Deduplicating And Analysing Time-Series Data With Apache Beam And QuestDB
About me: I like databases & open source

2006-2012 - Web developer

- MySQL, Redis, PostgreSQL, Sqlite, ElasticSearch

late nineties to 2005. Desktop/CGI/Servlets/ EJBs/CORBA

- MS Access, MySQL, Oracle, Sybase, Informix

The pre-SQL years

As a student/hobbyist (late eighties - early nineties)

- Amsbase, DBase III, DBase IV, Foxpro, Microsoft Works, Informix

The licensed SQL period

The libre and open SQL revolution / The NoSQL rise

2013-2018. Data Engineer/Big Data & Analytics consultant

- PostgreSQL, Redis, Neo4j, Google BigQuery, BigTable, Google Cloud Spanner, Apache Spark, Apache BEAM, Apache Flink, HBase, MongoDB, Presto

The hadoop dark ages / The python hegemony / The cloud database big migrations

2019-2022. Data & Analytics specialist at a cloud provider

- Amazon Aurora, Neptune, Athena, Timestream, DynamoDB, DocumentDB, Kinesis Data Streams, Kinesis Data Analytics, Redshift, ElastiCache for Redis, QLDB, ElasticSearch, OpenSearch, Cassandra, Spark...

The streaming era / The database as a service singularity

2022- today. Developer relations at an open source database vendor

- QuestDB, PostgreSQL, MongoDB, Timescale, InfluxDB, Apache Flink
Agenda

- The problem of data duplication
- The problem of data duplication
- The problem of data duplication
- The problem of data duplication
- Behold: a dashboard!
- The many challenges of time-series data
- QuestDB to the rescue
- Down the rabbit hole of writing a custom BEAM Sink
  - Finding several needles on a documentation haystack
  - When I sadly discovered Python streaming support is meh
  - The unsung hero saves the day (again): implementing the Sink in Java
HOW
WHAT
My lazy approach to choosing a database

If you can use only one database for everything, go with PostgreSQL*

* Or any other major and well supported RDBMS
Imagine…

- a factory floor with 500 machines, or
- a fleet with 500 vehicles, or
- 50 trains, with 10 cars each, or
- 500 users with a mobile phone, or
- 500 financial instruments generating tick data

...sending data every second
43,200,000 rows a day....
302,400,000 rows a week....
1,314,144,000 rows a month
Rashid Ashraf
@rshidashrf

Elon Musk: I'm putting people on Mars!
Developer: Fantastic, more timezone to support.

#ElonMusk

4:57 AM · Mar 19, 2018
Time-series analytics in a nutshell

Working with timestamped data in a database is tricky*

* specially working with analytics of data changing over time or at a high rate
We’d like to be known for

- Performance
  - Better performance with smaller machines

- Developer Experience

- Proudly Open Source (Apache 2.0)
A quick overview of some interesting queries
WHERE ... TIME RANGE

SELECT * from trips WHERE pickup_datetime in '2018';
SELECT * from trips WHERE pickup_datetime in '2018-06';
SELECT * from trips WHERE pickup_datetime in '2018-06-21T23:59';

SELECT * from trips WHERE pickup_datetime in '2018;2M' LIMIT -10;
SELECT * from trips WHERE pickup_datetime in '2018;10s' LIMIT -10;
SELECT * from trips WHERE pickup_datetime in '2018;-3d' LIMIT -10;

SELECT * from trips WHERE pickup_datetime in '2018-06-21T23:59:58;4s;1d;7'
SELECT * from trips WHERE pickup_datetime in '2018-06-21T23:59:58;4s;-1d;7'
Aggregates data in homogeneous time chunks

```
SELECT timestamp, min(tempF),
    max(tempF), avg(tempF)
FROM weather SAMPLE BY 1M;
```

```
SELECT
    timestamp,
    sum(price * amount) / sum(amount) AS vwap_price,
    sum(amount) AS volume
FROM trades
WHERE symbol = 'BTC-USD' AND timestamp > dateadd('d', -1, now())
SAMPLE BY 15m ALIGN TO CALENDAR;
```
Can fill missing time chunks using different strategies (NULL, constant, LINEAR, PREVIOUS value)

```
SELECT
  timestamp,
  sum(price * amount) / sum(amount) AS vwap_price,
  sum(amount) AS volume
FROM trades
WHERE symbol = 'BTC-USD' AND timestamp > dateadd('d', -1, now())
SAMPLE BY 1s FILL(NULL) ALIGN TO CALENDAR;
```
Retrieves the latest entry by timestamp for a given key or combination of keys, for scenarios where multiple time series are stored in the same table.

```
SELECT * FROM trades
WHERE symbol in ('BTC-USD', 'ETH-USD')
LATEST ON timestamp PARTITION BY symbol, side;
```
ASOF JOIN / LT JOIN

