Stream Reasoning For Linked Data

M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, E. Della Valle, and J.Z. Pan

http://streamreasoning.org/sr4ld2013

C-SPARQL: A Continuous Extension of SPARQL

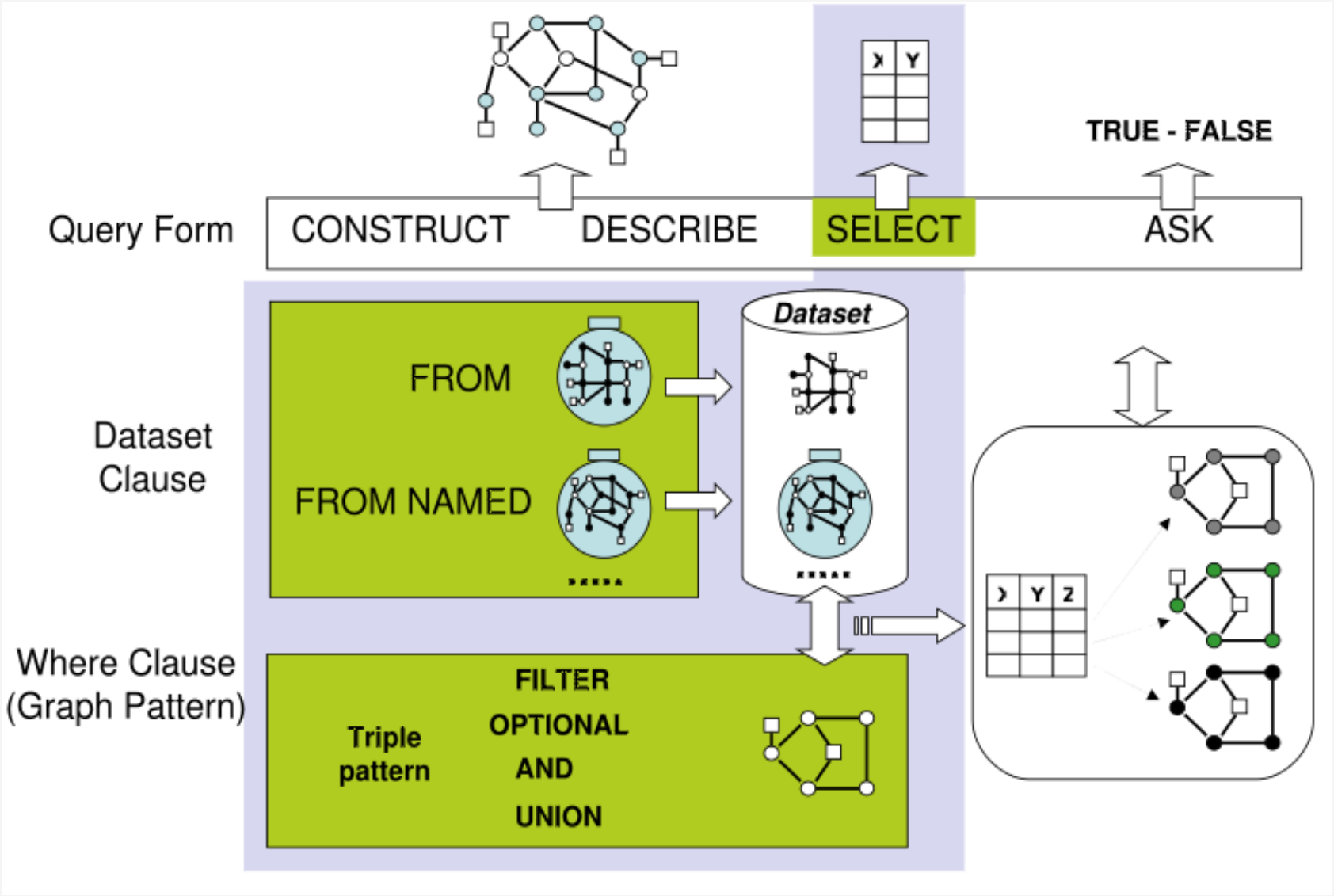
Marco Balduini
marco.balduini@polimi.it

Share, Remix, Reuse — Legally

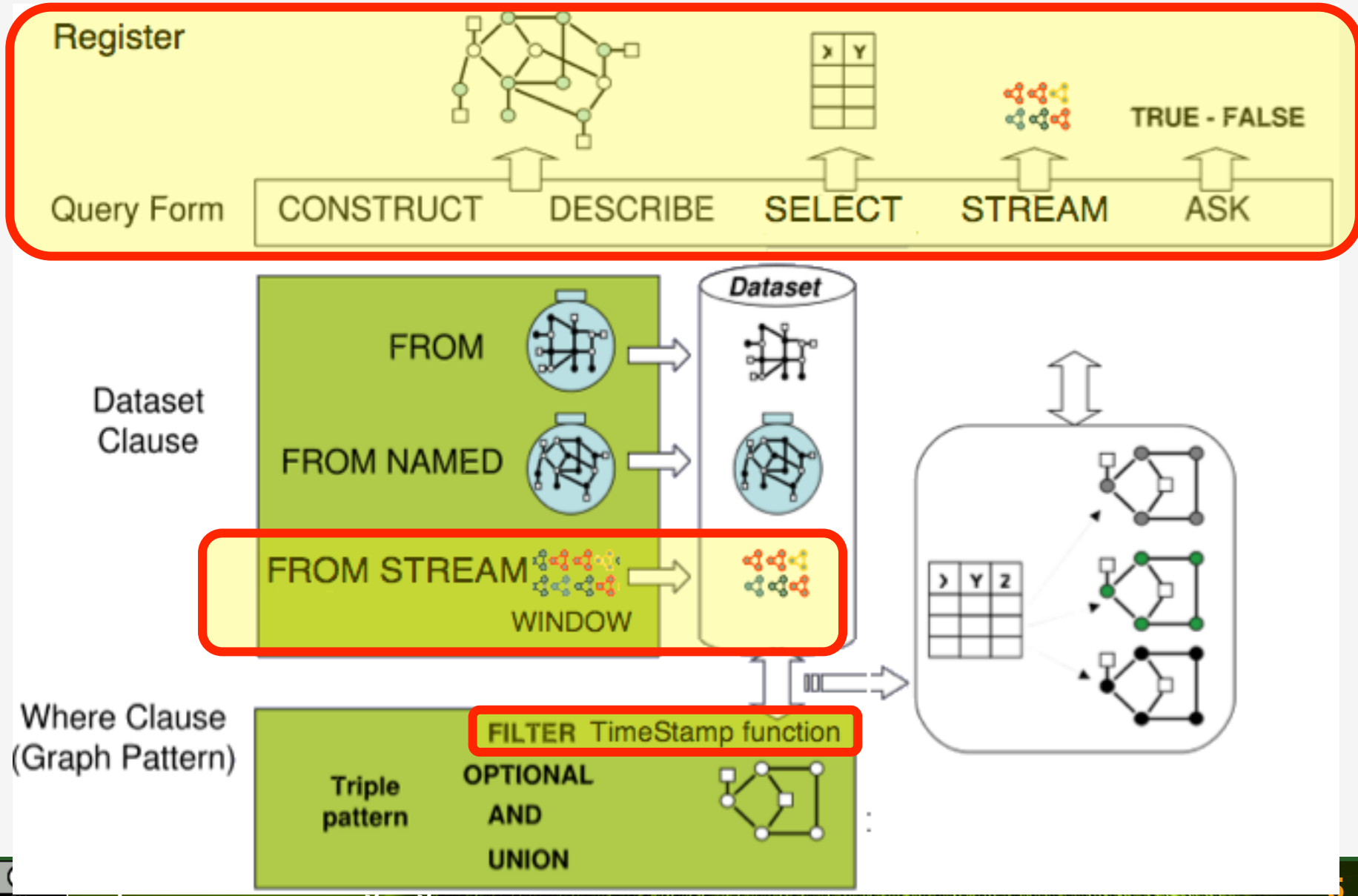
Agenda

- Introduction
- Running example
- C-SPARQL language
  - Query and Stream Registration
  - FROM STREAM Clause
  - TimeStamp Function
  - Query Chaining
  - Accessing background Information
