

# ClickHouseとは？

ClickHouseは、クエリのオンライン分析処理（OLAP）用の列指向のデータベース管理システム（DBMS）です。

「通常の」行指向のDBMSでは、データは次の順序で保存されます。

Row	WatchID	JavaEnable	Title	GoodEvent	EventTime
#0	89354350662	1	Investor Relations	1	2016-05-18 05:19:20
#1	90329509958	0	Contact us	1	2016-05-18 08:10:20
#2	89953706054	1	Mission	1	2016-05-18 07:38:00
#N	...	...	...	...	...

つまり、行に関連するすべての値は物理的に隣り合わせに格納されます。

行指向のDBMSの例：MySQL, Postgres および MS SQL Server

列指向のDBMSでは、データは次のように保存されます：

Row:	#0	#1	#2	#N
WatchID:	89354350662	90329509958	89953706054	...
JavaEnable:	1	0	1	...
Title:	Investor Relations	Contact us	Mission	...
GoodEvent:	1	1	1	...
EventTime:	2016-05-18 05:19:20	2016-05-18 08:10:20	2016-05-18 07:38:00	...

これらの例は、データが配置される順序のみを示しています。

異なる列の値は別々に保存され、同じ列のデータは一緒に保存されます。

列指向DBMSの例：Vertica, Paraccel (Actian Matrix and Amazon Redshift), Sybase IQ, Exasol, Infobright, InfiniDB, MonetDB (VectorWise and Actian Vector), LucidDB, SAP HANA, Google Dremel, Google PowerDrill, Druid および kdb+

異なったデータ格納の順序は、異なったシナリオにより適します。

データアクセスシナリオとは、クエリの実行内容、頻度、割合を指します。クエリで読み取られるの各種データの量（行、列、バイト）。データの読み取りと更新の関係。作業データのサイズとローカルでの使用方法。トランザクションが使用されるかどうか、およびそれらがどの程度分離されているか。データ複製と論理的整合性の要件。クエリの種類ごとの遅延とスループットの要件など。

システムの負荷が高いほど、使用シナリオの要件に一致するようにセットアップされたシステムをカスタマイズすることがより重要になり、このカスタマイズはより細かくなります。大きく異なるシナリオに等しく適したシステムはありません。システムがさまざまなシナリオに適応可能である場合、高負荷下では、システムはすべてのシナリオを同等に不十分に処理するか、1つまたはいくつかの可能なシナリオでうまく機能します。

## OLAPシナリオの主要なプロパティ

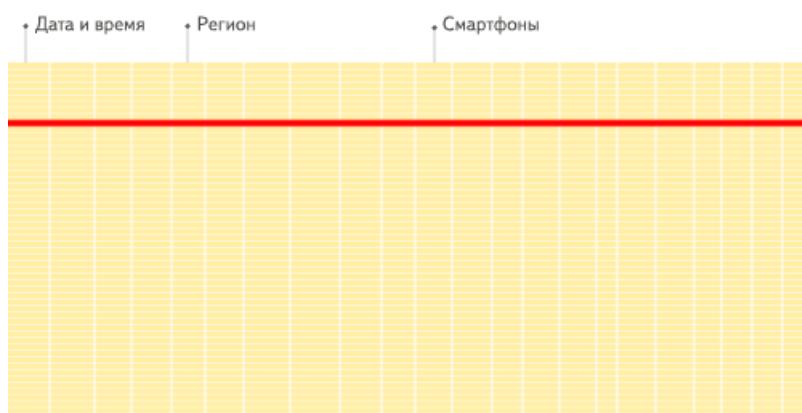
- リクエストの大部分は読み取りアクセス用である。
- データは、單一行ではなく、かなり大きなパッチ ( $> 1000$  行) で更新されます。または、まったく更新されない。
- データはDBに追加されるが、変更されない。
- 読み取りの場合、非常に多くの行がDBから抽出されるが、一部の列のみ。
- テーブルは「幅が広く」、多数の列が含まれる。
- クエリは比較的まれ（通常、サーバーあたり毎秒数百あるいはそれ以下の数のクエリ）。
- 単純なクエリでは、約50ミリ秒の遅延が容認される。
- 列の値はかなり小さく、数値や短い文字列（たとえば、URLごとに60バイト）。
- 単一のクエリを処理する場合、高いスループットが必要（サーバーあたり毎秒最大数十億行）。
- トランザクションは必要ない。
- データの一貫性の要件が低い。
- クエリごとに1つの大きなテーブルがある。1つを除くすべてのテーブルは小さい。
- クエリ結果は、ソースデータよりも大幅に小さくなる。つまり、データはフィルター処理または集計されるため、結果は単一サーバーのRAMに収まる。

OLAPシナリオは、他の一般的なシナリオ（OLTPやKey-Valueアクセスなど）とは非常に異なることが容易にわかります。したがって、まともなパフォーマンスを得るには、OLTPまたはKey-Value DBを使用して分析クエリを処理しようとするのは無意味です。たとえば、分析にMongoDBまたはRedisを使用しようとすると、OLAPデータベースに比べてパフォーマンスが非常に低下します。

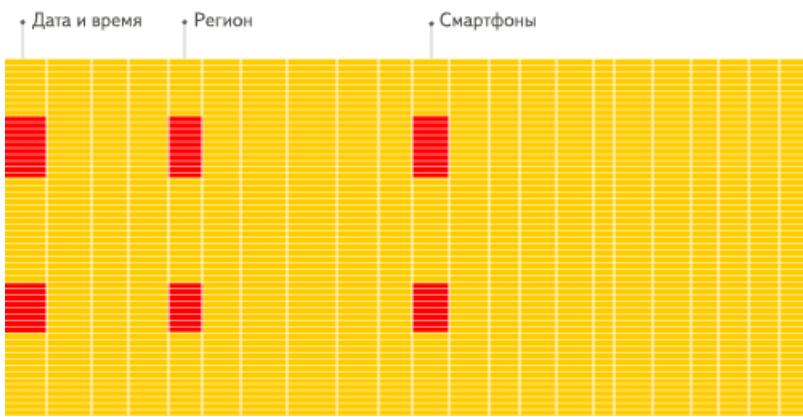
## OLAPシナリオで列指向データベースがよりよく機能する理由

列指向データベースは、OLAPシナリオにより適しています。ほとんどのクエリの処理が少なくとも100倍高速です。理由を以下に詳しく説明しますが、その根拠は視覚的に簡単に説明できます：

### 行指向DBMS



### 列指向DBMS



違いがわかりましたか？

## 入出力

- 分析クエリでは、少數のテーブル列のみを読み取る必要があります。列指向のデータベースでは、必要なデータのみを読み取ることができます。たとえば、100のうち5つの列が必要な場合、I/Oが20倍削減されることが期待できます。
- データはパケットで読み取られるため、圧縮が容易です。列のデータも圧縮が簡単です。これにより、I/Oボリュームがさらに削減されます。
- I/Oの削減により、より多くのデータがシステムキャッシュに収まります。

たとえば、「各広告プラットフォームのレコード数をカウントする」クエリでは、1つの「広告プラットフォームID」列を読み取る必要がありますが、これは非圧縮では1バイトの領域を要します。トラフィックのほとんどが広告プラットフォームからのものではない場合、この列は少なくとも10倍の圧縮が期待できます。高速な圧縮アルゴリズムを使用すれば、1秒あたり少なくとも非圧縮データに換算して数ギガバイトの速度でデータを展開できます。つまり、このクエリは、単一のサーバーで1秒あたり約数十億行の速度で処理できます。この速度はまさに実際に達成されます。

## CPU

クエリを実行するには大量の行を処理する必要があるため、個別の行ではなくベクター全体のすべての操作をディスパッチするか、ディスパッチコストがほとんどないようにクエリエンジンを実装すると効率的です。適切なディスクサブシステムでこれを行わないと、クエリインタープリターが必然的にCPUを失速させます。

データを列に格納し、可能な場合は列ごとに処理することは理にかなっています。

これを行うには2つの方法があります:

- ベクトルエンジン。すべての操作は、個別の値ではなく、ベクトルに対して記述されます。これは、オペレーションを頻繁に呼び出す必要がなく、ディスパッチコストが無視できることを意味します。操作コードには、最適化された内部サイクルが含まれています。
- コード生成。クエリ用に生成されたコードには、すべての間接的な呼び出しが含まれています。

これは、単純なクエリを実行する場合には意味がないため、「通常の」データベースでは実行されません。ただし、例外があります。たとえば、MemSQLはコード生成を使用して、SQLクエリを処理する際の遅延を減らします。（比較のために、分析DBMSではレイテンシではなくスループットの最適化が必要です。）

CPU効率のために、クエリ言語は宣言型（SQLまたはMDX）、または少なくともベクトル（J、K）でなければなりません。クエリには、最適化を可能にする暗黙的なループのみを含める必要があります。

## はじめに

あなたが ClickHouse は初めてで、そのパフォーマンスを体感するためのハンズオンを行いたい場合は、最初に [インストール](#) を行って下さい。

そうすると、以下を実施できるようになります。

- 詳細なチュートリアル
- データセット例を用いて試す

## インストール システム要件

ClickHouseは、x86\_64、AArch64、またはPowerPC64LE CPUアーキテクチャを持つLinux、FreeBSD、またはMac OS X上で実行できます。

公式のプレビルドバイナリは通常、x86\_64用にコンパイルされており、SSE 4.2命令セットを利用しています。現在のCPUがSSE 4.2をサポートしているかどうかを確認するコマンドは以下の通りです:

```
$ grep -q sse4_2 /proc/cpuinfo && echo "SSE 4.2 supported" || echo "SSE 4.2 not supported"
```

SSEをサポートしていないプロセッサ上でClickHouseを実行するには SSE 4.2 がサポートされているか、AArch64またはPowerPC64LEアーキテクチャで上で、適切な設定と調整を行い、ソースからClickHouseをビルドする必要があります。

## 利用可能なインストールオプション

### DEBパッケージから

Debian や Ubuntu 用にコンパイル済みの公式パッケージ `deb` を使用することをお勧めします。以下のコマンドを実行してパッケージをインストールして下さい:

```
sudo apt-get install apt-transport-https ca-certificates dirmngr
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv E0C56BD4

echo "deb https://repo.clickhouse.com/deb/stable/ main/" | sudo tee \
    /etc/apt/sources.list.d/clickhouse.list
sudo apt-get update

sudo apt-get install -y clickhouse-server clickhouse-client

sudo service clickhouse-server start
clickhouse-client
```

最新版を使いたい場合は、`stable`を`testing`に置き換えてください。（テスト環境ではこれを推奨します）

同様に、こちらからパッケージをダウンロードして、手動でインストールすることもできます。

### パッケージ

- `clickhouse-common-static` — コンパイルされた ClickHouse のバイナリファイルをインストールします。
- `clickhouse-server` — `clickhouse-server`へのシンボリックリンクを作成し、デフォルトのサーバ設定をインストールします。
- `clickhouse-client` — `clickhouse-server` および他のclient関連のツールへのシンボリックリンクを作成し、client関連の設定ファイルをインストールします。
- `clickhouse-common-static-dbg` — コンパイルされた ClickHouse のバイナリファイルを、デバッグ情報と一緒にインストールします。

### RPMパッケージから

CentOS、RedHat、その他すべてのrpmベースのLinuxディストリビューションでは、公式のコンパイル済み rpm パッケージを使用することを推奨します。

まず、公式リポジトリを追加する必要があります:

```
sudo yum install yum-utils
sudo rpm --import https://repo.clickhouse.com/CCLICKHOUSE-KEY.GPG
sudo yum-config-manager --add-repo https://repo.clickhouse.com/rpm/stable/x86_64
```

最新版を使いたい場合は stable を testing に置き換えてください。(テスト環境ではこれが推奨されています)。prestable もしばしば同様に利用できます。

そして、以下のコマンドを実行してパッケージをインストールします:

```
sudo yum install clickhouse-server clickhouse-client
```

同様に、こちら からパッケージをダウンロードして、手動でインストールすることもできます。

## Tgzアーカイブから

すべての Linux ディストリビューションで、deb や rpm パッケージがインストールできない場合は、公式のコンパイル済み tgz アーカイブを使用することをお勧めします。

必要なバージョンは、リポジトリ <https://repo.clickhouse.com/tgz/> から curl または wget でダウンロードできます。その後、ダウンロードしたアーカイブを解凍し、インストールスクリプトでインストールしてください。最新版の例は以下です:

```
export LATEST_VERSION=`curl https://api.github.com/repos/ClickHouse/ClickHouse/tags 2>/dev/null | grep -Eo '[0-9]+\.[0-9]+\.[0-9]+' | head -n 1`
curl -O https://repo.clickhouse.com/tgz/clickhouse-common-static-$LATEST_VERSION.tgz
curl -O https://repo.clickhouse.com/tgz/clickhouse-common-static-dbg-$LATEST_VERSION.tgz
curl -O https://repo.clickhouse.com/tgz/clickhouse-server-$LATEST_VERSION.tgz
curl -O https://repo.clickhouse.com/tgz/clickhouse-client-$LATEST_VERSION.tgz

tar -xzvf clickhouse-common-static-$LATEST_VERSION.tgz
sudo clickhouse-common-static-$LATEST_VERSION/install/doinst.sh

tar -xzvf clickhouse-common-static-dbg-$LATEST_VERSION.tgz
sudo clickhouse-common-static-dbg-$LATEST_VERSION/install/doinst.sh

tar -xzvf clickhouse-server-$LATEST_VERSION.tgz
sudo clickhouse-server-$LATEST_VERSION/install/doinst.sh
sudo /etc/init.d/clickhouse-server start

tar -xzvf clickhouse-client-$LATEST_VERSION.tgz
sudo clickhouse-client-$LATEST_VERSION/install/doinst.sh
```

本番環境では、最新の stable バージョンを使うことをお勧めします。GitHub のページ <https://github.com/ClickHouse/ClickHouse/tags> で接尾辞 -stable となっているバージョン番号として確認できます。

## Dockerイメージから

Docker内でClickHouseを実行するには、次の DockerHub のガイドに従います。それらのイメージでは内部で公式の deb パッケージを使っています。

## 非標準環境向けの事前コンパイルされたバイナリから

非LinuxオペレーティングシステムとAArch64 CPUアーキテクチャのために、ClickHouseのビルトは master ブランチ の最新のコミットからクロスコンパイルされたバイナリを提供しています。(数時間の遅延があります)

- macOS — curl -O 'https://builds.clickhouse.com/master/macos/clickhouse' && chmod a+x ./clickhouse
- FreeBSD — curl -O 'https://builds.clickhouse.com/master/freebsd/clickhouse' && chmod a+x ./clickhouse
- AArch64 — curl -O 'https://builds.clickhouse.com/master/aarch64/clickhouse' && chmod a+x ./clickhouse

ダウンロード後、`clickhouse client` を使ってサーバーに接続したり、`clickhouse local` を使ってローカルデータを処理したりすることができます。`clickhouse server` を実行するには、GitHubから `server` と `users` の設定ファイルを追加でダウンロードする必要があります。

これらのビルドは十分にテストされていないため、本番環境での使用は推奨されていませんが、自己責任で行うことができます。これらでは、ClickHouseの機能のサブセットのみが利用可能です。

## ソースから

ClickHouseを手動でコンパイルするには、次の `Linux` または `Mac OS X` の指示に従ってください。

パッケージをコンパイルしてインストールすることもできますし、パッケージをインストールせずにプログラムを使用することもできます。また、手動でビルドすることで、SSE 4.2 の要件を無効にしたり、AArch64 CPU 用にビルドしたりすることもできます。

```
Client: programs/clickhouse-client  
Server: programs/clickhouse-server
```

ユーザのために、データとメタデータのフォルダを作成して `chown` する必要があります。それらのパスはサーバ設定 (`src/programs/server/config.xml`) で変更することができます。デフォルトは以下です:

```
/opt/clickhouse/data/default/  
/opt/clickhouse/metadata/default/
```

Gentooでは、ソースから ClickHouseをインストールするために `emerge clickhouse` を使うことができます。

## 起動

サーバをデーモンとして起動するには:

```
$ sudo service clickhouse-server start
```

`service` コマンドがない場合は以下のように実行します:

```
$ sudo /etc/init.d/clickhouse-server start
```

`/var/log/clickhouse-server/` ディレクトリのログを参照してください。

サーバが起動しない場合は、`/etc/clickhouse-server/config.xml` ファイル内の設定を確認してください。

同様に、コンソールから以下のように手動で起動することができます:

```
$ clickhouse-server --config-file=/etc/clickhouse-server/config.xml
```

この場合、コンソールに開発時に便利なログが出力されます。設定ファイルがカレントディレクトリにある場合は、`--config-file` パラメータを指定する必要はありません。デフォルトでは `./config.xml` を使用します。

ClickHouseはアクセス制限の設定をサポートしています。それらは `users.xml` ファイル (`config.xml` の隣) にあります。デフォルトでは、`default` ユーザは、パスワードなしでどこからでもアクセスが許可されます。`user/default/networks` を参照し、詳細について、「[設定ファイル](#)」の項を参照してください。

サーバを起動した後、コマンドラインクライアントを使用してサーバに接続することができます:

```
$ clickhouse-client
```

デフォルトでは、ユーザ `default` で `localhost:9000` にパスワードなしで接続します。また、`--host` 引数を使ってリモートサーバに接続することもできます。

端末はUTF-8エンコーディングを使用する必要があります。詳細については、「[コマンドラインクライアント](#)」を参照してください。

例:

```
$ ./clickhouse-client
ClickHouse client version 0.0.18749.
Connecting to localhost:9000.
Connected to ClickHouse server version 0.0.18749.
```

```
:) SELECT 1
```

```
SELECT 1
```

```
[1]
```

```
1 rows in set. Elapsed: 0.003 sec.
```

```
:)
```

おめでとうございます！システムが動きました！

動作確認を続けるには、テストデータセットをダウンロードするか、[チュートリアル](#)を参照してください。

## ClickHouse チュートリアル

### このチュートリアルに期待されることとは？

このチュートリアルでは、単純なClickHouseクラスターを設定する方法について説明します。それは小規模ですが、耐障害性とスケーラブルになります。

次に、データセット例のいずれかを使用してデータを入力し、いくつかのデモクエリを実行します。

### 單一ノードの設定

分散環境は複雑なので、まずは単一のサーバーまたは仮想マシンにClickHouseを展開することから始めます。

ClickHouseは通常、[deb or rpm](#) パッケージからインストールしますが、サポートされていないOSのために[代替案](#)があります。

例えば、[deb](#) パッケージを選択して実行する場合:

```
sudo apt-get install apt-transport-https ca-certificates dirmngr
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv E0C56BD4

echo "deb https://repo.clickhouse.com/deb/stable/ main/" | sudo tee \
/etc/apt/sources.list.d/clickhouse.list
sudo apt-get update

sudo apt-get install -y clickhouse-server clickhouse-client

sudo service clickhouse-server start
clickhouse-client
```

インストールされたパッケージの内容:

- `clickhouse-client` パッケージには `clickhouse-client` アプリケーション、インタラクティブな ClickHouse コンソール クライアントが含まれています。
- `clickhouse-common` パッケージには、ClickHouse実行可能ファイルが含まれています。
- `clickhouse-server` パッケージには、ClickHouseをサーバとして実行するための設定ファイルが含まれています。

サーバ設定ファイルは `/etc/clickhouse-server/` にあります。先に進む前に、`config.xml` の `<path>` 要素に注目してください。

パスはデータを保存する場所を決めるので、ディスク容量の大きいボリュームに配置する必要があります。デフォルト値は `/var/lib/clickhouse/` です。

設定を調整したい場合、`config.xml` ファイルを直接編集するのは不便です。`config.xml` の要素をオーバーライドするために推奨される方法は、`config.xml` の「パッチ」として機能する `config.d` ディレクトリ内のファイルを作成することです。

お気づきかも知れませんが、`clickhouse-server` は パッケージインストール後に自動的に起動されたり、アップデート後に自動的に再起動されることはありません。サーバの起動方法は `init` システムに依存しており、大抵は以下のようになっています:

```
sudo service clickhouse-server start
```

または

```
sudo /etc/init.d/clickhouse-server start
```

サーバログのデフォルトの場所は `/var/log/clickhouse-server/` です。サーバが クライアントからの接続の準備が整うと、`Ready for connections` メッセージをログに出力します。