SPLICE JOIN

ASOF JOIN joins two different time-series measured. For each row in the first time-series, the ASOF JOIN takes from the second time-series a timestamp that meets both of the following criteria:

- The timestamp is the closest to the first timestamp.
- The timestamp is strictly prior or equal to the first timestamp.

```sql
WITH trips2018 AS (  
    SELECT * from trips WHERE pickup_datetime in '2016'
)  
SELECT pickup_datetime, fare_amount, tempF, windDir  
FROM trips2018  
ASOF JOIN weather;
```
Building a Sink connector

QuestDB cannot do in-stream deduplications.

Apache BEAM can help
The Python QuestDB Sink

- **WriteToQuestDB(PTransform) class**
  - Receives the args you need to pass to the sink
  - Implements the `expand` method, which receives the PCollection then invokes ParDo to `_WriteTOQuestDBFn`

- **_WriteToQuestDBFn(DoFn) class**
  - Instantiates `_QuestDBSink` on `start_bundle`
  - Flushes/releases `_QuestDBSink` on `finish_bundle`
  - Implements `display_data` to show info on the UI
  - Calls to `_QuestDBSink.write` on the `process` method

- **_QuestDBSink class**
  - Deals with the QuestDB connection itself
The Python QuestDB Sink

https://github.com/javier/questdb-beam/tree/main/python

```python
coll | WriteToQuestDB(table,
symbols=[list_of_symbols],
columns=[list_of_columns],
host=host,
port=port,
batch_size=optionalSizeOfBatch,
tls=optionalBoolean,
auth=optionalAuthDict)
```
I am dead inside.
The Java QuestDB Sink

- **QuestDbIO.Write class, extends PTransform**
  - Receives the args you need to pass to the sink
  - Uses @AutoValue to generate classes “magically”
  - Implements the `expand` method, which receives the PCollection then invokes ParDo to `QuestDbIO.Write.WriteFn` (with optional deduplication)
  - Implements `populateDisplayData`

- **QuestDbIO.Write.WriteFn class, extends DoFn**
  - Instantiates `QuestDBSender` on `start_bundle`
  - Flushes/closes `QuestDBSender` on `finish_bundle`
  - Parses/sends the `QuestDbRow` to `QuestDB` on the `process` method
keydAndWindowed = (PCollection) input.apply(WithKeys.of(new SerializableFunction<QuestDbRow, String>() {
    @Override
    public String apply(QuestDbRow r) {
        return String.valueOf(r.hashCode());
    }
}));

PCollection windowedItems = (PCollection)
    keydAndWindowed.apply(
        Window.<KV<String, String>>into(
            Sessions.
                withGapDuration(
                    Duration.standardSeconds(deduplicationDurationMillis()))
        )
    );

PCollection<QuestDbRow> uniqueRows = (PCollection<QuestDbRow>)
    ((PCollection) keydAndWindowed.apply(
        Deduplicate.keyedValues()
    )
).apply(Values.create());
The Java QuestDB Sink

https://github.com/javier/questdb-beam/tree/main/java

// pcoll needs to be of type QuestDbRow

pcoll.apply(ParDo.of(new LineToMapFn()));
parsedLines.apply(QuestDbIO.write()
  .withUri("your-instance-host.questdb.com:YOUR_PORT")
  .withTable("beam_demo")
  .withDeduplicationEnabled(true)
  .withDeduplicationByValue(false)
  .withDeduplicationDurationMillis(5L)
  .withSSLEnabled(true)
  .withAuthEnabled(true)
  .withAuthUser("admin")
  .withAuthToken("verySecretToken")
QUESTIONS?

Javier Ramirez
@supercoco9
https://github.com/javier/questdb-beam
https://github.com/javier/questdb-quickstart
https://github.com/questdb/questdb
https://demo.questdb.io
https://cloud.questdb.com