How OLTP to OLAP Archival Demystified

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

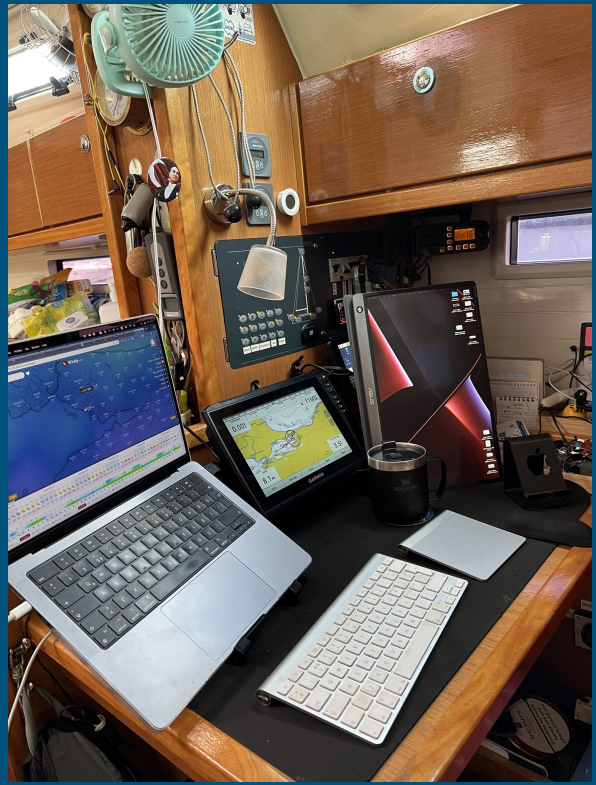
Let’s get connected with Alkin first

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

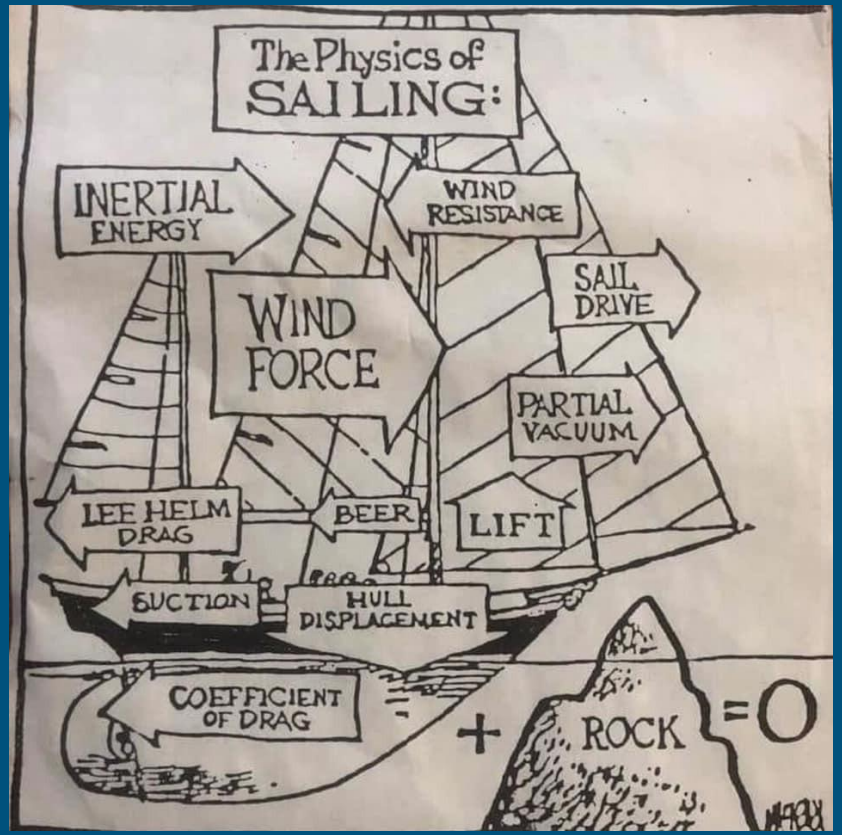
- Previously PlanetScale, Percona and Pythian as Technical Manager, SRE, DBA
- Previously Enterprise DBA , Informix, Oracle, DB2 , SQL Server



Catching winds
@svrubato

Born to Sail, Forced to Work!

