OEGTalks

Enabling Ontology-based Access to Streaming Data Sources

Jean-Paul Calbimonte[1], Oscar Corcho[1], Alasdair J G Gray[2]

[1]Ontology Engineering Group. Departamento de Inteligencia Artificial.
Facultad de Informática, Universidad Politécnica de Madrid.
Campus de Montegancedo s/n. 28660 Boadilla del Monte. Madrid. Spain
jp.calbimonte@upm.es;ocorcho@fi.upm.es

[2]School of Computer Science, The University of Manchester,
Oxford Road, Manchester M13 9PL, United Kingdom
a.gray@cs.man.ac.uk

Date: 28/10/2010

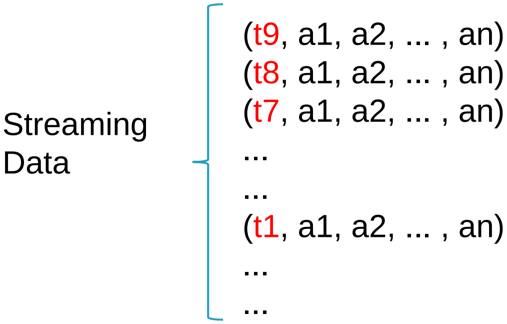
Outline

- Introduction & Motivation
- Background
- Approach
- Mapping Streams to Ontologies
- SPARQL Stream
- Query Translation
- Example
- Future work
- Conclusions

Introduction & Scope

- Sensor technologies
  - Ubiquitous data capture
  - Data processing
  - Cheap
  - Noisy, Unreliable
  - Low computational, power resources, storage

- Streaming Data
  - Continuously appended data
  - Potentially infinite
  - Time-stamped tuples
  - Continuous queries
  - Latest used in queries



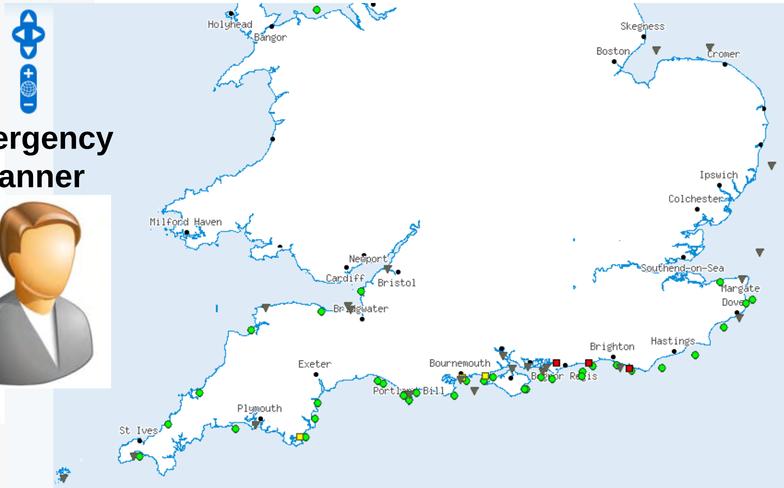
Streaming Data

(t9, a1, a2, ... , an)
(t8, a1, a2, ... , an)
(t7, a1, a2, ... , an)
...
...
(t1, a1, a2, ... , an)
...
...

- Applications in security surveillance, healthcare provision, environmental monitoring, you name it.

Motivation

Flood risk alert:
South East England



Emergency planner

Channel Coastal Observatory → Real-time data — Wave, Wind, Tide

Met Office → Meteorological forecasts

Environment Agency → Flood defences data

Other sources

...
...
... →

- Detect conditions likely to cause a flood
- Present data model in terms of the user domain: e.g. Flood risk assessment

Example:

- “provide me with the wind speed observations average over the last minute in the Solent region, if it is higher than the average of the last 2 to 3 hours”

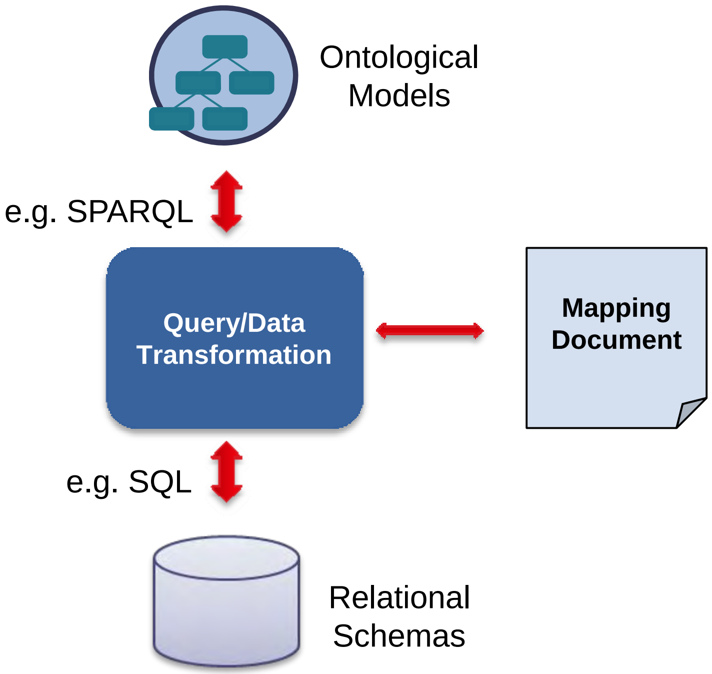
Motivation

- Ontologies can be used as such a common model
- Answer the requirements:
  - Establish mappings between ontological models and streaming data source schemas
  - Access streaming data sources through queries over ontology models

Background – Ontology-based Data Access

Generate Semantic Web content from existing relational data sources

Ontology-based Data Access

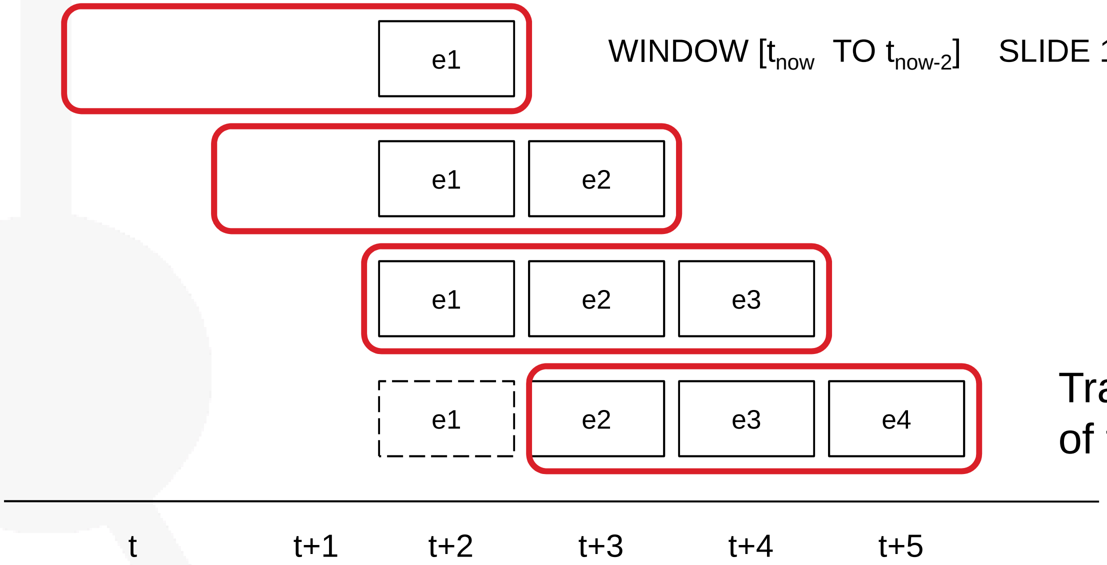


- R_2O + ODEMapster
- D2RQ
- SquirrelRDF
- RDBToOnto
- Relational.OWL
- SPASQL
- Virtuoso
- MASTRO

Background – Querying Relational Data Streams

Streaming Data

Event Streams
Acquisitional Streams



WINDOW [t_{now} TO t_{now-2}] SLIDE 1

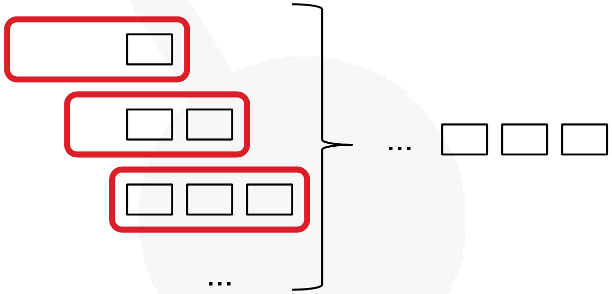
STREAM
Aurora/Borealis
Cougar
TinyDB
SNEE