一度 `clickhouse-server` を起動して実行すると、`clickhouse-client` を使ってサーバに接続し、`SELECT "Hello, world!";` のようなテストクエリを実行することができます。

### ▶ Clickhouse-クライアントのクイックtips

## サンプルデータセットのインポート

ClickHouseサーバーにいくつかのサンプルデータを入れてみましょう。

このチュートリアルでは、ClickHouseがオープンソースになる以前(詳しくは [歴史について](#) を参照)に初めて本番用途で使われたサービスである、Yandex.Metricaの匿名化されたデータを使用します。

[Yandex.Metrica データセットをインポートするにはいくつかの方法](#) がありますが、チュートリアルとして、最もよく使う方法を使用します。

## テーブルデータのダウンロードと展開

```
curl https://datasets.clickhouse.com/hits/tsv/hits_v1.tsv.xz | unxz --threads=`nproc` > hits_v1.tsv
curl https://datasets.clickhouse.com/visits/tsv/visits_v1.tsv.xz | unxz --threads=`nproc` > visits_v1.tsv
```

展開されたファイルのサイズは約10GBです。

## テーブルの作成

ほとんどのデータベース管理システムのように、ClickHouseは論理的にテーブルを "データベース" にグループ化します。

`default` データベースがありますが、`tutorial` という名前の新しいものを作成します:

```
clickhouse-client --query "CREATE DATABASE IF NOT EXISTS tutorial"
```

テーブルを作成するための構文はデータベースに比べてはるかに複雑です(参照)。通常、`CREATE TABLE` 文は3つのキーを指定しなければなりません。

1. 作成するテーブルの名前。
2. テーブルのスキーマ。つまり、カラムと **データ型** のリスト。
3. **テーブルエンジン** と、このテーブルへのクエリが物理的に実行される方法に関するすべての詳細を決定する設定。

Yandex.Metricaはウェブ解析サービスであり、サンプルデータセットでは機能が網羅されていないため、作成するテーブルは2つしかありません:

- `hits` はサービスの対象となるすべてのウェブサイト上ですべてのユーザが行った各アクションのテーブルです。
- `visits` は、個々のアクションではなく、あらかじめ構築されたセッションを含むテーブルです。

これらのテーブルの実際の `CREATE TABLE` クエリを実行しましょう:

```
CREATE TABLE tutorial.hits_v1
(
    `WatchID` UInt64,
    `JavaEnable` UInt8,
    `Title` String,
    `GoodEvent` Int16,
    `EventTime` DateTime,
    `EventDate` Date,
    `CounterID` UInt32,
    `ClientIP` UInt32,
    `ClientIP6` FixedString(16),
    `RegionID` UInt32,
    `UserID` UInt64,
    `CounterClass` Int8,
    `OS` UInt8,
    `UserAgent` UInt8,
    `URL` String,
    `Referer` String,
    `URLDomain` String,
    `RefererDomain` String,
    `Refresh` UInt8,
    `IsRobot` UInt8,
    `RefererCategories` Array(UInt16),
    `URLCategories` Array(UInt16),
    `URLRegions` Array(UInt32),
    `RefererRegions` Array(UInt32),
    `ResolutionWidth` UInt16,
    `ResolutionHeight` UInt16,
    `ResolutionDepth` UInt8,
    `FlashMajor` UInt8,
    `FlashMinor` UInt8,
    `FlashMinor2` String,
    `NetMajor` UInt8,
    `NetMinor` UInt8,
    `UserAgentMajor` UInt16,
    `UserAgentMinor` FixedString(2),
    `CookieEnable` UInt8,
    `JavascriptEnable` UInt8,
    `IsMobile` UInt8,
    `MobilePhone` UInt8,
    `MobilePhoneModel` String,
    `Params` String,
    `IPNetworkID` UInt32,
    `TraficSourceID` Int8,
    `SearchEngineID` UInt16,
    `SearchPhrase` String,
    `AdvEngineID` UInt8,
```

```
`IsArtifical` UInt8,
`WindowClientWidth` UInt16,
`WindowClientHeight` UInt16,
`ClientTimeZone` Int16,
`ClientEventTime` DateTime,
`SilverlightVersion1` UInt8,
`SilverlightVersion2` UInt8,
`SilverlightVersion3` UInt32,
`SilverlightVersion4` UInt16,
`PageCharset` String,
`CodeVersion` UInt32,
`IsLink` UInt8,
`IsDownload` UInt8,
`IsNotBounce` UInt8,
`FUniqID` UInt64,
`HID` UInt32,
`IsOldCounter` UInt8,
`IsEvent` UInt8,
`IsParameter` UInt8,
`DontCountHits` UInt8,
`WithHash` UInt8,
`HitColor` FixedString(1),
`UTCEventTime` DateTime,
`Age` UInt8,
`Sex` UInt8,
`Income` UInt8,
`Interests` UInt16,
`Robotness` UInt8,
`GeneralInterests` Array(UInt16),
`RemoteIP` UInt32,
`RemoteIP6` FixedString(16),
`WindowName` Int32,
`OpenerName` Int32,
`HistoryLength` Int16,
`BrowserLanguage` FixedString(2),
`BrowserCountry` FixedString(2),
`SocialNetwork` String,
`SocialAction` String,
`HTTPError` UInt16,
`SendTiming` Int32,
`DNSTiming` Int32,
`ConnectTiming` Int32,
`ResponseStartTiming` Int32,
`ResponseEndTiming` Int32,
`FetchTiming` Int32,
`RedirectTiming` Int32,
`DOMInteractiveTiming` Int32,
`DOMContentLoadedTiming` Int32,
`DOMCompleteTiming` Int32,
`LoadEventStartTiming` Int32,
`LoadEventEndTiming` Int32,
`NSToDOMContentLoadedTiming` Int32,
`FirstPaintTiming` Int32,
`RedirectCount` Int8,
`SocialSourceNetworkID` UInt8,
`SocialSourcePage` String,
`ParamPrice` Int64,
`ParamOrderID` String,
`ParamCurrency` FixedString(3),
`ParamCurrencyID` UInt16,
`GoalsReached` Array(UInt32),
`OpenstatServiceName` String,
`OpenstatCampaignID` String,
`OpenstatAdID` String,
`OpenstatSourceID` String,
`UTMSource` String,
`UTMMedium` String,
`UTMCampaign` String,
`UTMContent` String,
`UTMTerm` String,
`FromTag` String,
`HasGCLID` UInt8,
`RefererHash` UInt64,
`URLHash` UInt64,
`CLID` UInt32,
`YCLID` UInt64,
`ShareService` String,
```

```

`ShareURL` String,
`ShareTitle` String,
`ParsedParams` Nested(
    Key1 String,
    Key2 String,
    Key3 String,
    Key4 String,
    Key5 String,
    ValueDouble Float64),
`IslandID` FixedString(16),
`RequestNum` UInt32,
`RequestTry` UInt8
)
ENGINE = MergeTree()
PARTITION BY toYYYYMM(EventDate)
ORDER BY (CounterID, EventDate, intHash32(UserID))
SAMPLE BY intHash32(UserID)

```

```

CREATE TABLE tutorial.visits_v1
(
    `CounterID` UInt32,
    `StartDate` Date,
    `Sign` Int8,
    `IsNew` UInt8,
    `VisitID` UInt64,
    `UserID` UInt64,
    `StartTime` DateTime,
    `Duration` UInt32,
    `UTCStartTime` DateTime,
    `PageViews` Int32,
    `Hits` Int32,
    `IsBounce` UInt8,
    `Referer` String,
    `StartURL` String,
    `RefererDomain` String,
    `StartURLDomain` String,
    `EndURL` String,
    `LinkURL` String,
    `IsDownload` UInt8,
    `TraficSourceID` Int8,
    `SearchEngineID` UInt16,
    `SearchPhrase` String,
    `AdvEngineID` UInt8,
    `PlaceID` Int32,
    `RefererCategories` Array(UInt16),
    `URLCategories` Array(UInt16),
    `URLRegions` Array(UInt32),
    `RefererRegions` Array(UInt32),
    `IsYandex` UInt8,
    `GoalReachesDepth` Int32,
    `GoalReachesURL` Int32,
    `GoalReachesAny` Int32,
    `SocialSourceNetworkID` UInt8,
    `SocialSourcePage` String,
    `MobilePhoneModel` String,
    `ClientEventTime` DateTime,
    `RegionID` UInt32,
    `ClientIP` UInt32,
    `ClientIP6` FixedString(16),
    `RemoteIP` UInt32,
    `RemoteIP6` FixedString(16),
    `IPNetworkID` UInt32,
    `SilverlightVersion3` UInt32,
    `CodeVersion` UInt32,
    `ResolutionWidth` UInt16,
    `ResolutionHeight` UInt16,
    `UserAgentMajor` UInt16,
    `UserAgentMinor` UInt16,
    `WindowClientWidth` UInt16,
    `WindowClientHeight` UInt16,
    `SilverlightVersion2` UInt8,
    `SilverlightVersion4` UInt16,
    `FlashVersion3` UInt16,
    `FlashVersion4` UInt16,
    `ClientTimeZone` Int16,

```

```
`OS` UInt8,
`UserAgent` UInt8,
`ResolutionDepth` UInt8,
`FlashMajor` UInt8,
`FlashMinor` UInt8,
`NetMajor` UInt8,
`NetMinor` UInt8,
`MobilePhone` UInt8,
`SilverlightVersion1` UInt8,
`Age` UInt8,
`Sex` UInt8,
`Income` UInt8,
`JavaEnable` UInt8,
`CookieEnable` UInt8,
`JavascriptEnable` UInt8,
`IsMobile` UInt8,
`BrowserLanguage` UInt16,
`BrowserCountry` UInt16,
`Interests` UInt16,
`Robotness` UInt8,
`GeneralInterests` Array(UInt16),
`Params` Array(String),
`Goals` Nested(
    ID UInt32,
    Serial UInt32,
    EventTime DateTime,
    Price Int64,
    OrderID String,
    CurrencyID UInt32),
`WatchIDs` Array(UInt64),
`ParamSumPrice` Int64,
`ParamCurrency` FixedString(3),
`ParamCurrencyID` UInt16,
`ClickLogID` UInt64,
`ClickEventID` Int32,
`ClickGoodEvent` Int32,
`ClickEventTime` DateTime,
`ClickPriorityID` Int32,
`ClickPhraseID` Int32,
`ClickPageID` Int32,
`ClickPlaceID` Int32,
`ClickTypeID` Int32,
`ClickResourceID` Int32,
`ClickCost` UInt32,
`ClickClientIP` UInt32,
`ClickDomainID` UInt32,
`ClickURL` String,
`ClickAttempt` UInt8,
`ClickOrderID` UInt32,
`ClickBannerID` UInt32,
`ClickMarketCategoryID` UInt32,
`ClickMarketPP` UInt32,
`ClickMarketCategoryName` String,
`ClickMarketPPName` String,
`ClickAWAPSCampaignName` String,
`ClickPageName` String,
`ClickTargetType` UInt16,
`ClickTargetPhraseID` UInt64,
`ClickContextType` UInt8,
`ClickSelectType` Int8,
`ClickOptions` String,
`ClickGroupBannerID` Int32,
`OpenstatServiceName` String,
`OpenstatCampaignID` String,
`OpenstatAdID` String,
`OpenstatSourceID` String,
`UTMSource` String,
`UTMMedium` String,
`UTMCampaign` String,
`UTMContent` String,
`UTMTerm` String,
`FromTag` String,
`HasGCLID` UInt8,
`FirstVisit` DateTime,
`PredLastVisit` Date,
`LastVisit` Date,
`TotalVisits` UInt32,
```

```

`TrafficSource` Nested(
    ID UInt8,
    SearchEngineID UInt16,
    AdvEngineID UInt8,
    PlaceID UInt16,
    SocialSourceNetworkID UInt8,
    Domain String,
    SearchPhrase String,
    SocialSourcePage String),
`Attendance` FixedString(16),
`CLID` UInt32,
`YCLID` UInt64,
`NormalizedRefererHash` UInt64,
`SearchPhraseHash` UInt64,
`RefererDomainHash` UInt64,
`NormalizedStartURLHash` UInt64,
`StartURLDomainHash` UInt64,
`NormalizedEndURLHash` UInt64,
`TopLevelDomain` UInt64,
`URLScheme` UInt64,
`OpenstatServiceNameHash` UInt64,
`OpenstatCampaignIDHash` UInt64,
`OpenstatAdIDHash` UInt64,
`OpenstatSourceIDHash` UInt64,
`UTMSourceHash` UInt64,
`UTMMediumHash` UInt64,
`UTMCampaignHash` UInt64,
`UTMContentHash` UInt64,
`UTMTermHash` UInt64,
`FromHash` UInt64,
`WebVisorEnabled` UInt8,
`WebVisorActivity` UInt32,
`ParsedParams` Nested(
    Key1 String,
    Key2 String,
    Key3 String,
    Key4 String,
    Key5 String,
    ValueDouble Float64),
`Market` Nested(
    Type UInt8,
    GoalID UInt32,
    OrderID String,
    OrderPrice Int64,
    PP UInt32,
    DirectPlaceID UInt32,
    DirectOrderID UInt32,
    DirectBannerID UInt32,
    GoodID String,
    GoodName String,
    GoodQuantity Int32,
    GoodPrice Int64),
`IslandID` FixedString(16)
)
ENGINE = CollapsingMergeTree(Sign)
PARTITION BY toYYYYMM(StartDate)
ORDER BY (CounterID, StartDate, intHash32(UserID), VisitID)
SAMPLE BY intHash32(UserID)

```

これらのクエリは、clickhouse-client の対話型モード(事前にクエリを指定せずにターミナルで起動するだけです)を使って実行するか、代替インターフェイス で実行できます。

見ての通り、`hits_v1` は 基本的な MergeTree エンジン を使っており、`visits_v1` は Collapsing MergeTree という変種を使っています。

## データのインポート

ClickHouseへのデータのインポートは、他の多くのSQLデータベースのように`INSERT INTO`クエリを介して行われます。しかし、データは通常、`VALUES` 句 (これもサポートされています) の代わりに、サポートされているシリアル化形式 のいずれかで提供されます。

先ほどダウンロードしたファイルはタブ区切り形式になっており、コンソールクライアントから次のようにインポートします:

```
clickhouse-client --query "INSERT INTO tutorial.hits_v1 FORMAT TSV" --max_insert_block_size=100000 < hits_v1.tsv
clickhouse-client --query "INSERT INTO tutorial.visits_v1 FORMAT TSV" --max_insert_block_size=100000 < visits_v1.tsv
```

ClickHouseには多くの [調整のための設定](#) があり、コンソールクライアントで引数として指定する方法があります。どのような設定が利用可能で、それが何を意味し、デフォルトは何なのかを知る最も簡単な方法は、`system.settings` テーブルにクエリすることです:

```
SELECT name, value, changed, description
FROM system.settings
WHERE name LIKE '%max_insert_b%'
FORMAT TSV

max_insert_block_size 1048576 0 "The maximum block size for insertion, if we control the creation of blocks for
insertion."
```

必要に応じて、インポート後のテーブルを [OPTIMIZE](#)することができます。

MergeTree-familyのエンジンで設定されているテーブルは、常にバックグラウンドでデータ部分のマージを行い、データストレージを最適化します(あるいは、少なくともそれが意味のあるものかどうかをチェックします)。

以下のクエリは、テーブルエンジンがストレージの最適化を後で行うのではなく、今すぐに行うように強制します:

```
clickhouse-client --query "OPTIMIZE TABLE tutorial.hits_v1 FINAL"
clickhouse-client --query "OPTIMIZE TABLE tutorial.visits_v1 FINAL"
```

これらのクエリは、I/OとCPUに負荷のかかる処理を開始するので、テーブルが継続して新しいデータを受け取る場合には、そのままにしてバックグラウンドでマージを実行させた方が良いでしょう。

これで、テーブルのインポートが成功したかどうかを確認することができます:

```
clickhouse-client --query "SELECT COUNT(*) FROM tutorial.hits_v1"
clickhouse-client --query "SELECT COUNT(*) FROM tutorial.visits_v1"
```

## クエリの例

```
SELECT
    StartURL AS URL,
    AVG(Duration) AS AvgDuration
FROM tutorial.visits_v1
WHERE StartDate BETWEEN '2014-03-23' AND '2014-03-30'
GROUP BY URL
ORDER BY AvgDuration DESC
LIMIT 10
```

```
SELECT
    sum(Sign) AS visits,
    sumIf(Sign, has(Goals.ID, 1105530)) AS goal_visits,
    (100. * goal_visits) / visits AS goal_percent
FROM tutorial.visits_v1
WHERE (CounterID = 912887) AND (toYYYYMM(StartDate) = 201403) AND (domain(StartURL) = 'yandex.ru')
```

## クラスタのデプロイ

ClickHouseクラスタは均質なクラスタ(homogenous cluster)です。セットアップ手順は以下です:

1. クラスタのすべてのマシンにClickHouse serverをインストールする
2. 設定ファイルにクラスタ設定を行う
3. 各インスタンスにローカルテーブルを作成する
4. 分散テーブルを作成する

分散テーブルは、

実際には ClickHouse クラスタのローカルテーブルへの "ビュー" のようなものです。

分散テーブルからの SELECT クエリは、すべてのクラスタのシャードのリソースを使用して実行されます。

また、複数のクラスタのための設定を行い、異なるクラスタにビューを提供する複数の分散テーブルを作成することができます。

3つのシャード、各1つのレプリカを持つクラスタの設定例:

```
<remote_servers>
  <perftest_3shards_1replicas>
    <shard>
      <replica>
        <host>example-perftest01j.yandex.ru</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <replica>
        <host>example-perftest02j.yandex.ru</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <replica>
        <host>example-perftest03j.yandex.ru</host>
        <port>9000</port>
      </replica>
    </shard>
  </perftest_3shards_1replicas>
</remote_servers>
```

さらにデモンストレーションとして、hits\_v1 で使ったのと同じ CREATE TABLE クエリを使って新しいローカルテーブルを作成してみましょう:

```
CREATE TABLE tutorial.hits_local (...) ENGINE = MergeTree() ...
```

クラスタのローカルテーブルへのビューを提供する分散テーブルを作成します:

```
CREATE TABLE tutorial.hits_all AS tutorial.hits_local
ENGINE = Distributed(perftest_3shards_1replicas, tutorial, hits_local, rand());
```

通常は、クラスタのすべてのマシンに同様の分散テーブルを作成します。これにより、クラスタのどのマシンでも分散クエリを実行することができます。

また、remote テーブル関数を使用して、与えられたSELECTクエリのための一時的な分散テーブルを作成する代替オプションもあります。

分散テーブルに INSERT SELECT を実行して、テーブルを複数のサーバに分散させてみましょう。

```
INSERT INTO tutorial.hits_all SELECT * FROM tutorial.hits_v1;
```

## 注意

この方法は、大きなテーブルのシャーディングには適していません。

別のツール **clickhouse-copier** があり、任意の大きなテーブルを再シャーディングすることができます。

予想通り、計算量の多いクエリは、1台のサーバではなく3台のサーバを利用した方がN倍速く実行されます。

このケースでは、3つのシャードを持つクラスタを使用しており、それぞれに1つのレプリカが含まれています。

本番環境でレジリエンスを提供するためには、各シャードが複数のアベイラビリティゾーンまたはデータセンター（または少なくともラック）の間で2~3個のレプリカを含むことをお勧めします。

3つのレプリカを含む1つのシャードのクラスタの設定例：

```
<remote_servers>
...
<perftest_1shards_3replicas>
  <shard>
    <replica>
      <host>example-perftest01j.yandex.ru</host>
      <port>9000</port>
    </replica>
    <replica>
      <host>example-perftest02j.yandex.ru</host>
      <port>9000</port>
    </replica>
    <replica>
      <host>example-perftest03j.yandex.ru</host>
      <port>9000</port>
    </replica>
  </shard>
</perftest_1shards_3replicas>
</remote_servers>
```

ネイティブレプリケーションを有効にするには [ZooKeeper](#) が必要です。

ClickHouseは全てのレプリカ上でデータの整合性を取り、障害発生時には自動的にリストア処理を行います。

ZooKeeperクラスターは別サーバ(ClickHouseを含む他のプロセスが稼働していない場所)に配置することをお勧めします。

ネイティブ複製を有効にする [飼育係](#) 必須です。ClickHouseのデータの整合性はすべてのレプリカと回復手続き後の不動します。別のサーバー(ClickHouseを含む他のプロセスが実行されていない場所)にZooKeeperクラスターを開くことをお勧めします。

## 備考

ZooKeeperは厳密な要件ではありません：いくつかの簡単なケースでは、アプリケーションコードからすべてのレプリカにデータを書き込むことでデータを複製することができます。このケースにおいて、このアプローチは [お勧めできません](#)。ClickHouseはすべてのレプリカ上でデータの一貫性を保証することができません。したがって、それはあなたのアプリケーションの責任になります。

ZooKeeperの場所は設定ファイルで指定します：

```
<zookeeper>
  <node>
    <host>zoo01.yandex.ru</host>
    <port>2181</port>
  </node>
  <node>
    <host>zoo02.yandex.ru</host>
    <port>2181</port>
  </node>
  <node>
    <host>zoo03.yandex.ru</host>
    <port>2181</port>
  </node>
</zookeeper>
```

また、テーブル作成時に使用する各シャードとレプリカを識別するためのマクロを設定する必要があります:

```
<macros>
  <shard>01</shard>
  <replica>01</replica>
</macros>
```

レプリケートされたテーブルの作成時にレプリカが存在しない場合、新しい最初のレプリカがインスタンス化されます。既に有効なレプリカがある場合は、新しいレプリカが既存のレプリカからデータをクローンします。最初にすべての複製されたテーブルを作成し、そこにデータを挿入する方法があります。別の方法として、いくつかのレプリカを作成して、データ挿入後またはデータ挿入中に他のレプリカを追加するという方法もあります。

```
CREATE TABLE tutorial.hits_replica (...)  
ENGINE = ReplicatedMergeTree(  
  '/clickhouse_perftest/tables/{shard}/hits',  
  '{replica}'  
)  
...
```

ここでは、ReplicatedMergeTree テーブルエンジンを使用しています。  
パラメータには、シャードとレプリカの識別子を含む ZooKeeper path を指定します。

```
INSERT INTO tutorial.hits_replica SELECT * FROM tutorial.hits_local;
```

レプリケーションはマルチマスター mode で動作します。  
どのレプリカにもデータをロードすることができ、システムはそれを他のインスタンスと自動的に同期させます。  
レプリケーションは非同期なので、ある時点ですべてのレプリカに最近挿入されたデータが含まれているとは限りません。  
少なくとも1つのレプリカは、データの取り込みを可能にするために起動しておく必要があります。  
他のレプリカはデータを同期させ、再びアクティブになると整合性を修復します。  
この方法では、最近挿入されたデータが失われる可能性が低いことに注意してください。

## GitHub Events Dataset

Dataset contains all events on GitHub from 2011 to Dec 6 2020, the size is 3.1 billion records. Download size is 75 GB and it will require up to 200 GB space on disk if stored in a table with lz4 compression.

