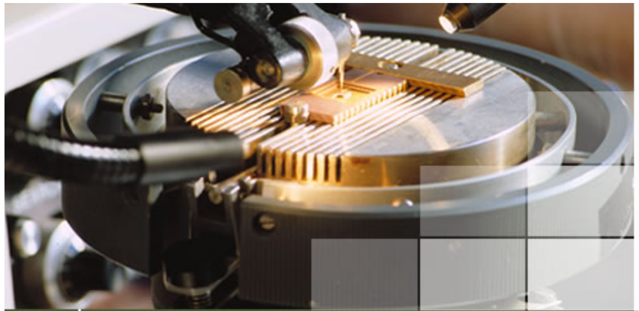
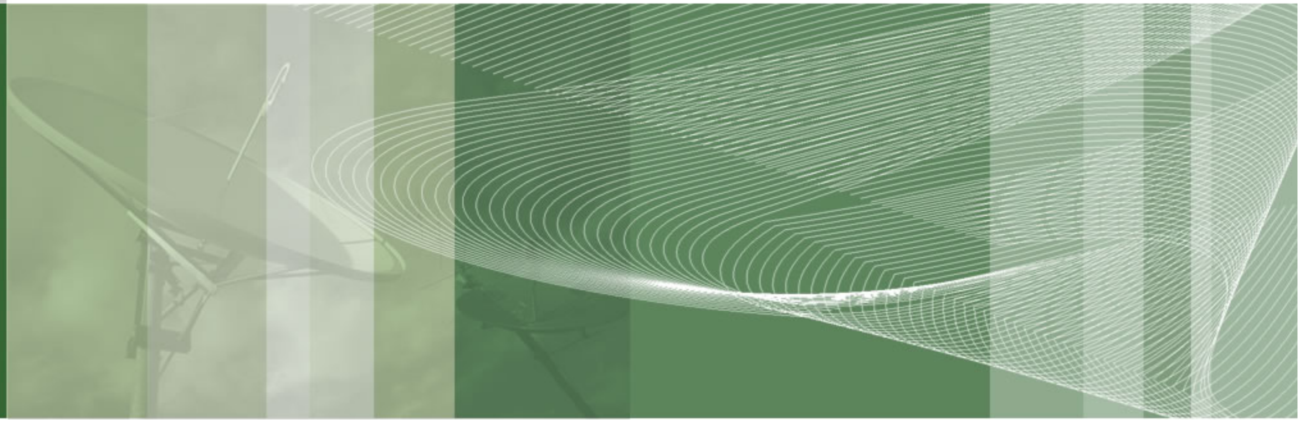


POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione



Data Stream Management Systems (DSMSs)

December 15, 2011

Davide Barbieri

Motivation

- Context:
  - Sensor networks (and IT network and traffic monitoring)
  - Financial applications
  - Web logs and click-streams

- Issues:
  - Real-time querying
  - analysis of huge amounts of data
  - Summarization with lossy approach instead of time-consuming processing of the whole database
  - Load is unpredictable (data reduction techniques needed)

Part 1

INTRODUCTION TO DATA STREAM MANAGEMENT

Data Streams

Data Streams: *Continuous, unbounded, rapid, time-varying, ordered sequence of data elements*

- Continuous:
  - Data is continuously flowing
  - We *can monitor* streams, we *can not store* them
- Unbounded
  - *The information carried can be thought as infinite, we can not forecast when a stream will stop to carry data*
- Rapid, Time-varying:
  - In principle we can not assume anything on the distribution of events over time
- Ordered
  - The arrival time is crucial in most applications