@ask_dba



Born to Sail, Forced to Work!

CENTRAL TURKEY Kahramanmaraş, Turkey	04:28 (31 min.ago)	6.7
TURKEY Gaziantep, Turkey	04:28 (31 min.ago)	6.1
TURKEY Kahramanmaraş, Turkey	04:28 (31 min.ago)	7.9
CENTRAL TURKEY Gaziantep, Turkey	04:17 (42 min.ago)	7.8
SOFALACA-SEHITKAMIL (GAZIANTEP) Gaziantep, Turkey	04:17 (42 min.ago)	7.4

TURKEY EARTHQUAKE

Help Turkey!

Turkey has been hit by devastating earthquakes reaching **magnitudes of 7.7**. **More than 1.500 casualties** have been reported but the number is expected to increase much more.

How can you help?

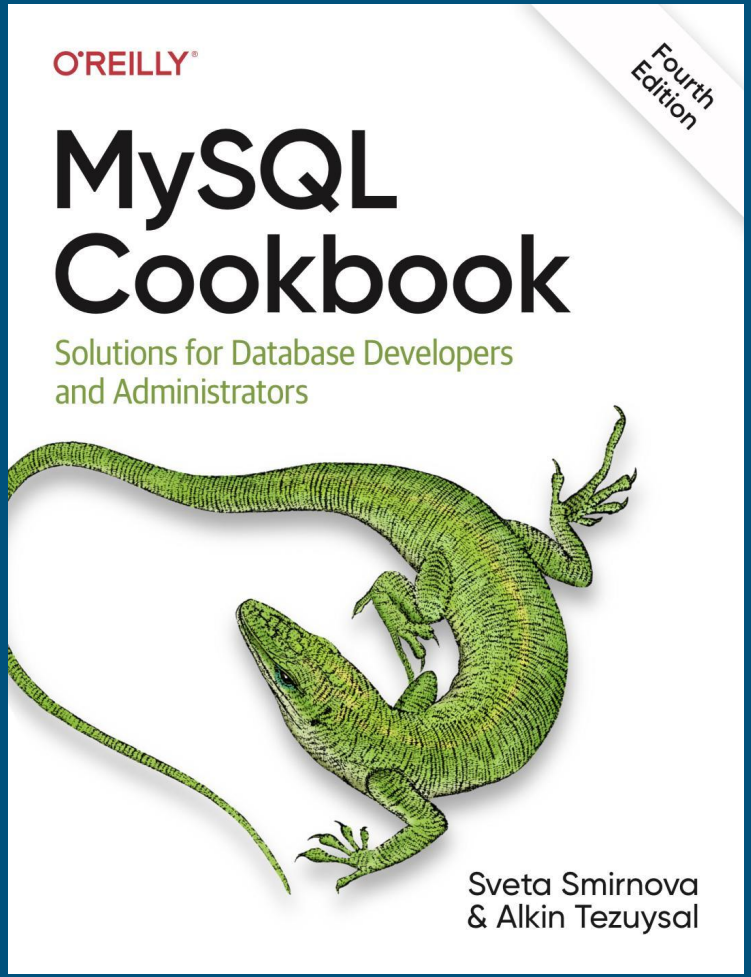
AFAD - Turkey's official Disaster and Emergency Management Presidency: afad.gov.tr/depremkampanyasi2 (donate in TRY, EUR and USD)

AKUT - Voluntary search, assist and rescue organization: akut.org.tr/en/donation (donate in TRY, EUR and USD)

AHBAP - One of the most active voluntary networks, active in the affected regions: bagis.ahbap.org (donate in TRY with credit card)

You can also raise awareness. Make the world aware of the crisis and **help us get help!**

