Sync Clickhouse with MySQL/MongoDB

About

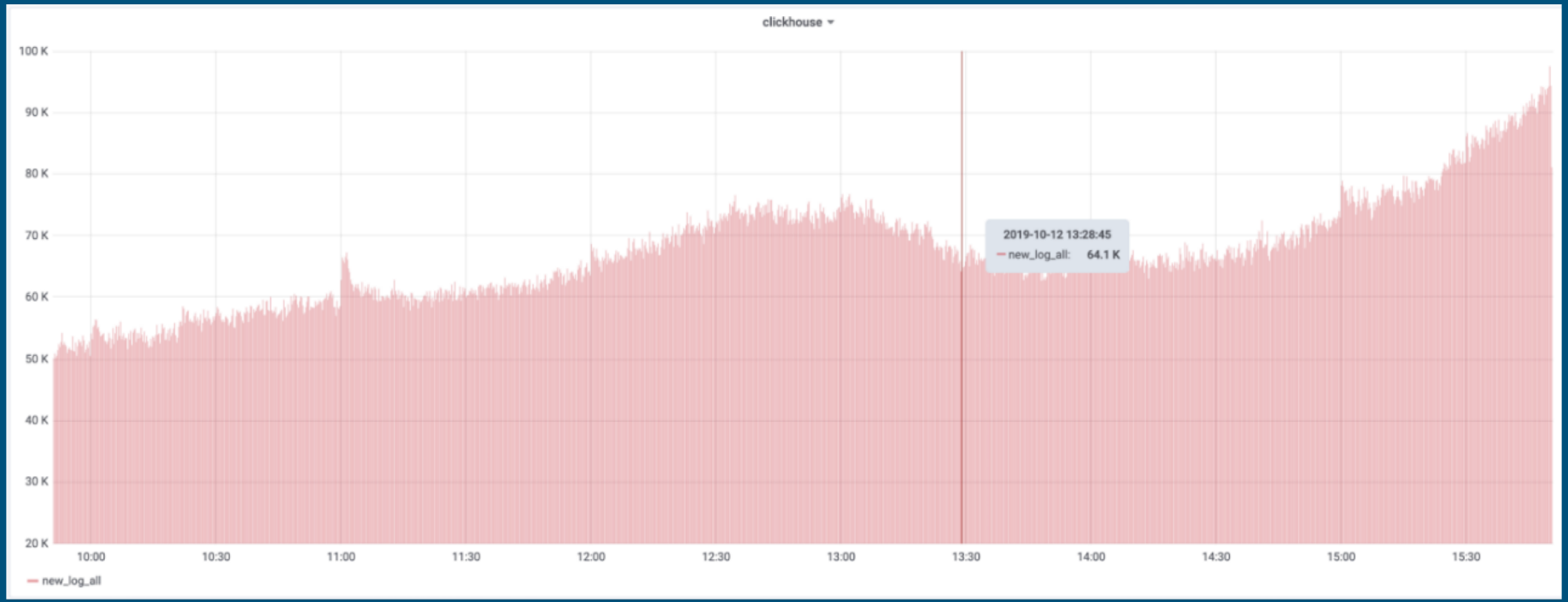
Company:
Xiaoxin Tech.

