ClickHouse Features 2021

In Previous Episodes: Autumn/Winter 2020

EXPLAIN queries　　Compact and In-memory data parts

PostgreSQL wire protocol　　LDAP authentication

Atomic database　　Background data recompression

Column transformers　　RabbitMQ integration

256 bit Decimal　　Kerberos for Kafka and HDFS

Query obfuscation and normalization

Embedded Web UI

GRPC Protocol

Uncomment `<grpc_port>9100</grpc_port>` in config.

Feature parity with native protocol:

— TLS, compression, query progress,
query cancellation, sessions, external data...

Example: clickhouse-client with GRPC in Python.

```
utils/grpc-client/clickhouse-grpc-client.py
```

Developer — Vitaly Baranov. Available since 21.1.

Native Integration With PostgreSQL

— **PostgreSQL** storage engine;

— **postgresql** table function;

— **postgresql** dictionary source;

— **PostgreSQL** database engine as a view to all tables in PG database;

Similar to MySQL integration... but for PostgreSQL.

Upcoming: replication from PostgreSQL.

Developer — Ksenia Sumarokova. Available since 21.2.

OpenTelemetry Support

Trace your ClickHouse queries within your infrastructure:

— enabled by default;

— supports opentelemetry HTTP headers;

— multiple spans are annotated inside ClickHouse;

— data is written to `system.opentelemetry_log`;

Developer — Alexander Kuzmenkov. Available since 20.11.

New Compression Formats For Import/Export

Transparent import/export of compressed files:

— gz, brotli;

— xz, zstd;

Example:

```
CREATE TABLE github_events_url
(
...
) ENGINE = URL(
    'https://datasets.clickhouse.tech/github_events_v2.native.xz',
    Native);
```

Developer — Abi Palagashvili. Available since 21.1.

SQL Enhancements

— UNION DISTINCT;

— REPLACE TABLE and CREATE OR REPLACE TABLE;

— aggregate_functions_null_for_empty;

— Extended CTE;

— Type cast for IN subquery;

— SHOW [CHANGED] SETTINGS;

— POSITION(needle IN haystack);

— DIV / MOD;

— SELECT ALL;

ANTLR grammar

— experimental and incomplete;

— allows to reuse it in third-party tools;

E.g. implement syntax highlight in your query editor.

Developer — Ivan Lezhankin. Available since 21.1.

Map Data Type

Example: `map Map(String, String)`

`SELECT map['hello']`

Developer — Hexiaoting.

EmbeddedRocksDB Table Engine

— good for key-value queries;

— intented to use as dictionary source;

Developer — Sundy Li.

ALTER Improvements

ALTER UPDATE / DELETE IN PARTITION:

— Limit the scope of mutations.

ALTER DROP PART:

— Available for Replicated tables.

TTL now removes empty parts:

— No confusion with leftover parts.

Semi-Duplicate Text Search

Text analysis with Min-Hash and Sim-Hash algorithms
to find similar / copy-pasted content.

Min-Hash algorithm:

1. Extract every N-word shingle from the text.

2. Calculate hashes of every shingle.

3. Select M hashes with minimum value and M hashes with maximum value.

4. Calculate two total hashes of min-hashes and max-hashes.

5. Texts are duplicates if at least one of their total hashes equals.

Developer — ucasFL. Available since 21.1.

Statistical Tests

Test that two samples have different means, useful for A/B testing.

— studentTTest;

— welchTTest;

— mannWitneyUTest;

Rank correlation:

— rankCorr.

Developer — Nikita Mikhailov et all. Available since 21.1.

Upcoming Features

ClickHouse Roadmap is publicly available on GitHub:

https://github.com/ClickHouse/ClickHouse/issues/17623

I will show you only some highlights and examples.

Main Tasks

Provide alternative for ZooKeeper Nested and semistructured data

Limited support for transactions Backups Hedged requests

Window functions Separation of storage and compute

Short-circuit evaluation Projections Lightweight DELETE/UPDATE

Workload management User Defined Functions

Simplify replication JOIN improvements

Embedded documentation Pluggable auth with tokens

https://github.com/ClickHouse/ClickHouse/issues/17623

Support for Nested and Semistructured Data

Work in progress. Initial support in version 21.1.

Multiple nesting:

```
cart Nested(
    item_id UInt64,
    item_price Decimal(20, 5),
    features Nested(
        ...))
```

`SELECT cart.item_id, cart.features.f1 FROM table`

`SELECT cart.* FROM table`

Maps naturally to nested JSON and Protobuf.

Developer — Anton Popov. Available since 21.1.

Window Functions

Work in progress. Initial support in version 21.1.

```
SET allow_experimental_window_functions = 1
```

Already supported:
— OVER (PARTITION BY ... ORDER BY ...)
— aggregate functions over windows;
— WINDOW clause;

Upcoming:
— non-aggregate window functions (rank, etc...);
— frame specifications;

Developer — Alexander Kuzmenkov.

Projections: Announced today by Amos Bird!

Multiple data representations inside a single table.

— different data order;
— subset of columns;
— subset of rows;
— aggregation.

Work in progress.

Difference to materialized views:

— projections data is always consistent;
— updated atomically with the table;
— replicated in the same way as the table;
— projection can be automatically used for SELECT query.

Alternative to ZooKeeper

Work in Progress.

— ZooKeeper network protocol is implemented;
— Abstraction layer over ZooKeeper is used;
— ZooKeeper data model is implemented for testing;
— TestKeeperServer: a server with ZooKeeper data model for testing;

Benefits:
— less operational complexity;
— fix "zxid overflow" issue;
— fix the issue with max packet size;
— fix "session expired" due to gc pauses;
— improve memory usage;
— allow compressed snapshots;
— allow embedding into clickhouse-server.

Developer — Alexander Sapin.

Short-circuit Evaluation

```
SELECT IF(number = 0, 0, 123 % number) FROM numbers(10)
```

— division by zero.

```
SELECT * FROM numbers(10) WHERE number > 0 AND 10 % number > 0
```

— division by zero.

— both branches of IF, AND, OR are always evaluated.

```
SELECT * FROM
(
    SELECT * FROM numbers(10)
    WHERE number > 0
)
WHERE 10 % number > 0
```

— division by zero.

User Defined Functions

We are considering five ways to implement UDF, two of them are mandatory:

1. UDF as SQL expressions.

```
CREATE FUNCTION f AS x -> x + 1
```

2. UDF as executable script.

Interaction via pipes, data is serialized using supported formats.

Hedged Requests

Send distributed query to multiple replicas — to mitigate tail latencies.

This is needed for distributed queries on large clusters (with large "fanout").

Work in progress.

* The largest ClickHouse cluster in Yandex is 630+ servers,
but there are many larger clusters in other companies.

Developer — Pavel Kruglov and Nikolai Kochetov.

?

Read the official roadmap and ask your questions:

https://github.com/ClickHouse/ClickHouse/issues/17623