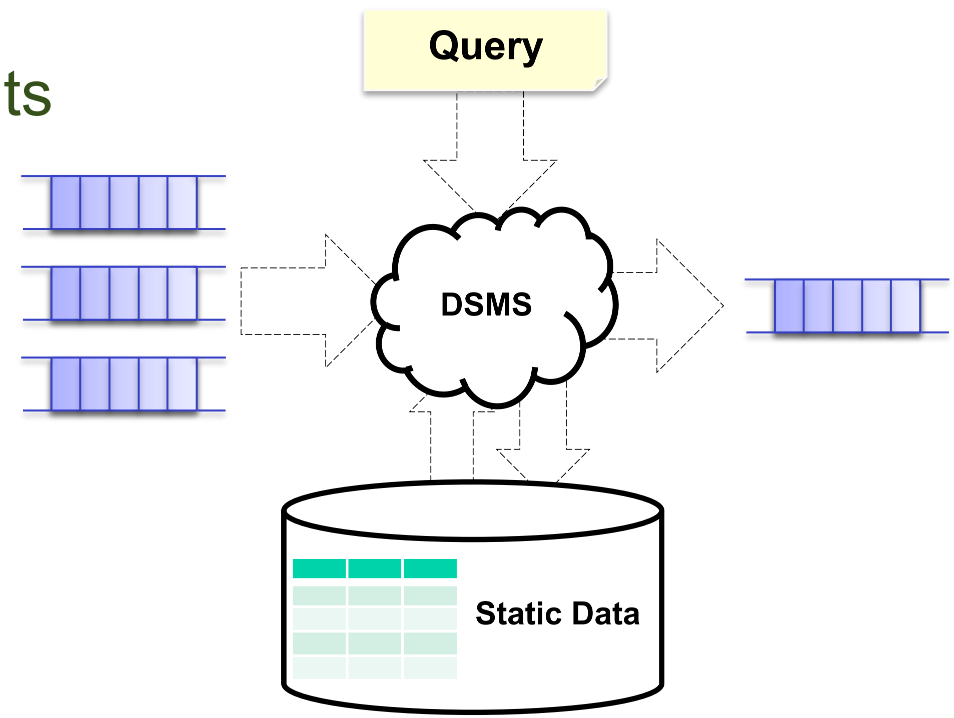
Comparison between DSMSs and DBMSs

DSMS	DBMS
Model: mainly **transient data**	Model: **persistent data**
Infinite sequence of tuples	Table: **set | bag** of tuples
Access: **sequential**	Access: **random**
Updates: **append only**	Updates: **all primitives**
Query: **persistent (continuous) queries**	Query: **transient queries**
Query Answer: **often approximate**	Query Answer: **typically exact**
Query Eval. **one-pass**	Query Eval. **multi-pass**
Operators: **unblocking only**	Operators: **blocking**
Query Plan: **adaptive**	Query Plan: **fixed**

DSMS Generic Architecture

A four step process:

1. Stream registration
2. Static data import (optional)
3. Query registration
4. Subscription to results

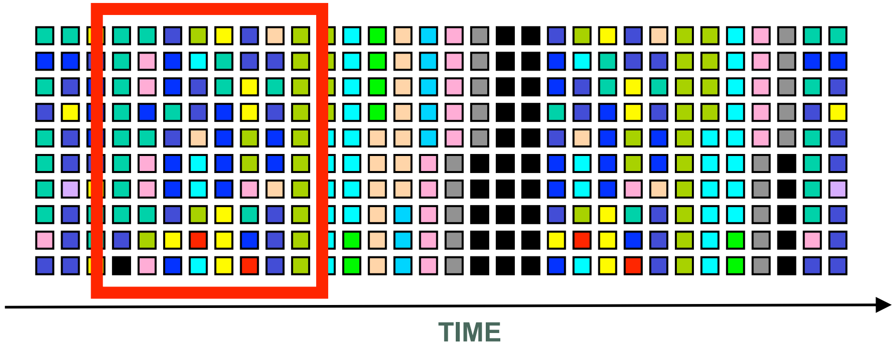


From Database Systems to DSMSs

- 1980s - Relational systems & query languages
- 1990s - Active rule systems
- 2000s - Continuous query engines

