CSE 544
Principles of Database Management Systems

Alvin Cheung

Fall 2015

Lecture 16 - Stream Processing

Announcements

- HW1 graded
  – Send staff an email if you have comments

- Lecture plan for last 2 weeks of classes posted online
  – Next Tuesday will be the last class

- No OH today

Course Outline

- Data Models
- Query Execution
- Data Analytics (OLAP)
- Transaction Processing (OLTP)
- Recovery and Replication

- Advanced Topics
  - Today: stream processing
  - Thursday: DBMS in the real world
  - Next Tuesday: NoSQL

References

- **Aurora: A New Model and Architecture for Data Stream Management.** Daniel Abadi et. al. VLDB Journal. 12(2). 2003

- Additional references:
  - Chandrasekaran et al, “TelegraphCQ: Continuous Dataflow Processing for an Uncertain World.” CIDR 2003.
  - The STREAM Group, “STREAM: The Stanford Stream Data Manager.” IEEE Data Engineering Bulletin, March 2003.
  - Meehan et al, “S-Store: Streaming Meets Transaction Processing.” PVLDB 8(13), 2015.

Outline

- **Stream processing applications**
  - Background
  - Examples
  - Requirements
- **Aurora system**
  - Stream model and query model
  - Processing model
  - Operators
  - Query examples
  - Other features
- **STREAMS system**
  - DSMS motivation
  - CQL
  - Query evaluation

Why data streams?

- Data constantly being generated all the time
  - Trading transactions, sensors, phones
- Real-time processing required
  - Update trade positions, people's locations, etc
  - Cannot wait until data are ingested into warehouse
- Too much data to store!
  - Airbus A350 generates 2.5Tb of data **per day** with 6000 sensors
  - New model in 2020 will capture 3x that amount

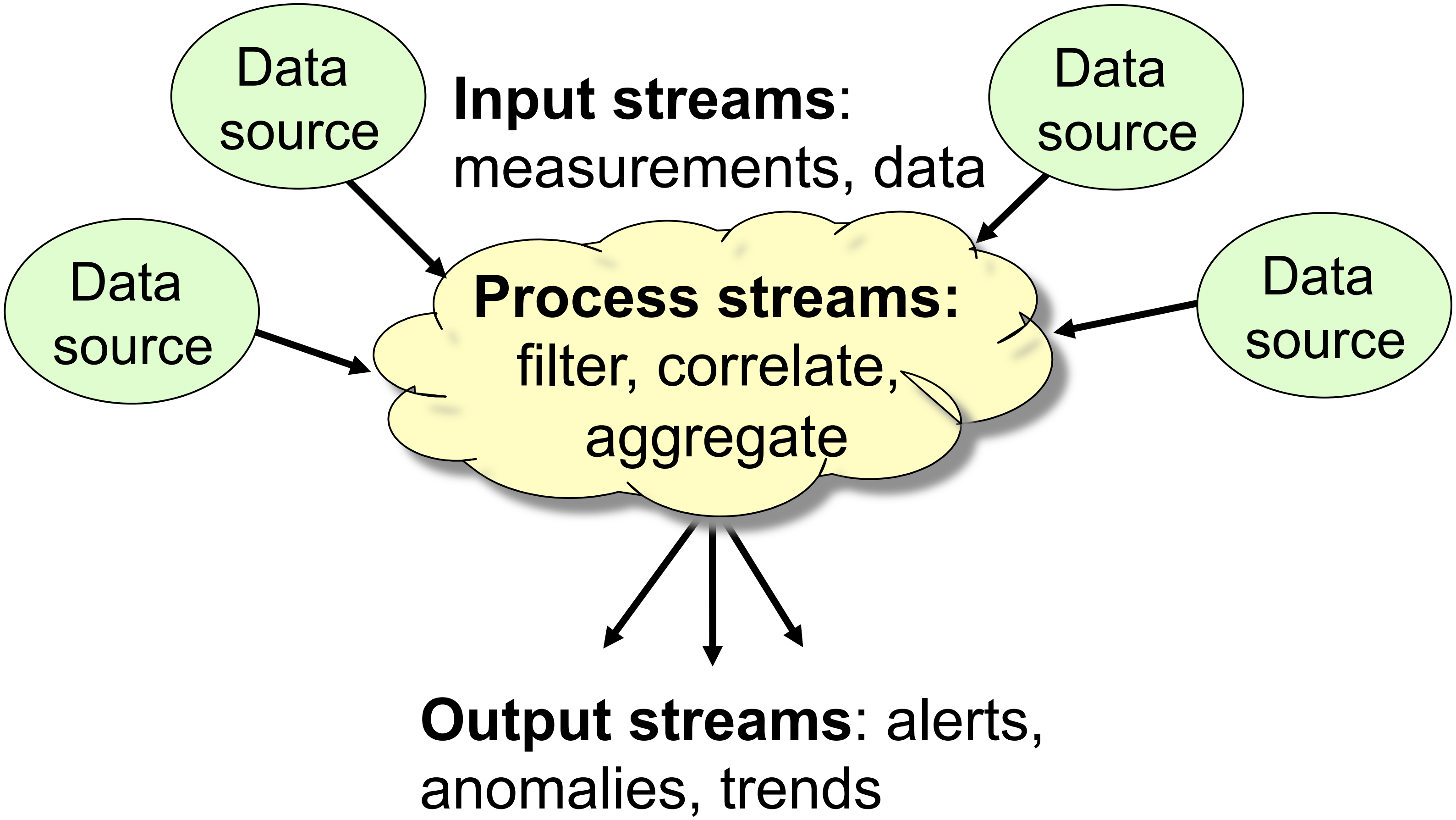
Why data streams?

- Four Vs of big data:
  - **V**olume
  - **V**elocity
  - **V**ariety
  - **V**eracity

Why data streams?

- Four Vs of big data:
  - **V**olume
  - **V**elocity
  - **V**ariety
  - **V**eracity

Stream Processing



Application Domains

- Network monitoring
  - Intrusion, fraud, anomaly detection, click streams
- Financial services
  - Market feed processing, ticker failure detection
- Sensor-based environment monitoring
  - Weather conditions, air quality, car traffic
  - Civil engineering, military applications, etc.
- Medical applications
  - Patient monitoring, equipment tracking
- Near real-time data analytics

Requirements

- **Input data is pushed continuously**
  - Traditional DBMSs not designed for continuous loading or inserting of individual data items
  - “DBMS-active, human passive” model
- **Users want to execute continuous queries**
  - Traditional DBMSs have no direct support for such queries. Can use triggers, but triggers do not scale
- **Low-latency processing**
  - Need to see results in near real-time
  - Data is possibly high-volume and high-rate

Other Requirements

- Distribution
- Load management and load shedding
- Approximate processing, approximate answers
- Fault-tolerance and revision processing
- Exploiting data archives

