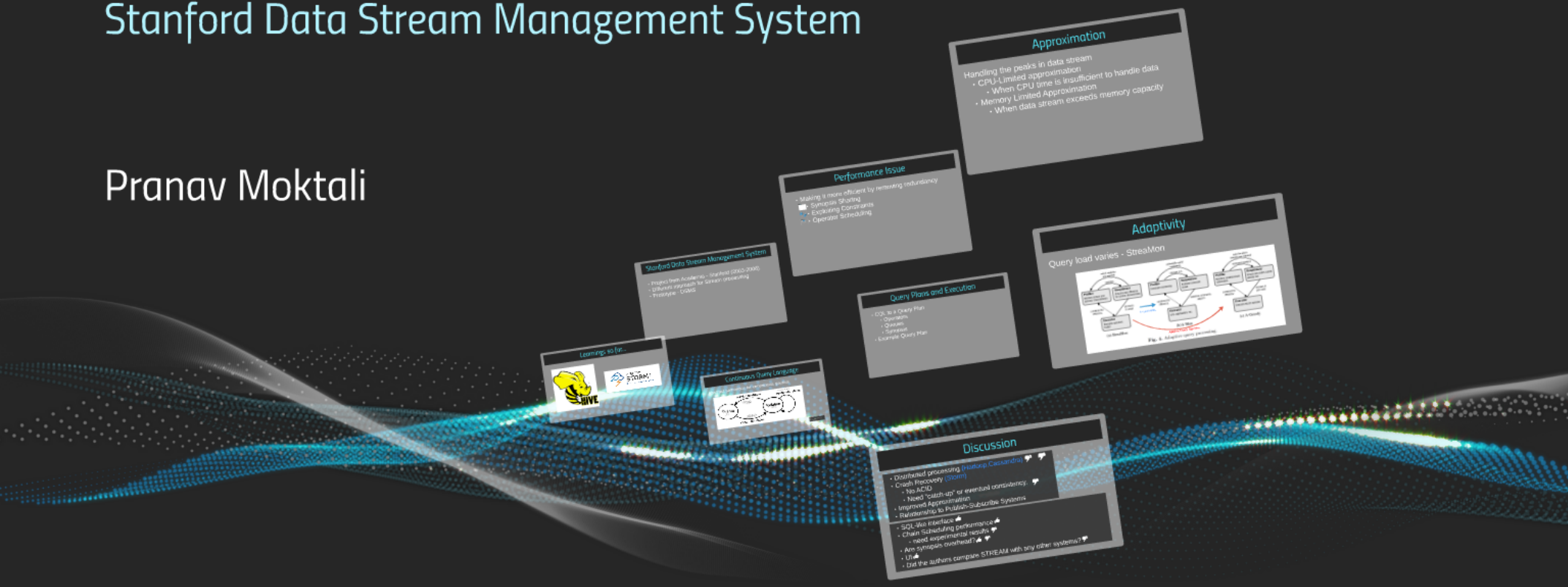
STREAM

Stanford Data Stream Management System

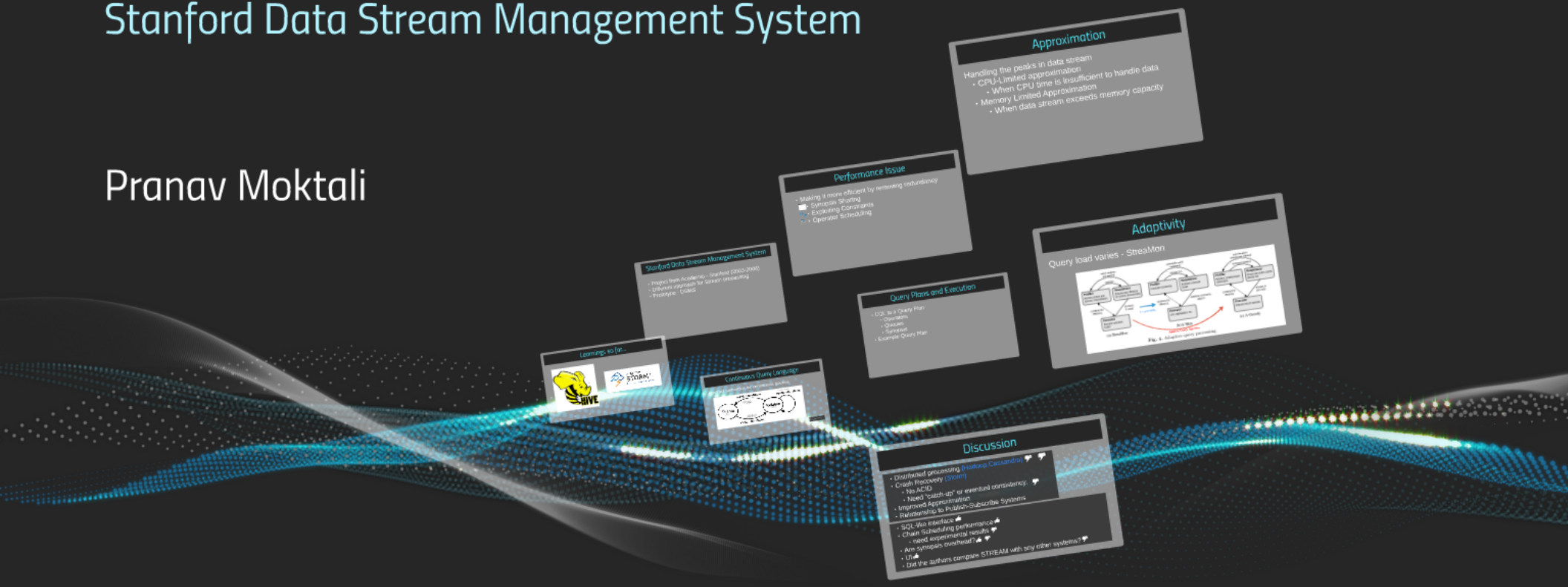
Pranav Moktali

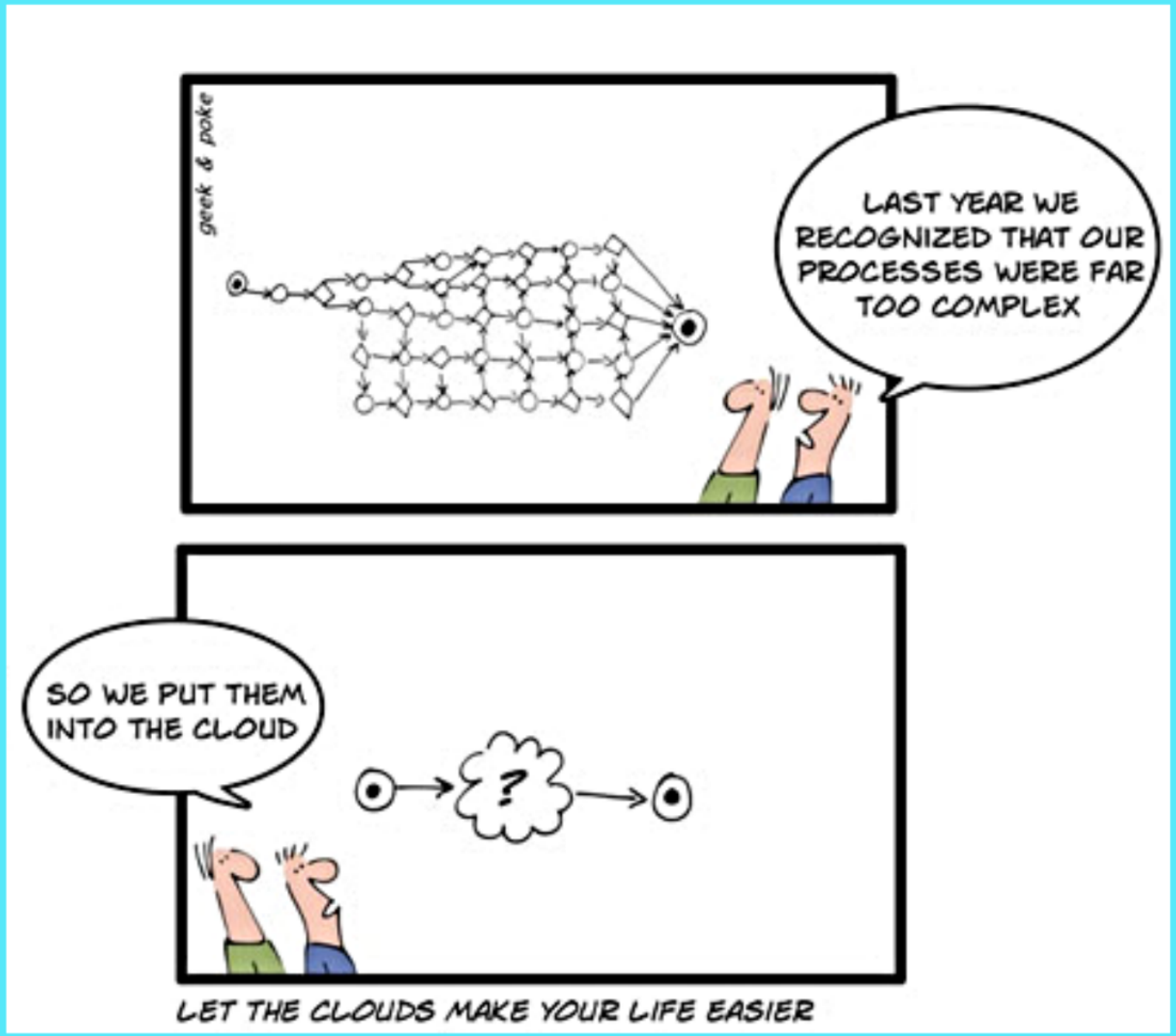


STREAM

Stanford Data Stream Management System

Pranav Moktali




geek & poke
LAST YEAR WE RECOGNIZED THAT OUR PROCESSES WERE FAR TOO COMPLEX
SO WE PUT THEM INTO THE CLOUD
?
LET THE CLOUDS MAKE YOUR LIFE EASIER

Learnings so far...

Hive

- Abstraction over Hadoop.
- Supports SQL-like queries.
- Runs Map-Reduces jobs from declarative statements.

Here comes continuous stream of data!

Apache Storm

- Stream processing system
- Spouts and Bolt
- A record is a tuple of <key, value(s)>

What about to SQL-like query?

