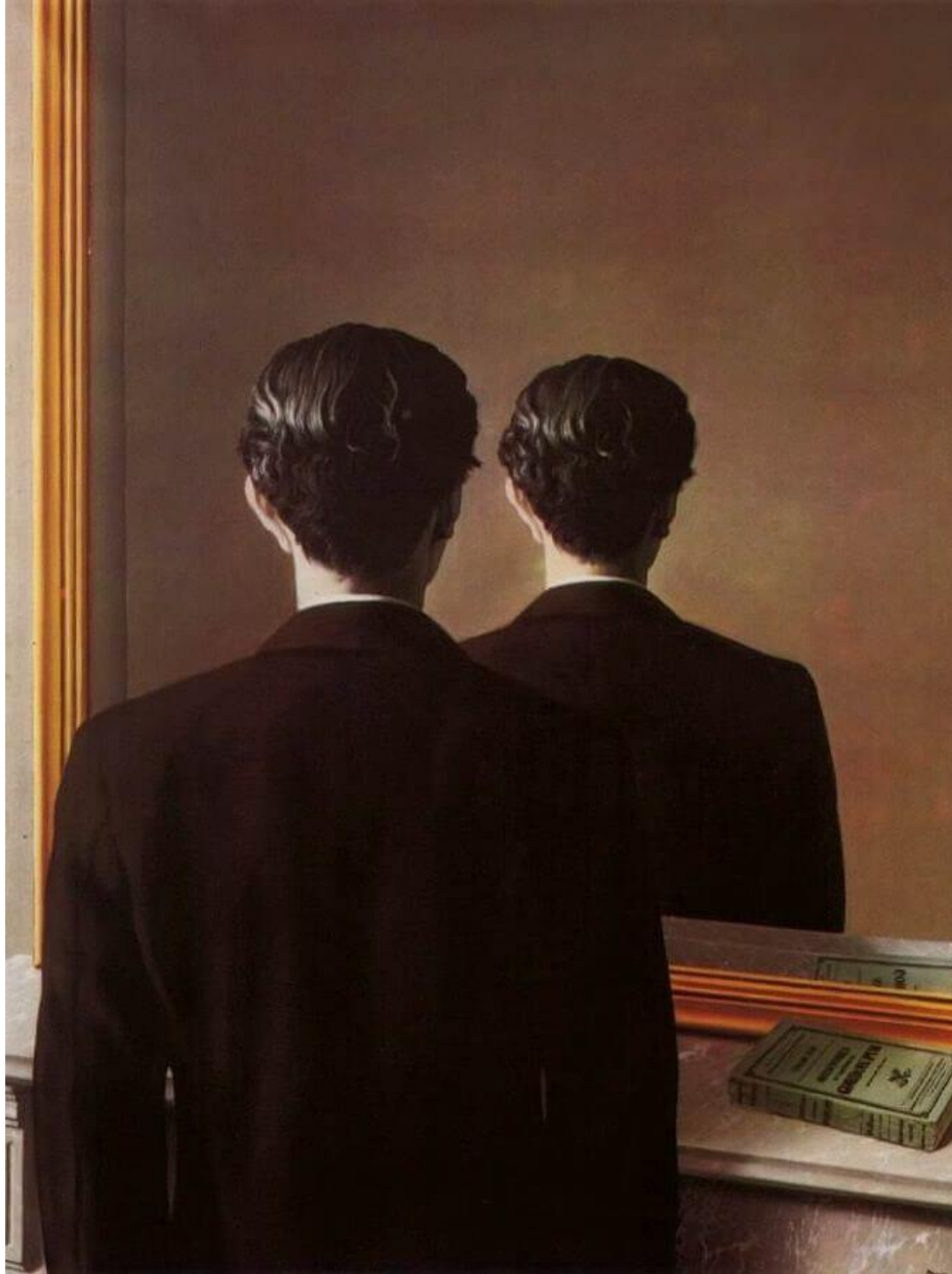
Jump TRADING

MySQL Binary Log Analytics

preFOSDEM 2026 January 30

Arnaud Adant



La reproduction interdite, 1937
Magritte

Agenda

- Introduction
- MySQL Binary Log
- Available tools
- How to Analyze the Binary log ?
- Dashboards
- AI Analysis

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

What is the MySQL Binary Log ?

