

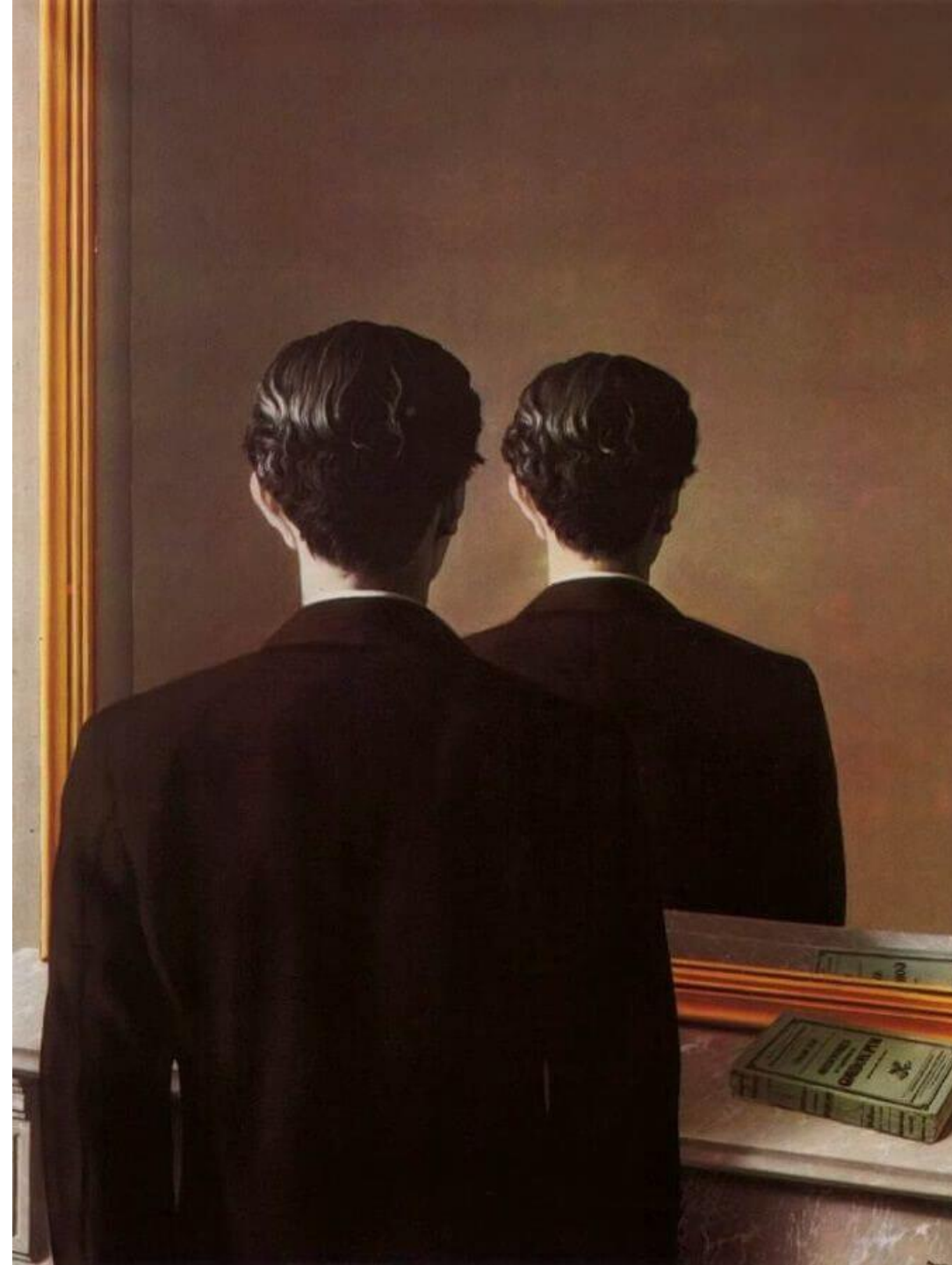


Replicating Transactional Databases to ClickHouse : Transaction Log Analysis and Time Travel

FOSDEM 2026 January 31

Arnaud Adant

La reproduction interdite, 1937
Not to be reproduced
Magritte



Agenda

- Introduction
- Requirements
- Design
- Implementation
- Questions (5 min)

Who I am

Arnaud Adant

- Database Team Lead
- Passionate about tech
- Jump Trading
 - Leading data and research-driven trading business
 - Based in Chicago, 10+ global offices
 - Trade all asset classes, all time horizons

Introduction

- Transactional databases



- Heterogeneous Replication

- One system to another, 2 paradigms

- The target : ClickHouse



What is ClickHouse ?