Query engines

CQL
SNEEql
TinyQL

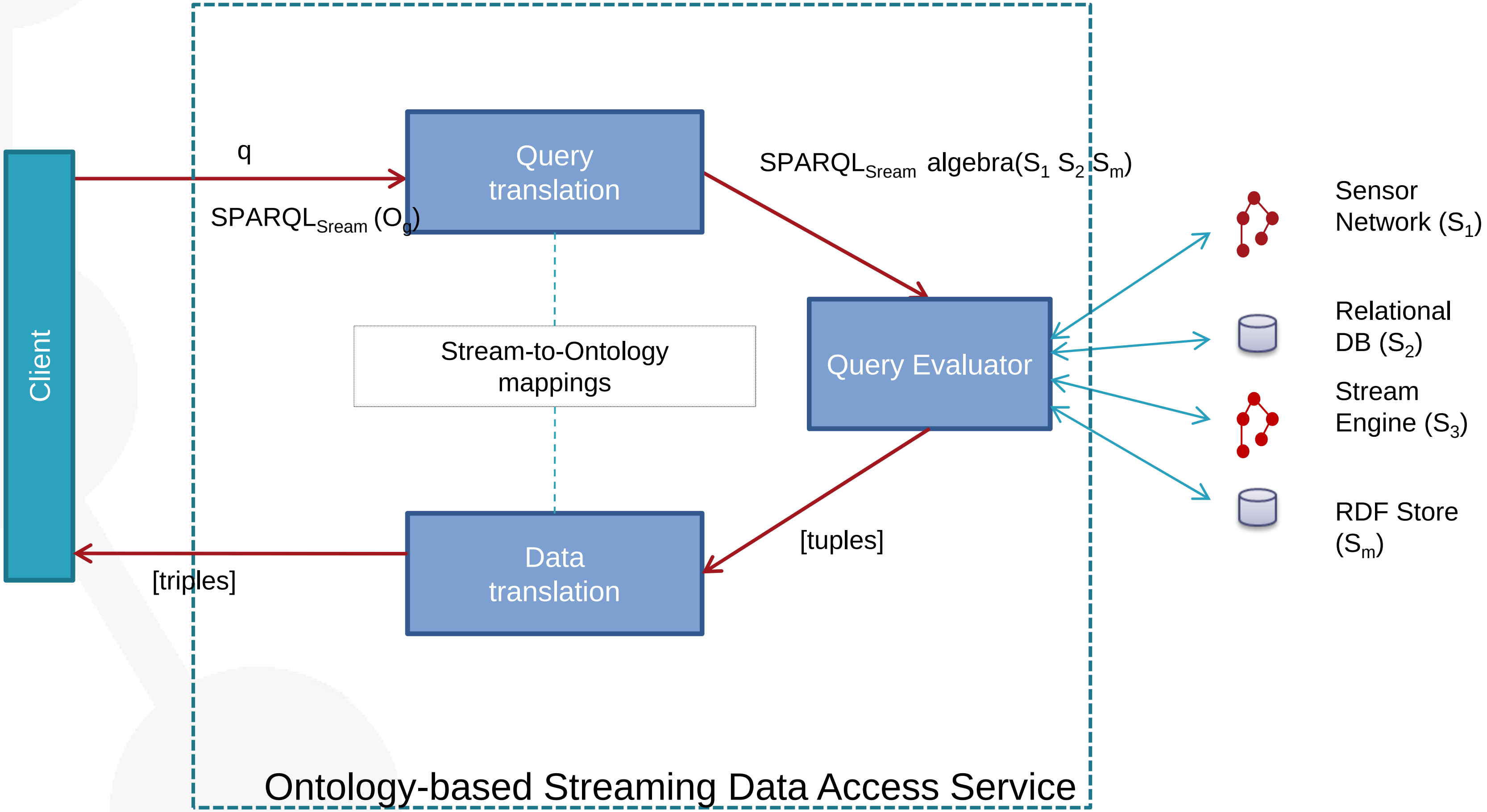
Query languages

Transform infinite sequence of tuples to bounded bag



Window-to-Stream operators: convert stream of windows to stream of tuples

Ontology-based Streaming Data Access



Ontology-based Streaming Data Access

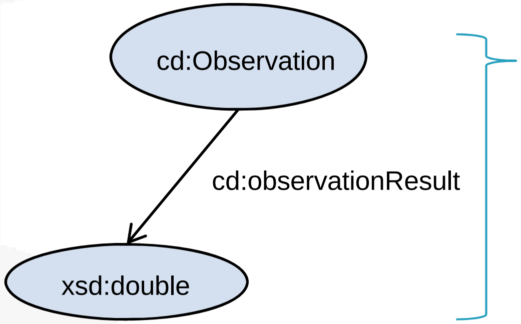
- Mappings from relational streams to ontological concepts
    - Extend stored data schema mappings
    - Study translation semantics

- Provide with a stream query language at ontological level
    - Use notion of RDF stream
    - Extend SPARQL
    - Window operator, window-to-stream operators

$\text{SPARQL}_{\text{Stream}}$

Example:

- “provide me with the wind speed observations over the last minute in the Solent Region ”



STREAM
<*http://www.semsorgrid4env.eu/ccometeo.srdf*>

…
…
(<ssg4e:Obs1,rdf:type, cd:Observation>, t_i),
(<ssg4e:Obs1,cd:observationResult,”34.5”>, t_i),
(<ssg4e:Obs2,rdf:type, cd:Observation>, t_{i+1}),
(<ssg4e:Obs2,cd:observationResult,”20.3”>, t_{i+1}),
…
…

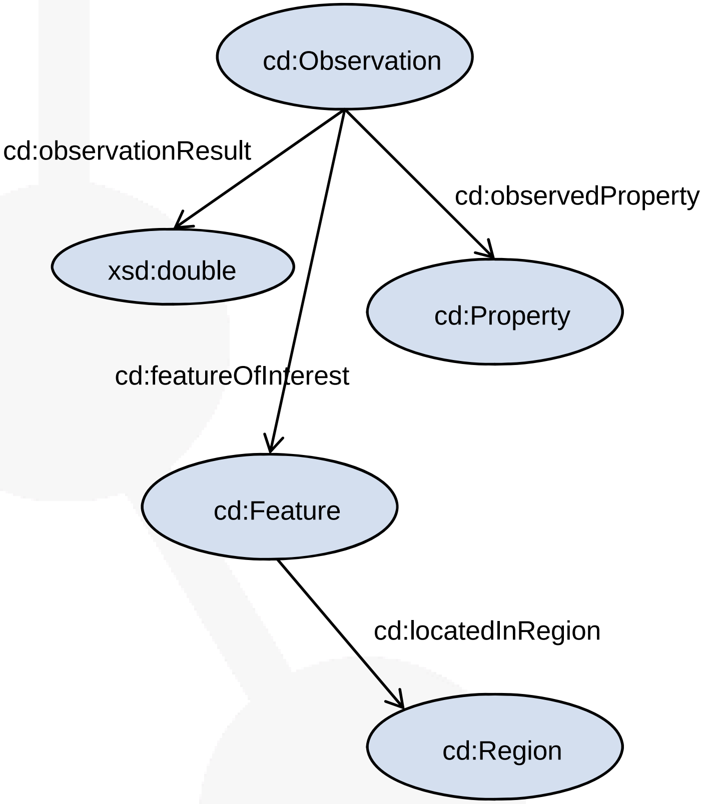
- RDF-Stream

…
…
($<s_{i-1}, p_{i-1}, o_{i-1}>$, t_{i-1}),
($<s_i, p_i, o_i>$, t_i),
($<s_{i+1}, p_{i+1}, o_{i+1}>$, t_{i+1}),
…
…