Outline

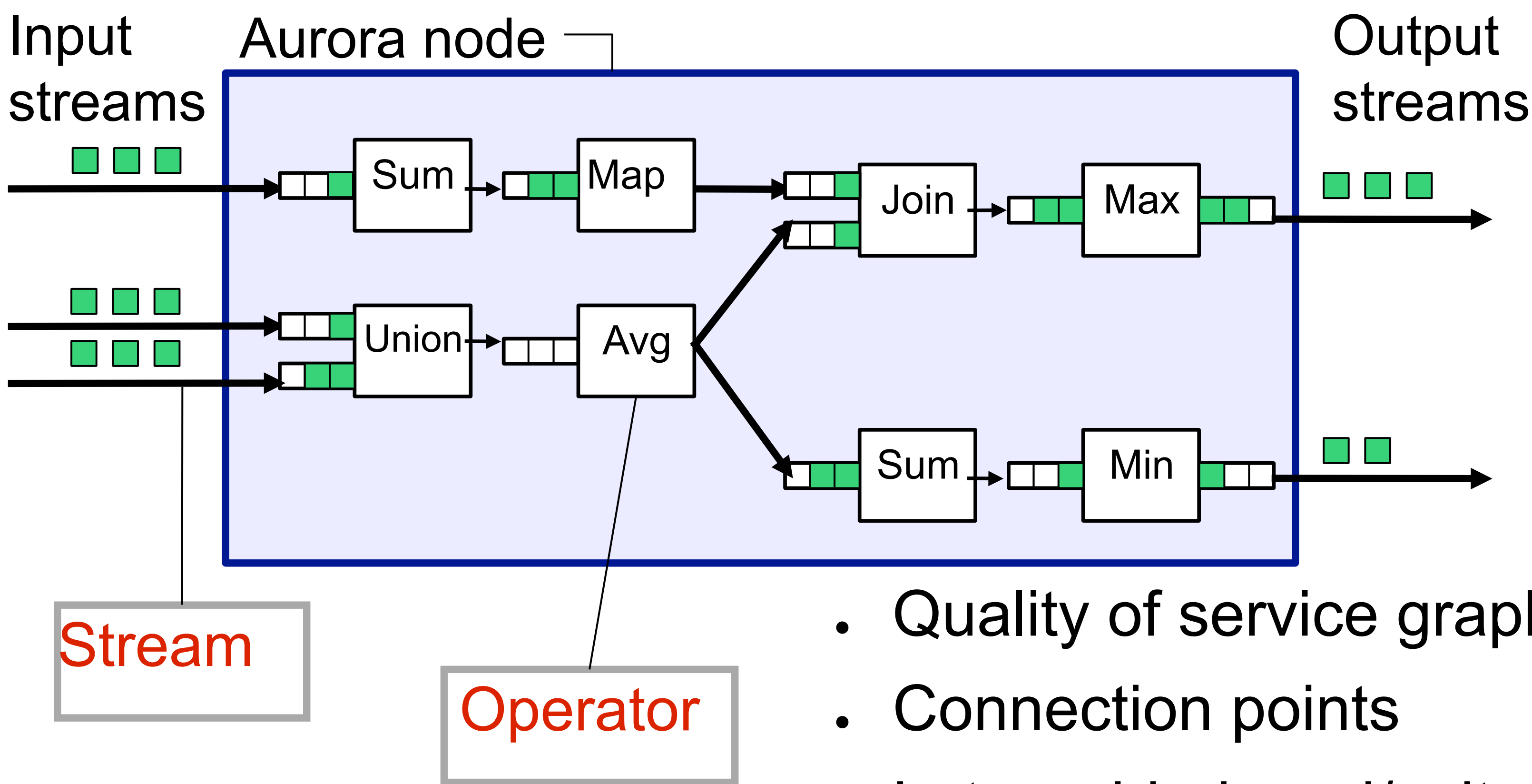
- **Stream processing applications**
  - Examples
  - Requirements
- **Aurora system**
  - Stream model and query model
  - Processing model
  - Operators
  - Query examples
  - Other features
- **STREAMS system**
  - DSMS motivation
  - CQL
  - Query evaluation

Stream Data Model

Tuple: $(\underbrace{\texttt{timestamp},}_{\text{header}}\ \underbrace{\texttt{v}_1\texttt{,...,v}_n}_{\text{data}})$

- Stream: append-only sequence of tuples
- All tuples on a stream have same schema
- Timestamp is used for QoS

Query Model



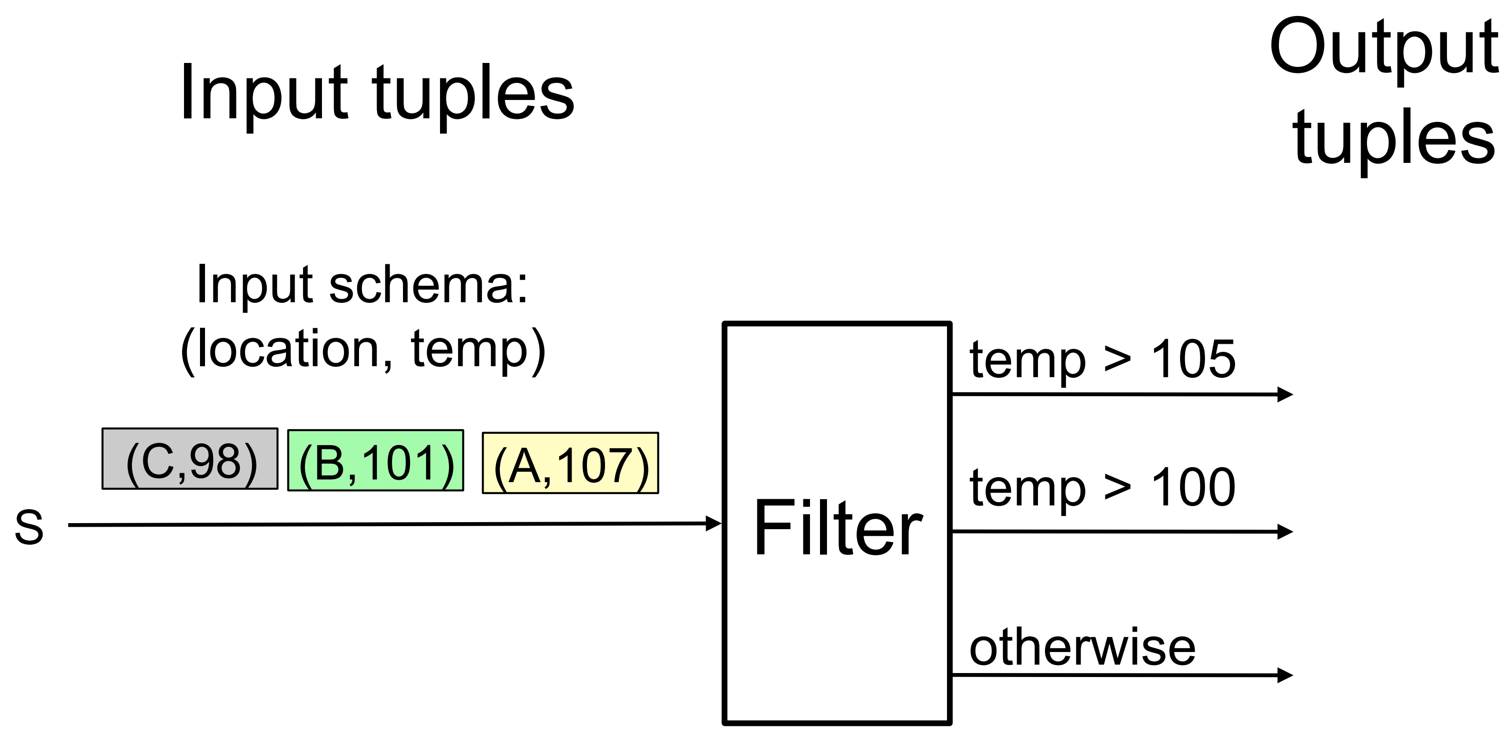
- Quality of service graphs
- Connection points
- Later added read/write ops
- **No query language (!)**

