Streaming Data Integration: Challenges and Opportunities

Nesime Tatbul

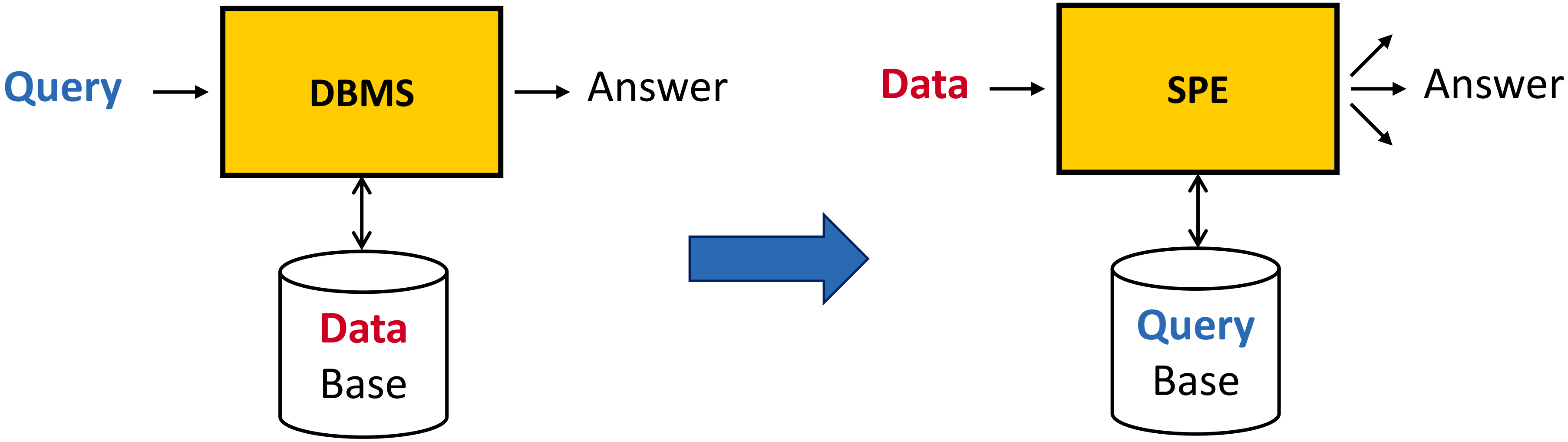
Systems@ETH Zürich

Talk Outline

- Integrated data stream processing
- An example project: MaxStream
    - Architecture
    - Query model
- Conclusions

Data Stream Processing

- **Monitoring applications** require collecting, processing, disseminating, and reacting to real-time events from push-based data sources.
- **"Store and Pull"** model of traditional databases does not work well.



Traditional Database Systems

Stream Processing Engines

“Integrated” Data Stream Processing

- Today, integration support for SPEs is needed in three main forms:
  1. across multiple streaming data sources
     - example: news feeds, weather sensors, traffic cameras
  2. over multiple SPEs
     - example: supply-chain management
  3. between SPEs and traditional DBMSs
     - example: operational business intelligence

#1: Streaming Data Source Integration

- **Goal: Integrated querying over multiple, potentially heterogeneous streaming data sources**
- **Challenges:**
  - Schemas of different sources can differ from one another and from the input schemas of the already running CQs.
  - Input sources or the network can introduce imperfections into the stream.
  - Adapters may become a bottleneck.
- **Current state of the art:**
  - Commercial SPEs offer a collection of common adapters and SDKs for developing custom ones.
  - Mapping Data to Queries [Hentschel et al]
  - ASPEN project [Ives et al]

