An Introduction to Pulsar's Database Table Abstraction

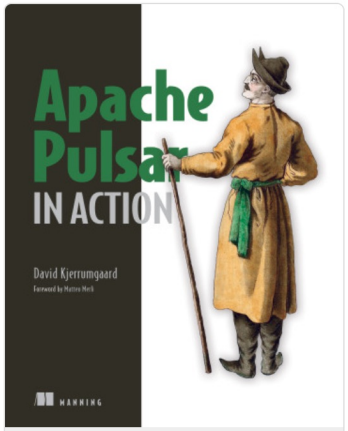
David Kjerrumgaard| Developer Advocate

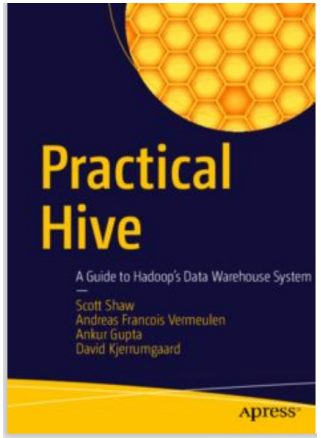
David Kjerrumgaard
Developer Advocate

- Apache Pulsar Committer
- Former Principal Software Engineer on Splunk's messaging team that is responsible for Splunk's internal Pulsar-as-a-Service platform.
- Former Director of Solution Architecture at Streamlio.
- Global practice director of Professional Services at Hortonworks.

David Kjerrumgaard
Author

- Author of **Pulsar In Action.**
- Co-Author of ***Practical Hive***





https://streamnative.io/download/manning-ebook-apache-pulsar-in-action

TableView

Pulsar's Database Table Abstraction

Why a Table Abstraction?

- In many use cases, applications want to consume data from a Pulsar Topic as if it were a database table, where each new message is an “update” to the table.

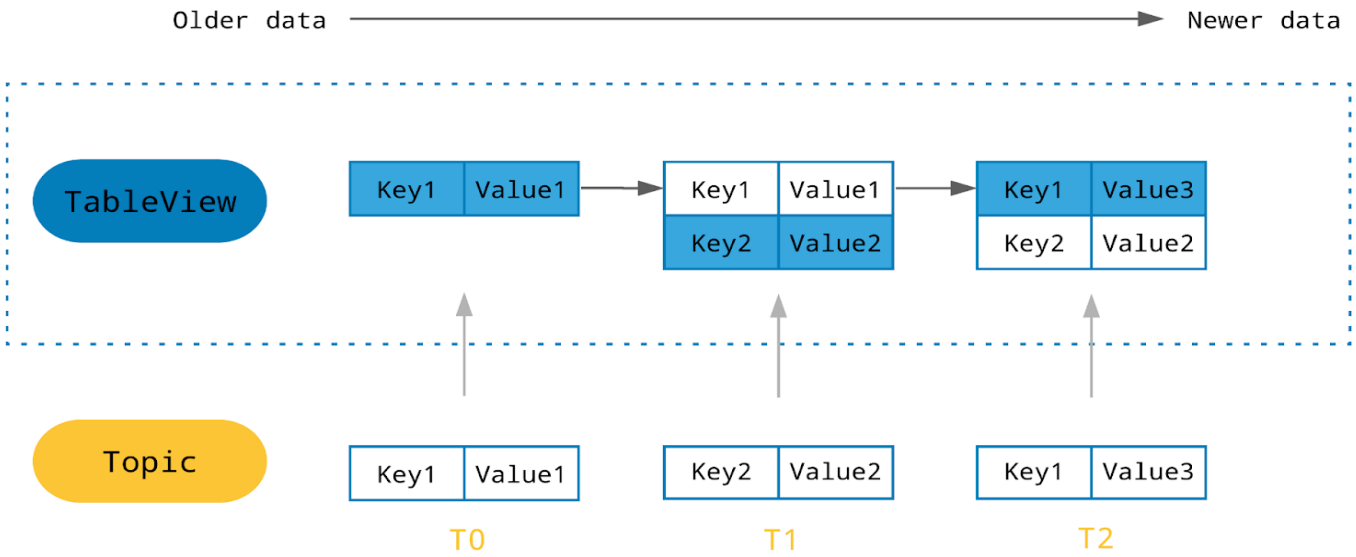
- Up until now, applications used Pulsar consumers or readers to fetch all the updates from a topic and construct a map with the latest value of each key for received messages.

