How OLTP to OLAP
Archival Demystified

ALKIN TEZUYSAL @ask_dba
SCALE 20X Pasadena, CA Mar 2023

@ChistaDATA Inc. 2023
Let’s get connected with Alkin first

Alkin Tezuysal - EVP - Global Services @chistadata

- Linkedin: https://www.linkedin.com/in/askdba/
- Twitter: https://twitter.com/ask_dba
- Blog: https://askdba.net/blog/

Open Source Database Evangelist

- Previously PlanetScale, Percona and Pythian as Technical Manager, SRE, DBA
- Previously Enterprise DBA, Informix, Oracle, DB2, SQL Server
Catching winds
@svrubato

Born to Sail, Forced to Work!
@ChistaDATA Inc. 2023
Born to Sail, Forced to Work!

@ChistaDATA Inc. 2023
TURKEY EARTHQUAKE

@ChistaDATA Inc. 2023
About ChistaDATA Inc.

Founded in 2021 by Shiv Iyer - CEO and Principal
Has received 3M USD seed investment (2021)
Focusing on ClickHouse infrastructure engineering and performance operations
What’s ClickHouse anyway?
Services and Products around dedicated DBaaS, Managed Services, Support and Consulting

www.chistadata.io  www.chistadata.com

@ChistaDATA Inc. 2023
About ClickHouse

- Columnar Storage
- SQL Compatible
- Open Source (Apache 2.0)
- Shared Nothing Architecture
- Parallel Execution
- Rich in Aggregate Functions
- Super fast for Analytics workload
About MySQL

- Row Based Storage
- SQL Compatible
- Open Source (Apache 2.0)
- Pluggable Engine Architecture
- Multi Threaded Execution
- ACID Compliant
- De Facto standard for OLTP workloads in FOSS

@ChistaDATA Inc. 2023
Agenda

● Analytical data on OLTP databases
● Analytical data on OLAP databases
● Archiving / Streaming / CDC Use Cases
● Demo
Analytical data on OLTP databases

- Often unwanted operation but needed (Pain Points):
  - CDC
  - DWH
  - BI
  - ML/AI
  - Time Series data
  - Logging
OLTP vs OLAP

- **Indexing**
  - B-tree self balancing
- **Distribution**
  - Single (partition)
- **Size**
  - Heavy Weight
- **Maintenance**
  - Cumbersome

- **Column based**
- **Denormalized**
- **Execution**
- **Vectorized**
- **Compression**
  - Rich and many options for Time Series data
- **Distribution**
  - Sharding and Partitioning

@ChistaDATA Inc. 2023
OLTP Table Archiving Workflow

Architecture and Components:

Source Databases

- MySQL
- MariaDB
- Hadoop

Debezium connector for CDC purpose

Kafka Connector and topics

Kafka

High level Architecture ChistaData Connector

ChistaData Connector for ClickHouse

ClickHouse

Destination database

@ChistaDATA Inc. 2023
Debezium is an open source distributed platform for change data capture. Start it up, point it at your databases, and your apps can start responding to all of the inserts, updates, and deletes that other apps commit to your databases. Debezium is durable and fast, so your apps can respond quickly and never miss an event, even when things go wrong.
Debezium Operations

Bulk Operations (op: r)

```json
{
  "before":null,
  "after":{
    "a_column":1,
    "b_column":2,
    "c_column":false,
    "d_column":10424,
    "e_column":3.9,
    "f_column":111,
    "g_column":5.8,
    "h_column":6.9,
    "i_column":7,
    "j_column":8,
    "k_column":9,
    "l_column":18,
    "m_column":3856184750,
    "n_column":175921756104758,
    "o_column":a6d455ba-c97a-4484-8997-5d28d615c74
  },
  "source":{
    "version":"2.1.1.Final",
    "connector":"postgresql",
    "name":"some",
    "tsNs":1678288582088,
    "snapshot":true,
    "db":"price",
    "sequence":null,"7644674784"},
  "schema":"public",
  "table":"employees",
  "txId":7878,
  "ts":7644674784,
  "min":null
},
"op":"r",
"ts_ns":1678288582180,
"transaction":null
}
```