Industry:
Education

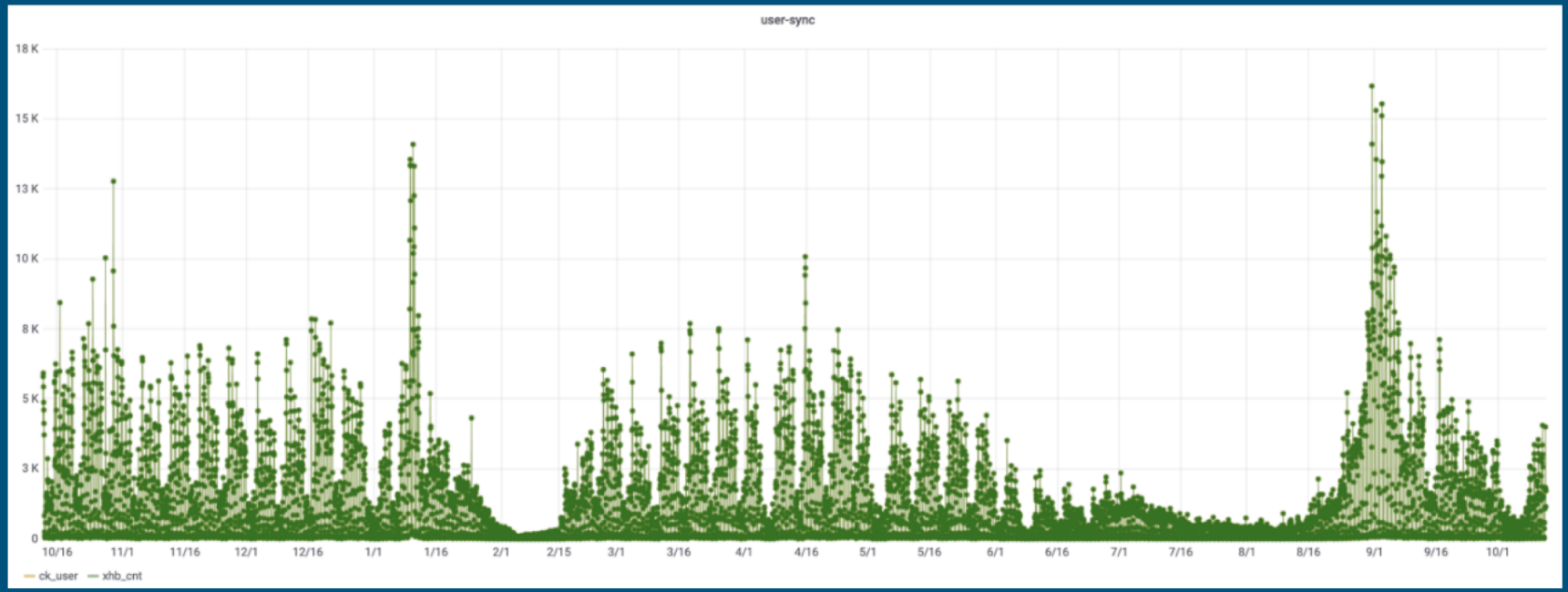
Team:
Big Data

Leader:
wangchao@xiaoheiban.cn

100 billion data this year till now

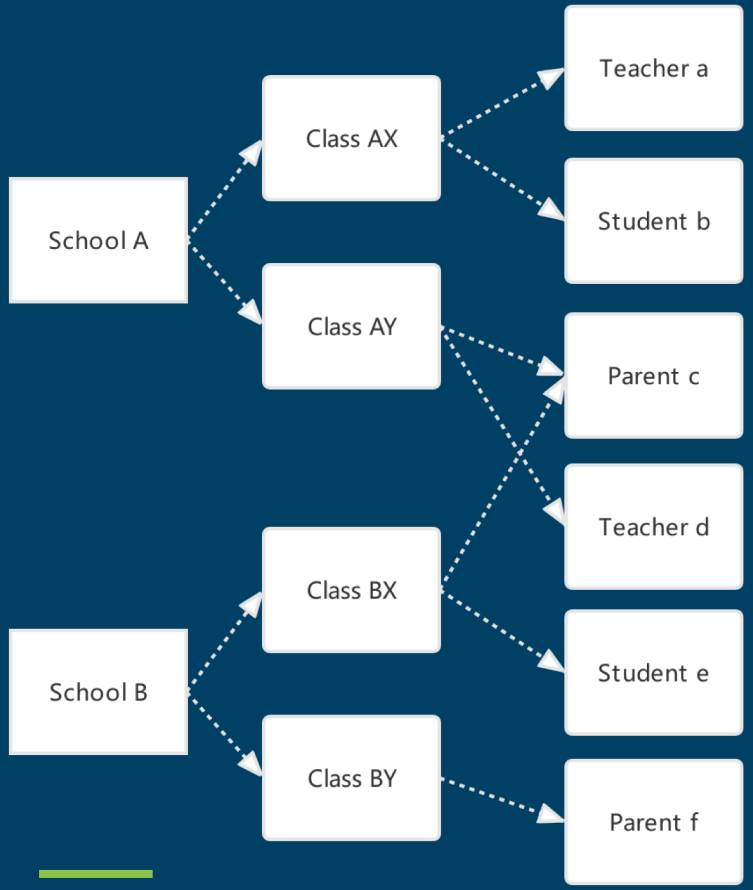


30 million users



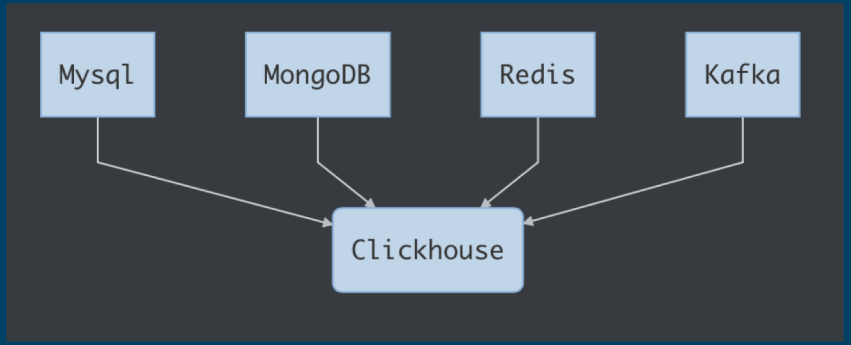
We use Clickhouse in our daily tasks

Chanllenges



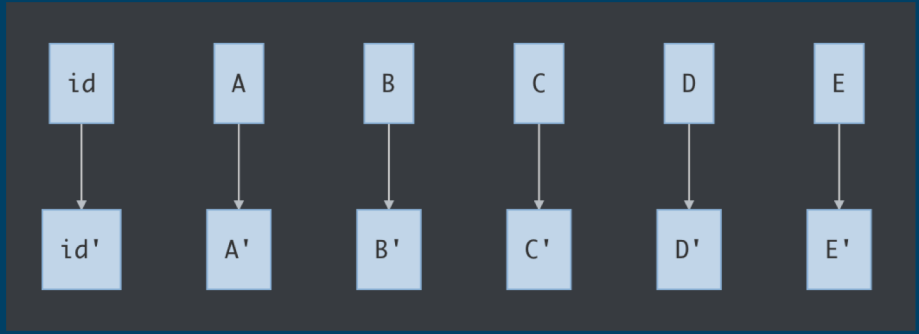
Chanllenges

Complex Datasource



Chanllenges

Frequent Updates



Possible Solutions 1.

Replay binlog/oplog CRUD directly

Can’t update/delete table frequently in Clickhouse

Possible Solutions 2.

MySQL Engine

Not suitable for big tables

Not suitable for MongoDB

MySQL

Creating a Database ¶

```
CREATE DATABASE [IF NOT EXISTS] db_name [ON CLUSTER cluster]
ENGINE = MySQL('host:port', 'database', 'user', 'password')
```

Possible Solutions 3.

Reinit whole table every day……

Possible Solutions 4.

CollapsingMergeTree

- FINAL is slow
- GROUP BY id HAVING sum(sign)>0
  - Need to use GROUP BY in every query
  - Not suitable for multi-column primary key

Our Solution: PTS

Key Features

- Only one config file needed for a new Clickhouse table
- Init and keep syncing data in one app for a table
- Sync multiple data source to Clickhouse in minutes

