Tutorial: An Outlook to Declarative Languages for Big Streaming Data

DEBS 2019, Darmstadt, Germany, Europe, Earth,
Solar System, Milky Way, Universe

Riccardo Tommasini, Sherif Sakr,
Marco Balduini, and Emanuele Della Valle

Who we are

Agenda

- Introduction on Stream Processing Models
- Declarative Language: Opportunities, and Design Principles
- Comparison of Prominent Streaming SQL Dialects for Big Stream Processing Systems
- Conclusion

Unbounded yet time-ordered sequence of data

Jennifer Widom

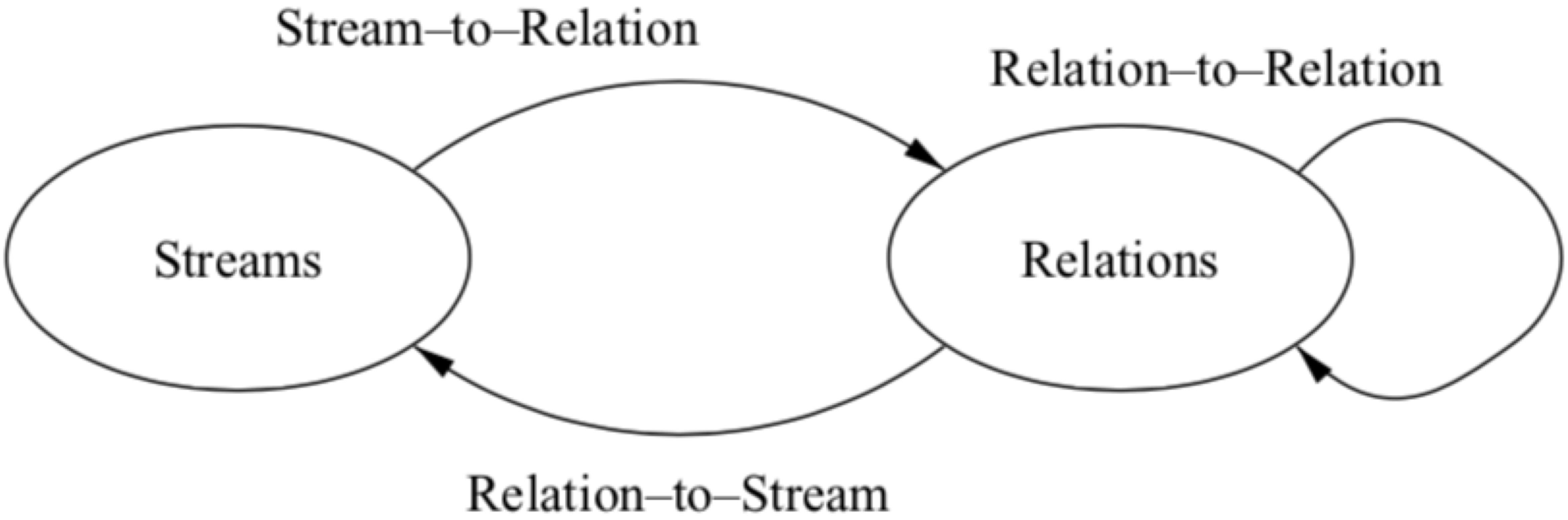
Stream Processing 101

- Data Stream Management Systems (DSMS)
  - Found origin within databased community
  - focus on continuous query answering and analytics
  - Reference Models inspired by Relational Algebra: CQL, Secret
- Complex Event Processing (CEP)
  - Found origin within software engineering community
  - focus on continuous detection of patterns
  - Reference Models inspired by regular languages: NFA, SNOOP

Time Management

- **Processing Time** (consumer) implies a total order on the stream.
- **Event Time** (producer), implies a partial order on the data.

DSMS (CQL)


Stream–to–Relation
Relation–to–Relation
Streams
Relations
Relation–to–Stream

DSMS (CQL)

- Expressive Languages (SQL++)
  - Windowing
    - *Canonical*: logical/physical sliding/tumbling
    - *Custom*: session, data-driven, event-driven

DSMS (CQL)