Stanford Data Stream Management System

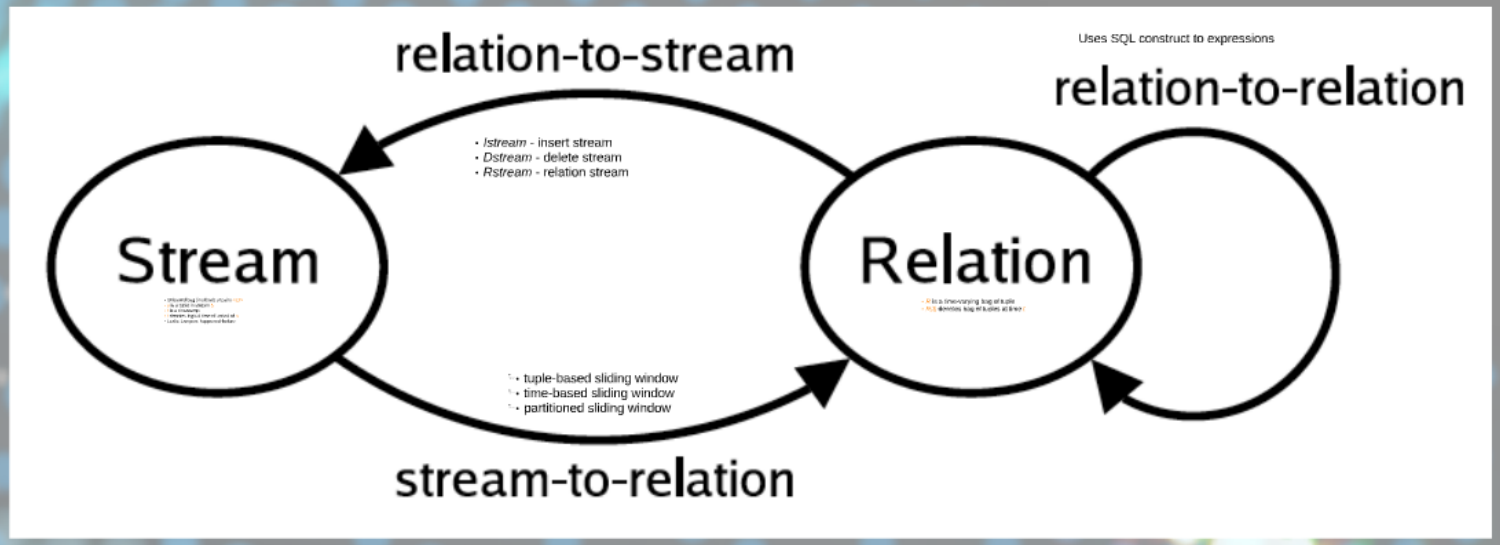
- Project from Academia - Stanford (2003-2006)
- Different approach for Stream processing
- Prototype - DSMS

Data Stream Management System

- CQL: Continuous Query Language
- CQL Under the hood
- Optimize
- Handle high volume
- User Interface to query and visualize