@ask_dba



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

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

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

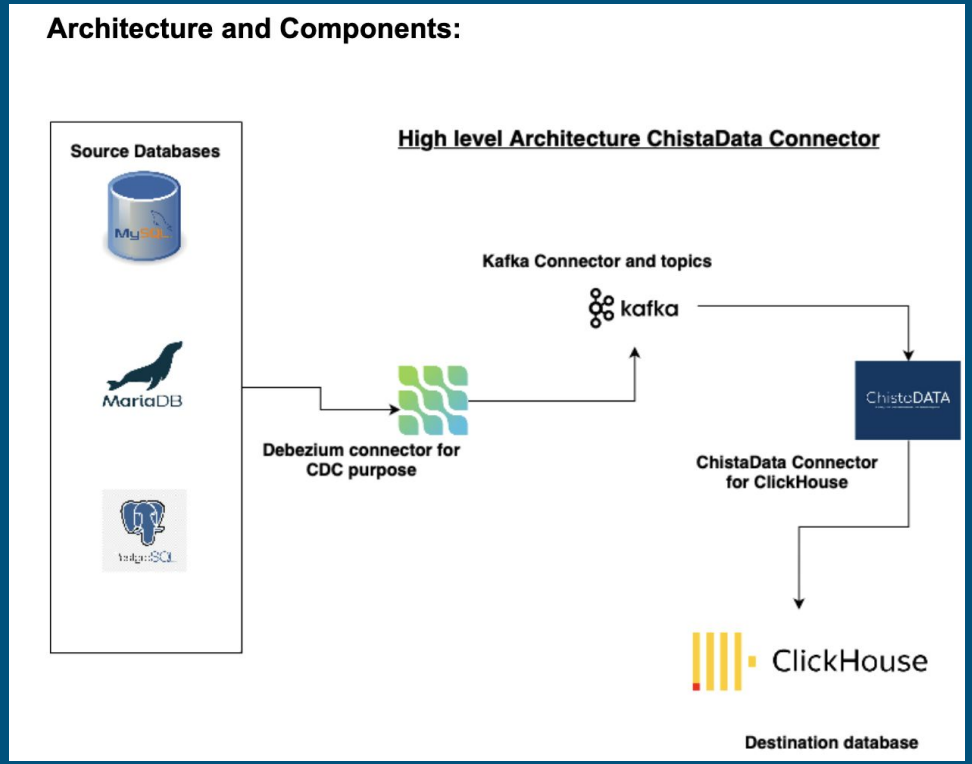
- Compression

Rich and many options for Time Series data

- Distribution

Sharding and Partitioning

OLTP Table Archiving Workflow



Debezium Connector

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)


```
{
   "before":null,
   "after":{
      "a_column":1,
      "b_column":2,
      "c_column":false,
      "d_column":19424,
      "e_column":3.0,
      "f_column":101,
      "g_column":5.0,
      "h_column":6.0,
      "i_column":7,
      "j_column":8,
      "k_column":9,
      "l_column":"10",
      "m_column":38156104758,
      "n_column":1678271756104758,
      "o_column":"a6d455ba-c97a-4a84-889f-5d28d6815c74"
   },
   "source":{
      "version":"2.1.1.Final",
      "connector":"postgresql",
      "name":"some",
      "ts_ms":1678288582098,
      "snapshot":"last",
      "db":"price",
      "sequence":"[null,\"7644674784\"]",
      "schema":"public",
      "table":"employees",
      "txId":786,
      "lsn":7644674784,
      "xmin":null
   },
   "op":"r",
   "ts_ms":1678288582180,
   "transaction":null
}
```


Insert Operations (op: c)


```
{
   "before":null,
   "after":{
      "a_column":11,
      "b_column":12,
      "c_column":false,
      "d_column":19424,
      "e_column":3.0,
      "f_column":4,
      "g_column":5.0,
      "h_column":77.0,
      "i_column":7,
      "j_column":8,
      "k_column":9,
      "l_column":"yeni",
      "m_column":55150444838,
      "n_column":1678288750444838,
      "o_column":"16988e19-1a15-4bb2-a174-34648269a40c"
   },
   "source":{
      "version":"2.1.1.Final",
      "connector":"postgresql",
      "name":"some",
      "ts_ms":1678288750445,
      "snapshot":"false",
      "db":"price",
      "sequence":"[null,\"7644674880\"]",
      "schema":"public",
      "table":"employees",
      "txId":787,
      "lsn":7644674880,
      "xmin":null
   },
   "op":"c",
   "ts_ms":1678288750716,
   "transaction":null
}
```