  - Q/A under RDFS entailment regime
  - Q/A under RDFS++ entailment regime
- C-SPARQL Engine
- Resources

A Recall of SPARQL



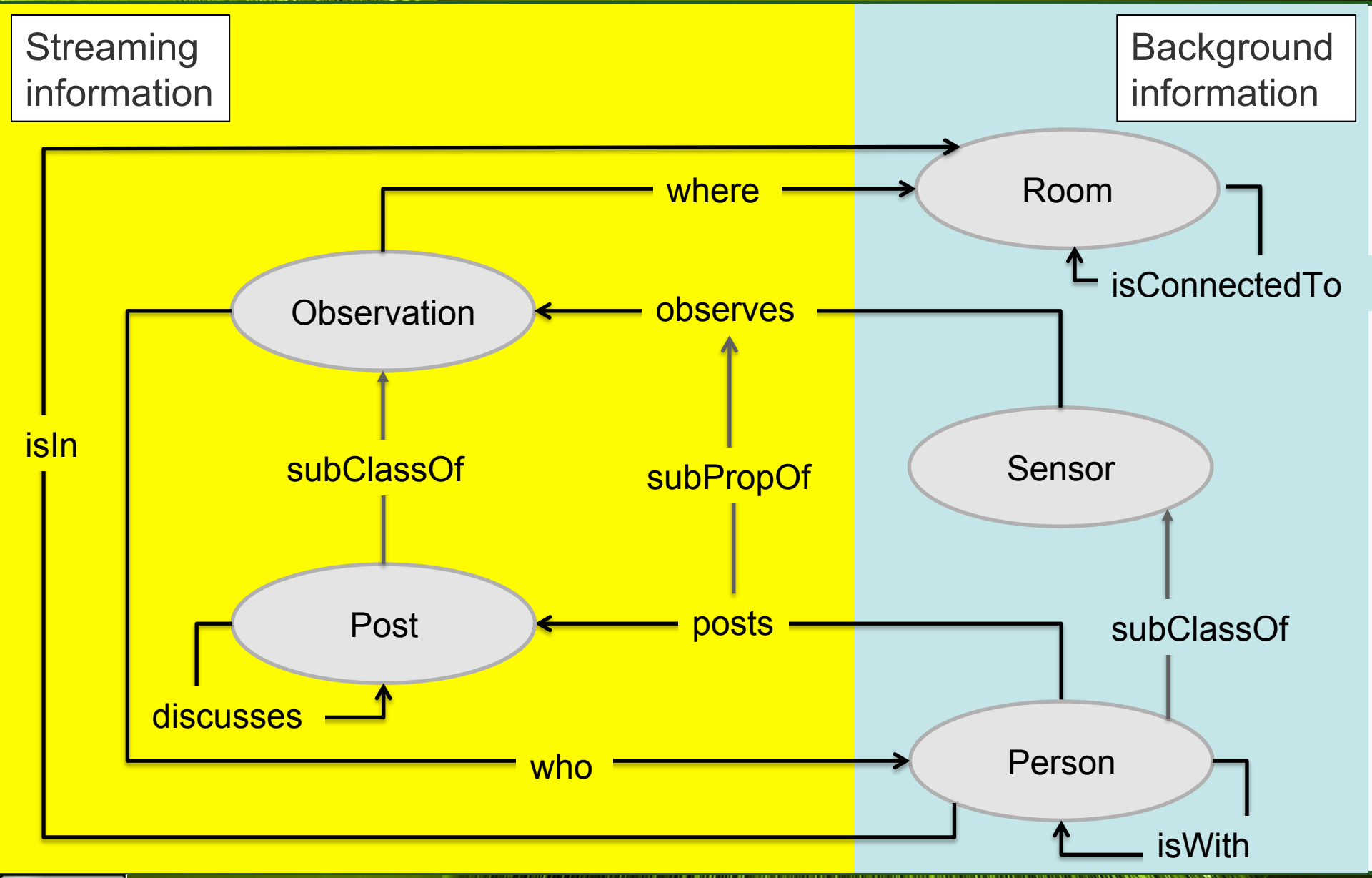
Where C-SPARQL Extends SPARQL



C-SPARQL language

- C-SPARQL is an extension of SPARQL 1.1
- C-SPARQL queries
  - Adds to SPARQL 1.1 query forms the STREAM form
  - Changes the semantics of SPARQL 1.1 query forms from the one-time semantics to the continuous one (i.e., instantaneous) and
  - Adds to SPARQL 1.1 datasets clauses the FROM STREAM one
  - Adds a built-in function to access the timestamp of a triple