Aurora Operators

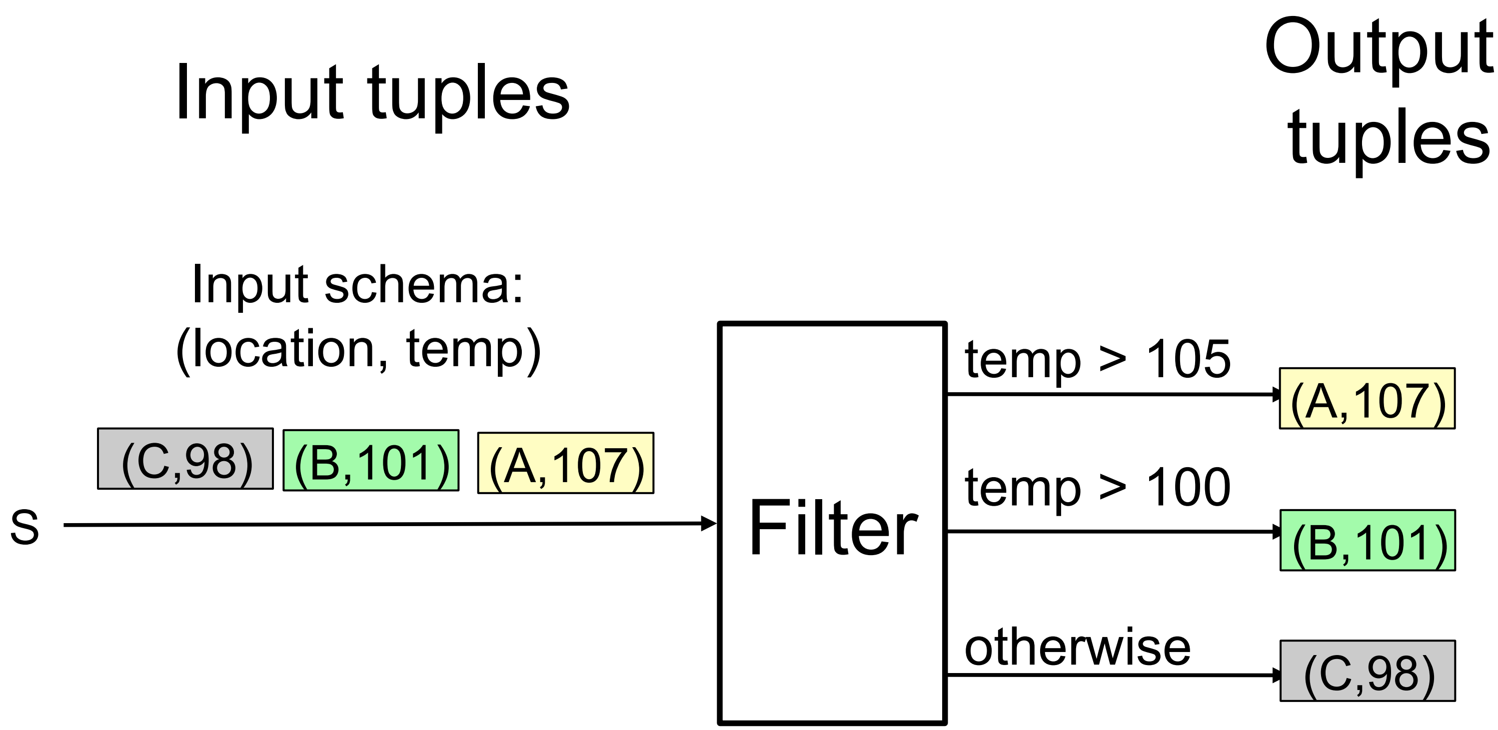
- Order-agnostic
  - Filter
  - Map
  - Union
- Order-sensitive
  - Aggregate
  - Join
  - BSort, Resample

- **Why do we need new operators?**
  - Ops cannot block & cannot accumulate state that grows with input

Filter Example



Filter Example



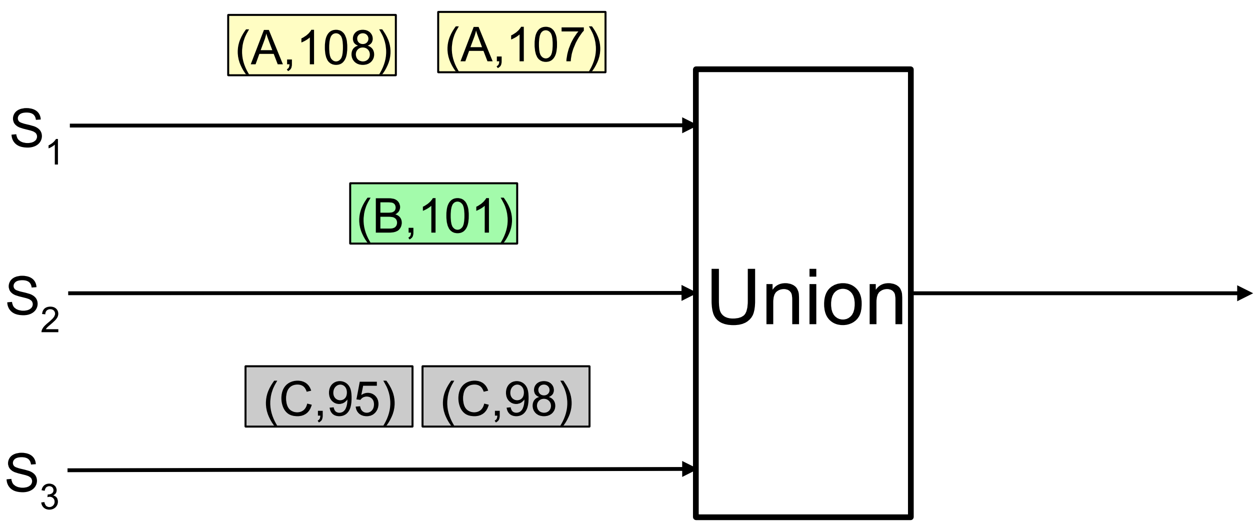
Map Example

new.location = old.location
new.temp_celcius = 5/9*(old.temp - 32)

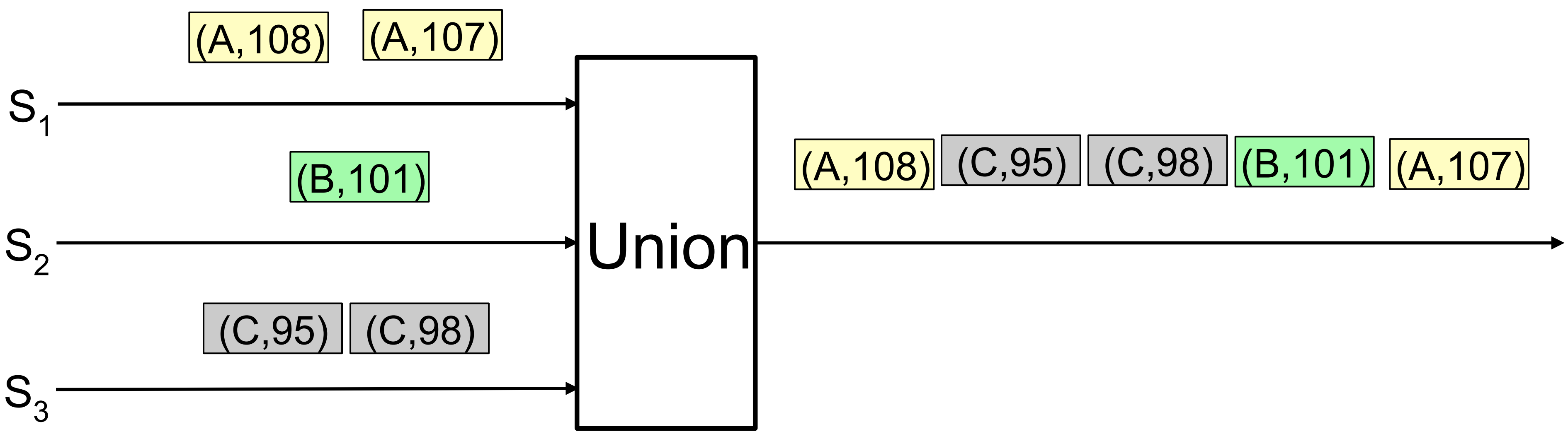
Map Example

new.location = old.location
new.temp_celcius = 5/9*(old.temp - 32)