SPARQL$_{\text{Stream}}$

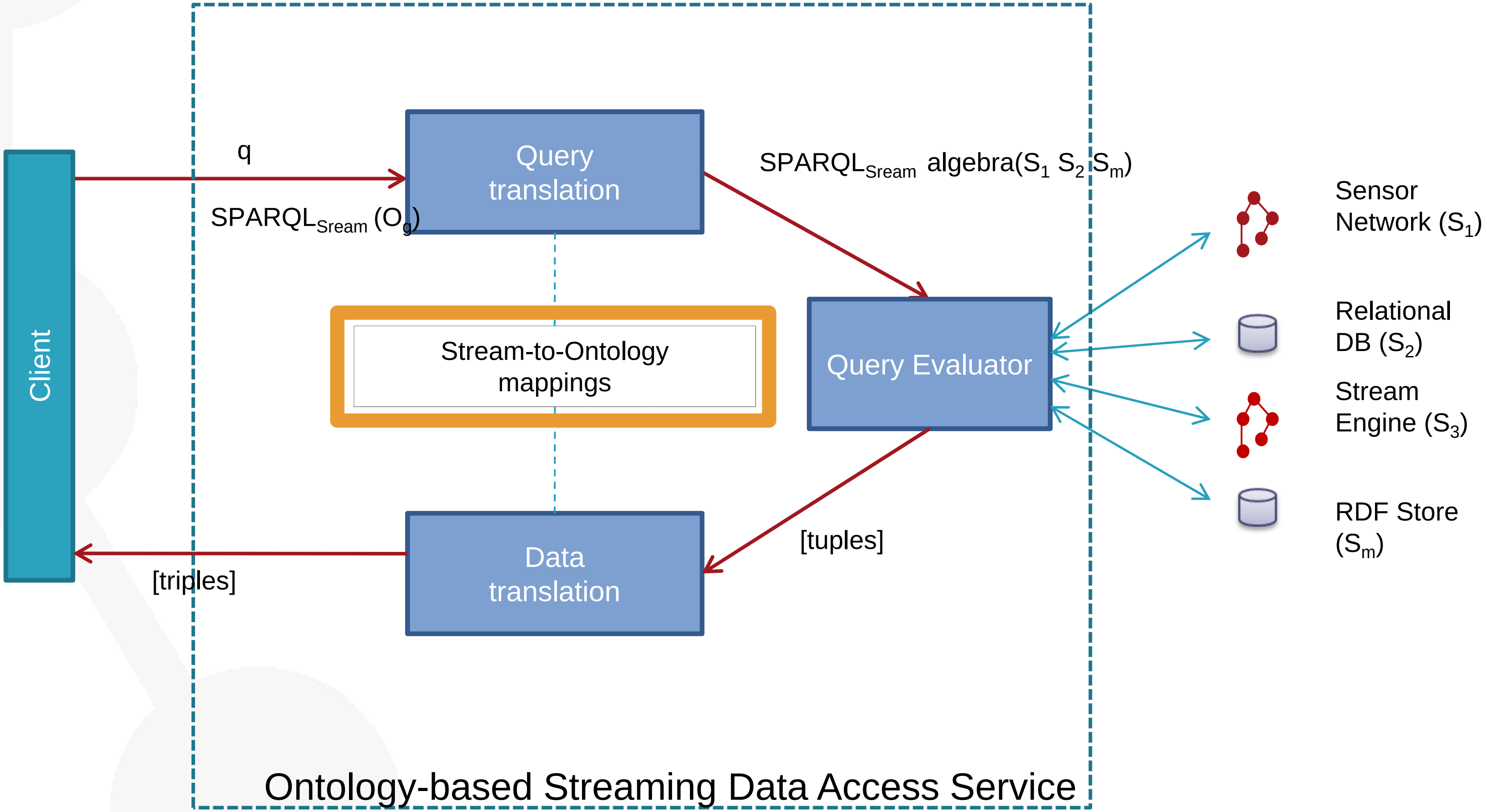
Example:

- “provide me with the wind speed observations over the last minute in the Solent Region ”



```
PREFIX cd: <http://www.semsorgrid4env.eu/ontologies/CoastalDefences.owl#>
PREFIX sb: <http://www.w3.org/2009/SSN-XG/Ontologies/SensorBasis.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT  ?windspeed ?windts
FROM STREAM <http://www.semsorgrid4env.eu/ccometeo.srdf>
[ NOW – 1 MINUTE TO NOW – 0 MINUTES ]
WHERE
{
 ?WindObs a cd:Observation;
   cd:observationResult ?windspeed;
   cd:observationResultTime ?windts;
   cd:observedProperty ?windProperty;
   cd:featureOfInterest ?windFeature.
 ?windFeature a cd:Feature;
   cd:locatedInRegion cd:SolentCCO.
 ?windProperty a cd:WindSpeed.
 }
```

Ontology-based Streaming Data Access



S2O Mappings

- R2O Mappings
  - Schema descriptions (probably mostly unused)

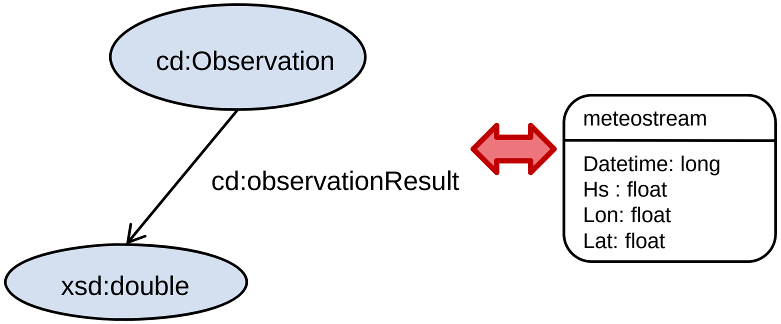
```
<has-table name="raingauge">
    <keycol-desc name="id"/>
    <keycol-desc name="ts"/>
    <nonkeycol-desc name="level"/>
</has-table>
```

```
<has-stream name="windsamples" streamType="push">
  <keycol-desc name="id"/>
  <keycol-desc name="ts"/>
  <timestamp-desc name="ts" />
  <nonkeycol-desc name="speed"/>
  <nonkeycol-desc name="temperature"/>
  <nonkeycol-desc name="direction"/>
</has-stream>
```

S2O Mappings

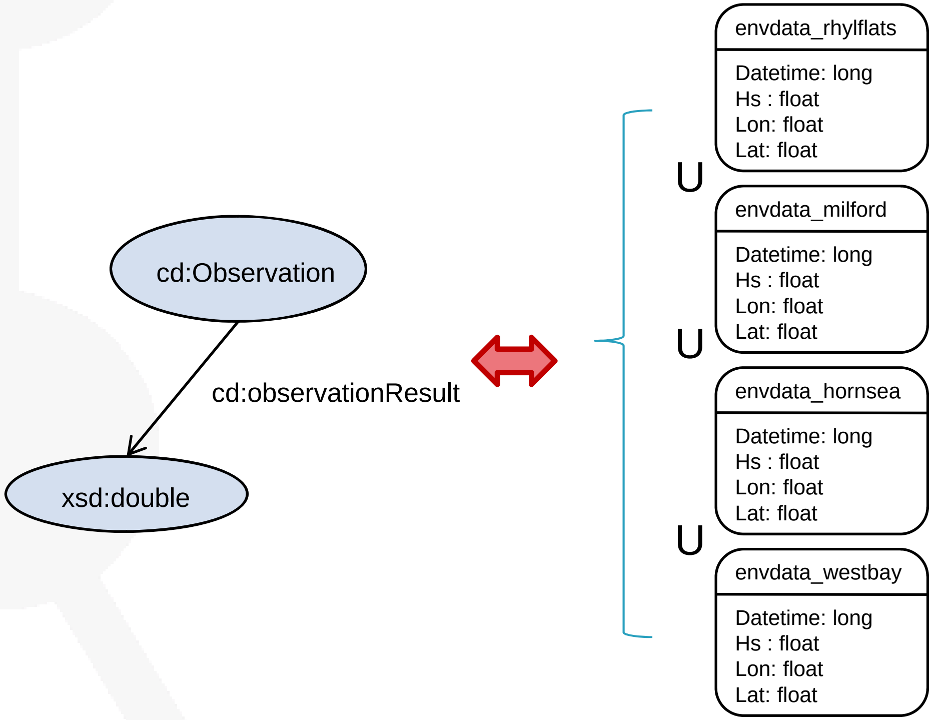
```
<conceptmap-def id="Observation_wind"
name="http://www.semsorgrid4env.eu/ontologies/CoastalDefences.owl#Observation"
virtualStream="http://www.semsorgrid4env/ccometeo.srdf">
 <uri-as>
   <operation oper-id="concat">
        <arg-restriction on-param="string1">
           <has-value>http://www.semsorgrid4env.eu/data#ObservationWind</has-value>
       </arg-restriction>
       <arg-restriction on-param="string2">
           <has-column>meteostream.DateTime</has-column>
       </arg-restriction>
   </operation>
 </uri-as>


<attributemap-def
name="http://www.semsorgrid4env.eu/ontologies/CoastalDefences.owl#observationResult"
dataType="xsd:double">
<selector>
 <aftertransform>
  <operation oper-id="constant">
   <arg-restriction on-param="const-val">
    <has-column>meteostream.Hs</has-column>
   </arg-restriction>
  </operation>
 </aftertransform>
</selector>
</attributemap-def>
```