- An Open Source columnar database
- MergeTree data structure
- Created in 2009
- 45.5k stars (Jan 2026)
- Realtime Analytics
- VLDB 2024 - ClickHouse: Lightning Fast Analytics for Everyone
- Over 2000 contributors



How fast is it ?

Some SQL query ... using 20 nodes on Apache Iceberg

```
:) select d , count(*) from db.table group by d order by d;
```

...

```
12 rows in set. Elapsed: 9.648 sec. Processed 376.03 billion rows,  
109.50 TB (38.97 billion rows/s., 11.35 TB/s.)
```

What is replication ?

- Data synchronization between a primary and replica(s)
- Homogeneous (easier)
- Heterogeneous (challenging)
- Logical replication
- Log based
- Change Data Capture

Why replication ?

- Scale out reads
- Migration : system A to B
- Continuous synchronization
- Fault tolerance
- Disaster recovery
- Very useful building block

Requirements

- No Data Loss Replication
- Low Latency
- Exactly Once Delivery
- Operational Simplicity
- Fault Tolerance / HA
- Fully Open Source

Design

- **Keep It Simple Stupid**

- o A docker container, simple yaml config



- **Standing on the shoulders of giants :**

- o MySQL binary logs, ANTLR, PG Logical replication



- **Do not re-invent the wheel :**

- o Debezium for Change Data Capture



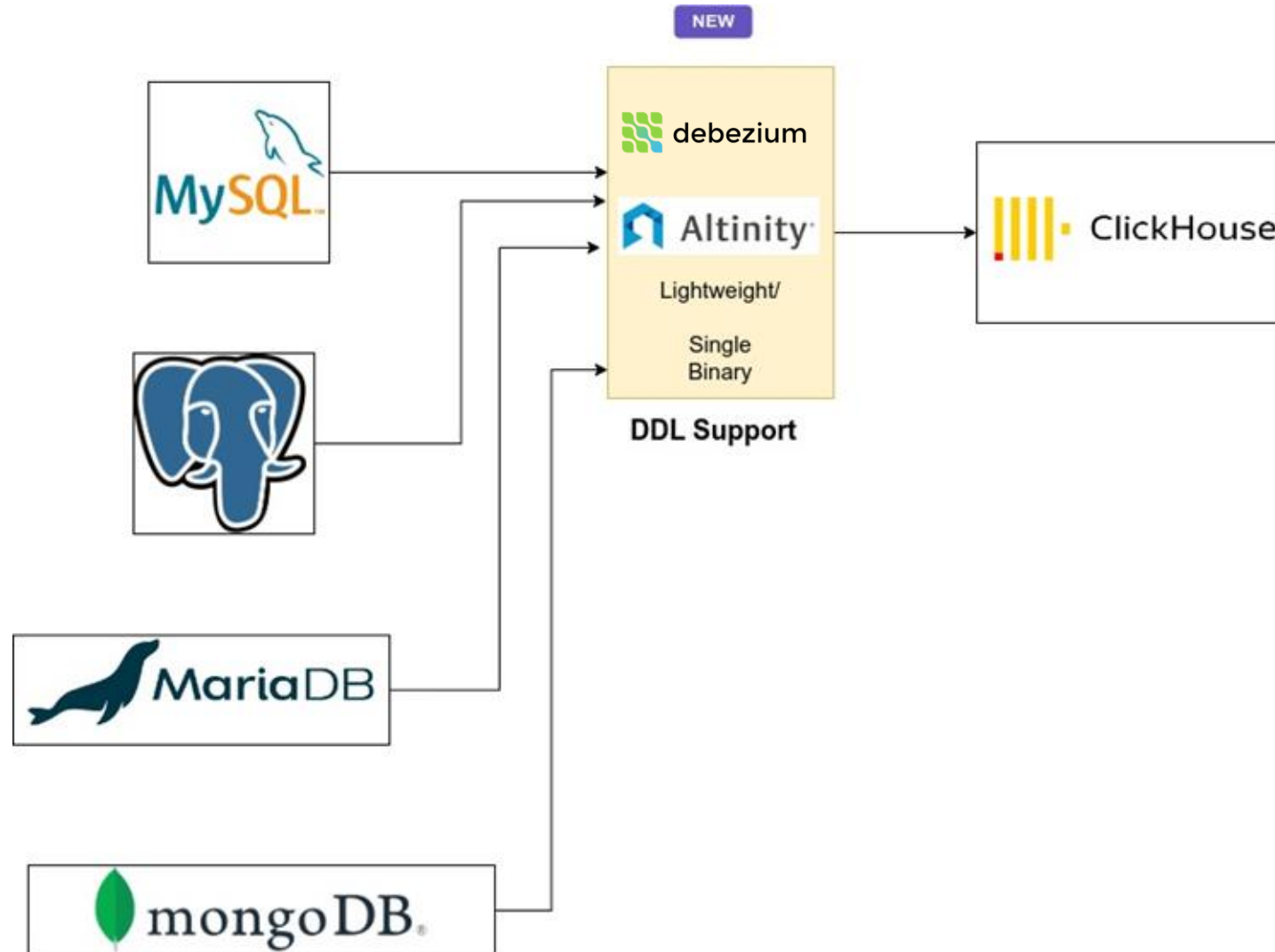
Design in ClickHouse (1/2)

