Databases and Stream Processing:

A Future of Consolidation

Ben Stopford

Office of the CTO, Confluent

Marc Andreessen:
Software is Eating the World

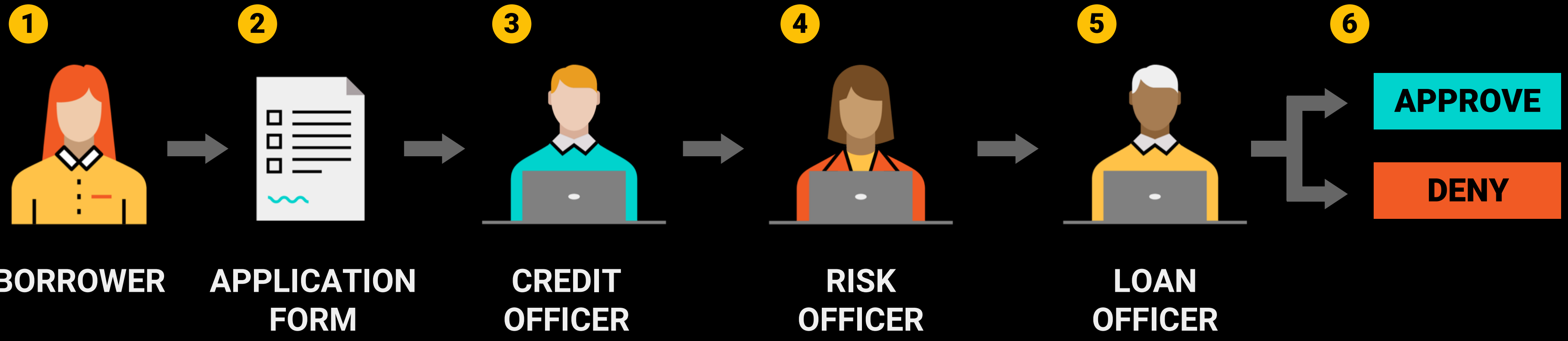
Weak Form

Companies are
USING MORE SOFTWARE

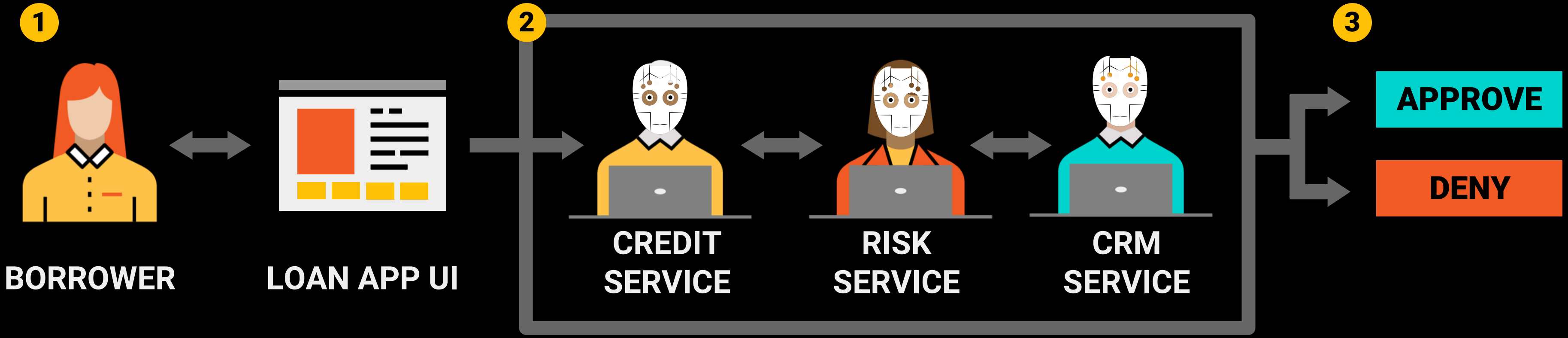
Strong Form

Companies are
BECOMING SOFTWARE

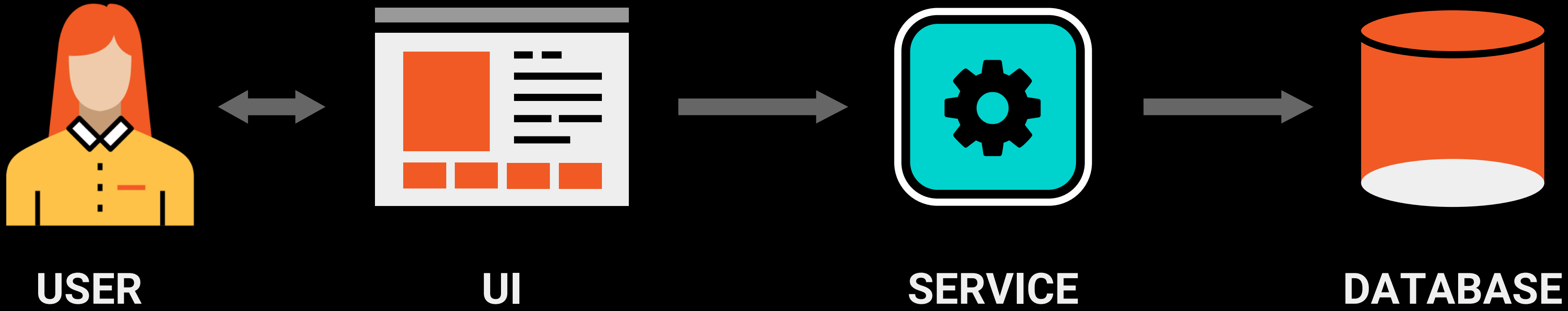
Loan Application Using Software



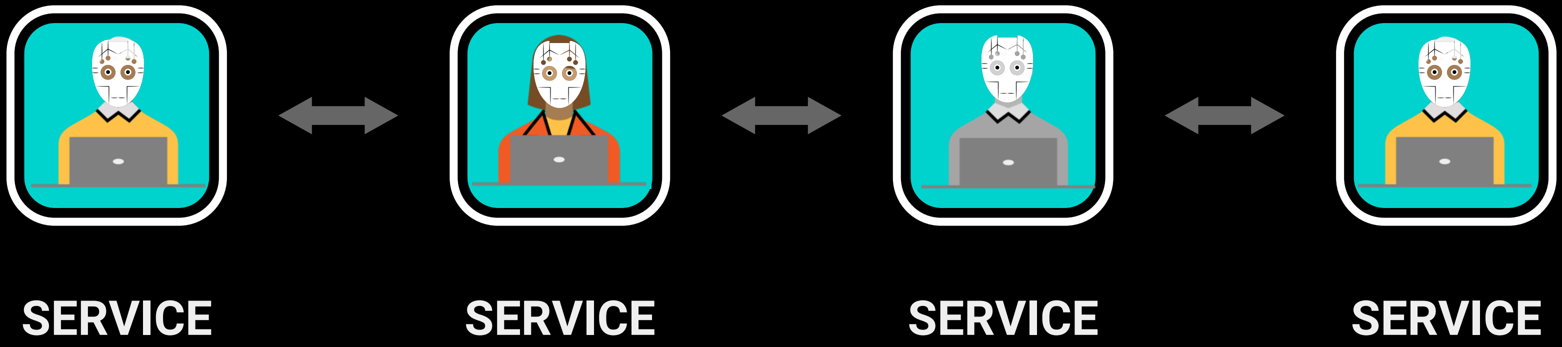
Loan Application in Software



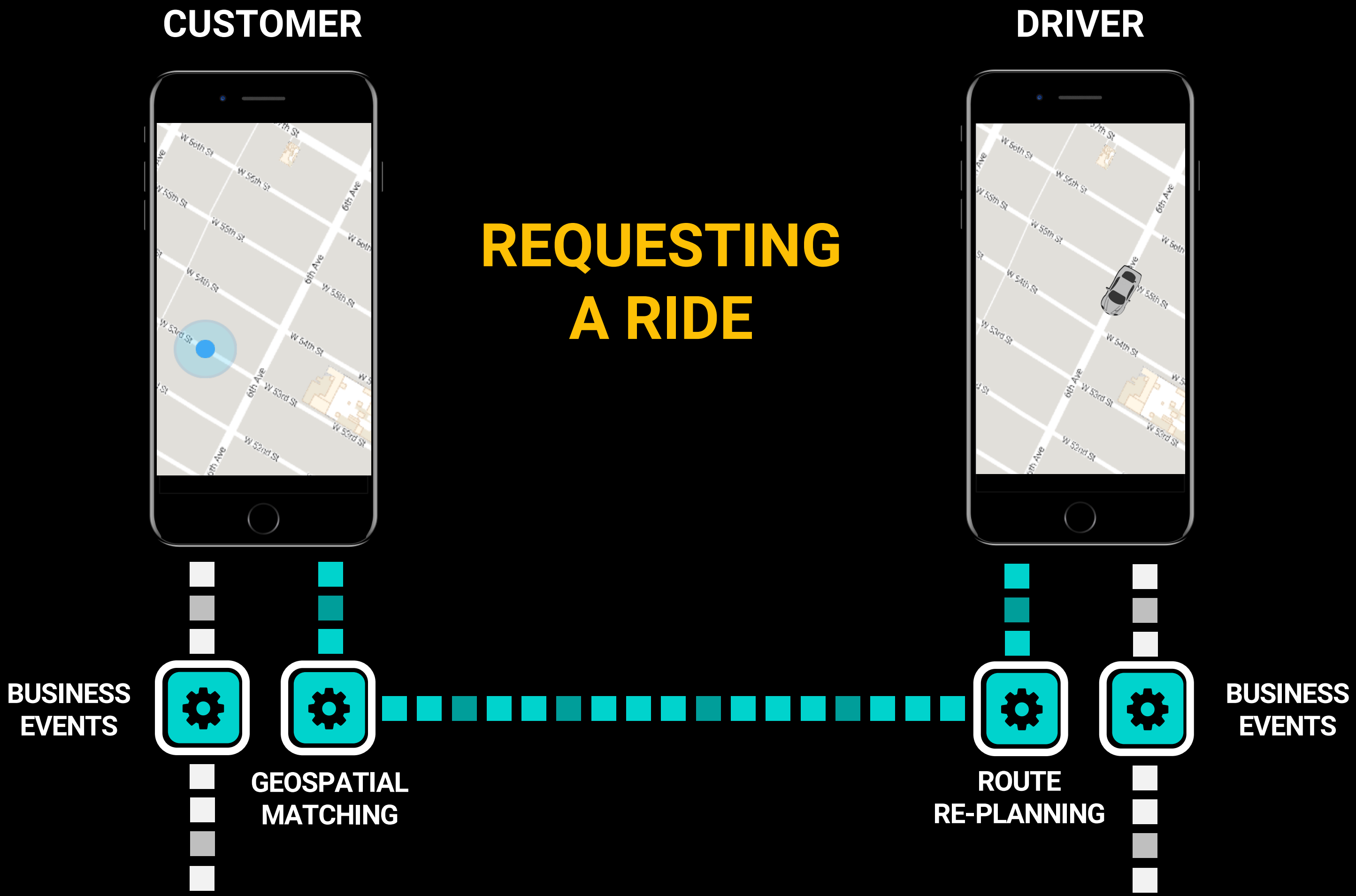
Using Software:
Classic Three-Tier Architecture



