Движки баз данных в ClickHouse

Александр Токмаков, разработчик ClickHouse

Движки баз данных

- Ordinary

```
:) CREATE DATABASE new_db;

Ok.

:) SHOW CREATE DATABASE new_db FORMAT TSVRaw;

CREATE DATABASE new_db
ENGINE = Ordinary
```

Движки баз данных

- Ordinary
- Memory
- Lazy
- Dictionary
- MySQL

```
:) CREATE DATABASE new_db;

Ok.

:) SHOW CREATE DATABASE new_db FORMAT TSVRaw;

CREATE DATABASE new_db
ENGINE = Ordinary
```

Движки баз данных

- Ordinary
- Memory
- Lazy
- Dictionary
- MySQL
- Atomic
- MaterializeMySQL
- [WIP] Replicated

```
:) CREATE DATABASE new_db;

Ok.

:) SHOW CREATE DATABASE new_db FORMAT TSVRaw;

CREATE DATABASE new_db
ENGINE = Ordinary
```

Ordinary

- Обычный движок

Ordinary

- Обычный движок

```
$ cat /var/lib/clickhouse/metadata/new_db.sql
ATTACH DATABASE new_db
ENGINE = Ordinary

$ cat /var/lib/clickhouse/metadata/new_db/test.sql
ATTACH TABLE test
(
    `n` UInt64
)
ENGINE = MergeTree
ORDER BY n
SETTINGS index_granularity = 8192

$ ls /var/lib/clickhouse/data/new_db/test/
all_1_1_0  all_2_2_0  detached  format_version.txt
```

Memory

- Не сохраняет метаданные таблиц на диске
- Используется для базы default в clickhouse-local

Lazy

- Похожа на Ordinary
- Но только для таблиц Log, StripeLog и TinyLog
- Загружает таблицы лениво
- Выгружает неиспользуемые

Dictionary

- Содержит таблицы с движком Dictionary для всех словарей

Dictionary

- Содержит таблицы с движком Dictionary для всех словарей

```
:) SELECT dictGet('default.dict', 'value1', toUInt64(1)) AS v1,
          dictGet('default.dict', 'value1', toUInt64(1)) AS v2

┌─v1─┬─v2─┐
│    │    │
└────┴────┘


:) CREATE DATABASE dict_db ENGINE=Dictionary
:) select * from dict_db.`default.dict`

┌───key─┬─value1─┬─value2─┐
│ 12345 │ hello  │ world  │
│ 67890 │ test   │ dict   │
└───────┴────────┴────────┘
```

MySQL

- Подключается к MySQL серверу, загружает список таблиц
- Создаёт таблицы с ENGINE=MySQL
- Запросом DROP можно скрыть таблицу из базы

Atomic

- Альтернатива Ordinary
- Неблокирующие DROP и RENAME TABLE
- Атомарный EXCHANGE TABLES t1 AND t2
- RENAME DICTIONARY, RENAME DATABASE
- Уникальные пути в ФС и в ZooKeeper для ReplicatedMergeTree

Atomic и недостатки Ordinary

- DB::Exception: Possible deadlock avoided. Client should retry.
- В Ordinary имя таблицы присутствует в нескольких местах
- В директории с данными таблицы может остаться мусор
- RENAME TABLE a TO a_old, a_new TO a;

Atomic

- У баз, таблиц и DDL-словарей есть UUID

```
$ cat metadata/atomic_db.sql
ATTACH DATABASE _ UUID 'd8ece42d-06d6-401c-bc6d-cc4317cf1f23'
ENGINE = Atomic

$ ls -l /var/lib/clickhouse/metadata/atomic_db
... metadata/atomic_db -> /var/lib/clickhouse/store/d8e/d8ece42d-06d6-401c-bc6d-cc4317cf1f23

$ cat store/d8e/d8ece42d-06d6-401c-bc6d-cc4317cf1f23/test.sql
ATTACH TABLE _ UUID '773738d0-f2ec-41ba-add8-d852b7e1b69d'
...

$ ls store/773/773738d0-f2ec-41ba-add8-d852b7e1b69d/
all_1_1_0  all_2_2_0  detached  format_version.txt
```

