FLINK SQL IN ACTION

TIMO WALTHER, SOFTWARE ENGINEER

FLINK FORWARD, BERLIN

SEPTEMBER 5, 2018

dataArtisans

ABOUT DATA ARTISANS

Original creators of Apache Flink®

Open Source Apache Flink
+ dA Application Manager
+ dA Streaming Ledger

BIG APACHE FLINK SQL USERS

Alibaba.com™

HUAWEI

lyft

UBER

yelp

FLINK'S POWERFUL ABSTRACTIONS

Layered abstractions to
navigate simple to complex use cases

```
SELECT room, TUMBLE_END(rowtime, INTERVAL '1' HOUR), AVG(temp)
FROM sensors
GROUP BY TUMBLE(rowtime, INTERVAL '1' HOUR), room
```

High-level Analytics API — **SQL / Table API (dynamic tables)**

Stream- & Batch Data Processing — **DataStream API (streams, windows)**

```
val stats = stream
   .keyBy("sensor")
   .timeWindow(Time.seconds(5))
   .sum((a, b) -> a.add(b))
```

Stateful Event-Driven Applications — **Process Function (events, state, time)**

```
def processElement(event: MyEvent, ctx: Context, out: Collector[Result]) = {
  // work with event and state
  (event, state.value) match { … }

  out.collect(…) // emit events
  state.update(…) // modify state

  // schedule a timer callback
  ctx.timerService.registerEventTimeTimer(event.timestamp + 500)
}
```

APACHE FLINK'S RELATIONAL APIS

ANSI SQL

```
SELECT user, COUNT(url) AS cnt
FROM clicks
GROUP BY user
```

LINQ-style Table API

```
tableEnvironment
  .scan("clicks")
  .groupBy('user)
  .select('user, 'url.count as 'cnt)
```

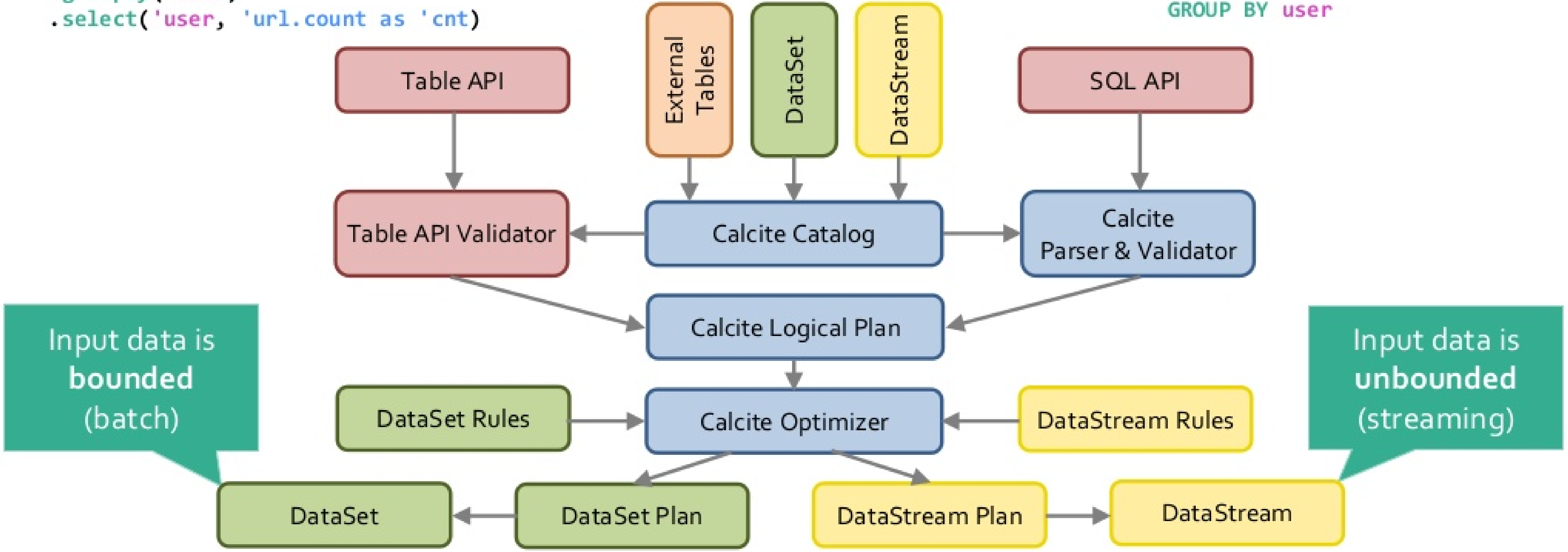
Unified APIs for batch & streaming data

A query specifies exactly the same result regardless whether its input is static batch data or streaming data.