- The Table Abstraction provides a standard implementation of this message consumption pattern for any given keyed topic.

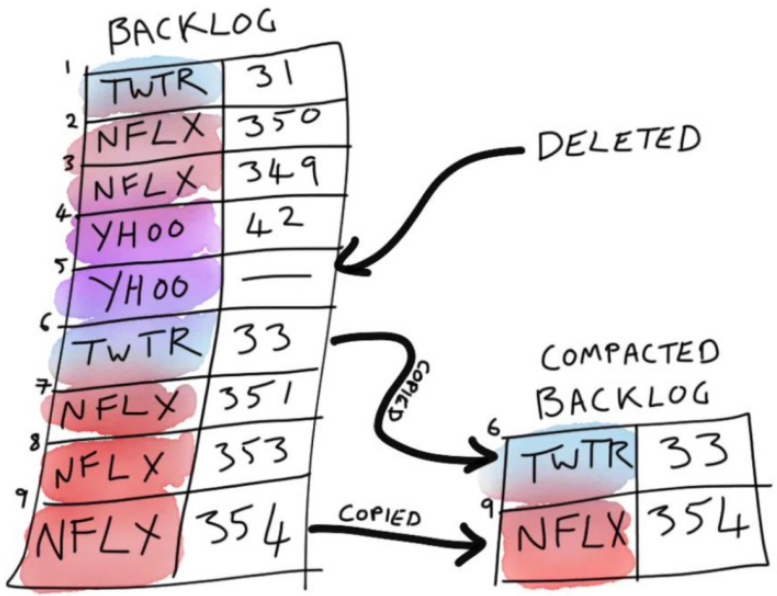
TableView

- New Consumer type added in Pulsar 2.10 that provides a continuously updated key-value map view of compacted topic data.
- An abstraction of a changelog stream from a primary-keyed table, where each record in the changelog stream is an update on the primary-keyed table with the record key as the primary key.
- READ ONLY DATA STRUCTURE!

What is it?


Older data
Newer data
TableView
Key1
Value1
Key1
Value1
Key2
Value2
Key1
Value3
Key2
Value2
Topic
Key1
Value1
Key2
Value2
Key1
Value3
T0
T1
T2

How does it work?



- When you create a TableView, and additional compacted topic is created.
- In a compacted topic, only the most recent value associated with each key is retained.
- A background reader consumes from the compacted topic and updates the map when new messages arrive.

Working with TableView

Building a TableView

```
TableView<StockQuote> stockQuoteTable = pulsarClient
  .newTableViewBuilder(Schema.JSON(StockQuote.class))
     .autoUpdatePartitionsInterval(2, TimeUnit.SECONDS)
     .topic("persistent://public/default/stock-quotes")
     .create();
```

Reading from a TableView

Method	Description
`get(String key)`	Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
`forEach( BiConsumer<String,T> action)`	Performs the given action for each entry in this map until all entries have been processed or the action throws an exception.
`forEachAndListen( BiConsumer<String,T> action)`	Performs the given action for each entry in this map and registers a listener which triggers the action to be performed when the table is updated.