DSMS name	Year	Querying paradigm/ model	Architecture
Gigascope	2002	GSQL (SQL-like syntax)	Centralized
TelegraphCQ	2003	SQL-like language with programmatic definition of selection windows	Centralized
Aurora	2003	Algebraic, WYSIWYG Query Plan Editor	Centralized
STREAM	2004	CQL (SQL-like language)	Centralized
Borealis	2005	Same as Aurora	Distributed
Wavescope	2008	Programmatic approach, Wavescript (Ocaml-like language)	Generates distributed applications

Selection over Data Streams: Windows



- Landmark window: starts with the stream, unbounded
  - good for materialization of “slow” streams
- Window features:
  - Size
    - Physical (by number of elements)
    - Logical (by time)
  - Stride (pace)
    - Tumbling (not overlapping)
    - Sliding

Join and Aggregation

- *Approximate join* is needed
  1. Join windows from each stream
  2. merge the result
- *Aggregation:* first-class citizen in streaming applications
  - Data has to be summarized, can not be stored
  - Usually synopses are sufficient to accomplish application tasks
  - Synopses can be stored as opposed to entire streams
- Load management challenges
  - Quality of service (regular DBs weak here)
  - Multi-query optimization
  - Careful resource allocation & use
  - Graceful load shedding
  - Sampling

Part 2

C-SPARQL: BRIDGING STREAMING DATA AND REASONING

Motivations

- The are many scenarios requiring a fast solution to complex reasoning problems
  - Traffic jam detection
  - Routing traffic to avoid congestion
- Querying very large RDF knowledge bases with SPARQL has known scalability issues
- Combination of static and streaming RDF data leads to *stream reasoning* [1]
  - Enabling current reasoners to also consider rapidly changing data

[1] E. Della Valle, S. Ceri, D. Barbieri, D. Braga, A. Campi: **A First Step Towards Stream Reasoning** In Proceedings of **FIS Future Internet Symposium**. Vienna, Austria, September 2008

Reasoning on Streaming Data

- **RDF data streams:** new data format set at the confluence of relational data streams and RDF data, defined as

“an ordered sequence of pairs of an RDF triple and its timestamp”

(<subj, pred, obj>, ts)

Timestamps are not required to be unique

Querying RDF streams: C-SPARQL

- Continuous SPARQL:
  - minimal extension to SPARQL [2]
  - compliant with version 1.1 [3]
  - operational formal semantics
- C-SPARQL engine:
  - an optimized execution environment
  - extensible plug-in architecture
  - Efficient maintenance of ontological entailments
  - Validated on social data use cases

- [2] D. Barbieri, D. Braga, S. Ceri, E. Della Valle, M. Grossniklaus**: C-SPARQL: SPARQL for Continuous Querying**, WWW 2009
- [3] D. Barbieri, D. Braga, S. Ceri, E. Della Valle, M. Grossniklaus: **Querying RDF Streams with C-SPARQL - ACM SIGMOD Record**, 2010

Query and Stream Registration

- All queries over RDF data streams are continuous
  - Registered through the **`REGISTER QUERY`** statement
  - Run at a frequency specified by the optional **`COMPUTED EVERY`** clause
- The output of queries is in the form of tables of variable bindings or RDF graphs
- If it is a **`CONSTRUCT`** or **`DESCRIBE`** query, can be registered as a new RDF stream (through the **`REGISTER STREAM`** statement)
- **Composability**:
  - Query results registered as streams can feed other registered queries like any other RDF stream

Selection Over Streams

- In a C-SPARQL query, data streams are associated with an IRI
  - As in standard **`FROM`** clause

- Windows over streams are specified through the **`FROM STREAM`** clause
  - Can be physical or logical, tumbling or sliding

```
– FROM STREAM <http://streams.org/p.trdf> [TRIPLES 100]
– FROM STREAM <http://streams.org/l.trdf> [RANGE 30m STEP 5m]
```