PTS

Provider Transform Sinker

- Major Provider Must Listen
- Merge sharding tables
- SolidKey

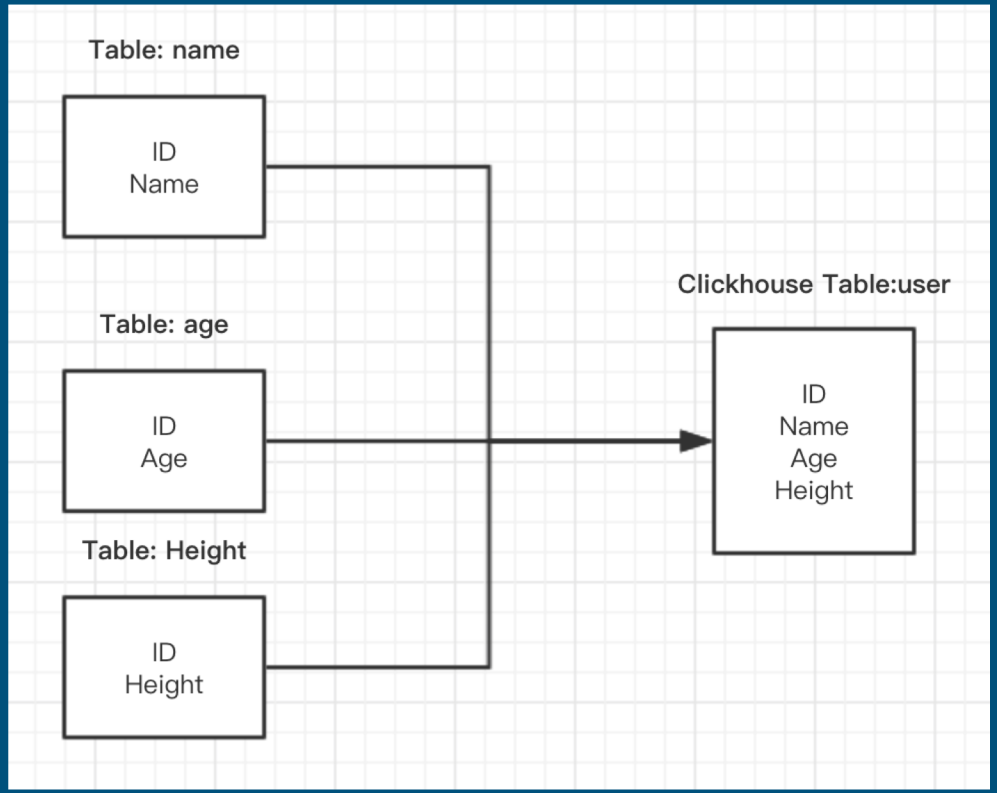
Typical Provider Config

```
{
        Type: mysql,            // mysql, mongodb, redis
        Listen: binlog, // binlog, kafka
        DataSource: user:pass@tcp(example.com:3306)/user,

        Table: user,
        QueryKeys: [ // usually primary key
                id
        ],
        Pairs: { // field mapping
                id: id,
                name: name
        }
}
```