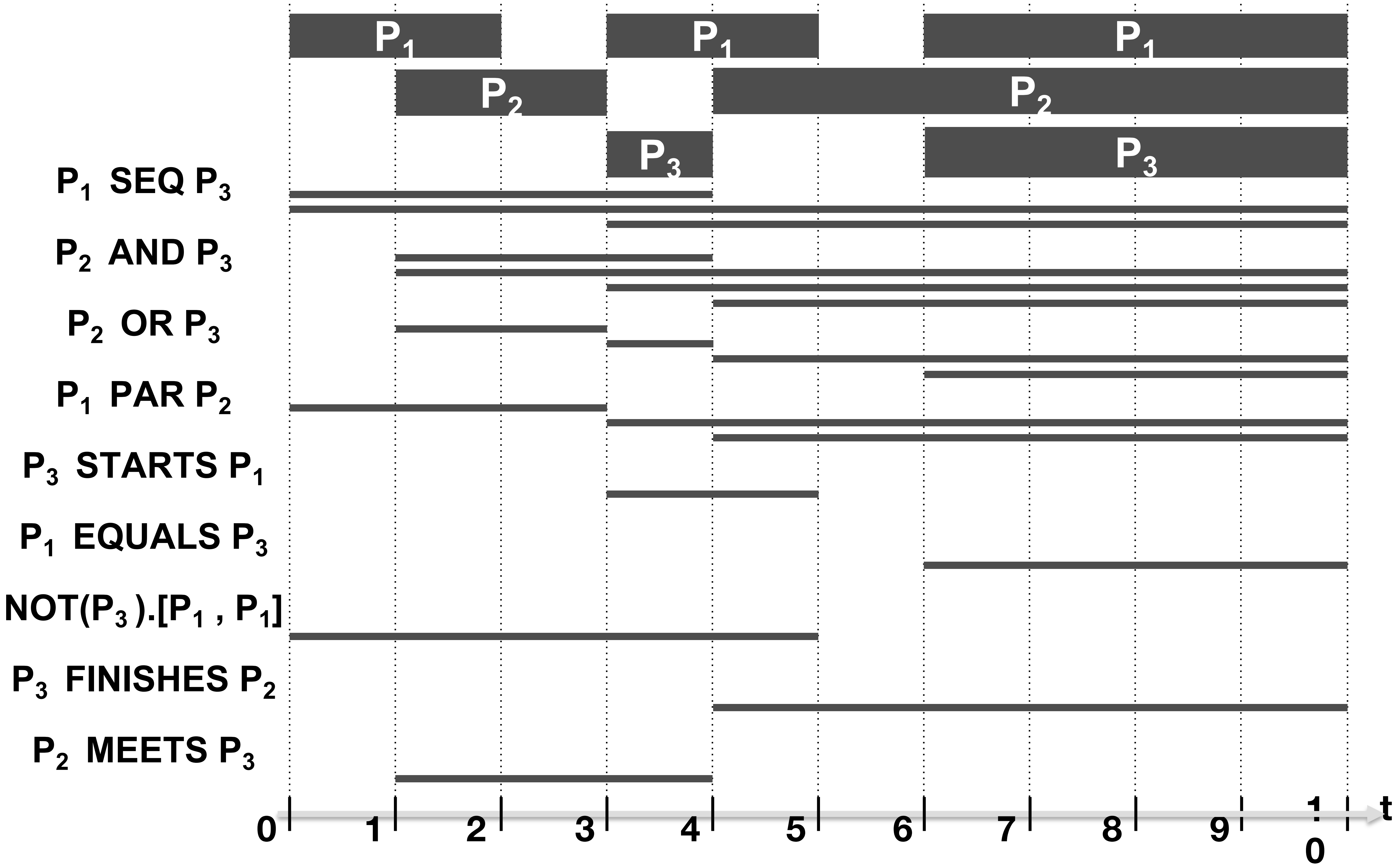
```
CREATE SCHEMA Stock(id string, price float, timestamp int)

SELECT avg(price)
FROM Stock#time(10 minutes)
OUTPUT EVERY 1 minutes
GROUP BY id
```

CEP

- Regular Languages (Declarative)
  - Core from SNOOP (SEQ, AND, OR, NOT, FIRST, LAST)
- Allen’s Algebra
- Finite state machines (non-det)

CEP (SNOOP)


P_1
P_1
P_1
P_2
P_2
P_3
P_3
P_1 SEQ P_3
P_2 AND P_3
P_2 OR P_3
P_1 PAR P_2
P_3 STARTS P_1
P_1 EQUALS P_3
NOT(P_3).[P_1 , P_1]
P_3 FINISHES P_2
P_2 MEETS P_3
0
1
2
3
4
5
6
7
8
9
10
t

CEP (SNOOP)

```
CREATE SCHEMA Selling(from string, to string, price float, ts int)

CREATE SCHEMA Buying(from string, to string, price float, ts int)

CREATE SCHEMA Fraud (sell string, by string)

INSERT INTO Fraud
SELECT a.id, b.id
FROM PATTERN
[EVERY a=Selling -> b=Buying(a.from=b.to, a.price >
price)]#time(10min)
OUTPUT EVERY 1 min
```