S2O Mappings

- Union of extents-streams



cd:Observation

cd:observationResult

xsd:double

envdata_rhylflats
Datetime: long
Hs : float
Lon: float
Lat: float

U

envdata_milford
Datetime: long
Hs : float
Lon: float
Lat: float

U

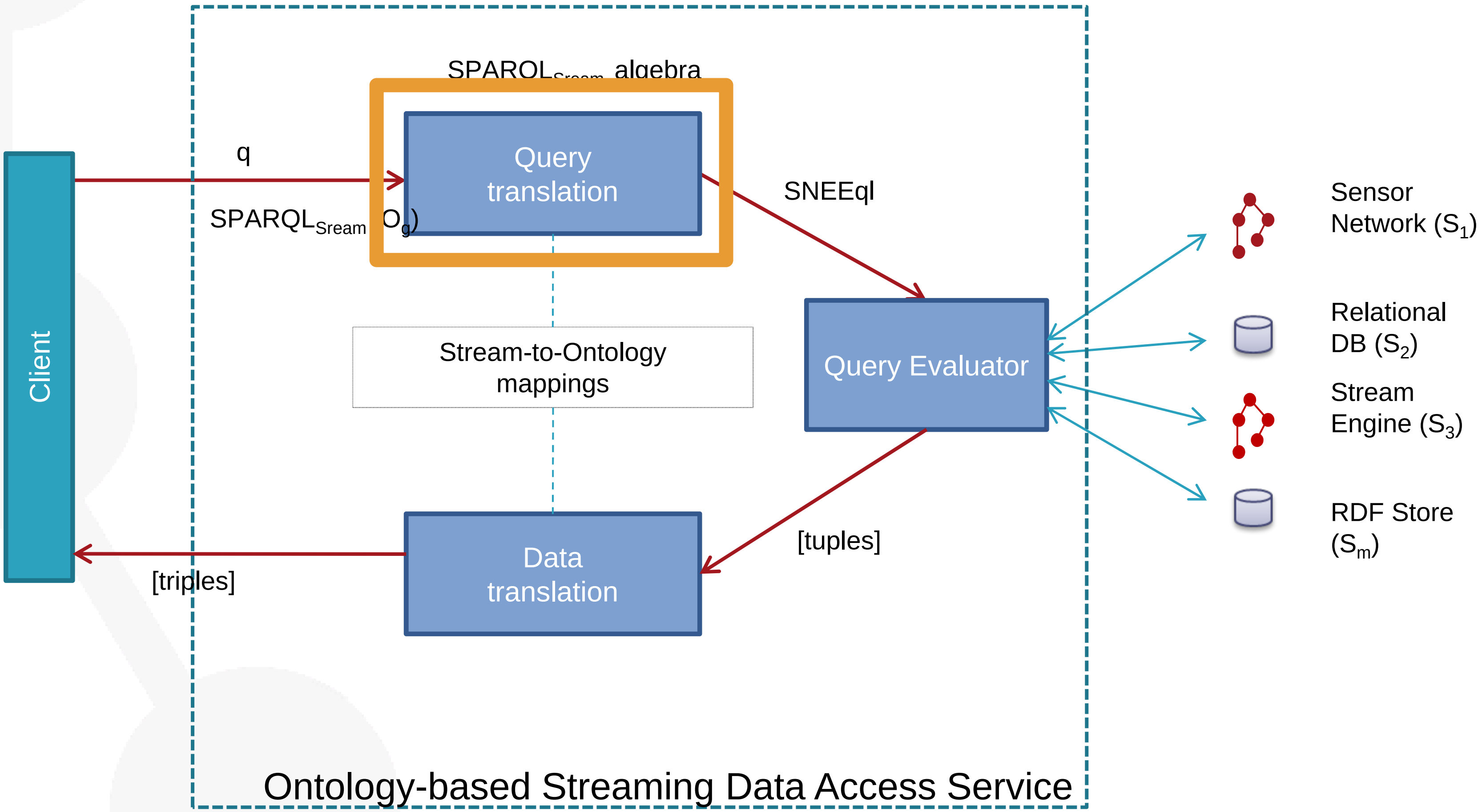
envdata_hornsea
Datetime: long
Hs : float
Lon: float
Lat: float

U

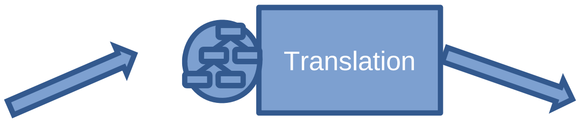
envdata_westbay
Datetime: long
Hs : float
Lon: float
Lat: float

```
<union name="meteostream">
   <map value="http://www.semsorgrid4env.eu/ontologies/
            CoastalDefences.owl#SolentCCO">
      <extent name="envdata_hornsea"></extent>
      <extent name="envdata_rhylflats"></extent>
      <extent name="envdata_milford"></extent>
      <extent name="envdata_westbay"></extent>
    </map>
</union>
```

Ontology-based Streaming Data Access



Stored Data Query Translation



```
SELECT ?y
FROM STREAM < STREAM
<http://www.semsorgrid4env.eu/ccometeo.srdf>
[ NOW – 1 MINUTE TO NOW – 0 MINUTES ] >
WHERE
{ ?x a cd:Observation;
    cd:observationResult ?y. }
```

```
SELECT   concat( ‘ssg4env.eu#ObservationWind' ,
                 envdata_milford.datetime ) as  x,
FROM     envdata_milford [ FROM NOW - 1 TO NOW MIN]
```

```
SELECT   concat( ‘ssg4env.eu#ObservationWind' ,
                 envdata_milford.datetime ) as  x,
         envdata_milford.Hs as y
FROM     envdata_milford [ FROM NOW - 1 TO NOW MIN]
```

Redundant queries, unacceptable for stream processing

Query Translation

- Queries:

```
SELECT ?y
WHERE
{ ?x a cd:Observation;
    cd:observationResult ?y. }
```

⇔

$$q(y) \leftarrow Observation(x) \wedge observationResult(x, y)$$

$$q(\vec{x}) \leftarrow \phi(\vec{x}, \vec{y}))$$

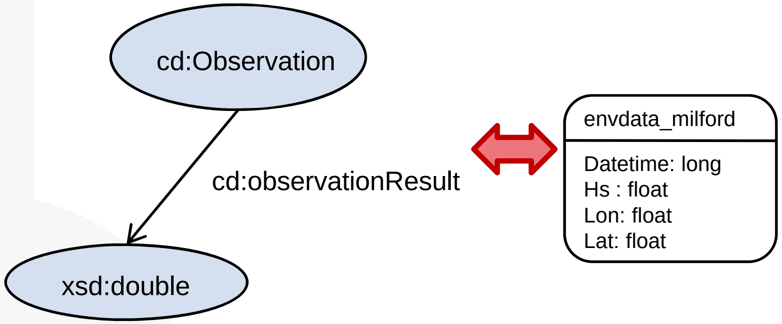
```
SELECT ?y
FROM STREAM < STREAM <http://www.semsorgrid4env.eu/ccometeo.srdf>
[ NOW – 1 MINUTE TO NOW – 0 MINUTES ] >
WHERE
{ ?x a cd:Observation;
    cd:observationResult ?y. }
```

$$q(y)[t_i, t_f, \delta] \leftarrow (Observation(x) \wedge observationResult(x, y))[t_i, t_f, \delta]$$

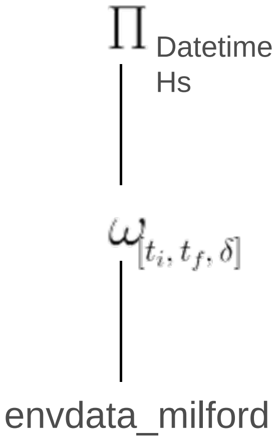
Query Translation

- Mappings:

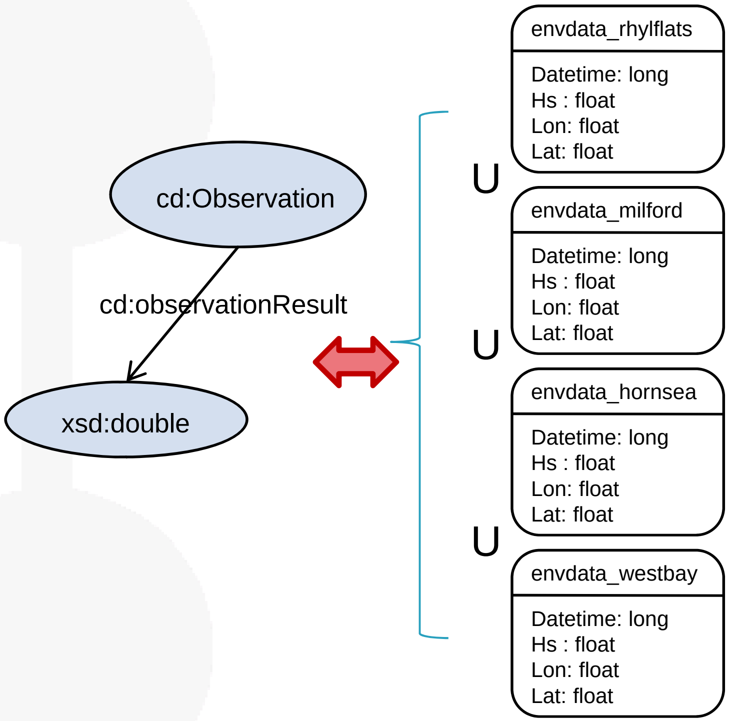
query → $\Psi \rightsquigarrow \Phi$ ← algebra expression over sources



$$(Observation(x) \wedge observationResult(x,y))[t_i,t_f,\delta] \rightsquigarrow$$



Query Translation



cd:Observation

cd:observationResult

xsd:double

envdata_rhylflats

Datetime: long
Hs : float
Lon: float
Lat: float

U

envdata_milford

Datetime: long
Hs : float
Lon: float
Lat: float

U

envdata_hornsea

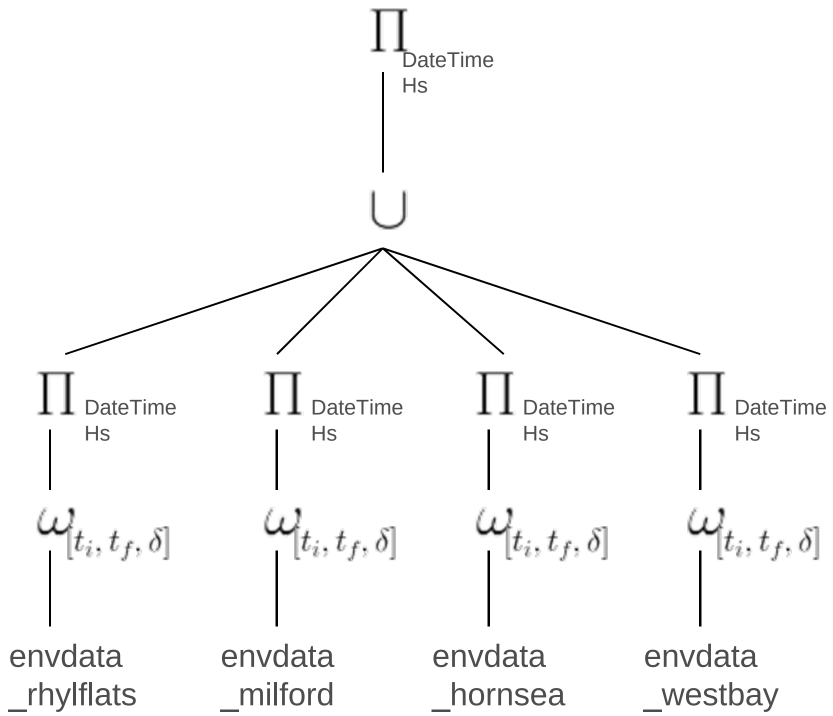
Datetime: long
Hs : float
Lon: float
Lat: float

U

envdata_westbay

Datetime: long
Hs : float
Lon: float
Lat: float

$$(Observation(x) \wedge observationResult(x,y))[t_i, t_f, \delta] \rightsquigarrow$$



$\Pi_{DateTime, Hs}$

$\cup$

$\Pi_{DateTime, Hs}$ $\Pi_{DateTime, Hs}$ $\Pi_{DateTime, Hs}$ $\Pi_{DateTime, Hs}$

$\omega_{[t_i, t_f, \delta]}$ $\omega_{[t_i, t_f, \delta]}$ $\omega_{[t_i, t_f, \delta]}$ $\omega_{[t_i, t_f, \delta]}$