Atomic и DROP TABLE

- Таблица помечается удалённой
- Запросы продолжают выполнятся
- Фоновый поток собирает мусор

```
<database_atomic_delay_before_drop_table_sec>
    480
</database_atomic_delay_before_drop_table_sec>
```

Atomic и DROP TABLE

- Особенности неблокирующего DROP:

```
:) CREATE TABLE atomic_db.test (n UInt64)
ENGINE=ReplicatedMergeTree('/tables/test', 'r1') ORDER BY n

Ok.

:) DROP TABLE atomic_db.test

Ok.

:) CREATE TABLE atomic_db.test (n UInt64)
ENGINE=ReplicatedMergeTree('/tables/test', 'r1') ORDER BY n

Code: 253. DB::Exception: Received from localhost:9000. DB::Exception: Replica
/tables/test/replicas/r1 already exists..
```

Atomic и ReplicatedMergeTree

```
:) CREATE TABLE atomic_db.test ON CLUSTER cluster (n UInt64)
ENGINE=ReplicatedMergeTree ORDER BY n
```

Atomic и ReplicatedMergeTree

```
:) CREATE TABLE atomic_db.test ON CLUSTER cluster (n UInt64)
ENGINE=ReplicatedMergeTree ORDER BY n

:) CREATE TABLE atomic_db.test ON CLUSTER cluster (n UInt64)
ENGINE=ReplicatedMergeTree('/clickhouse/tables/{uuid}/{shard}', '{replica}') ORDER BY n
```

```
<default_replica_path>/clickhouse/tables/{uuid}/{shard}</default_replica_path>

<default_replica_name>{replica}</default_replica_name>
```

Atomic и DETACH TABLE

- Особенности неблокирующего DETACH:

```
:) DETACH TABLE atomic_db.test

Ok.

:) ATTACH TABLE atomic_db.test

Code: 57. DB::Exception: Received from localhost:9000. DB::Exception: Cannot attach table
with UUID 33e8fd7d-44c0-4ed3-9663-eb4c42e8edc2, because it was detached but still used by
some query. Retry later.
```

Atomic и DETACH TABLE

- Особенности неблокирующего DETACH:

```
:) DETACH TABLE atomic_db.test SYNC

Ok.

:) ATTACH TABLE atomic_db.test

Ok.
```

Atomic и DETACH TABLE

- Особенности неблокирующего DETACH:

```
:) DETACH TABLE atomic_db.test SYNC

Ok.

:) ATTACH TABLE atomic_db.test

Ok.
```

```
:) DROP TABLE atomic_db.replicated_table SYNC

Ok.
```

Atomic и Changelog

- `Появилось в 20.4`
- `allow_experimental_database_atomic=1` в 20.7
- Включена для базы system в 20.8
- `default_database_engine='Atomic'` в 20.10
- DROP/DETACH … SYNC в 20.10

MaterializeMySQL

- ClickHouse как реплика MySQL
- Читает Binlog и выполняет запросы в ClickHouse
- Можно попробовать с 20.8

MaterializeMySQL

```
mysql> CREATE DATABASE db;

mysql> CREATE TABLE db.test (a INT
PRIMARY KEY, b INT);

mysql> INSERT INTO db.test VALUES (1,
11), (2, 22);

mysql> DELETE FROM db.test WHERE a=1;

mysql> ALTER TABLE db.test ADD COLUMN
c VARCHAR(16);

mysql> UPDATE db.test SET c='Wow!',
b=222;
```

```
:) CREATE DATABASE mysql ENGINE =
MaterializeMySQL('localhost:3306', 'db',
'user', '***')
:) SHOW TABLES FROM mysql
┌─name─┐
│ test │
└──────┘
:) SELECT * FROM mysql.test
┌─a─┬──b─┐
│ 1 │ 11 │
│ 2 │ 22 │
└───┴────┘
:) SELECT * FROM mysql.test
┌─a─┬───b─┬─c────┐
│ 2 │ 222 │ Wow! │
└───┴─────┴──────┘
```