Full dataset description, insights, download instruction and interactive queries are posted [here](#).

## データセット例

このセクションでは、データセット例を取得し、ClickHouseにそれをインポートする方法について説明します。いくつかのデータセット例では、クエリも利用可能です。

- 匿名Yandexの。メトリカデータセット
- Star Schemaベンチマーク
- ウィキスタッフ
- Criteoからのクリックログのテラバイト
- AMPLab Big Dataベンチマーク
- ニューヨークタクシ
- オンタイム

## 匿名化された Yandex.Metrica データ

データセットは、Yandex.Metricaのヒット数(hits\_v1)と訪問数(visits\_v1)に関する匿名化されたデータを含む2つのテーブルから構成されています。

Yandex.Metricaについての詳細は [ClickHouse history](#) のセクションを参照してください。

データセットは2つのテーブルから構成されており、どちらも圧縮された tsv.xz ファイルまたは準備されたパーティションとしてダウンロードすることができます。

さらに、1億行を含むhitsテーブルの拡張版が TSVとして

[https://datasets.clickhouse.com/hits/tsv/hits\\_100m\\_obfuscated\\_v1.tsv.xz](https://datasets.clickhouse.com/hits/tsv/hits_100m_obfuscated_v1.tsv.xz) に、準備されたパーティションとして [https://datasets.clickhouse.com/hits/partitions/hits\\_100m\\_obfuscated\\_v1.tar.xz](https://datasets.clickhouse.com/hits/partitions/hits_100m_obfuscated_v1.tar.xz) にあります。

### パーティション済みテーブルの取得

hits テーブルのダウンロードとインポート:

```
curl -O https://datasets.clickhouse.com/hits/partitions/hits_v1.tar  
tar xvf hits_v1.tar -C /var/lib/clickhouse # ClickHouse のデータディレクトリへのパス  
## 展開されたデータのパーティションをチェックし、必要に応じて修正します。  
sudo service clickhouse-server restart  
clickhouse-client --query "SELECT COUNT(*) FROM datasets.hits_v1"
```

visits のダウンロードとインポート:

```
curl -O https://datasets.clickhouse.com/visits/partitions/visits_v1.tar  
tar xvf visits_v1.tar -C /var/lib/clickhouse # ClickHouse のデータディレクトリへのパス  
## 展開されたデータのパーティションをチェックし、必要に応じて修正します。  
sudo service clickhouse-server restart  
clickhouse-client --query "SELECT COUNT(*) FROM datasets.visits_v1"
```

### 圧縮されたTSVファイルからのテーブルの取得

圧縮TSVファイルのダウンロードと hits テーブルのインポート:

```

curl https://datasets.clickhouse.com/hits/tsv/hits_v1.tsv.xz | unxz --threads=`nproc` > hits_v1.tsv
## now create table
clickhouse-client --query "CREATE DATABASE IF NOT EXISTS datasets"
clickhouse-client --query "CREATE TABLE datasets.hits_v1 ( WatchID UInt64, JavaEnable UInt8, Title String,
GoodEvent Int16, EventTime DateTime, EventDate Date, CounterID UInt32, ClientIP UInt32, ClientIP6
FixedString(16), RegionID UInt32, UserID UInt64, CounterClass Int8, OS UInt8, UserAgent UInt8, URL String,
Referer String, URLDomain String, RefererDomain String, Refresh UInt8, IsRobot UInt8, RefererCategories
Array(UInt16), URLCategories Array(UInt16), URLRegions Array(UInt32), RefererRegions Array(UInt32),
ResolutionWidth UInt16, ResolutionHeight UInt16, ResolutionDepth UInt8, FlashMajor UInt8, FlashMinor UInt8,
FlashMinor2 String, NetMajor UInt8, NetMinor UInt8, UserAgentMajor UInt16, UserAgentMinor FixedString(2),
CookieEnable UInt8, JavascriptEnable UInt8, IsMobile UInt8, MobilePhone UInt8, MobilePhoneModel String, Params
String, IPNetworkID UInt32, TraficSourceID UInt8, SearchEngineID UInt16, SearchPhrase String, AdvEngineID UInt8,
IsArtifical UInt8, WindowClientWidth UInt16, WindowClientHeight UInt16, ClientTimeZone Int16, ClientEventTime
DateTime, SilverlightVersion1 UInt8, SilverlightVersion2 UInt8, SilverlightVersion3 UInt32, SilverlightVersion4
UInt16, PageCharset String, CodeVersion UInt32, IsLink UInt8, IsDownload UInt8, IsNotBounce UInt8, FUniqID
UInt64, HID UInt32, IsOldCounter UInt8, IsEvent UInt8, IsParameter UInt8, DontCountHits UInt8, WithHash UInt8,
HitColor FixedString(1), UTCEventTime DateTime, Age UInt8, Sex UInt8, Income UInt8, Interests UInt16, Robotness
UInt8, GeneralInterests Array(UInt16), RemoteIP UInt32, RemoteIP6 FixedString(16), WindowName Int32,
OpenerName Int32, HistoryLength Int16, BrowserLanguage FixedString(2), BrowserCountry FixedString(2),
SocialNetwork String, SocialAction String, HTTPError UInt16, SendTiming Int32, DNSTiming Int32, ConnectTiming
Int32, ResponseStartTiming Int32, ResponseEndTiming Int32, FetchTiming Int32, RedirectTiming Int32,
DOMInteractiveTiming Int32, DOMContentLoadedTiming Int32, DOMCompleteTiming Int32, LoadEventStartTiming
Int32, LoadEventEndTiming Int32, NSToDOMContentLoadedTiming Int32, FirstPaintTiming Int32, RedirectCount Int8,
SocialSourceNetworkID UInt8, SocialSourcePage String, ParamPrice Int64, ParamOrderID String, ParamCurrency
FixedString(3), ParamCurrencyID UInt16, GoalsReached Array(UInt32), OpenstatServiceName String,
OpenstatCampaignID String, OpenstatAdID String, OpenstatSourceID String, UTMSource String, UTMMedium String,
UTMCampaign String, UTMContent String, UTMTerm String, FromTag String, HasGCLID UInt8, RefererHash UInt64,
URLHash UInt64, CLID UInt32, YCLID UInt64, ShareService String, ShareURL String, ShareTitle String,
ParsedParams Nested(Key1 String, Key2 String, Key3 String, Key4 String, Key5 String, ValueDouble Float64),
IslandID FixedString(16), RequestNum UInt32, RequestTry UInt8) ENGINE = MergeTree() PARTITION BY
toYYYYMM(EventDate) ORDER BY (CounterID, EventDate, intHash32(UserID)) SAMPLE BY intHash32(UserID)
SETTINGS index_granularity = 8192"
## import data
cat hits_v1.tsv | clickhouse-client --query "INSERT INTO datasets.hits_v1 FORMAT TSV" --
max_insert_block_size=100000
## optionally you can optimize table
clickhouse-client --query "OPTIMIZE TABLE datasets.hits_v1 FINAL"
clickhouse-client --query "SELECT COUNT(*) FROM datasets.hits_v1"

```

圧縮TSVファイルのダウンロードと visits テーブルのインポート：

```

curl https://datasets.clickhouse.com/visits/tsv/visits_v1.tsv.xz | unxz --threads=`nproc` > visits_v1.tsv
## now create table
clickhouse-client --query "CREATE DATABASE IF NOT EXISTS datasets"
clickhouse-client --query "CREATE TABLE datasets.visits_v1 ( CounterID UInt32, StartDate Date, Sign Int8, IsNew
UInt8, VisitID UInt64, UserID UInt64, StartTime DateTime, Duration UInt32, UTCStartTime DateTime, PageViews
Int32, Hits Int32, IsBounce UInt8, Referer String, StartURL String, RefererDomain String, StartURLDomain String,
EndURL String, LinkURL String, IsDownload UInt8, TraficSourceID Int8, SearchEngineID UInt16, SearchPhrase
String, AdvEngineID UInt8, PlaceID Int32, RefererCategories Array(UInt16), URLCategories Array(UInt16),
URLRegions Array(UInt32), RefererRegions Array(UInt32), IsYandex UInt8, GoalReachesDepth Int32,
GoalReachesURL Int32, GoalReachesAny Int32, SocialSourceNetworkID UInt8, SocialSourcePage String,
MobilePhoneModel String, ClientEventTime DateTime, RegionID UInt32, ClientIP UInt32, ClientIP6 FixedString(16),
RemoteIP UInt32, RemoteP6 FixedString(16), IPNetworkID UInt32, SilverlightVersion3 UInt32, CodeVersion UInt32,
ResolutionWidth UInt16, ResolutionHeight UInt16, UserAgentMajor UInt16, UserAgentMinor UInt16,
WindowClientWidth UInt16, WindowClientHeight UInt16, SilverlightVersion2 UInt8, SilverlightVersion4 UInt16,
FlashVersion3 UInt16, FlashVersion4 UInt16, ClientTimeZone Int16, OS UInt8, UserAgent UInt8, ResolutionDepth
UInt8, FlashMajor UInt8, FlashMinor UInt8, NetMajor UInt8, NetMinor UInt8, MobilePhone UInt8, SilverlightVersion1
UInt8, Age UInt8, Sex UInt8, Income UInt8, JavaEnable UInt8, CookieEnable UInt8, JavascriptEnable UInt8, IsMobile
UInt8, BrowserLanguage UInt16, BrowerCountry UInt16, Interests UInt16, Robotness UInt8, GeneralInterests
Array(UInt16), Params Array(String), Goals Nested(ID UInt32, Serial UInt32, EventTime DateTime, Price Int64,
OrderID String, CurrencyID UInt32), WatchIDs Array(UInt64), ParamSumPrice Int64, ParamCurrency FixedString(3),
ParamCurrencyID UInt16, ClickLogID UInt64, ClickEventID Int32, ClickGoodEvent Int32, ClickEventTime DateTime,
ClickPriorityID Int32, ClickPhraseID Int32, ClickPageID Int32, ClickPlaceID Int32, ClickTypeID Int32, ClickResourceID
Int32, ClickCost UInt32, ClickClientIP UInt32, ClickDomainID UInt32, ClickURL String, ClickAttempt UInt8,
ClickOrderID UInt32, ClickBannerID UInt32, ClickMarketCategoryID UInt32, ClickMarketPP UInt32,
ClickMarketCategoryName String, ClickMarketPPName String, ClickAWAPSCampaignName String, ClickPageName
String, ClickTargetType UInt16, ClickTargetPhraseID UInt64, ClickContextType UInt8, ClickSelectType Int8,
ClickOptions String, ClickGroupBannerID Int32, OpenstatServiceName String, OpenstatCampaignID String,
OpenstatAdID String, OpenstatSourceID String, UTMSource String, UTMMedium String, UTMCampaign String,
UTMContent String, UTMTerm String, FromTag String, HasGCLID UInt8, FirstVisit DateTime, PredLastVisit Date,
LastVisit Date, TotalVisits UInt32, TraficSource Nested(ID Int8, SearchEngineID UInt16, AdvEngineID UInt8, PlaceID
UInt16, SocialSourceNetworkID UInt8, Domain String, SearchPhrase String, SocialSourcePage String), Attendance
FixedString(16), CLID UInt32, YCLID UInt64, NormalizedRefererHash UInt64, SearchPhraseHash UInt64,
RefererDomainHash UInt64, NormalizedStartURLHash UInt64, StartURLDomainHash UInt64, NormalizedEndURLHash
UInt64, TopLevelDomain UInt64, URLscheme UInt64, OpenstatServiceNameHash UInt64, OpenstatCampaignIDHash
UInt64, OpenstatAdIDHash UInt64, OpenstatSourceIDHash UInt64, UTMSourceHash UInt64, UTMMediumHash
UInt64, UTMCampaignHash UInt64, UTMContentHash UInt64, UTMTermHash UInt64, FromHash UInt64,
WebVisorEnabled UInt8, WebVisorActivity UInt32, ParsedParams Nested(Key1 String, Key2 String, Key3 String,
Key4 String, Key5 String, ValueDouble Float64), Market Nested(Type UInt8, GoalID UInt32, OrderID String,
OrderPrice Int64, PP UInt32, DirectPlaceID UInt32, DirectOrderID UInt32, DirectBannerID UInt32, GoodID String,
GoodName String, GoodQuantity Int32, GoodPrice Int64), IslandID FixedString(16)) ENGINE =
CollapsingMergeTree(Sign) PARTITION BY toYYYYMM(StartDate) ORDER BY (CounterID, StartDate, intHash32(UserID),
VisitID) SAMPLE BY intHash32(UserID) SETTINGS index_granularity = 8192"
## import data
cat visits_v1.tsv | clickhouse-client --query "INSERT INTO datasets.visits_v1 FORMAT TSV" --
max_insert_block_size=100000
## optionally you can optimize table
clickhouse-client --query "OPTIMIZE TABLE datasets.visits_v1 FINAL"
clickhouse-client --query "SELECT COUNT(*) FROM datasets.visits_v1"

```

## クエリの例

[ClickHouse tutorial](#) は Yandex.Metrica のデータセットに基づいているため、このチュートリアルを実施するのがおすすめです。

これらのテーブルに関する他のクエリ例は、ClickHouse の [stateful tests](#) で見つけることができます。(それらは `test.hirsts` と `test.visits` という名前です)

## OnTime

このデータセットは二つの方法で取得できます:

- 生データからインポート
- パーティション済みのダウンロード

## 生データからインポート

データのダウンロード:

```
for s in `seq 1987 2018`  
do  
for m in `seq 1 12`  
do  
wget  
https://transtats.bts.gov/PREZIP/On_Time_Reported_Carrier_On_Time_Performance_1987_present_${s}_${m}.zip  
done  
done
```

(<https://github.com/Percona-Lab/ontime-airline-performance/blob/master/download.sh> より)

テーブルの作成:

```
CREATE TABLE `ontime`  
(  
`Year`          UInt16,  
`Quarter`       UInt8,  
`Month`         UInt8,  
`DayOfMonth`    UInt8,  
`DayOfWeek`     UInt8,  
`FlightDate`    Date,  
`Reporting_Airline` String,  
`DOT_ID_Reported_Airline` Int32,  
`IATA_CODE_Reported_Airline` String,  
`Tail_Number`   Int32,  
`Flight_Number_Reported_Airline` String,  
`OriginAirportID` Int32,  
`OriginAirportSeqID` Int32,  
`OriginCityMarketID` Int32,  
`Origin`         FixedString(5),  
`OriginCityName` String,  
`OriginState`    FixedString(2),  
`OriginStateFips` String,  
`OriginStateName` String,  
`OriginWac`      Int32,  
`DestAirportID`  Int32,  
`DestAirportSeqID` Int32,  
`DestCityMarketID` Int32,  
`Dest`           FixedString(5),  
`DestCityName`   String,  
`DestState`      FixedString(2),  
`DestStateFips`  String,  
`DestStateName`  String,  
`DestWac`        Int32,  
`CRSDepTime`    Int32,  
`DepTime`        Int32,  
`DepDelay`       Int32,  
`DepDelayMinutes` Int32,  
`DepDel15`       Int32,  
`DepartureDelayGroups` String,  
`DepTimeBlk`     String,  
`TaxiOut`        Int32,  
`WheelsOff`      Int32,  
`WheelsOn`       Int32,  
`TaxiIn`         Int32,  
`CRSArrTime`    Int32,  
`ArrTime`        Int32,  
`ArrDelay`       Int32,  
`ArrDelayMinutes` Int32,  
`ArrDel15`       Int32,  
`ArrivalDelayGroups` Int32,  
`ArrTimeBlk`     String,  
`Cancelled`      UInt8,  
`CancellationCode` FixedString(1),  
`Diverted`       UInt8,  
`CRSElapsedTime` Int32,  
`ActualElapsedTime` Int32,  
`AirTime`        Nullable(Int32),  
`Flights`        Int32,  
`Distance`       Int32,  
`DistanceGroup`  UInt8,
```

```

`CarrierDelay`          Int32,
`WeatherDelay`         Int32,
`NASDelay`             Int32,
`SecurityDelay`        Int32,
`LateAircraftDelay`   Int32,
`FirstDepTime`         String,
`TotalAddGTime`        String,
`LongestAddGTime`      String,
`DivAirportLandings`   String,
`DivReachedDest`       String,
`DivActualElapsedTime` String,
`DivArrDelay`          String,
`DivDistance`          String,
`Div1Airport`           String,
`Div1AirportID`         Int32,
`Div1AirportSeqID`      Int32,
`Div1WheelsOn`          String,
`Div1TotalGTime`        String,
`Div1LongestGTime`      String,
`Div1WheelsOff`         String,
`Div1TailNum`           String,
`Div2Airport`            String,
`Div2AirportID`          Int32,
`Div2AirportSeqID`       Int32,
`Div2WheelsOn`           String,
`Div2TotalGTime`         String,
`Div2LongestGTime`       String,
`Div2WheelsOff`          String,
`Div2TailNum`            String,
`Div3Airport`             String,
`Div3AirportID`          Int32,
`Div3AirportSeqID`        Int32,
`Div3WheelsOn`            String,
`Div3TotalGTime`          String,
`Div3LongestGTime`        String,
`Div3WheelsOff`           String,
`Div3TailNum`             String,
`Div4Airport`              String,
`Div4AirportID`           Int32,
`Div4AirportSeqID`         Int32,
`Div4WheelsOn`            String,
`Div4TotalGTime`          String,
`Div4LongestGTime`        String,
`Div4WheelsOff`           String,
`Div4TailNum`              String,
`Div5Airport`                String,
`Div5AirportID`             Int32,
`Div5AirportSeqID`          Int32,
`Div5WheelsOn`              String,
`Div5TotalGTime`            String,
`Div5LongestGTime`          String,
`Div5WheelsOff`             String,
`Div5TailNum`               String
) ENGINE = MergeTree
    PARTITION BY Year
    ORDER BY (IATA_CODE_Reported_Airline, FlightDate)
    SETTINGS index_granularity = 8192;

