Shinkorenok Arina

Group J4132c

Report

Clickhouse Lab

Table schemas

List of all tables:

```
SHOW TABLES FROM ashinkorenok_412704
```

```
┌─name─────────────────────────────────┐
│ .inner_id.786b11ea-35a4-4203-b86b-11ea35a4d203 │
│ .inner_id.9da06db5-5592-4eef-9da0-6db55592ceef │
│ current_users_saldo_distributed      │
│ current_users_saldo_mv               │
│ monthly_transaction_sums_distributed │
│ monthly_transaction_sums_mv          │
│ transactions                         │
│ transactions_distributed             │
└──────────────────────────────────────┘
```

Main table – *transactions* (and *transactions_distributed*):

```
DESCRIBE TABLE ashinkorenok_412704.transactions
-- Similarly for
-- DESCRIBE TABLE ashinkorenok_412704.transactions_distributed
```

```
┌─name─────────┬─type─────┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ amount       │ Float32  │              │                    │         │                  │                │
│ datetime     │ DateTime │              │                    │         │                  │                │
│ important    │ UInt8    │              │                    │         │                  │                │
│ user_id_out  │ UInt32   │              │                    │         │                  │                │
│ user_id_in   │ UInt32   │              │                    │         │                  │                │
└──────────────┴──────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
```

MV and its distributed version for the task *"The sums for incoming and outcoming transactions by months for each user"*:

```
DESCRIBE TABLE ashinkorenok_412704.monthly_transaction_sums_mv
-- Similarly for
-- DESCRIBE TABLE ashinkorenok_412704.monthly_transaction_sums_distributed
```

```
┌─name──────┬─type─────────────┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ user_id   │ UInt32           │              │                    │         │                  │                │
│ date      │ String           │              │                    │         │                  │                │
│ total_in  │ Decimal(18, 2)   │              │                    │         │                  │                │
│ total_out │ Decimal(18, 2)   │              │                    │         │                  │                │
└───────────┴──────────────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
```

MV and its distributed version for the task *"Users' saldo for the current moment"*:

```
DESCRIBE TABLE ashinkorenok_412704.current_users_saldo_mv
-- Similarly for
-- DESCRIBE TABLE ashinkorenok_412704.current_users_saldo_distributed
```

name	type	default_type	default_expression	comment	codec_expression	ttl_expression
user_id	UInt32					
saldo	Decimal(18, 2)					

Main table, sharding expression

Script code for creating the main table (transactions):

```
CREATE TABLE ashinkorenok_412704.transactions ON CLUSTER
kube_clickhouse_cluster
(
    amount Float32,
    datetime DateTime,
    important UInt8,
    user_id_out UInt32,
    user_id_in UInt32
)
ENGINE = MergeTree()
PARTITION BY toYYYYMM(datetime)
ORDER BY (important, user_id_out, user_id_in, amount);
```

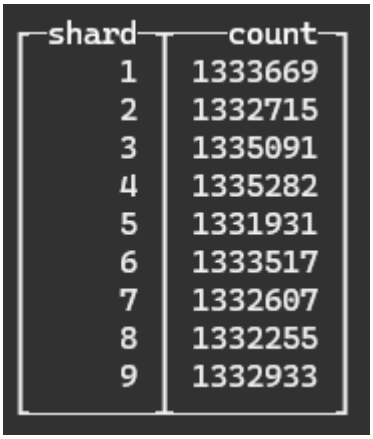
Script code for creating a distributed table for the main table (transactions_distribute):

```
CREATE TABLE ashinkorenok_412704.transactions_distributed
ON CLUSTER kube_clickhouse_cluster
AS ashinkorenok_412704.transactions
ENGINE = Distributed(kube_clickhouse_cluster, ashinkorenok_412704,
transactions, xxHash64(datetime));
```

The `xxHash64` hashing algorithm was chosen to express sharding due to its high speed and efficiency, which is very important for processing large amounts of data in real-time environments. Its non-cryptographic nature ensures minimal computational overhead while distributing data evenly across cluster nodes, which reduces the risk of data hotspots and improves overall query performance. In addition, xxHash64 is known for its low collision rate, which helps maintain data integrity and consistency.