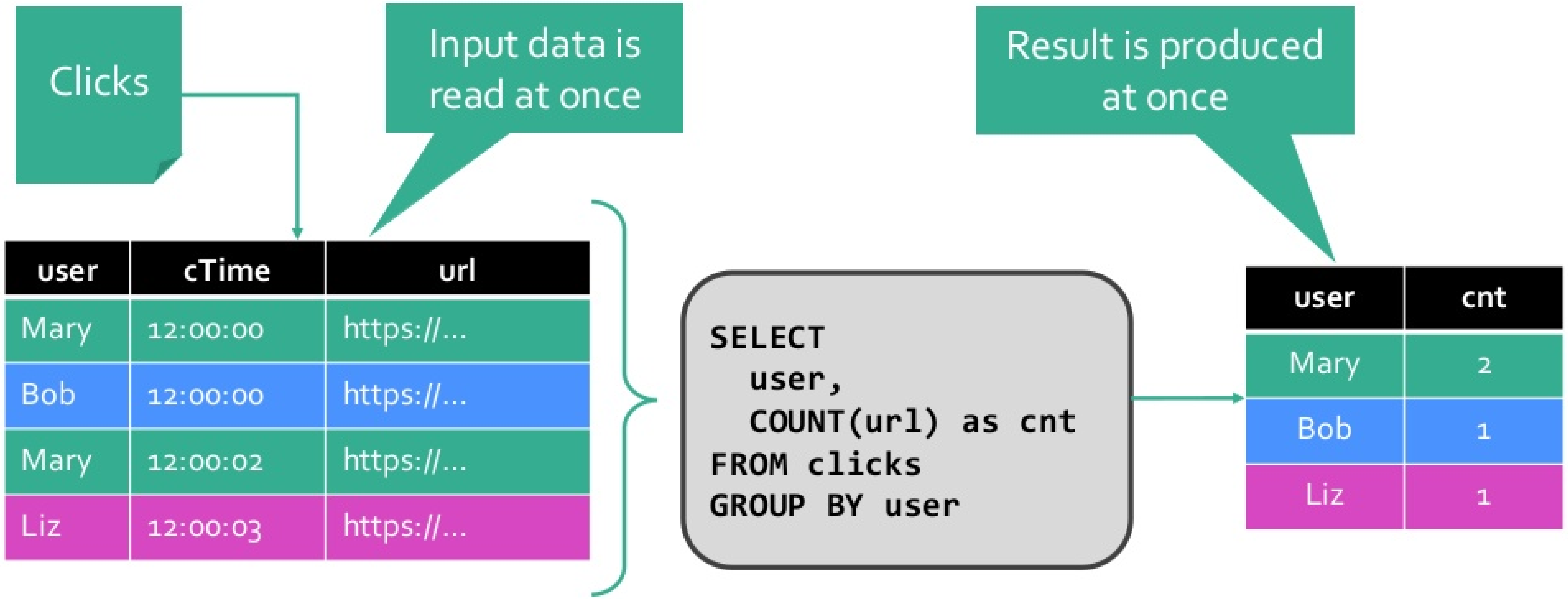
QUERY TRANSLATION

```
tableEnvironment
  .scan("clicks")
  .groupBy('user)
  .select('user, 'url.count as 'cnt)
```

```
SELECT user, COUNT(url) AS cnt
 FROM clicks
 GROUP BY user
```



WHAT IF “CLICKS” IS A FILE?