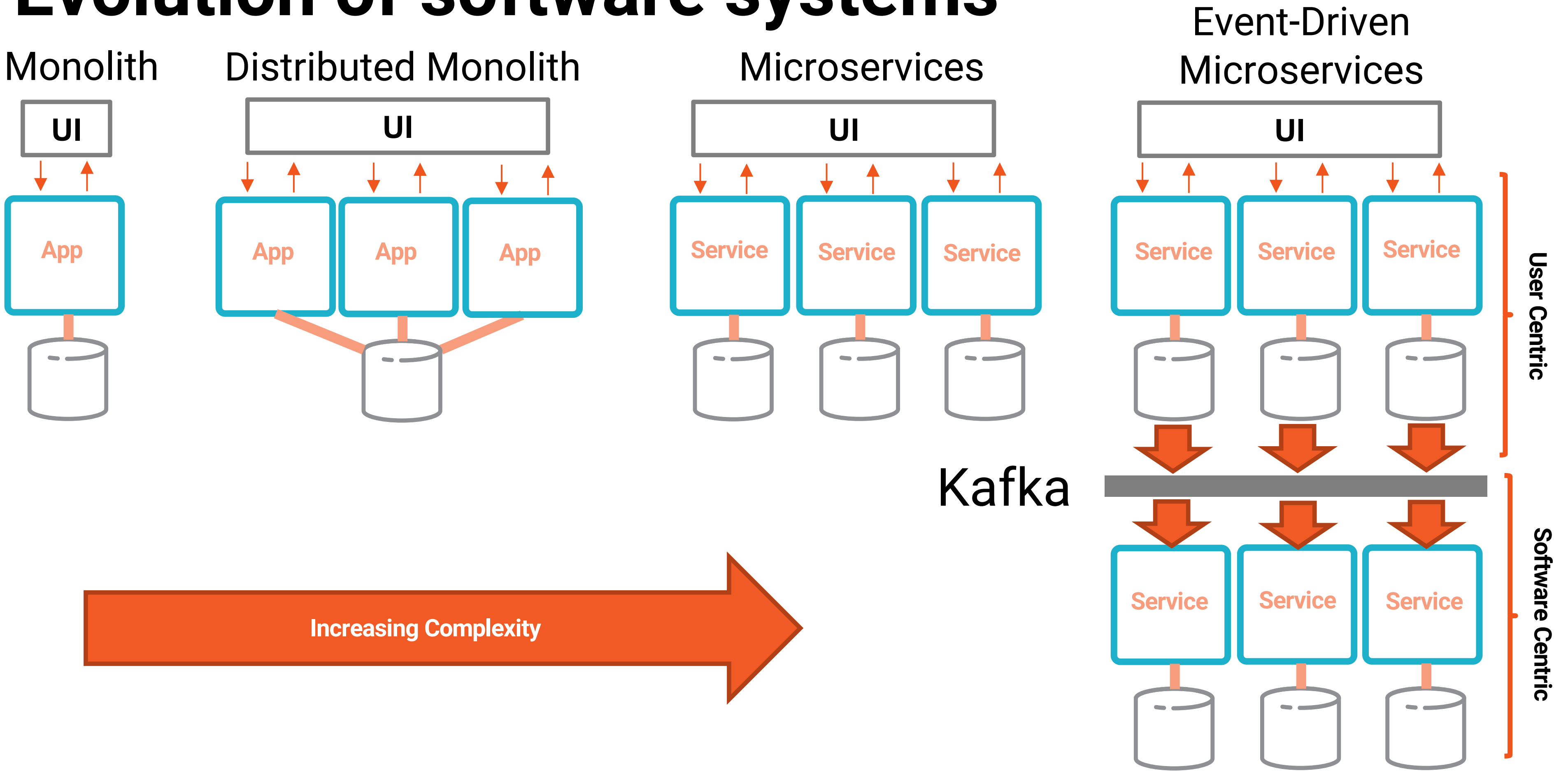
Becoming Software:
Services Talking To Each Other With APIs



REQUESTING A RIDE



Evolution of software systems



THE USER OF THE SOFTWARE

IS MORE SOFTWARE

What does this mean for databases?

We have hundreds of databases...

FUNDAMENTAL ASSUMPTION:

DATA IS PASSIVE

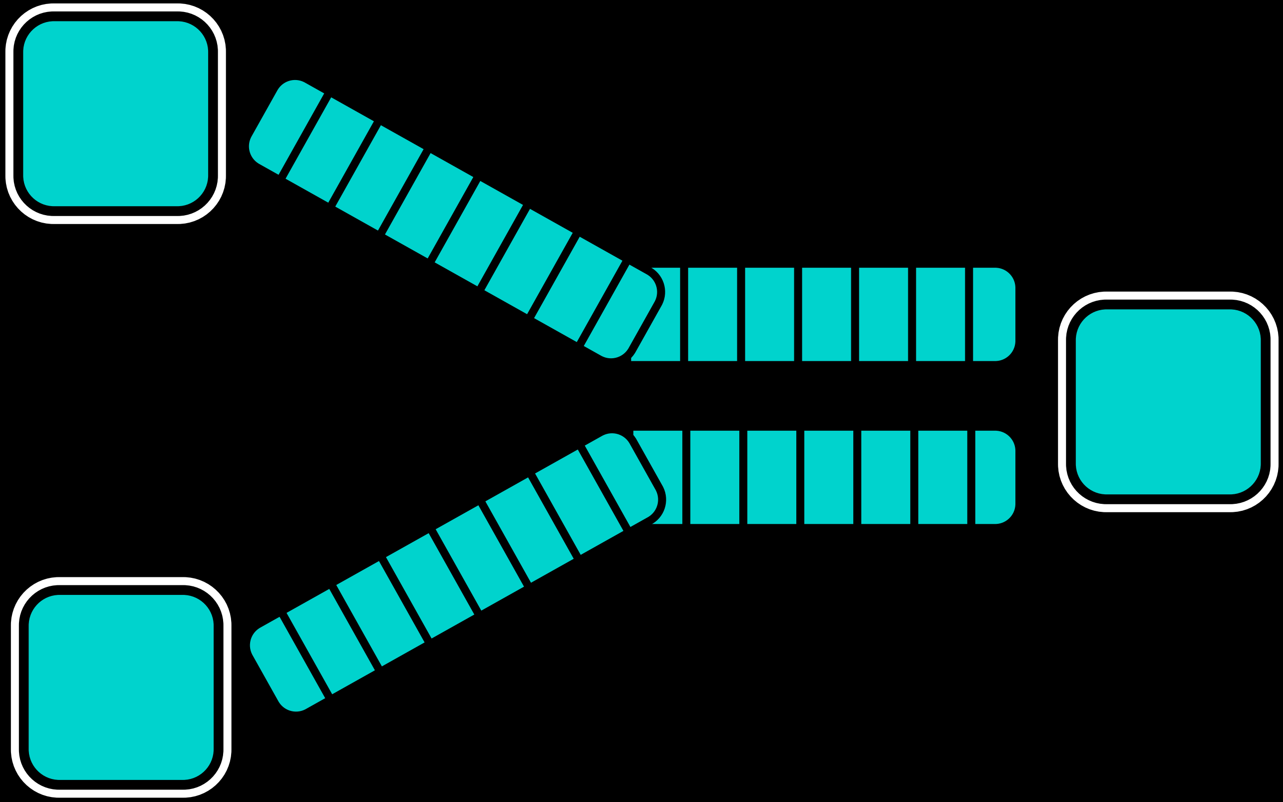
We have hundreds of databases.

Databases are designed to help *you*!

Unless there is a user and UI waiting, why should it be synchronous?

The Alternative: Event Streams

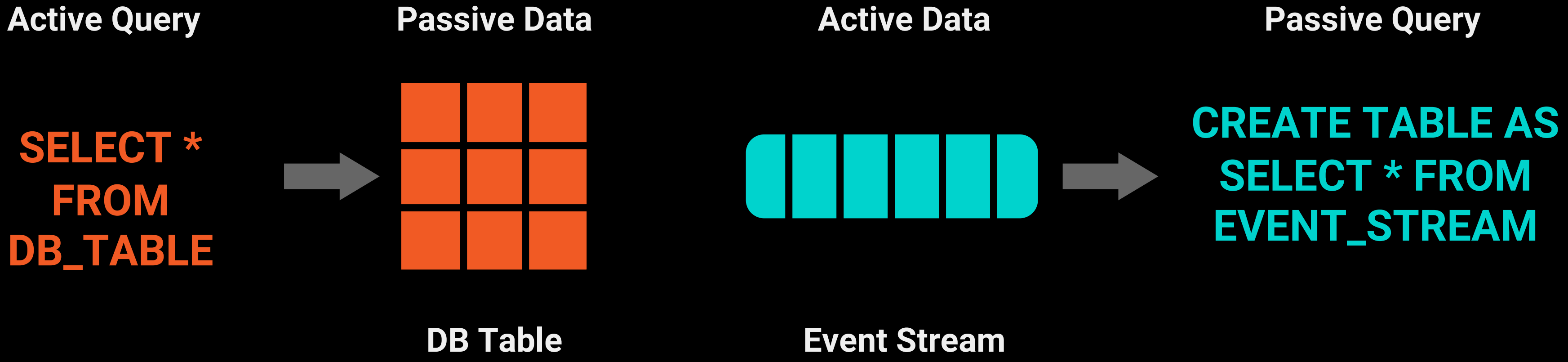
Stream Processors are built for Asynchronicity



Stream Processors have a different interaction model

TRADITIONAL DATABASE

EVENT STREAM PROCESSING



Streams or Tables?

An Event
records the fact that something happened

A good
was sold

An invoice
was issued

A payment
was made

A new customer
registered

Events are state changes, they carry intent

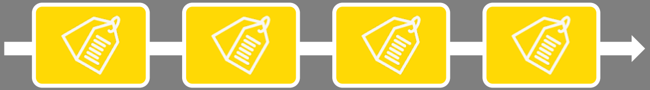
State:

Bob works at Google

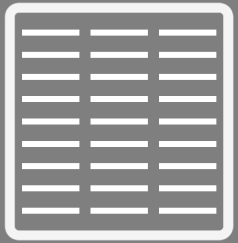
Event:

Bob moved from Google to Amazon

Streams
record exactly what happened



Tables
current state

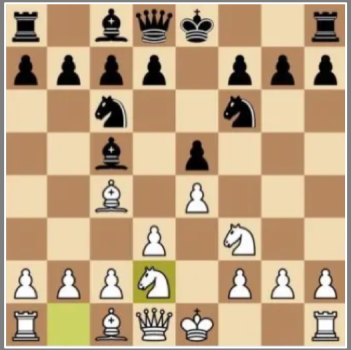


Where you have been	vs.	Where you are now
Payments you made	vs.	Your account balance