```

データのロード:

```
ls -1 *.zip | xargs -I{} -P $(nproc) bash -c "echo {}; unzip -cq {} '*.csv' | sed 's/\.\.00//g' | clickhouse-client --input_format_with_names_use_header=0 --query='INSERT INTO ontime FORMAT CSVWithNames'"
```

## パーティション済みデータのダウンロード

```
$ curl -O https://datasets.clickhouse.com/ontime/partitions/ontime.tar
$ tar xvf ontime.tar -C /var/lib/clickhouse # path to ClickHouse data directory
$ # check permissions of unpacked data, fix if required
$ sudo service clickhouse-server restart
$ clickhouse-client --query "select count(*) from datasets.ontime"
```

## 情報

以下で説明するクエリを実行する場合は、`datasets.ontime` のような 完全なテーブル名を使用する必要があります。

## クエリ

Q0.

```
SELECT avg(c1)
FROM
(
  SELECT Year, Month, count(*) AS c1
  FROM ontime
  GROUP BY Year, Month
);
```

Q1. 2000年から2008年までの一日あたりのフライト数

```
SELECT DayOfWeek, count(*) AS c
FROM ontime
WHERE Year>=2000 AND Year<=2008
GROUP BY DayOfWeek
ORDER BY c DESC;
```

Q2. 2000年から2008年までの10分以上遅延したフライトの数を曜日ごとにグループ化

```
SELECT DayOfWeek, count(*) AS c
FROM ontime
WHERE DepDelay>10 AND Year>=2000 AND Year<=2008
GROUP BY DayOfWeek
ORDER BY c DESC;
```

Q3. 2000年から2008年までの空港別の遅延件数

```
SELECT Origin, count(*) AS c
FROM ontime
WHERE DepDelay>10 AND Year>=2000 AND Year<=2008
GROUP BY Origin
ORDER BY c DESC
LIMIT 10;
```

Q4. 2007年のキャリア別の遅延の数

```
SELECT IATA_CODE_Reported_Airline AS Carrier, count(*)
FROM ontime
WHERE DepDelay>10 AND Year=2007
GROUP BY IATA_CODE_Reported_Airline
ORDER BY count(*) DESC;
```

Q5. 2007年のキャリア別遅延の割合

```

SELECT Carrier, c, c2, c*100/c2 as c3
FROM
(
  SELECT
    IATA_CODE_Reported_Airline AS Carrier,
    count(*) AS c
  FROM ontime
  WHERE DepDelay>10
    AND Year=2007
  GROUP BY Carrier
) q
JOIN
(
  SELECT
    IATA_CODE_Reported_Airline AS Carrier,
    count(*) AS c2
  FROM ontime
  WHERE Year=2007
  GROUP BY Carrier
) qq USING Carrier
ORDER BY c3 DESC;

```

同じクエリのより良いバージョン：

```

SELECT IATA_CODE_Reported_Airline AS Carrier, avg(DepDelay>10)*100 AS c3
FROM ontime
WHERE Year=2007
GROUP BY IATA_CODE_Reported_Airline
ORDER BY c3 DESC

```

Q6. 前のリクエストを2000年から2008年までに広げたもの

```

SELECT Carrier, c, c2, c*100/c2 as c3
FROM
(
  SELECT
    IATA_CODE_Reported_Airline AS Carrier,
    count(*) AS c
  FROM ontime
  WHERE DepDelay>10
    AND Year>=2000 AND Year<=2008
  GROUP BY Carrier
) q
JOIN
(
  SELECT
    IATA_CODE_Reported_Airline AS Carrier,
    count(*) AS c2
  FROM ontime
  WHERE Year>=2000 AND Year<=2008
  GROUP BY Carrier
) qq USING Carrier
ORDER BY c3 DESC;

```

同じクエリのより良いバージョン：

```

SELECT IATA_CODE_Reported_Airline AS Carrier, avg(DepDelay>10)*100 AS c3
FROM ontime
WHERE Year>=2000 AND Year<=2008
GROUP BY Carrier
ORDER BY c3 DESC;

```

Q7. 年別の、10分以上遅延したフライトの割合

```

SELECT Year, c1/c2
FROM
(
  select
    Year,
    count(*)*100 as c1
  from ontime
  WHERE DepDelay>10
  GROUP BY Year
) q
JOIN
(
  select
    Year,
    count(*) as c2
  from ontime
  GROUP BY Year
) qq USING (Year)
ORDER BY Year;

```

同じクエリのより良いバージョン:

```

SELECT Year, avg(DepDelay>10)*100
FROM ontime
GROUP BY Year
ORDER BY Year;

```

Q8. 複数年の、直行都市数別の人気の高い目的地

```

SELECT DestCityName, uniqExact(OriginCityName) AS u
FROM ontime
WHERE Year >= 2000 and Year <= 2010
GROUP BY DestCityName
ORDER BY u DESC LIMIT 10;

```

Q9.

```

SELECT Year, count(*) AS c1
FROM ontime
GROUP BY Year;

```

Q10.

```

SELECT
  min(Year), max(Year), IATA_CODE_Reported_Airline AS Carrier, count(*) AS cnt,
  sum(ArrDelayMinutes>30) AS flights_delayed,
  round(sum(ArrDelayMinutes>30)/count(*),2) AS rate
FROM ontime
WHERE
  DayOfWeek NOT IN (6,7) AND OriginState NOT IN ('AK', 'HI', 'PR', 'VI')
  AND DestState NOT IN ('AK', 'HI', 'PR', 'VI')
  AND FlightDate < '2010-01-01'
GROUP by Carrier
HAVING cnt>100000 and max(Year)>1990
ORDER by rate DESC
LIMIT 1000;

```

ボーナス:

```

SELECT avg(cnt)
FROM
(
  SELECT Year,Month,count(*) AS cnt
  FROM ontime
  WHERE DepDel15=1
  GROUP BY Year,Month
);
SELECT avg(c1) FROM
(
  SELECT Year,Month,count(*) AS c1
  FROM ontime
  GROUP BY Year,Month
);
SELECT DestCityName, uniqExact(OriginCityName) AS u
FROM ontime
GROUP BY DestCityName
ORDER BY u DESC
LIMIT 10;
SELECT OriginCityName, DestCityName, count() AS c
FROM ontime
GROUP BY OriginCityName, DestCityName
ORDER BY c DESC
LIMIT 10;
SELECT OriginCityName, count() AS c
FROM ontime
GROUP BY OriginCityName
ORDER BY c DESC
LIMIT 10;

```

このパフォーマンステストは、Vadim Tkachenkoによって作成されました。以下を参照してください。

- <https://www.percona.com/blog/2009/10/02/analyzing-air-traffic-performance-with-infobright-and-monetdb/>
- <https://www.percona.com/blog/2009/10/26/air-traffic-queries-in-luciddb/>
- <https://www.percona.com/blog/2009/11/02/air-traffic-queries-in-infinidb-early-alpha/>
- <https://www.percona.com/blog/2014/04/21/using-apache-hadoop-and-impala-together-with-mysql-for-data-analysis/>
- <https://www.percona.com/blog/2016/01/07/apache-spark-with-air-ontime-performance-data/>
- <http://nickmakos.blogspot.ru/2012/08/analyzing-air-traffic-performance-with.html>

## Recipes Dataset

RecipeNLG dataset is available for download [here](#). It contains 2.2 million recipes. The size is slightly less than 1 GB.

### Download and Unpack the Dataset

1. Go to the download page <https://recipenlg.cs.put.poznan.pl/dataset>.
2. Accept Terms and Conditions and download zip file.
3. Unpack the zip file with `unzip`. You will get the `full_dataset.csv` file.

### Create a Table

Run `clickhouse-client` and execute the following CREATE query:

```

CREATE TABLE recipes
(
    title String,
    ingredients Array(String),
    directions Array(String),
    link String,
    source LowCardinality(String),
    NER Array(String)
) ENGINE = MergeTree ORDER BY title;

```

## Insert the Data

Run the following command:

```

clickhouse-client --query "
INSERT INTO recipes
SELECT
    title,
    JSONExtract(ingredients, 'Array(String)'),
    JSONExtract(directions, 'Array(String)'),
    link,
    source,
    JSONExtract(NER, 'Array(String)')
FROM input('num UInt32, title String, ingredients String, directions String, link String, source LowCardinality(String),
NER String')
FORMAT CSVWithNames
" --input_format_with_names_use_header 0 --format_csv_allow_single_quote 0 --input_format_allow_errors_num 10 <
full_dataset.csv"

```

This is a showcase how to parse custom CSV, as it requires multiple tunes.

Explanation:

- The dataset is in CSV format, but it requires some preprocessing on insertion; we use table function `input` to perform preprocessing;
- The structure of CSV file is specified in the argument of the table function `input`;
- The field `num` (row number) is unneeded - we parse it from file and ignore;
- We use `FORMAT CSVWithNames` but the header in CSV will be ignored (by command line parameter `--input_format_with_names_use_header 0`), because the header does not contain the name for the first field;
- File is using only double quotes to enclose CSV strings; some strings are not enclosed in double quotes, and single quote must not be parsed as the string enclosing - that's why we also add the `--format_csv_allow_single_quote 0` parameter;
- Some strings from CSV cannot parse, because they contain `\M` sequence at the beginning of the value; the only value starting with backslash in CSV can be `\N` that is parsed as SQL NULL. We add `--input_format_allow_errors_num 10` parameter and up to ten malformed records can be skipped;
- There are arrays for ingredients, directions and NER fields; these arrays are represented in unusual form: they are serialized into string as JSON and then placed in CSV - we parse them as String and then use `JSONExtract` function to transform it to Array.

## Validate the Inserted Data

By checking the row count:

Query:

```
SELECT count() FROM recipes;
```

Result:

```
count()  
2231141 |
```

## Example Queries

### Top Components by the Number of Recipes:

In this example we learn how to use **arrayJoin** function to expand an array into a set of rows.

Query:

```
SELECT  
    arrayJoin(NER) AS k,  
    count() AS c  
FROM recipes  
GROUP BY k  
ORDER BY c DESC  
LIMIT 50
```

Result:

k	c
salt	890741
sugar	620027
butter	493823
flour	466110
eggs	401276
onion	372469
garlic	358364
milk	346769
water	326092
vanilla	270381
olive oil	197877
pepper	179305
brown sugar	174447
tomatoes	163933
egg	160507
baking powder	148277
lemon juice	146414
Salt	122557
cinnamon	117927
sour cream	116682
cream cheese	114423
margarine	112742
celery	112676
baking soda	110690
parsley	102151
chicken	101505
onions	98903
vegetable oil	91395
oil	85600
mayonnaise	84822
pecans	79741
nuts	78471
potatoes	75820
carrots	75458
pineapple	74345
soy sauce	70355
black pepper	69064
thyme	68429
mustard	65948
chicken broth	65112
bacon	64956
honey	64626
oregano	64077
ground beef	64068
unsalted butter	63848
mushrooms	61465
Worcestershire sauce	59328
cornstarch	58476
green pepper	58388
Cheddar cheese	58354

50 rows in set. Elapsed: 0.112 sec. Processed 2.23 million rows, 361.57 MB (19.99 million rows/s., 3.24 GB/s.)

## The Most Complex Recipes with Strawberry

```

SELECT
    title,
    length(NER),
    length(directions)
FROM recipes
WHERE has(NER, 'strawberry')
ORDER BY length(directions) DESC
LIMIT 10

```

Result:

title			length(NER)	length(directions)
Chocolate-Strawberry-Orange Wedding Cake		24	126	
Strawberry Cream Cheese Crumble Tart		19	47	
Charlotte-Style Ice Cream	11		45	
Sinfully Good a Million Layers Chocolate Layer Cake, With Strawb		31		45
Sweetened Berries With Elderflower Sherbet	24		44	
Chocolate-Strawberry Mousse Cake	15		42	
Rhubarb Charlotte with Strawberries and Rum	20		42	
Chef Joey's Strawberry Vanilla Tart	7		37	
Old-Fashioned Ice Cream Sundae Cake	17		37	
Watermelon Cake	16	36		

10 rows in set. Elapsed: 0.215 sec. Processed 2.23 million rows, 1.48 GB (10.35 million rows/s., 6.86 GB/s.)

In this example, we involve **has** function to filter by array elements and sort by the number of directions.

There is a wedding cake that requires the whole 126 steps to produce! Show that directions:

Query:

```
SELECT arrayJoin(directions)
FROM recipes
WHERE title = 'Chocolate-Strawberry-Orange Wedding Cake'
```

Result:

arrayJoin(directions)

Position 1 rack in center and 1 rack in bottom third of oven and preheat to 350F.

Butter one 5-inch-diameter cake pan with 2-inch-high sides, one 8-inch-diameter cake pan with 2-inch-high sides and one 12-inch-diameter cake pan with 2-inch-high sides.

Dust pans with flour; line bottoms with parchment.

Combine 1/3 cup orange juice and 2 ounces unsweetened chocolate in heavy small saucepan.

Stir mixture over medium-low heat until chocolate melts.

Remove from heat.

Gradually mix in 1 2/3 cups orange juice.

Sift 3 cups flour, 2/3 cup cocoa, 2 teaspoons baking soda, 1 teaspoon salt and 1/2 teaspoon baking powder into medium bowl.

using electric mixer, beat 1 cup (2 sticks) butter and 3 cups sugar in large bowl until blended (mixture will look grainy).

Add 4 eggs, 1 at a time, beating to blend after each.

Beat in 1 tablespoon orange peel and 1 tablespoon vanilla extract.

Add dry ingredients alternately with orange juice mixture in 3 additions each, beating well after each addition.

Mix in 1 cup chocolate chips.

Transfer 1 cup plus 2 tablespoons batter to prepared 5-inch pan, 3 cups batter to prepared 8-inch pan and remaining batter (about 6 cups) to 12-inch pan.

Place 5-inch and 8-inch pans on center rack of oven.

Place 12-inch pan on lower rack of oven.

Bake cakes until tester inserted into center comes out clean, about 35 minutes.

Transfer cakes in pans to racks and cool completely.

Mark 4-inch diameter circle on one 6-inch-diameter cardboard cake round.

Cut out marked circle.

Mark 7-inch-diameter circle on one 8-inch-diameter cardboard cake round.

Cut out marked circle.

Mark 11-inch-diameter circle on one 12-inch-diameter cardboard cake round.

Cut out marked circle.

Cut around sides of 5-inch-cake to loosen.

Place 4-inch cardboard over pan.

Hold cardboard and pan together; turn cake out onto cardboard.

Peel off parchment. Wrap cakes on its cardboard in foil.

Repeat turning out, peeling off parchment and wrapping cakes in foil, using 7-inch cardboard for 8-inch cake and 11-inch cardboard for 12-inch cake.

Using remaining ingredients, make 1 more batch of cake batter and bake 3 more cake layers as described above.

Cool cakes in pans.

Cover cakes in pans tightly with foil.

(Can be prepared ahead.)

Let stand at room temperature up to 1 day or double-wrap all cake layers and freeze up to 1 week.

Bring cake layers to room temperature before using.)

Place first 12-inch cake on its cardboard on work surface.

Spread 2 3/4 cups ganache over top of cake and all the way to edge.

Spread 2/3 cup jam over ganache, leaving 1/2-inch chocolate border at edge.

Drop 1 3/4 cups white chocolate frosting by spoonfuls over jam.

Gently spread frosting over jam, leaving 1/2-inch chocolate border at edge.

Rub some cocoa powder over second 12-inch cardboard.

Cut around sides of second 12-inch cake to loosen.

Place cardboard, cocoa side down, over pan.

Turn cake out onto cardboard.

Peel off parchment.

Carefully slide cake off cardboard and onto filling on first 12-inch cake.

Refrigerate.

Place first 8-inch cake on its cardboard on work surface.

Spread 1 cup ganache over top all the way to edge.

Spread 1/4 cup jam over, leaving 1/2-inch chocolate border at edge.

Drop 1 cup white chocolate frosting by spoonfuls over jam.

Gently spread frosting over jam, leaving 1/2-inch chocolate border at edge.

Rub some cocoa over second 8-inch cardboard.

Cut around sides of second 8-inch cake to loosen.

Place cardboard, cocoa side down, over pan.

Turn cake out onto cardboard.

Peel off parchment.

Slide cake off cardboard and onto filling on first 8-inch cake.

Refrigerate.

Place first 5-inch cake on its cardboard on work surface.

Spread 1/2 cup ganache over top of cake and all the way to edge.

Spread 2 tablespoons jam over, leaving 1/2-inch chocolate border at edge.

Drop 1/3 cup white chocolate frosting by spoonfuls over jam.

Gently spread frosting over jam, leaving 1/2-inch chocolate border at edge.

Rub cocoa over second 6-inch cardboard.

Cut around sides of second 5-inch cake to loosen.

Place cardboard, cocoa side down, over pan.

Turn cake out onto cardboard.

Peel off parchment.

Slide cake off cardboard and onto filling on first 5-inch cake.

Chill all cakes 1 hour to set filling.

Place 12-inch tiered cake on its cardboard on revolving cake stand.

Spread 2 2/3 cups frosting over top and sides of cake as a first coat.

Refrigerate cake.

Place 8-inch tiered cake on its cardboard on cake stand.

Spread 1 1/4 cups frosting over top and sides of cake as a first coat.

Refrigerate cake.

Place 5-inch tiered cake on its cardboard on cake stand.

Spread 3/4 cup frosting over top and sides of cake as a first coat.

Refrigerate all cakes until first coats of frosting set, about 1 hour.

(Cakes can be made to this point up to 1 day ahead; cover and keep refrigerate.)

Prepare second batch of frosting, using remaining frosting ingredients and following directions for first batch.

Spoon 2 cups frosting into pastry bag fitted with small star tip.

Place 12-inch cake on its cardboard on large flat platter.

Place platter on cake stand.

Using icing spatula, spread 2 1/2 cups frosting over top and sides of cake; smooth top.

Using filled pastry bag, pipe decorative border around top edge of cake.

Refrigerate cake on platter.

Place 8-inch cake on its cardboard on cake stand.

Using icing spatula, spread 1 1/2 cups frosting over top and sides of cake; smooth top.

Using pastry bag, pipe decorative border around top edge of cake.

Refrigerate cake on its cardboard.

Place 5-inch cake on its cardboard on cake stand.

Using icing spatula, spread 3/4 cup frosting over top and sides of cake; smooth top.

Using pastry bag, pipe decorative border around top edge of cake, spooning more frosting into bag if necessary.

Refrigerate cake on its cardboard.

Keep all cakes refrigerated until frosting sets, about 2 hours.

(Can be prepared 2 days ahead.

Cover loosely; keep refrigerated.)

Place 12-inch cake on platter on work surface.

Press 1 wooden dowel straight down into and completely through center of cake.

Mark dowel 1/4 inch above top of frosting.

Remove dowel and cut with serrated knife at marked point.

Cut 4 more dowels to same length.

Press 1 cut dowel back into center of cake.

Press remaining 4 cut dowels into cake, positioning 3 1/2 inches inward from cake edges and spacing evenly.

Place 8-inch cake on its cardboard on work surface.

Press 1 dowel straight down into and completely through center of cake.

Mark dowel 1/4 inch above top of frosting.

Remove dowel and cut with serrated knife at marked point.

Cut 3 more dowels to same length

Cut 3 more dowels to same length.  
Press 1 cut dowel back into center of cake.

Press remaining 3 cut dowels into cake, positioning 2 1/2 inches inward from edges and spacing evenly.

Using large metal spatula as aid, place 8-inch cake on its cardboard atop dowels in 12-inch cake, centering carefully.

Gently place 5-inch cake on its cardboard atop dowels in 8-inch cake, centering carefully.

Using citrus stripper, cut long strips of orange peel from oranges.

Cut strips into long segments.

To make orange peel coils, wrap peel segment around handle of wooden spoon; gently slide peel off handle so that peel keeps coiled shape.

Garnish cake with orange peel coils, ivy or mint sprigs, and some berries.

(Assembled cake can be made up to 8 hours ahead.

Let stand at cool room temperature.)

Remove top and middle cake tiers.

Remove dowels from cakes.

Cut top and middle cakes into slices.

To cut 12-inch cake: Starting 3 inches inward from edge and inserting knife straight down, cut through from top to bottom to make 6-inch-diameter circle in center of cake.