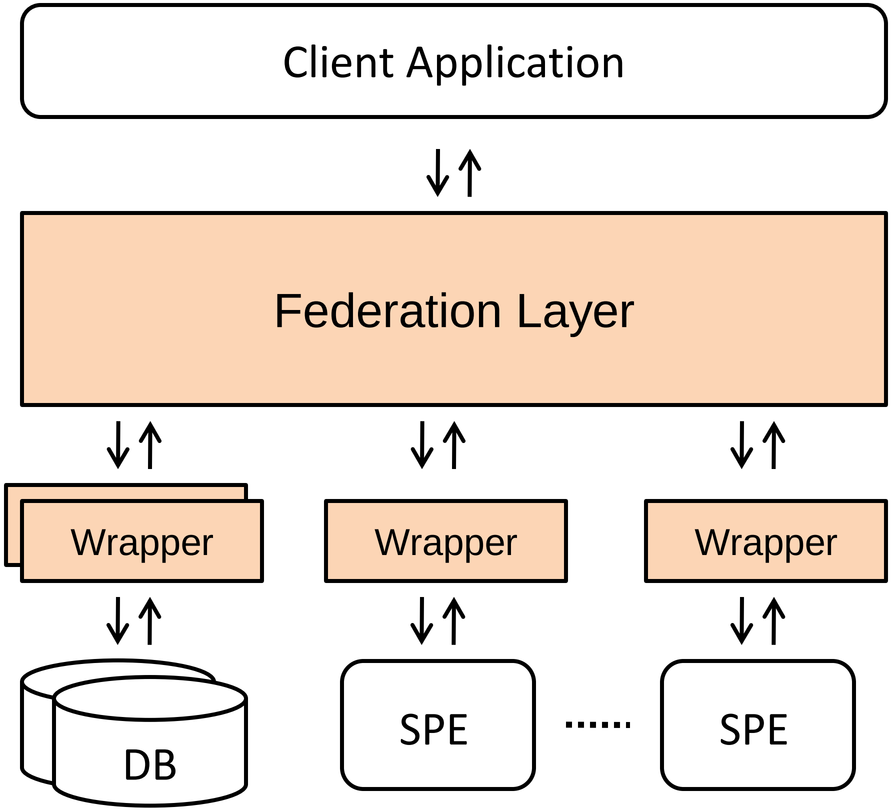
#2: SPE-SPE Integration

- **Goal: Integrated querying over multiple, potentially heterogeneous SPEs**
  - to exploit the advantages of distributed operation
  - to exploit specialized capabilities and strengths of SPEs
  - to provide higher-level monitoring over large-scale enterprises with loosely-coupled operational units
- **Challenges:**
  - The need for functional integration
  - The need to deal with heterogeneity at different levels (e.g., query models, capabilities, performance, interfaces)
- **Current state of the art:**
  - MaxStream project [Tatbul et al]

#3: SPE-DBMS Integration

- **Goal: Integrated querying over SPEs and traditional database systems**
- **Challenges:**
  - Bridge the “data vs. operation” gap between the two worlds.
  - Find the right language and architecture primitives for the required level of querying, persistence, and performance.
- **Current state of the art:**
  - Languages [STREAM CQL, StreamSQL]
  - Architectures
    - SPE-based [typical SPEs such as Coral8, StreamBase]
    - DBMS-based [“stream-relational” systems such as TelegraphCQ/Truviso, DataCell, DejaVu, MaxStream]

MaxStream: A Platform for SPE-SPE and SPE-DBMS Integration



- Key design ideas:
  - Uniform query language and API
  - Relational database infrastructure as the basis for the federation layer (in our case: SAP MaxDB and SAP MaxDB Federator)
  - “Just enough” streaming capability inside the federation layer