Insert Operations (op: c)

```json
{
  "before":null,
  "after":{
    "a_column":11,
    "b_column":10,
    "c_column":false,
    "d_column":19424,
    "e_column":3.9,
    "f_column":4,
    "g_column":5.9,
    "h_column":77.9,
    "i_column":7,
    "j_column":8,
    "k_column":9,
    "l_column":19,
    "m_column":"yens",
    "n_column":15154646838,
    "o_column":1678228758464838,
    "q_column":16988e19-1615-4b62-a174-34648209a40c
  },
  "source":{
    "version":"2.1.1.Final",
    "connector":"postgresql",
    "name":"some",
    "tsNs":1678288584845,
    "snapshot":false,
    "dp":"price",
    "sequence":null,"7644674784"},
  "schema":"public",
  "table":"employees",
  "txId":7878,
  "ts":7644674888,
  "min":null
},
"op":"c",
"ts_ns":1678288758716,
"transaction":null
}
```
Debezium Operations

Update Operations (op: u)

```json
{
  "before": {
    "a_column": 11,
    "b_column": 11,
    "c_column": false,
    "d_column": 19424,
    "e_column": 3.8,
    "f_column": 4,
    "g_column": 5.9,
    "h_column": 7.74,
    "i_column": 17,
    "j_column": 18,
    "k_column": 9,
    "l_column": "yaml",
    "m_column": "5158444839",
    "n_column": 167828875844838,
    "o_column": "a6d455ba-c97a-4a84-889f-5d28d6615c74"
  },
  "after": {
    "a_column": 13,
    "b_column": 12,
    "c_column": false,
    "d_column": 19424,
    "e_column": 3.9,
    "f_column": 4,
    "g_column": 5.9,
    "h_column": 7.74,
    "i_column": 17,
    "j_column": 16,
    "k_column": 9,
    "l_column": "yaml & str",
    "m_column": "5158444939",
    "n_column": 167828875844838,
    "o_column": "a6d455ba-c97a-4a84-889f-5d28d6615c74"
  },
  "source": {
    "version": "2.1.1 Final",
    "connector": "postgresql",
    "name": "some",
    "ts_ms": 167828888515,
    "snapshot": true,
    "db": "price",
    "sequence": 
      "\"7644678392\", \"7644678448\"
    "schema": "public",
    "table": "employees",
    "xId": 7891,
    "lsn": 7644678448,
    "xmin": null
  }
}
```

Delete Operations (op: d)

```json
{
  "before": {
    "a_column": 1,
    "b_column": 2,
    "c_column": false,
    "d_column": 19424,
    "e_column": 3.9,
    "f_column": 101,
    "g_column": 5.8,
    "h_column": 16.8,
    "i_column": 17,
    "j_column": 8,
    "k_column": 9,
    "l_column": "yaml",
    "m_column": "5158444839",
    "n_column": 1678271756184758,
    "o_column": "a6d455ba-c97a-4a84-889f-5d28d6615c74"
  },
  "after": null,
  "source": {
    "version": "2.1.1 Final",
    "connector": "postgresql",
    "name": "some",
    "ts_ms": 167828888515,
    "snapshot": true,
    "db": "price",
    "sequence": "\"7644678392\", \"7644678448\"
    "schema": "public",
    "table": "employees",
    "xId": 7891,
    "lsn": 7644678448,
    "xmin": null
  }
}
```

@ChistaDATA Inc. 2023
Converting Data Types from MySQL to ClickHouse

Data types will convert as is to the ClickHouse. Other data types will convert as `String` in ClickHouse.
Debezium Supported Databases

Available

- MongoDB
- MySQL
- PostgreSQL
- SQL Server
- Oracle
- Db2

Incubating

- Cassandra
- Vitess (MySQL Sharding Framework)
- Spanner
Debezium Operations

PostgreSQL - ClickHouse

Date > Date (This epoch times will convert via Python side.)
Timestamp > DateTime (This epoch times will convert via Python side.)