Cut outer portion of cake into slices; cut inner portion into slices and serve with strawberries.

126 rows in set. Elapsed: 0.011 sec. Processed 8.19 thousand rows, 5.34 MB (737.75 thousand rows/s., 480.59 MB/s.)

## Online Playground

The dataset is also available in the [Online Playground](#).

## ニューヨークタクシー

このデータセットは二つの方法で取得できます:

- 生データからインポート
- パーティション済みデータのダウンロード

## 生データのインポート方法

データセットの説明とダウンロード方法については、<https://github.com/toddwschneider/nyc-taxi-data> と <http://tech.marksblogg.com/billion-nyc-taxi-rides-redshift.html> を参照してください。

ダウンロードすると、CSVファイルで約227GBの非圧縮データが生成されます。ダウンロードは約1Gbitの回線で1時間以上かかります。(s3.amazonaws.com からの並列ダウンロードは1Gbitチャネルの少なくとも半分を回復します)。

一部のファイルは完全にダウンロードできない場合があります。ファイルサイズを確認し、疑わしいと思われるものは再ダウンロードしてください。

ファイルの中には、無効な行が含まれている場合があり、以下のように修正することができます。

```
sed -E '/(.*){18,}/d' data/yellow_tripdata_2010-02.csv > data/yellow_tripdata_2010-02.csv_
sed -E '/(.*){18,}/d' data/yellow_tripdata_2010-03.csv > data/yellow_tripdata_2010-03.csv_
mv data/yellow_tripdata_2010-02.csv_ data/yellow_tripdata_2010-02.csv
mv data/yellow_tripdata_2010-03.csv_ data/yellow_tripdata_2010-03.csv
```

その後、PostgreSQLでデータを前処理する必要があります。これにより、(地図上の点とニューヨーク市の行政区を一致させるために) ポリゴン内の点の選択を作成し、JOINを使用してすべてのデータを单一の非正規化されたフラットテーブルに結合します。これを行うには、PostGISをサポートしたPostgreSQLをインストールする必要があります。

`initialize_database.sh` を実行する際には注意が必要で、すべてのテーブルが正しく作成されていることを手動で再確認してください。

1ヶ月分のデータをPostgreSQLで処理するのに約20～30分、合計で約48時間かかります。

ダウンロードした行数は以下のように確認できます：

```
$ time psql nyc-taxi-data -c "SELECT count(*) FROM trips;"  
### Count  
1298979494  
(1 row)  
  
real 7m9.164s
```

(これはMark Litwintschik氏が一連のブログ記事で報告した11億行をわずかに上回っています)

PostgreSQLのデータは370GBの容量を使用します。

PostgreSQLからデータをエクスポート：

```

COPY (
  SELECT trips.id,
    trips.vendor_id,
    trips.pickup_datetime,
    trips.dropoff_datetime,
    trips.store_and_fwd_flag,
    trips.rate_code_id,
    trips.pickup_longitude,
    trips.pickup_latitude,
    trips.dropoff_longitude,
    trips.dropoff_latitude,
    trips.passenger_count,
    trips.trip_distance,
    trips.fare_amount,
    trips.extra,
    trips.mta_tax,
    trips.tip_amount,
    trips.tolls_amount,
    trips.ehail_fee,
    trips.improvement_surcharge,
    trips.total_amount,
    trips.payment_type,
    trips.trip_type,
    trips.pickup,
    trips.dropoff,
    cab_types.type cab_type,
    weather.precipitation_tenths_of_mm rain,
    weather.snow_depth_mm,
    weather.snowfall_mm,
    weather.max_temperature_tenths_degrees_celsius max_temp,
    weather.min_temperature_tenths_degrees_celsius min_temp,
    weather.average_wind_speed_tenths_of_meters_per_second wind,
    pick_up.gid pickup_nyct2010_gid,
    pick_up.ctlabel pickup_ctlabel,
    pick_up.borocode pickup_borocode,
    pick_up.boroname pickup_boroname,
    pick_up.ct2010 pickup_ct2010,
    pick_up.boroct2010 pickup_boroct2010,
    pick_up.cdeligibil pickup_cdeligibil,
    pick_up.ntacode pickup_ntacode,
    pick_up.ntaname pickup_ntaname,
    pick_up.puma pickup_puma,
    drop_off.gid dropoff_nyct2010_gid,
    drop_off.ctlabel dropoff_ctlabel,
    drop_off.borocode dropoff_borocode,
    drop_off.boroname dropoff_boroname,
    drop_off.ct2010 dropoff_ct2010,
    drop_off.boroct2010 dropoff_boroct2010,
    drop_off.cdeligibil dropoff_cdeligibil,
    drop_off.ntacode dropoff_ntacode,
    drop_off.ntaname dropoff_ntaname,
    drop_off.puma dropoff_puma
  FROM trips
  LEFT JOIN cab_types
    ON trips.cab_type_id = cab_types.id
  LEFT JOIN central_park_weather_observations_raw weather
    ON weather.date = trips.pickup_datetime::date
  LEFT JOIN nyct2010 pick_up
    ON pick_up.gid = trips.pickup_nyct2010_gid
  LEFT JOIN nyct2010 drop_off
    ON drop_off.gid = trips.dropoff_nyct2010_gid
) TO '/opt/milovidov/nyc-taxi-data/trips.tsv';

```

データのスナップショットは、毎秒約50MBの速度で作成されます。スナップショットを作成している間、PostgreSQLは毎秒約28MBの速度でディスクから読み込みます。

これには約5時間かかります。結果として得られるTSVファイルは 590612904969 バイトです。

ClickHouseで一時テーブルを作成します:

```

CREATE TABLE trips
(
    trip_id          UInt32,
    vendor_id        String,
    pickup_datetime DateTime,
    dropoff_datetime Nullable(DateTime),
    store_and_fwd_flag Nullable(FixedString(1)),
    rate_code_id    Nullable(UInt8),
    pickup_longitude Nullable(Float64),
    pickup_latitude  Nullable(Float64),
    dropoff_longitude Nullable(Float64),
    dropoff_latitude Nullable(Float64),
    passenger_count Nullable(UInt8),
    trip_distance   Nullable(Float64),
    fare_amount     Nullable(Float32),
    extra           Nullable(Float32),
    mta_tax          Nullable(Float32),
    tip_amount       Nullable(Float32),
    tolls_amount     Nullable(Float32),
    ehail_fee        Nullable(Float32),
    improvement_surcharge Nullable(Float32),
    total_amount    Nullable(Float32),
    payment_type    Nullable(String),
    trip_type        Nullable(UInt8),
    pickup           Nullable(String),
    dropoff          Nullable(String),
    cab_type         Nullable(String),
    precipitation    Nullable(UInt8),
    snow_depth       Nullable(UInt8),
    snowfall         Nullable(UInt8),
    max_temperature Nullable(UInt8),
    min_temperature Nullable(UInt8),
    average_wind_speed Nullable(UInt8),
    pickup_nyct2010_gid Nullable(UInt8),
    pickup_ctlabel   Nullable(String),
    pickup_borocode Nullable(UInt8),
    pickup_boroname Nullable(String),
    pickup_ct2010    Nullable(String),
    pickup_boroc2010 Nullable(String),
    pickup_cdeligibil Nullable(FixedString(1)),
    pickup_ntacode   Nullable(String),
    pickup_ntaname   Nullable(String),
    pickup_puma      Nullable(String),
    dropoff_nyct2010_gid Nullable(UInt8),
    dropoff_ctlabel  Nullable(String),
    dropoff_borocode Nullable(UInt8),
    dropoff_boroname Nullable(String),
    dropoff_ct2010    Nullable(String),
    dropoff_boroc2010 Nullable(String),
    dropoff_cdeligibil Nullable(String),
    dropoff_ntacode  Nullable(String),
    dropoff_ntaname  Nullable(String),
    dropoff_puma     Nullable(String)
) ENGINE = Log;

```

これは、フィールドをより正しいデータ型に変換したり、可能であればNULLを排除したりするために必要です。

```
$ time clickhouse-client --query="INSERT INTO trips FORMAT TabSeparated" < trips.tsv
```

```
real 75m56.214s
```