CEP (SNOOP)

```
  CREATE SCHEMA AltitudeChange(starts long, ends long, ialt long, falt
long)

  CREATE SCHEMA CruisePeriod(onts long, offts long)

  SELECT *
FROM CruisePeriod#lastevent AS a,
AltitudeChange#lastevent AS b where a.overlaps(b)
```

Big Stream Processing

Ecosystem

Flink
STORM
samza
distributed stream processing
VOLTDB
APACHE
Spark
WSO2

@humbertostreb

Big Stream Processing

Fully-Distributed Systems

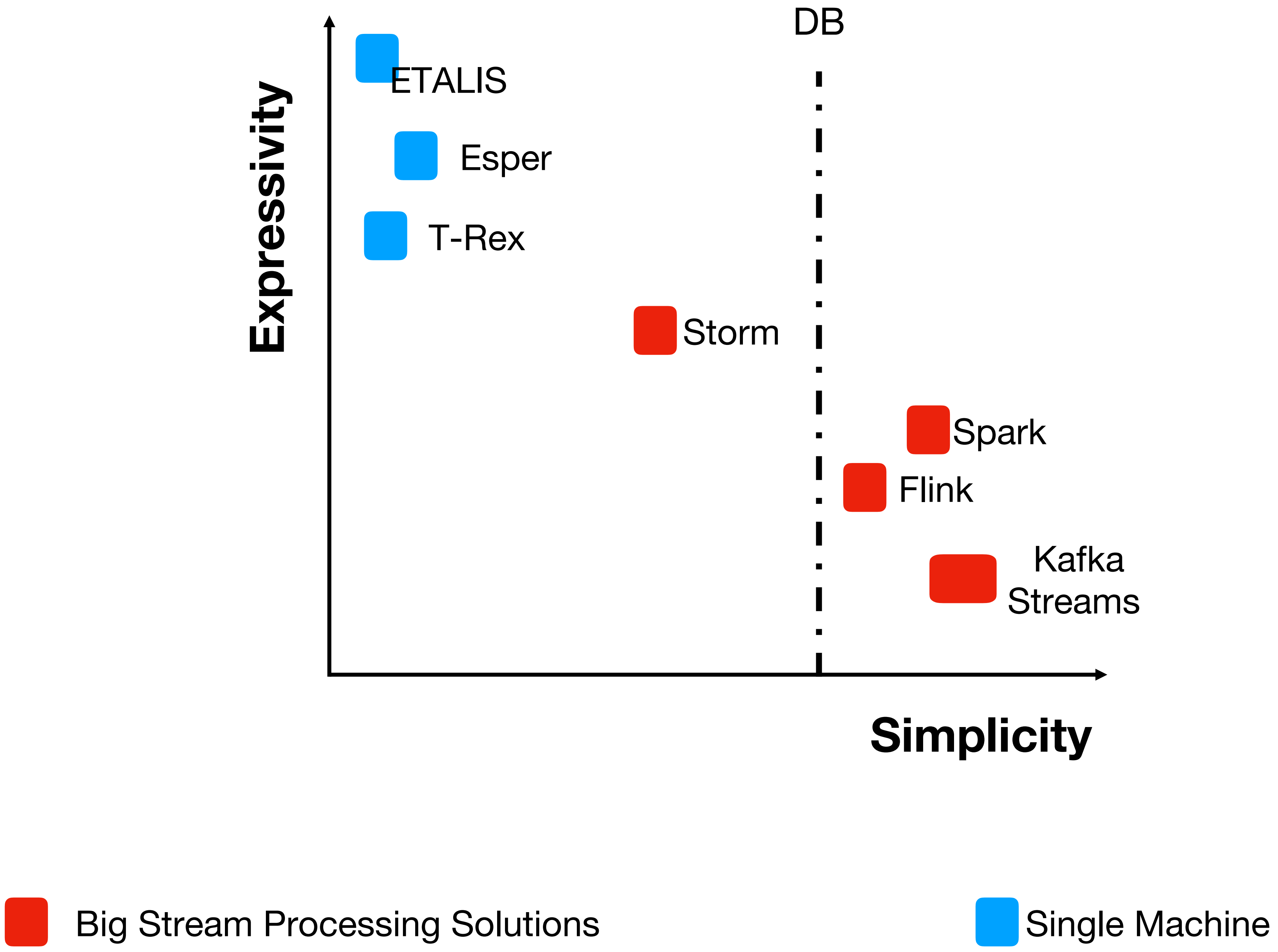
- Fault-Tolerant:
    - **At-Least-Once** or **Exactly-Once** semantics
- Scalable (millions of tuples per minute)
- Flexible Programmatic API that guides towards the creation of Direct Acyclic Graph