- A bunch of **binary** files
- Store all write events and schema changes
- Used for
  - Replication
  - Point-in-Time recovery (PIT)
  - Change Data Capture (CDC)
  - Audit (all writes + DDL)
  - Flashback

What is the MySQL Binary Log ?

- Different from the InnoDB **Redo Log**
  - Used for crash recovery only
- Storage Engine Independent
- One of the crown jewels of MySQL
- Immutable
- Append only
- Limited Retention

Where is the MySQL Binary Log ?

- log-bin=/var/lib/mysql/logs

```
$ pwd
/var/lib/mysql/logs

$ ls -al | head
total 1571071048
drwxr-xr-x  3 mysql mysql       69632 Jan 25 07:00 .
drwxr-x--x 13 mysql mysql        4096 Jan 25 05:28 ..
-rw-r-----  1 mysql mysql  1124172569 Jan 18 10:38 binary.165361
-rw-r-----  1 mysql mysql  1371848684 Jan 18 13:34 binary.165362
-rw-r-----  1 mysql mysql  1074536803 Jan 18 15:03 binary.165363
-rw-r-----  1 mysql mysql   448894139 Jan 18 16:36 binary.165364
-rw-r-----  1 mysql mysql         201 Jan 18 16:36 binary.165365
-rw-r-----  1 mysql mysql  1085670995 Jan 18 16:44 binary.165366
-rw-r-----  1 mysql mysql  1076060570 Jan 18 18:09 binary.165367
```

MySQL Binary Log Format

- SET GLOBAL binlog_format = 'ROW'; -- Options: ROW, STATEMENT, MIXED
-
  - `binlog_row_image`

Command-Line Format	`--binlog-row-image=image_type`
System Variable	`binlog_row_image`
Scope	Global, Session
Dynamic	Yes
`SET_VAR` **Hint Applies**	No
Type	Enumeration
Default Value	`full`
Valid Values	`full` (Log all columns) `minimal` (Log only changed columns, and columns needed to identify rows) `noblob` (Log all columns, except for unneeded BLOB and TEXT columns)

MySQL Binary Log Format

- Can be compressed (recommended !), ZSTD
  - `binlog_transaction_compression`

Command-Line Format	`--binlog-transaction-compression[={OFF|ON}]`
System Variable	`binlog_transaction_compression`
Scope	Global, Session
Dynamic	Yes
`SET_VAR` **Hint Applies**	No
Type	Boolean
Default Value	`OFF`

- binlog_transaction_compression_level_zstd = 1
- Can be encrypted

What is inside the MySQL Binary Log ?

```
þbin<9c>Àli^OÙ^@^@^@z^@^@^@~^@^@^@^@^@^D^@8.0.33-
25^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^S^@^M^@^H^@^@^@^@^D^@^D^@^@^@b^@^D^Z^H^@^@^
@^H^H^H^B^@^@^@

**^@^R4^@
(^@^AÖH^HN<9c>Àli#Ù^@^@^@^_^@^@^@<9d>^@^@^@<80>^@^@^@^@^@^@^@^@^@^@ÑçÂ®
"<9d>Àli"Ø^@^@^@Z^@^@^@÷^@^@^@<80>^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^@^B^@^@^@^@^@^@^@^@^@^A^@^@^@^@^@^@^@^@[p^SÂ®§H<86>^YL^S
Â®§H^Fü¯^A¡8^A<80><9f>8^A^@¿<9c>uô<9d>Àli^BØ^@^@^@L^@^@^@C^A^@^@^H^@<
80>C^F^@^@^@^@^@
```