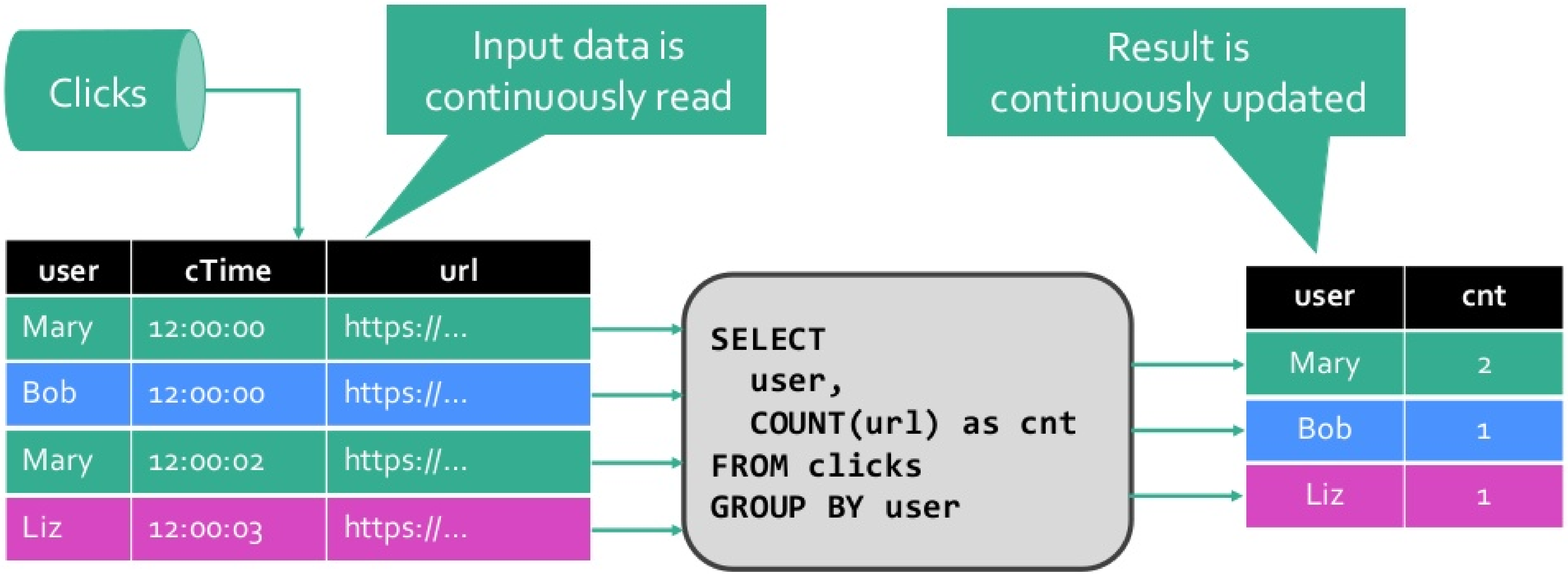


user	cTime	url
Mary	12:00:00	https://...
Bob	12:00:00	https://...
Mary	12:00:02	https://...
Liz	12:00:03	https://...

```
SELECT
  user,
  COUNT(url) as cnt
FROM clicks
GROUP BY user
```

user	cnt
Mary	2
Bob	1
Liz	1

WHAT IF "CLICKS" IS A STREAM?


Clicks
Input data is continuously read
Result is continuously updated

user	cTime	url
Mary	12:00:00	https://...
Bob	12:00:00	https://...
Mary	12:00:02	https://...
Liz	12:00:03	https://...

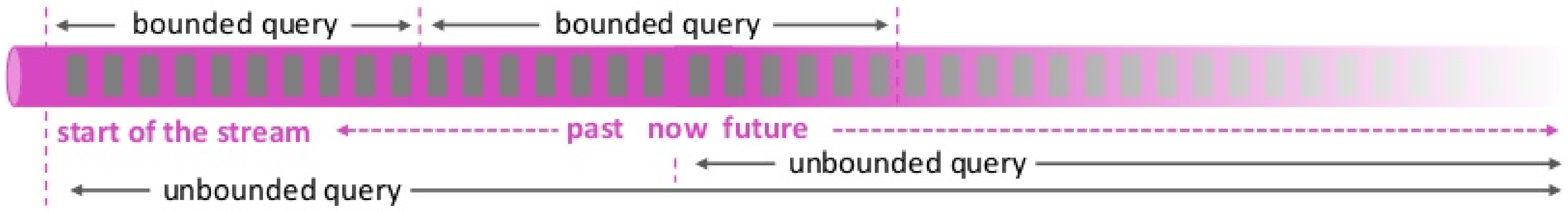
```
SELECT
  user,
  COUNT(url) as cnt
FROM clicks
GROUP BY user
```

user	cnt
Mary	2
Bob	1
Liz	1

The result is the same!

WHY IS STREAM-BATCH UNIFICATION IMPORTANT?

- Usability
  - ANSI SQL syntax: No custom "StreamSQL" syntax.
  - ANSI SQL semantics: No stream-specific results.
- Portability
  - Run the same query on *bounded* and *unbounded* data
  - Run the same query on *recorded* and *real-time* data



- How can we achieve SQL semantics on streams?

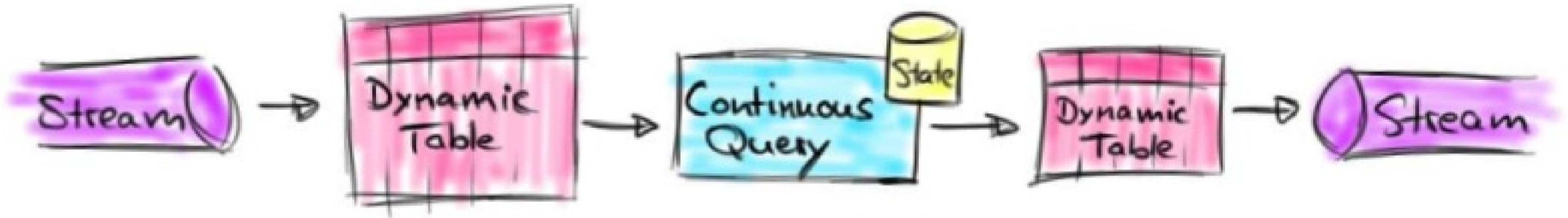
DATABASE SYSTEMS RUN QUERIES ON STREAMS

- Materialized views (MV) are similar to regular views, but persisted to disk or memory
  - Used to speed-up analytical queries
  - MVs need to be updated when the base tables change

