

Create 0.5-threshold NDVI map of the coverage identified as "mycoverage", using its :nir and :red properties

```
SELECT (MAP(xsd:integer, NDVI(*, *, 0.5), ?nir_array, ?red_array)
       AS ?result)
 WHERE { ?c :id "mycoverage" ;
            :nir ?nir_array ;
            :red ?red_array }
```

lexical closure



SciSPARQL Example

Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
        AS ?result)
```

:integer

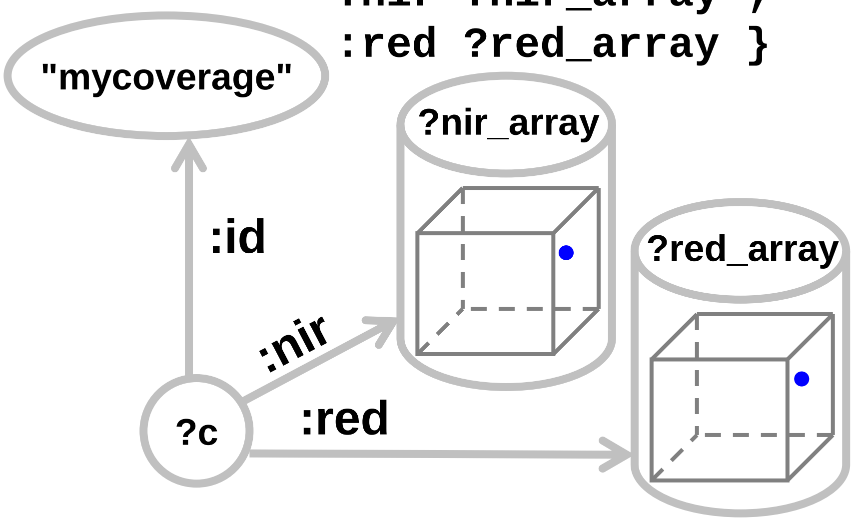
either 0 or 255

Create 0.5-threshold NDVI map of the coverage identified as "mycoverage", using its :nir and :red properties

```
SELECT (MAP(xsd:integer, NDVI(*, *, 0.5), ?nir_array, ?red_array)
       AS ?result)
 WHERE { ?c :id "mycoverage" ;
            :nir ?nir_array ;
            :red ?red_array }
```

lexical closure

'free' arguments - bound by the mapper



SciSPARQL Example

Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
        AS ?result)
```