Debezium Operations

Update Operations (op: u)


```
{
   "before":{
      "a_column":11,
      "b_column":12,
      "c_column":false,
      "d_column":19424,
      "e_column":3.0,
      "f_column":4,
      "g_column":5.0,
      "h_column":77.0,
      "i_column":7,
      "j_column":8,
      "k_column":9,
      "l_column":"yeni",
      "m_column":55150444838,
      "n_column":1678288750444838,
      "o_column":"16988e19-1a15-4bb2-a174-34648269a40c"
   },
   "after":{
      "a_column":11,
      "b_column":12,
      "c_column":false,
      "d_column":19424,
      "e_column":3.0,
      "f_column":4,
      "g_column":5.0,
      "h_column":77.0,
      "i_column":7,
      "j_column":8,
      "k_column":9,
      "l_column":"yeni sürüm",
      "m_column":55150444838,
      "n_column":1678288750444838,
      "o_column":"16988e19-1a15-4bb2-a174-34648269a40c"
   },
   "source":{
      "version":"2.1.1.Final",
      "connector":"postgresql",
      "name":"some",
      "ts_ms":1678288810731,
      "snapshot":"false",
      "db":"price",
      "sequence":"[\"7644676256\",\"7644676544\"]",
      "schema":"public",
      "table":"employees",
      "txId":789,
      "lsn":7644676544,
      "xmin":null
   },
   "op":"u",
   "ts_ms":1678288810886,
   "transaction":null
}
```


Delete Operations (op: d)


```
{
   "before":{
      "a_column":1,
      "b_column":2,
      "c_column":false,
      "d_column":19424,
      "e_column":3.0,
      "f_column":101,
      "g_column":5.0,
      "h_column":6.0,
      "i_column":7,
      "j_column":8,
      "k_column":9,
      "l_column":"10",
      "m_column":38156104758,
      "n_column":1678271756104758,
      "o_column":"a6d455ba-c97a-4a84-889f-5d28d6815c74"
   },
   "after":null,
   "source":{
      "version":"2.1.1.Final",
      "connector":"postgresql",
      "name":"some",
      "ts_ms":1678288885515,
      "snapshot":"false",
      "db":"price",
      "sequence":"[\"7644678392\",\"7644678448\"]",
      "schema":"public",
      "table":"employees",
      "txId":791,
      "lsn":7644678448,
      "xmin":null
   },
   "op":"d",
   "ts_ms":1678288885716,
   "transaction":null
}
```

Converting Data Types from MySQL to ClickHouse

Data types will convert as is to the ClickHouse.

MySQL	ClickHouse
int	Int32
varchar	String
bit	UInt64
bigInt	Int64
bool	Int8
boolean	Int8
float	Float32
double	Float64
date	Date
dateTime	DateTime
timestamp	DateTime
binary	FixedString(10)
tinyInt	Int8
smallInt	Int16
mediumInt	Int32
integer	Int32
doublePrecision	Float64

Other data types will convert as `String` in ClickHouse.

MySQL	ClickHouse
time	String
year	String
char	String
decimal	String
varBinary	String
tinyBlob	String
tinyText	String
text	String
blob	String
mediumText	String
mediumBlob	String
longText	String
longblob	String
enum	String
set	String