- MV maintenance is very similar to SQL on streams
  - Base table updates are a stream of DML statements
  - MV definition query is evaluated on that stream
  - MV is query result and continuously updated

CONTINUOUS QUERIES IN FLINK

- Core concept is a *"Dynamic Table"*
  - Dynamic tables are changing over time
- Queries on dynamic tables
  - produce new dynamic tables (which are updated based on input)
  - do not terminate
- Stream ↔ Dynamic table conversions



STREAM ↔ DYNAMIC TABLE CONVERSIONS

- Append Conversions
  - Records are only inserted (appended)

```
SELECT user, url
FROM clicks
WHERE url LIKE '%xyz.com'
```

- Upsert Conversions
  - Records are upserted/deleted
  - Records have a (composite) unique key

```
SELECT user, COUNT(url)
FROM clicks
GROUP BY user
```

- Retract Conversions
  - Records are inserted/deleted

SQL FEATURES

SQL FEATURE SET IN FLINK 1.6.0

- SELECT FROM WHERE
- GROUP BY / HAVING
  - Non-windowed, TUMBLE, HOP, SESSION windows
- JOIN / IN
  - Windowed INNER, LEFT / RIGHT / FULL OUTER JOIN
  - Non-windowed INNER, LEFT / RIGHT / FULL OUTER JOIN
- [streaming only] OVER / WINDOW
  - UNBOUNDED / BOUNDED PRECEDING
- [batch only] UNION / INTERSECT / EXCEPT / ORDER BY

SQL FEATURE SET IN FLINK 1.6.0

- Support for POJOs, maps, arrays, and other nested types
- Large set of built-in functions (150+)
  - LIKE, EXTRACT, TIMESTAMPADD, FROM_BASE64, MD5, STDDEV_POP, AVG, …
- Support for custom UDFs (scalar, table, aggregate)

See also:
https://ci.apache.org/projects/flink/flink-docs-master/dev/table/functions.html
https://ci.apache.org/projects/flink/flink-docs-master/dev/table/udfs.html

UPCOMING SQL FEATURES

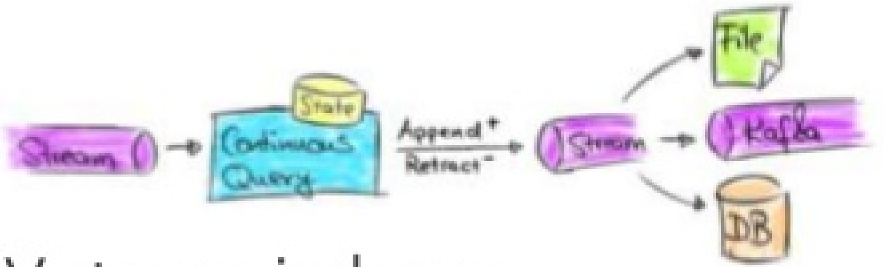
- Streaming enrichment joins (Temporal joins) [FLINK-9712]

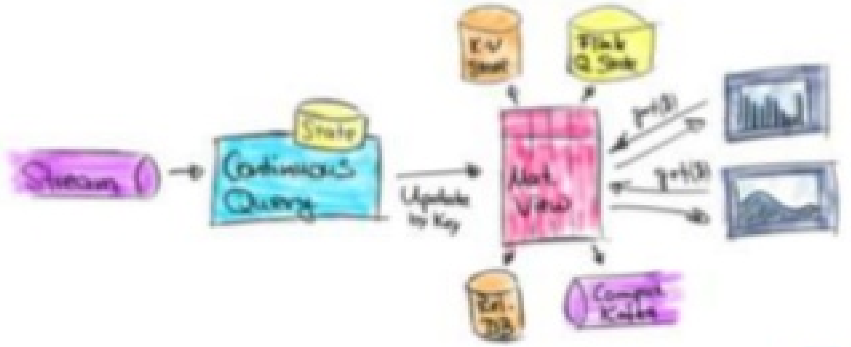
```
SELECT
  SUM(o.amount * r.rate) AS amount
FROM
  Orders AS o,
  LATERAL TABLE (Rates(o.rowtime)) AS r
WHERE r.currency = o.currency;
```

- Support for complex event processing (CEP) [FLINK-6935]
  - MATCH_RECOGNIZE
- More connectors and formats [FLINK-8535]

WHAT CAN I BUILD WITH THIS?