:integer

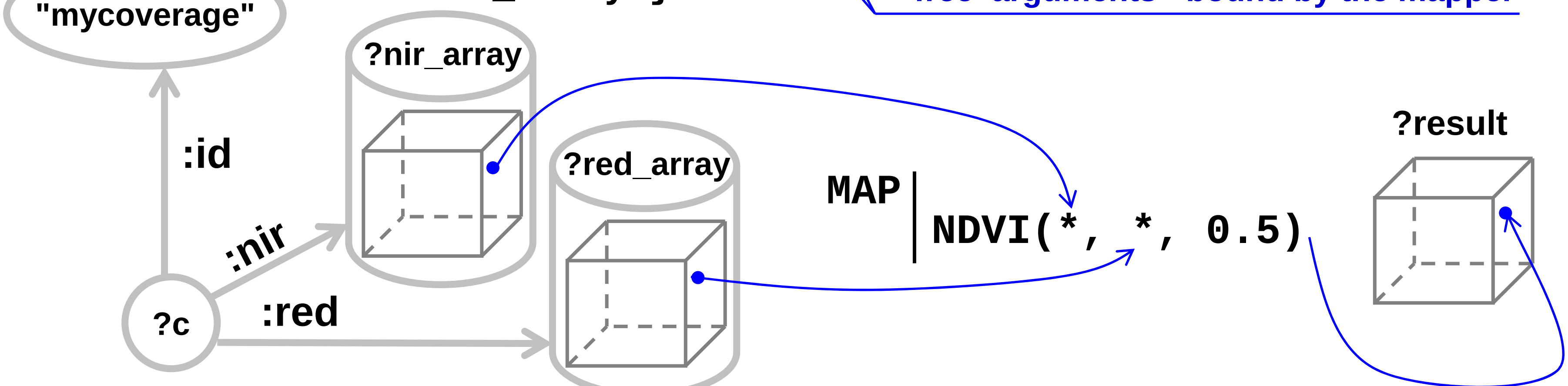
either 0 or 255

Create 0.5-threshold NDVI map of the coverage identified as "mycoverage", using its :nir and :red properties

```
SELECT (MAP(xsd:integer, NDVI(*, *, 0.5), ?nir_array, ?red_array)
        AS ?result)
 WHERE { ?c :id "mycoverage" ;
            :nir ?nir_array ;
            :red ?red_array }
```

lexical closure

'free' arguments - bound by the mapper



SciSPARQL Example

Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
        AS ?result)
```

either 0 or 255

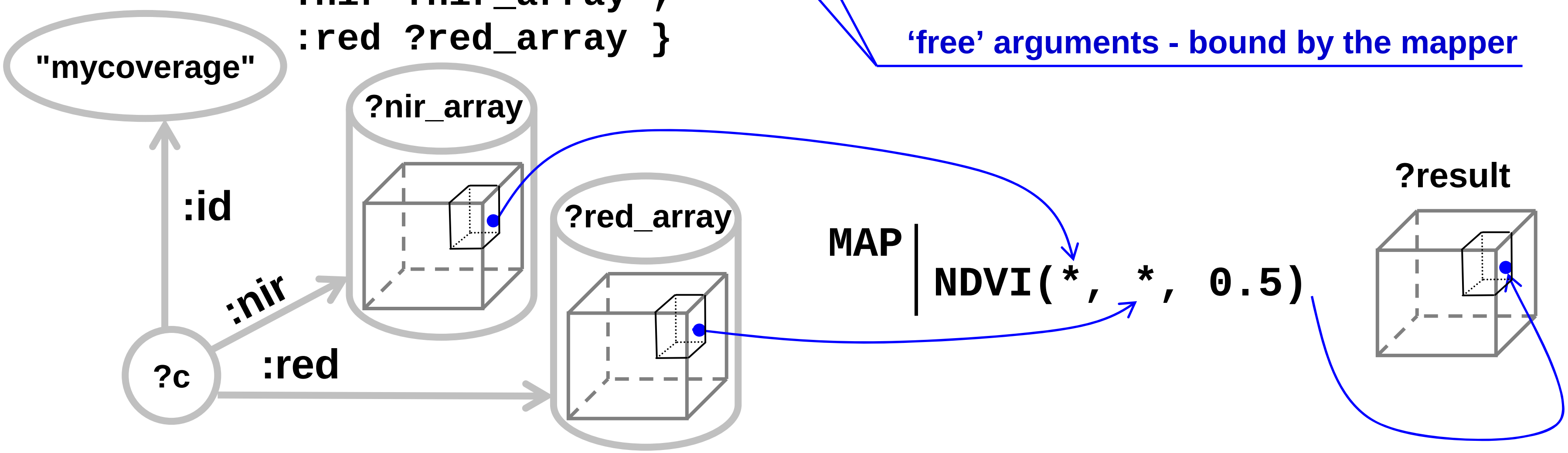
:integer

Create 0.5-threshold NDVI map of the coverage identified as "mycoverage", using its :nir and :red properties