データの読み込み速度は 112~140Mb/秒です。

1ストリームで、Log型のテーブルにデータをロードするのに76分かかりました。

このテーブルのデータは 142GB を使用します。

(Postgresから直接データをインポートするには、`COPY ... TO PROGRAM`を使用しても可能です。

残念ながら、天気に関連するフィールド(降水量...平均風速)はすべてNULLで埋め尽くされていました。このため、最終的なデータセットから削除します。

まず、一つのサーバにテーブルを作成します。そのあとで、テーブルを分散させます。

サマリーテーブルを作成します:

```
CREATE TABLE trips_mergetree
ENGINE = MergeTree(pickup_date, pickup_datetime, 8192)
AS SELECT

trip_id,
CAST(vendor_id AS Enum8('1' = 1, '2' = 2, 'CMT' = 3, 'VTS' = 4, 'DDS' = 5, 'B02512' = 10, 'B02598' = 11, 'B02617' = 12, 'B02682' = 13, 'B02764' = 14)) AS vendor_id,
toDate(pickup_datetime) AS pickup_date,
ifNull(pickup_datetime, toDateTime(0)) AS pickup_datetime,
toDate(dropoff_datetime) AS dropoff_date,
ifNull(dropoff_datetime, toDateTime(0)) AS dropoff_datetime,
assumeNotNull(store_and_fwd_flag) IN ('Y', '1', '2') AS store_and_fwd_flag,
assumeNotNull(rate_code_id) AS rate_code_id,
assumeNotNull(pickup_longitude) AS pickup_longitude,
assumeNotNull(pickup_latitude) AS pickup_latitude,
assumeNotNull(dropoff_longitude) AS dropoff_longitude,
assumeNotNull(dropoff_latitude) AS dropoff_latitude,
assumeNotNull(passenger_count) AS passenger_count,
assumeNotNull(trip_distance) AS trip_distance,
assumeNotNull(fare_amount) AS fare_amount,
assumeNotNull(extra) AS extra,
assumeNotNull(mta_tax) AS mta_tax,
assumeNotNull(tip_amount) AS tip_amount,
assumeNotNull(tolls_amount) AS tolls_amount,
assumeNotNull(ehail_fee) AS ehail_fee,
assumeNotNull(improvement_surcharge) AS improvement_surcharge,
assumeNotNull(total_amount) AS total_amount,
CAST((assumeNotNull(payment_type) AS pt) IN ('CSH', 'CASH', 'Cash', 'CAS', 'Cas', '1') ? 'CSH' : (pt IN ('CRD', 'Credit', 'Cre', 'CRE', 'CREDIT', '2') ? 'CRE' : (pt IN ('NOC', 'No Charge', 'No', '3') ? 'NOC' : (pt IN ('DIS', 'Dispute', 'Dis', '4') ? 'DIS' : 'UNK'))) AS Enum8('CSH' = 1, 'CRE' = 2, 'UNK' = 0, 'NOC' = 3, 'DIS' = 4)) AS payment_type_,
assumeNotNull(trip_type) AS trip_type,
ifNull(toFixedString(unhex(pickup), 25), toFixedString("", 25)) AS pickup,
ifNull(toFixedString(unhex(dropoff), 25), toFixedString("", 25)) AS dropoff,
CAST(assumeNotNull(cab_type) AS Enum8('yellow' = 1, 'green' = 2, 'uber' = 3)) AS cab_type,

assumeNotNull(pickup_nyct2010_gid) AS pickup_nyct2010_gid,
toFloat32(ifNull(pickup_ctlable, '0')) AS pickup_ctlable,
assumeNotNull(pickup_borocode) AS pickup_borocode,
CAST(assumeNotNull(pickup_boroname) AS Enum8('Manhattan' = 1, 'Queens' = 4, 'Brooklyn' = 3, '' = 0, 'Bronx' = 2, 'Staten Island' = 5)) AS pickup_boroname,
toFixedString(ifNull(pickup_ct2010, '000000'), 6) AS pickup_ct2010,
toFixedString(ifNull(pickup_boroc2010, '0000000'), 7) AS pickup_boroc2010,
CAST(assumeNotNull(ifNull(pickup_cdeligibil, ' ')) AS Enum8(' ' = 0, 'E' = 1, 'I' = 2)) AS pickup_cdeligibil,
toFixedString(ifNull(pickup_ntacode, '0000'), 4) AS pickup_ntacode,

CAST(assumeNotNull(pickup_ntaname) AS Enum16(' ' = 0, 'Airport' = 1, 'Allerton-Pelham Gardens' = 2, 'Annadale-Huguenot-Prince's Bay-Elttingville' = 3, 'Arden Heights' = 4, 'Astoria' = 5, 'Auburndale' = 6, 'Baisley Park' = 7, 'Bath Beach' = 8, 'Battery Park City-Lower Manhattan' = 9, 'Bay Ridge' = 10, 'Bayside-Bayside Hills' = 11, 'Bedford' = 12, 'Bedford Park-Fordham North' = 13, 'Bellerose' = 14, 'Belmont' = 15, 'Bensonhurst East' = 16, 'Bensonhurst West' = 17, 'Borough Park' = 18, 'Breezy Point-Belle Harbor-Rockaway Park-Broad Channel' = 19, 'Briarwood-Jamaica Hills' = 20, 'Brighton Beach' = 21, 'Bronxdale' = 22, 'Brooklyn Heights-Cobble Hill' = 23, 'Brownsville' = 24, 'Bushwick North' = 25, 'Bushwick South' = 26, 'Cambria Heights' = 27, 'Canarsie' = 28, 'Carroll Gardens-Columbia Street-Red Hook' = 29, 'Central Harlem North-Polo Grounds' = 30, 'Central Harlem South' = 31, 'Charleston-Richmond Valley-Tottenville' = 32, 'Chinatown' = 33, 'Clarendon-Bathgate' = 34, 'Clinton' = 35, 'Clinton Hill' = 36, 'Co-op City' = 37, 'College Point' = 38, 'Corona' = 39, 'Crotona Park East' = 40, 'Crown Heights North' = 41, 'Crown Heights South' = 42, 'Cypress Hills-City Line' = 43, 'DUMBO-Vinegar Hill-Downtown Brooklyn-Boerum Hill' = 44, 'Douglas Manor-Douglaslaston-Little Neck' = 45, 'Dyker Heights' = 46, 'East Concourse-Concourse Village' = 47, 'East Elmhurst' = 48, 'East Flatbush-Farragut' = 49, 'East Flushing' = 50, 'East Harlem North' = 51, 'East Harlem South' = 52, 'East New York' = 53, 'East New York (Pennsylvania Ave)' = 54, 'East Tremont' = 55, 'East Village' = 56, 'East Williamsburg' = 57, 'Eastchester-Edenwald-Baychester' = 58, 'Elmhurst' = 59, 'Elmhurst-Maspeth' = 60, 'Erasmus' = 61, 'Far Rockaway-Bayswater' = 62, 'Flatbush' = 63, 'Flatlands' = 64, 'Flushing' = 65, 'Fordham South' = 66, 'Forest Hills' = 67, 'Fort Greene' = 68, 'Fresh Meadows-Utopia' = 69, 'Ft. Totten-Bay Terrace-Clearview' = 70, 'Georgetown-Marine Park-Bergen Beach-Mill Basin' = 71, 'Glen Oaks-Floral Park-New Hyde Park' = 72, 'Glendale' = 73, 'Gramercy' = 74, 'Grasmere-Arrochar-Ft. Wadsworth' = 75, 'Gravesend' = 76, 'Great Kills' = 77, 'Greenpoint' = 78, 'Grymes Hill-Clifton-Fox Hills' = 79, 'Hamilton Heights' = 80, 'Hammels-Arverne-Edgemere' = 81, 'Highbridge' = 82, 'Hollis' = 83, 'Homecrest' = 84, 'Hudson Yards-Chelsea-Flatiron-Union Square' = 85, 'Hunters Point-Sunnyside-West Maspeth' = 86, 'Hunts Point' = 87, 'Jackson Heights' = 88, 'Jamaica' = 89, 'Jamaica Estates-Holliswood' = 90, 'Kensington-Ocean Parkway' = 91, 'Kew Gardens' = 92, 'Kew Gardens Hills' = 93, 'Kingsbridge Heights' = 94, 'Laurelton' = 95, 'Lenox Hill-Roosevelt Island' = 96, 'Lincoln Square' = 97, 'Lindenwood-Howard Beach' = 98, 'Longwood' = 99, 'Lower East Side' = 100, 'Madison' = 101, 'Manhattanville' = 102, 'Marble Hill-Inwood' = 103, 'Mariner's Harbor-Arlington-Port Ivory-Graniteville' = 104, 'Maspeth' = 105, 'Melrose South-Mott Haven North' = 106, 'Middle Village' = 107, 'Midtown' = 108, 'Newark' = 109, 'Ozone Park' = 110, 'Polaris' = 111, 'Prospect Lefferts Gardens' = 112, 'Ridgewood' = 113, 'Rockaway Beach' = 114, 'South Bronx' = 115, 'South Brooklyn' = 116, 'South Bronx' = 117, 'South Brooklyn' = 118, 'South Bronx' = 119, 'South Brooklyn' = 120, 'South Bronx' = 121, 'South Brooklyn' = 122, 'South Bronx' = 123, 'South Brooklyn' = 124, 'South Bronx' = 125, 'South Brooklyn' = 126, 'South Bronx' = 127, 'South Brooklyn' = 128, 'South Bronx' = 129, 'South Brooklyn' = 130, 'South Bronx' = 131, 'South Brooklyn' = 132, 'South Bronx' = 133, 'South Brooklyn' = 134, 'South Bronx' = 135, 'South Brooklyn' = 136, 'South Bronx' = 137, 'South Brooklyn' = 138, 'South Bronx' = 139, 'South Brooklyn' = 140, 'South Bronx' = 141, 'South Brooklyn' = 142, 'South Bronx' = 143, 'South Brooklyn' = 144, 'South Bronx' = 145, 'South Brooklyn' = 146, 'South Bronx' = 147, 'South Brooklyn' = 148, 'South Bronx' = 149, 'South Brooklyn' = 150, 'South Bronx' = 151, 'South Brooklyn' = 152, 'South Bronx' = 153, 'South Brooklyn' = 154, 'South Bronx' = 155, 'South Brooklyn' = 156, 'South Bronx' = 157, 'South Brooklyn' = 158, 'South Bronx' = 159, 'South Brooklyn' = 160, 'South Bronx' = 161, 'South Brooklyn' = 162, 'South Bronx' = 163, 'South Brooklyn' = 164, 'South Bronx' = 165, 'South Brooklyn' = 166, 'South Bronx' = 167, 'South Brooklyn' = 168, 'South Bronx' = 169, 'South Brooklyn' = 170, 'South Bronx' = 171, 'South Brooklyn' = 172, 'South Bronx' = 173, 'South Brooklyn' = 174, 'South Bronx' = 175, 'South Brooklyn' = 176, 'South Bronx' = 177, 'South Brooklyn' = 178, 'South Bronx' = 179, 'South Brooklyn' = 180, 'South Bronx' = 181, 'South Brooklyn' = 182, 'South Bronx' = 183, 'South Brooklyn' = 184, 'South Bronx' = 185, 'South Brooklyn' = 186, 'South Bronx' = 187, 'South Brooklyn' = 188, 'South Bronx' = 189, 'South Brooklyn' = 190, 'South Bronx' = 191, 'South Brooklyn' = 192, 'South Bronx' = 193, 'South Brooklyn' = 194, 'South Bronx' = 195, 'South Brooklyn' = 196, 'South Bronx' = 197, 'South Brooklyn' = 198, 'South Bronx' = 199, 'South Brooklyn' = 200, 'South Bronx' = 201, 'South Brooklyn' = 202, 'South Bronx' = 203, 'South Brooklyn' = 204, 'South Bronx' = 205, 'South Brooklyn' = 206, 'South Bronx' = 207, 'South Brooklyn' = 208, 'South Bronx' = 209, 'South Brooklyn' = 210, 'South Bronx' = 211, 'South Brooklyn' = 212, 'South Bronx' = 213, 'South Brooklyn' = 214, 'South Bronx' = 215, 'South Brooklyn' = 216, 'South Bronx' = 217, 'South Brooklyn' = 218, 'South Bronx' = 219, 'South Brooklyn' = 220, 'South Bronx' = 221, 'South Brooklyn' = 222, 'South Bronx' = 223, 'South Brooklyn' = 224, 'South Bronx' = 225, 'South Brooklyn' = 226, 'South Bronx' = 227, 'South Brooklyn' = 228, 'South Bronx' = 229, 'South Brooklyn' = 230, 'South Bronx' = 231, 'South Brooklyn' = 232, 'South Bronx' = 233, 'South Brooklyn' = 234, 'South Bronx' = 235, 'South Brooklyn' = 236, 'South Bronx' = 237, 'South Brooklyn' = 238, 'South Bronx' = 239, 'South Brooklyn' = 240, 'South Bronx' = 241, 'South Brooklyn' = 242, 'South Bronx' = 243, 'South Brooklyn' = 244, 'South Bronx' = 245, 'South Brooklyn' = 246, 'South Bronx' = 247, 'South Brooklyn' = 248, 'South Bronx' = 249, 'South Brooklyn' = 250, 'South Bronx' = 251, 'South Brooklyn' = 252, 'South Bronx' = 253, 'South Brooklyn' = 254, 'South Bronx' = 255, 'South Brooklyn' = 256, 'South Bronx' = 257, 'South Brooklyn' = 258, 'South Bronx' = 259, 'South Brooklyn' = 260, 'South Bronx' = 261, 'South Brooklyn' = 262, 'South Bronx' = 263, 'South Brooklyn' = 264, 'South Bronx' = 265, 'South Brooklyn' = 266, 'South Bronx' = 267, 'South Brooklyn' = 268, 'South Bronx' = 269, 'South Brooklyn' = 270, 'South Bronx' = 271, 'South Brooklyn' = 272, 'South Bronx' = 273, 'South Brooklyn' = 274, 'South Bronx' = 275, 'South Brooklyn' = 276, 'South Bronx' = 277, 'South Brooklyn' = 278, 'South Bronx' = 279, 'South Brooklyn' = 280, 'South Bronx' = 281, 'South Brooklyn' = 282, 'South Bronx' = 283, 'South Brooklyn' = 284, 'South Bronx' = 285, 'South Brooklyn' = 286, 'South Bronx' = 287, 'South Brooklyn' = 288, 'South Bronx' = 289, 'South Brooklyn' = 290, 'South Bronx' = 291, 'South Brooklyn' = 292, 'South Bronx' = 293, 'South Brooklyn' = 294, 'South Bronx' = 295, 'South Brooklyn' = 296, 'South Bronx' = 297, 'South Brooklyn' = 298, 'South Bronx' = 299, 'South Brooklyn' = 300, 'South Bronx' = 301, 'South Brooklyn' = 302, 'South Bronx' = 303, 'South Brooklyn' = 304, 'South Bronx' = 305, 'South Brooklyn' = 306, 'South Bronx' = 307, 'South Brooklyn' = 308, 'South Bronx' = 309, 'South Brooklyn' = 310, 'South Bronx' = 311, 'South Brooklyn' = 312, 'South Bronx' = 313, 'South Brooklyn' = 314, 'South Bronx' = 315, 'South Brooklyn' = 316, 'South Bronx' = 317, 'South Brooklyn' = 318, 'South Bronx' = 319, 'South Brooklyn' = 320, 'South Bronx' = 321, 'South Brooklyn' = 322, 'South Bronx' = 323, 'South Brooklyn' = 324, 'South Bronx' = 325, 'South Brooklyn' = 326, 'South Bronx' = 327, 'South Brooklyn' = 328, 'South Bronx' = 329, 'South Brooklyn' = 330, 'South Bronx' = 331, 'South Brooklyn' = 332, 'South Bronx' = 333, 'South Brooklyn' = 334, 'South Bronx' = 335, 'South Brooklyn' = 336, 'South Bronx' = 337, 'South Brooklyn' = 338, 'South Bronx' = 339, 'South Brooklyn' = 340, 'South Bronx' = 341, 'South Brooklyn' = 342, 'South Bronx' = 343, 'South Brooklyn' = 344, 'South Bronx' = 345, 'South Brooklyn' = 346, 'South Bronx' = 347, 'South Brooklyn' = 348, 'South Bronx' = 349, 'South Brooklyn' = 350, 'South Bronx' = 351, 'South Brooklyn' = 352, 'South Bronx' = 353, 'South Brooklyn' = 354, 'South Bronx' = 355, 'South Brooklyn' = 356, 'South Bronx' = 357, 'South Brooklyn' = 358, 'South Bronx' = 359, 'South Brooklyn' = 360, 'South Bronx' = 361, 'South Brooklyn' = 362, 'South Bronx' = 363, 'South Brooklyn' = 364, 'South Bronx' = 365, 'South Brooklyn' = 366, 'South Bronx' = 367, 'South Brooklyn' = 368, 'South Bronx' = 369, 'South Brooklyn' = 370, 'South Bronx' = 371, 'South Brooklyn' = 372, 'South Bronx' = 373, 'South Brooklyn' = 374, 'South Bronx' = 375, 'South Brooklyn' = 376, 'South Bronx' = 377, 'South Brooklyn' = 378, 'South Bronx' = 379, 'South Brooklyn' = 380, 'South Bronx' = 381, 'South Brooklyn' = 382, 'South Bronx' = 383, 'South Brooklyn' = 384, 'South Bronx' = 385, 'South Brooklyn' = 386, 'South Bronx' = 387, 'South Brooklyn' = 388, 'South Bronx' = 389, 'South Brooklyn' = 390, 'South Bronx' = 391, 'South Brooklyn' = 392, 'South Bronx' = 393, 'South Brooklyn' = 394, 'South Bronx' = 395, 'South Brooklyn' = 396, 'South Bronx' = 397, 'South Brooklyn' = 398, 'South Bronx' = 399, 'South Brooklyn' = 400, 'South Bronx' = 401, 'South Brooklyn' = 402, 'South Bronx' = 403, 'South Brooklyn' = 404, 'South Bronx' = 405, 'South Brooklyn' = 406, 'South Bronx' = 407, 'South Brooklyn' = 408, 'South Bronx' = 409, 'South Brooklyn' = 410, 'South Bronx' = 411, 'South Brooklyn' = 412, 'South Bronx' = 413, 'South Brooklyn' = 414, 'South Bronx' = 415, 'South Brooklyn' = 416, 'South Bronx' = 417, 'South Brooklyn' = 418, 'South Bronx' = 419, 'South Brooklyn' = 420, 'South Bronx' = 421, 'South Brooklyn' = 422, 'South Bronx' = 423, 'South Brooklyn' = 424, 'South Bronx' = 425, 'South Brooklyn' = 426, 'South Bronx' = 427, 'South Brooklyn' = 428, 'South Bronx' = 429, 'South Brooklyn' = 430, 'South Bronx' = 431, 'South Brooklyn' = 432, 'South Bronx' = 433, 'South Brooklyn' = 434, 'South Bronx' = 435, 'South Brooklyn' = 436, 'South Bronx' = 437, 'South Brooklyn' = 438, 'South Bronx' = 439, 'South Brooklyn' = 440, 'South Bronx' = 441, 'South Brooklyn' = 442, 'South Bronx' = 443, 'South Brooklyn' = 444, 'South Bronx' = 445, 'South Brooklyn' = 446, 'South Bronx' = 447, 'South Brooklyn' = 448, 'South Bronx' = 449, 'South Brooklyn' = 450, 'South Bronx' = 451, 'South Brooklyn' = 452, 'South Bronx' = 453, 'South Brooklyn' = 454, 'South Bronx' = 455, 'South Brooklyn' = 456, 'South Bronx' = 457, 'South Brooklyn' = 458, 'South Bronx' = 459, 'South Brooklyn' = 460, 'South Bronx' = 461, 'South Brooklyn' = 462, 'South Bronx' = 463, 'South Brooklyn' = 464, 'South Bronx' = 465, 'South Brooklyn' = 466, 'South Bronx' = 467, 'South Brooklyn' = 468, 'South Bronx' = 469, 'South Brooklyn' = 470, 'South Bronx' = 471, 'South Brooklyn' = 472, 'South Bronx' = 473, 'South Brooklyn' = 474, 'South Bronx' = 475, 'South Brooklyn' = 476, 'South Bronx' = 477, 'South Brooklyn' = 478, 'South Bronx' = 479, 'South Brooklyn' = 480, 'South Bronx' = 481, 'South Brooklyn' = 482, 'South Bronx' = 483, 'South Brooklyn' = 484, 'South Bronx' = 485, 'South Brooklyn' = 486, 'South Bronx' = 487, 'South Brooklyn' = 488, 'South Bronx' = 489, 'South Brooklyn' = 490, 'South Bronx' = 491, 'South Brooklyn' = 492, 'South Bronx' = 493, 'South Brooklyn' = 494, 'South Bronx' = 495, 'South Brooklyn' = 496, 'South Bronx' = 497, 'South Brooklyn' = 498, 'South Bronx' = 499, 'South Brooklyn' = 500, 'South Bronx' = 501, 'South Brooklyn' = 502, 'South Bronx' = 503, 'South Brooklyn' = 504, 'South Bronx' = 505, 'South Brooklyn' = 506, 'South Bronx' = 507, 'South Brooklyn' = 508, 'South Bronx' = 509, 'South Brooklyn' = 510, 'South Bronx' = 511, 'South Brooklyn' = 512, 'South Bronx' = 513, 'South Brooklyn' = 514, 'South Bronx' = 515, 'South Brooklyn' = 516, 'South Bronx' = 517, 'South Brooklyn' = 518, 'South Bronx' = 519, 'South Brooklyn' = 520, 'South Bronx' = 521, 'South Brooklyn' = 522, 'South Bronx' = 523, 'South Brooklyn' = 524, 'South Bronx' = 525, 'South Brooklyn' = 526, 'South Bronx' = 527, 'South Brooklyn' = 528, 'South Bronx' = 529, 'South Brooklyn' = 530, 'South Bronx' = 531, 'South Brooklyn' = 532, 'South Bronx' = 533, 'South Brooklyn' = 534, 'South Bronx' = 535, 'South Brooklyn' = 536, 'South Bronx' = 537, 'South Brooklyn' = 538, 'South Bronx' = 539, 'South Brooklyn' = 540, 'South Bronx' = 541, 'South Brooklyn' = 542, 'South Bronx' = 543, 'South Brooklyn' = 544, 'South Bronx' = 545, 'South Brooklyn' = 546, 'South Bronx' = 547, 'South Brooklyn' = 548, 'South Bronx' = 549, 'South Brooklyn' = 550, 'South Bronx' = 551, 'South Brooklyn' = 552, 'South Bronx' = 553, 'South Brooklyn' = 554, 'South Bronx' = 555, 'South Brooklyn' = 556, 'South Bronx' = 557, 'South Brooklyn' = 558, 'South Bronx' = 559, 'South Brooklyn' = 560, 'South Bronx' = 561, 'South Brooklyn' = 562, 'South Bronx' = 563, 'South Brooklyn' = 564, 'South Bronx' = 565, 'South Brooklyn' = 566, 'South Bronx' = 567, 'South Brooklyn' = 568, 'South Bronx' = 569, 'South Brooklyn' = 570, 'South Bronx' = 571, 'South Brooklyn' = 572, 'South Bronx' = 573, 'South Brooklyn' = 574, 'South Bronx' = 575, 'South Brooklyn' = 576, 'South Bronx' = 577, 'South Brooklyn' = 578, 'South Bronx' = 579, 'South Brooklyn' = 580, 'South Bronx' = 581, 'South Brooklyn' = 582, 'South Bronx' = 583, 'South Brooklyn' = 584, 'South Bronx' = 585, 'South Brooklyn' = 586, 'South Bronx' = 587, 'South Brooklyn' = 588, 'South Bronx' = 589, 'South Brooklyn' = 590, 'South Bronx' = 591, 'South Brooklyn' = 592, 'South Bronx' = 593, 'South Brooklyn' = 594, 'South Bronx' = 595, 'South Brooklyn' = 596, 'South Bronx' = 597, 'South Brooklyn' = 598, 'South Bronx' = 599, 'South Brooklyn' = 600, 'South Bronx' = 601, 'South Brooklyn' = 602, 'South Bronx' = 603, 'South Brooklyn' = 604, 'South Bronx' = 605, 'South Brooklyn' = 606, 'South Bronx' = 607, 'South Brooklyn' = 608, 'South Bronx' = 609, 'South Brooklyn' = 610, 'South Bronx' = 611, 'South Brooklyn' = 612, 'South Bronx' = 613, 'South Brooklyn' = 614, 'South Bronx' = 615, 'South Brooklyn' = 616, 'South Bronx' = 617, 'South Brooklyn' = 618, 'South Bronx' = 619, 'South Brooklyn' = 620, 'South Bronx' = 621, 'South Brooklyn' = 622, 'South Bronx' = 623, 'South Brooklyn' = 624, 'South Bronx' = 625, 'South Brooklyn' = 626, 'South Bronx' = 627, 'South Brooklyn' = 628, 'South Bronx' = 629, 'South Brooklyn' = 630, 'South Bronx' = 631, 'South Brooklyn' = 632, 'South Bronx' = 633, 'South Brooklyn' = 634, 'South Bronx' = 635, 'South Brooklyn' = 636, 'South Bronx' = 637, 'South Brooklyn' = 638, 'South Bronx' = 639, 'South Brooklyn' = 640, 'South Bronx' = 641, 'South Brooklyn' = 642, 'South Bronx' = 643, 'South Brooklyn' = 644, 'South Bronx' = 645, 'South Brooklyn' = 646, 'South Bronx' = 647, 'South Brooklyn' = 648, 'South Bronx' = 649, 'South Brooklyn' = 650, 'South Bronx' = 651, 'South Brooklyn' = 652, 'South Bronx' = 653, 'South Brooklyn' = 654, 'South Bronx' = 655, 'South Brooklyn' = 656, 'South Bronx' = 657, 'South Brooklyn' = 658, 'South Bronx' = 659, 'South Brooklyn' = 660, 'South Bronx' = 661, 'South Brooklyn' = 662, 'South Bronx' = 663, 'South Brooklyn' = 664, 'South Bronx' = 665, 'South Brooklyn' = 666, 'South Bronx' = 667, 'South Brooklyn' = 668, 'South Bronx' = 669, 'South Brooklyn' = 670, 'South Bronx' = 671, 'South Brooklyn' = 672, 'South Bronx' = 673, 'South Brooklyn' = 674, 'South Bronx' = 675, 'South Brooklyn' = 676, 'South Bronx' = 677, 'South Brooklyn' = 678, 'South Bronx' = 679, 'South Brooklyn' = 680, 'South Bronx' = 681, 'South Brooklyn' = 682, 'South Bronx' = 683, 'South Brooklyn' = 684, 'South Bronx' = 685, 'South Brooklyn' = 686, 'South Bronx' = 687, 'South Brooklyn' = 688, 'South Bronx' = 689, 'South Brooklyn' = 690, 'South Bronx' = 691, 'South Brooklyn' = 692, 'South Bronx' = 693, 'South Brooklyn' = 694, 'South Bronx' = 695, 'South Brooklyn' = 696, 'South Bronx' = 697, 'South Brooklyn' = 698, 'South Bronx' = 699, 'South Brooklyn' = 700, 'South Bronx' = 701, 'South Brooklyn' = 702, 'South Bronx' = 703, 'South Brooklyn' = 704, 'South Bronx' = 705, 'South Brooklyn' = 706, 'South Bronx' = 707, 'South Brooklyn' = 708, 'South Bronx' = 709, 'South Brooklyn' = 710, 'South Bronx' = 711, 'South Brooklyn' = 712, 'South Bronx' = 713, 'South Brooklyn' = 714, 'South Bronx' = 715, 'South Brooklyn' = 716, 'South Bronx' = 717, 'South Brooklyn' = 718, 'South Bronx' = 719, 'South Brooklyn' = 720, 'South Bronx' = 721, 'South Brooklyn' = 722, 'South Bronx' = 723, 'South Brooklyn' = 724, 'South Bronx' = 725, 'South Brooklyn' = 726, 'South Bronx' = 727, 'South Brooklyn' = 728, 'South Bronx' = 729, 'South Brooklyn' = 730, 'South Bronx' = 731, 'South Brooklyn' = 732, 'South Bronx' = 733, 'South Brooklyn' = 734, 'South Bronx' = 735, 'South Brooklyn' = 736, 'South Bronx' = 737, 'South Brooklyn' = 738, 'South Bronx' = 739, 'South Brooklyn' = 740, 'South Bronx' = 741, 'South Brooklyn' = 742, 'South Bronx' = 743, 'South Brooklyn' = 744, 'South Bronx' = 745, 'South Brooklyn' = 746, 'South Bronx' = 747, 'South Brooklyn' = 748, 'South Bronx' = 749, 'South Brooklyn' = 750, 'South Bronx' = 751, 'South Brooklyn' = 752, 'South Bronx' = 753, 'South Brooklyn' = 754, 'South Bronx' = 755, 'South Brooklyn' = 756, 'South Bronx' = 757, 'South Brooklyn' = 758, 'South Bronx' = 759, 'South Brooklyn' = 760, 'South Bronx' = 761, 'South Brooklyn' = 762, 'South Bronx' = 763, 'South Brooklyn' = 764, 'South Bronx' = 765, 'South Brooklyn' = 766, 'South Bronx' = 767, 'South Brooklyn' = 768, 'South Bronx' = 769, 'South Brooklyn' = 770, 'South Bronx' = 771, 'South Brooklyn' = 772, 'South Bronx' = 773, 'South Brooklyn' = 774, 'South Bronx' = 775, 'South Brooklyn' = 776, 'South Bronx' = 777, 'South Brooklyn' = 778, 'South Bronx' = 779, 'South Brooklyn' = 780, 'South Bronx' = 781, 'South Brooklyn' = 782, 'South Bronx' = 783, 'South Brooklyn' = 784, 'South Bronx' = 785, 'South Brooklyn' = 786, 'South Bronx' = 787, 'South Brooklyn' = 788, 'South Bronx' = 789, 'South Brooklyn' = 790, 'South Bronx' = 791, 'South Brooklyn' = 792, 'South Bronx' = 793, 'South Brooklyn' = 794, 'South Bronx' = 795, 'South Brooklyn' = 796, 'South Bronx' = 797, 'South Brooklyn' = 798, 'South Bronx' = 799, 'South Brooklyn' = 800, 'South Bronx' = 801, 'South Brooklyn' = 802, 'South Bronx' = 803, 'South Brooklyn' = 804, 'South Bronx' = 805, 'South Brooklyn' = 806, 'South Bronx' = 807, 'South Brooklyn' = 808, 'South Bronx' = 809, 'South Brooklyn' = 810, 'South Bronx' = 811, 'South Brooklyn' = 812, 'South Bronx' = 813, 'South Brooklyn' = 814, 'South Bronx' = 815, 'South Brooklyn' = 816, 'South Bronx' = 817, 'South Brooklyn' = 818, 'South Bronx' = 819, 'South Brooklyn' = 820, 'South Bronx' = 821, 'South Brooklyn' = 822, 'South Bronx' = 823, 'South Brooklyn' = 824, 'South Bronx' = 825, 'South Brooklyn' = 826, 'South Bronx' = 827, 'South Brooklyn' = 828, 'South Bronx' = 829, 'South Brooklyn' = 830, 'South Bronx' = 831, 'South Brooklyn' = 832, 'South Bronx' = 833, 'South Brooklyn' = 834, 'South Bronx' = 835, 'South Brooklyn' = 836, 'South Bronx' = 837, 'South Brooklyn' = 838, 'South Bronx' = 839, 'South Brooklyn' = 840, 'South Bronx' = 841, 'South Brooklyn' = 842, 'South Bronx' = 843, 'South Brooklyn' = 844, 'South Bronx' = 845, 'South Brooklyn' = 846, 'South Bronx' = 847, 'South Brooklyn' = 848, 'South Bronx' = 849, 'South Brooklyn' = 850, 'South Bronx' = 851, 'South Brooklyn' = 852, 'South Bronx' = 853, 'South Brooklyn' = 854, 'South Bronx' = 855, 'South Brooklyn' = 856, 'South Bronx' = 857, 'South Brooklyn' = 858, 'South Bronx' = 859, 'South Brooklyn' = 860, 'South Bronx' = 861, 'South Brooklyn' = 862, 'South Bronx' = 863, 'South Brooklyn' = 864, 'South Bronx' = 865, 'South Brooklyn' = 866, 'South Bronx' = 867, 'South Brooklyn' = 868, 'South Bronx' = 869, 'South Brooklyn' = 870, 'South Bronx' = 871, 'South Brooklyn' = 872, 'South Bronx' = 873, 'South Brooklyn' = 874, 'South Bronx' = 875, 'South Brooklyn' = 876, 'South Bronx' = 877, 'South Brooklyn' = 878, 'South Bronx' = 879, 'South Brooklyn' = 880, 'South Bronx' = 881, 'South Brooklyn' = 882, 'South Bronx' = 883, 'South Brooklyn' = 884, 'South Bronx' = 885, 'South Brooklyn' = 886, 'South Bronx' = 887, 'South Brooklyn' = 888, 'South Bronx' = 889, 'South Brooklyn' = 890, 'South Bronx' = 891, 'South Brooklyn' = 892, 'South Bronx' = 893, 'South Brooklyn' = 894, 'South Bronx' = 895, 'South Brooklyn' = 896, 'South Bronx' = 897, 'South Brooklyn' = 898, 'South Bronx' = 899, 'South Brooklyn' = 900, 'South Bronx' = 901, 'South Brooklyn' = 902, 'South Bronx' = 903, 'South Brooklyn' = 904, 'South Bronx' = 905, 'South Brooklyn' = 906, 'South Bronx' = 907, 'South Brooklyn' = 908, 'South Bronx' = 909, 'South Brooklyn' = 910, 'South Bronx' = 911, 'South Brooklyn' = 912, 'South Bronx' = 913, 'South Brooklyn' = 914, 'South Bronx' = 915, 'South Brooklyn' = 916, 'South Bronx' = 917, 'South Brooklyn' = 918, 'South Bronx' = 919, 'South Brooklyn' = 920, 'South Bronx' = 921, 'South Brooklyn' = 922, 'South Bronx' = 923, 'South Brooklyn' = 924, 'South Bronx' = 925, 'South Brooklyn' = 926, 'South Bronx' = 927, 'South Brooklyn' = 928, 'South Bronx' = 929, 'South Brooklyn' = 930, 'South Bronx' = 931, 'South Brooklyn' = 932, 'South Bronx' = 933, 'South Brooklyn' = 934, 'South Bronx' = 935, 'South Brooklyn' = 936, 'South Bronx' = 937, 'South Brooklyn' = 938, 'South Bronx' = 939, 'South Brooklyn' = 940, 'South Bronx' = 941, 'South Brooklyn' = 942, 'South Bronx' = 943, 'South Brooklyn' = 944, 'South Bronx' = 945, 'South Brooklyn' = 946, 'South Bronx' = 947, 'South Brooklyn' = 948, 'South Bronx' = 949, 'South Brooklyn' = 950, 'South Bronx' = 951, 'South Brooklyn' = 952, 'South Bronx' = 953, 'South Brooklyn' = 954, 'South Bronx' = 955, 'South Brooklyn' = 956, 'South Bronx' = 957, 'South Brooklyn' = 958, 'South Bronx' = 959, 'South Brooklyn' = 960, 'South Bronx' = 961, 'South Brooklyn' = 962, 'South Bronx' = 963, 'South Brooklyn' = 964, 'South Bronx' = 965, 'South Brooklyn' = 966, 'South Bronx' = 967, 'South Brooklyn' = 968, 'South Bronx' = 969, 'South Brooklyn' = 970, 'South Bronx' = 971, 'South Brooklyn' = 972, 'South Bronx' = 973, 'South Brooklyn' = 974, 'South Bronx' = 975, 'South Brooklyn' = 976, 'South Bronx' = 977, 'South Brooklyn' = 978, 'South Bronx' = 979, 'South Brooklyn' = 980, 'South Bronx' = 981, 'South Brooklyn' = 982, 'South Bronx' = 983, 'South Brooklyn' = 984, 'South Bronx' = 985, 'South Brooklyn' = 986, 'South Bronx' = 987, 'South Brooklyn' = 988, 'South Bronx' = 989, 'South Brooklyn' = 990, 'South Bronx' = 991, 'South Brooklyn' = 992, 'South Bronx' = 993, 'South Brooklyn'
```