- Data Pipelines
  - Transform, aggregate, and move events in real-time
- Low-latency ETL
  - Convert and write streams to file systems, DBMS, K-V stores, indexes, …
  - Ingest appearing files to produce streams
- Stream & Batch Analytics
  - Run analytical queries over bounded and unbounded data
  - Query and compare historic and real-time data
- Power Live Dashboards
  - Compute and update data to visualize in real-time



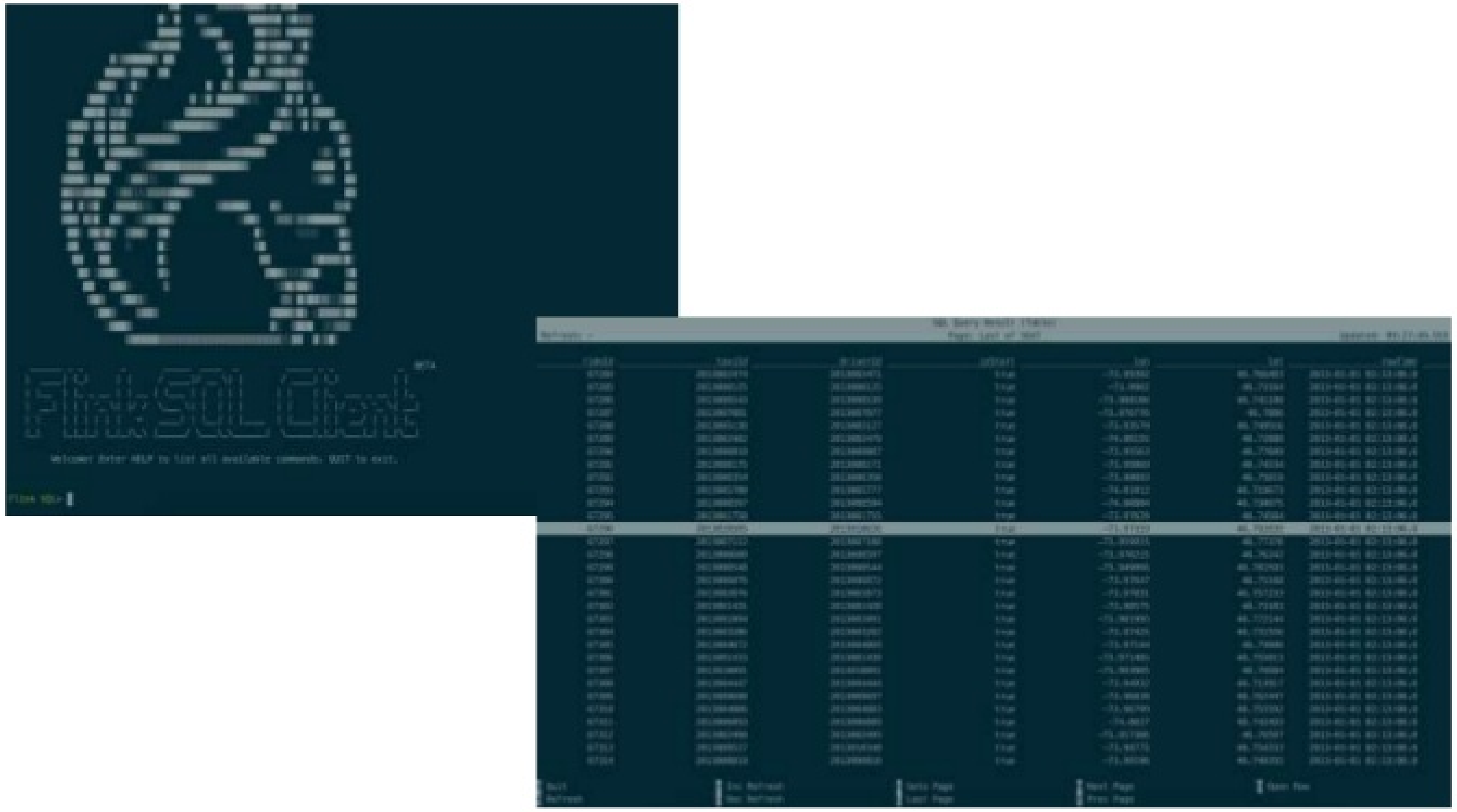


SQL CLIENT BETA

dataArtisans

INTRODUCTION TO SQL CLIENT

- Newest member of the Flink SQL family (since Flink 1.5)