Continuous Query Language

Abstract semantics for continuous queries



Examples

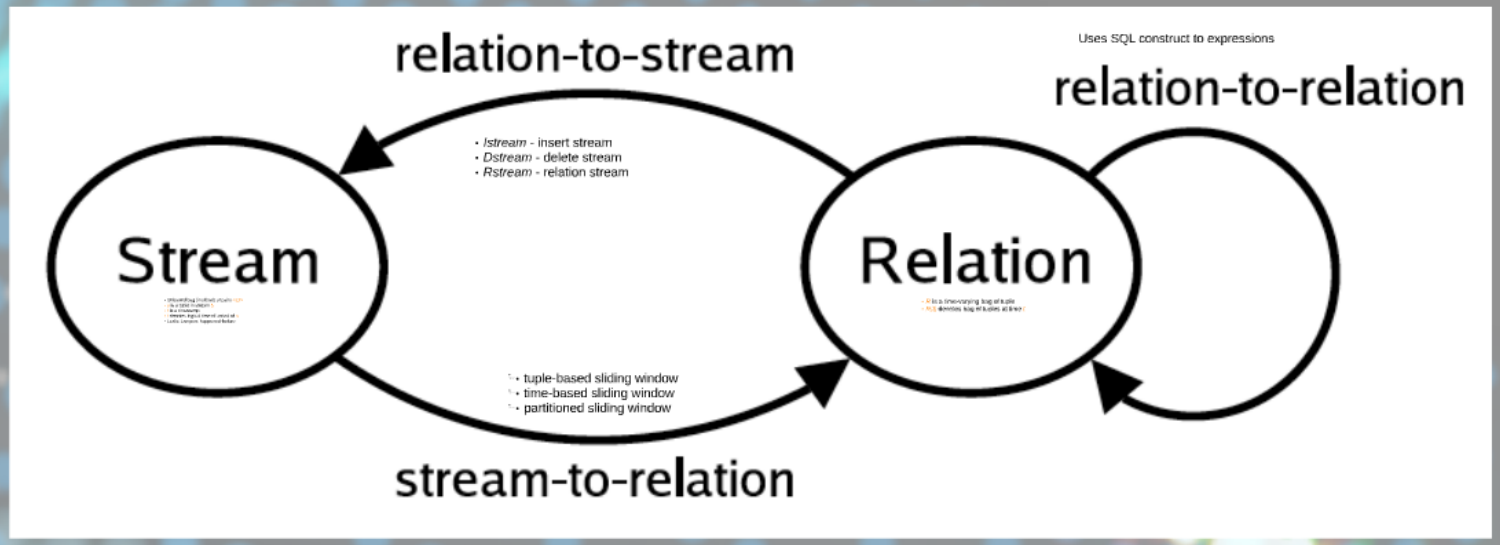
```
Select Istream(*) From S [Rows Unbounded] Where S.A > 10

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

- Unbound bag (multiset) of pairs <*s*,*t*>
- *s* is a tuple in stream *S*
- *t* is a timestamp
- *t* denotes logical time of arrival of *s*
- Leslie Lamport: happened-before

- *R* is a time-varying bag of tuple
- *R(t)* denotes bag of tuples at time *t*

Continuous Query Language

Abstract semantics for continuous queries



Examples

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

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

queries

Uses SQL construct to expressions

relation-to-relation

- tuple-based sliding window
- time-based sliding window
- partitioned sliding window

stream-to-relation

tuple-based

- Input:
  - int *N >0* at t
- Output:
  - *R(t)*, N tuples with timestamp <= t