Streams
A sequence of moves

1.	e4	e5
2.	Nf3	Nc6
3.	Bc4	Bc5
4.	d3	Nf6
5.	Nbd2	

Tables
Position of each piece

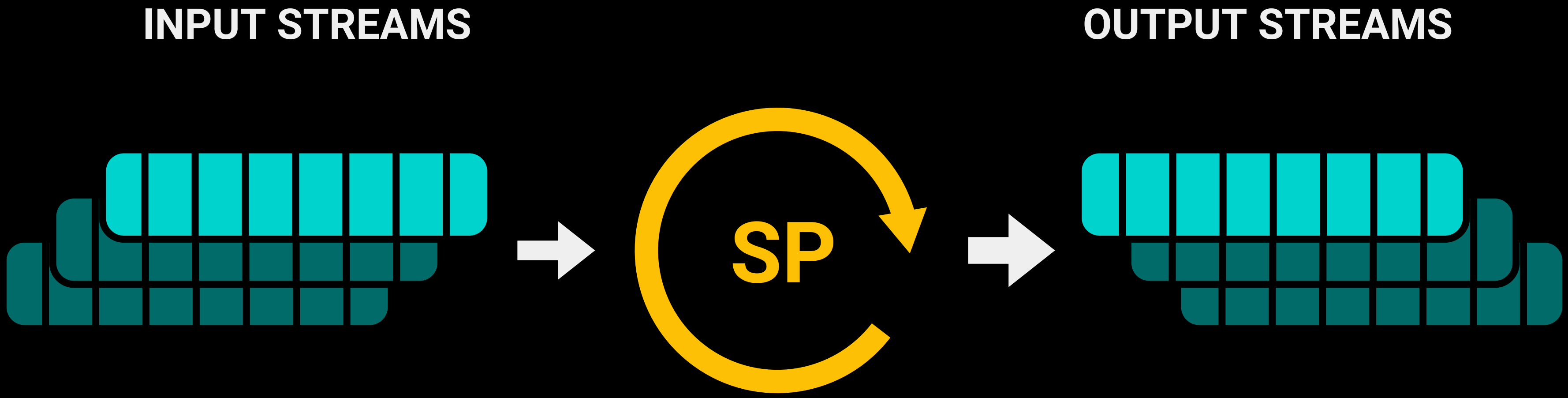


Streams = INSERT only
Immutable, append-only

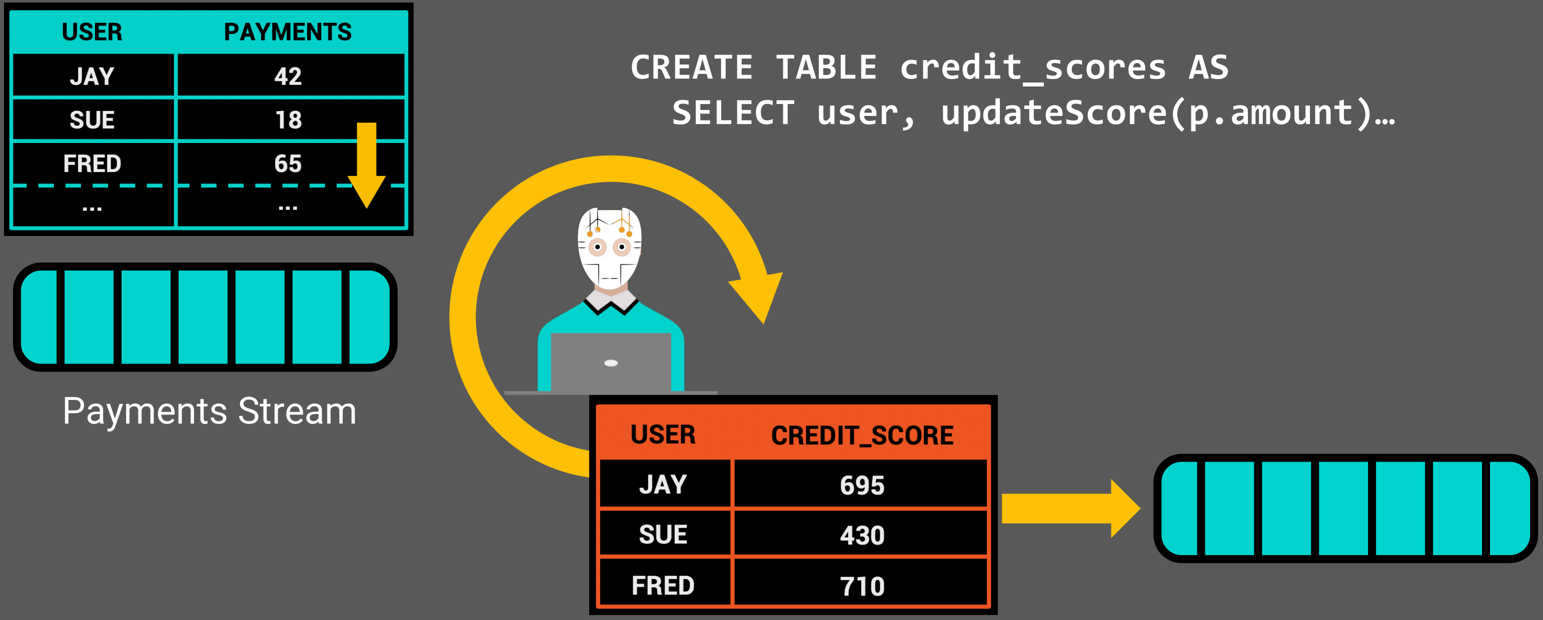
Tables = INSERT, UPDATE, DELETE
Mutable, Primary Key

A stream can be considered as an **immutable, append-only** table

Stream Processors Communicate Through Streams



But internally they use tables



USER	PAYMENTS
JAY	42
SUE	18
FRED	65
...	...

Payments Stream

```
CREATE TABLE credit_scores AS
  SELECT user, updateScore(p.amount)…
```

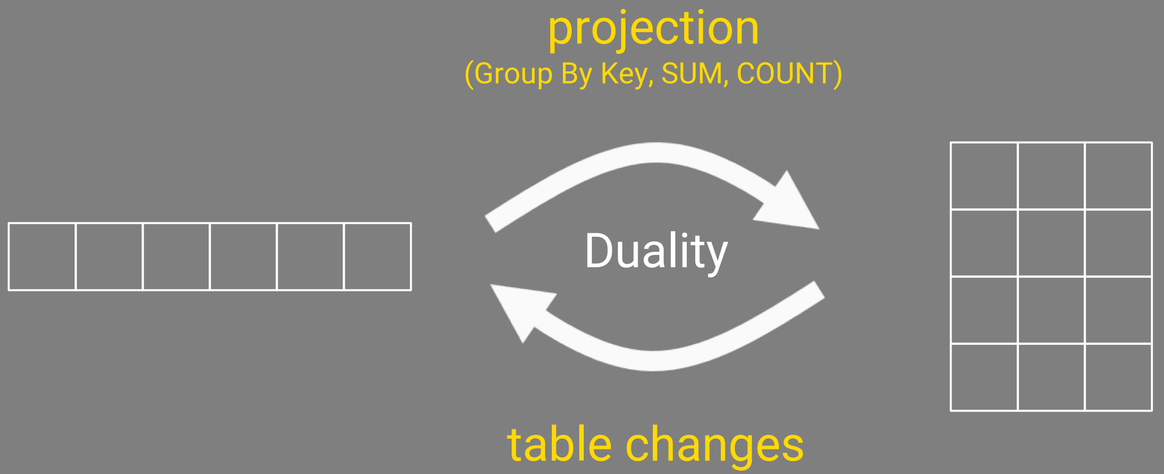
USER	CREDIT_SCORE
JAY	695
SUE	430
FRED	710

Credit Score Table

Credit Score Stream

Streams
record history

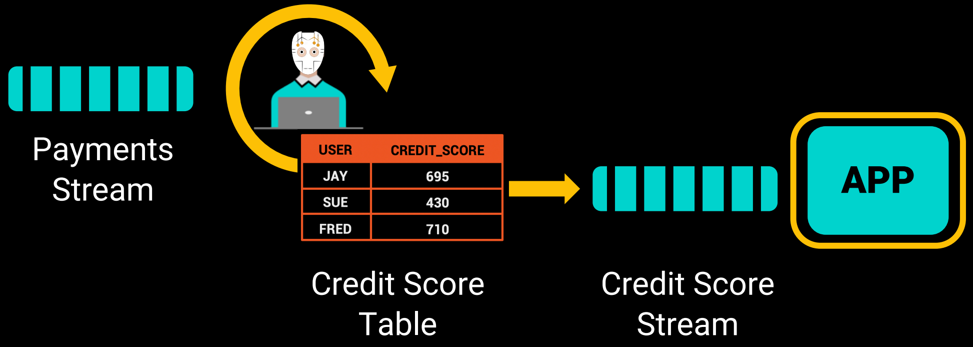
Tables
represent state



See Streams and Tables: Two Sides of the Same Coin, M. Sax et al., BIRTE '18

Similar to a materialized view in a database

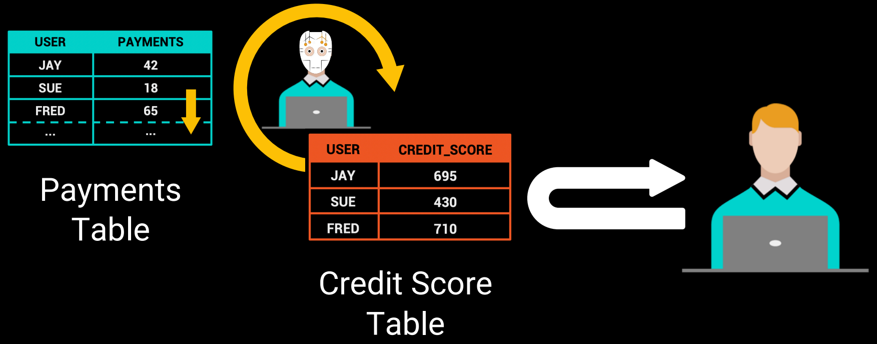
STREAM PROCESSOR



USER	CREDIT_SCORE
JAY	695
SUE	430
FRED	710

- Asynchronous
- Push query

ACTIVE DATABASE

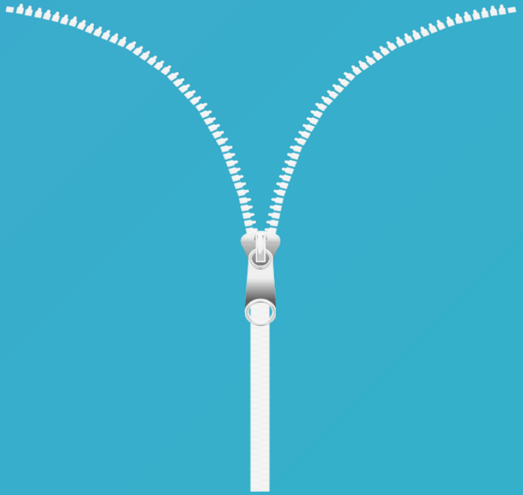


USER	PAYMENTS
JAY	42
SUE	18
FRED	65
...	...