Big Stream Processing

Languages

- Offer languages that are embedded in a general-purpose host language, typically Java
- Encourage developers to **explicitly** code a Direct-Acyclic Graph
- Provide relational operators, but also expose low-level details such as partitioning, timestamp extraction

Solution Landscape (qualitative)



Big Stream Processing

(Issues)

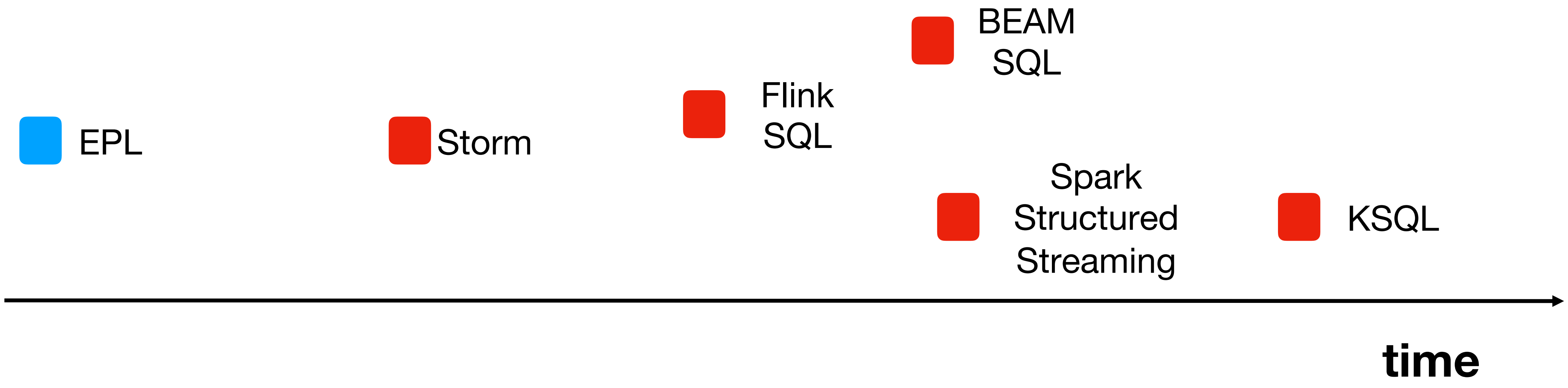
- Distributions makes out-of-order handling a primary problem, and, thus solutions appears in the programmatic APIs.

- Languages are not self-contained, thus, are **hard to isolate** clearly from the host language

- Debugging, benchmarking, and standardisation becomes hard

Major systems started migrating towards a fully-declarative approach ultimately evolved into SQL-like streaming DSL.

Adoption of SQL-like interface



🟦 EPL

🟥 Storm

🟥 Flink SQL

🟥 BEAM SQL

🟥 Spark Structured Streaming

🟥 KSQL

time

🟥 Big Stream Processing Solutions

🟦 Single Machine

One SQL to Rule Them All: An Efficient and Syntactically Idiomatic Approach to Management of Streams and Tables

An Industrial Paper

Edmon Begoli
Oak Ridge National Laboratory / Apache Calcite
Oak Ridge, Tennessee, USA
begoli@apache.org

Tyler Akidau
Google Inc. / Apache Beam
Seattle, WA, USA
takidau@apache.org

Fabian Hueske
Ververica / Apache Flink
Berlin, Germany
fhueske@apache.org

Julian Hyde
Looker Inc. / Apache Calcite
San Francisco, California, USA
jhyde@apache.org

Kathryn Knight
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
knightke@ornl.gov

Kenneth Knowles
Google Inc. / Apache Beam
Seattle, WA, USA
kenn@apache.org

ABSTRACT

Real-time data analysis and management are increasingly critical for today's businesses. SQL is the de facto *lingua franca* for these endeavors, yet support for robust streaming analysis and management with SQL remains limited. Many approaches restrict semantics to a reduced subset of features and/or require a suite of non-standard constructs. Additionally, use of event timestamps to provide native support for analyzing events according to when they actually occurred is not pervasive, and often comes with important limitations.

We present a three-part proposal for integrating robust streaming into the SQL standard, namely: (1) time-varying relations as a foundation for classical tables as well as stream-

CCS CONCEPTS

• **Information systems → Stream management**; **Query languages**;

KEYWORDS

stream processing, data management, query processing

ACM Reference Format:
Edmon Begoli, Tyler Akidau, Fabian Hueske, Julian Hyde, Kathryn Knight, and Kenneth Knowles. 2019. One SQL to Rule Them All: An Efficient and Syntactically Idiomatic Approach to Management of Streams and Tables: An Industrial Paper. In *2019 International Conference on Management of Data (SIGMOD '19), June 30–July 5,*