MaxStream vs. Traditional Virtual Integration

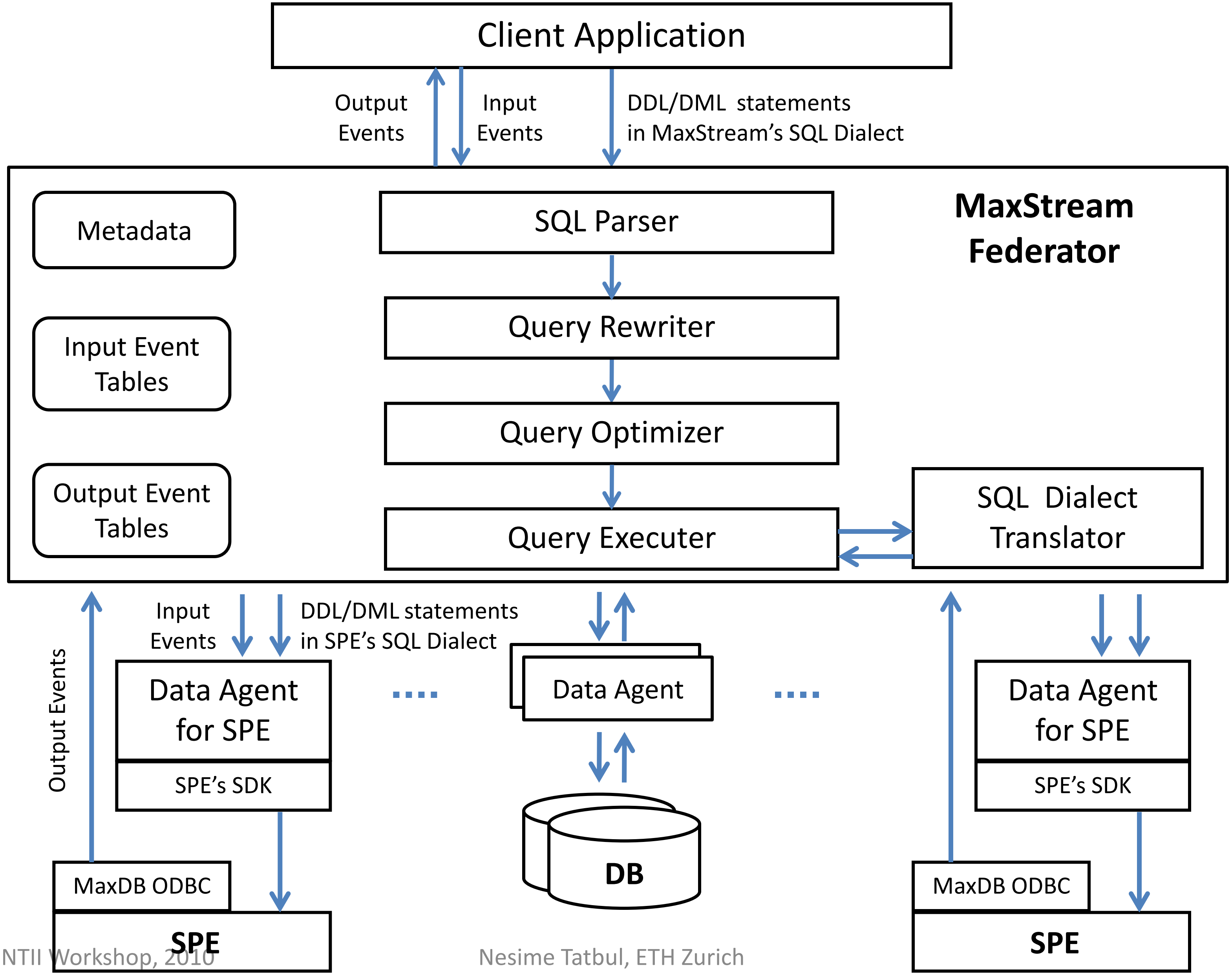
Traditional Virtual Integration

- Goal: to query across multiple autonomous & heterogeneous **data sources**
- Source wrappers send **queries** and receive answers
- Sources host the **data**
- Mapping between **source schemas and a global schema**
- **Queries are posed** against the global schema
- More focus on **data locality**

MaxStream

- Goal: to query across multiple autonomous & heterogeneous **SPEs (and DBMSs)**
- SPE wrappers send **queries and data**, and receive answers
- SPEs host the **CQs**
- Mapping between **SPE CQ models and a global CQ model**
- **CQs are posed and data is fed** against the global CQ model
- More focus on **functional heterogeneity**

MaxStream Architecture



MaxStream Architecture

Two Key Building Blocks

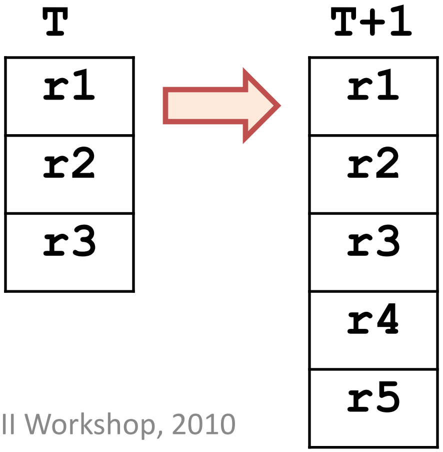
- Streaming inputs through MaxStream
  - **ISTREAM operator** for persistent input events
  - Tuple queues for transient input events