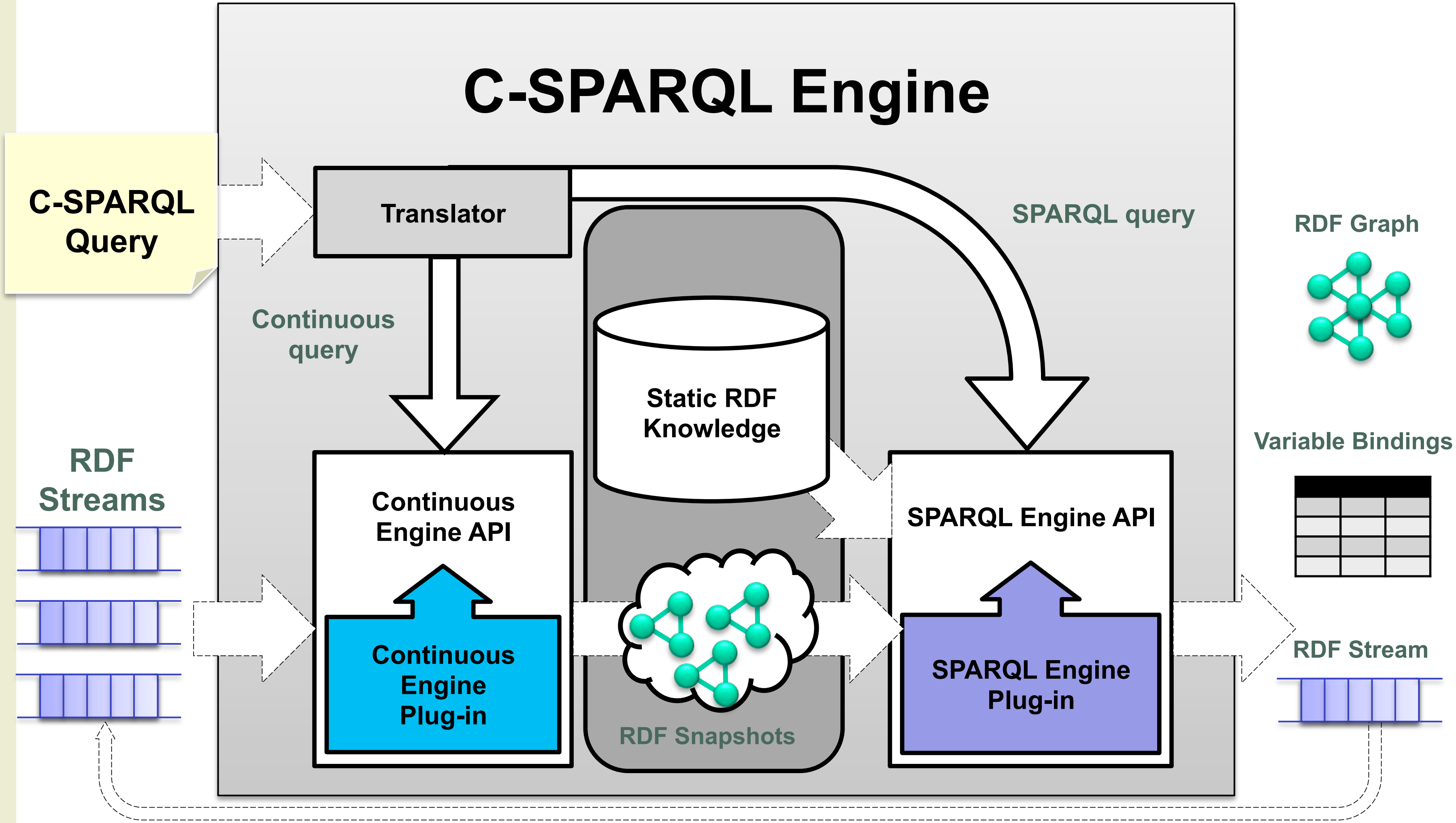
- C-SPARQL queries can combine triples from more than one RDF stream
  - More than one **`FROM STREAM`** clause in a query are supported
  - Every **`FROM STREAM`** can have its own window definition