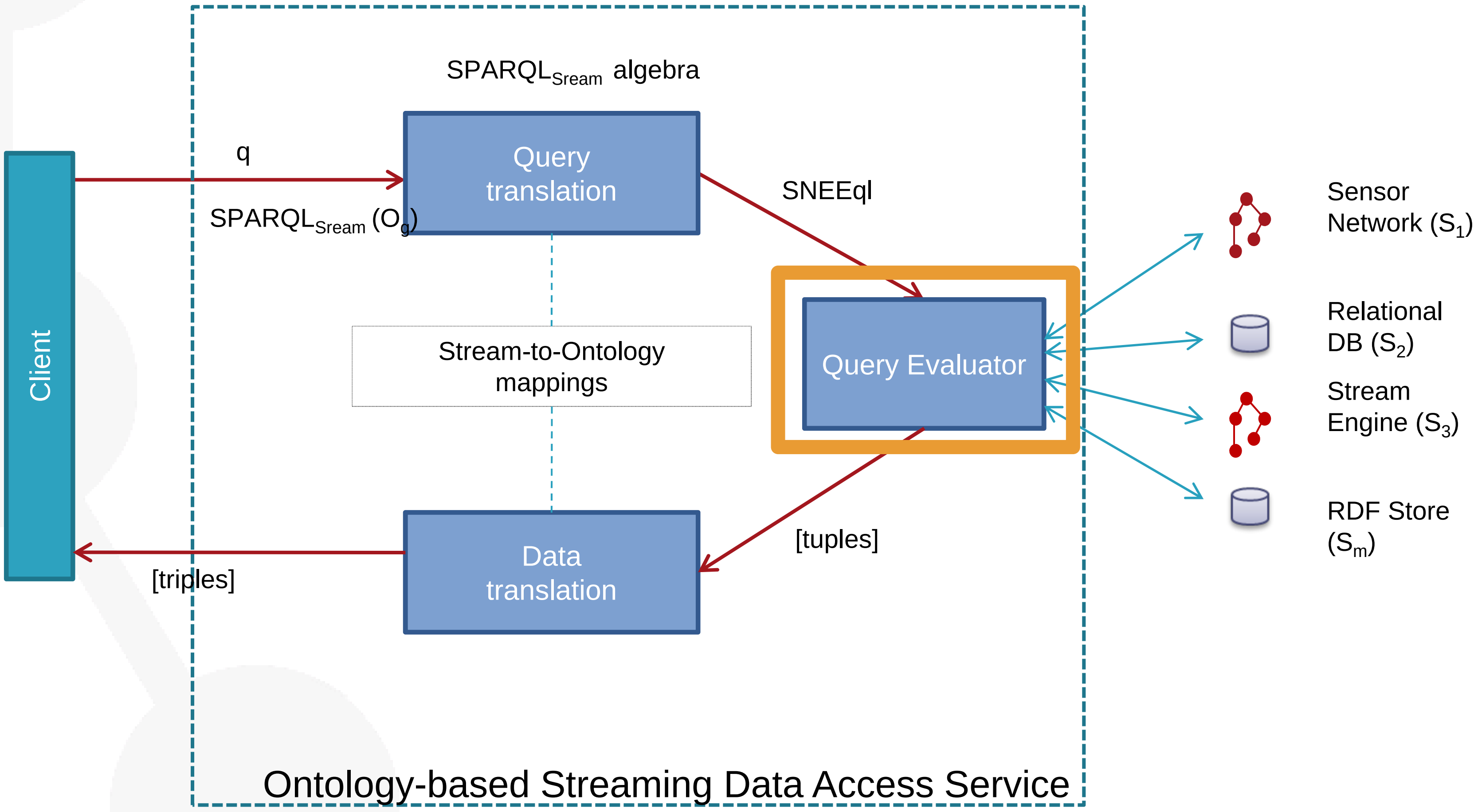
envdata_rhylflats

envdata_milford

envdata_hornsea

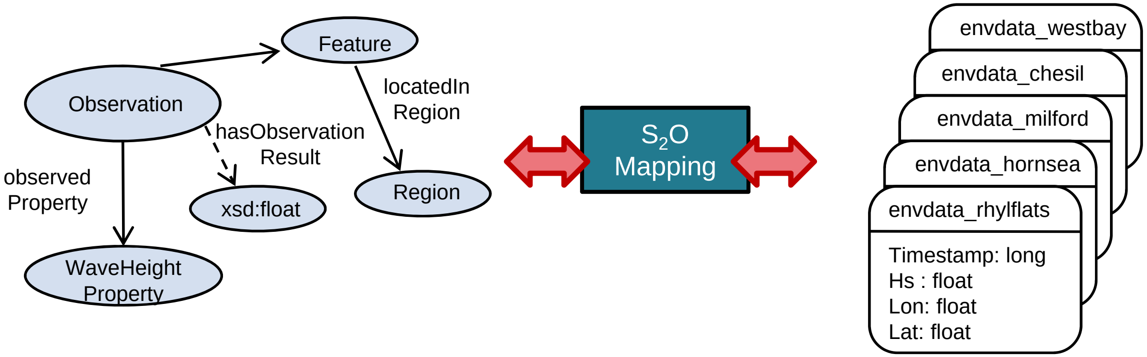
envdata_westbay

Ontology-based Streaming Data Access



Query Execution

Ontologies



Streams

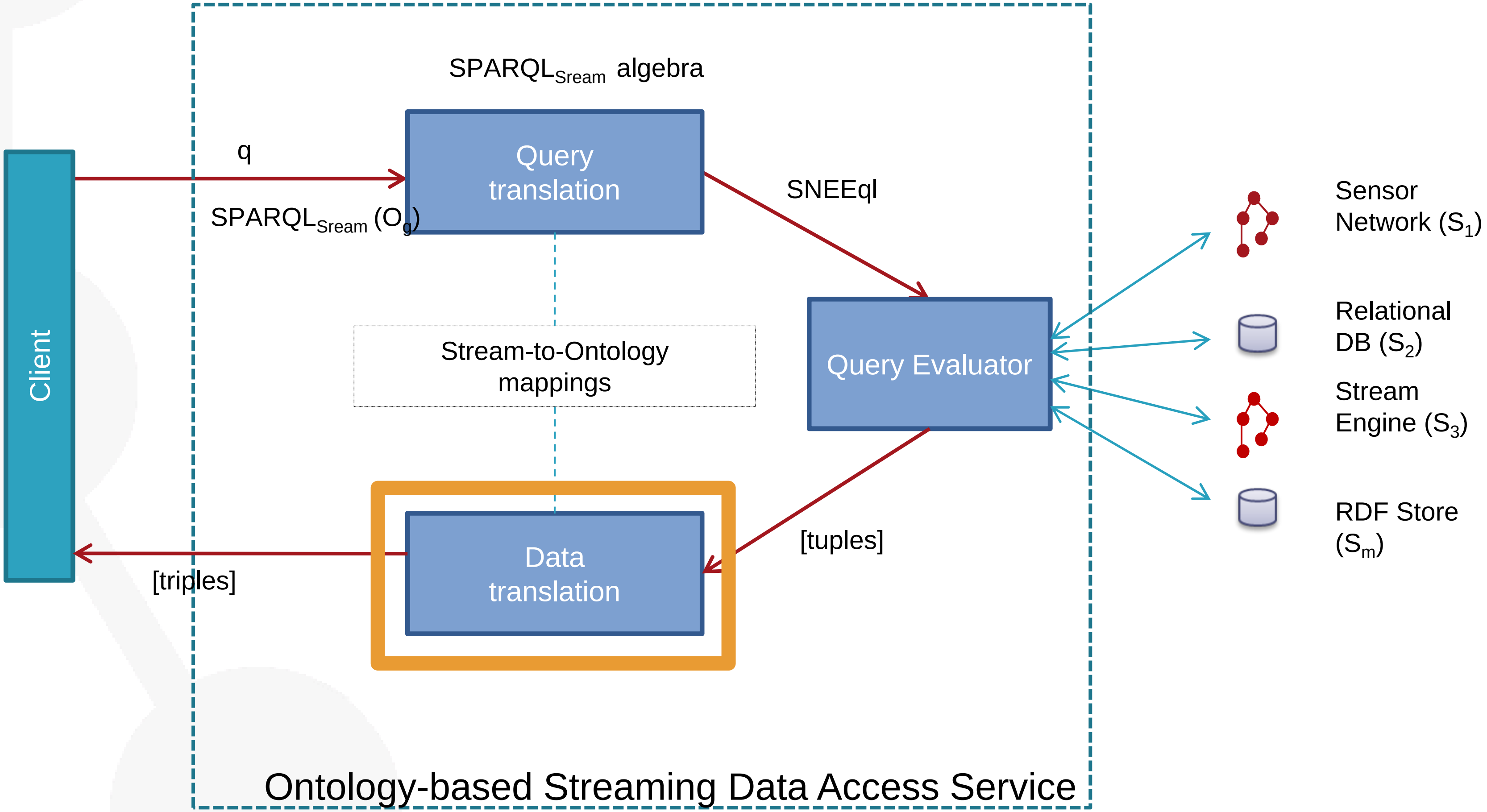
```
PREFIX cd: <http://www.semsorgrid4env.eu/ontologies/CoastalDefences.owl#>
PREFIX sb: <http://www.w3.org/2009/SSN-XG/Ontologies/SensorBasis.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT  ?waveheight ?wavets  ?lat ?lon
FROM STREAM <http://www.semsorgrid4env/ccometeo.srdf>
WHERE
{
 ?WaveObs a cd:Observation;
   cd:observationResult ?waveheight;
   cd:observationResultTime ?wavets;
   cd:observationResultLatitude ?lat;
   cd:observationResultLongitude ?lon;
   cd:observedProperty ?waveProperty;
   cd:featureOfInterest ?waveFeature.
 ?waveFeature a cd:Feature;
   cd:locatedInRegion cd:SouthEastEnglandCCO.
 ?waveProperty a cd:WaveHeight.
 }
```

SPARQL$_{Stream}$

```
(SELECT Lon,timestamp,Hs,Lat FROM envdata_rhylflats) UNION
(SELECT Lon,timestamp,Hs,Lat FROM envdata_hornsea) UNION
(SELECT Lon,timestamp,Hs,Lat FROM envdata_milford) UNION
(SELECT Lon,timestamp,Hs,Lat FROM envdata_chesil) UNION
(SELECT Lon,timestamp,Hs,Lat FROM envdata_perranporth) UNION
(SELECT Lon,timestamp,Hs,Lat FROM envdata_westbay) UNION
(SELECT Lon,timestamp,Hs,Lat FROM envdata_pevenseybay)
```

SNEEql

Ontology-based Streaming Data Access



Data Translation

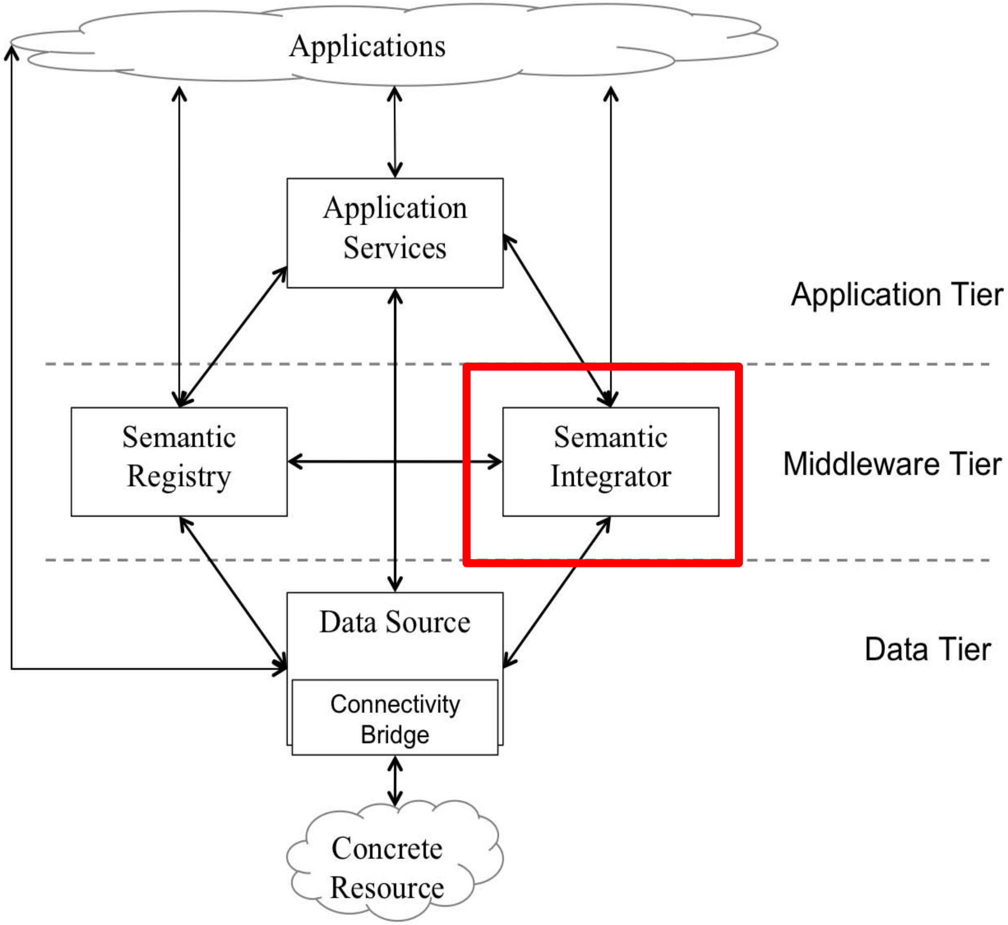
- Data translation:

  Tagged tuples → SPARQL bound variables

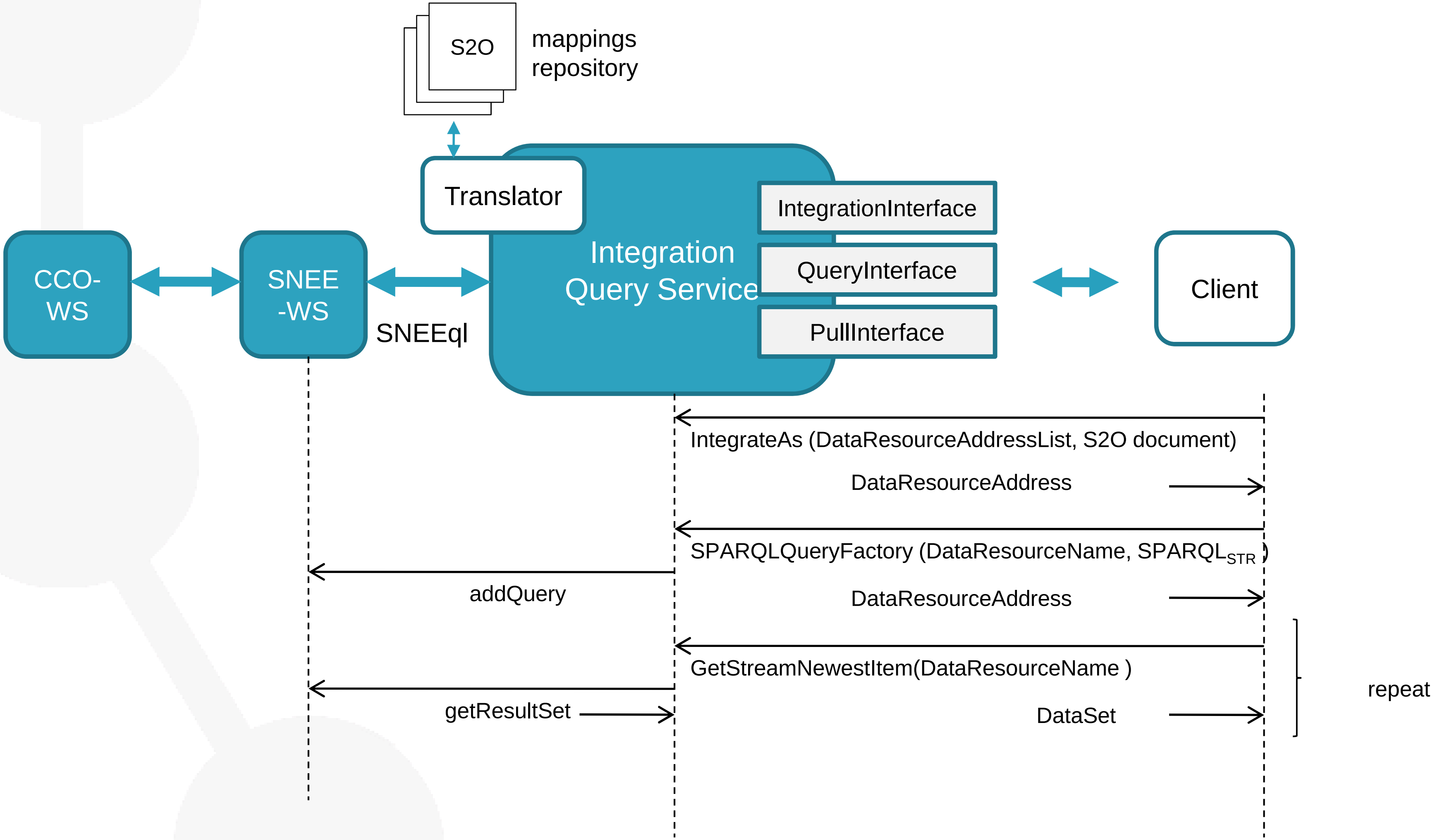
```
<ns9:sparql>
 <ns9:head>
   <ns9:variable name="waveheight"/>
   <ns9:variable name="wavets"/>
 </ns9:head>
 <ns9:results>
   <ns9:result>
     <ns9:binding name="waveheight">
        <ns9:literal datatype="http://www.w3.org/2001/XMLSchema#double">4.850</ns9:literal>
     </ns9:binding>
     <ns9:binding name="wavets">
        <ns9:literal datatype="http://www.w3.org/2001/XMLSchema#long">1272588663</ns9:literal>
     </ns9:binding>
    </ns9:result>
    <ns9:result>
     <ns9:binding name="waveheight">
       <ns9:literal datatype="http://www.w3.org/2001/XMLSchema#double">2.1230</ns9:literal>
     </ns9:binding>
     <ns9:binding name="wavets">
       <ns9:literal datatype="http://www.w3.org/2001/XMLSchema#long">1272587400</ns9:literal>
     </ns9:binding>
    </ns9:result>
</ns9:results>
</ns9:sparql>
```

Implementation

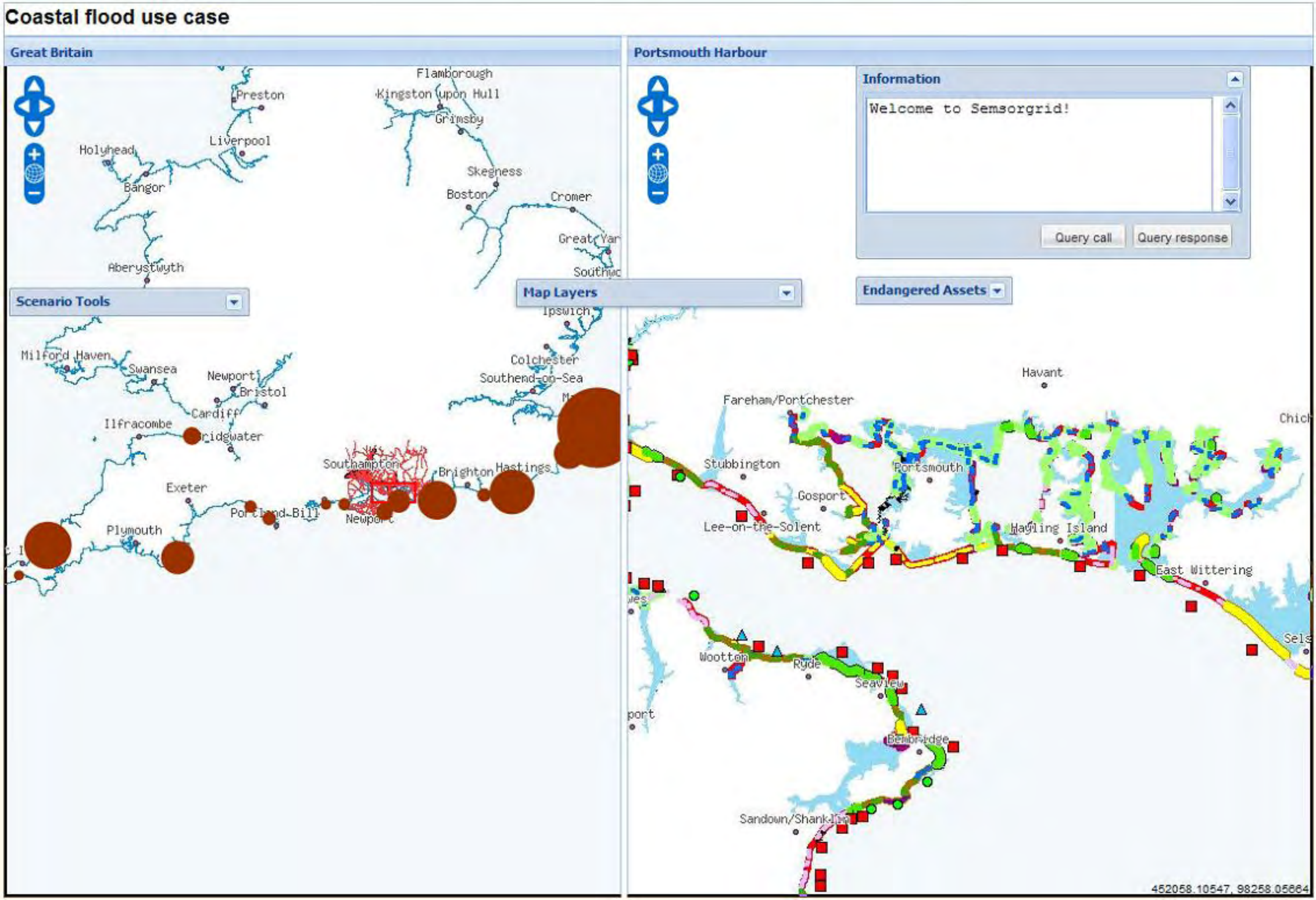
- Design, implement and deploy a *Semantic Integration Service*
- Extend existing ontology-based data integration models to take into account sensor networks streaming data, semantic heterogeneity and quality of service