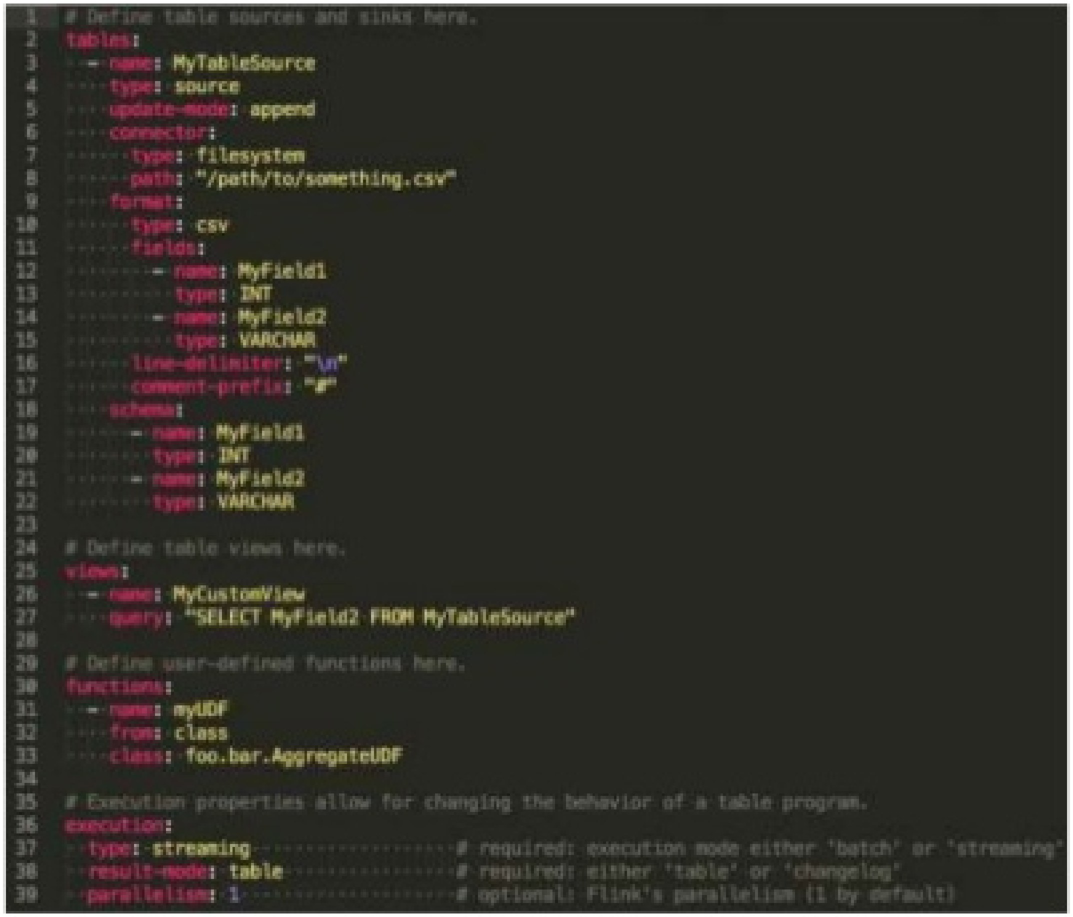
```
Welcome! Enter HELP to list all available commands. QUIT to exit.

Flink SQL> 
```

```
SQL Query Result (Table)
[illegible]
```

INTRODUCTION TO SQL CLIENT

- Goal: Flink without a single line of code
  - only SQL and YAML
  - *"drag&drop"* SQL JAR files for connectors and formats
- Build on top of Flink's Table & SQL API
- Useful for prototyping & submission