USER	CREDIT_SCORE
JAY	695
SUE	430
FRED	710

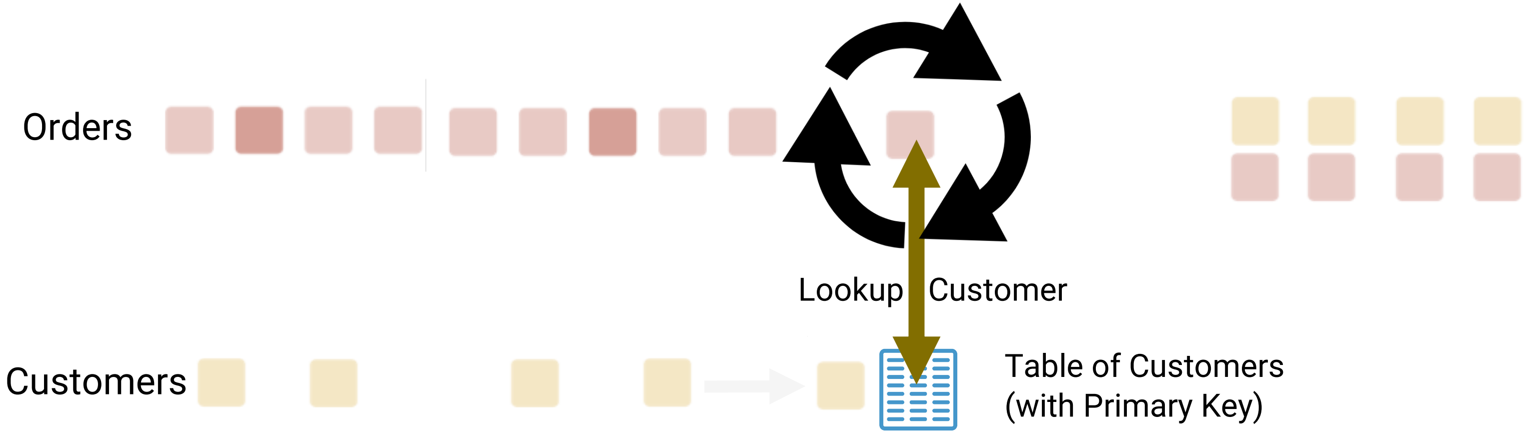
- Synchronous
- Pull query

Joins

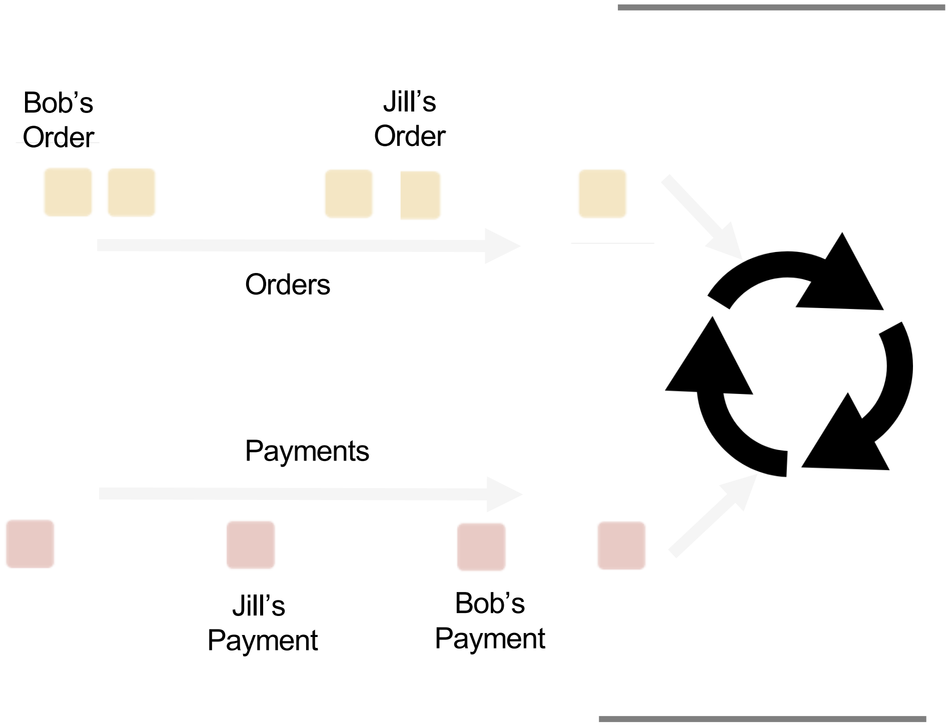


confluent

Joining a stream with a table

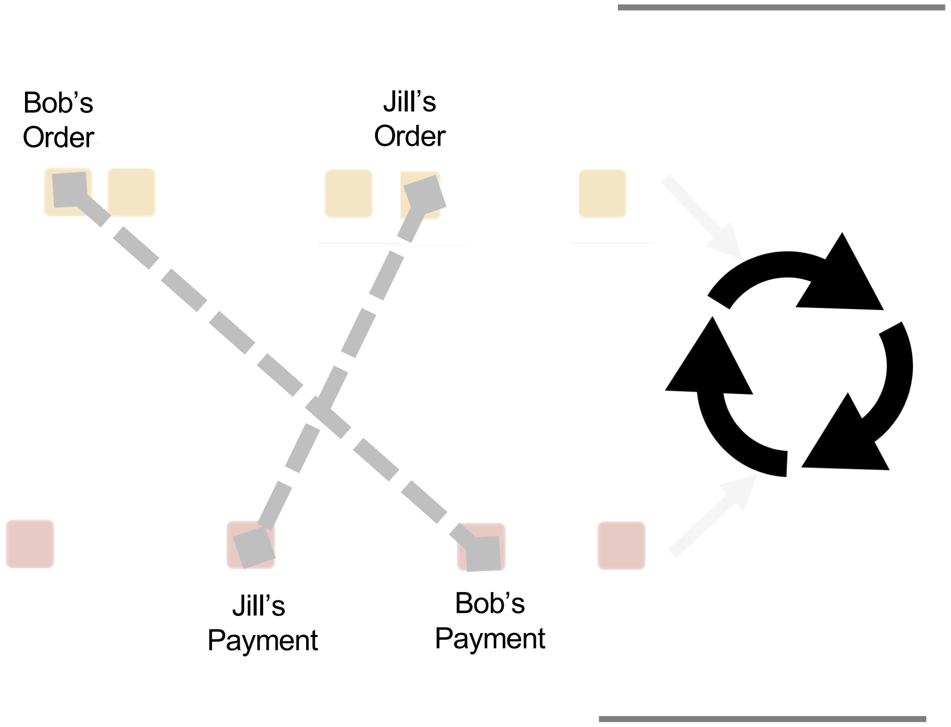


Joining two streams



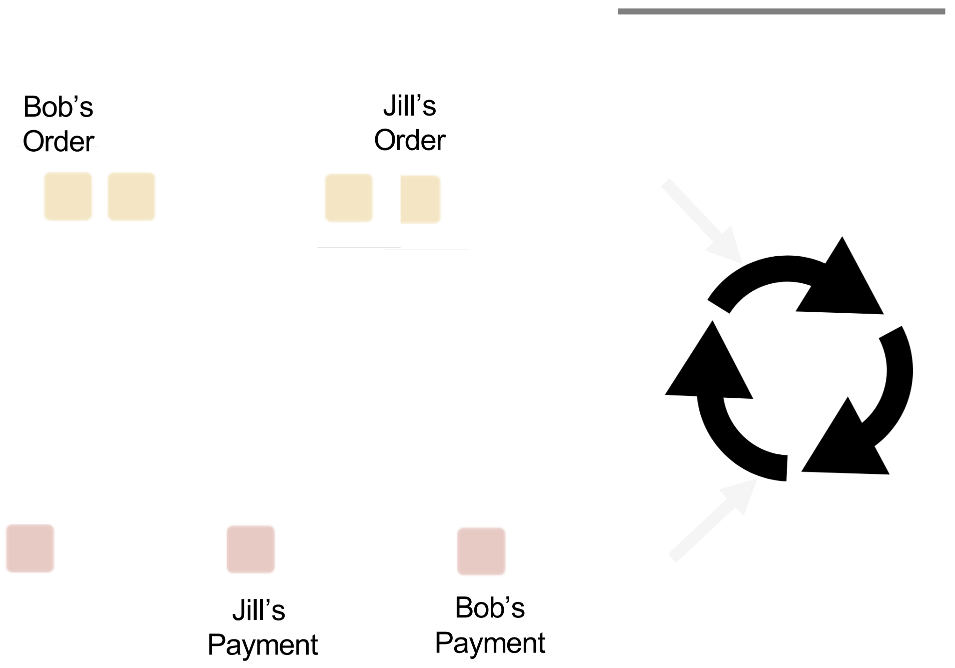
```
orders.join(payments)
```

Joining two streams



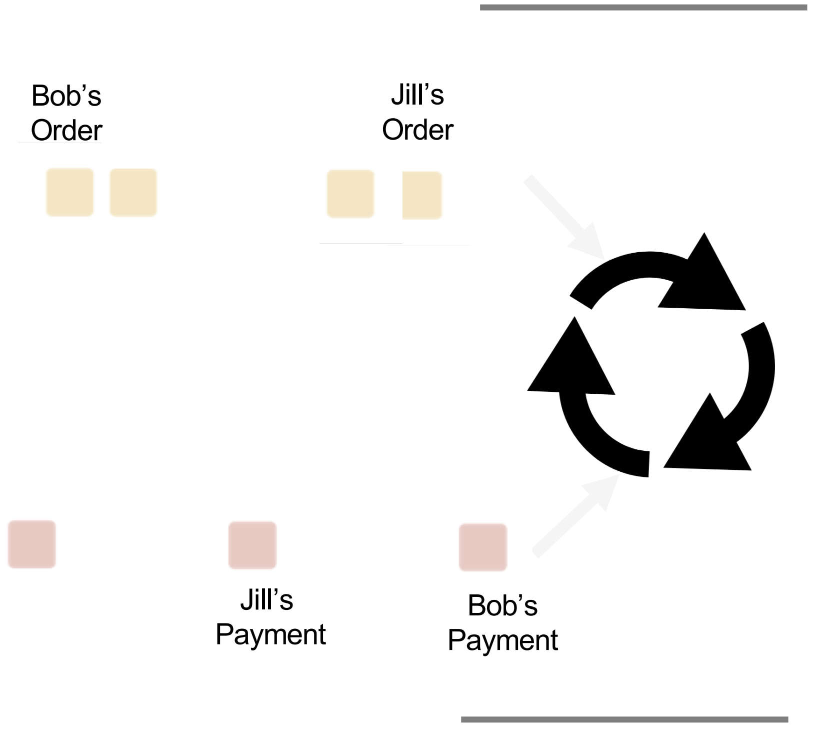
```
orders.join(payments)
```

Joining two streams



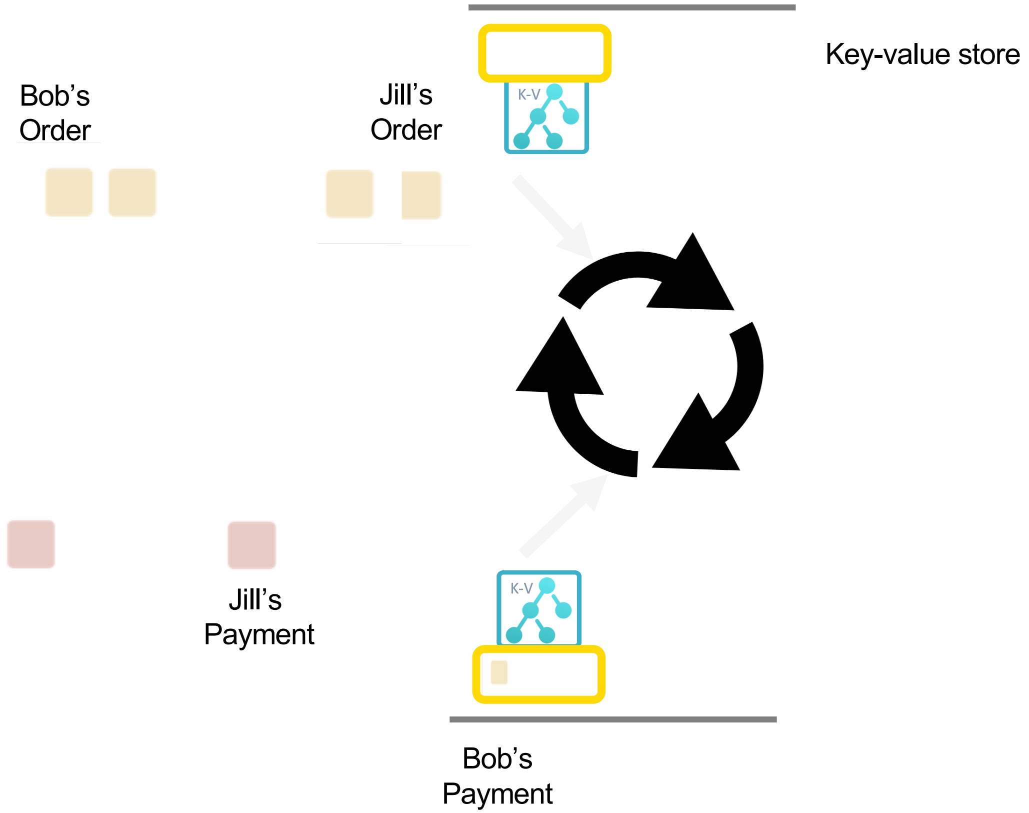
```
orders.join(payments)
```

Joining two streams

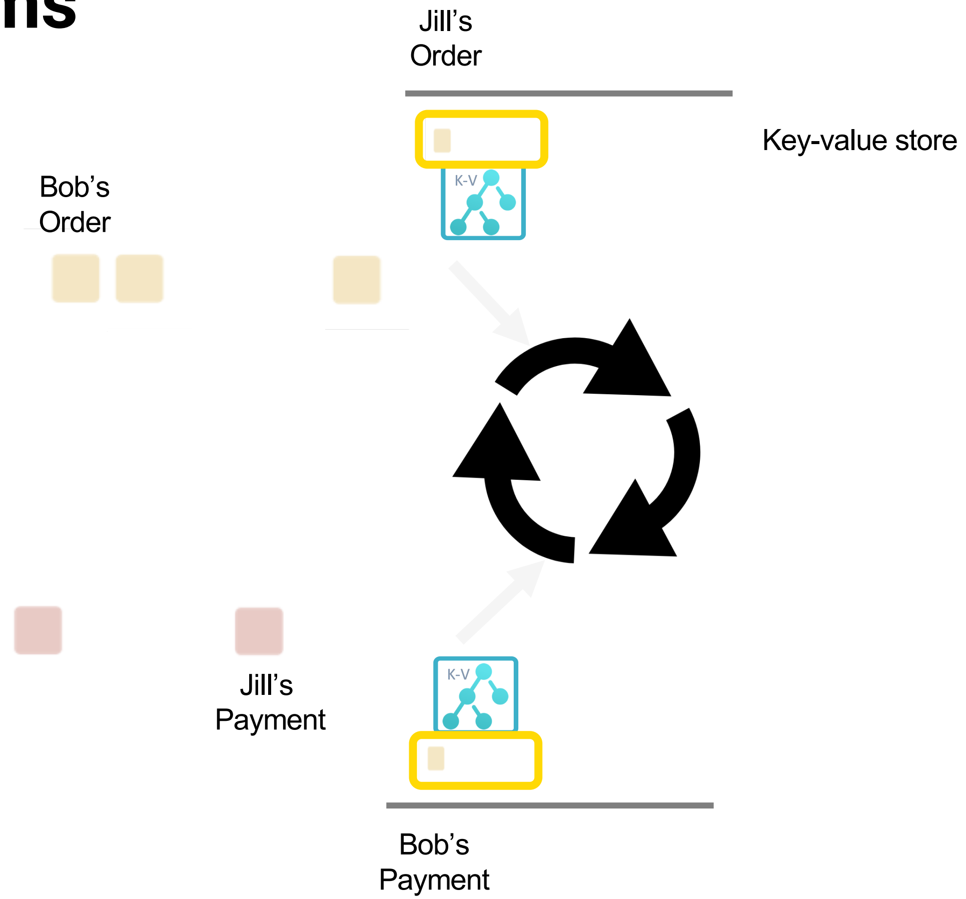


```
orders.join(payments)
```

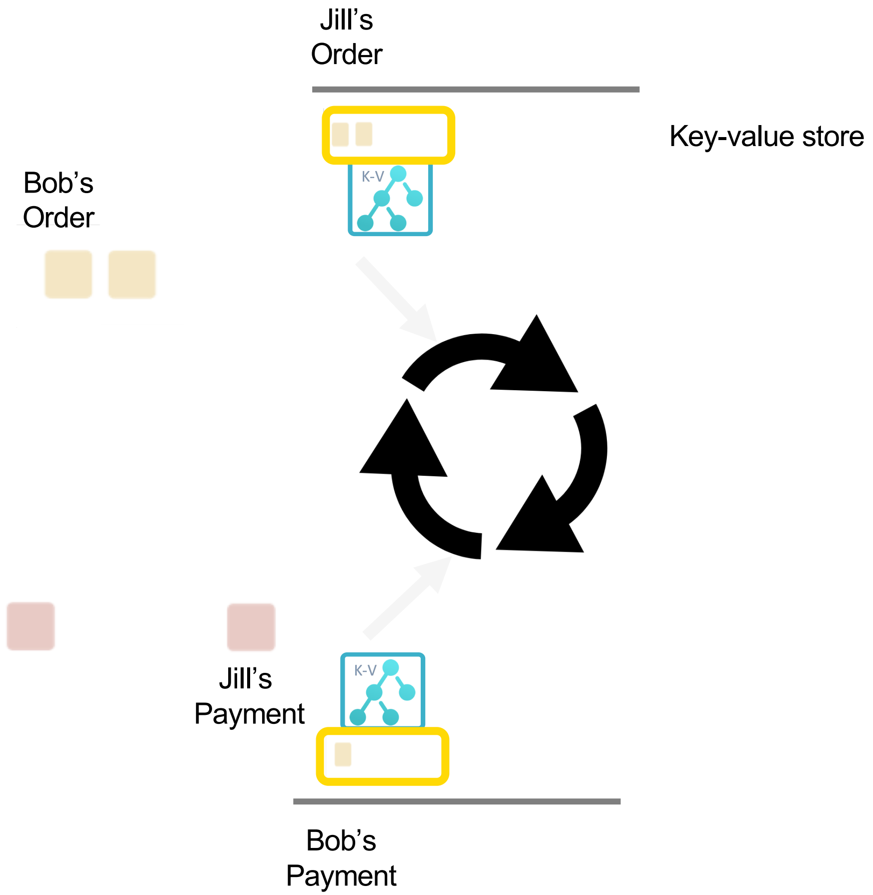
Joining two streams



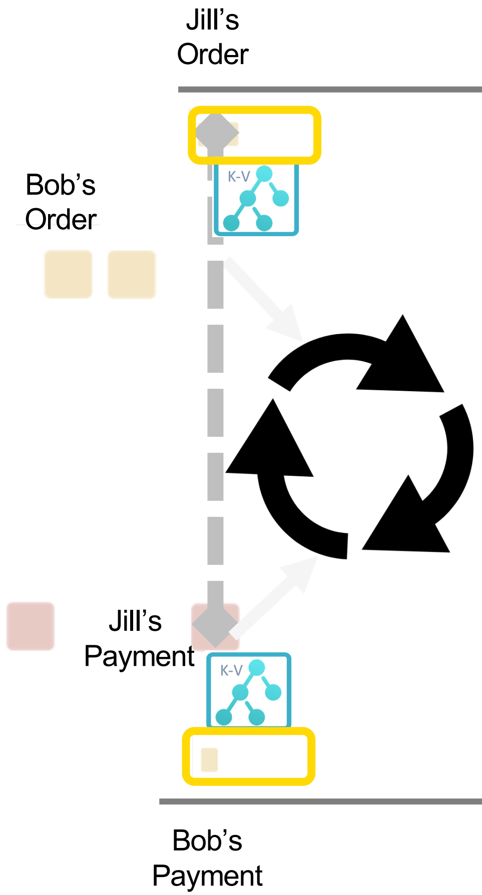
Joining two streams