MaterializeMySQL и DDL-запросы

- DDL запросы конвертируются в ClickHouse SQL
- Движок ReplacingMergeTree со столбцами _sign и _version
- PRIMARY KEY + INDEX -> ORDER BY
- PARTITION BY toYYYYMM(datetime) или intDiv(num, const)

MaterializeMySQL и DML-запросы

- _version - аналог счётчика транзакций
- DELETE превращается в INSERT с _sign=-1
- UPDATE превращается в DELETE + INSERT
- Вставки через буфер

MaterializeMySQL и SELECT

- Если _version не используется в запросе - добавляется FINAL
- Если _sign не используется - добавляется WHERE _sign=1
- FINAL выбирает строки с максимальным _version
- WHERE _sign=1 пропускает удалённые строки

MaterializeMySQL - что происходит?

```
mysql> CREATE DATABASE db;
mysql> CREATE TABLE db.test (a INT PRIMARY KEY, b INT);


mysql> INSERT INTO db.test VALUES (1, 11), (2, 22);
mysql> DELETE FROM db.test WHERE a=1;
mysql> ALTER TABLE db.test ADD COLUMN c VARCHAR(16);
mysql> UPDATE db.test SET c='Wow!', b=222;


mysql> SELECT * FROM test;
+---+------+------+
| a | b    | c    |
+---+------+------+
| 2 |  222 | Wow! |
+---+------+------+
```

MaterializeMySQL - что происходит?

```
:) CREATE DATABASE db ENGINE=Ordinary;
mysql> CREATE TABLE db.test (a INT PRIMARY KEY, b INT);


mysql> INSERT INTO db.test VALUES (1, 11), (2, 22);
mysql> DELETE FROM db.test WHERE a=1;
mysql> ALTER TABLE db.test ADD COLUMN c VARCHAR(16);
mysql> UPDATE db.test SET c='Wow!', b=222;


mysql> SELECT * FROM test;
+---+------+------+
| a | b    | c    |
+---+------+------+
| 2 |  222 | Wow! |
+---+------+------+
```

MaterializeMySQL - что происходит?

```
:) CREATE DATABASE db ENGINE=Ordinary;
:) CREATE TABLE db.test(a Int32, b Nullable(Int32),  _sign Int8, _version UInt64)
ENGINE = ReplacingMergeTree(_version) PARTITION BY intDiv(a, 4294967) ORDER BY a;

mysql> INSERT INTO db.test VALUES (1, 11), (2, 22);
mysql> DELETE FROM db.test WHERE a=1;
mysql> ALTER TABLE db.test ADD COLUMN c VARCHAR(16);
mysql> UPDATE db.test SET c='Wow!', b=222;


mysql> SELECT * FROM test;
+---+------+------+
| a | b    | c    |
+---+------+------+
| 2 |  222 | Wow! |
+---+------+------+
```

MaterializeMySQL - что происходит?

```
:) CREATE DATABASE db ENGINE=Ordinary;
:) CREATE TABLE db.test(a Int32, b Nullable(Int32),  _sign Int8, _version UInt64)
ENGINE = ReplacingMergeTree(_version) PARTITION BY intDiv(a, 4294967) ORDER BY a;

:) INSERT INTO db.test VALUES (1, 11, /* sing */  1, /* version */ 1), (2, 22, 1, 1);
mysql> DELETE FROM db.test WHERE a=1;
mysql> ALTER TABLE db.test ADD COLUMN c VARCHAR(16);
mysql> UPDATE db.test SET c='Wow!', b=222;


mysql> SELECT * FROM test;
+---+------+------+
| a | b    | c    |
+---+------+------+
| 2 |  222 | Wow! |
+---+------+------+
```