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

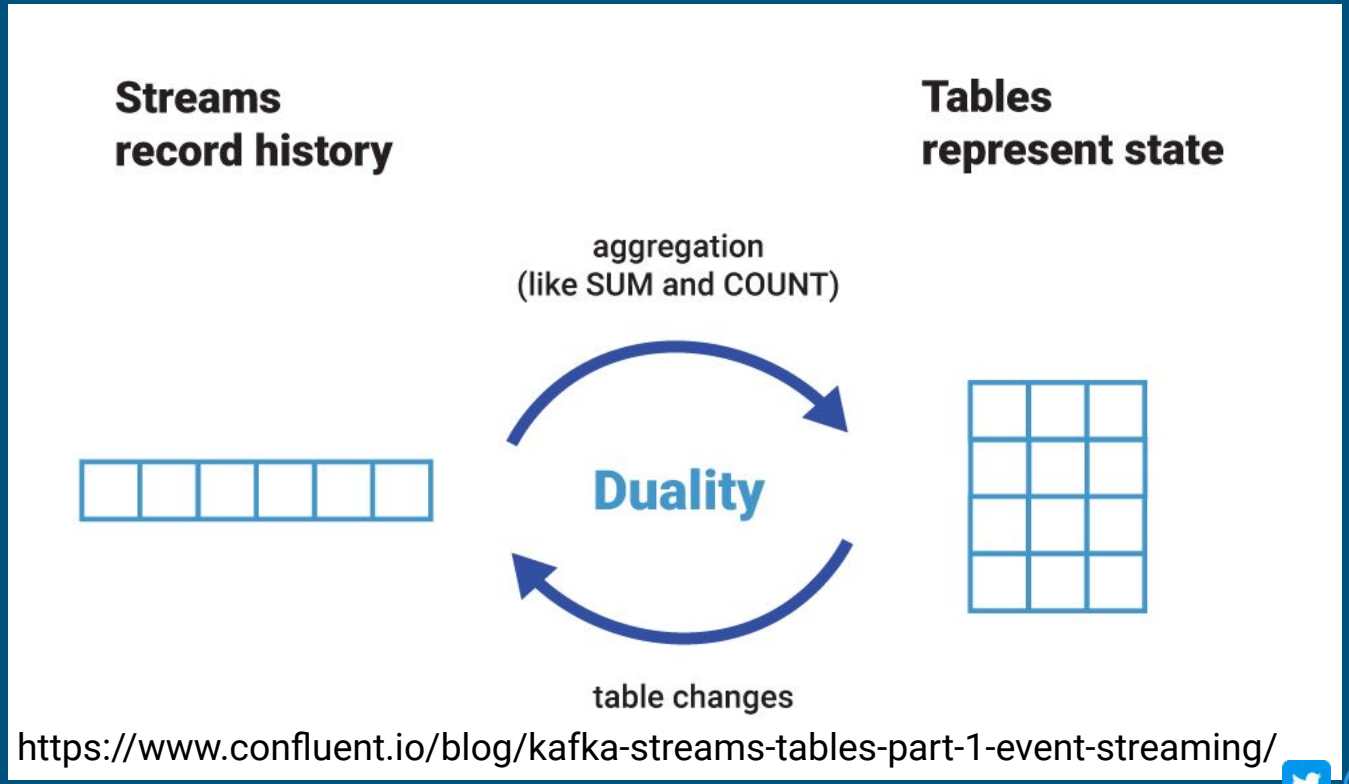
Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

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

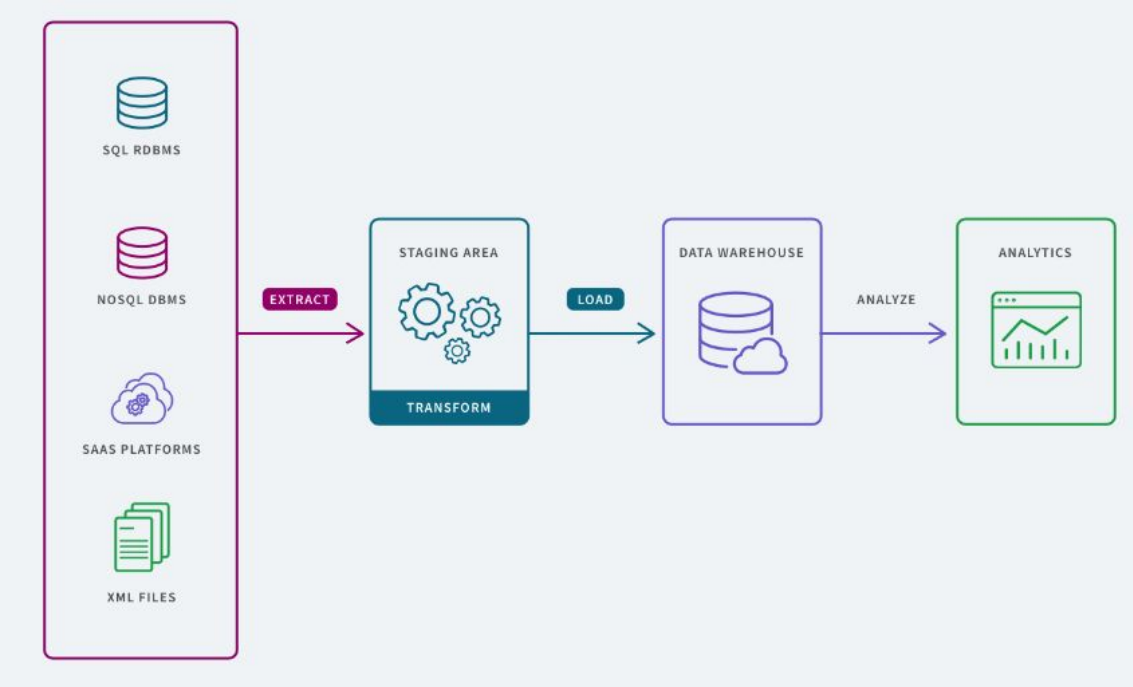
- Data Integration
- Metrics and Monitoring
- Log Aggregation
- Stream Processing
- Message Broker