Limitations:
Time(only clock) > String
Bool > UInt8

These datatypes will be convert as is in the near future.
Kafka

Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

1. It lets you publish and subscribe to events
2. It lets you store events for as long as you want
3. It lets you process and analyze events
Kafka Use Cases

- Data Integration
- Metrics and Monitoring
- Log Aggregation
- Stream Processing
- Message Broker

@ChistaDATA Inc. 2023
Kafka Streams and Table

Streams record history

Duality

Tables represent state

aggregation

(like SUM and COUNT)

table changes

https://www.confluent.io/blog/kafka-streams-tables-part-1-event-streaming/
Streaming / Archiving / CDC
Kafka topic details c, u, d, r

```json
{
    "before": null,
    "after": {
        "id": 1004,
        "first_name": "Anne",
        "last_name": "Kretchmar",
        "email": "anne@noanswer.org"
    },
    "source": {
        "name": "dbserver1",
        "server_id": 0,
        "ts_sec": 0,
        "file": "mysql-bin.000003",
        "pos": 154,
        "row": 0,
        "snapshot": true,
        "db": "inventory",
        "table": "customers"
    },
    "op": "c",
    "ts_ms": 1486500577691
}
```
Sink Connector (ChistaDATA) - Alfa

Sink connector delivering Kafka topics to destination database.

- Currently MySQL and PostgreSQL supported.
- Improved bulk load process
- Restart handling
Sink Connector - Prerequisites

MySQL
- `log_bin` (enabled)
- `server-id` (configured)
- `binlog_format=row`
- `binlog_row_image=full`
- Table should have the PRIMARY KEY to better work with ReplacingMergeTree engine.

PostgreSQL
- WAL (Write Ahead Log)

MongoDB
- Oplog

ClickHouse
- `DataType: int as int256`
- `Engine: ReplacingMergeTree vs MergeTree`
Sink Connector - Supported Data Types

- uuid
- Numeric
- Boolean
- Int
- BigInt
- SmallInt
- MediumInt
- Varchar
- Decimal
- Text
- Float
- Double
- String
- Null
- Int with Auto Increment and Unsigned

Work in progress

- DateTime (PostgreSQL)
- Timestamp
- Schema Registry

@ChistaDATA Inc. 2023
Converting Data Types from PostgreSQL to ClickHouse

All of these data types will convert as is to the ClickHouse.
`String` in ClickHouse

<table>
<thead>
<tr>
<th>PostgreSQL</th>
<th>ClickHouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>bigint</td>
<td>int64</td>
</tr>
<tr>
<td>bigserial</td>
<td>uint64</td>
</tr>
<tr>
<td>boolean</td>
<td>bool</td>
</tr>
<tr>
<td>date</td>
<td>date</td>
</tr>
<tr>
<td>double precision</td>
<td>float64</td>
</tr>
<tr>
<td>integer</td>
<td>int32</td>
</tr>
<tr>
<td>numeric(2,0)</td>
<td>decimal(2,0)</td>
</tr>
<tr>
<td>real</td>
<td>float32</td>
</tr>
<tr>
<td>smallint</td>
<td>int16</td>
</tr>
<tr>
<td>smallserial</td>
<td>int16</td>
</tr>
<tr>
<td>serial</td>
<td>int32</td>
</tr>
<tr>
<td>text</td>
<td>string</td>
</tr>
<tr>
<td>timestamp</td>
<td>DateTime</td>
</tr>
<tr>
<td>uuid</td>
<td>uuid</td>
</tr>
</tbody>
</table>