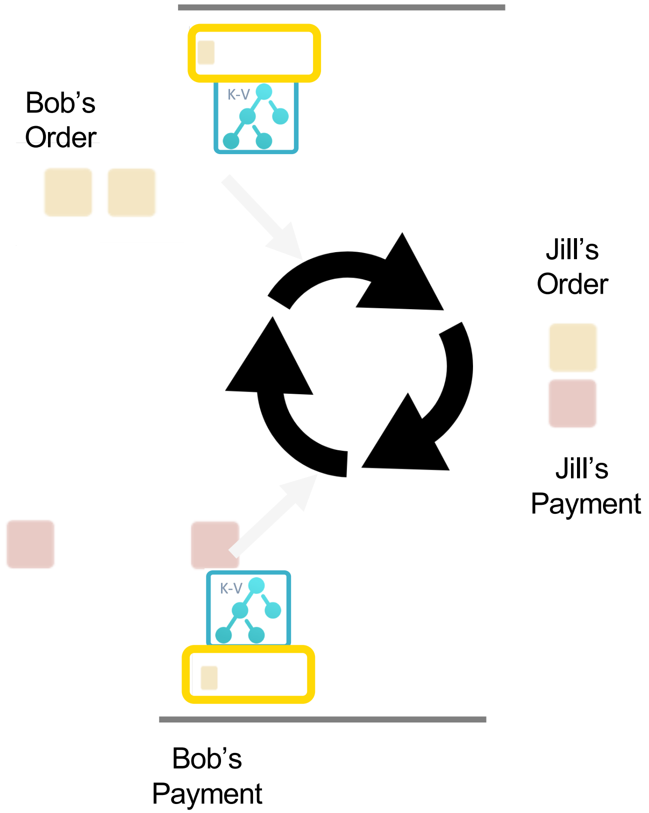
Joining two streams



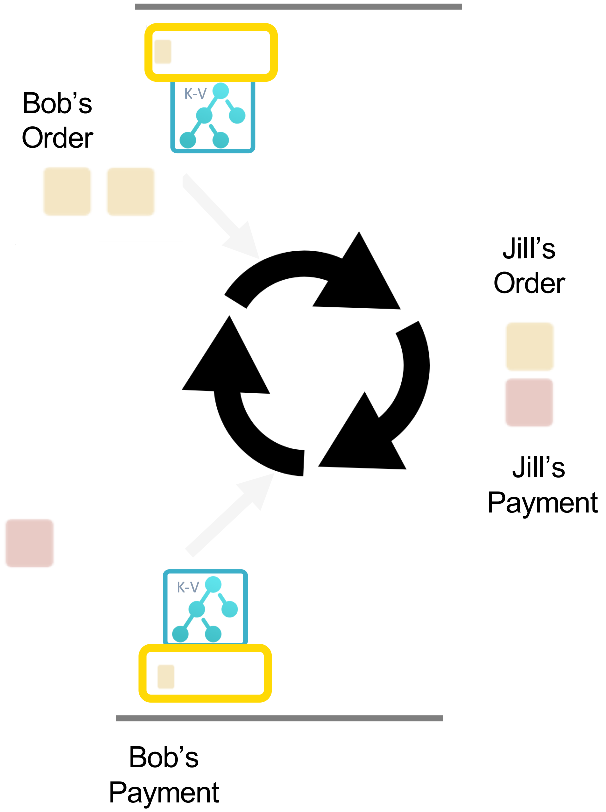
Joining two streams



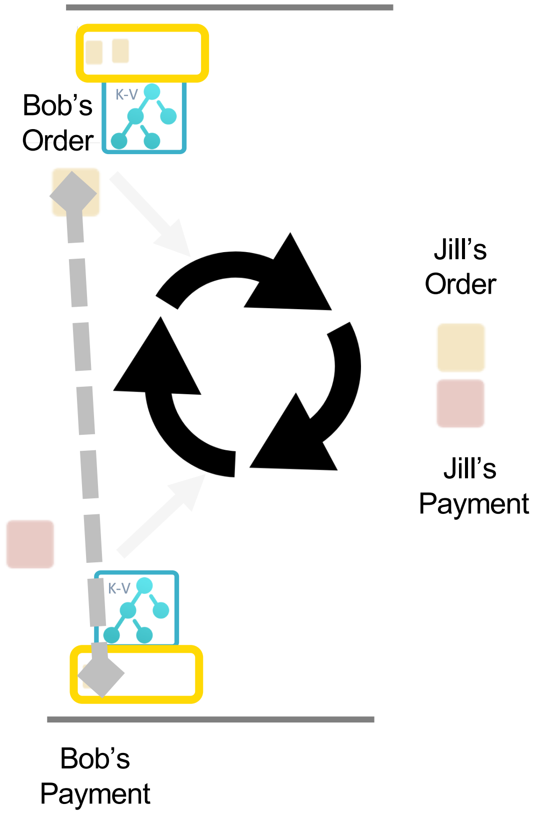
Joining two streams



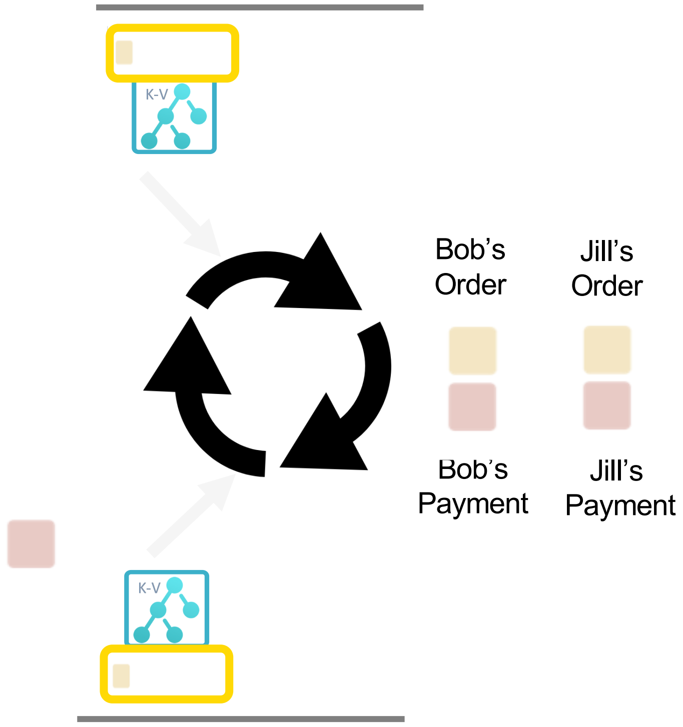
Joining two streams



Joining two streams



Joining two streams



Streams represent history –> Cartesian Product

Orders Stream

101	Boots
200	Hat
101	Boots2
105	Pants
200	Hat2

Join Output (Stream)

Payments Stream

101	$50
200	$10
105	$3
200	$12
101	$60

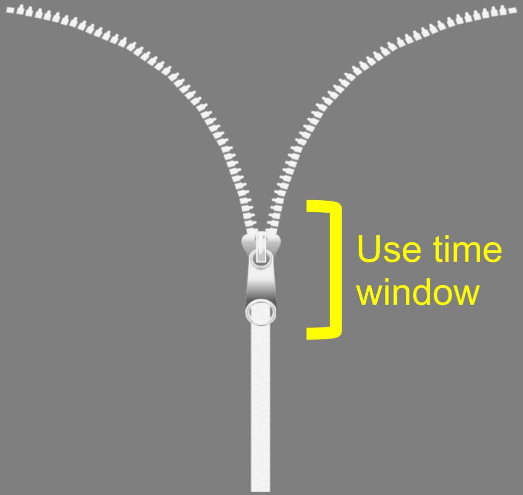
Joining Streams to Streams

Orders Stream

101	Boots
200	Hat
101	Boots2
105	Pants
200	Hat2

Payments Stream

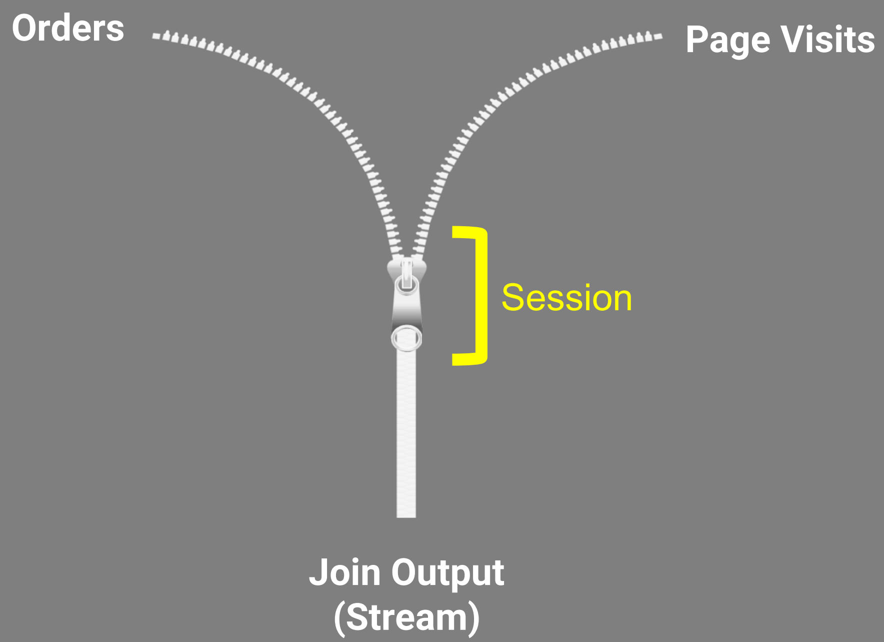
101	$50
200	$10
105	$3
200	$12
101	$60



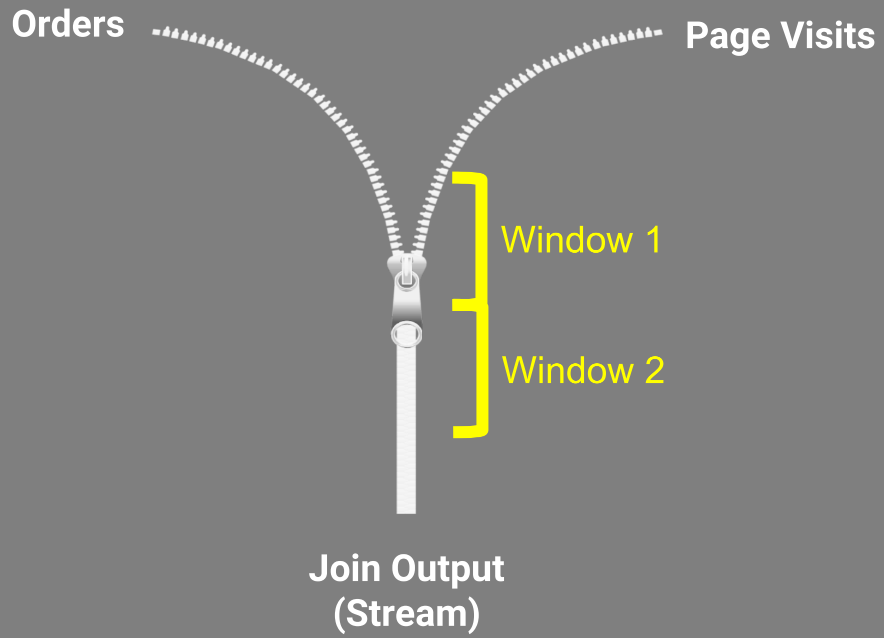
Join Output (Stream)