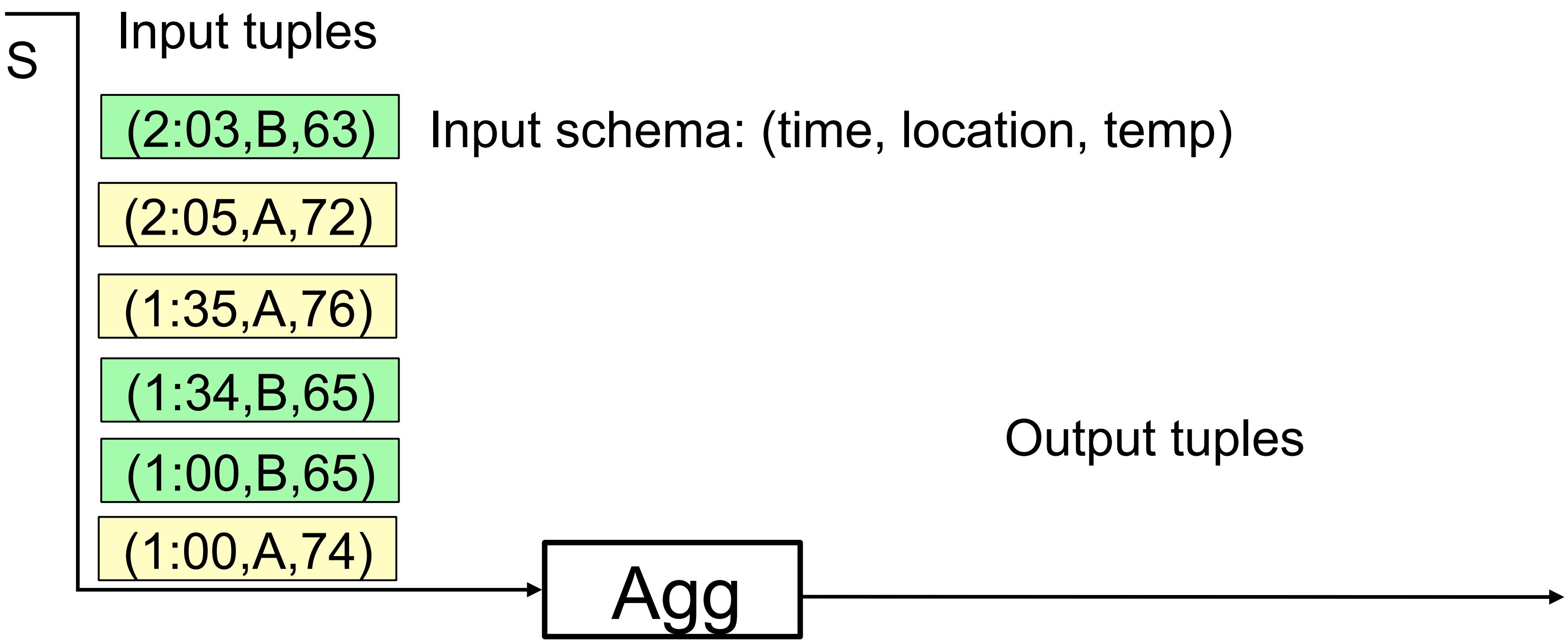
Union Example



Union Example

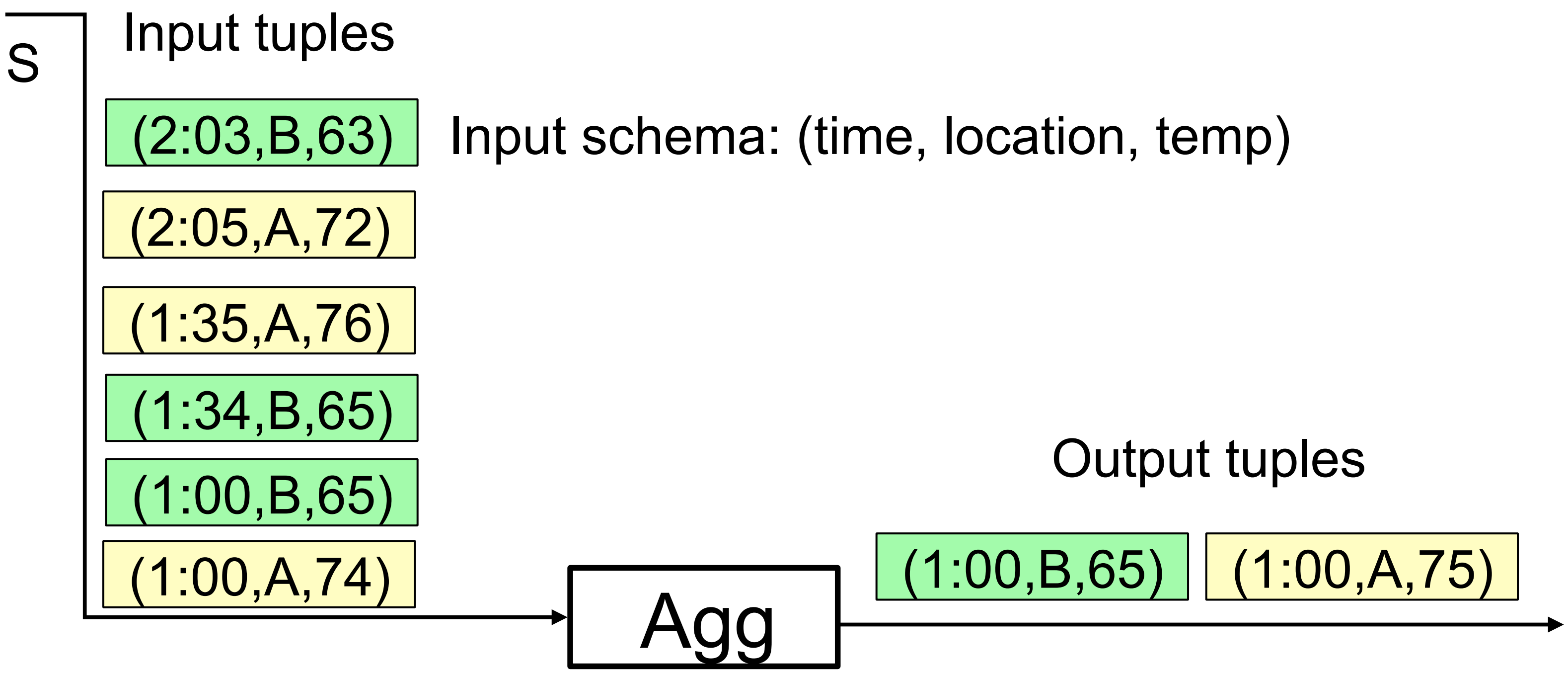


Aggregate Example



Operator parameters:
- group by location
- average temp,
- order on time, window size 1 h, advance 1h

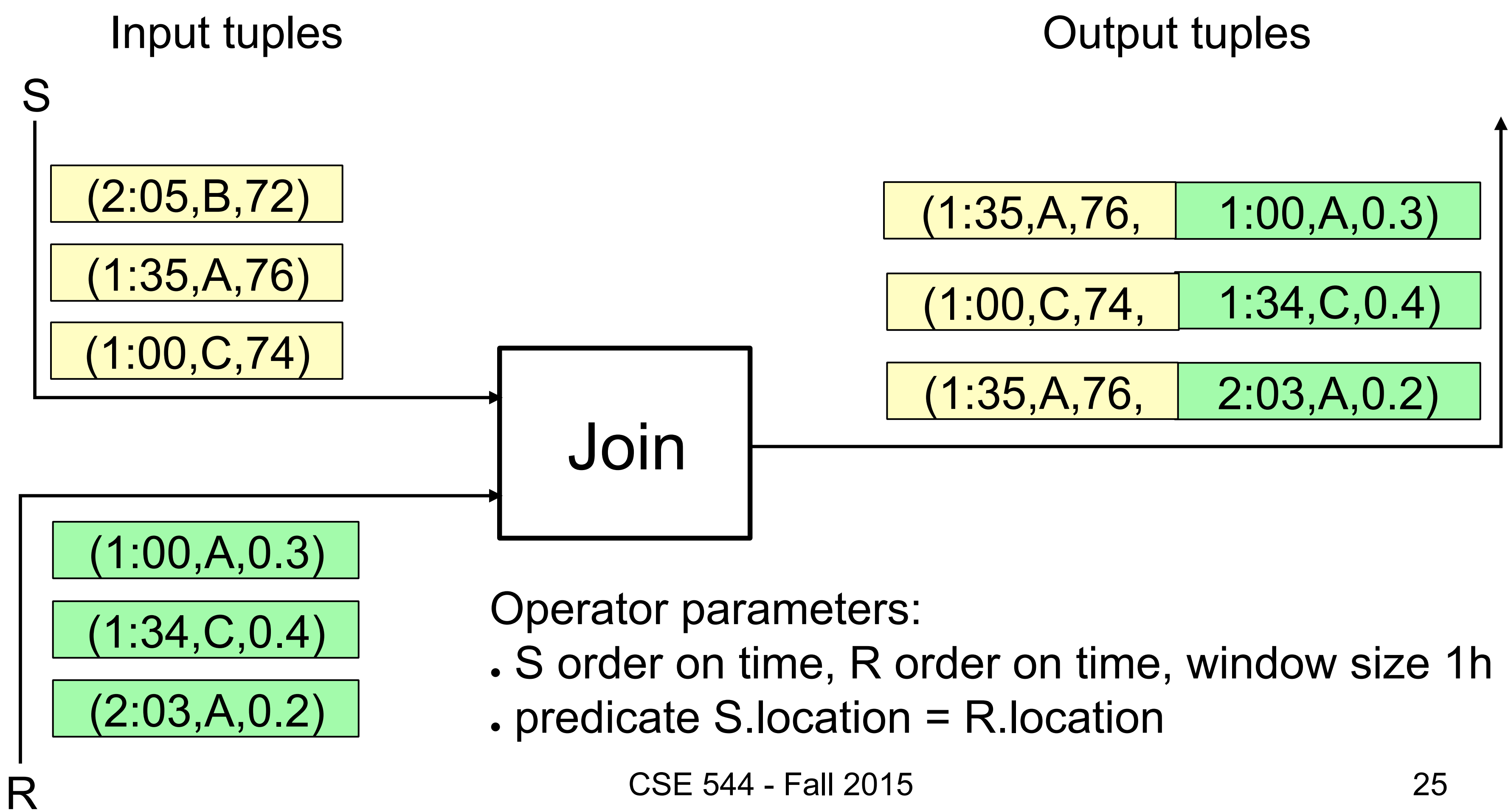
Aggregate Example



Operator parameters:

- group by location
- average temp,
- order on time, window size 1 h, advance 1h

Join Example



Operator parameters:
- S order on time, R order on time, window size 1h
- predicate S.location = R.location

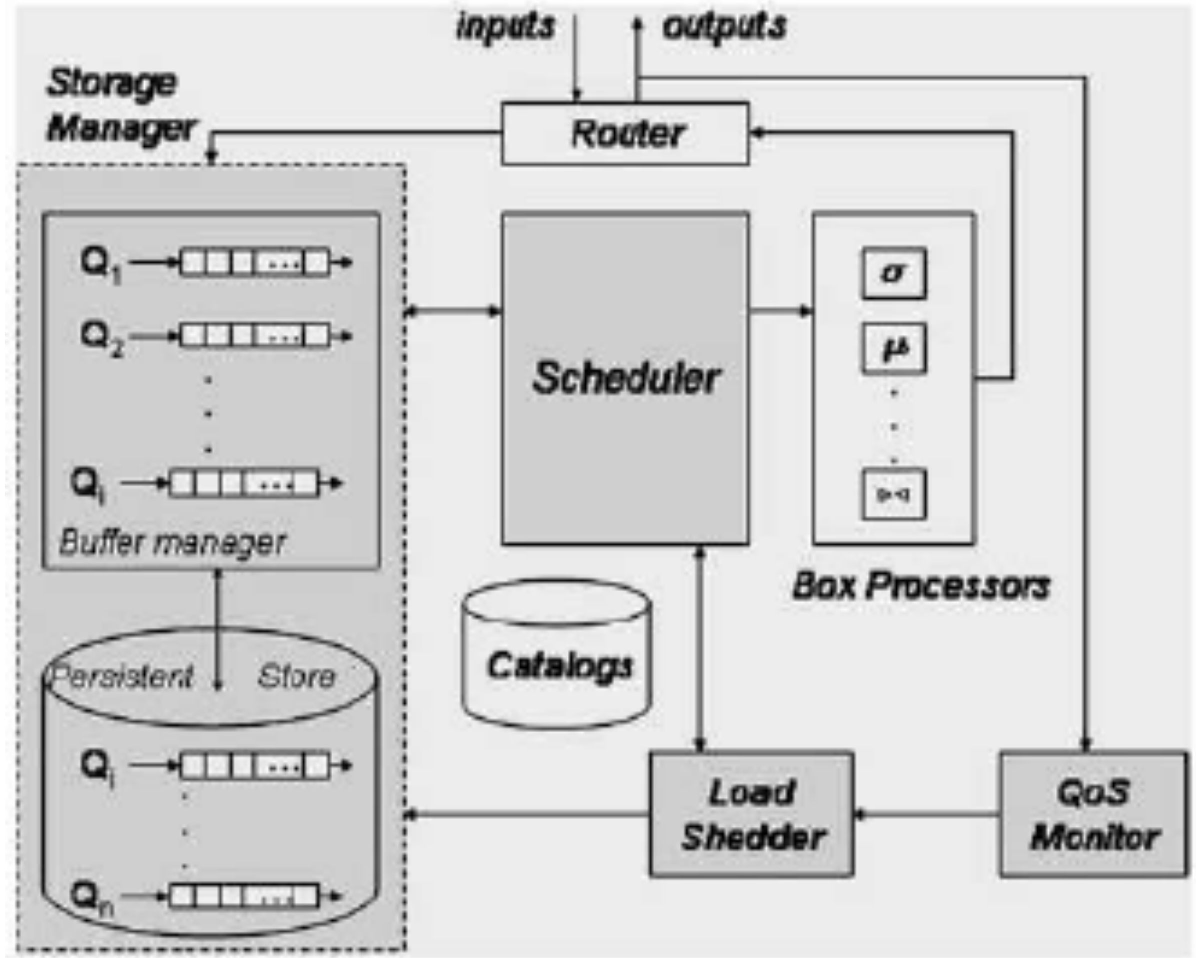
Sample Query

- Application: network intrusion detection
- Schema of input stream

  `(src_ip,src_port,dst_ip,dst_port,time)`

- **Query**
  - Alert me if an IP address establishes more than 100 connections per minute
  - and within 30 seconds of that event
  - the IP tries to connect to more than 10 distinct ports within a minute

Processing Model



[Figure 3 from Abadi 03]