Other data types will convert as

<table>
<thead>
<tr>
<th>PostgreSQL</th>
<th>ClickHouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit(4)</td>
<td>string</td>
</tr>
<tr>
<td>varbit(4)</td>
<td>string</td>
</tr>
<tr>
<td>box</td>
<td>string</td>
</tr>
<tr>
<td>bytea</td>
<td>string</td>
</tr>
<tr>
<td>char(5)</td>
<td>string</td>
</tr>
<tr>
<td>bit_varying(5)</td>
<td>string</td>
</tr>
<tr>
<td>cidr</td>
<td>string</td>
</tr>
<tr>
<td>circle</td>
<td>string</td>
</tr>
<tr>
<td>inet</td>
<td>string</td>
</tr>
<tr>
<td>interval</td>
<td>string</td>
</tr>
<tr>
<td>json</td>
<td>string</td>
</tr>
<tr>
<td>jsonb</td>
<td>string</td>
</tr>
<tr>
<td>line</td>
<td>string</td>
</tr>
<tr>
<td>lseg</td>
<td>string</td>
</tr>
<tr>
<td>macaddr</td>
<td>string</td>
</tr>
<tr>
<td>macaddr8</td>
<td>string</td>
</tr>
<tr>
<td>money</td>
<td>string</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
</tr>
<tr>
<td>pg_lsn</td>
<td>string</td>
</tr>
<tr>
<td>pg_snapshot</td>
<td>string</td>
</tr>
<tr>
<td>point</td>
<td>string</td>
</tr>
<tr>
<td>polygon</td>
<td>string</td>
</tr>
<tr>
<td>time</td>
<td>string</td>
</tr>
<tr>
<td>tsquery</td>
<td>string</td>
</tr>
<tr>
<td>tsvector</td>
<td>string</td>
</tr>
<tr>
<td>txid_snapshot</td>
<td>string</td>
</tr>
<tr>
<td>xml</td>
<td>string</td>
</tr>
</tbody>
</table>
DEMO Scenario

- **AirLine OnTime** DATA on MySQL
  - Previously demonstrated on MyRocks, InnoDB, MySQL, ClickHouse
  - Based on Vadim’s exploration
- We will bulk load data into MySQL
- We will attach Debezium connector MySQL
- We will let Kafka read the stream
- Sink connector will read Kafka topics and write into ClickHouse
  - Split Kafka topics to 8 threads and batches of 25K rows
Objectives of Demo

1. Show disk space utilization -
2. Show query speed -
3. Explore potential use cases -
Disk space utilisation of raw data (2018-2022)

MySQL - Single Table {ontime}
~31 million rows, ~100 columns, PK
~7Gb at filesystem level

ClickHouse - Single Table {ontime}
~31 million rows, ~100 columns, PK
~2.5 Gb at filesystem level
Monitoring

- Debezium
- Kafka
- MySQL / PostgreSQL (Source)
- ClickHouse (Destination)
- Hosts
Debezium monitoring

@ask_dba
Debezium MySQL metrics

Debezium MySQL Connector Metrics

The Debezium MySQL connector has three metric types in addition to the built-in support for JMX metrics that Zookeeper, Kafka, and Kafka Connect have.

- snapshot metrics
- binlog metrics
- schema history metrics

For more detailed information about the metrics please visit the Debezium documentation page.

@ChistaDATA Inc. 2023
Kafka Topic Overview

Message in per second

Message in per minute

Lag by Consumer Group

Message consume per minute

@ChistaDATA Inc. 2023
Q1. Avg flight duration ClickHouse

```
SELECT AVG(c1)
FROM
(
  SELECT Year,
         Month,
         count(*) AS c1
  FROM ontime_innodb_compressed3
  GROUP BY Year, Month

)

Query id: d81981dc-d627-407a-9418-b967149b4f67

avg(c1)
527567.1166666667
```

1 row in set. Elapsed: 0.040 sec. Processed 31.65 million rows, 94.96 MB (797.48 million rows/s., 2.39 GB/s.)
Q1. Avg flight duration MySQL

```sql
mysql> SELECT avg(c1) FROM ( SELECT Year, Month, count(*) AS c1 FROM time_innodb_compressed_ch_table GROUP BY Year, Month ) as ch;
+
<table>
<thead>
<tr>
<th>avg(c1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>527567.1167</td>
</tr>
</tbody>
</table>
+
1 row in set (3 min 8.16 sec)

mysql>
```
Q2. The number of flights per day from the year 2018 to 2023

```sql
SELECT DayOfWeek, count(*) AS c FROM ontime_innodb_compressed3 WHERE Year >= 2018 AND Year <= 2023 GROUP BY DayOfWeek ORDER BY c DESC;
```