Available tools

- mysqlbinlog

```
logs]$ mysqlbinlog --base64-output=decode-rows -vv binary.165361 | head

# The proper term is pseudo_replica_mode, but we use this compatibility alias
# to make the statement usable on server versions 8.0.24 and older.
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=1*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
# at 4
#260118  5:14:36 server id 217  end_log_pos 126 CRC32 0x4e0848d6     Start: binlog v 4, server v
8.0.33-25 created 260118  5:14:36
# at 126
#260118  5:14:36 server id 217  end_log_pos 157 CRC32 0x22aee7d1     Previous-GTIDs
# [empty]
# at 157
#260118  5:14:37 server id 216  end_log_pos 247 CRC32 0xf4759cbf     Anonymous_GTID   last_committed=0
        sequence_number=1rbr_only=yes     original_committed_timestamp=1768734877502489
        immediate_commit_timestamp=1768734877511771         transaction_length=431
/*!50718 SET TRANSACTION ISOLATION LEVEL READ COMMITTED*//*!*/;
# original_commit_timestamp=1768734877502489 (2026-01-18 05:14:37.502489 CST)
# immediate_commit_timestamp=1768734877511771 (2026-01-18 05:14:37.511771 CST)
```

Before image / after image

- mysqlbinlog

Operation	Before Image	After Image
Insert	No	Yes
Update	Yes	Yes
Delete	Yes	No

mysqlbinlog

- Full remote client
  - --host
- Batch only
- Sequential
  - --start-position
  - --end-position
- Hard to search (regexp …)
- Hard to figure out the columns in the events

mysqlbinlog (Before and After image)

```
#260118  5:14:43 server id 216  end_log_pos 995 CRC32 0xa3a3d784        Update_rows: table id 17850 flags:
STMT_END_F
### UPDATE `server_team_replicate`.`heartbeat`
### WHERE
###   @1='2026-01-18T05:14:33.000550' /* VARSTRING(26) meta=26 nullable=0 is_null=0 */
###   @2=216 /* INT meta=0 nullable=0 is_null=0 */
###   @3='binary.165091' /* VARSTRING(255) meta=255 nullable=1 is_null=0 */
###   @4=152833698 /* LONGINT meta=0 nullable=1 is_null=0 */
###   @5=NULL /* VARSTRING(255) meta=255 nullable=1 is_null=1 */
###   @6=NULL /* LONGINT meta=0 nullable=1 is_null=1 */
### SET
###   @1='2026-01-18T05:14:43.000550' /* VARSTRING(26) meta=26 nullable=0 is_null=0 */
###   @2=216 /* INT meta=0 nullable=0 is_null=0 */
###   @3='binary.165091' /* VARSTRING(255) meta=255 nullable=1 is_null=0 */
###   @4=153083795 /* LONGINT meta=0 nullable=1 is_null=0 */
###   @5=NULL /* VARSTRING(255) meta=255 nullable=1 is_null=1 */
###   @6=NULL /* LONGINT meta=0 nullable=1 is_null=1 */
# at 995
#260118  5:14:43 server id 216  end_log_pos 1026 CRC32 0xc46f87c5       Xid = 192004197
COMMIT/*!*/;
```

Other tools

- **Binlog2sql**
  - Used for Flashback
  - https://github.com/danfengcao/binlog2sql
  - Generate SQL statements to undo a (bad) change
- **Debezium** : Binary Log to Kafka events
- **go-mysql**
  - https://github.com/go-mysql-org/go-mysql
- **Python-mysql-replication**
  - https://github.com/julien-duponchelle/python-mysql-replication

Analytics Requirements

- No Information Loss
- Long Retention
  - Weeks, months
- Low Latency, seconds not minutes / hours
- Interactive Response Time << minutes
  - Humans
  - AI Agents
- Reasonable HW costs