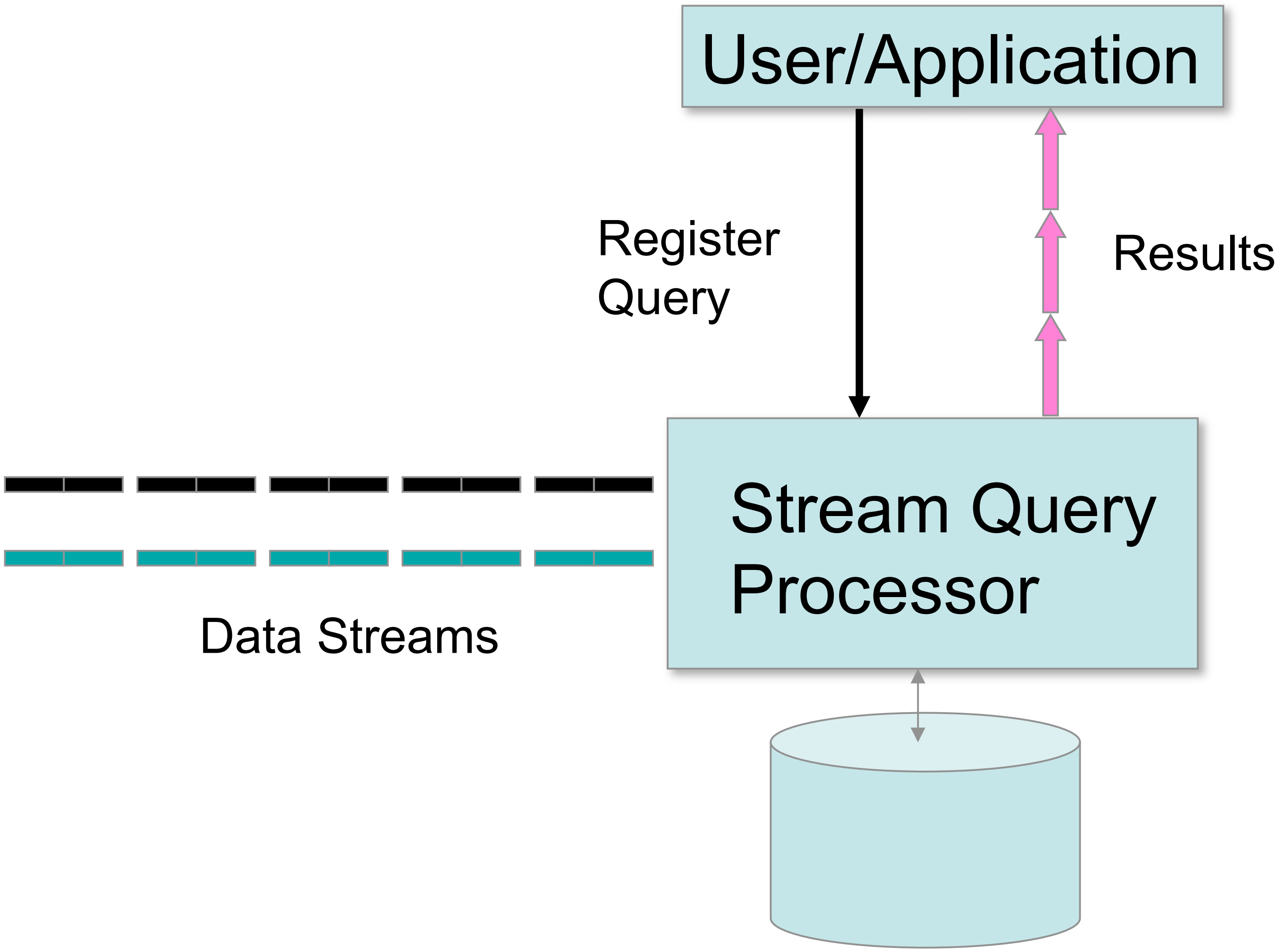
Additional Features

- **Load management**
  - What happens when system is overloaded?
- **Fault-tolerance**
  - What happens if a node fails?
  - What happens if the network fails?
  - What happens if input data is wrong?
- **Exploiting data archives**
  - Historical queries, ad-hoc queries
  - Integrating push-based processing with pull-based

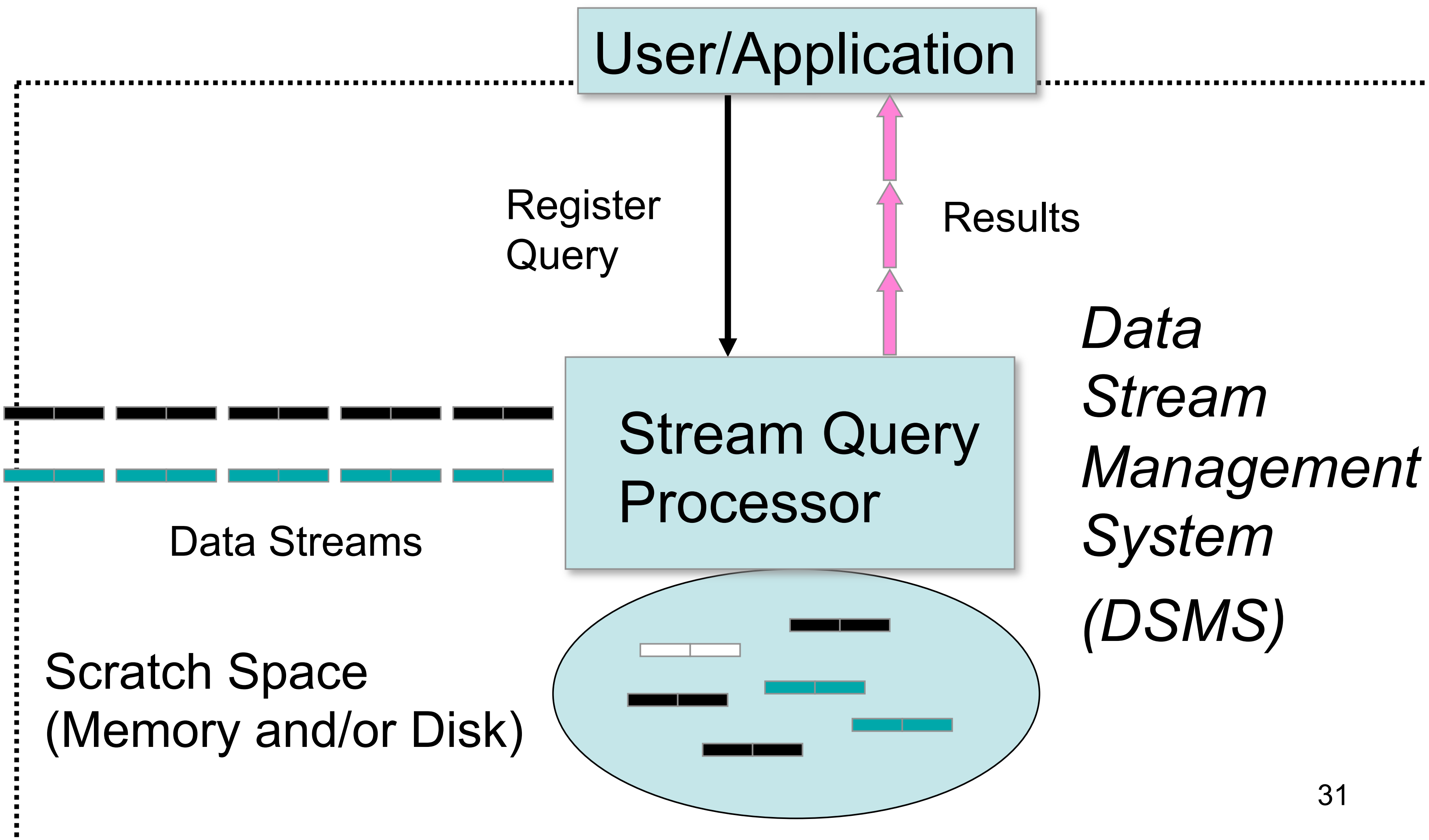
Outline

- **Stream processing applications**
  - Examples
  - Requirements
- **Aurora system**
  - Stream model and query model
  - Processing model
  - Operators
  - Query examples
  - Other features
- **STREAMS system**
  - DSMS motivation
  - CQL
  - Query evaluation

System Model



New Approach for Data Streams



DBMS versus DSMS

- Persistent relations
- One-time queries
- Random access
- Access plan determined by query processor and physical DB design
- “Unbounded” disk store

- Transient streams (and persistent relations)
- Continuous queries
- Sequential access
- Unpredictable data arrival and characteristics
- Bounded main memory

Query Language & Semantics

- Specifying queries over streams
  - SQL-like versus dataflow network of operators
  - **Sliding windows** as a query construct
- Semantic issues
  - Blocking operators, e.g., *aggregation, order-by*
  - Streams as sets versus lists
  - Timestamping
  - (compare to Aurora)

Issues in Query Evaluation

- Approximation
- Adaptivity
- Multiple queries
- Distributed streams

Query Evaluation – Approximation

- Why approximate?
  - Streams are coming too fast
  - Exact answer requires unbounded storage or significant computational resources
  - Ad hoc queries reference history

- Issues in approximation
  - Sliding windows, sampling, synopses, …
  - How is approximation controlled?
  - How is it understood by user?

- Tradeoff between accuracy / efficiency / storage
- A lot of work on **streaming algorithms**

Query Evaluation – Adaptivity

- Why adaptivity?
  - Queries are long-running
  - Fluctuating stream arrival & data characteristics
  - Evolving query loads
- Issues in adaptivity
  - Adaptive resource allocation (memory, computation)
  - Adaptive query execution plans

Query Evaluation – Multiple Queries

- Possibly large number of continuous queries
- Long-running
- Shared resources
- Multi-query optimization