Kafka Streams and Table



https://www.confluent.io/blog/kafka-streams-tables-part-1-event-streaming/

Streaming / Archiving / CDC



Kafka topic details c, u, d, r

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

Converting Data Types from PostgreSQL to ClickHouse

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

PostgreSQL	ClickHouse
bigint	int64
bigserial	uint64
boolean	bool
date	date
double precision	float64
integer	int32
numeric(2,0)	decimal(2,0)
real	float32
smallint	int16
smallserial	int16
serial	int32
text	string
timestamp	DateTime
uuid	uuid

Other data types will convert as `String` in ClickHouse

PostgreSQL	ClickHouse
bit(4)	string
varbit(4)	string
box	string
bytea	string
char(5)	string
bit varying(5)	string
cidr	string
circle	string
inet	string
interval	string
json	string
jsonb	string
line	string
lseg	string
macaddr	string
macaddr8	string
money	string
path	string
pg_lsn	string
pg_snapshot	string
point	string
polygon	string
time	string
tsquery	string
tsvector	string
txid_snapshot	string
xml	string

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

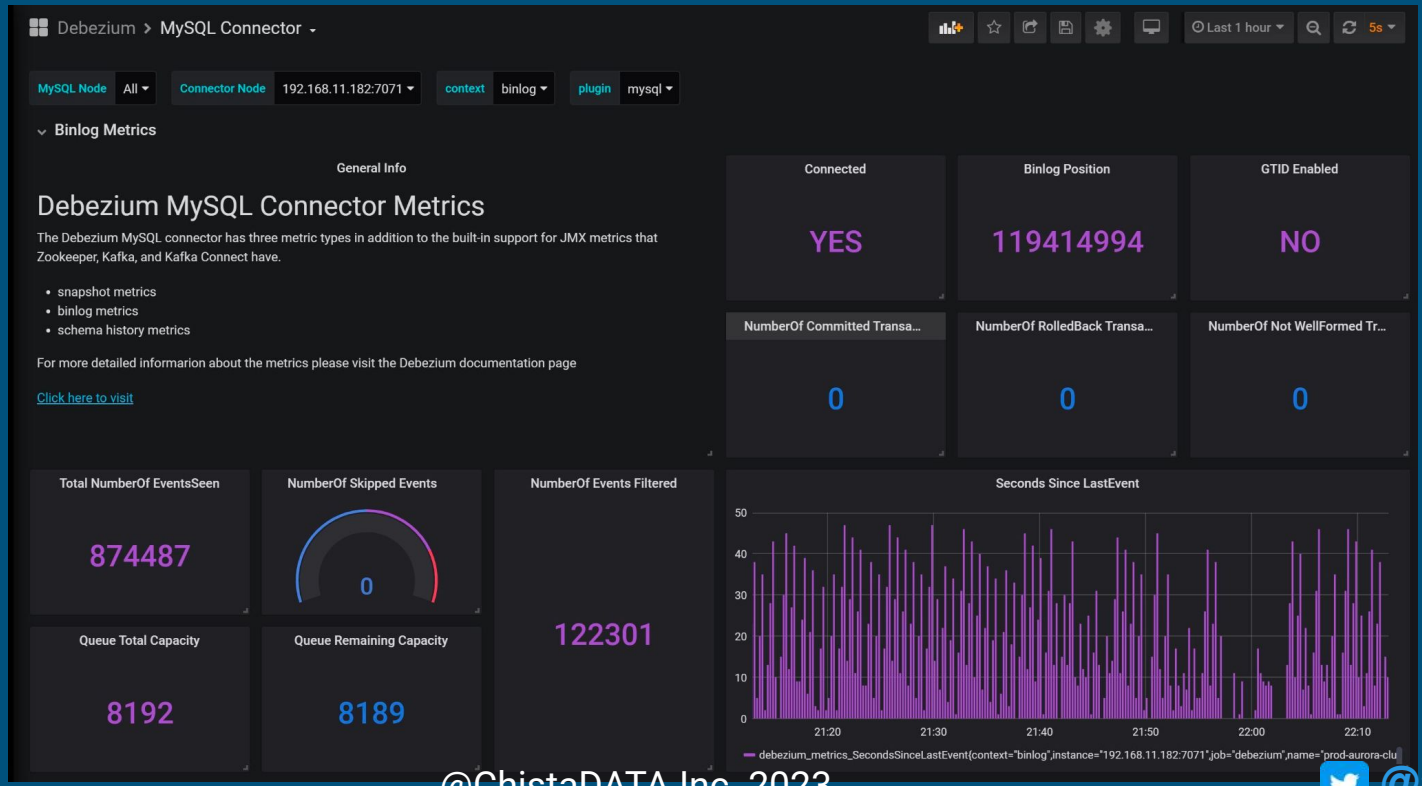
~2.5 Gb at filesystem level

Monitoring

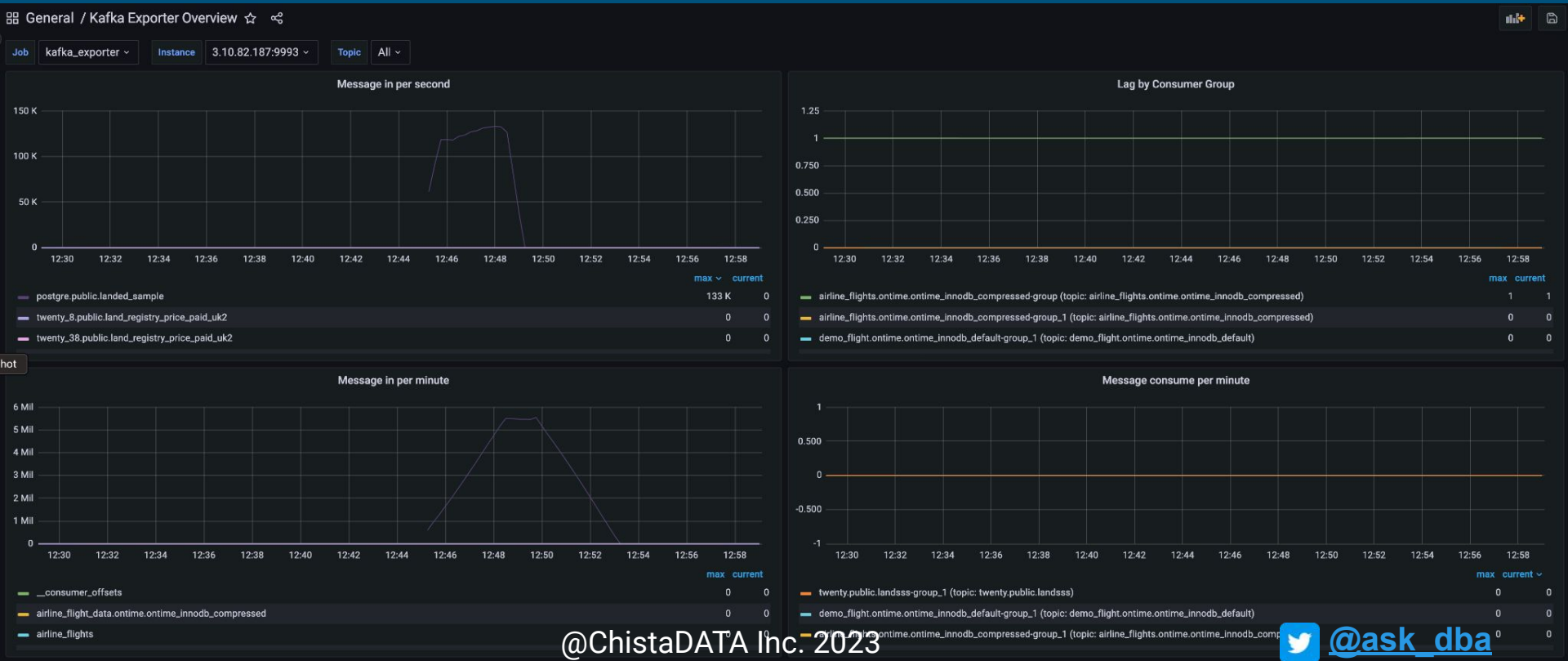
- Debezium
- Kafka
- MySQL / PostgreSQL (Source)
- ClickHouse (Destination)
- Hosts