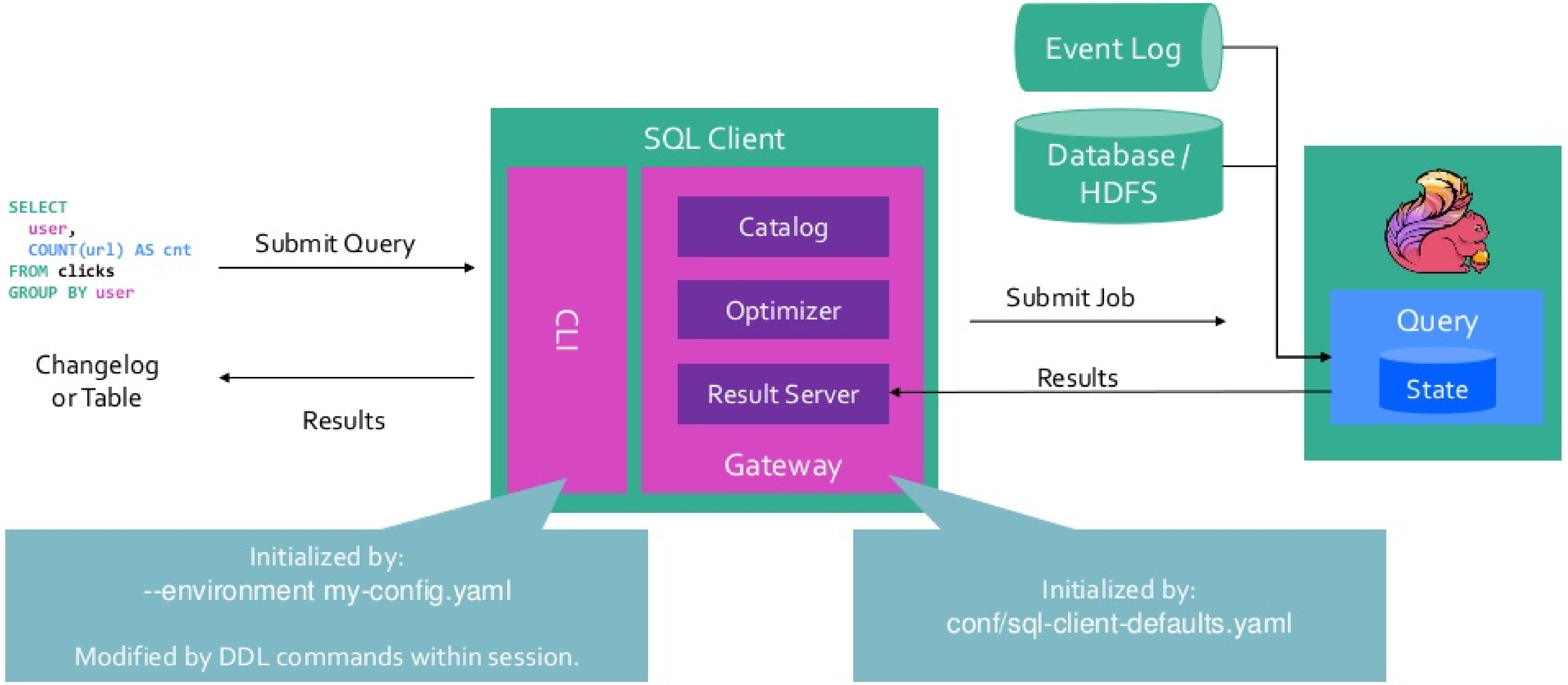
SQL CLIENT CONFIGURATION



```yaml
# Define table sources and sinks here.
tables:
  - name: MyTableSource
    type: source
    update-mode: append
    connector:
      type: filesystem
      path: "/path/to/something.csv"
    format:
      type: csv
      fields:
        - name: MyField1
          type: INT
        - name: MyField2
          type: VARCHAR
      line-delimiter: "\n"
      comment-prefix: "#"
    schema:
      - name: MyField1
        type: INT
      - name: MyField2
        type: VARCHAR

# Define table views here.
views:
  - name: MyCustomView
    query: "SELECT MyField2 FROM MyTableSource"

# Define user-defined functions here.
functions:
  - name: myUDF
    from: class
    class: foo.bar.AggregateUDF

# Execution properties allow for changing the behavior of a table program.
execution:
  type: streaming                    # required: execution mode either 'batch' or 'streaming'
  result-mode: table                 # required: either 'table' or 'changelog'
  parallelism: 1                     # optional: Flink's parallelism (1 by default)
```

See also:
https://ci.apache.org/projects/flink/flink-docs-master/dev/table/sqlClient.html