Implementation



- Flood Warning Application


Coastal flood use case

Welcome to Semsorgrid!

So Far…

Ontology-base data access

- Define stream extensions for R2O
- Define SPARQL$_{\text{Stream}}$ language syntax and semantics
- Define S2O mappings-based translation semantics

Implementation

- Enable engine support for « S2O » documents, SPARQL$_{\text{Stream}}$ queries
- Enabled engine support for SNEEql translation and connection
- Limited to non-distributed scenario initially

Future Works

- Ontology-based data access
  - SPARQL construct expressions, aggregates, projected operators
  - Implement adapters for other streaming sources
  - Add query rewriting algorithms
- Ontology-based streaming data integration
  - Horizontal & vertical integration
  - Integrate streaming + stored data
  - RDF data sources integration
- Streaming query optimization
  - Analyze cost models
  - Streaming sources statistics and metadata
- Quantitative evaluation

Thanks!

Enabling Ontology-based Access to Streaming Data Sources

Internship

- **Period**: 21.05.2010 - 29.07.2010
- **Institution**: School of Computer Science, The University of Manchester

- **Research contact**: Alasdair Gray
- **Professors**:

  Norman Paton,

  Alvaro Fernandes

Objectives

- Study the existing approaches on distributed SPARQL query processing.
- Study the existing SPARQL query optimization approaches.
- Study the existing streaming SPARQL query processing approaches.
- Propose a coherent solution for streaming distributed query processing using SPARQL extensions.
- Propose a study use-case in the context of the SSG4E project that highlights the streaming query processing capabilities.
- Prepare a formal publishable version of the proposal for the consideration of the community

Research Outcome

- Stuff for each of the objectives available in draft documents:
  - http://delicias.dia.fi.upm.es/wiki/index.php/ManchesterJPC
- Material in development for future interesting work

Integration Use-cases

- **Horizontal integration** of data sources. Sources provide the same type of information with respect to the global schema. The data may overlap (redundancy) or be complementary (partitioned) and it can be seen as union of homogeneous information.

- **Vertical integration** of sources. Sources provide different information with respect to the global schema. In this case data of each source adds additional information to an original set. In this case the joins are between different kinds of information and finding their relationships is not always straightforward.

- **Integration of different streaming sources**. Answer queries including data from 2+ streaming sources.

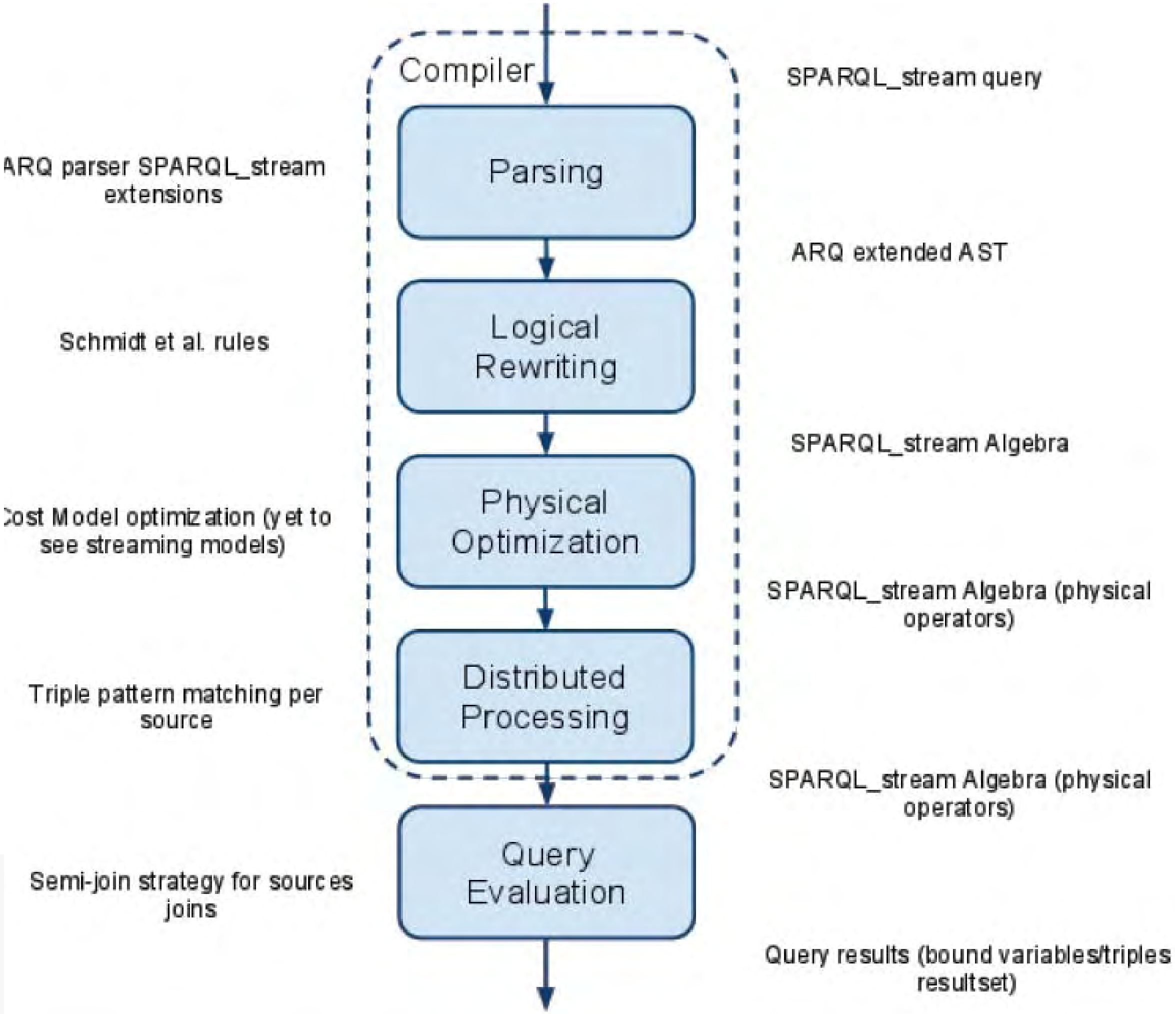
- **Integration of streaming and stored data**.

- **Integration of streaming data in the past** and current streaming data

Completing SPARQL$_{\text{Stream}}$

- Adjust semantics of « triple-based » windows
- Fixes on types: stream of triples, stream of triple windows.
- Window-to-Stream operation semantics
- Stored data scans

Query Optimization



Compiler

SPARQL_stream query

ARQ parser SPARQL_stream extensions

Parsing

ARQ extended AST

Schmidt et al. rules

Logical Rewriting

SPARQL_stream Algebra

Cost Model optimization (yet to see streaming models)

Physical Optimization

SPARQL_stream Algebra (physical operators)

Triple pattern matching per source

Distributed Processing

SPARQL_stream Algebra (physical operators)

Semi-join strategy for sources joins

Query Evaluation

Query results (bound variables/triples resultset)

Open for Collaboration

- Any streaming data in your project? e.g. geo information?
- Mapping-based query translation
- Distributed query processing
- Discovery/invocation of data sources
- Others…

Thanks