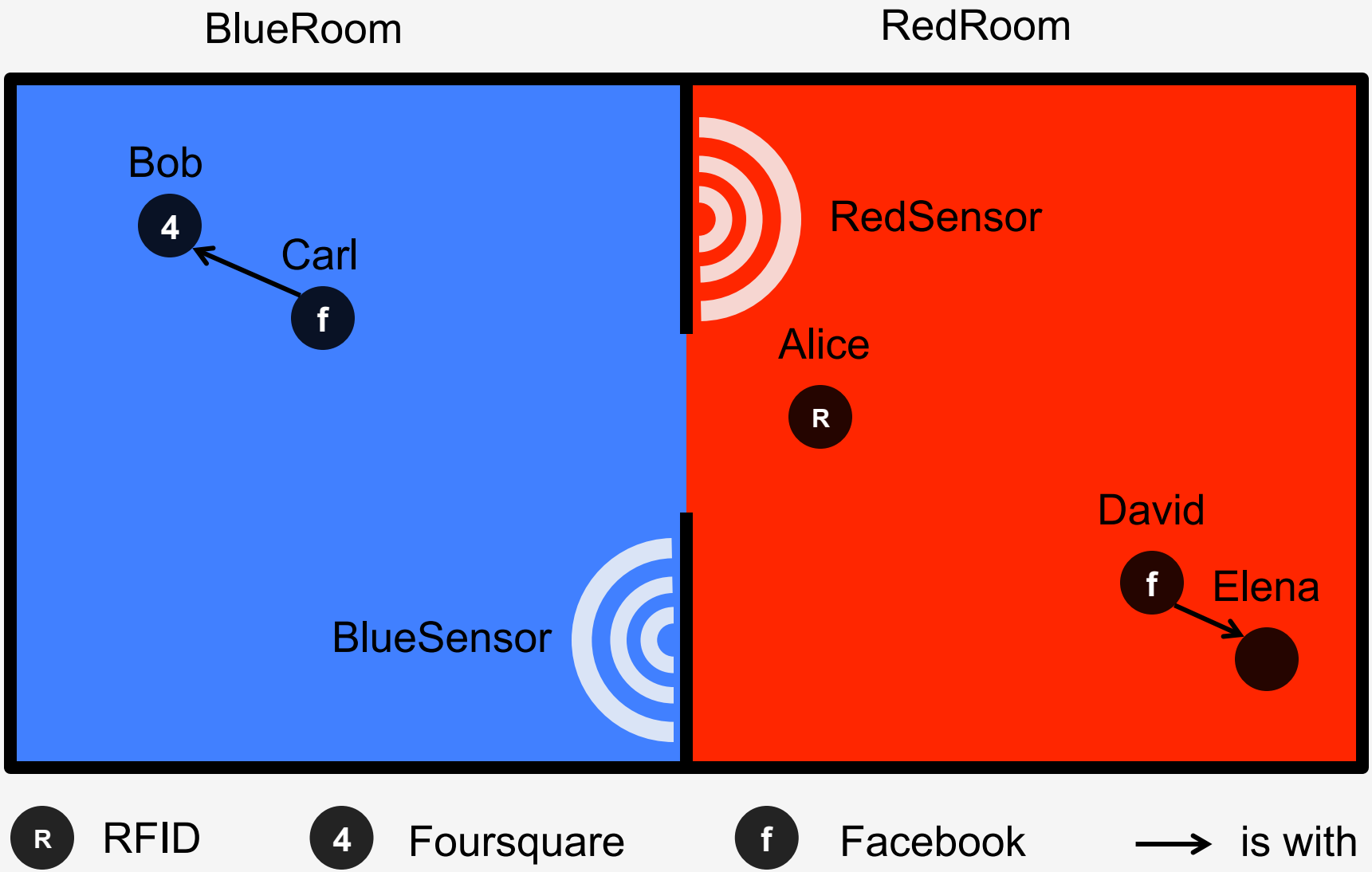
Running Example – Data Model



Background Information

- The Ontology
  - http://www.streamreasoning.org/ontologies/sr4ld2013-onto.rdf
- The Instances
  - :Alice a :Person .
  - :Bob a :Person .
  - :Carl a :Person .
  - :David a :Person .
  - :Elen a :Person .
  - :RedRoom a :Room .
  - :BlueRoom a :Room .
  - :RedRoom :isConnectedTo :BlueRoom .
  - :RedSensor a :Sensor .
  - :BlueSensor a :Sensor .

Running Example



Example data in the streams

- Four ways to learn who is where

Sensor	Room	Person	Time-stamp
RedSensor	RedRoom	Alice	T_1
…	…	…	…

Person	ChecksIn	Time-stamp
Bob	BlueRoom	T_2
…	…	…

Person	IsIn	With	Time-stamp
Carl	null	Bob	T_2
David	RedRoom	Elena	T_3
…	…	…	…

Streaming Information

- RDF Stream Data Type
  - Ordered sequence of pairs, where each pair is made of an RDF triple and its timestamp

$$\ldots$$
$$(\langle subj_i, pred_i, obj_i \rangle , \tau_i)$$
$$(\langle subj_{i+1}, pred_{i+1}, obj_{i+1} \rangle , \tau_{i+1})$$
$$\ldots$$