Tools for correlating recent events in time

More advanced temporal functions



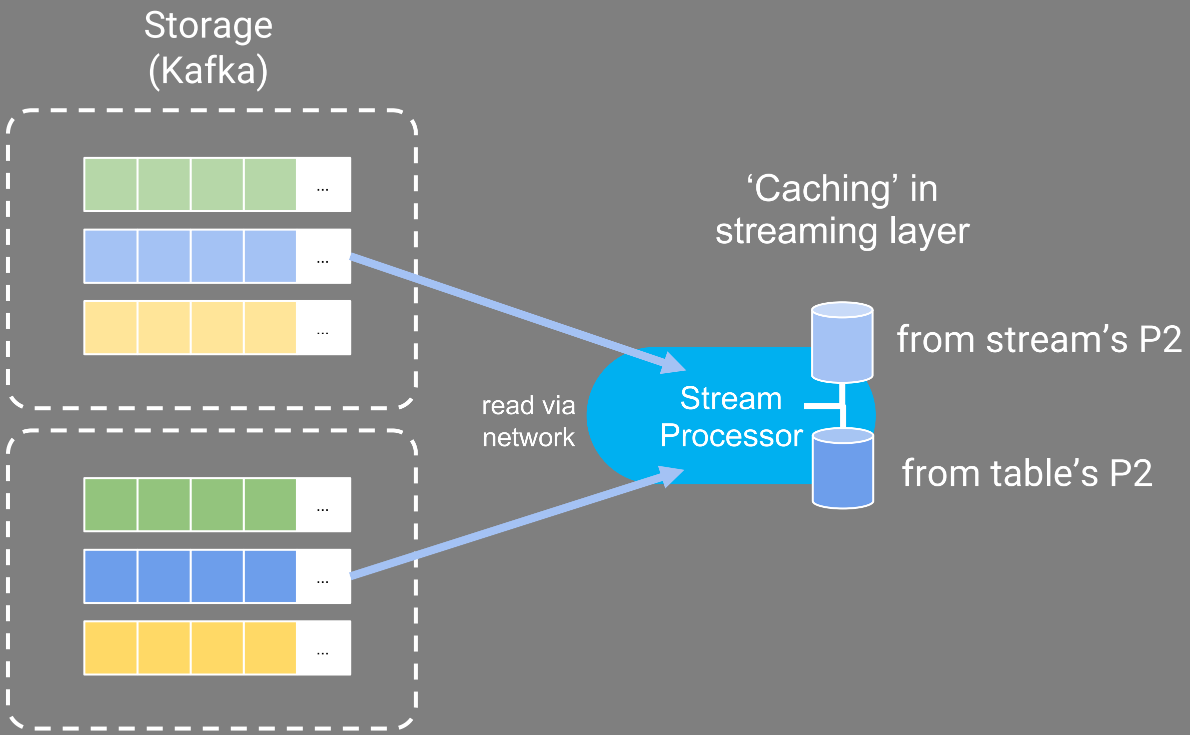
Late and out-of-order data



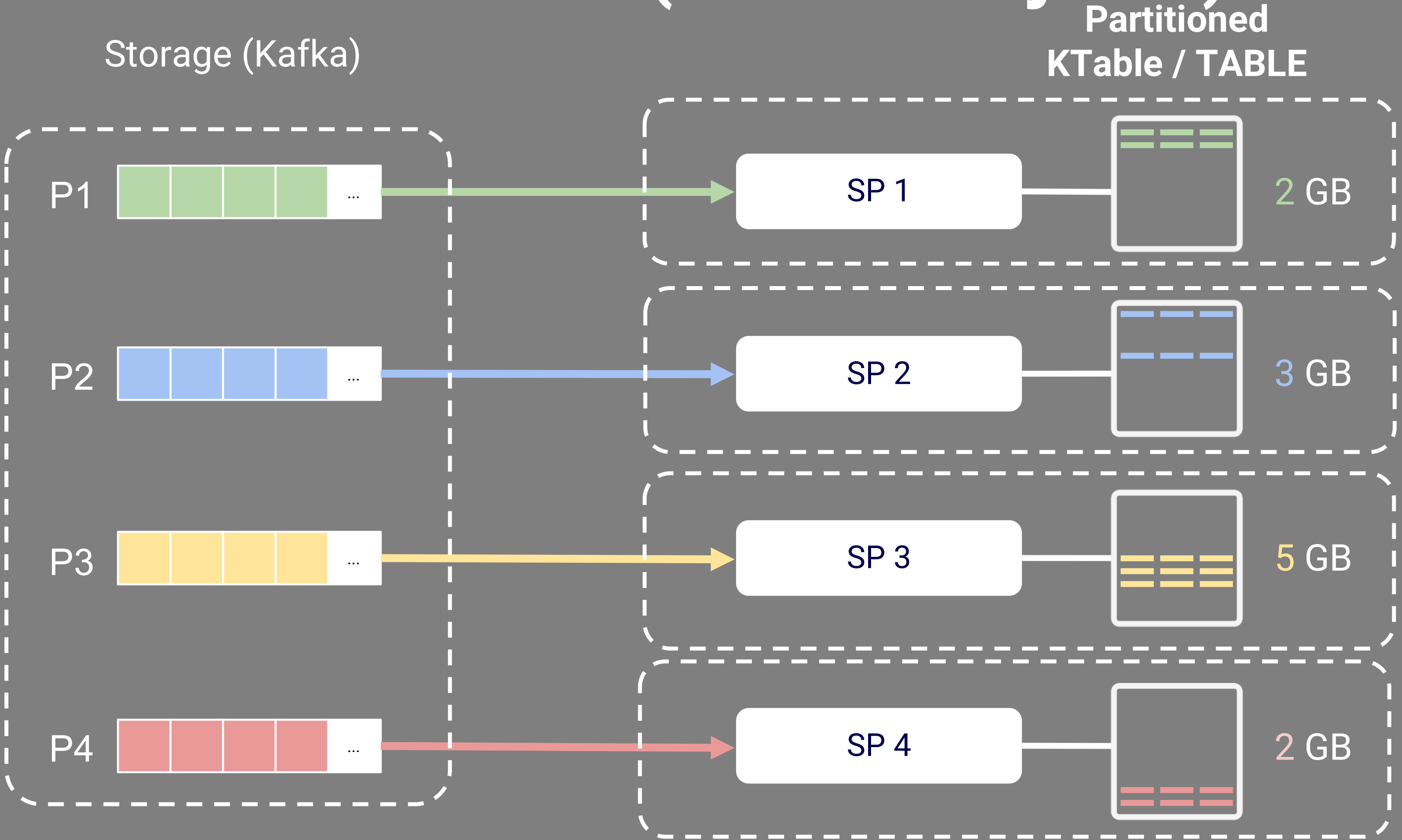
Stream processors provide tools that handle **asynchronicity**, leverage **time** and focus on ‘**now**’

Data Placement

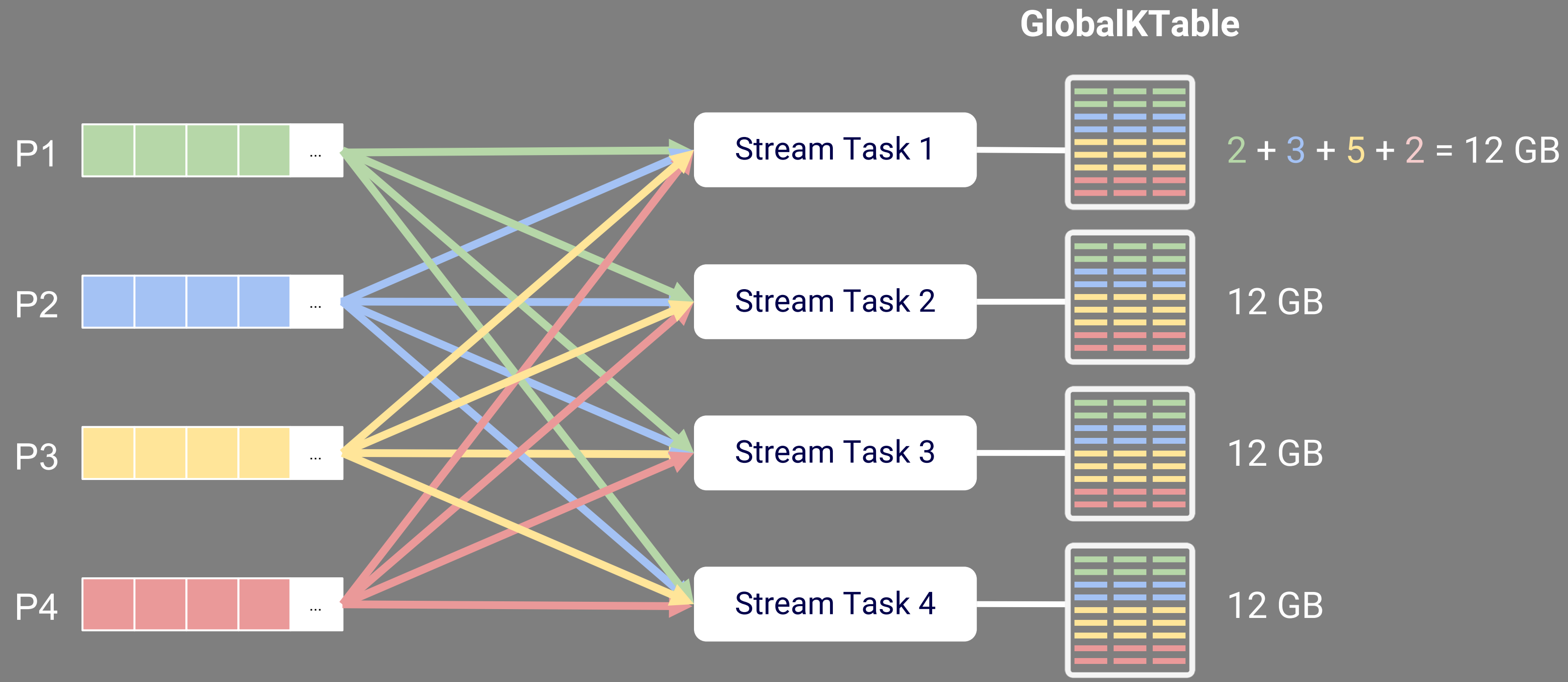
Layered storage model



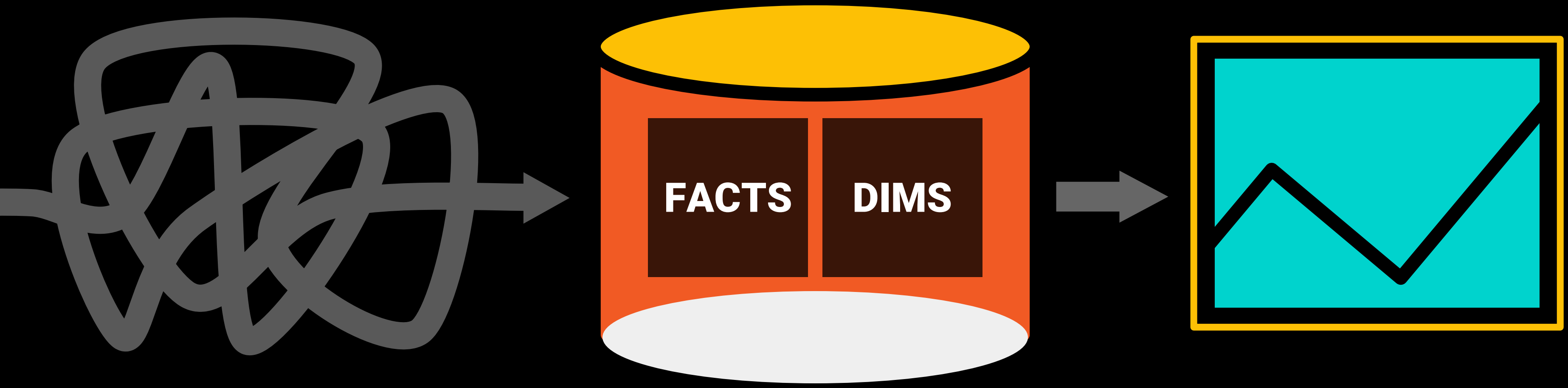
Partitioned Data (Fact-Fact joins)