- Streaming outputs through MaxStream
  - **Monitoring Select operator** over event tables
    - Persistent event tables for persistent output events
    - In-memory event tables for transient output events

Streaming Input Events

- The ISTREAM (“Insert STREAM”) Operator
  - “Inspired” by the relation-to-stream operator of the same name in the STREAM Project, that streams new tuples being inserted into a given relation.
  - Example: OrdersTable(ClientId, OrderId, ProductId, Quantity)

```
INSERT INTO STREAM OrdersStream
       SELECT ClientId, OrderId, ProductId, Quantity
       FROM ISTREAM(OrdersTable);
```



T

r1
r2
r3

T+1

r1
r2
r3
r4
r5

`ISTREAM(OrdersTable)` at `T+1` returns:

`<r4, T+1>, <r5, T+1>`

Streaming Output Events

- Opposite of streaming input events, but…
  - Unlike the SPE interface, the client application interface is not push-based.
- The Monitoring Select Operator
  - Select operation blocks until there is at least one row to return.
  - For continuous monitoring, the client program re-issues Monitoring Select in a loop.
  - Monitoring Select operates on “event tables”.
- Example: Detect unusually large order volumes.

```
SELECT *
FROM /*+ EVENT */ TotalSalesTable
WHERE TotalSales > 500000;
```

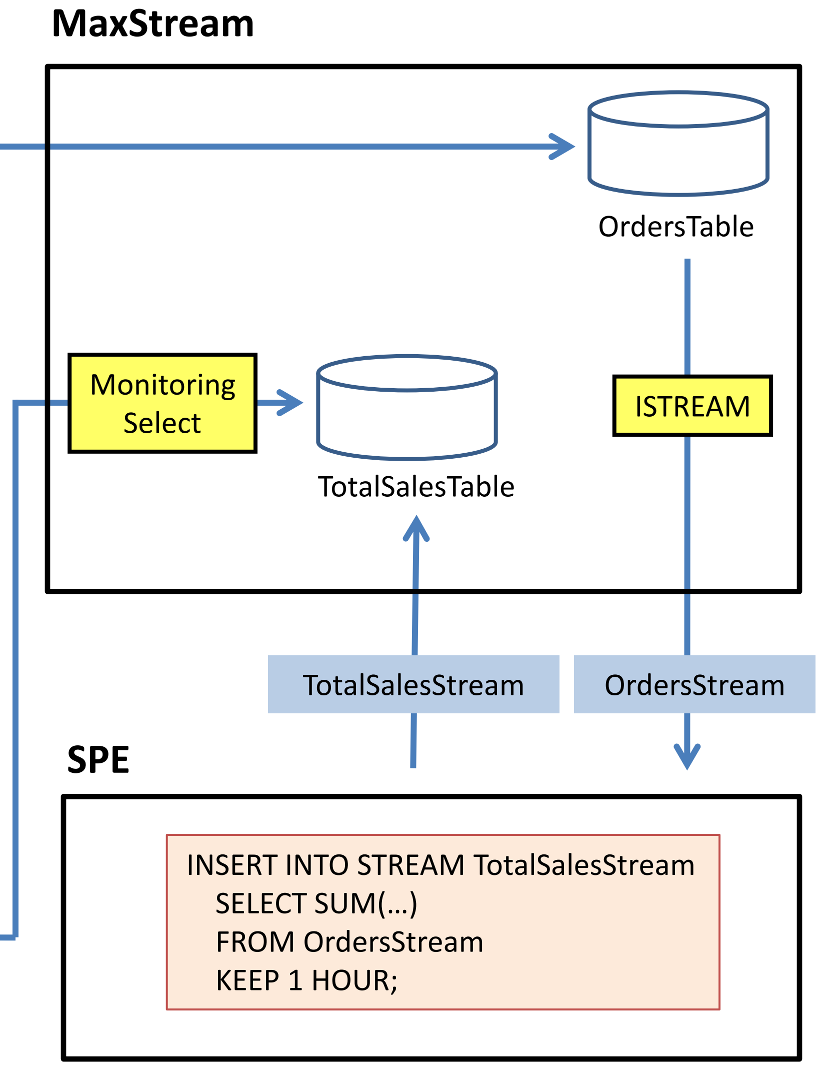
ISTREAM and Monitoring Select in Action

Data Feeder Client

```
// Continuous Insert into OrdersTable kept in MaxStream
CREATE TABLE OrdersTable;
WHILE (true) {
   INSERT INTO OrdersTable VALUES (…);
   sleep(period);
};
// Continuous Insert into OrdersStream kept in the SPE
CREATE STREAM OrdersStream;
INSERT INTO STREAM OrdersStream
   SELECT …
   FROM ISTREAM(OrdersTable);
```