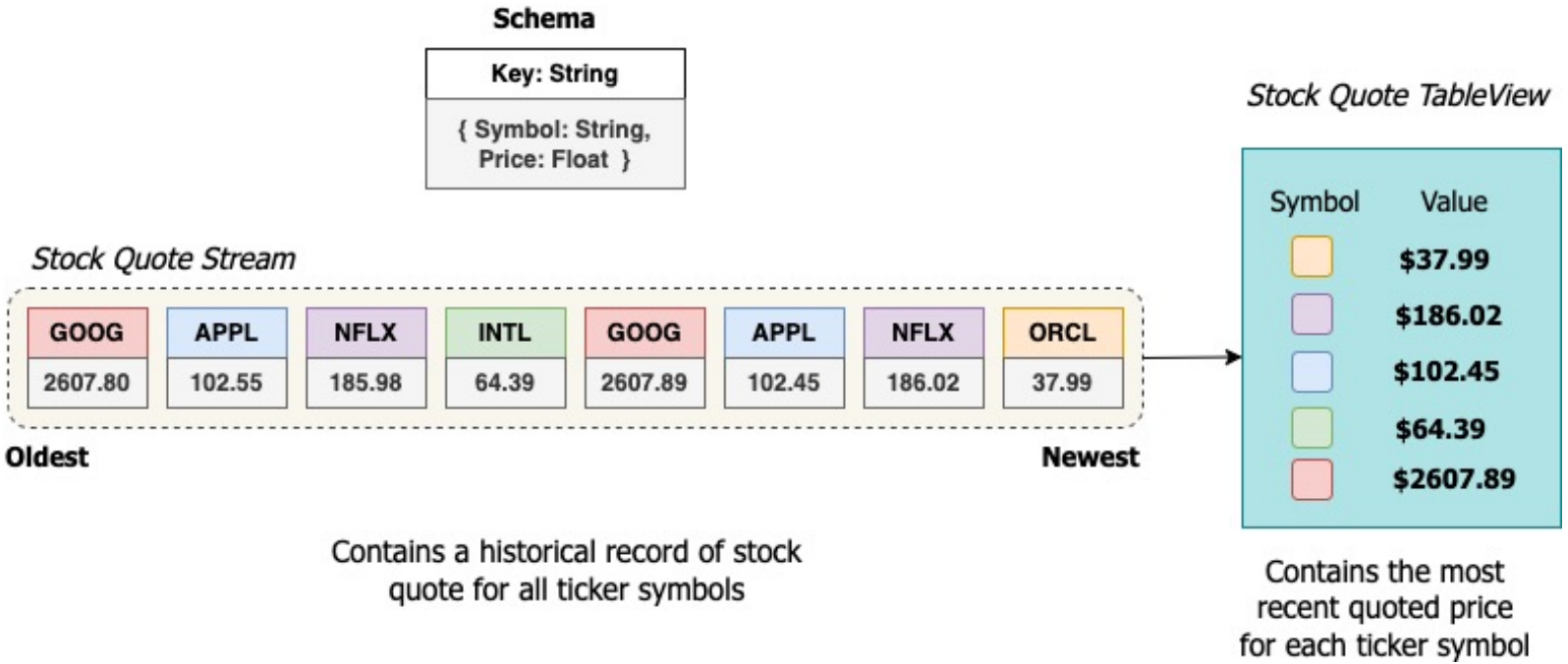
How to use the API?

- The get() method can be used to join a table to an active stream consumer.
- The forEach() method can be used to scan the table and perform an action on the contents one time.
- The forEachAndListen() method can be used to scan the table and perform an action on the contents periodically.