Broadcast Data (Fact-Dimension Joins)



Architecturally there are parallels e.g. Data Warehousing

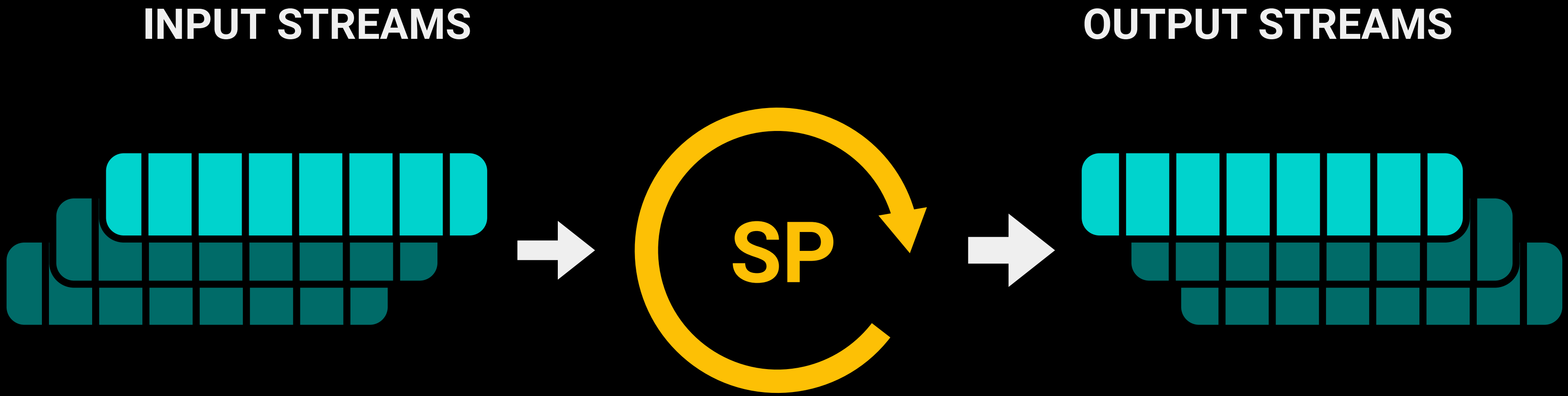


ETL

REPORTING

Interaction Model

Stream Processors Continuously Process Input to Output

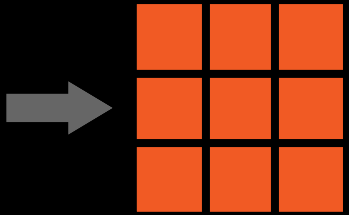


TRADITIONAL DATABASE

EVENT STREAM PROCESSING

Active Query

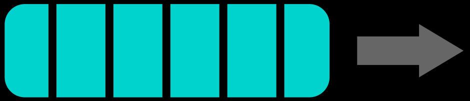
SELECT * FROM DB_TABLE



Passive Data

DB Table

Active Data

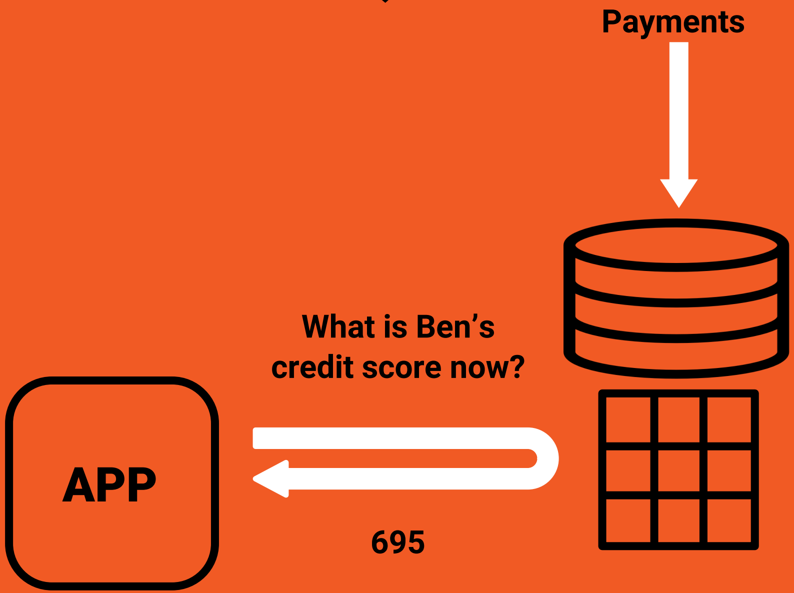


Event Stream

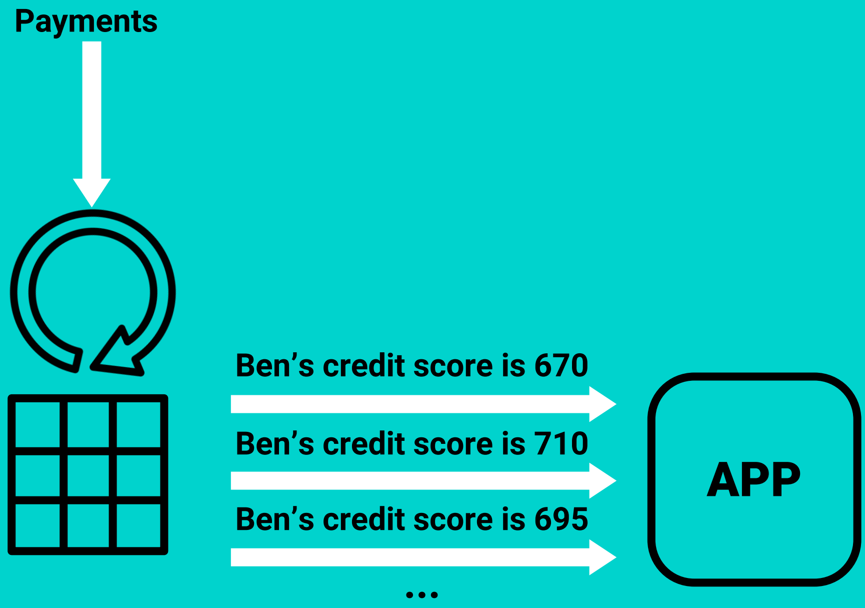
Passive Query

CREATE TABLE AS SELECT * FROM EVENT_STREAM

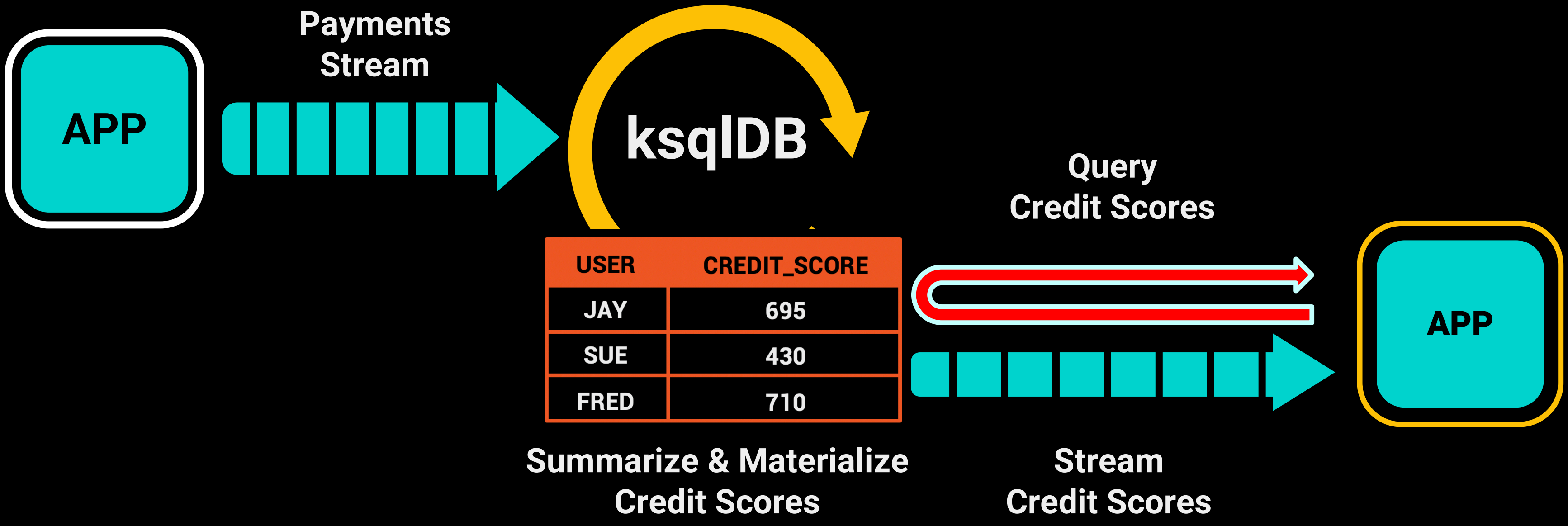
Databases are Pull Queries



Stream Processors are Push Queries


Payments
Ben's credit score is 670
Ben's credit score is 710
Ben's credit score is 695
...
APP