'Midtown South' = 108, 'Midwood' = 109, 'Morningside Heights' = 110, 'Morrisania-Melrose' = 111, 'Mott Haven-Port Morris' = 112, 'Mount Hope' = 113, 'Murray Hill' = 114, 'Murray Hill-Kips Bay' = 115, 'New Brighton-Silver Lake' = 116, 'New Dorp-Midland Beach' = 117, 'New Springville-Bloomfield-Travis' = 118, 'North Corona' = 119, 'North Riverdale-Fieldston-Riverdale' = 120, 'North Side-South Side' = 121, 'Norwood' = 122, 'Oakland Gardens' = 123, 'Oakwood-Oakwood Beach' = 124, 'Ocean Hill' = 125, 'Ocean Parkway South' = 126, 'Old Astoria' = 127, 'Old Town-Dongan Hills-South Beach' = 128, 'Ozone Park' = 129, 'Park Slope-Gowanus' = 130, 'Parkchester' = 131, 'Pelham Bay-Country Club-City Island' = 132, 'Pelham Parkway' = 133, 'Pomonok-Flushing Heights-Hillcrest' = 134, 'Port Richmond' = 135, 'Prospect Heights' = 136, 'Prospect Lefferts Gardens-Wingate' = 137, 'Queens Village' = 138, 'Queensboro Hill' = 139, 'Queensbridge-Ravenswood-Long Island City' = 140, 'Rego Park' = 141, 'Richmond Hill' = 142, 'Ridgewood' = 143, 'Rikers Island' = 144, 'Rosedale' = 145, 'Rossville-Woodrow' = 146, 'Rugby-Remsen Village' = 147, 'Schuylerville-Throgs Neck-Edgewater Park' = 148, 'Seagate-Coney Island' = 149, 'Sheepshead Bay-Gerritsen Beach-Manhattan Beach' = 150, 'SoHo-TriBeCa-Civic Center-Little Italy' = 151, 'Soundview-Bruckner' = 152, 'Soundview-Castle Hill-Clason Point-Harding Park' = 153, 'South Jamaica' = 154, 'South Ozone Park' = 155, 'Springfield Gardens North' = 156, 'Springfield Gardens South-Brookville' = 157, 'Spuyten Duyvil-Kingsbridge' = 158, 'St. Albans' = 159, 'Stapleton-Rosebank' = 160, 'Starrett City' = 161, 'Steinway' = 162, 'Stuyvesant Heights' = 163, 'Stuyvesant Town-Cooper Village' = 164, 'Sunset Park East' = 165, 'Sunset Park West' = 166, 'Todt Hill-Emerson Hill-Heartland Village-Lighthouse Hill' = 167, 'Turtle Bay-East Midtown' = 168, 'University Heights-Morris Heights' = 169, 'Upper East Side-Carnegie Hill' = 170, 'Upper West Side' = 171, 'Van Cortlandt Village' = 172, 'Van Nest-Morris Park-Westchester Square' = 173, 'Washington Heights North' = 174, 'Washington Heights South' = 175, 'West Brighton' = 176, 'West Concourse' = 177, 'West Farms-Bronx River' = 178, 'West New Brighton-New Brighton-St. George' = 179, 'West Village' = 180, 'Westchester-Unionport' = 181, 'Westerleigh' = 182, 'Whitestone' = 183, 'Williamsbridge-Olinville' = 184, 'Williamsburg' = 185, 'Windsor Terrace' = 186, 'Woodhaven' = 187, 'Woodlawn-Wakefield' = 188, 'Woodside' = 189, 'Yorkville' = 190, 'park-cemetery-etc-Bronx' = 191, 'park-cemetery-etc-Brooklyn' = 192, 'park-cemetery-etc-Manhattan' = 193, 'park-cemetery-etc-Queens' = 194, 'park-cemetery-etc-Staten Island' = 195)) AS pickup\_ntaname,

toUInt16(ifNull(pickup\_puma, '0')) AS pickup\_puma,

assumeNotNull(dropoff\_nyct2010\_gid) AS dropoff\_nyct2010\_gid,  
toFloat32(ifNull(dropoff\_ctlabel, '0')) AS dropoff\_ctlabel,  
assumeNotNull(dropoff\_borocode) AS dropoff\_borocode,  
CAST(assumeNotNull(dropoff\_boroname) AS Enum8('Manhattan' = 1, 'Queens' = 4, 'Brooklyn' = 3, '' = 0, 'Bronx' = 2, 'Staten Island' = 5)) AS dropoff\_boroname,  
toFixedString(ifNull(dropoff\_ct2010, '000000'), 6) AS dropoff\_ct2010,  
toFixedString(ifNull(dropoff\_boroc2010, '0000000'), 7) AS dropoff\_boroc2010,  
CAST(assumeNotNull(ifNull(dropoff\_cdeligibil, ' ')) AS Enum8(' ' = 0, 'E' = 1, 'I' = 2)) AS dropoff\_cdeligibil,  
toFixedString(ifNull(dropoff\_ntacode, '0000'), 4) AS dropoff\_ntacode,

CAST(assumeNotNull(dropoff\_ntaname) AS Enum16(' ' = 0, 'Airport' = 1, 'Allerton-Pelham Gardens' = 2, 'Annadale-Huguenot-Prince's Bay-Eltingville' = 3, 'Arden Heights' = 4, 'Astoria' = 5, 'Auburndale' = 6, 'Baisley Park' = 7, 'Bath Beach' = 8, 'Battery Park City-Lower Manhattan' = 9, 'Bay Ridge' = 10, 'Bayside-Bayside Hills' = 11, 'Bedford' = 12, 'Bedford Park-Fordham North' = 13, 'Bellerose' = 14, 'Belmont' = 15, 'Bensonhurst East' = 16, 'Bensonhurst West' = 17, 'Borough Park' = 18, 'Breezy Point-Belle Harbor-Rockaway Park-Broad Channel' = 19, 'Briarwood-Jamaica Hills' = 20, 'Brighton Beach' = 21, 'Bronxdale' = 22, 'Brooklyn Heights-Cobble Hill' = 23, 'Brownsville' = 24, 'Bushwick North' = 25, 'Bushwick South' = 26, 'Cambria Heights' = 27, 'Canarsie' = 28, 'Carroll Gardens-Columbia Street-Red Hook' = 29, 'Central Harlem North-Polo Grounds' = 30, 'Central Harlem South' = 31, 'Charleston-Richmond Valley-Tottenville' = 32, 'Chinatown' = 33, 'Claremont-Bathgate' = 34, 'Clinton' = 35, 'Clinton Hill' = 36, 'Co-op City' = 37, 'College Point' = 38, 'Corona' = 39, 'Crotona Park East' = 40, 'Crown Heights North' = 41, 'Crown Heights South' = 42, 'Cypress Hills-City Line' = 43, 'DUMBO-Vinegar Hill-Downtown Brooklyn-Boerum Hill' = 44, 'Douglas Manor-Douglaslaston-Little Neck' = 45, 'Dyker Heights' = 46, 'East Concourse-Concourse Village' = 47, 'East Elmhurst' = 48, 'East Flatbush-Farragut' = 49, 'East Flushing' = 50, 'East Harlem North' = 51, 'East Harlem South' = 52, 'East New York' = 53, 'East New York (Pennsylvania Ave)' = 54, 'East Tremont' = 55, 'East Village' = 56, 'East Williamsburg' = 57, 'Eastchester-Edenwald-Baychester' = 58, 'Elmhurst' = 59, 'Elmhurst-Maspeth' = 60, 'Erasmus' = 61, 'Far Rockaway-Bayswater' = 62, 'Flatbush' = 63, 'Flatlands' = 64, 'Flushing' = 65, 'Fordham South' = 66, 'Forest Hills' = 67, 'Fort Greene' = 68, 'Fresh Meadows-Utopia' = 69, 'Ft. Totten-Bay Terrace-Clearview' = 70, 'Georgetown-Marine Park-Bergen Beach-Mill Basin' = 71, 'Glen Oaks-Floral Park-New Hyde Park' = 72, 'Glendale' = 73, 'Gramercy' = 74, 'Grasmere-Arrochar-Ft. Wadsworth' = 75, 'Gravesend' = 76, 'Great Kills' = 77, 'Greenpoint' = 78, 'Grymes Hill-Clifton-Fox Hills' = 79, 'Hamilton Heights' = 80, 'Hammels-Arverne-Edgemere' = 81, 'Highbridge' = 82, 'Hollis' = 83, 'Homecrest' = 84, 'Hudson Yards-Chelsea-Flatiron-Union Square' = 85, 'Hunters Point-Sunnyside-West Maspeth' = 86, 'Hunts Point' = 87, 'Jackson Heights' = 88, 'Jamaica' = 89, 'Jamaica Estates-Holliswood' = 90, 'Kensington-Ocean Parkway' = 91, 'Kew Gardens' = 92, 'Kew Gardens Hills' = 93, 'Kingsbridge Heights' = 94, 'Laurelton' = 95, 'Lenox Hill-Roosevelt Island' = 96, 'Lincoln Square' = 97, 'Lindenwood-Howard Beach' = 98, 'Longwood' = 99, 'Lower East Side' = 100, 'Madison' = 101, 'Manhattanville' = 102, 'Marble Hill-Inwood' = 103, 'Mariner's Harbor-Arlington-Port Ivory-Graniteville' = 104, 'Maspeth' = 105, 'Melrose South-Mott Haven North' = 106, 'Middle Village' = 107, 'Midtown-Midtown South' = 108, 'Midwood' = 109, 'Morningside Heights' = 110, 'Morrisania-Melrose' = 111, 'Mott Haven-Port Morris' = 112, 'Mount Hope' = 113, 'Murray Hill' = 114, 'Murray Hill-Kips Bay' = 115, 'New Brighton-Silver Lake' = 116, 'New Dorp-Midland Beach' = 117, 'New Springville-Bloomfield-Travis' = 118, 'North Corona' = 119, 'North Riverdale-Fieldston-Riverdale' = 120, 'North Side-South Side' = 121, 'Norwood' = 122, 'Oakland Gardens' = 123, 'Oakwood-Oakwood Beach' = 124, 'Ocean Hill' = 125, 'Ocean Parkway South' = 126, 'Old Astoria' = 127, 'Old Town-Dongan Hills-South Beach' = 128, 'Ozone Park' = 129, 'Park Slope-Gowanus' = 130, 'Parkchester' = 131, 'Pelham Bay-Country Club-City Island' = 132, 'Pelham Parkway' = 133, 'Pomonok-Flushing Heights-Hillcrest' = 134, 'Port Richmond' = 135, 'Prospect Heights' = 136, 'Prospect Lefferts Gardens-Wingate' = 137, 'Queens Village' = 138, 'Queensboro Hill' = 139, 'Queensbridge-Ravenswood-Long Island City' = 140, 'Rego Park' = 141, 'Richmond Hill' = 142, 'Ridgewood' = 143, 'Rikers Island' = 144, 'Rosedale' = 145, 'Rossville-Woodrow' = 146, 'Rugby-Remsen Village' = 147, 'Schuylerville-Throgs Neck-Edgewater Park' = 148, 'Seagate-Coney Island' = 149, 'Sheepshead Bay-Gerritsen Beach-Manhattan Beach' = 150, 'SoHo-TriBeCa-Civic Center-Little Italy' = 151, 'Soundview-Bruckner' = 152, 'Soundview-Castle Hill-Clason Point-Harding Park' = 153, 'South Jamaica' = 154, 'South Ozone Park' = 155, 'Springfield Gardens North' = 156, 'Springfield Gardens South-Brookville' = 157, 'Spuyten Duyvil-Kingsbridge' = 158, 'St. Albans' = 159, 'Stapleton-

```
Rosebank' = 160, 'Starrett City' = 161, 'Steinway' = 162, 'Stuyvesant Heights' = 163, 'Stuyvesant Town-Cooper Village' = 164, 'Sunset Park East' = 165, 'Sunset Park West' = 166, 'Todt Hill-Emerson Hill-Heartland Village-Lighthouse Hill' = 167, 'Turtle Bay-East Midtown' = 168, 'University Heights-Morris Heights' = 169, 'Upper East Side-Carnegie Hill' = 170, 'Upper West Side' = 171, 'Van Cortlandt Village' = 172, 'Van Nest-Morris Park-Westchester Square' = 173, 'Washington Heights North' = 174, 'Washington Heights South' = 175, 'West Brighton' = 176, 'West Concourse' = 177, 'West Farms-Bronx River' = 178, 'West New Brighton-New Brighton-St. George' = 179, 'West Village' = 180, 'Westchester-Unionport' = 181, 'Westerleigh' = 182, 'Whitestone' = 183, 'Williamsbridge-Olinville' = 184, 'Williamsburg' = 185, 'Windsor Terrace' = 186, 'Woodhaven' = 187, 'Woodlawn-Wakefield' = 188, 'Woodside' = 189, 'Yorkville' = 190, 'park-cemetery-etc-Bronx' = 191, 'park-cemetery-etc-Brooklyn' = 192, 'park-cemetery-etc-Manhattan' = 193, 'park-cemetery-etc-Queens' = 194, 'park-cemetery-etc-Staten Island' = 195)) AS dropoff_ntaname,
```

```
toUInt16(ifNull(dropoff_puma, '0')) AS dropoff_puma
```

```
FROM trips
```

これは1秒間に約428,000行の速度で3030秒かかります。

より速くロードするには、MergeTreeの代わりにLogエンジンでテーブルを作成します。この場合、ダウンロードは200秒よりも速く動作します。

テーブルは126GBのディスク領域を使用します。

```
SELECT formatReadableSize(sum(bytes)) FROM system.parts WHERE table = 'trips_mergetree' AND active
```

```
formatReadableSize(sum(bytes))—  
126.18 GiB |
```

その他に、MergeTreeでOPTIMIZEクエリを実行することができます。しかし、これがなくてもうまく動作するので、必須ではありません。

## パーティションされたデータのダウンロード

```
$ curl -O https://datasets.clickhouse.com/trips_mergetree/partitions/trips_mergetree.tar  
$ tar xvf trips_mergetree.tar -C /var/lib/clickhouse # path to ClickHouse data directory  
$ # check permissions of unpacked data, fix if required  
$ sudo service clickhouse-server restart  
$ clickhouse-client --query "select count(*) from datasets.trips_mergetree"
```

## 情報

以下で説明するクエリを実行する場合は、`datasets.trips_mergetree`のように完全なテーブル名を使用する必要があります。

## 単一サーバーでの結果

Q1:

```
SELECT cab_type, count(*) FROM trips_mergetree GROUP BY cab_type
```

0.490秒

Q2:

```
SELECT passenger_count, avg(total_amount) FROM trips_mergetree GROUP BY passenger_count
```

1.224秒

Q3:

```
SELECT passenger_count, toYear(pickup_date) AS year, count(*) FROM trips_mergetree GROUP BY passenger_count, year
```

2.104秒

Q4:

```
SELECT passenger_count, toYear(pickup_date) AS year, round(trip_distance) AS distance, count(*) FROM trips_mergetree GROUP BY passenger_count, year, distance ORDER BY year, count(*) DESC
```

3.593秒

使用したサーバは以下の通りです。

Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz が2つ、合計16個の物理カーネル、128GiB RAM、RAID-5 の 8 x 6 TB HDD。

実行時間は、3回実行した中で最もよい結果です。しかし、2回目の実行からは、クエリはファイルシステムのキャッシュからデータを読み込みます。それ以上のキャッシュは発生しません：データは読み出され、各実行で処理されます。