Monitoring Clients

```
// Setting up the Continuous Query to push down to the SPE
INSERT INTO STREAM TotalSalesStream
   SELECT SUM(…)
   FROM OrdersStream
   KEEP 1 HOUR;
// Streaming Output Events inserted by the SPE
CREATE TABLE TotalSalesTable;
// Sales spikes Query to run in MaxStream
WHILE (true) {
   SELECT *
   FROM /*+EVENT*/ TotalSalesTable
   WHERE TotalSales > 500000;
};
```



```
INSERT INTO STREAM TotalSalesStream
   SELECT SUM(…)
   FROM OrdersStream
   KEEP 1 HOUR;
```

Hybrid Queries in MaxStream

- Hybrid queries are continuous queries that join Streams with Tables.
- Two important factors that affect efficiency:
  - The streaming data source must be first in the join ordering.
  - Hybrid queries can be rewritten to perform the join within the MaxStream Federator, removing the need for the SPE to establish connections to external databases.
- One can conveniently use hybrid queries in MaxStream in two ways:
  - To enrich the input stream before it is passed to the SPE
  - To enrich the output stream after it is received from the SPE

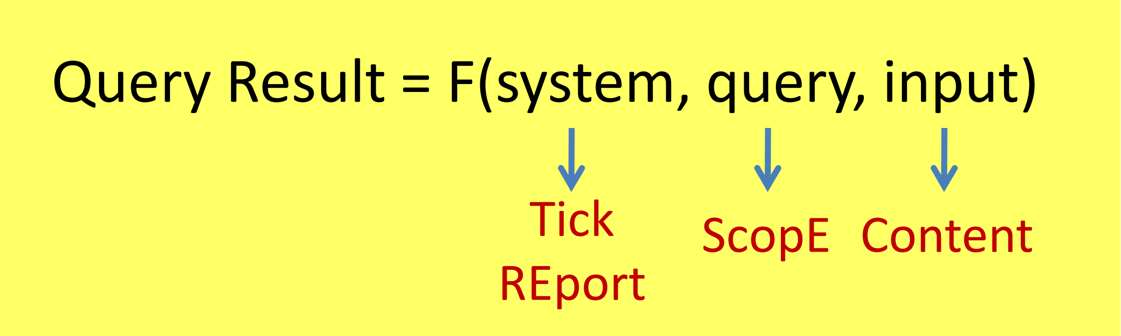
MaxStream Query Model

- Problem: Heterogeneity of SPE query models
  - Syntax heterogeneity
    - Language clauses/keywords for common constructs syntactically differ.
  - Capability heterogeneity
    - Support for certain query types differs.
  - Execution model heterogeneity
    - Underlying query execution models differ.
    - ➢ Not exposed to the application developer at the language syntax level.

- First step towards a solution: Create a model to analyze and predict the query execution semantics of SPEs.