PTS

Provider Transform Sinker

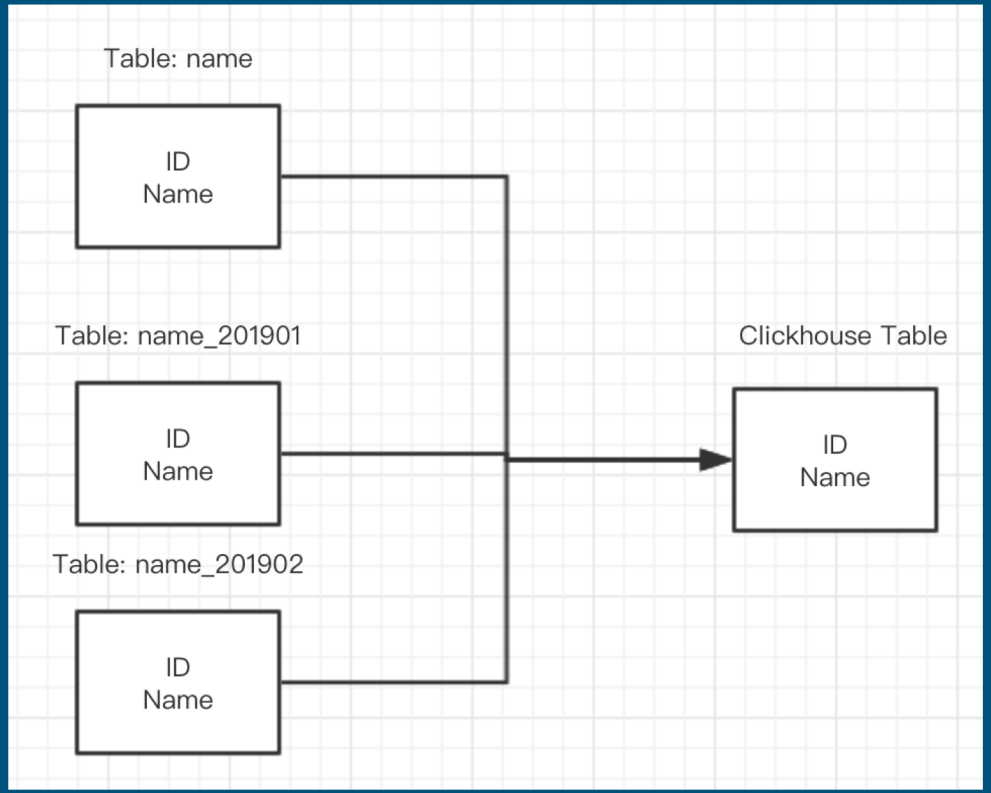
- Major Provider
- Secondary Providers



PTS

Provider Transform Sinker

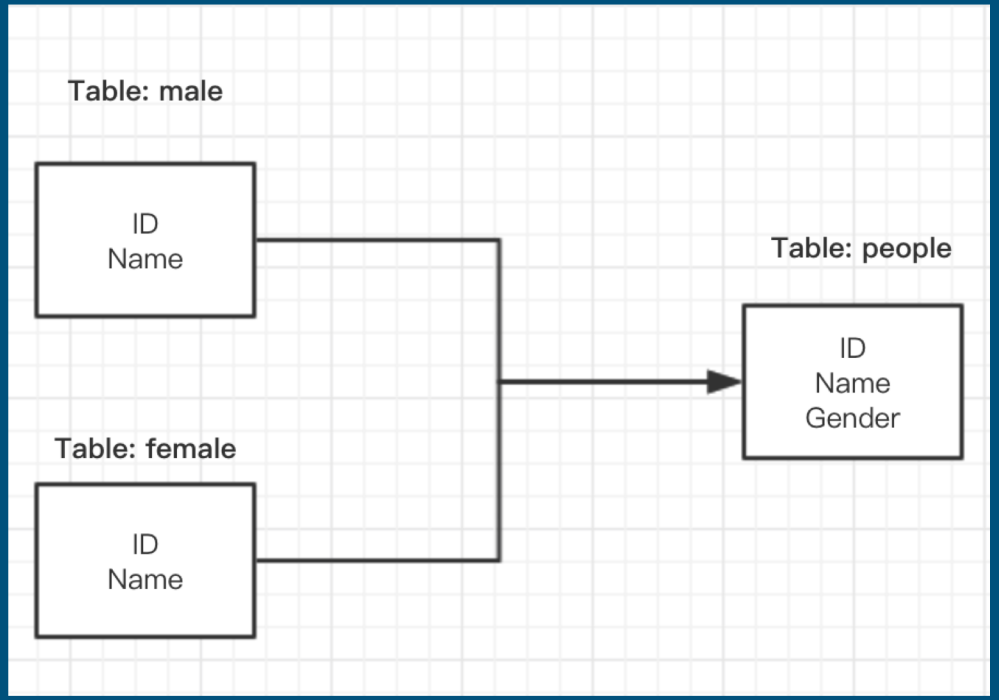
- Merge sharding tables



PTS

Provider Transform Sinker

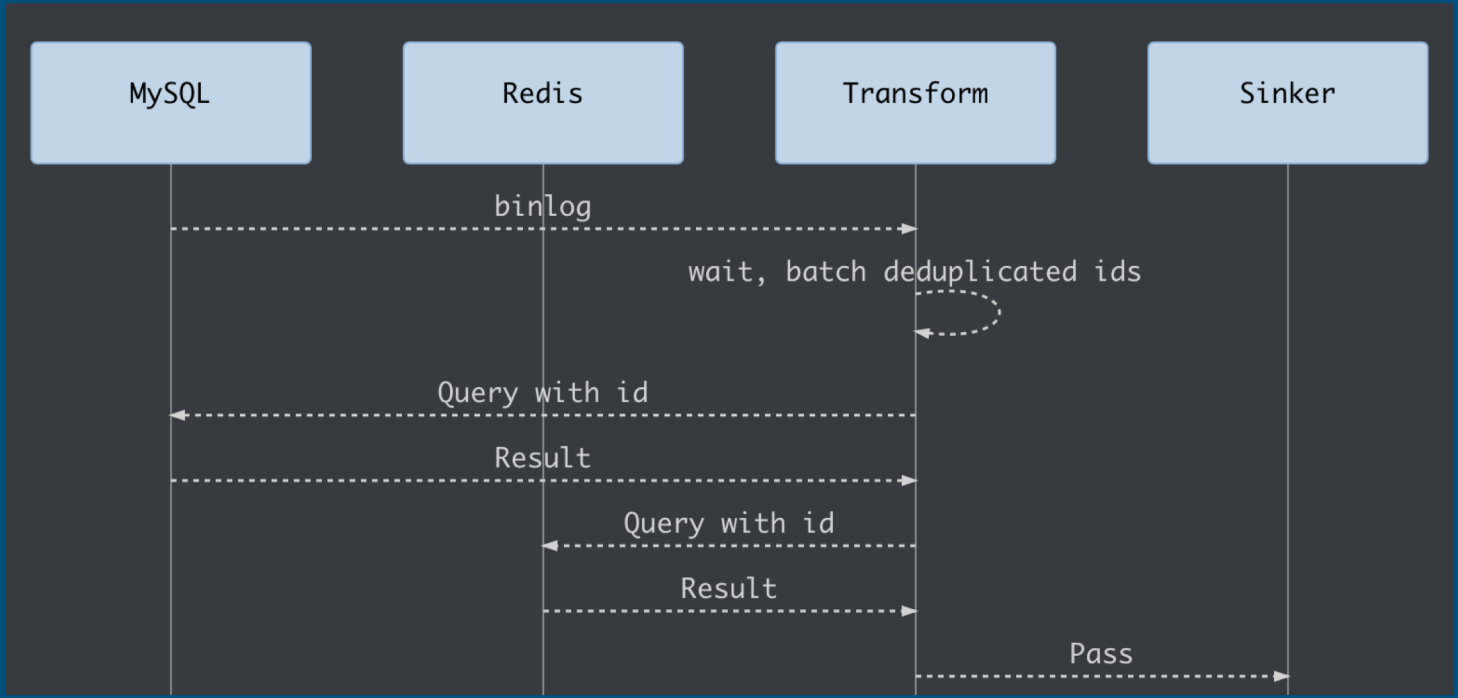
- SolidKey



PTS

Provider Transform Sinker

Transform



PTS

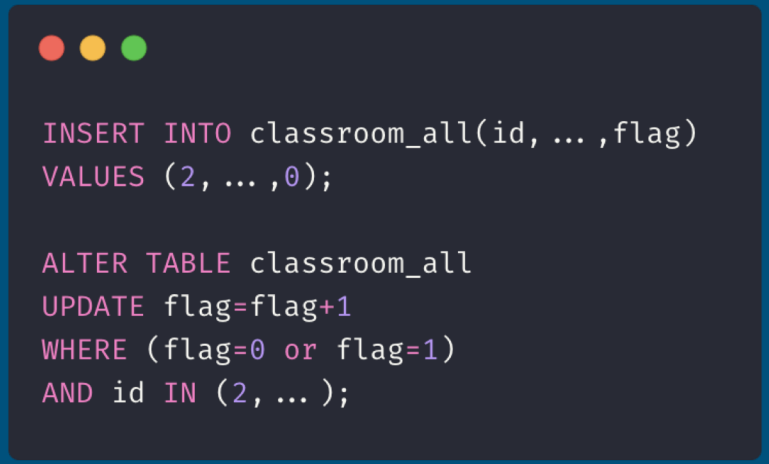