- One to One Table mapping
- Primary key
- Data Type Mapping
- ReplacingMergeTree engine, aka RMT
 - Turn all operations into inserts !
 - Merge on reads (final=1)
- “the same queries should return the same results”
- Full DDL Support
- Timezones

Design in ClickHouse (2/2)

- Replication state stored in ClickHouse
- MySQL replication like behavior (stop / start / status)
- Retry on failure, both DML and DDL
- Replication filters
- Checksums
- Efficient dumpers and loaders

Architecture



Implementation : Sink Connector Lightweight

- <https://github.com/Altinity/clickhouse-sink-connector>
- One container or one systemd service
- Docker compose
- Java and Debezium based
- Multi-threaded Applier
- Eventually consistent
- Low Latency



Table Migration



```
CREATE TABLE `orders` (  
  `orderNumber` int NOT NULL,  
  `orderDate` date NOT NULL,  
  `requiredDate` date NOT NULL,  
  `shippedDate` date DEFAULT NULL,  
  `status` varchar(15) NOT NULL,  
  `comments` text,  
  `customerNumber` int NOT NULL,
```

```
  PRIMARY KEY (`orderNumber`),  
  KEY `customerNumber` (`customerNumber` )  
  ENGINE=InnoDB DEFAULT CHARSET=latin1
```

```
  PARTITION BY RANGE COLUMNS(orderDate) (  
    (PARTITION p20201231 VALUES LESS THAN ('2021-01-01'),  
    PARTITION p20211230 VALUES LESS THAN ('2021-12-31')  
  )
```

```
CREATE TABLE test.orders (  
  `orderNumber` Int32,  
  `orderDate` Date32,  
  `requiredDate` Date32,  
  `shippedDate` Nullable(Date32),  
  `status` String,  
  `comments` Nullable(String),  
  `customerNumber` Int32,
```

```
  `_version` UInt64,  
  `is_deleted` UInt8 )
```

```
  ENGINE = ReplacingMergeTree(_version, is_deleted)
```

```
  ORDER BY orderNumber
```

```
  PARTITION BY orderDate
```

```
  SETTINGS index_granularity = 8192
```

What about the history ?

- We added an history mode
- Separate schema *_history*
- **Transaction Log**
 - Big fat table
 - Essential for auditing
 - Schema changes tracking
- **History tables**
 - One to one
 - Same name in a separate schema
 - Time travel
 - Includes the current version
 - SCD2

Transaction Log (MySQL)

```
CREATE TABLE binlog_history.history
(
  `gtid` String,
  `database` LowCardinality(String),
  `table` LowCardinality(String),
  `ddl` String,
  `before` String,
  `after` String,
  `_raw` String,
  `_time` DateTime('America/Chicago'),
  `is_deleted` UInt8,
  `_operation` LowCardinality(String),
  `_version` UInt64,
  `host` LowCardinality(String),
  `logfile` LowCardinality(String),
  `position` UInt64,
  `primary_host` LowCardinality(String),
  `server_id` UInt32,
  `row` UInt32,
  `sequence` UInt64
)
ENGINE = ReplacingMergeTree(_version, is_deleted)
PARTITION BY toDate(_time)
ORDER BY (server_id, logfile, position, sequence, _time)
TTL toDate(_time) + toIntervalDay(30)
SETTINGS index_granularity = 8192
```

gtid
database
table

before / after Image
_raw event in JSON
_operation : type of event

timestamp (second resolution)
host (replica)
primary host (primary)
server_id
binary log file
binary log position
row : position within the event
sequence number