- Usage
  - [Rows N]
  - [Rows Unbounded]

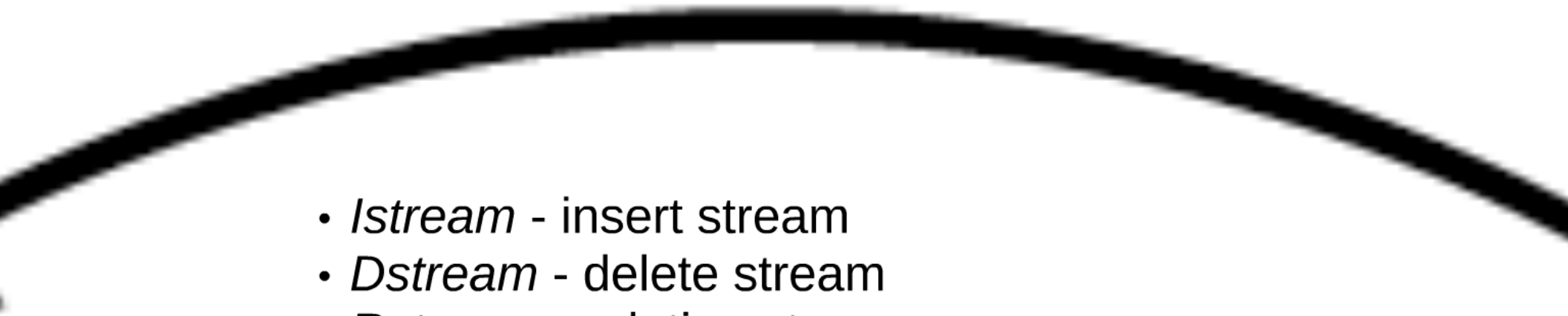
time-based

- Input:
  - timestamp *w*
- Output:
  - *R(t)*, tuples between *t-w* and *t*
- [Range w]
- [Now] (w=0)

partitioned

- Input:
  - int *N* and set *{A1,...,Ak}*
- Output:
  - *R(t)*, union of substreams by attr of size N
- [Partition By A1,...,Ak Rows N]

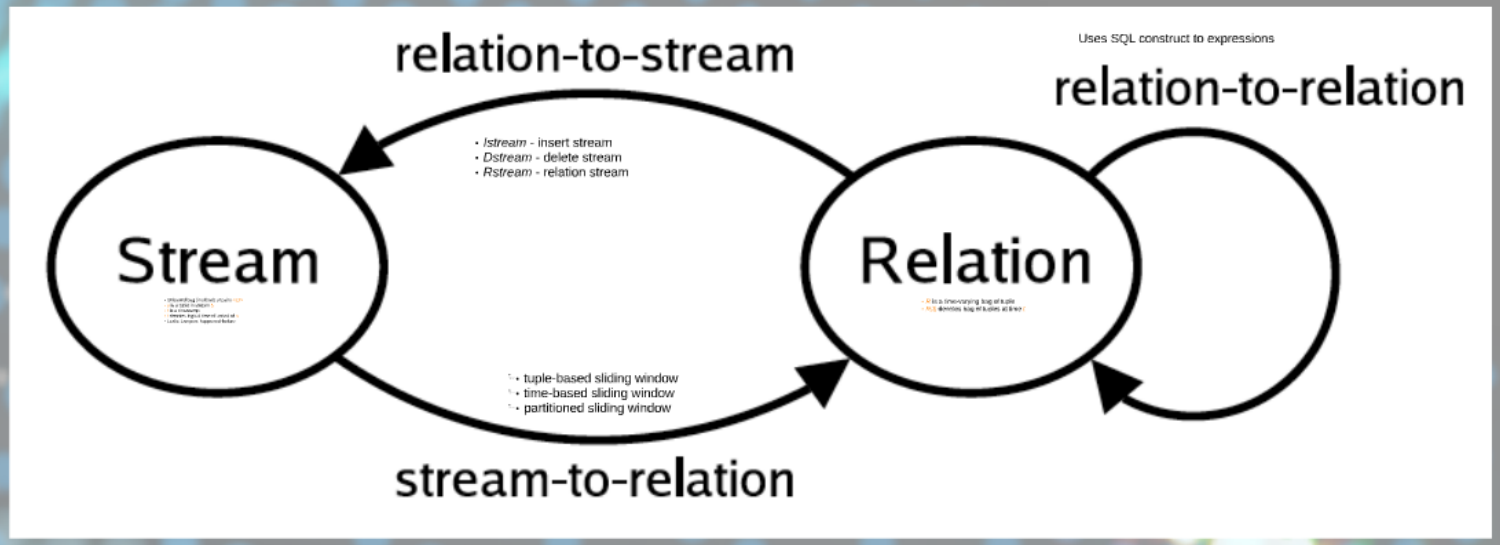
relation-to-stream



- *Istream* - insert stream
- *Dstream* - delete stream
- *Rstream* - relation stream

Continuous Query Language

Abstract semantics for continuous queries



Examples

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

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

Examples

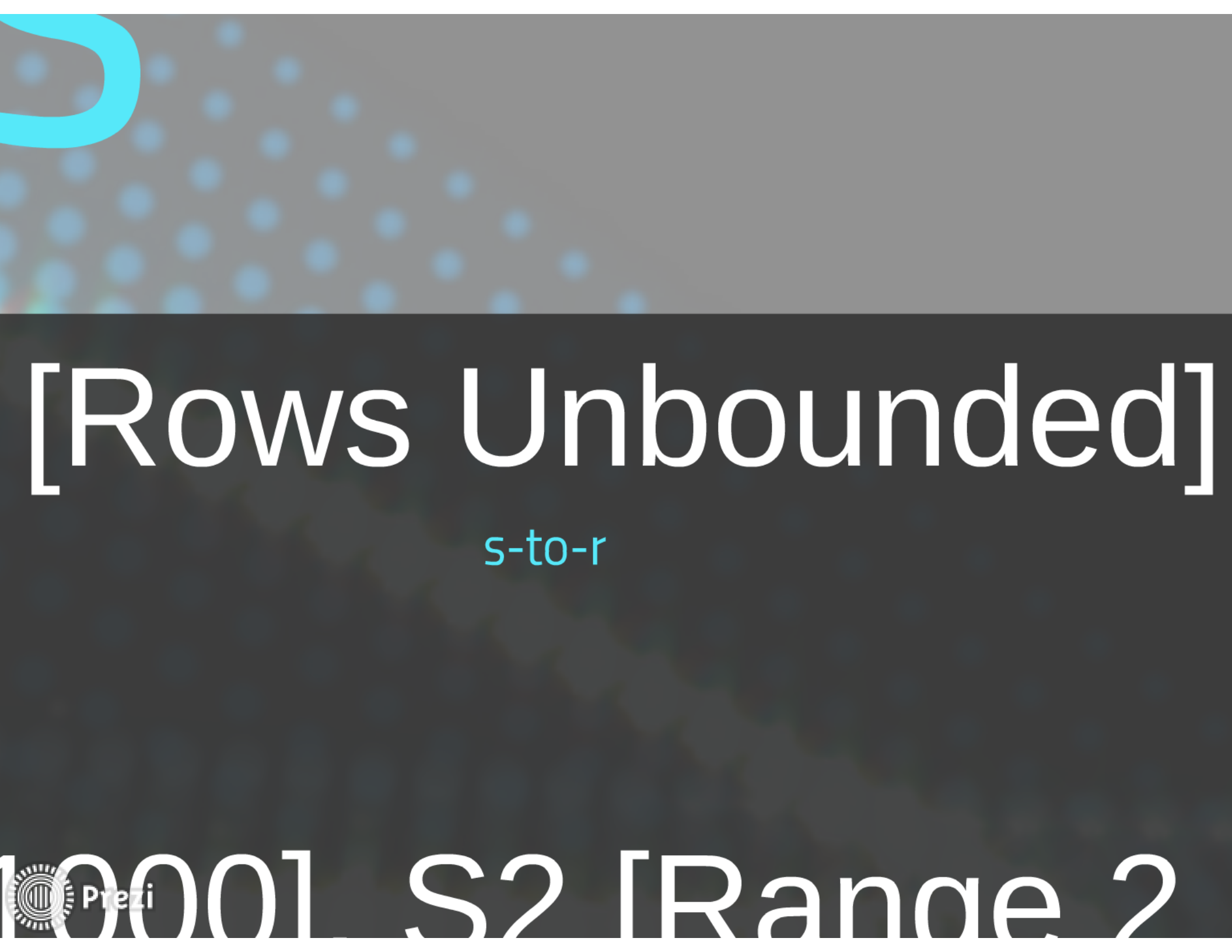
Select Istream(*) From S [Rows Unbounded] Where S.A > 10

r-to-s s-to-r r-to-r

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

lstream(*)

r-to-s


[Rows Unbounded]
s-to-r
1000] S2 [Range 2
Prezi