```
SELECT (MAP(xsd:integer, NDVI(*, *, 0.5), ?nir_array, ?red_array)
       AS ?result)
 WHERE { ?c :id "mycoverage" ;
            :nir ?nir_array ;
            :red ?red_array }
```

lexical closure

'free' arguments - bound by the mapper



SciSPARQL Example

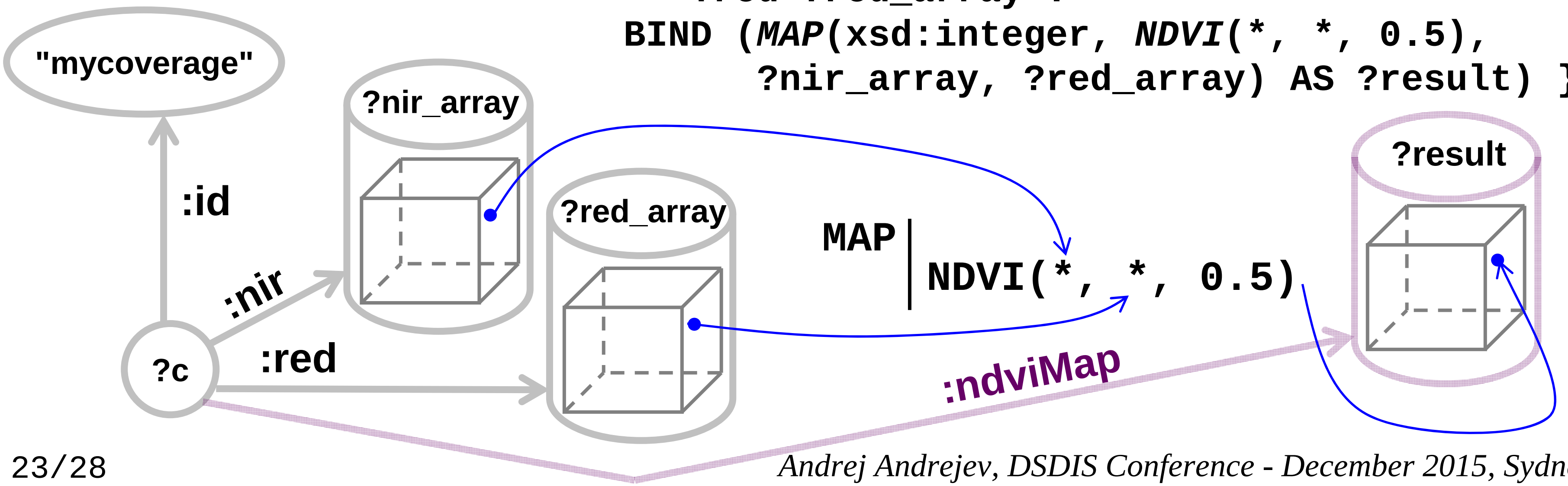
Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
        AS ?result)
```

→ either 0 or 255   :integer

Create 0.5-threshold NDVI map of the coverage identified as "mycoverage", using its :nir and :red properties, and insert it as :ndviMap property

```
INSERT { ?c :ndviMap ?result }
 WHERE { ?c :id "mycoverage" ;
            :nir ?nir_array ;
            :red ?red_array .
        BIND (MAP(xsd:integer, NDVI(*, *, 0.5),
              ?nir_array, ?red_array) AS ?result) }
```



Comparison

Doing the same thing
in a programming language of your choice?