MaterializeMySQL - что происходит?

```
:) CREATE DATABASE db ENGINE=Ordinary;
:) CREATE TABLE db.test(a Int32, b Nullable(Int32),  _sign Int8, _version UInt64)
ENGINE = ReplacingMergeTree(_version) PARTITION BY intDiv(a, 4294967) ORDER BY a;

:) INSERT INTO db.test VALUES (1, 11, /* sing */  1, /* version */ 1), (2, 22, 1, 1);
:) INSERT INTO db.test VALUES (1, 11, /* sing */ -1, /* version */ 2);
mysql> ALTER TABLE db.test ADD COLUMN c VARCHAR(16);
mysql> UPDATE db.test SET c='Wow!', b=222;


mysql> SELECT * FROM test;
+---+------+------+
| a | b    | c    |
+---+------+------+
| 2 |  222 | Wow! |
+---+------+------+
```

MaterializeMySQL - что происходит?

```
:) CREATE DATABASE db ENGINE=Ordinary;
:) CREATE TABLE db.test(a Int32, b Nullable(Int32),  _sign Int8, _version UInt64)
ENGINE = ReplacingMergeTree(_version) PARTITION BY intDiv(a, 4294967) ORDER BY a;

:) INSERT INTO db.test VALUES (1, 11, /* sing */  1, /* version */ 1), (2, 22, 1, 1);
:) INSERT INTO db.test VALUES (1, 11, /* sing */ -1, /* version */ 2);
:) ALTER TABLE db.test ADD COLUMN c Nullable(String);
mysql> UPDATE db.test SET c='Wow!', b=222;


mysql> SELECT * FROM test;
+---+------+------+
| a | b    | c    |
+---+------+------+
| 2 |  222 | Wow! |
+---+------+------+
```

MaterializeMySQL - что происходит?

```
:) CREATE DATABASE db ENGINE=Ordinary;
:) CREATE TABLE db.test(a Int32, b Nullable(Int32),  _sign Int8, _version UInt64)
ENGINE = ReplacingMergeTree(_version) PARTITION BY intDiv(a, 4294967) ORDER BY a;

:) INSERT INTO db.test VALUES (1, 11, /* sing */  1, /* version */ 1), (2, 22, 1, 1);
:) INSERT INTO db.test VALUES (1, 11, /* sing */ -1, /* version */ 2);
:) ALTER TABLE db.test ADD COLUMN c Nullable(String);
:) INSERT INTO db.test VALUES (2, 22, /* sing */ -1, /* version */ 3, ''),
                              (2, 222, /* sing */ 1, /* version */ 3, 'Wow!');

mysql> SELECT * FROM test;
+---+------+------+
| a | b    | c    |
+---+------+------+
| 2 |  222 | Wow! |
+---+------+------+
```

MaterializeMySQL - что происходит?

```
:) CREATE DATABASE db ENGINE=Ordinary;
:) CREATE TABLE db.test(a Int32, b Nullable(Int32),  _sign Int8, _version UInt64)
ENGINE = ReplacingMergeTree(_version) PARTITION BY intDiv(a, 4294967) ORDER BY a;

:) INSERT INTO db.test VALUES (1, 11, /* sing */  1, /* version */ 1), (2, 22, 1, 1);
:) INSERT INTO db.test VALUES (1, 11, /* sing */ -1, /* version */ 2);
:) ALTER TABLE db.test ADD COLUMN c Nullable(String);
:) INSERT INTO db.test VALUES (2, 22, /* sing */ -1, /* version */ 3, ''),
                              (2, 222, /* sing */ 1, /* version */ 3, 'Wow!');

:) SELECT a, b, c FROM db.test FINAL WHERE _sign=1;
┌─a─┬──b─┬─c────┐
│ 2 │ 222 │ Wow! │
└───┴─────┴──────┘
```

MaterializeMySQL - особенности

- Легко сломать репликацию
- Строки с _sign=-1 не удаляются (пока)
- Не работают каскадные UPDATE/DELETE
- Запрещены ручные манипуляции с базой и таблицами
- Движок экспериментальный

[WIP] Replicated

- Реплицируемая Atomic
- Метаданные в ZooKeeper
- Все DDL запросы - автоматически ON CLUSTER
- Новая реплика при подключении создаёт все таблицы
- https://github.com/ClickHouse/ClickHouse/pull/10485

Спасибо!