S.A > 10

r-to-r

Examples

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

r-to-s s-to-r r-to-r

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

Query Plans and Execution

- CQL to a Query Plan
  - Operators
  - Queues
  - Synopsis
- Example Query Plan

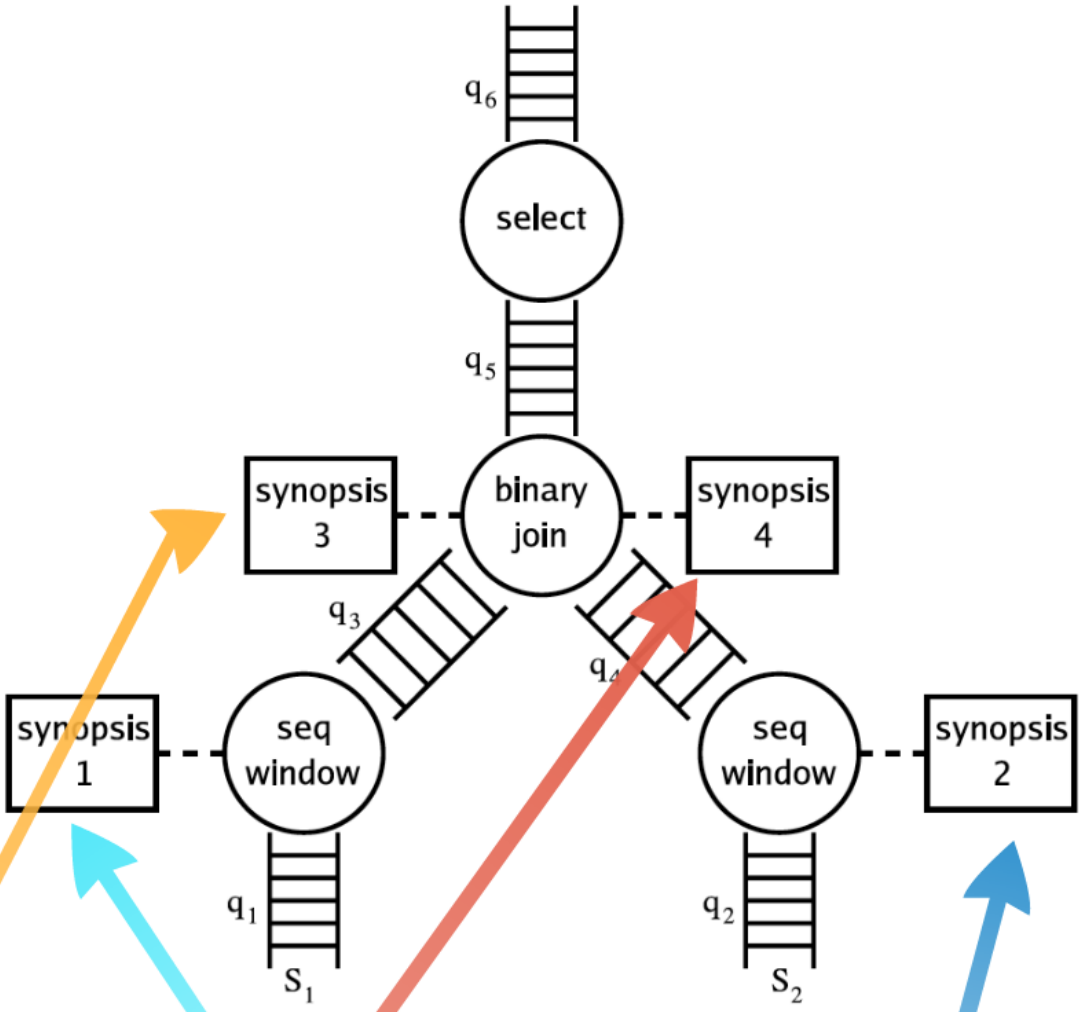
Name	Operator Type	Description
`select`	relation-to-relation	Filters elements based on predicate(s)
`project`	relation-to-relation	Duplicate-preserving projection
`binary-join`	relation-to-relation	Joins two input relations
`mjoin`	relation-to-relation	Multiway join from [22]
`union`	relation-to-relation	Bag union
`except`	relation-to-relation	Bag difference
`intersect`	relation-to-relation	Bag intersection
`antisemijoin`	relation-to-relation	Antisemijoin of two input relations
`aggregate`	relation-to-relation	Performs grouping and aggregation
`duplicate-eliminate`	relation-to-relation	Performs duplicate elimination