Another C-SPARQL Example

```
REGISTER STREAM CarsEnteringCityCenterPerDistrict COMPUTED EVERY 5m
   AS

CONSTRUCT {?district t:has-entering-cars COUNT(?car) AS ?passages}

FROM <http://city.org/info.rdf>

FROM STREAM <http://streams.org/gates.trdf> [RANGE 30m STEP 5m]

WHERE { ?tollgate t:registers ?car .

        ?district c:contains ?street .

        ?tollgate c:placedIn ?street . }

GROUP BY ?district
```

Subject	Predicate	Object	Timestamp
district-1	has-entering-cars	21	T_1
district-2	has-entering-cars	34	T_1
district-1	has-entering-cars	15	T_2
district-2	has-entering-cars	7	T_2

C-SPARQL Engine Architecture [4]

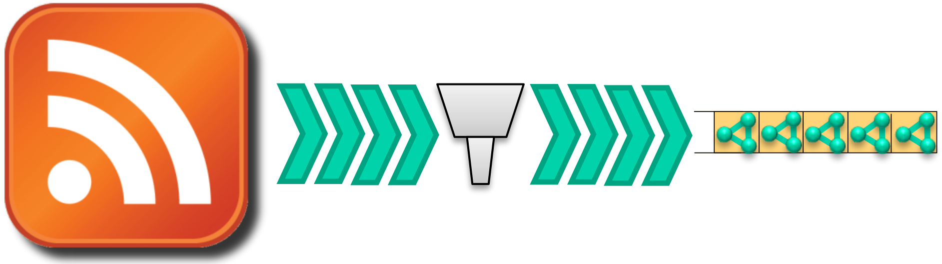


[4] D. Barbieri, D. Braga, S. Ceri and M. Grossniklaus: An Execution Environment for C-SPARQL Queries International Conference on Extending Database Technology (EDBT) 2010

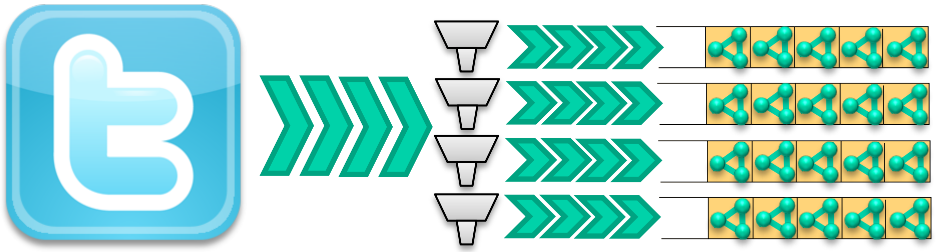
Use cases and APIs

- Some RDF-Stream generators have been developed:

RSS2RDFStream



Twitter2RDFStream [7] [8]



A ReST API has been designed in order to allow the invocation of C-SPARQL Engine as a service [8]

- [7] D. Barbieri, D. Braga, S. Ceri, E. Della Valle and M. Grossniklaus: ***Continuous Queries and Real-time Analysis of Social Semantic Data with C-SPARQL***, **(SDoW2009)**
- [8] D. Barbieri, and E. Della Valle: **A Proposal for Publishing Data Streams as Linked Data - A Position Paper -** Linked Data on the Web (**LDOW2010)**
- [9] D. Barbieri et al.: **Deductive and Inductive Stream Reasoning for Semantic Social Media Analytics - IEEE Intelligent Systems** 2010

Future Work

- Extending reasoning:
  - techniques that can exploit the strong temporal connotation of the triples in a RDF stream
- Optimizations:
  - Multi-query optimization
  - Distribution and parallelism
  - Low-level improvements (e.g. anticipation of aggregates)
- Extending the implementation:
  - Dynamic adaptation of query computation to load/ source availability