Query Evaluation – Distributed Streams

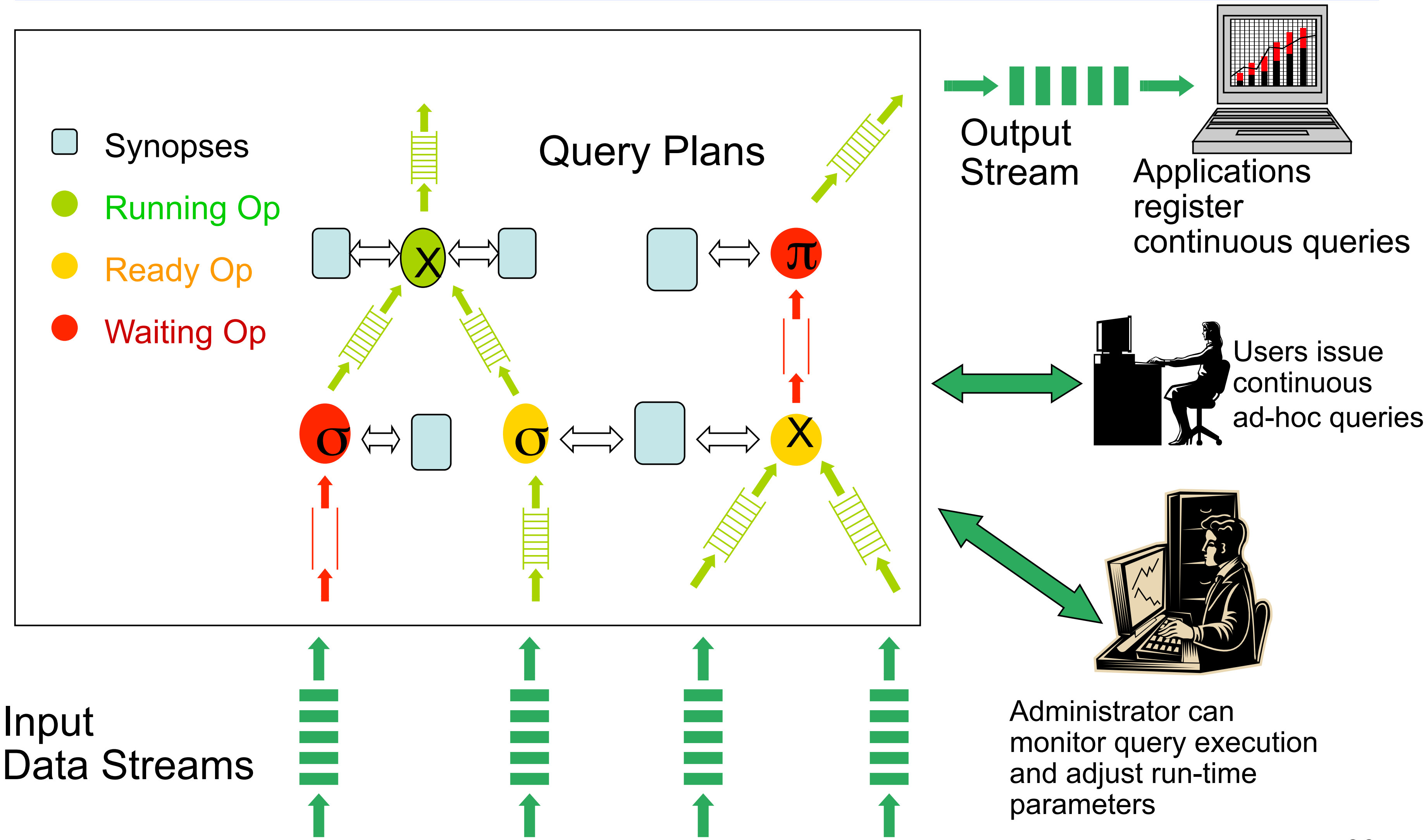
1 Many physical streams but one logical stream
  - e.g., maintain top 100 visited pages at Yahoo

2 Correlate streams at distributed servers
  - e.g., network monitoring

3 Many streams controlled by a few servers
  - e.g., sensor networks

- Issues
  - Move processing to streams, not streams to processor
  - **Approximation-bandwidth tradeoff**

STREAM Architecture



STREAM Internals

- Query plans: **operators, synopses, queues**

- Memory management
  - Dynamic allocation to buffers, queues, synopses
  - Accuracy vs. memory tradeoff
  - Operators adapt gracefully to memory reallocation

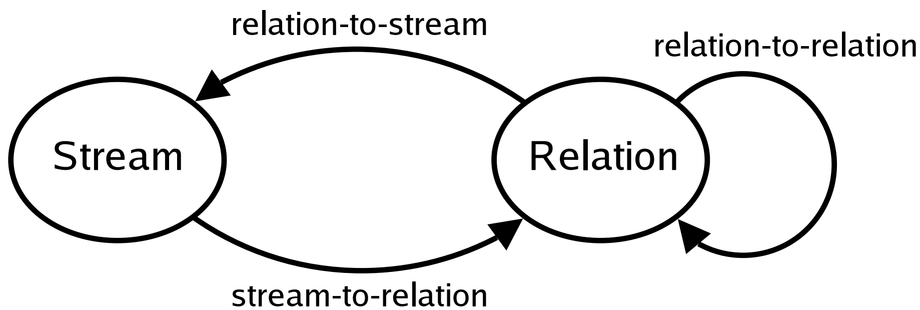
- Scheduler
  - Handles variable-rate input streams
  - Handles varying operator and query requirements

CQL: Data Models

- Continuous Query Language
- Data models: **both** data streams and relations
  - Streams: unbounded bag (multiset) of (s, t) pairs
    - s: tuple
    - t: timestamp of the arrival time of s
  - Relations: time-varying bags of tuples
    - R(t): bag of tuples at time t
    - Also called an **instantaneous relation**

CQL: Operators

- Should be able to convert from relations to streams, streams to relations, relations to relations



Stream-to-Relation Operators

- Tuple-based sliding window
  - [Rows N] : returns the N tuples from stream with largest timestamps from a relation
  - Example: R(t) [Rows N]
  - [Rows Unbounded] means all return tuples from relation
- Time-based sliding window
  - [Range w] : returns all tuples from a relation with timestamps between t and w
  - Example: R(t) [Range w]
- Partitioned sliding window
  - $\{A_1, A_2, \ldots, A_k\}$: divide stream into k different substreams where each A_i is true

Relation-to-Stream Operators

- Istream (insert stream) : returns a stream from relation R, with a tuple generated whenever a tuple is inserted into R
- Dstream (delete stream) : returns a stream from relation R, with a tuple generated whenever at a tuple is deleted from R
- Rstream (relation stream) : returns a stream that contains a snapshot of relation R at particular time instant t

CQL Example

- Select Istream(*)
  From S [Rows Unbounded]
  Where S.A > 10

- Evaluation:
  - Convert S to a relation by applying [Rows Unbounded]
  - Evaluate predicate S.A > 10
  - Convert results into a stream by applying Istream(*)

- What query is this equivalent to?

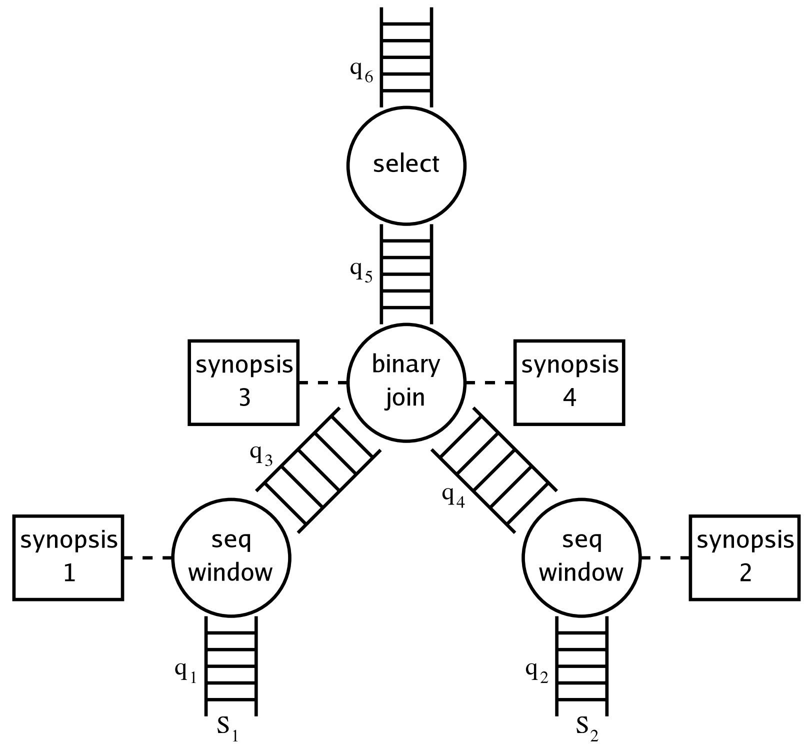
Plan Implementation

- A CQL query plan contains three components:
  - Operators
  - Queues
  - Synopses

- Operators: filters, R-to-S, S-to-R, etc
- Queues: buffers that store intermediate outputs between operators
  - Why is that needed?
- Synopses: current state of each operator
  - Last timestamp of processed tuples in a join