`seq-window`	stream-to-relation	Implements time-based, tuple-based, and partitioned windows
`i-stream`	relation-to-stream	Implements *Istream* semantics
`d-stream`	relation-to-stream	Implements *Dstream* semantics
`r-stream`	relation-to-stream	Implements *Rstream* semantics

Table 1. Operators used in STREAM query plans.

Query Plans and Execution

- CQL to a Query Plan
  - Operators
  - Queues
  - Synopsis
- Example Query Plan

Example



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

Performance Issue

- Making it more efficient by removing redundancy
  -  Synopsis Sharing
  -  Exploiting Constraints
  -  Operator Scheduling

Synopsis Sharing

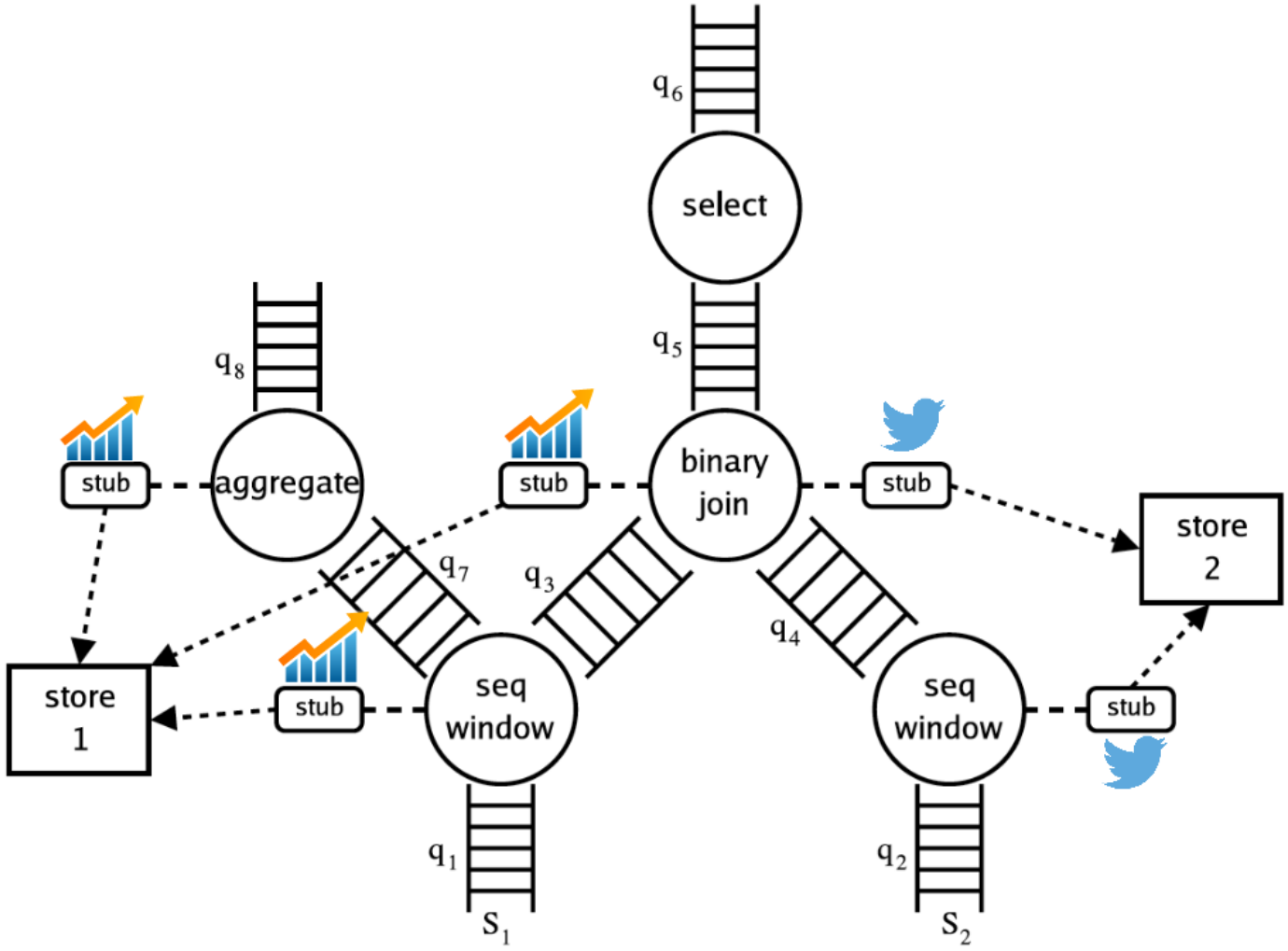
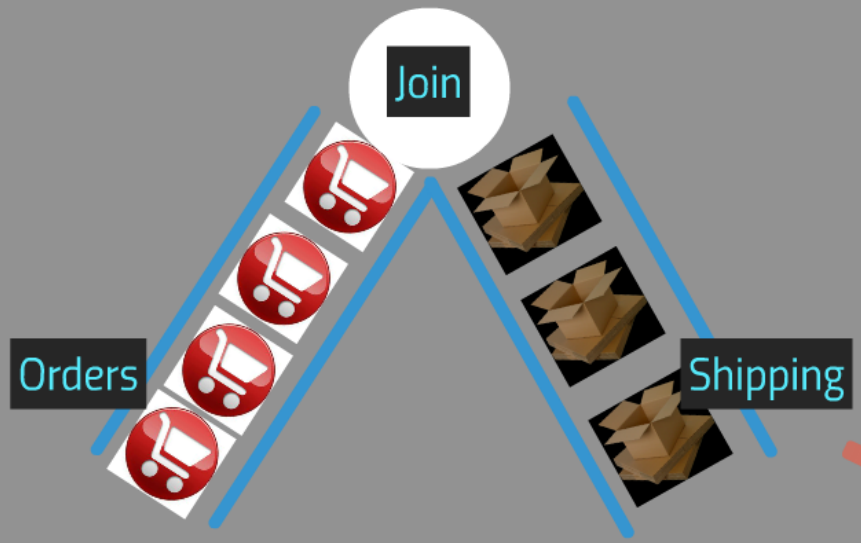
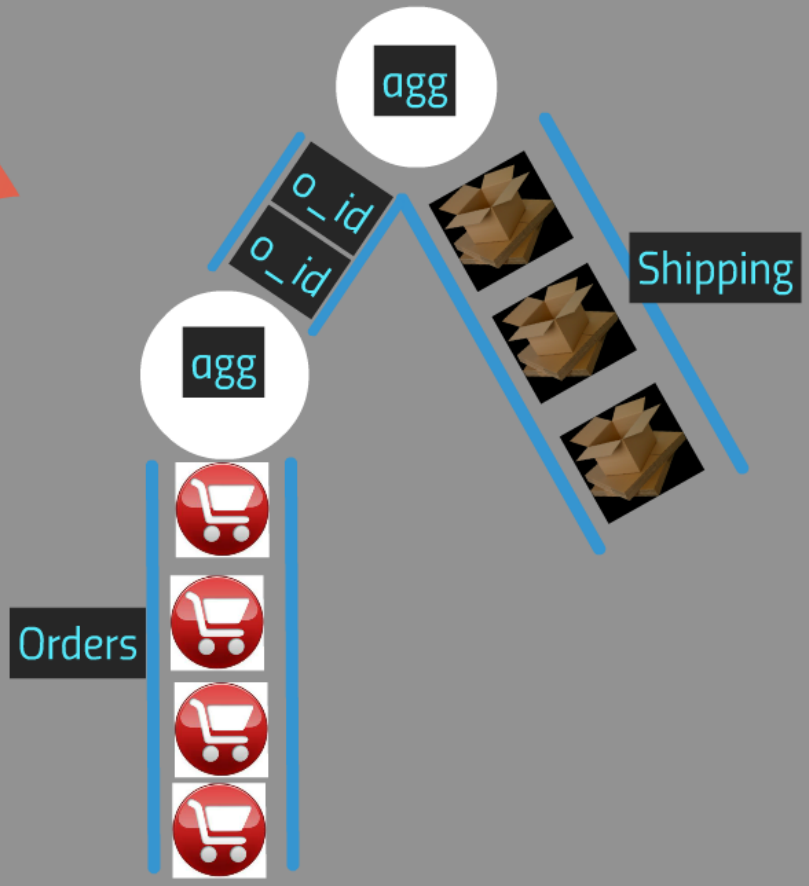


Fig. 3. A query plan illustrating synopsis sharing.

Exploiting Constraints



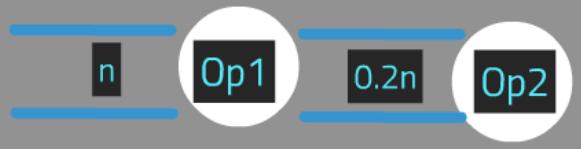


- k-constraint
  - referential integrity
  - ordered-arrival