<table>
<thead>
<tr>
<th>DayOfWeek</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4737017</td>
</tr>
<tr>
<td>5</td>
<td>4718538</td>
</tr>
<tr>
<td>4</td>
<td>4692468</td>
</tr>
<tr>
<td>7</td>
<td>4591545</td>
</tr>
<tr>
<td>3</td>
<td>4490900</td>
</tr>
<tr>
<td>2</td>
<td>4418143</td>
</tr>
<tr>
<td>6</td>
<td>4005357</td>
</tr>
</tbody>
</table>

7 rows in set. Elapsed: 0.020 sec. Processed 31.65 million rows, 94.96 MB (1.56 billion rows/s., 4.68 GB/s.)
Q2. The number of flights per day from the year 2018 to 2023 MySQL

```sql
mysql> SELECT
    ->   DayOfWeek,
    ->   count(*) AS c
    -> FROM ontime_innodb_compressed
    -> WHERE (Year >= 2018) AND (Year <= 2023)
    -> GROUP BY DayOfWeek
    -> ORDER BY c DESC;

+----------+-----+
| DayOfWeek| c   |
+----------+-----+
| 1        | 4737017 |
| 5        | 4718538 |
| 4        | 4692468 |
| 7        | 4591545 |
| 3        | 4490900 |
| 2        | 4418143 |
| 6        | 4005357 |
+----------+-----+
7 rows in set (3 min 6.55 sec)
```
Q3. The number of delays by airport for 2018-2023

MySQL

```sql
mysql> SELECT Carrier, count(*) FROM ontime_innodb_compressed WHERE (DepDelay > 10) AND (Year = 2022) GROUP BY Carrier ORDER BY count(*) DESC;
```

<table>
<thead>
<tr>
<th>Carrier</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WN</td>
<td>393684</td>
</tr>
<tr>
<td>AA</td>
<td>187036</td>
</tr>
<tr>
<td>DL</td>
<td>153586</td>
</tr>
<tr>
<td>UA</td>
<td>125061</td>
</tr>
<tr>
<td>OO</td>
<td>115204</td>
</tr>
<tr>
<td>B6</td>
<td>84802</td>
</tr>
<tr>
<td>9E</td>
<td>66624</td>
</tr>
<tr>
<td>NK</td>
<td>54020</td>
</tr>
<tr>
<td>YX</td>
<td>52582</td>
</tr>
<tr>
<td>F9</td>
<td>47938</td>
</tr>
<tr>
<td>AS</td>
<td>40037</td>
</tr>
<tr>
<td>MQ</td>
<td>38762</td>
</tr>
<tr>
<td>OH</td>
<td>38700</td>
</tr>
<tr>
<td>G4</td>
<td>35111</td>
</tr>
<tr>
<td>YV</td>
<td>22360</td>
</tr>
<tr>
<td>HA</td>
<td>17191</td>
</tr>
<tr>
<td>QX</td>
<td>13646</td>
</tr>
</tbody>
</table>

17 rows in set (3 min 1.18 sec)
Sink Connector (ChistaDATA)

To Do

- CDC Feature
- Schema tracking at source
- Archival (removal of source data)
- Smart proxy implementation
Conclusion

- Archive or remove unwanted data from your OLTP systems.
  - pt-archiver - Part of Percona toolkit
  - MySQL - The ARCHIVE Storage Engine (unindexed data)
- Move to appropriate data to appropriate data store
Want to try yourself?

Currently implementations on MySQL and PostgreSQL:

https://github.com/askdba/data-archival-lab

https://github.com/Percona-Lab/ontime-airline-performance
References

- https://www.confluent.io/blog/kafka-streams-tables-part-1-event-streaming/
- https://developer.confluent.io/what-is-apache-kafka/
- https://kafka.apache.org/
- http://devdoc.net/database/ClickhouseDocs_19.4.1.3-docs/getting_started/example_datasets/ontime/
- https://www.transtats.bts.gov/Homepage.asp
THANK YOU

Q&A