Transaction Log (MySQL)



```
CREATE TABLE binlog_history.history
(
  `gtid` String,
  `database` LowCardinality(String),
  `table` LowCardinality(String),
  `ddl` String,
  `before` String,
  `after` String,
  `_raw` String,
  `_time` DateTime('America/Chicago'),
  `is_deleted` UInt8,
  `_operation` LowCardinality(String),
  `_version` UInt64,
  `host` LowCardinality(String),
  `logfile` LowCardinality(String),
  `position` UInt64,
  `primary_host` LowCardinality(String),
  `server_id` UInt32,
  `row` UInt32,
  `sequence` UInt64
)
ENGINE = ReplacingMergeTree(_version, is_deleted)
PARTITION BY toDate(_time)
ORDER BY (server_id, logfile, position, sequence, _time)

TTL toDate(_time) + toIntervalDay(30), toDate(_time) + toIntervalDay(1) RECOMPRESS CODEC(ZSTD(8))
SETTINGS index_granularity = 8192
```

Table Engine : RMT with *_version* and *is_deleted*

Partitioning by day

Sort Key (uniqueness)

TTL, 30 days delete, recompression after 1 day

History tables



ClickHouse

```
CREATE TABLE `orders` (  
  `orderNumber` int NOT NULL,  
  `orderDate` date NOT NULL,  
  `requiredDate` date NOT NULL,  
  `shippedDate` date DEFAULT NULL,  
  `status` varchar(15) NOT NULL,  
  `comments` text,  
  `customerNumber` int NOT NULL,
```

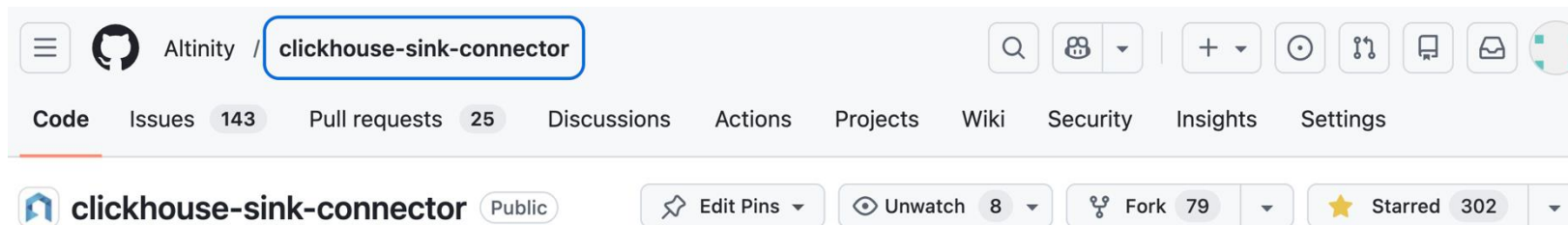
```
PRIMARY KEY (`orderNumber`),  
KEY `customerNumber` (`customerNumber` )  
ENGINE=InnoDB DEFAULT CHARSET=latin1  
  
PARTITION BY RANGE COLUMNS(orderDate)  
(  
  PARTITION p20201231 VALUES LESS THAN ('2021-01-01'),  
  PARTITION p20211230 VALUES LESS THAN ('2021-12-31')  
)
```

```
CREATE TABLE test_history.orders (  
  `orderNumber` Int32,  
  `orderDate` Date32,  
  `requiredDate` Date32,  
  `shippedDate` Nullable(Date32),  
  `status` String,  
  `comments` Nullable(String),  
  `customerNumber` Int32,  
  
  `_version` UInt64,  
  `is_deleted` UInt8,  
  
  `_valid_from` DateTime DEFAULT '2100-01-01 00:00:00',  
  `_valid_to` DateTime. DEFAULT '2100-01-01 00:00:00',  
  `_operation` LowCardinality(String) )  
  
ENGINE = ReplacingMergeTree(_version, is_deleted)  
ORDER BY (orderNumber, _valid_to)  
PARTITION BY toDate(_valid_to)  
  
TTL _valid_to + toIntervalDay(30),  
+ valid_to toIntervalDay(1) RECOMPRESS CODEC(ZSTD(8))
```

RMT

SCD2

Development Status



- One to one table : Generally Available, use 2.8.0
 - development started in 2022
- Binary log table : beta, 2.9.0
- History tables : beta, 2.9.0

Similar project : PeerDB developed by ClickHouse



Q&A





CHICAGO • LONDON • SINGAPORE • NEW YORK • SHANGHAI • BRISTOL
AMSTERDAM • HONG KONG • PARIS • SYDNEY • GIFT CITY • AUSTIN • MUMBAI