    - Timestamps are not required to be unique, they must be non-decreasing

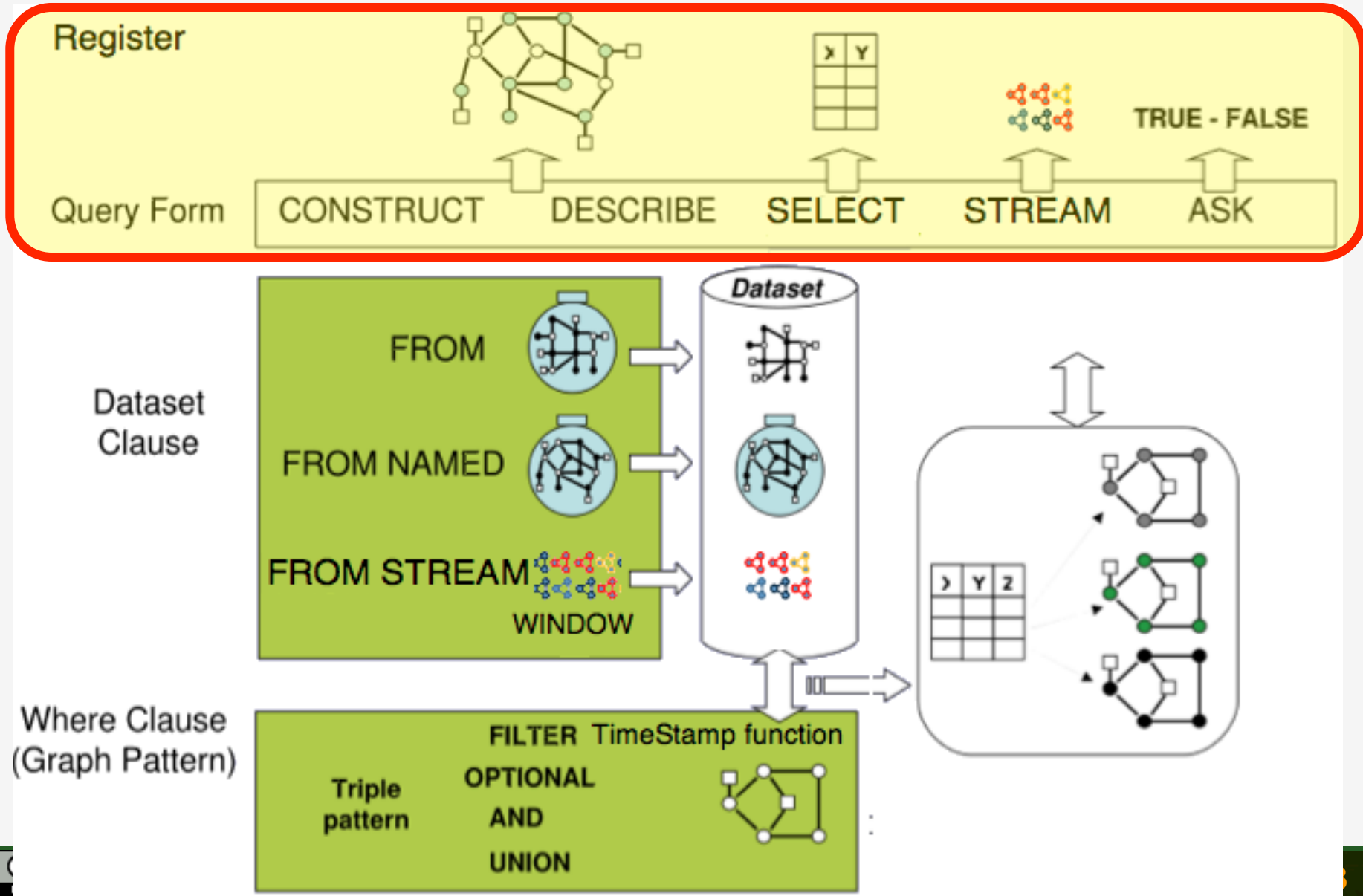
- E.g.,

RDF graph	Time-stamp	Stream
:RedSensor :observes [:who :Alice; :where :RedRoom] .	T_1	rfid
:Bob :posts [:who :Bob ; :where :BlueRoom] .	T_2	fs
:Carl :posts [:who :Carl , :Bob] .	T_2	fb
:David :posts [:who :David , :Elena ; :where :RedRoom]	T_3	fb

C-SPARQL language

- Features illustrated in the rest of this session
  - register continuous queries
    - QUERY form
    - STREAM form
  - identify relevant information in one or more RDF streams
  - derive and aggregate information from one or more RDF streams
  - join or merge RDF streams
  - feed results of one C-SPARQL query to a subsequent C-SPARQL query
  - access background RDF graphs
  - answer under RDFS entailment regime
  - answer under RDFS++ entailment regime

Query and Stream Registration



Query and Stream Registration

- All C-SPARQL queries over RDF streams are continuous
  - Registered through the REGISTER statement
- The output of queries is in the form of
  - Instantaneous tables of variable bindings
  - Instantaneous RDF graphs
  - RDF stream
- Only queries in the CONSTRUCT form can be registered as generators of RDF streams

$$\begin{aligned} \textit{Registration} \rightarrow\ & \text{`REGISTER'}\ (\text{`QUERY'} |\ \text{`STREAM'})\ \textit{QueryName} \\ & [\text{`COMPUTED EVERY'}\ \textit{Number}\ \textit{TimeUnit}]\ \text{`AS'}\ \textit{Query} \end{aligned}$$

- Composability:
  - Query results registered as streams can feed other registered queries just like every other RDF stream

Query registration - Example

- Using the social stream fb, Who is where?

```
REGISTER QUERY QWhoIsWhereOnFb AS
PREFIX : <http://…/sr4ld2013-onto#>
SELECT ?room ?person
FROM STREAM <http://…/fb> [RANGE 1m STEP 10s]
WHERE {
  ?person1 :posts [ :who ?person ; :where ?room ] .
}
```

- The resulting variable bindings has to be interpreted as an instantaneous. It expires as soon as the query is recomputed

Stream registration - Example

- Results of a C-SPARQL query can be stream out for down stream queries

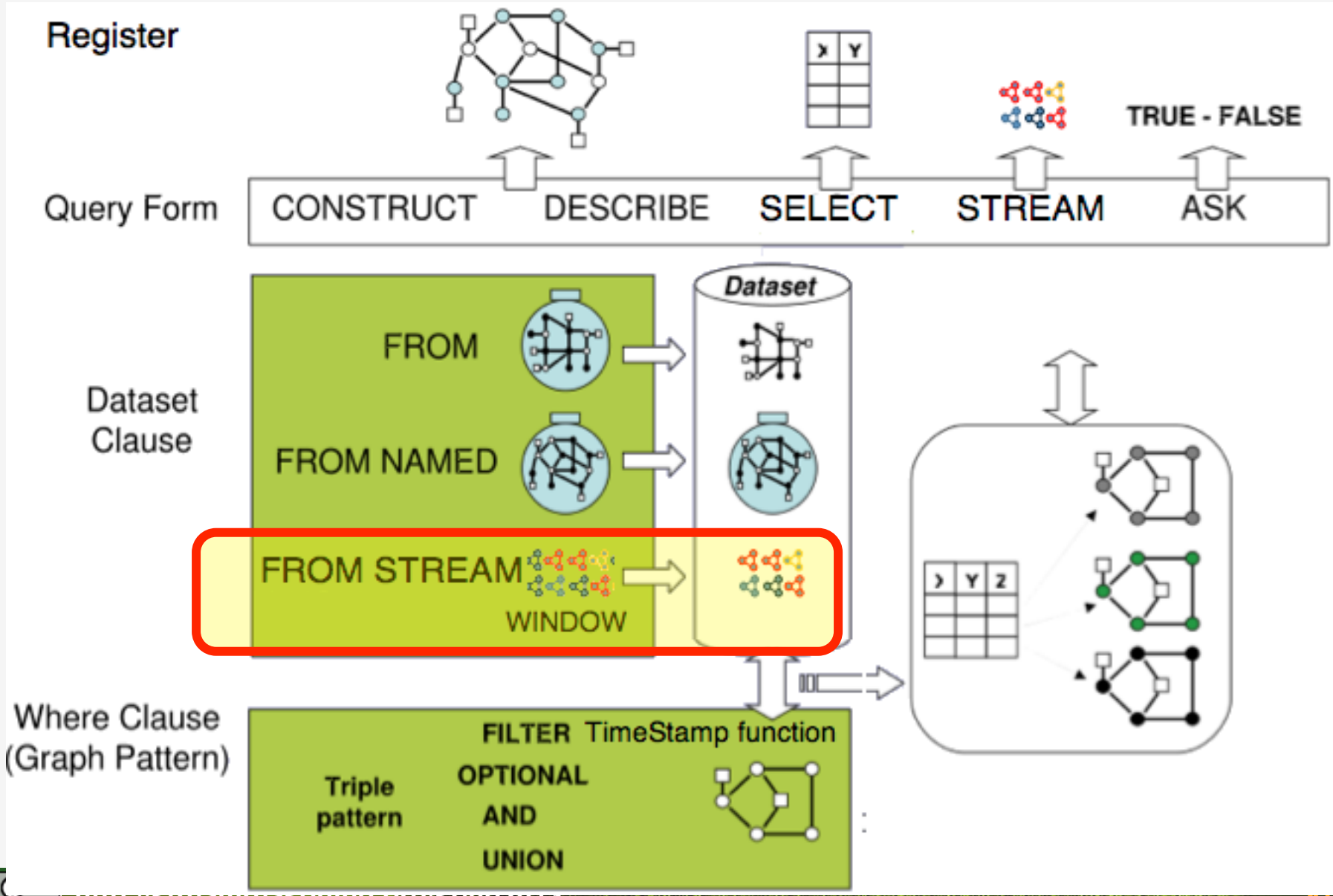
```
REGISTER STREAM SWhoIsWhereOnFb AS
PREFIX : <http://…/sr4ld2013-onto#>
CONSTRUCT { ?person :isIn ?room }
FROM STREAM <http://…/fb> [RANGE 1m STEP 10s]
WHERE {
  ?person1 :posts [ :who ?person ; :where ?room ] .
}
```