Provider Transform Sinker

- Batch Inserts
  - Time trigger
  - Number trigger
- Magical Flag

PTS

Magical Flag

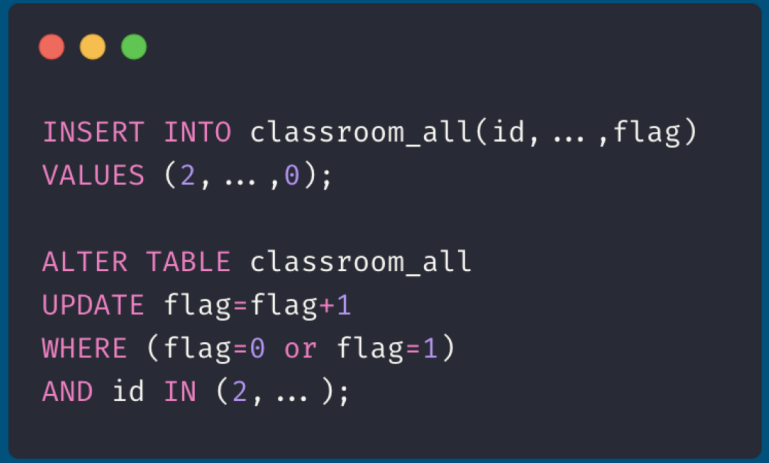
Quiz



PTS

Magical Flag

Create



```
INSERT INTO classroom_all(id, ... ,flag)
VALUES (2, ... ,0);

ALTER TABLE classroom_all
UPDATE flag=flag+1
WHERE (flag=0 or flag=1)
AND id IN (2, ... );
```