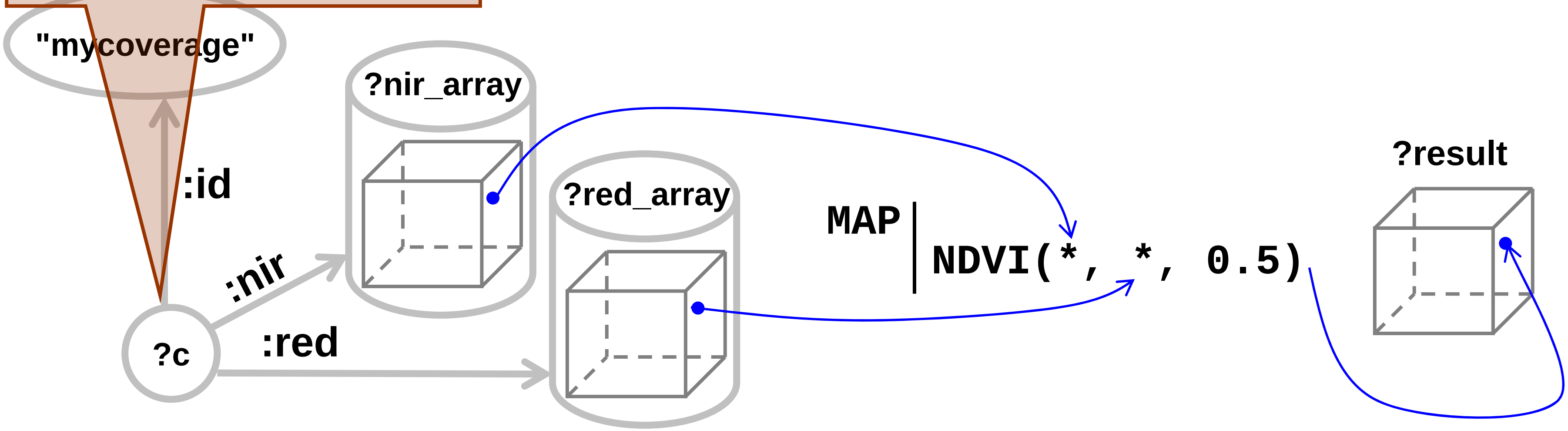
Comparison

Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
        AS ?result)
```

Create 0.5-threshold NDVI map of the coverage identified as "`mycoverage`", using its `:nir` and `:red` properties

Find the right instance of a Gridded Coverage and all the associated data needed for the task

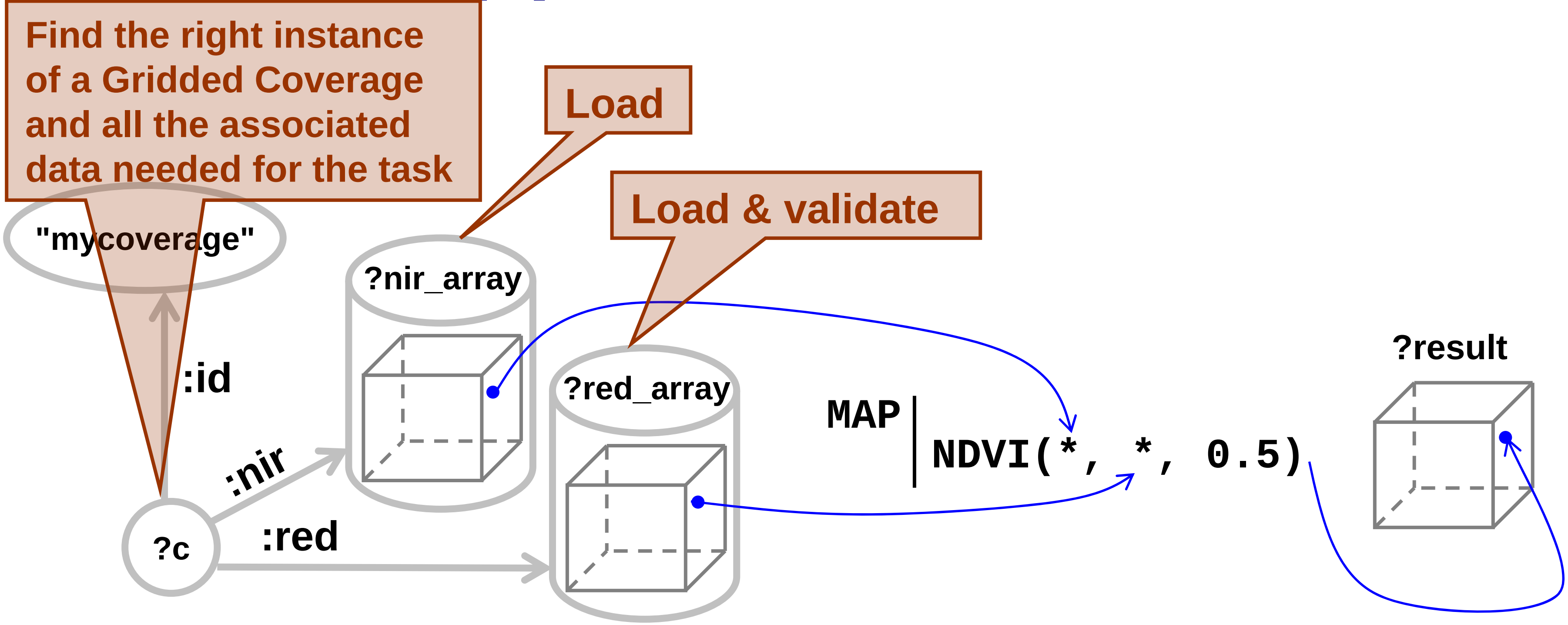


Comparison

Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
       AS ?result)
```