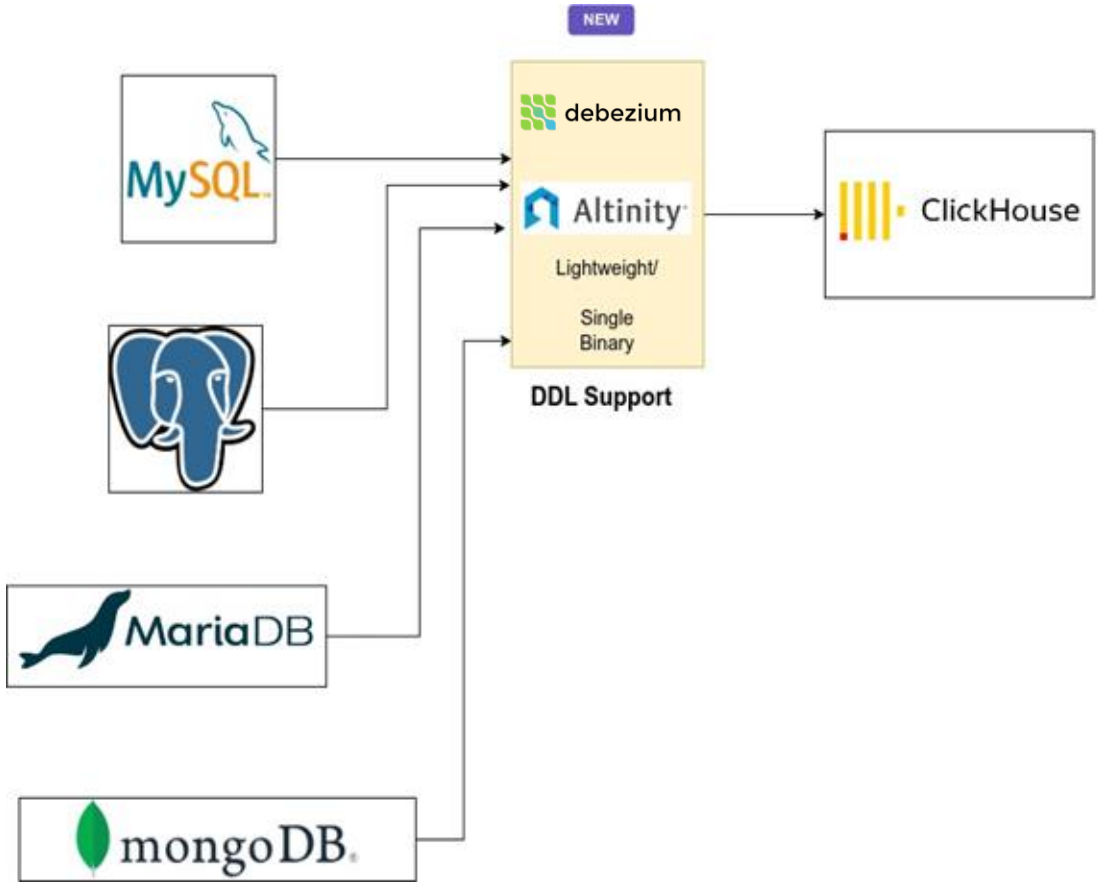
Analytics Solution

- Log all events to one big fat table
- Log all information
  - Timestamp
  - Raw event (decoded)
  - Before / after Image
  - Type of Event
  - …
- Stream it, do not batch it
- This is a **real time analytics / observability** problem

Implementation

- Vanilla MySQL ?!
  - **No**
- Possible options
  - Turn the files into Parquet files
  - Stream into a Real Time Analytics Database
    - Low latency
      - ClickHouse
      - StarRocks / Doris
      - Heatwave
    - …

Architecture



Open Source Implementation

- “standing on the shoulder of giants”
- Docker image or system service
  - **Debezium** embedded
  - **JDBC**
- **ClickHouse**, as Real Time Data Warehouse

Sink Connector Lightweight

- One container
- Docker compose / systemd
- Java and Debezium based
- Multi-threaded Applier
- Eventually consistent
- Low Latency

Sink Connector Lightweight

- Originally designed to replicate like for like tables
- 1 table to 1 table
- For OLTP to OLAP (see my presentation at FOSDEM 2026)

- Extended to support
    - Binary Log History (binlog_history)
    - Time Travel with SCD2

binlog_history Table Schema

```
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

_

Table Schema, ClickHouse specific