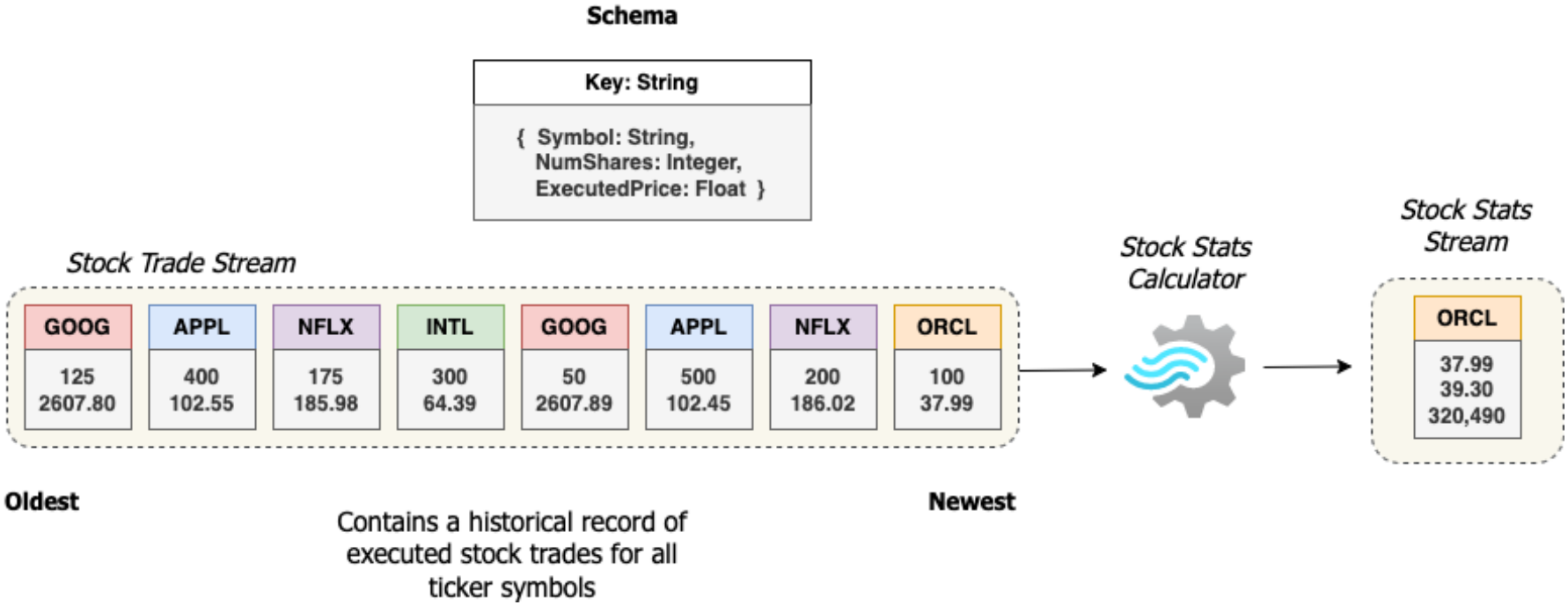
Use Case

Stock Trading Platform

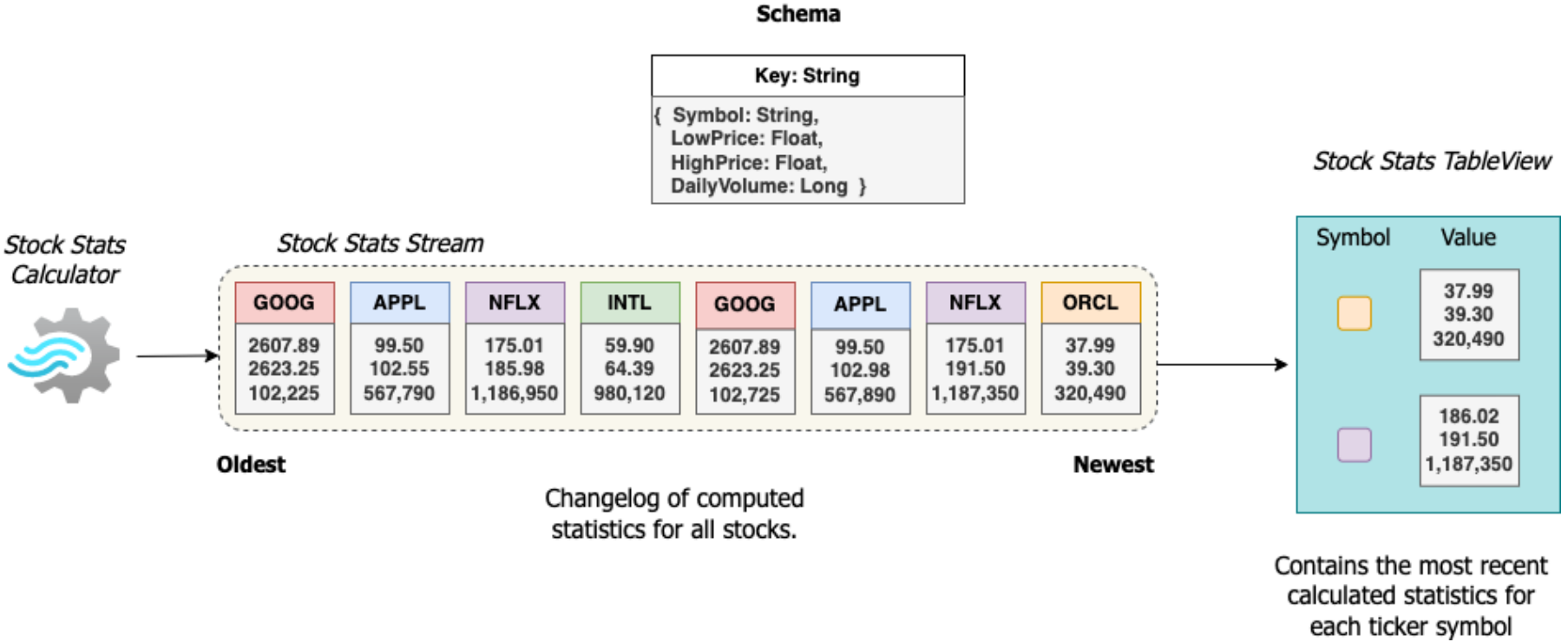
Stock Quotes Topic



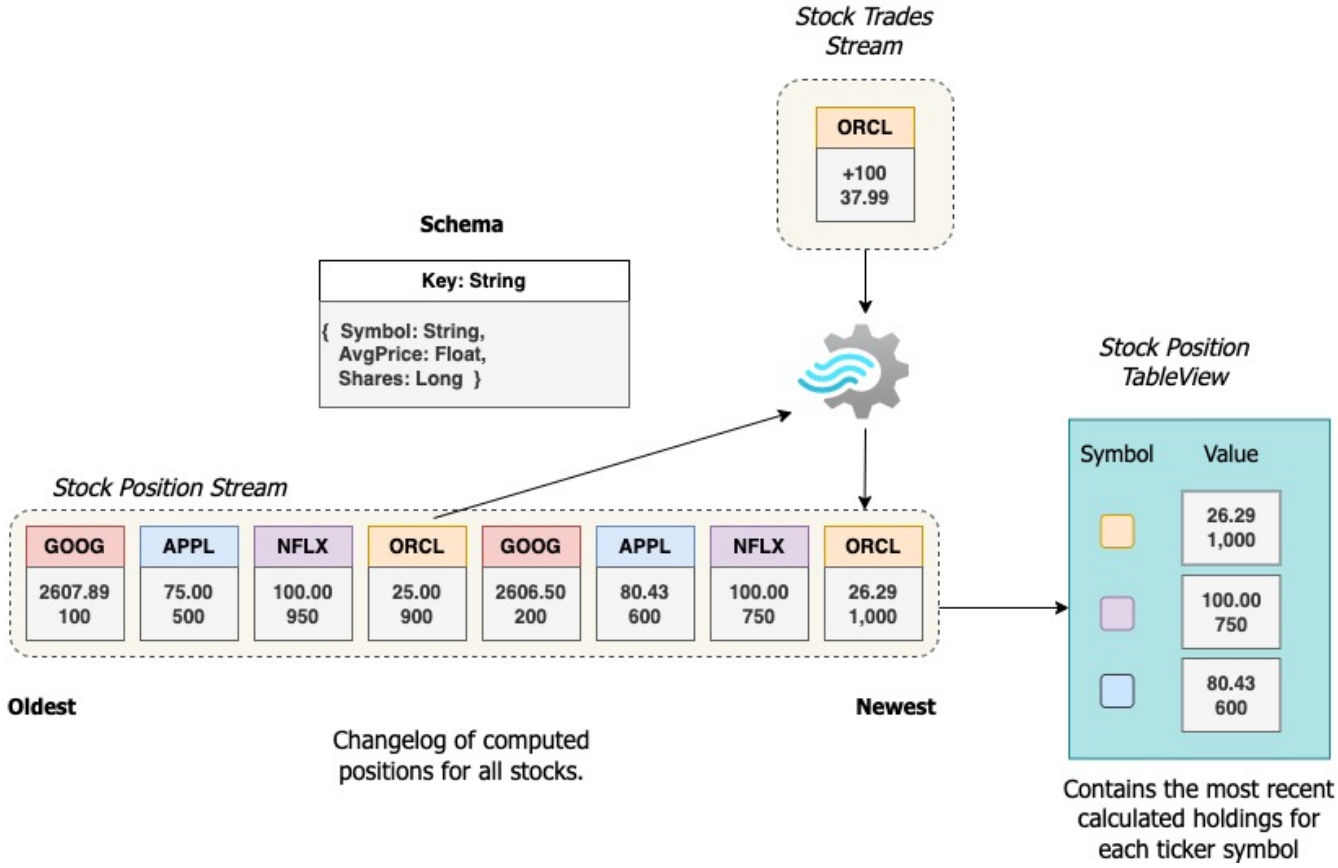
Stock Trades Topic



Stock Statistics Topic



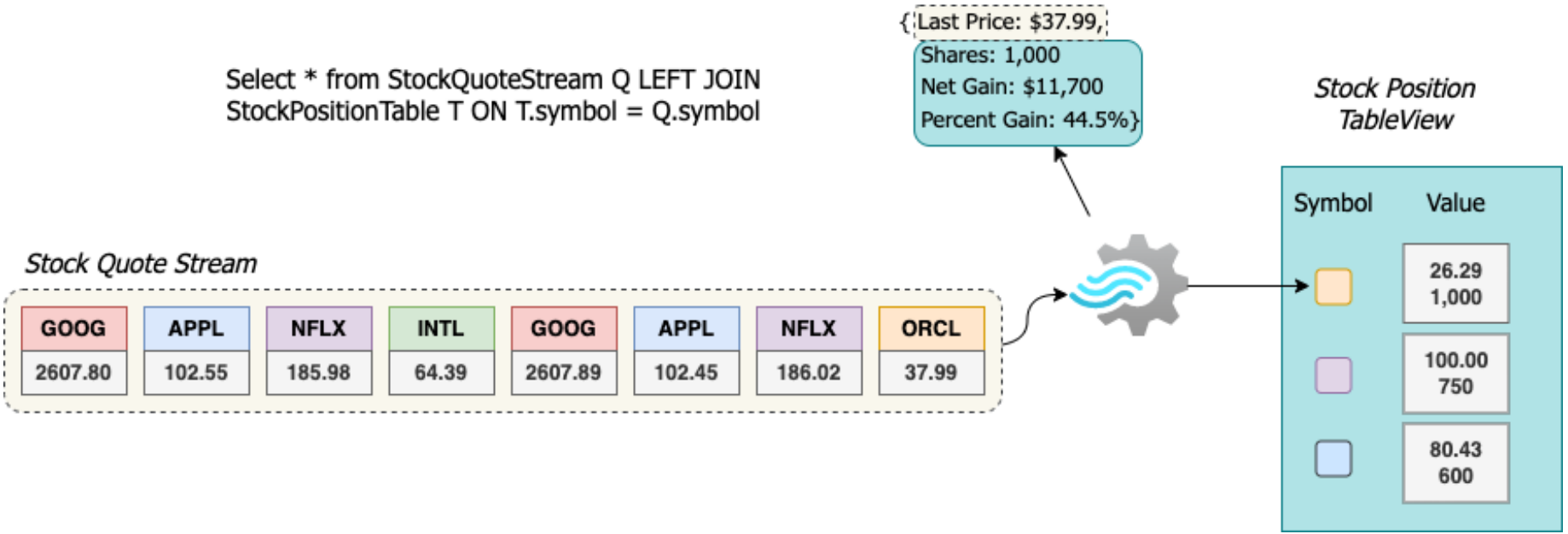
Stock Positions Topic



TableView Patterns

- Table Lookup
- Single Scan.
- Periodic Scan.

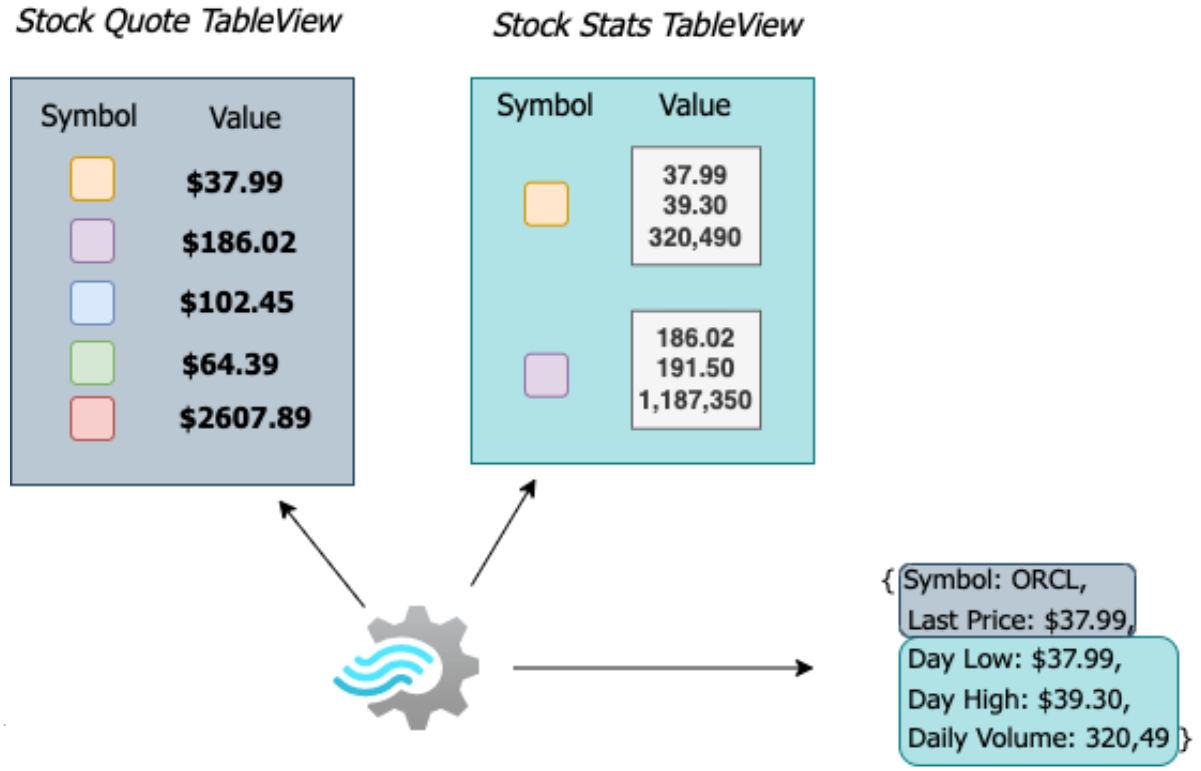
Join Streams to TableView



```
public void join() throws PulsarClientException {
    table = getClient().newTableViewBuilder(Schema.JSON(StockPosition.class))
            .autoUpdatePartitionsInterval( interval: 2, TimeUnit.SECONDS)
            .topic(tableTopic)
            .create();

    stockQuoteStream = getClient().newConsumer(Schema.JSON(StockQuote.class))
            .subscriptionName("portfolio")
            .topic(streamTopic)
            .messageListener((con, msg) -> {
                StockPosition position = table.get(msg.getValue().getTickerSymbol());
                if (position != null) {
                    System.out.println(String.format
                            ("%s [ Last Trade: $%.2f  Purchase Price: $%.2f  Shares: %,d  Net Gain: $%,.2f ]",
                                    msg.getValue().getTickerSymbol(), msg.getValue().getQuotePrice(),
                                    position.getPurchasePrice(), position.getQuantity(),
                                    ((msg.getValue().getQuotePrice()) - position.getPurchasePrice()) *
                                            position.getQuantity()));
                }
                try {
                    con.acknowledge(msg);
                } catch (PulsarClientException e) {
```