- The resulting RDF triples are streamed out on an RDF stream
  - More details in the C-SPARQL Engine hands-on session

Stream Registration - Notes

- The output is constructed in the format of an RDF stream.
- Every query execution may produce from a minimum of zero triples to a maximum of an entire graph.
- The timestamp is always dependent on the query execution time only, and is not taken from the triples that match the patterns in the WHERE clause.

FROM STREAM Clause

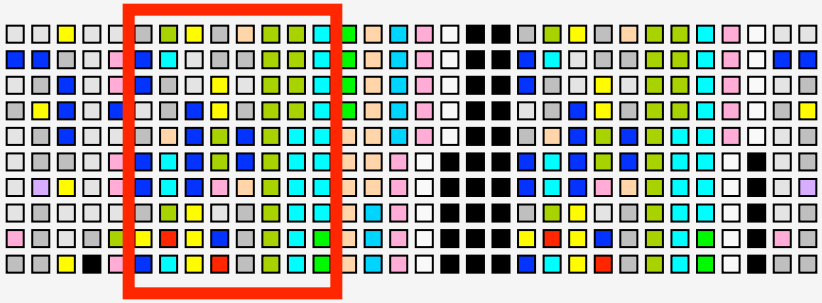


FROM STREAM Clause

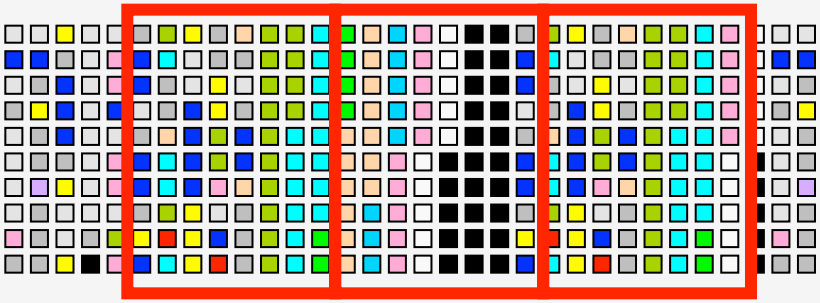
- FROM STREAM clauses are similar to SPARQL datasets
  - They identify RDF stream data sources
  - They represent windows over a RDF stream
- They define the RDF triples available for querying and filtering.

FROM STREAM Clause - windows

- physical: a given number of triples
- logical: a variable number of triples which occur during a given time interval (e.g., 1 hour)
  - Sliding: they are progressively advanced of a given STEP (e.g., 5 minutes)



  - Tumbling: they are advanced of exactly their time interval



FROM STREAM Clause - Windows

- Grammar of the FROM STREAM clause

$$
\begin{aligned}
\textit{FromStrClause} &\longrightarrow \text{`FROM'}\ [\text{`NAMED'}]\ \text{`STREAM'}\ \textit{StreamIRI} \\
&\quad\ \text{`[ RANGE'}\ \textit{Window}\ \text{`]'} \\
\textit{Window} &\longrightarrow \textit{LogicalWindow} \mid \textit{PhysicalWindow} \\
\textit{LogicalWindow} &\longrightarrow \textit{Number}\ \textit{TimeUnit}\ \textit{WindowOverlap} \\
\textit{TimeUnit} &\longrightarrow \text{`ms'} \mid \text{`s'} \mid \text{`m'} \mid \text{`h'} \mid \text{`d'} \\
\textit{WindowOverlap} &\longrightarrow \text{`STEP'}\ \textit{Number}\ \textit{TimeUnit} \mid \text{`TUMBLING'} \\
\textit{PhysicalWindow} &\longrightarrow \text{`TRIPLES'}\ \textit{Number}
\end{aligned}
$$

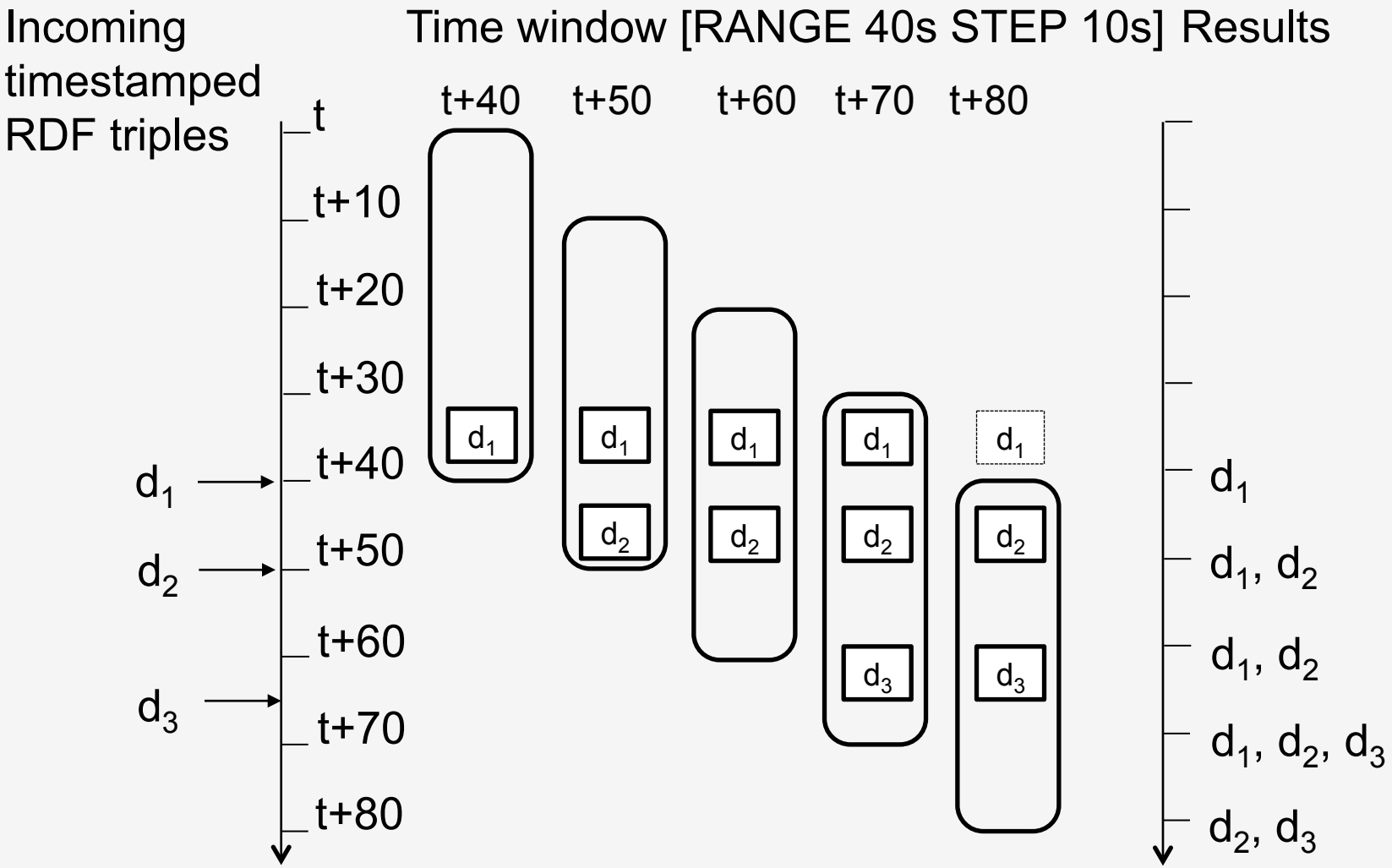
- The optional NAMED keyword works exactly like when applied to the standard SPARQL FROM clause for tracking the provenance of triples.
  - It binds the IRI of a stream to a variable which is later accessible through the GRAPH clause.