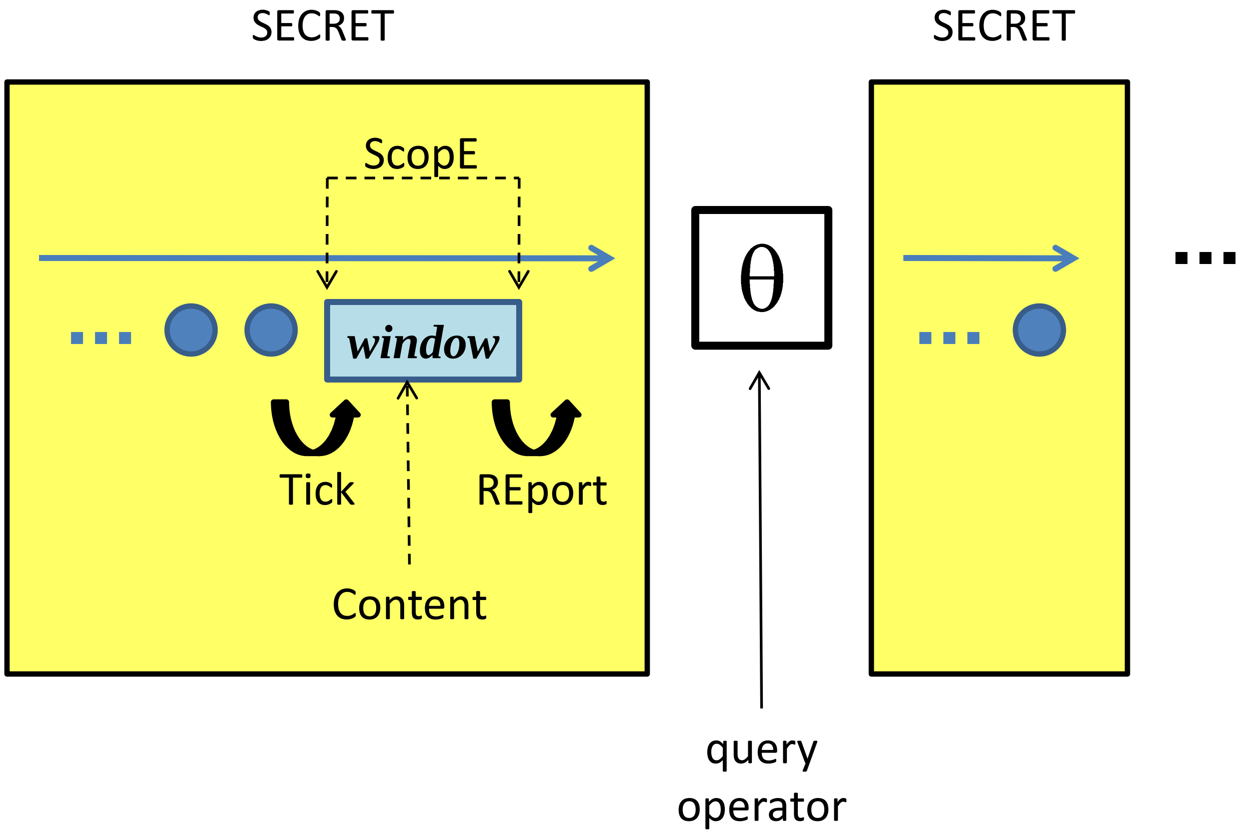
The SECRET Model

- What affects query results produced by an SPE?
  - **ScopE:** Given a query with certain window properties, what are the potential ***window intervals***?
  - Content: Given an input stream, what are the ***actual contents*** for those window intervals?
  - **RE**port: Under what conditions, do those window contents become ***visible*** to the query processor for evaluation?
  - Tick: What drives an SPE to ***take action*** on a given input stream?



Query Result = F(system, query, input)

The SECRET of a Query Plan



SECRET

ScopE

window

Tick

REport

Content

θ

query operator

SECRET

…

The SECRET of an SPE

- Tick:
  - – tuple-driven (e.g., Aurora, Borealis, StreamBase, TelegraphCQ, Truviso)
  - – time-driven (e.g., STREAM, Oracle CEP)
  - – batch-driven (e.g., Coral8, [Jain et al, VLDB'08])
- REport:
  - – window close & non-empty (e.g., StreamBase)
  - – content change & non-empty (e.g., Coral8)
  - – window close & content change & non-empty (e.g., STREAM)

MaxStream: Future Outlook

- Query model
  - how to extend SECRET (other query types, analysis of other SPEs, input imperfections)
  - how to use SECRET in MaxStream (→ SECRET-based query and SPE capability analyzer)
- Capability- and Cost-based query optimization
- Transactional stream processing
- Distributed operation

Conclusions

- Today, integration support for SPEs is needed in three main forms: across sources, SPE-SPE, SPE-DBMS.
- There are many open research challenges.
- MaxStream takes up some of these challenges for SPE-SPE and SPE-DBMS integration.
- More information about MaxStream:

  http://www.systems.ethz.ch/research/projects/maxstream/