Example of Physical Plan

- Select *
  From S1 [Rows 1000],
  S2 [Range 2 Minutes]
  Where S1.A = S2.A
  And S1.A > 10



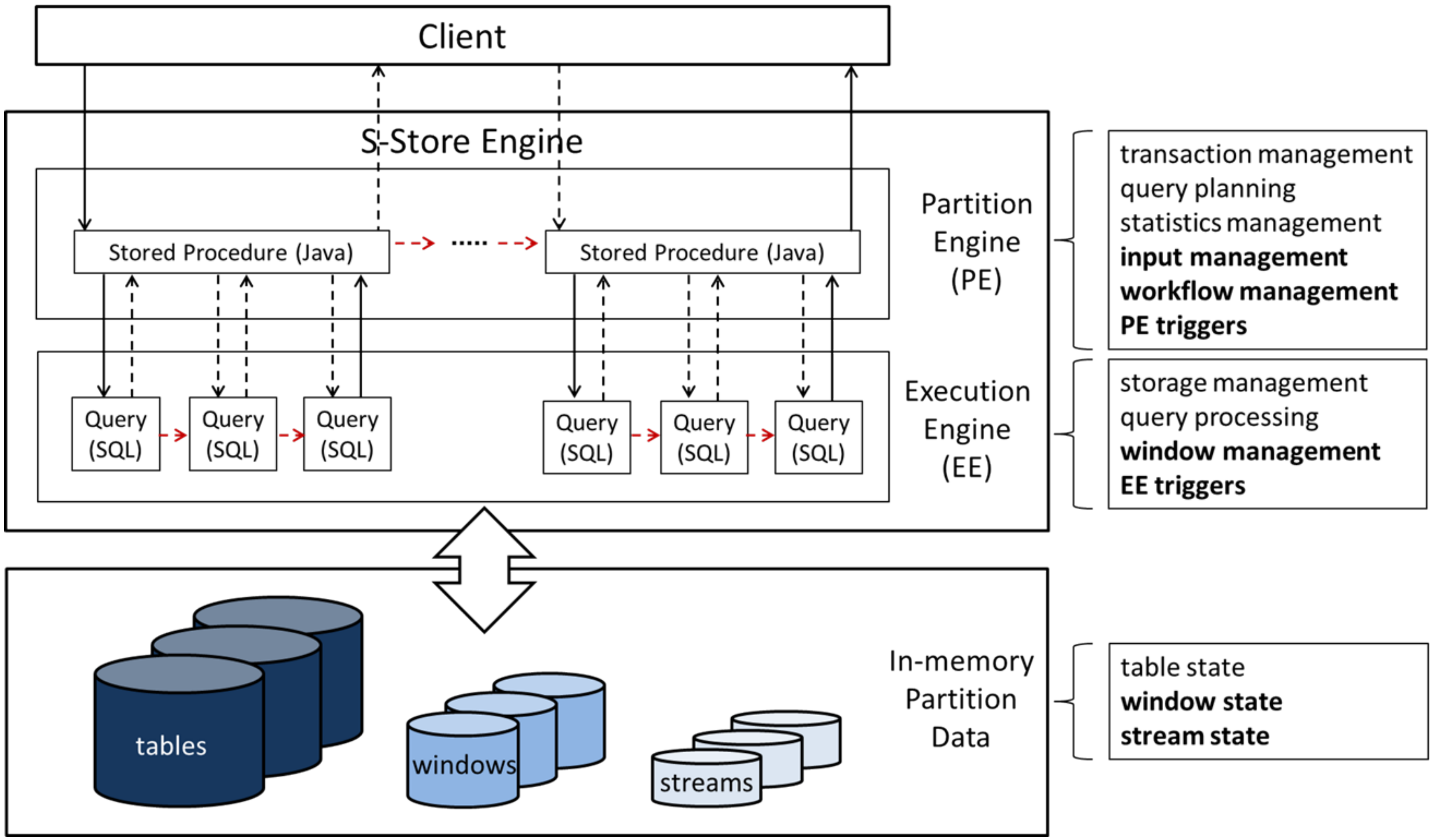
Encore: A Decade Later

S-Store

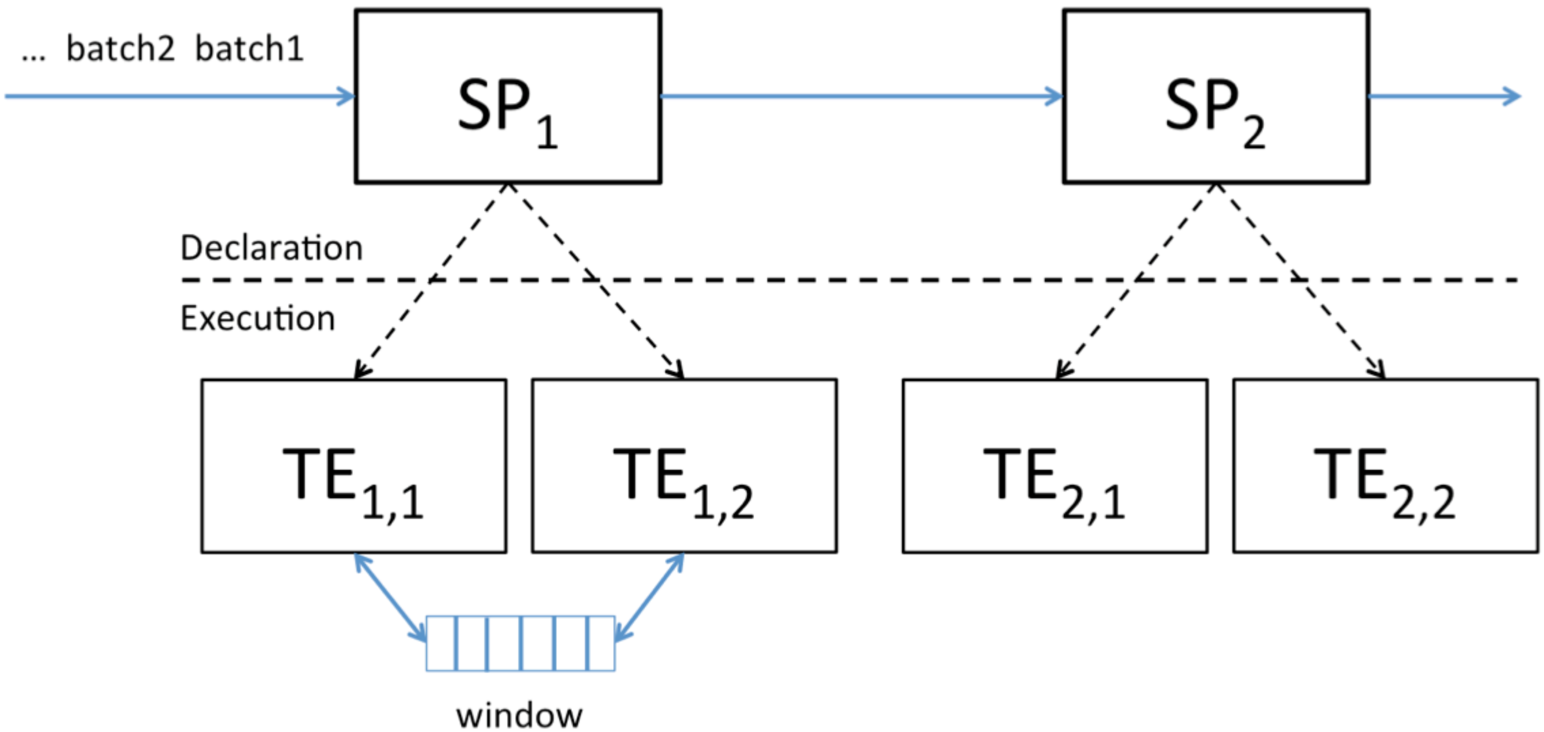
- Streaming system with transactional support
- Built on top of H-Store (!)

- Why build transactions on top of streams?

S-Store Architecture



S-Store Transactions



Conclusion

- Streaming data model
  - Streams + relations
- Stream queries
  - Extensions on top of SQL
- Applications
  - Stocks, real-time apps, update machine learning models
- Issues:
  - ACID
  - Replication