Hybrid stream processors provide both interaction models

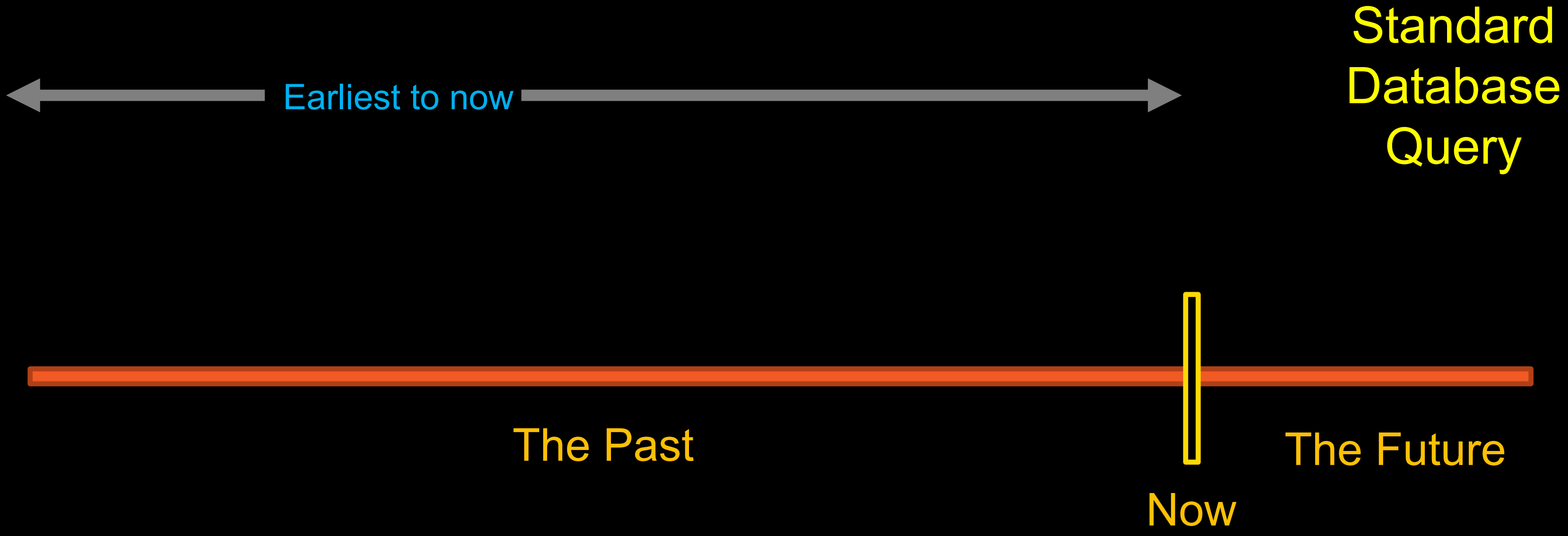


USER	CREDIT_SCORE
JAY	695
SUE	430
FRED	710

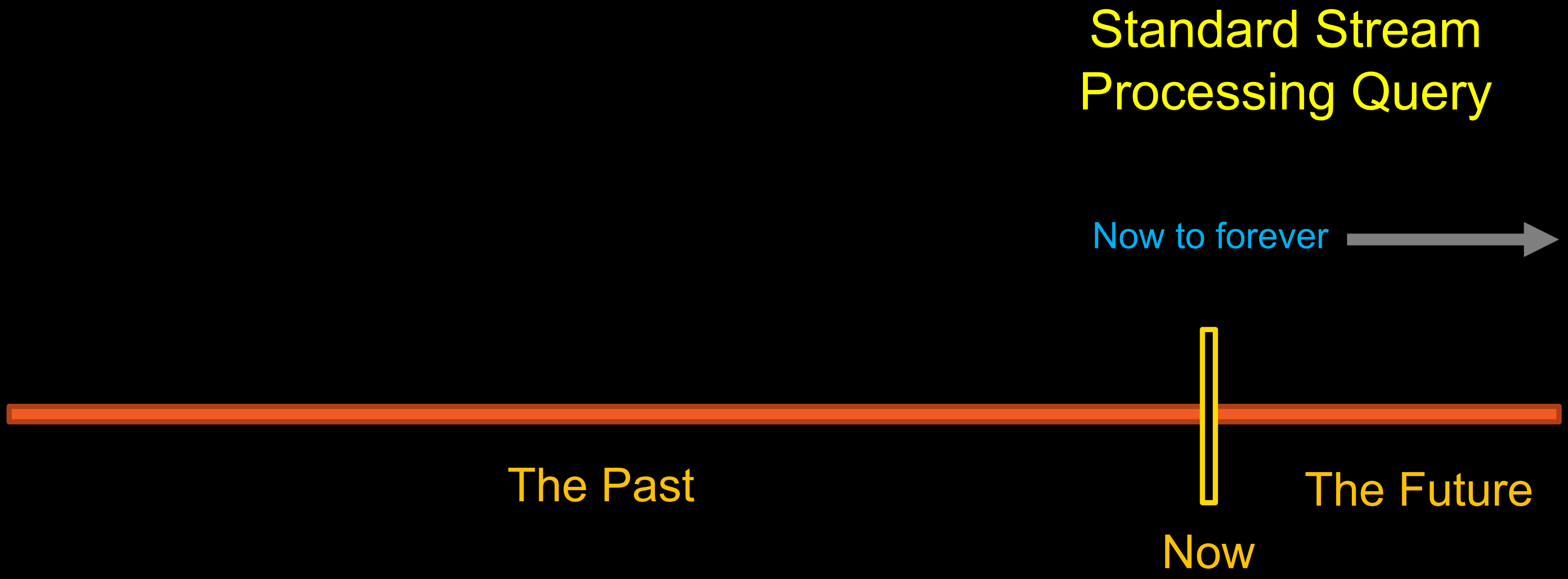
Unified Model For:

1. The Asynchronous and the Synchronous
2. Interaction with Active or Passive Data

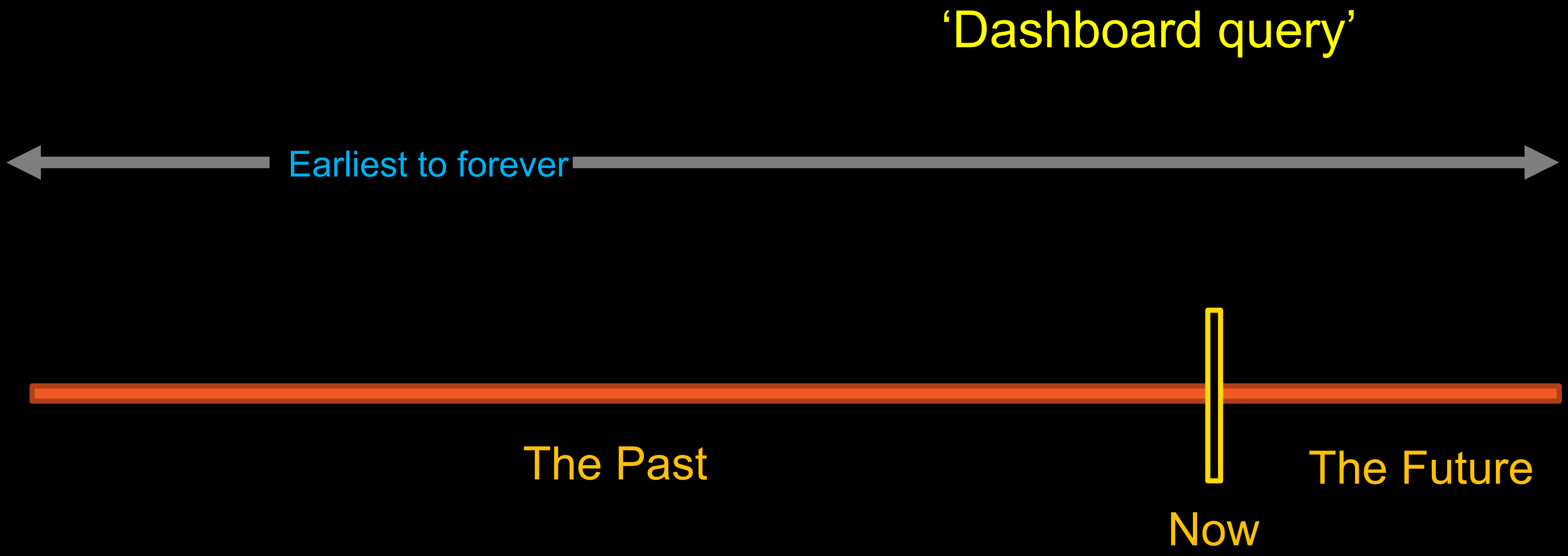
Unified interaction model



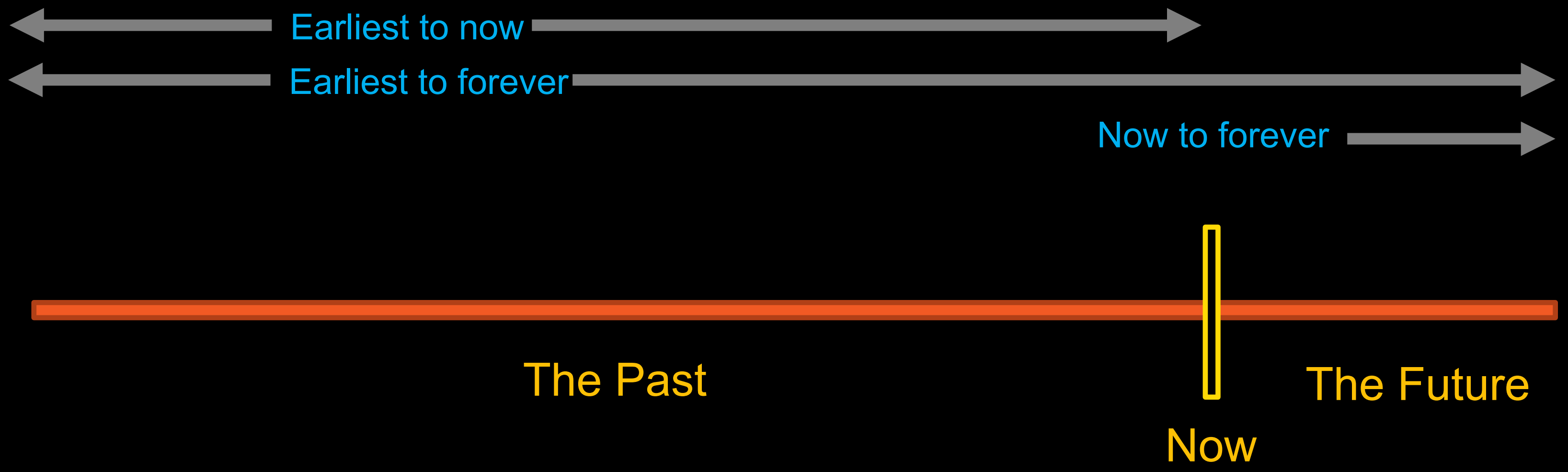
Unified interaction model



Unified interaction model



Unified Interaction Model



PUSH

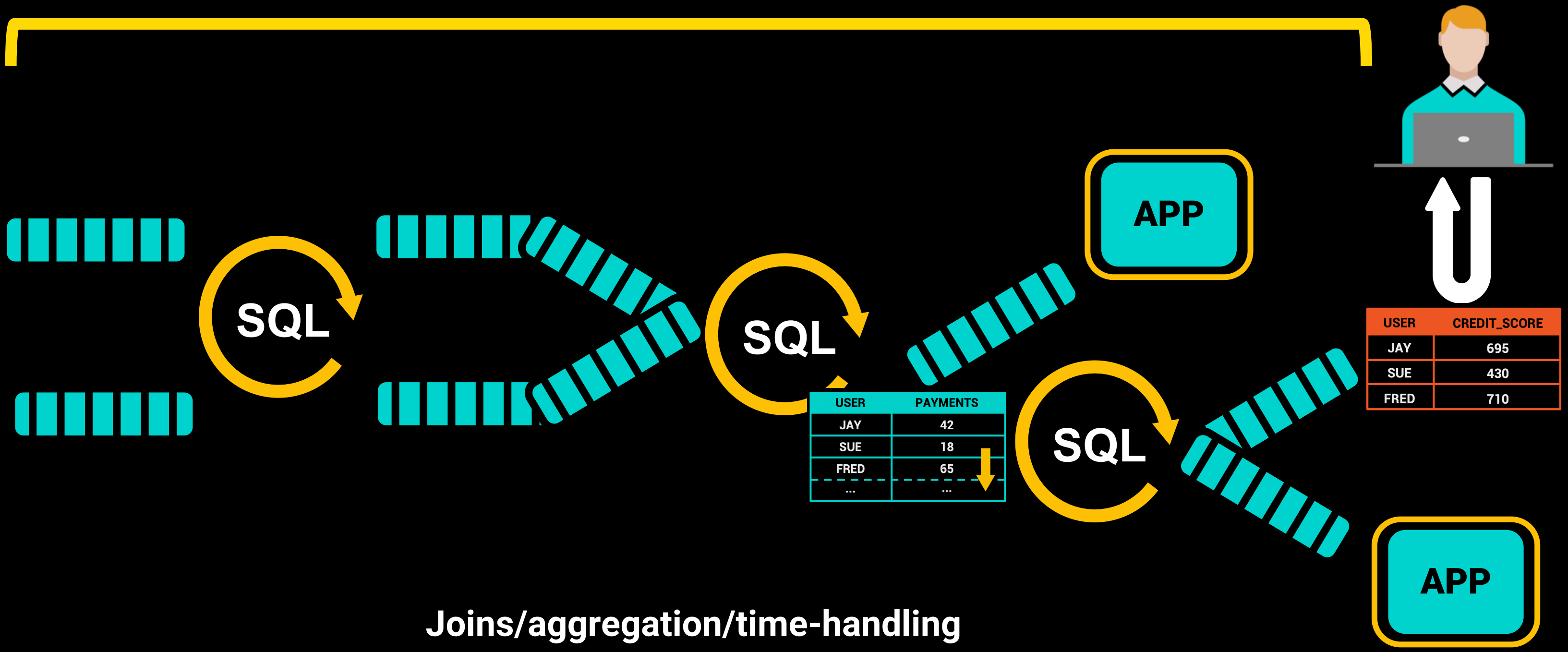
```
SELECT user, credit_score
  FROM orders
  WHERE ROWKEY = ‘bob’
  EMIT CHANGES;
```

PULL

```
SELECT user, credit_score
  FROM orders
  WHERE ROWKEY = ‘bob’;
```

Asynchronous => Pipelines

Transactions



USER	PAYMENTS
JAY	42
SUE	18
FRED	65
...	...

USER	CREDIT_SCORE
JAY	695
SUE	430
FRED	710

Joins/aggregation/time-handling

Other important variants

- Stream processors are often programming frameworks today
  - Storm
  - Flink
  - Kafka Streams
- Today we have active databases that include change streams:
  - Mongo
  - Couchbase
  - RethinkDB

As Software Eats the World

THE USER OF THE SOFTWARE

IS MORE SOFTWARE

We need
Asynchronous + Synchronous
Active + Passive

We still need all of these

So is the traditional perception of “a database” enough?

Ben Stopford
Confluent
@benstopford
ben@confluent.io