Join TableView to TableView



Select * from StockQuoteTable Q LEFT JOIN StockStatsTable T ON T.symbol = Q.symbol

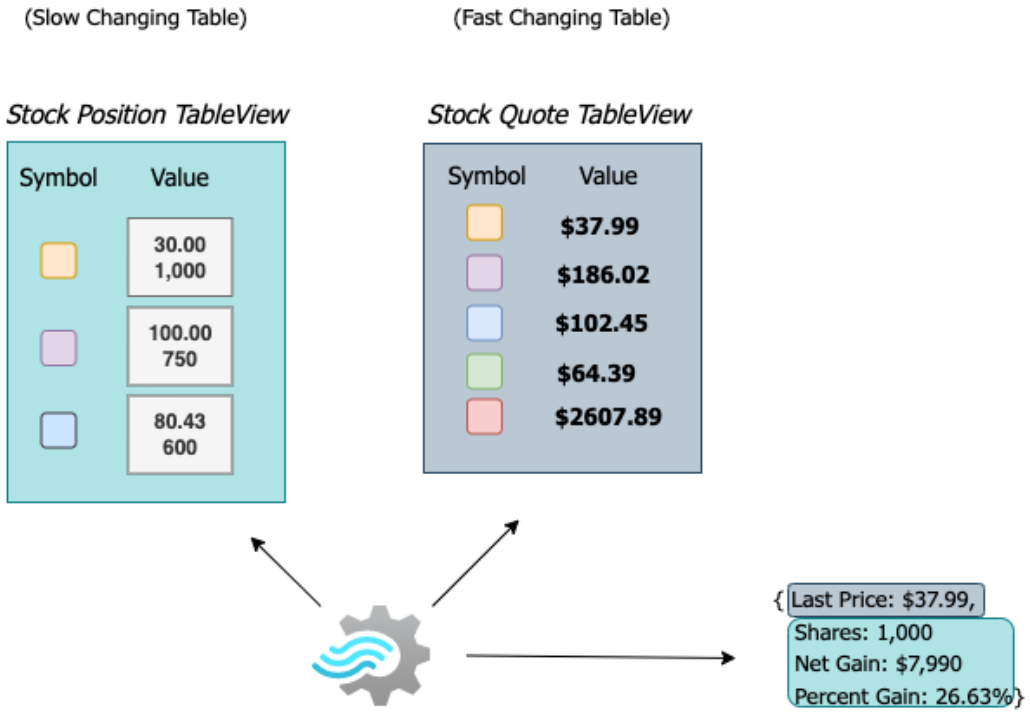
```
public void join() throws PulsarClientException {
    stockQuoteTable = getClient().newTableViewBuilder(Schema.JSON(StockQuote.class))
            .autoUpdatePartitionsInterval( interval: 2, TimeUnit.SECONDS)
            .topic(stockQuoteTopic)
            .create();

    stockStatisticsTable = getClient().newTableViewBuilder(Schema.JSON(StockStatistics.class))
            .autoUpdatePartitionsInterval( interval: 2, TimeUnit.SECONDS)
            .topic(stockStatsTopic)
            .create();

    stockQuoteTable.forEach((symbol, quote) -> {
        StockStatistics stats = stockStatisticsTable.get(symbol);

        if (stats != null) {
            System.out.println(String.format("%s [Quote: %.2f Low: %.2f  High: %.2f  Volume: %,d  Time: %
                    symbol, quote.getQuotePrice(), stats.getLowValue(),
                    stats.getHighValue(), stats.getTradeVolume(), stats.getTimestamp()));
        }
    });
}
```