Distribution of data by shards:

```
SELECT
    _shard_num AS shard,
    COUNT() AS count
FROM ashinkorenok_412704.transactions_distributed
GROUP BY shard
ORDER BY shard;
```



shard	count
1	1333669
2	1332715
3	1335091
4	1335282
5	1331931
6	1333517
7	1332607
8	1332255
9	1332933

Materialized Views

1. The sums for incoming and outcoming transactions by months for each user.

Script for creating MV:

```
CREATE MATERIALIZED VIEW ashinkorenok_412704.monthly_transaction_sums_mv
ON CLUSTER kube_clickhouse_cluster
ENGINE = AggregatingMergeTree
ORDER BY (user_id, date)
AS WITH
incoming AS (
    SELECT
        user_id_in AS user_id,
        formatDateTime(datetime, '%m.%Y') AS date,
        sumState(amount) AS total_in
    FROM ashinkorenok_412704.transactions
    GROUP BY user_id, date
),
outgoing AS (
    SELECT
        user_id_out AS user_id,
        formatDateTime(datetime, '%m.%Y') AS date,
        sumState(amount) AS total_out
    FROM ashinkorenok_412704.transactions
    GROUP BY user_id, date
)
SELECT
    i.user_id AS user_id,
    i.date as date,
    CAST(finalizeAggregation(i.total_in) AS Decimal(18, 2)) AS total_in,
    CAST(finalizeAggregation(o.total_out) AS Decimal(18, 2)) AS total_out
FROM incoming AS i
FULL OUTER JOIN outgoing AS o
ON i.user_id = o.user_id AND i.date = o.date
WHERE user_id != 0;
```

Script for creating a distributed MV:

```
CREATE TABLE ashinkorenok_412704.monthly_transaction_sums_distributed
ON CLUSTER kube_clickhouse_cluster
AS ashinkorenok_412704.monthly_transaction_sums_mv
ENGINE = Distributed(
    kube_clickhouse_cluster,
    ashinkorenok_412704,
    monthly_transaction_sums_mv,
    xxHash64(user_id)
);
```

Example of calling this MV:

```
SELECT
user_id,
date,
sum(total_in) as total_in,
sum(total_out) as total_out
FROM ashinkorenok_412704.monthly_transaction_sums_distributed
GROUP BY user_id, date
ORDER BY user_id, date
LIMIT 10;
```

user_id	date	total_in	total_out
1	01.2018	48198.33	57899.60
1	02.2018	54189.25	41120.82
1	03.2018	44345.81	38580.70
1	04.2018	50036.72	52733.60
1	05.2018	50458.78	49866.47
1	06.2018	48170.01	62601.17
1	07.2018	47026.45	54746.34
1	08.2018	56406.57	39402.42
1	09.2018	47653.10	52476.84
1	10.2018	50680.31	47336.41

2. Users' saldo for the current moment.

Script for creating MV:

```
CREATE MATERIALIZED VIEW ashinkorenok_412704.current_users_saldo_mv
ON CLUSTER kube_clickhouse_cluster
ENGINE = AggregatingMergeTree
ORDER BY user_id
AS WITH
incoming_sums AS (
    SELECT
        user_id_in AS user_id,
        sumState(amount) AS total_in
    FROM ashinkorenok_412704.transactions
    GROUP BY user_id
),
outgoing_sums AS (
    SELECT
        user_id_out AS user_id,
        sumState(amount) AS total_out
    FROM ashinkorenok_412704.transactions
    GROUP BY user_id
)
SELECT
    inc.user_id AS user_id,
    CAST(finalizeAggregation(inc.total_in) -
COALESCE(finalizeAggregation(out.total_out), 0) AS Decimal(18, 2)) AS
saldo
FROM incoming_sums AS inc
FULL OUTER JOIN outgoing_sums AS out
ON inc.user_id = out.user_id
WHERE user_id != 0;
```

Script for creating a distributed MV:

```
CREATE TABLE ashinkorenok_412704.current_users_saldo_distributed
ON CLUSTER kube_clickhouse_cluster AS
ashinkorenok_412704.current_users_saldo_mv
ENGINE = Distributed(
    kube_clickhouse_cluster,
    ashinkorenok_412704,
    current_users_saldo_mv,
    xxHash64(user_id)
);
```

Example of calling this MV:

```
SELECT
user_id,
sum(saldo) as saldo
FROM ashinkorenok_412704.current_users_saldo_distributed
GROUP BY user_id
ORDER BY user_id
LIMIT 10;
```

user_id	saldo
1	16700.85
2	8173.69
3	-29191.35
4	-27961.06
5	33883.98
6	6250.16
7	30681.65
8	-32012.44
9	19803.78
10	16318.77