```
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, 1 day recompression

Example Event

```
gtid:          -1
database:      server_team_replicate
table:         heartbeat
ddl:
before:        [{"name":"ts","index":0,"schema":{"type":"STRING","optional":false},"value":"2026-01-
21T13:30:13.000540"},{"name":"server_id","index":1,"schema":{"type":"INT64","optional":false},"value":212},{"
name":"file","index":2,"schema":{"type":"STRING","optional":true},"value":"binary.161643"},{"name":"position"
,"index":3,"schema":{"type":"INT64","optional":true},"value":533208690}]
after:         [{"name":"ts","index":0,"schema":{"type":"STRING","optional":false},"value":"2026-01-
21T13:30:23.000540"},{"name":"server_id","index":1,"schema":{"type":"INT64","optional":false},"value":212},{"
name":"file","index":2,"schema":{"type":"STRING","optional":true},"value":"binary.161643"},{"name":"position"
,"index":3,"schema":{"type":"INT64","optional":true},"value":536918241}]
_raw:
{"key":{"server_id":212},"value":{"op":"u","before":{"file":"binary.161643","exec_master_log_pos":null,"relay
_master_log_file":null,"position":533208690,"server_id":212,"ts":"2026-01-
21T13:30:13.000540"},"ts_us":1769301398302440,"after":{"file":"binary.161643","exec_master_log_pos":null,"rel
ay_master_log_file":null,"position":536918241,"server_id":212,"ts":"2026-01-
21T13:30:23.000540"},"source":{"ts_us":1769023823000000,"query":null,"thread":1109746267,"server_id":212,"ver
sion":"3.1.3.Final","sequence":null,"file":"binary.155412","connector":"mysql","pos":136833281,"name":"embedd
edconnector","gtid":null,"row":0,"ts_ns":1769023823000000000,"ts_ms":1769023823000,"snapshot":"false","db":"s
erver_team_replicate","table":"heartbeat"},"ts_ns":1769301398302440722,"transaction":null,"ts_ms":17693013983
02},"topic":"embeddedconnector.server_team_replicate.heartbeat","sourceOffset":{"ts_sec":1769023823,"file":"b
inary.155412","pos":136833281,"row":1,"server_id":212},"sourcePartition":{"server":"embeddedconnector"}}
_time:         2026-01-21 13:30:23
is_deleted:    0
_operation:    UPDATE
```

Binary log Analytics in Action

- Troubleshoot replication
  - Lags
- Audit Trail
- Calculate the information size and rate as we have the _raw per table
- Explain metrics
- Spot the very large events
- Spot the very frequent events
- Spot the very frequent DDL
- Feed the data to an AI and ask questions
- Generate a flashback script for tables between 2 points in time

Absolute Lag

```
SELECT
    _time,
    fromUnixTimestamp64Nano(toUInt64(JSON_VALUE(_raw, '$.value.ts_ns')), 'America/Chicago') AS debezium_time,
    toDateTime(debezium_time) - _time AS debezium_lag,
    formatReadableTimeDelta(debezium_lag) AS readable_lag
FROM binlog_history.history
WHERE (database = 'server_team_replicate') AND (table = 'heartbeat')
ORDER BY _time DESC
LIMIT 1


Row 1:
──────
_time:          2026-01-22 15:49:33
debezium_time: 2026-01-25 20:04:31.965019214
debezium_lag:  274498
readable_lag:  3 days, 4 hours, 14 minutes and 58 seconds
```

Audit Trail

```
SELECT
    _time,
    JSONExtractRaw(_raw, 'key') AS key,
    JSON_VALUE(before, '$[0].value') AS before_value,
    JSON_VALUE(after , '$[0].value') AS after_value,
    _operation
FROM binlog_history.history
WHERE table = 'heartbeat'
ORDER BY
    _time DESC,
    sequence DESC
LIMIT 1

Row 1:
──────
_time:        2026-01-22 08:19:53
key:          {"server_id":212}
before_value: 2026-01-22T08:19:43.000710
after_value:  2026-01-22T08:19:53.000530
_operation:   UPDATE
```

Audit Trail

```
SELECT
    _time,
    JSONExtractRaw(_raw, 'key') AS key,
    JSON_VALUE(before, '$[0].value') AS before_value,
    JSON_VALUE(after , '$[0].value') AS after_value,
    _operation
FROM binlog_history.history
WHERE table = 'heartbeat'
ORDER BY
    _time DESC,
    sequence DESC
LIMIT 1

Row 1:
──────
_time:        2026-01-22 08:19:53
key:          {"server_id":212}
before_value: 2026-01-22T08:19:43.000710
after_value:  2026-01-22T08:19:53.000530
_operation:   UPDATE
```

Jump TRADING

Q&A

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