PTS

Magical Flag

Create

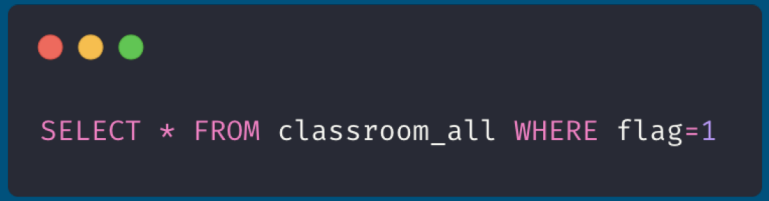
id	name	flag
2	Bob1	0

id	name	flag
2	Bob1	1

PTS

Magical Flag

Read



PTS

Magical Flag

Update

```
INSERT INTO classroom_all(id, ... ,flag)
VALUES (2, ... ,0);

ALTER TABLE classroom_all
UPDATE flag=flag+1
WHERE (flag=0 or flag=1)
AND id IN (2, ... );
```

PTS

Magical Flag

Update

id	name	flag
2	Bob1	1
2	Bob2	0

id	name	flag
2	Bob1	2
2	Bob2	1

PTS

Magical Flag

Delete

```
INSERT INTO classroom_all(id, ... ,flag) VALUES (2 ... ,3);

ALTER TABLE classroom_all
UPDATE flag=flag+1
WHERE flag=1
AND id IN (2 ... );
```

PTS

Magical Flag

Delete

id	name	flag
2	Bob1	2
2	Bob2	1
2	Bob2	3

id	name	flag
2	Bob1	2
2	Bob2	2
2	Bob2	3

PTS

Magical Flag

Add new column

```
SELECT id FROM source_table;
INSERT INTO classroom_all(id, ... ,flag) VALUES (2 ... ,5);

ALTER TABLE classroom_all
UPDATE flag=(flag+3)%7
WHERE flag=1
AND id IN (2 ... );

ALTER TABLE classroom_all
DELETE WHERE id IN (2 ... );
```

PTS

Magical Flag

Why fast?

Clickhouse Mutation

```
MergeTreeData::MutableDataPartPtr MergeTreeDataMergerMutator::mutatePartToTemporaryPart
...
NameSet files_to_skip = {"checksums.txt", "columns.txt"}; //files to skip hardlink
...
for (const auto & entry : in_header) // mutation fields, only flag
{
    IDataType::StreamCallback callback = [&](const IDataType::SubstreamPath & substream_path)
    {
        String stream_name = IDataType::getFileNameForStream(entry.name, substream_path);
        files_to_skip.insert(stream_name + ".bin");
        files_to_skip.insert(stream_name + mrk_extension);
    };
    IDataType::SubstreamPath stream_path;
    entry.type->enumerateStreams(callback, stream_path);
}
for (Poco::DirectoryIterator dir_it(source_part->getFullPath()); dir_it != dir_end; ++dir_it)
{
    if (files_to_skip.count(dir_it.name()))
        continue;
    Poco::Path destination(new_part_tmp_path);
    destination.append(dir_it.name());
    createHardLink(dir_it.path().toString(), destination.toString());
}
```

PTS

Magical Flag

Why fast?

Clickhouse Mutation

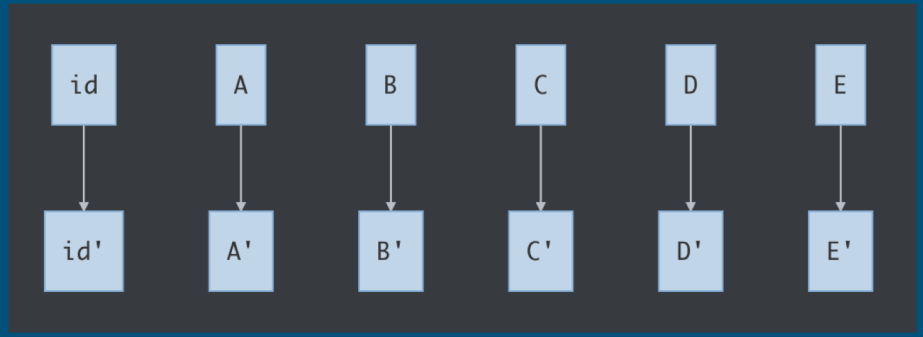
```
MergedColumnOnlyOutputStream out(
    data,
    in_header,
    new_part_tmp_path,
    /* sync = */ false,
    compression_codec,
    /* skip_offsets = */ false,
    unused_written_offsets,
    source_part->index_granularity,
    &source_part->index_granularity_info
);

in->readPrefix();
out.writePrefix();

Block block;
while (check_not_cancelled() && (block = in->read()))
{
    out.write(block);

    merge_entry->rows_written += block.rows();
    merge_entry->bytes_written_uncompressed += block.bytes();
}
```