Continuous TableView Join



Select * from StockPositionTable Q LEFT JOIN
StockQuoteTable T ON T.symbol = Q.symbol

```
public void join() throws PulsarClientException {
    getStockPositionTable().forEachAndListen((symbol, position) -> {
        StockQuote quote = null;
        try {
            quote = getStockQuoteTable().get(symbol);
        } catch (PulsarClientException e) {
            e.printStackTrace();
        }

        if (quote != null) {
            System.out.println(String.format
                    ("%s [ Last Trade: $%.2f  Purchase Price: $%.2f  Shares: %,d  Net Gain:
                            symbol, quote.getQuotePrice(), position.getPurchasePrice(),
                            position.getQuantity(),
                            ((quote.getQuotePrice()) - position.getPurchasePrice()) * posit
        }
    });
}
```

DEMO TIME

- Demo 1: Stream/Table Join (Lookup)
- Demo 2: Table/Table Join (Single Scan)
- Demo 3: Table/Table join (Periodic Scan)

https://github.com/david-streamlio/table-view-demo

Summary

- Pulsar's new TableView API provides a database table abstraction on top of Topic data.
- I demonstrated how it can be used to emulate SQL joins with Streams or other TableViews.
  - Join Streams to TableViews
  - Join TableViews to Other TableViews

Thanks for attending!

Scan the QR Code to learn more about Apache Pulsar.

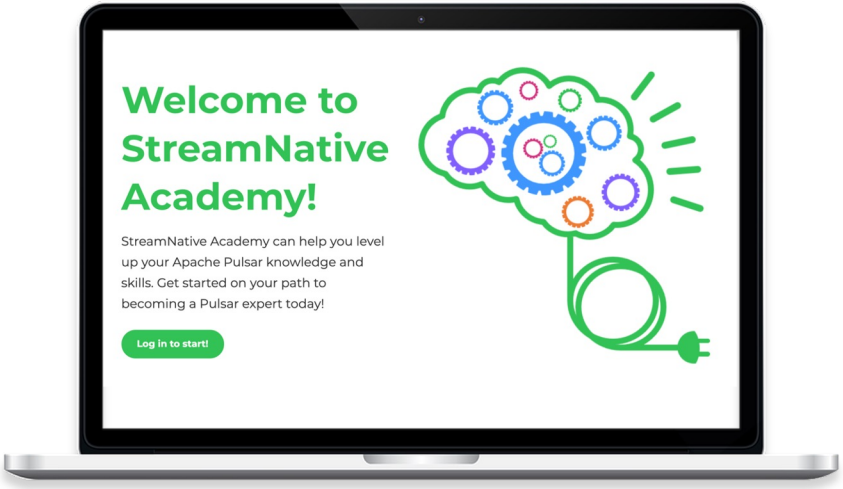
Explore the Code:

https://github.com/david-streamlio/table-view-demo

Now Available

On-Demand Pulsar Training

Academy.StreamNative.io



We're Hiring

streamnative.io/careers/

Questions

Let’s Keep in Touch!

David Kjerrumgaard

Developer Advocate

@Dkjerrumg1

https://www.linkedin.com/davidkj

https://github.com/david-streamlio

StreamNative