FROM STREAM Clause - Example

- Using the social stream fb, how many people are in the same room? Count on a window of 1 seconds that slides every 10 seconds

```
REGISTER QUERY HowManyPoepleAreInTheSameRoom AS
PREFIX : <http://…/sr4ld2013-onto#>
SELECT ?room (count(?s) as ?person)
FROM STREAM <http://…/fb> [RANGE 1m STEP 10s]
WHERE {
  ?person1 :posts [ :who ?person ; :where ?room ] .
}
GROUP BY ?room
```

C-SPARQL reports only snapshots

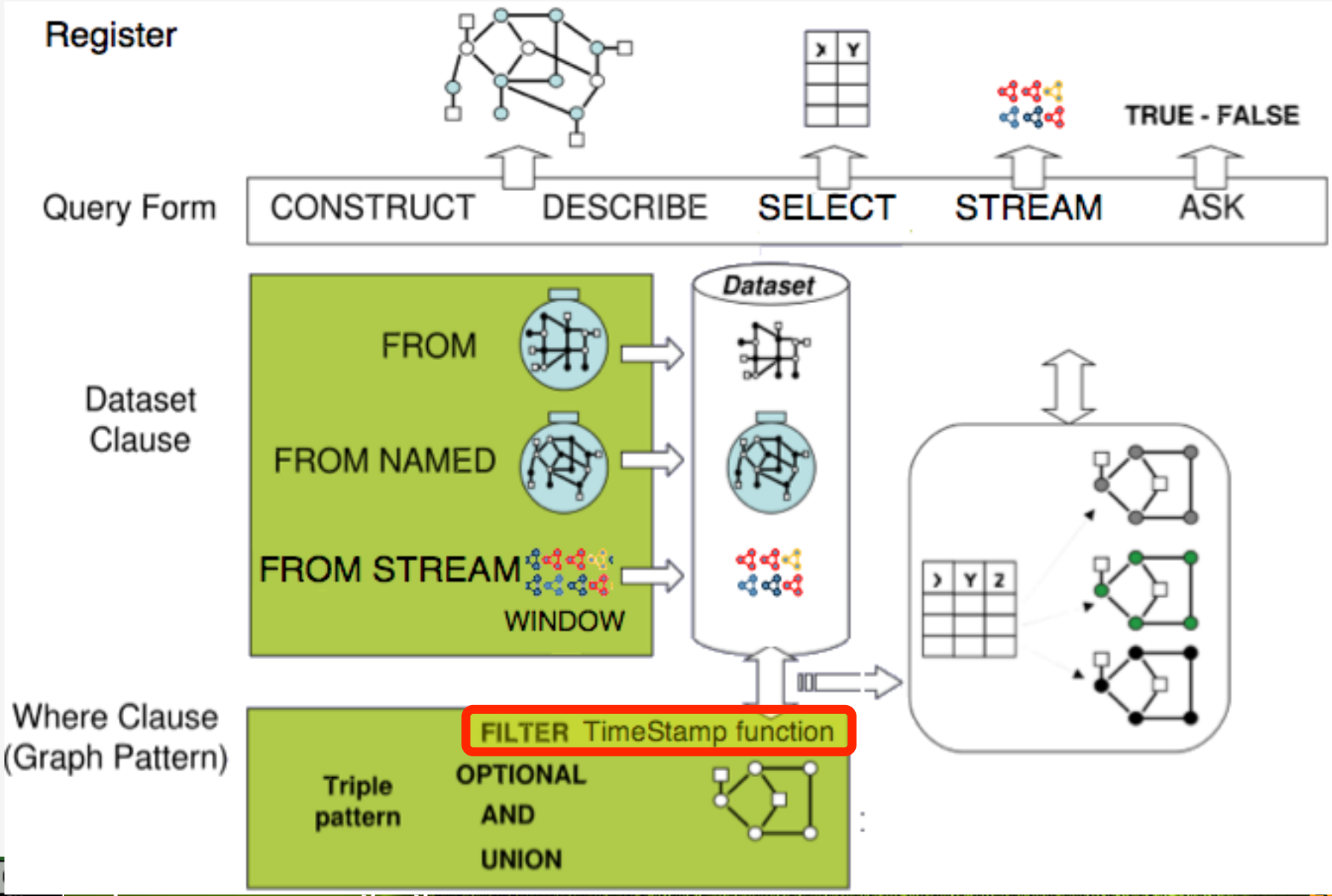


Multiple FROM STREAM Clause - Example

- **Using the social stream fb and fs**, how many people are in the same room? Count on a window of 1 minute that slides every 10 seconds

```
REGISTER QUERY HowManyPoepleAreInTheSameRoom AS
PREFIX : <http://…/sr4ld2013-onto#>
SELECT ?room (count(?s) as ?person)
FROM STREAM <http://…/fb> [RANGE 1m STEP 10s]
FROM STREAM <http://…/fs> [RANGE 1m STEP 10s]
WHERE {
  ?person1 :posts [ :who ?person ; :where ?room ] .
}
GROUP BY ?room
```