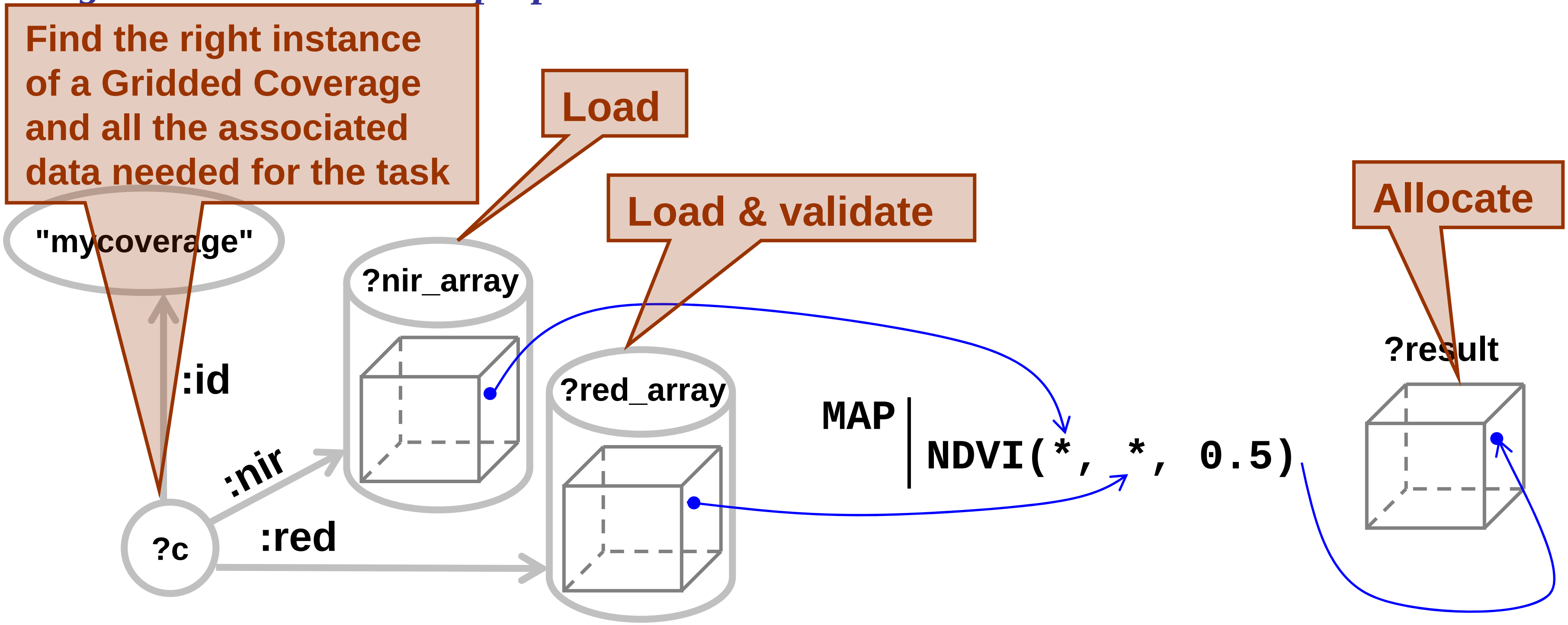
Create 0.5-threshold NDVI map of the coverage identified as "`mycoverage`", using its `:nir` and `:red` properties



Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
        AS ?result)
```

Create 0.5-threshold NDVI map of the coverage identified as "`mycoverage`", using its `:nir` and `:red` properties

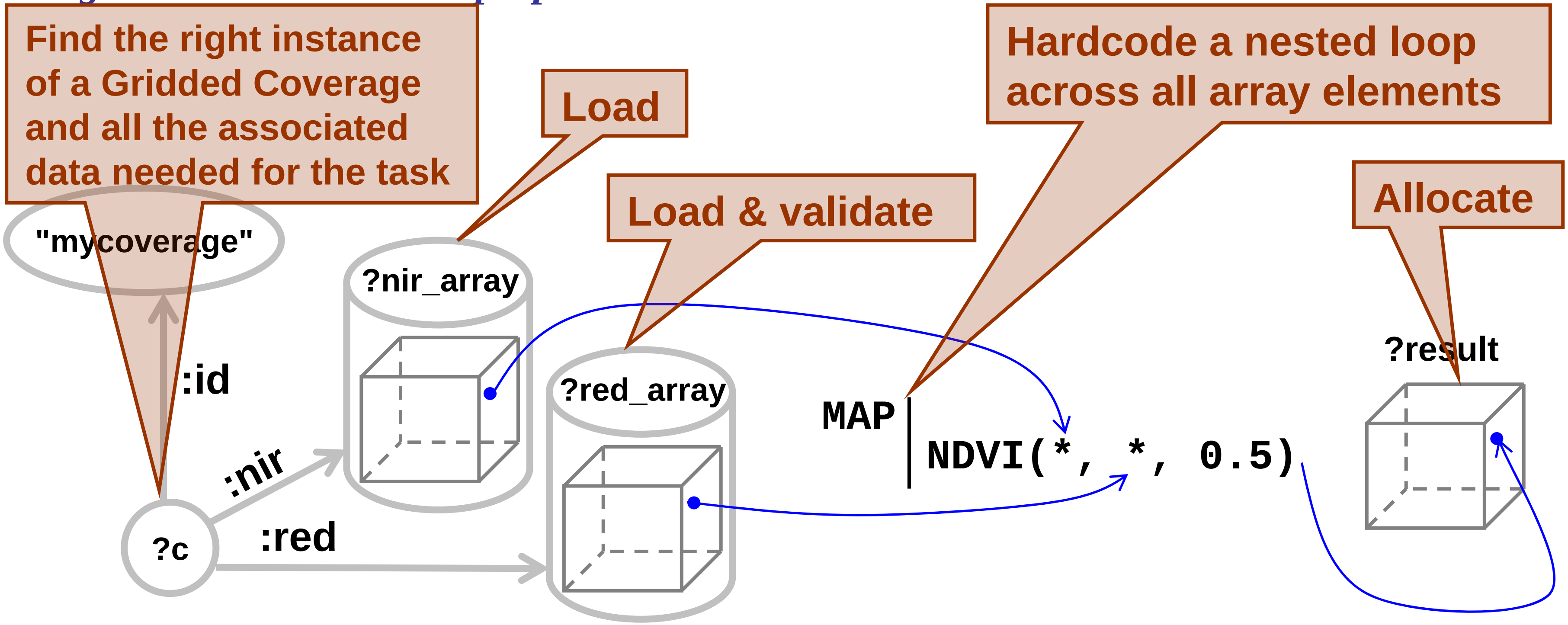


Comparison

Normalized Difference Vegetation Index (NDVI)

```
DEFINE FUNCTION NDVI(?nir ?red ?x) AS
SELECT (255 * xsd:integer(((?nir - ?red) / (?nir + ?red)) > ?x)
        AS ?result)
```

Create 0.5-threshold NDVI map of the coverage identified as "`mycoverage`", using its `:nir` and `:red` properties



Benefits of combining data and metadata within a single query:

- **More transparent and self-contained queries**
  are easier to maintain

- **Result selection and postprocessing on the server**
  whenever metadata is needed, it is in the same place

- **Single round-trip to the server**
  instead of retrieving metedata in order to guide the data retrieval and processing

- **More freedom for the optimizer**
  one complex query is better than a series of simple ones

UPPSALA UNIVERSITET

Summary of contributions

- **RDF with Arrrays**

a flexible way to model Gridded Coverages with any application-relevant metadata

- **Hybrid data store**

storing RDF graphs in-memory and gridded data on disk (distributed and parallel)

- **SciSPARQL queries**

combining graph patterns and array operations: metadata conditions and data filtering / postprocessing

- **Mediator architecture**

performing computations where the data are

The software, documentation, and examples are available at

http://user.it.uu.se/~udbl/SciSPARQL

This work is supported by