  - clustured-arrival

Operator Scheduling

Possible schedulers

- FIFO
- Greedy



Introducing Chain Scheduling

- Greedy chain creation
- FIFO operations in chain

Adaptivity

Query load varies - StreaMon

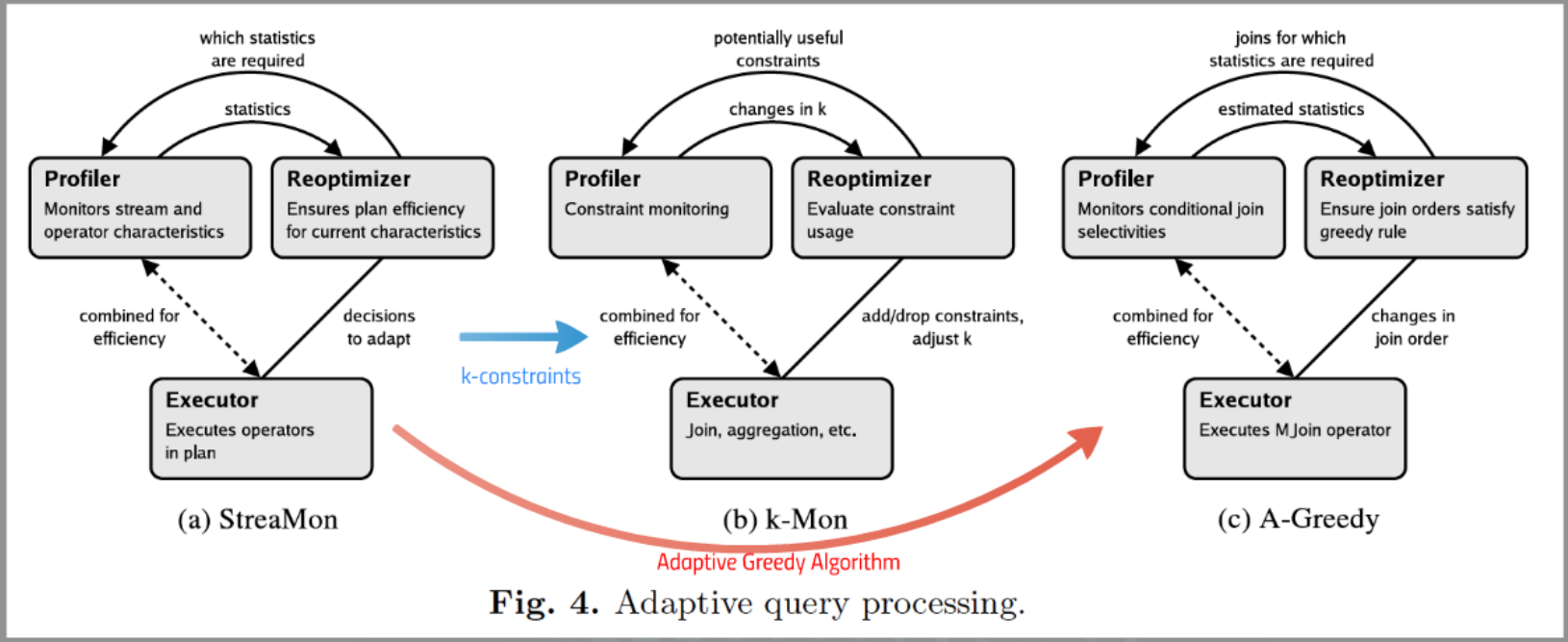


Fig. 4. Adaptive query processing.

Approximation

Handling the peaks in data stream

- CPU-Limited approximation
  - When CPU time is insufficient to handle data
- Memory Limited Approximation
  - When data stream exceeds memory capacity

Discussion

- Distributed processing (Hadoop,Cassandra) 👎 👎
- Crash Recovery (Storm)
  - No ACID
  - Need "catch-up" or eventual consistency. 👎
- Improved Approximation
- Relationship to Publish-Subscribe Systems

- SQL-like interface 👍
- Chain Scheduling performance 👍
  - need experimental results 👎
- Are synopsis overhead? 👍 👎
- UI 👍
- Did the authors compare STREAM with any other systems? 👎

STREAM

Stanford Data Stream Management System

Pranav Moktali