PLAY AROUND WITH FLINK SQL



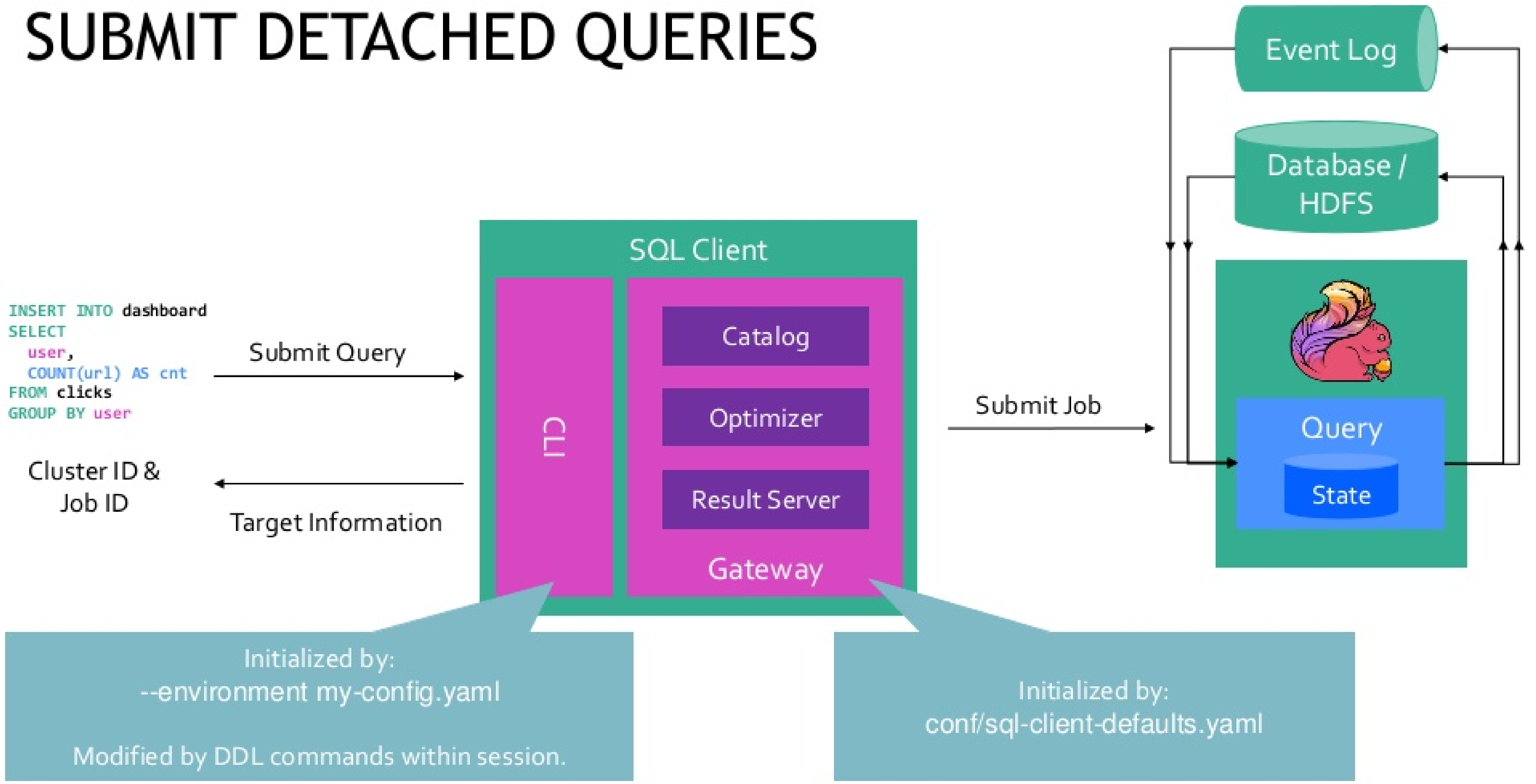
```
SELECT
  user,
  COUNT(url) AS cnt
FROM clicks
GROUP BY user
```

Initialized by:
--environment my-config.yaml

Modified by DDL commands within session.

Initialized by:
conf/sql-client-defaults.yaml

SUBMIT DETACHED QUERIES



```
INSERT INTO dashboard
SELECT
  user,
  COUNT(url) AS cnt
FROM clicks
GROUP BY user
```

Initialized by:
--environment my-config.yaml

Modified by DDL commands within session.

Initialized by:
conf/sql-client-defaults.yaml

ACTION TIME!

HTTPS://GITHUB.COM/DATAARTISANS/SQL-TRAINING

dataArtisans

SUMMARY

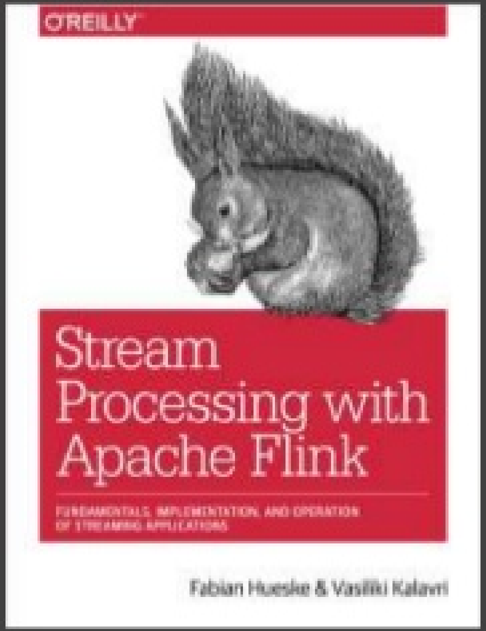
- Unification of stream and batch is important.

- Flink's SQL solves many streaming and batch use cases.
- Runs in production at Alibaba, Uber, and others.

- The community is working on improving user interfaces.
- Get involved, discuss, and contribute!

THANK YOU!

@twalthr
@dataArtisans
@ApacheFlink


O'REILLY
Stream Processing with Apache Flink
FUNDAMENTALS, IMPLEMENTATION, AND OPERATION OF STREAMING APPLICATIONS
Fabian Hueske & Vasiliki Kalavri

Available on O'Reilly Early Release!

WE ARE HIRING

data-artisans.com/careers

dataArtisans