TimeStamp Function



TimeStamp Function – Syntax and Semantics

- The timestamp of a triple can be bound to a variable using a timestamp() function
- Syntax
  - timestamp(variable|IRI|bn, variable|IRI, variable|IRI|bn|literal)
- Semantics

Triple	Result of evalutaion
It is not in the window	Type Error
It appears once in the window	Timestamp of triple
It appears multiple times in the window	The timestamp of the most recent triple

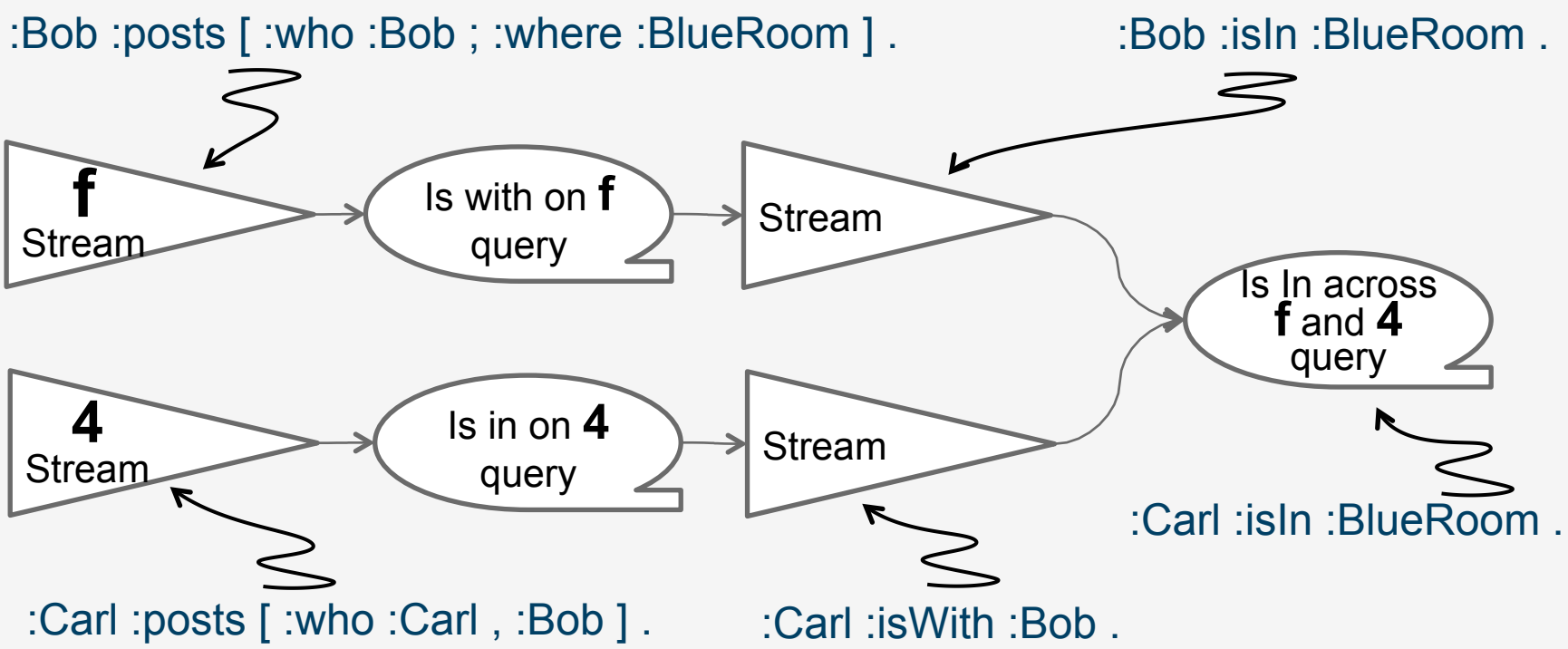
TimeStamp Function - Example

- Who are the opinion makers? i.e., the users who are likely to influence the behavior of other users

```
REGISTER QUERY FindOpinionMakers AS
PREFIX f: <http://larkc.eu/csparql/sparql/jena/ext#>
PREFIX : <http://…/sr4ld2013-onto#>
SELECT ?someOne ?room (COUNT(?someOneElse) AS ?n)
FROM STREAM <http://…/fb> [RANGE 1m STEP 10s]
WHERE {
 ?someOne :posts ?p1 .
 ?p1 :where ?room .
 ?someOneElse :posts ?p2 .
 ?p2 :where ?room .
FILTER(?someOne!=?someOneElse )
FILTER (f:timestamp(?p1 :where ?room) >
        f:timestamp(?p2 :where ?room))
}
GROUP BY ?someOne ?room
HAVING (?n>3)
```

Query Chaining

- A C-SPARQL query Q_1 registered using the STREAM clause streams results on an RDF stream
- A down stream C-SPARQL query Q_2 can open a window on the RDF stream of Q_1 using the FROM STREAM clause
- E.g.,



Accessing background Information

- C-SPARQL allows for asking the engine to issue the query also against RDF graphs using the FROM clauses.
- E.g., Where else can Alice go?

```
REGISTER QUERY WhereElseCanAliceGo AS
PREFIX : <http://…/sr4ld2013-onto#>
SELECT ?room
FROM STREAM <http://…/isIn> [RANGE 10m STEP 10m]
FROM <http://…/bgInfo.rdf>
WHERE {
  ?:Alice :isIn ?someRoom .
  ?someRoom :isConnectedTo ?room .
}
```

IRI identifying the dataset containing the background information

Alert in accessing background Information

- Accessing background information in DSMSs and CEPs is problematic and it can spoil performances
- The C-SPARQL Engine 0.9.1 allows for managing background information using a SPARQL update endpoint
- For a better solution see:
  - Le-Phuoc, D., Dao-Tran, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: International Semantic Web Conference (ISWC 2011). Volume 1380., Bonn, Germany, Springer (2011) 370–388

C-SPARQL queries and reasoning

- SPARQL is orthogonal to reasoning, so C-SPARQL is
- The C-SPARQL Engine 0.9.1
  - supports data-driven naïve stream reasoning
    - This is to be intended as a term of comparison for stream reasoning research
  - can answer queries under
    - RDFS entailment regime
    - RDFS++ entailment regime

C-SPARQL queries and reasoning - example

- Memo
  - *posts* is a sub property of *observes*
- Data

RDF graph	Time-stamp	Stream
:RedSensor :observes [:who :Alice; :where :RedRoom] .	T_1	rfid
:Bob :posts [:who :Bob ; :where :BlueRoom] .	T_2	fs