Debezium monitoring

Debezium MySQL metrics



Kafka Topic Overview



Q1. Avg flight duration ClickHouse

```
ip-172-31-13-227.eu-west-2.compute.internal :) select avg(c1) from (select Year, Month, count(*) as c1
from ontime_innodb_compressed3 group by Year, Month);

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

```
mysql> SELECT avg(c1) FROM (     SELECT         Year,         Month,         count(*) AS c1     FROM on
time_innodb_compressed ch_table    GROUP BY         Year,         Month ) as ch;
+-------------+
| avg(c1)     |
+-------------+
| 527567.1167 |
+-------------+
1 row in set (3 min 8.16 sec)

mysql>
```

Q2. The number of flights per day from the year 2018 to 2023 ClickHouse

```
ip-172-31-13-227.eu-west-2.compute.internal :) SELECT DayOfWeek, count(*) AS c FROM ontime_innodb_compr
essed3 WHERE Year >= 2018 AND Year <= 2023 GROUP BY DayOfWeek ORDER BY c DESC;

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

Query id: 83597ccd-49a6-4758-9208-b903e7dc4715

┌─DayOfWeek─┬───────c─┐
│         1 │ 4737017 │
│         5 │ 4718538 │
│         4 │ 4692468 │
│         7 │ 4591545 │
│         3 │ 4490900 │
│         2 │ 4418143 │
│         6 │ 4005357 │
└───────────┴─────────┘

7 rows in set. Elapsed: 0.020 sec. Processed 31.65 million rows, 94.96 MB (1.56 billion rows/s., 4.68 G
B/s.)

ip-172-31-13-227.eu-west-2.compute.internal :)
```

Q2. The number of flights per day from the year 2018 to 2023 MySQL

```
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)

mysql>
```

Q3. The number of delays by airport for 2018-2023 MySQL

```
mysql> SELECT     Carrier,     count(*) FROM ontime_innodb_compressed WHERE (DepDelay > 10) AND (Year =
 2022) GROUP BY Carrier ORDER BY count(*) DESC;
+---------+----------+
| Carrier | count(*) |
+---------+----------+
| WN      |   393684 |
| AA      |   187036 |
| DL      |   153586 |
| UA      |   125061 |
| OO      |   115204 |
| B6      |    84802 |
| 9E      |    66624 |
| NK      |    54020 |
| YX      |    52582 |
| F9      |    47938 |
| AS      |    40037 |
| MQ      |    38762 |
| OH      |    38700 |
| G4      |    35111 |
| YV      |    22360 |
| HA      |    17191 |
| QX      |    13646 |
+---------+----------+
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://debezium.io/documentation/reference/stable/connectors/index.html
- https://developer.confluent.io/what-is-apache-kafka/
- https://kafka.apache.org/
- https://www.percona.com/blog/apache-spark-with-air-ontime-performance-data/
- 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