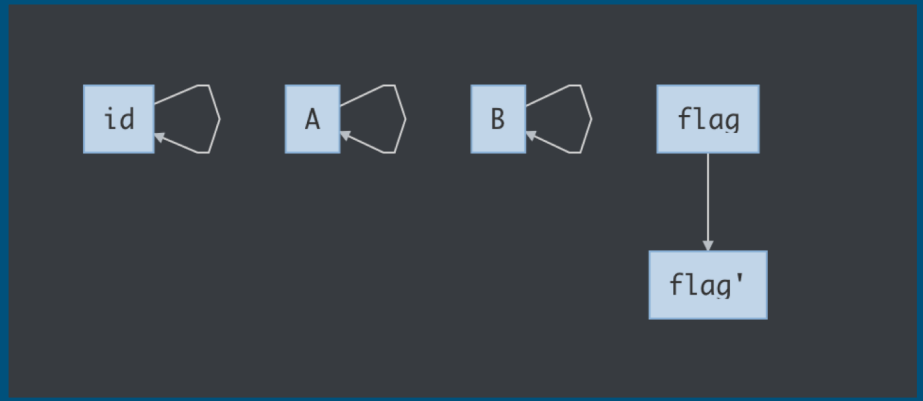
PTS

Magical Flag

Why fast?

Clickhouse Mutation





Trouble Shoot

- SQL is too big(max_ast_elements, max_expanded_ast_elements)
- Mutations are stuck (KILL MUTATION)
- Zookeeper OOM because of SQL length (Put ids in a Memory Engine temp table)

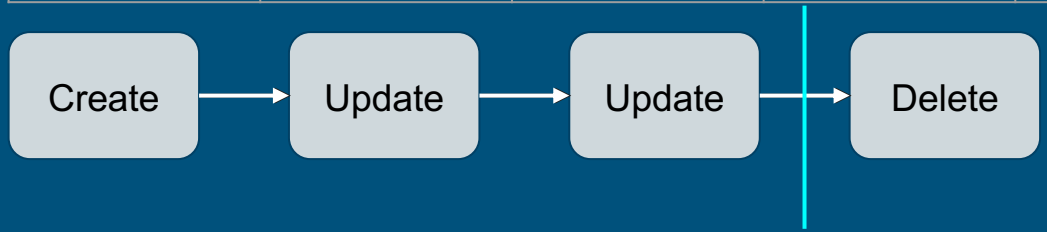
Final Product

- Only one config file needed for a new Clickhouse table
- Init and keep syncing data in one app for a table
- Sync multiple data source to Clickhouse in minutes

BONUS

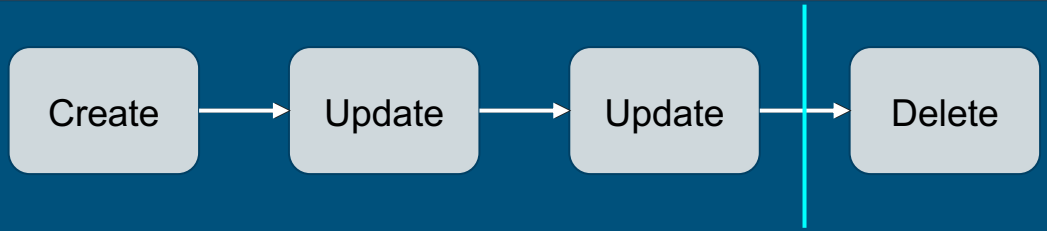
Time travel history state

insert_id	id	name	flag	update_time
UUID0	2	Bob1	2	2019-10-01 00:00:00
UUID1	2	Bob1.5	2	2019-10-01 12:00:00
UUID2	2	Bob2	2	2019-10-02 00:00:00
UUID3	2	Bob2	3	2019-10-03 00:00:00



BONUS Time travel history state

```
WITH toDateTime('2019-10-02 12:00:00') AS target_time
SELECT argMin(insert_id , target_time-update_time) insert_id,
       argMin(flag, target_time-update_time) flag
FROM classroom_all
WHERE target_time-update_time>0
GROUP BY id
HAVING flag<3;
```



Future

- Auto configure through web
- Auto deploy on Kubernetes
- Open source?
- Github: kevwan

Q&A

Thanks