- Query

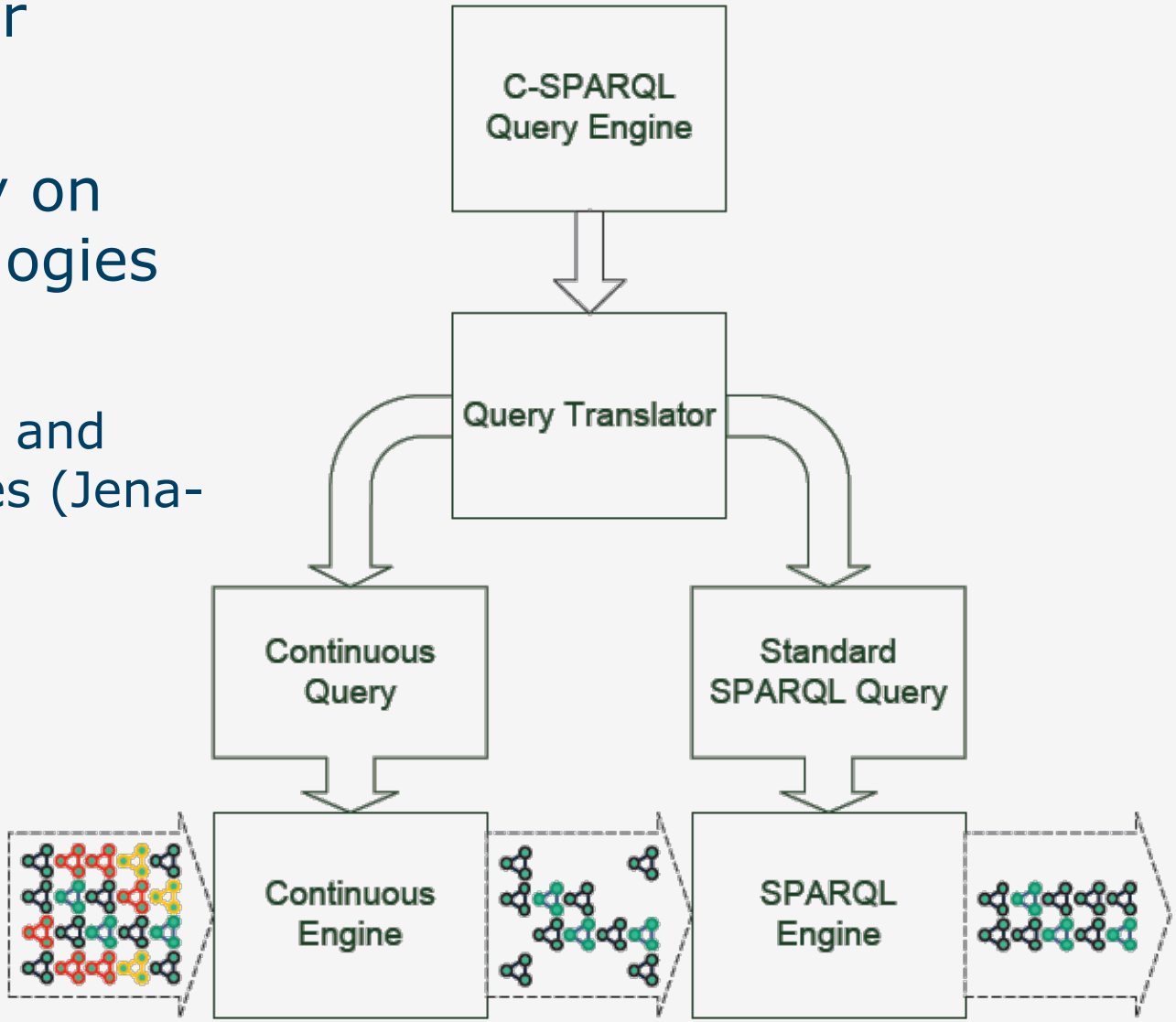
```
REGISTER QUERY QueryUnderRDFSEntailmentRegime AS
PREFIX : <http://…/sr4ld2013-onto#>
SELECT ?x ?room ?person
FROM STREAM <http://…/fs> [RANGE 1m STEP 10s]
FROM STREAM <http://…/rfid> [RANGE 1m STEP 10s]
WHERE { ?x :observes [ :who ?person ; :where ?room ] .}
```

- Results

?x	?room	?person
:RedSensor	:RedRoom	:Alice
:Bob	:RedRoom	:Bob

C-SPARQL Engine Architecture

- Simple, modular architecture
- It relies entirely on existing technologies
- Integration of
  - DSMSs (Esper) and
  - SPARQL engines (Jena-ARQ)



C-SPARQL Engine Features at a glance 1/3

- Efficient RDF stream Processing
  - Continuous queries, filtering, aggregations, joins, sub-queries via C-SPARQL (a SPARQL 1.1 extension)
  - Push based
  - High throughput, low latency
- Minimal support for
  - pattern detection via timestamp function
  - Background RDF graph access
- Naïve support for
  - RDFS reasoning
  - RDFS++ reasoning

- **Alert**! using timestamp function, "static" RDF graphs, and reasoning spoil performances

C-SPARQL Engine Features at a glance 2/3

- Extensible Middleware
  - Runtime management of
    - RDF streams
    - C-SPARQL query
    - Result listerners
  - API driven
  - RESTful service driven

- Quick start available
  - C-SPARQL Engine
    - http://streamreasoning.org/download/csparqlreadytogopack
  - RDF Stream Processging RESTful Interface (RSP-service) for C-SPARQL Engine
    - http://streamreasoning.org/download/rsp-service4csparql

C-SPARQL Engine Features at a glance 3/3

- Released open source under Apache 2.0
  - C-SPARQL Engine
    - https://github.com/streamreasoning/CSPARQL-engine
    - https://github.com/streamreasoning/CSPARQL-ReadyToGoPack
  - RSP-services
    - https://github.com/streamreasoning/rsp-services-csparql
    - https://github.com/streamreasoning/rsp-services-api
    - https://github.com/streamreasoning/rsp-services-client-example

Resources

- Read out more
  - C-SPARQL semantics
    - Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, Michael Grossniklaus: C-SPARQL: a Continuous Query Language for RDF Data Streams. Int. J. Semantic Computing 4(1): 3-25 (2010)
  - Most recent syntax
    - D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, M. Grossniklaus, Querying RDF streams with C-SPARQL, SIGMOD Record 39 (1) (2010) 20–26.
- Downloads
  - http://streamreasoning.org/download/csparqlreadytogopack
  - http://streamreasoning.org/download/rsp-service4csparql
- See demos
  - http://streamreasoning.org/demos
- Contact point
  - marco.balduini@polimi.it
  - emanuele.dellavalle@polimi.it

Stream Reasoning For Linked Data

M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, E. Della Valle, and J.Z. Pan

http://streamreasoning.org/sr4ld2013

C-SPARQL: A Continuous Extension of SPARQL

Marco Balduini
marco.balduini@polimi.it