3台のサーバーにテーブルを作成する：

各サーバー上で：

```
CREATE TABLE default.trips_mergetree_third ( trip_id UInt32, vendor_id Enum8('1' = 1, '2' = 2, 'CMT' = 3, 'VTS' = 4, 'DDS' = 5, 'B02512' = 10, 'B02598' = 11, 'B02617' = 12, 'B02682' = 13, 'B02764' = 14), pickup_date Date, pickup_datetime DateTime, dropoff_date Date, dropoff_datetime DateTime, store_and_fwd_flag UInt8, rate_code_id UInt8, pickup_longitude Float64, pickup_latitude Float64, dropoff_longitude Float64, dropoff_latitude Float64, passenger_count UInt8, trip_distance Float64, fare_amount Float32, extra Float32, mta_tax Float32, tip_amount Float32, tolls_amount Float32, ehail_fee Float32, improvement_surcharge Float32, total_amount Float32, payment_type_Enum8('UNK' = 0, 'CSH' = 1, 'CRE' = 2, 'NOC' = 3, 'DIS' = 4), trip_type UInt8, pickup FixedString(25), dropoff FixedString(25), cab_type Enum8('yellow' = 1, 'green' = 2, 'uber' = 3), pickup_nyct2010_gid UInt8, pickup_ctlabel Float32, pickup_borocode UInt8, pickup_boroname Enum8('' = 0, 'Manhattan' = 1, 'Bronx' = 2, 'Brooklyn' = 3, 'Queens' = 4, 'Staten Island' = 5), pickup_ct2010 FixedString(6), pickup_boroct2010 FixedString(7), pickup_cdeligibil Enum8('' = 0, 'E' = 1, 'I' = 2), pickup_ntacode FixedString(4), pickup_ntaname Enum16('' = 0, 'Airport' = 1, 'Allerton-Pelham Gardens' = 2, 'Annadale-Huguenot-Prince\l's Bay-Eltingville' = 3, 'Arden Heights' = 4, 'Astoria' = 5, 'Auburndale' = 6, 'Baisley Park' = 7, 'Bath Beach' = 8, 'Battery Park City-Lower Manhattan' = 9, 'Bay Ridge' = 10, 'Bayside-Bayside Hills' = 11, 'Bedford' = 12, 'Bedford Park-Fordham North' = 13, 'Bellerose' = 14, 'Belmont' = 15, 'Bensonhurst East' = 16, 'Bensonhurst West' = 17, 'Borough Park' = 18, 'Breezy Point-Belle Harbor-Rockaway Park-Broad Channel' = 19, 'Briarwood-Jamaica Hills' = 20, 'Brighton Beach' = 21, 'Bronxdale' = 22, 'Brooklyn Heights-Cobble Hill' = 23, 'Brownsville' = 24, 'Bushwick North' = 25, 'Bushwick South' = 26, 'Cambria Heights' = 27, 'Canarsie' = 28, 'Carroll Gardens-Columbia Street-Red Hook' = 29, 'Central Harlem North-Polo Grounds' = 30, 'Central Harlem South' = 31, 'Charleston-Richmond Valley-Tottenville' = 32, 'Chinatown' = 33, 'Clarendon-Bathgate' = 34, 'Clinton' = 35, 'Clinton Hill' = 36, 'Co-op City' = 37, 'College Point' = 38, 'Corona' = 39, 'Crotona Park East' = 40, 'Crown Heights North' = 41, 'Crown Heights South' = 42, 'Cypress Hills-City Line' = 43, 'DUMBO-Vinegar Hill-Downtown Brooklyn-Boerum Hill' = 44, 'Douglas Manor-Douglaston-Little Neck' = 45, 'Dyker Heights' = 46, 'East Concourse-Concourse Village' = 47, 'East Elmhurst' = 48, 'East Flatbush-Farragut' = 49, 'East Flushing' = 50, 'East Harlem North' = 51, 'East Harlem South' = 52, 'East New York' = 53, 'East New York (Pennsylvania Ave)' = 54, 'East Tremont' = 55, 'East Village' = 56, 'East Williamsburg' = 57, 'Eastchester-Edenwald-Baychester' = 58, 'Elmhurst' = 59, 'Elmhurst-Maspeth' = 60, 'Erasmus' = 61, 'Far Rockaway-Bayswater' = 62, 'Flatbush' = 63, 'Flatlands' = 64, 'Flushing' = 65, 'Fordham South' = 66, 'Forest Hills' = 67, 'Fort Greene' = 68, 'Fresh Meadows-Utopia' = 69, 'Ft. Totten-Bay Terrace-Clearview' = 70, 'Georgetown-Marine Park-Bergen Beach-Mill Basin' = 71, 'Glen Oaks-Floral Park-New Hyde Park' = 72, 'Glendale' = 73, 'Gramercy' = 74, 'Grasmere-Arrochar-Ft. Wadsworth' = 75, 'Gravesend' = 76, 'Great Kills' = 77, 'Greenpoint' = 78, 'Grymes Hill-Clifton-Fox Hills' = 79, 'Hamilton Heights' = 80, 'Hammels-Arverne-Edgemere' = 81, 'Highbridge' = 82, 'Hollis' = 83, 'Homecrest' = 84, 'Hudson Yards-Chelsea-Flatiron-Union Square' = 85, 'Hunters Point-Sunnyside-West Maspeth' = 86, 'Hunts Point' = 87, 'Jackson Heights' = 88, 'Jamaica' = 89, 'Jamaica Estates-Holliswood' = 90, 'Kensington-Ocean Parkway' = 91, 'Kew Gardens' = 92, 'Kew Gardens Hills' = 93, 'Kingsbridge Heights' = 94, 'Laurelton' = 95, 'Lenox Hill-Roosevelt Island' = 96, 'Lincoln Square' = 97, 'Lindenwood-Howard Beach' = 98, 'Longwood' = 99, 'Lower East Side' = 100, 'Madison' = 101, 'Manhattanville' = 102, 'Marble Hill-Inwood' = 103, 'Mariner\l's Harbor-Arlington-Port Ivory-Graniteville' = 104, 'Maspeth' = 105, 'Melrose South-Mott Haven North' = 106, 'Middle Village' = 107, 'Midtown-Midtown South' = 108, 'Midwood' = 109, 'Morningside Heights' = 110, 'Morrisania-Melrose' = 111, 'Mott Haven-Port Morris' = 112, 'Mount Hope' = 113, 'Murray Hill' = 114, 'Murray Hill-Kips Bay' = 115, 'New Brighton-Silver Lake' = 116, 'New Dorp-Midland Beach' = 117, 'New Springville-Bloomfield-Travis' = 118, 'North Corona' = 119, 'North Riverdale-
```

Fieldston-Riverdale' = 120, 'North Side-South Side' = 121, 'Norwood' = 122, 'Oakland Gardens' = 123, 'Oakwood-Oakwood Beach' = 124, 'Ocean Hill' = 125, 'Ocean Parkway South' = 126, 'Old Astoria' = 127, 'Old Town-Dongan Hills-South Beach' = 128, 'Ozone Park' = 129, 'Park Slope-Gowanus' = 130, 'Parkchester' = 131, 'Pelham Bay-Country Club-City Island' = 132, 'Pelham Parkway' = 133, 'Pomonok-Flushing Heights-Hillcrest' = 134, 'Port Richmond' = 135, 'Prospect Heights' = 136, 'Prospect Lefferts Gardens-Wingate' = 137, 'Queens Village' = 138, 'Queensboro Hill' = 139, 'Queensbridge-Ravenswood-Long Island City' = 140, 'Rego Park' = 141, 'Richmond Hill' = 142, 'Ridgewood' = 143, 'Rikers Island' = 144, 'Rosedale' = 145, 'Rossville-Woodrow' = 146, 'Rugby-Remsen Village' = 147, 'Schuylerville-Throgs Neck-Edgewater Park' = 148, 'Seagate-Coney Island' = 149, 'Sheepshead Bay-Gerritsen Beach-Manhattan Beach' = 150, 'SoHo-TriBeCa-Civic Center-Little Italy' = 151, 'Soundview-Bruckner' = 152, 'Soundview-Castle Hill-Clason Point-Harding Park' = 153, 'South Jamaica' = 154, 'South Ozone Park' = 155, 'Springfield Gardens North' = 156, 'Springfield Gardens South-Brookville' = 157, 'Spuyten Duyvil-Kingsbridge' = 158, 'St. Albans' = 159, 'Stapleton-Rosebank' = 160, 'Starrett City' = 161, 'Steinway' = 162, 'Stuyvesant Heights' = 163, 'Stuyvesant Town-Cooper Village' = 164, 'Sunset Park East' = 165, 'Sunset Park West' = 166, 'Todt Hill-Emerson Hill-Heartland Village-Lighthouse Hill' = 167, 'Turtle Bay-East Midtown' = 168, 'University Heights-Morris Heights' = 169, 'Upper East Side-Carnegie Hill' = 170, 'Upper West Side' = 171, 'Van Cortlandt Village' = 172, 'Van Nest-Morris Park-Westchester Square' = 173, 'Washington Heights North' = 174, 'Washington Heights South' = 175, 'West Brighton' = 176, 'West Concourse' = 177, 'West Farms-Bronx River' = 178, 'West New Brighton-New Brighton-St. George' = 179, 'West Village' = 180, 'Westchester-Unionport' = 181, 'Westerleigh' = 182, 'Whitestone' = 183, 'Williamsbridge-Olinville' = 184, 'Williamsburg' = 185, 'Windsor Terrace' = 186, 'Woodhaven' = 187, 'Woodlawn-Wakefield' = 188, 'Woodside' = 189, 'Yorkville' = 190, 'park-cemetery-etc-Bronx' = 191, 'park-cemetery-etc-Brooklyn' = 192, 'park-cemetery-etc-Manhattan' = 193, 'park-cemetery-etc-Queens' = 194, 'park-cemetery-etc-Staten Island' = 195), pickup\_puma UInt16, dropoff\_nyct2010\_gid UInt8, dropoff\_ctlabel Float32, dropoff\_borocode UInt8, dropoff\_boroname Enum8(" = 0, 'Manhattan' = 1, 'Bronx' = 2, 'Brooklyn' = 3, 'Queens' = 4, 'Staten Island' = 5), dropoff\_ct2010 FixedString(6), dropoff\_boroc2010 FixedString(7), dropoff\_cdeligibil Enum8(" = 0, 'E' = 1, 'I' = 2), dropoff\_ntacode FixedString(4), dropoff\_ntaname Enum16(" = 0, 'Airport' = 1, 'Allerton-Pelham Gardens' = 2, 'Annadale-Huguenot-Prince's Bay-Eltingville' = 3, 'Arden Heights' = 4, 'Astoria' = 5, 'Auburndale' = 6, 'Baisley Park' = 7, 'Bath Beach' = 8, 'Battery Park City-Lower Manhattan' = 9, 'Bay Ridge' = 10, 'Bayside-Bayside Hills' = 11, 'Bedford' = 12, 'Bedford Park-Fordham North' = 13, 'Bellerose' = 14, 'Belmont' = 15, 'Bensonhurst East' = 16, 'Bensonhurst West' = 17, 'Borough Park' = 18, 'Breezy Point-Belle Harbor-Rockaway Park-Broad Channel' = 19, 'Briarwood-Jamaica Hills' = 20, 'Brighton Beach' = 21, 'Bronxdale' = 22, 'Brooklyn Heights-Cobble Hill' = 23, 'Brownsville' = 24, 'Bushwick North' = 25, 'Bushwick South' = 26, 'Cambria Heights' = 27, 'Canarsie' = 28, 'Carroll Gardens-Columbia Street-Red Hook' = 29, 'Central Harlem North-Polo Grounds' = 30, 'Central Harlem South' = 31, 'Charleston-Richmond Valley-Tottenville' = 32, 'Chinatown' = 33, 'Clarendon-Bathgate' = 34, 'Clinton' = 35, 'Clinton Hill' = 36, 'Co-op City' = 37, 'College Point' = 38, 'Corona' = 39, 'Crotona Park East' = 40, 'Crown Heights North' = 41, 'Crown Heights South' = 42, 'Cypress Hills-City Line' = 43, 'DUMBO-Vinegar Hill-Downtown Brooklyn-Boerum Hill' = 44, 'Douglas Manor-Douglaslaston-Little Neck' = 45, 'Dyker Heights' = 46, 'East Concourse-Concourse Village' = 47, 'East Elmhurst' = 48, 'East Flatbush-Farragut' = 49, 'East Flushing' = 50, 'East Harlem North' = 51, 'East Harlem South' = 52, 'East New York' = 53, 'East New York (Pennsylvania Ave)' = 54, 'East Tremont' = 55, 'East Village' = 56, 'East Williamsburg' = 57, 'Eastchester-Edenvale-Baychester' = 58, 'Elmhurst' = 59, 'Elmhurst-Maspeth' = 60, 'Erasmus' = 61, 'Far Rockaway-Bayswater' = 62, 'Flatbush' = 63, 'Flatlands' = 64, 'Flushing' = 65, 'Fordham South' = 66, 'Forest Hills' = 67, 'Fort Greene' = 68, 'Fresh Meadows-Utopia' = 69, 'Ft. Totten-Bay Terrace-Clearview' = 70, 'Georgetown-Marine Park-Bergen Beach-Mill Basin' = 71, 'Glen Oaks-Floral Park-New Hyde Park' = 72, 'Glendale' = 73, 'Gramercy' = 74, 'Grasmere-Arrochar-Ft. Wadsworth' = 75, 'Gravesend' = 76, 'Great Kills' = 77, 'Greenpoint' = 78, 'Grymes Hill-Clifton-Fox Hills' = 79, 'Hamilton Heights' = 80, 'Hammels-Arverne-Edgemere' = 81, 'Highbridge' = 82, 'Hollis' = 83, 'Homecrest' = 84, 'Hudson Yards-Chelsea-Flatiron-Union Square' = 85, 'Hunters Point-Sunnyside-West Maspeth' = 86, 'Hunts Point' = 87, 'Jackson Heights' = 88, 'Jamaica' = 89, 'Jamaica Estates-Holliswood' = 90, 'Kensington-Ocean Parkway' = 91, 'Kew Gardens' = 92, 'Kew Gardens Hills' = 93, 'Kingsbridge Heights' = 94, 'Laurelton' = 95, 'Lenox Hill-Roosevelt Island' = 96, 'Lincoln Square' = 97, 'Lindenwood-Howard Beach' = 98, 'Longwood' = 99, 'Lower East Side' = 100, 'Madison' = 101, 'Manhattanville' = 102, 'Marble Hill-Inwood' = 103, 'Mariner's Harbor-Arlington-Port Ivory-Graniteville' = 104, 'Maspeth' = 105, 'Melrose South-Mott Haven North' = 106, 'Middle Village' = 107, 'Midtown-Midtown South' = 108, 'Midwood' = 109, 'Morningside Heights' = 110, 'Morrisania-Melrose' = 111, 'Mott Haven-Port Morris' = 112, 'Mount Hope' = 113, 'Murray Hill' = 114, 'Murray Hill-Kips Bay' = 115, 'New Brighton-Silver Lake' = 116, 'New Dorp-Midland Beach' = 117, 'New Springville-Bloomfield-Travis' = 118, 'North Corona' = 119, 'North Riverdale-Fieldston-Riverdale' = 120, 'North Side-South Side' = 121, 'Norwood' = 122, 'Oakland Gardens' = 123, 'Oakwood-Oakwood Beach' = 124, 'Ocean Hill' = 125, 'Ocean Parkway South' = 126, 'Old Astoria' = 127, 'Old Town-Dongan Hills-South Beach' = 128, 'Ozone Park' = 129, 'Park Slope-Gowanus' = 130, 'Parkchester' = 131, 'Pelham Bay-Country Club-City Island' = 132, 'Pelham Parkway' = 133, 'Pomonok-Flushing Heights-Hillcrest' = 134, 'Port Richmond' = 135, 'Prospect Heights' = 136, 'Prospect Lefferts Gardens-Wingate' = 137, 'Queens Village' = 138, 'Queensboro Hill' = 139, 'Queensbridge-Ravenswood-Long Island City' = 140, 'Rego Park' = 141, 'Richmond Hill' = 142, 'Ridgewood' = 143, 'Rikers Island' = 144, 'Rosedale' = 145, 'Rossville-Woodrow' = 146, 'Rugby-Remsen Village' = 147, 'Schuylerville-Throgs Neck-Edgewater Park' = 148, 'Seagate-Coney Island' = 149, 'Sheepshead Bay-Gerritsen Beach-Manhattan Beach' = 150, 'SoHo-TriBeCa-Civic Center-Little Italy' = 151, 'Soundview-Bruckner' = 152, 'Soundview-Castle Hill-Clason Point-Harding Park' = 153, 'South Jamaica' = 154, 'South Ozone Park' = 155, 'Springfield Gardens North' = 156, 'Springfield Gardens South-Brookville' = 157, 'Spuyten Duyvil-Kingsbridge' = 158, 'St. Albans' = 159, 'Stapleton-Rosebank' = 160, 'Starrett City' = 161, 'Steinway' = 162, 'Stuyvesant Heights' = 163, 'Stuyvesant Town-Cooper Village' = 164, 'Sunset Park East' = 165, 'Sunset Park West' = 166, 'Todt Hill-Emerson Hill-Heartland Village-Lighthouse Hill' = 167, 'Turtle Bay-East Midtown' = 168, 'University Heights-Morris Heights' = 169, 'Upper East Side-Carnegie Hill' = 170, 'Upper West Side' = 171, 'Van Cortlandt Village' = 172, 'Van Nest-Morris Park-Westchester Square' = 173, 'Washington Heights North' = 174, 'Washington Heights South' = 175, 'West Brighton' = 176, 'West Concourse' = 177, 'West Farms-Bronx River' = 178, 'West New Brighton-New Brighton-St. George' = 179, 'West Village' = 180, 'Westchester-Unionport' = 181, 'Westerleigh' = 182, 'Whitestone' = 183, 'Williamsbridge-Olinville' = 184, 'Williamsburg' = 185, 'Windsor Terrace' = 186, 'Woodhaven' = 187, 'Woodlawn-Wakefield' = 188, 'Woodside' = 189, 'Yorkville' = 190, 'park-cemetery-etc-Bronx' = 191, 'park-cemetery-etc-Brooklyn' = 192, 'park-cemetery-etc-Manhattan' = 193, 'park-cemetery-etc-Queens' = 194, 'park-cemetery-etc-Staten Island' = 195), dropoff\_puma UInt16) ENGINE = MergeTree(pickup\_date, pickup\_datetime, 8192)

移行元サーバー上:

```
CREATE TABLE trips_mergetree_x3 AS trips_mergetree_third ENGINE = Distributed(perftest, default,
trips_mergetree_third, rand())
```

次のクエリで、データを再配布します:

```
INSERT INTO trips_mergetree_x3 SELECT * FROM trips_mergetree
```

これには2454秒かかります。

三つのサーバー上でクエリを実行すると:

Q1:0.212秒

Q2:0.438秒

Q3:0.733秒

Q4:1.241秒

クエリは線形にスケーリングされているので、ここでは予想通りです。

また、140台のサーバーのクラスタからの結果も得られます:

Q1:0.028秒

Q2:0.043秒

Q3:0.051秒

Q4:0.072秒

この場合、クエリの処理時間は、ネットワークのレイテンシによって決定されます。

フィンランドのYandexデータセンターにあるクライアントをロシアのクラスター上に置いてクエリを実行したところ、約20ミリ秒のレイテンシが追加されました。

## サマリ

サーバ	Q1	Q2	Q3	Q4
1	0.490	1.224	2.104	3.593
3	0.212	0.438	0.733	1.241
140	0.028	0.043	0.051	0.072

## AMPLab Big Data ベンチマーク

<https://amplab.cs.berkeley.edu/benchmark/> を参照して下さい。

<https://aws.amazon.com> で無料アカウントにサインアップしてください。クレジットカード、電子メール、電話番号が必要です。

[https://console.aws.amazon.com/iam/home?nc2=h\\_m\\_sc#security\\_credential](https://console.aws.amazon.com/iam/home?nc2=h_m_sc#security_credential) で新しいアクセスキーを取得します。

コンソールで以下を実行します:

```
$ sudo apt-get install s3cmd
$ mkdir tiny; cd tiny;
$ s3cmd sync s3://big-data-benchmark/pavlo/text-deflate/tiny/ .
$ cd ..
$ mkdir 1node; cd 1node;
$ s3cmd sync s3://big-data-benchmark/pavlo/text-deflate/1node/ .
$ cd ..
$ mkdir 5nodes; cd 5nodes;
$ s3cmd sync s3://big-data-benchmark/pavlo/text-deflate/5nodes/ .
$ cd ..
```

次のClickHouseクエリを実行します:

```
CREATE TABLE rankings_tiny
(
    pageURL String,
    pageRank UInt32,
    avgDuration UInt32
) ENGINE = Log;

CREATE TABLE uservisits_tiny
(
    sourceIP String,
    destinationURL String,
    visitDate Date,
    adRevenue Float32,
    UserAgent String,
    cCode FixedString(3),
    lCode FixedString(6),
    searchWord String,
    duration UInt32
) ENGINE = MergeTree(visitDate, visitDate, 8192);

CREATE TABLE rankings_1node
(
    pageURL String,
    pageRank UInt32,
    avgDuration UInt32
) ENGINE = Log;

CREATE TABLE uservisits_1node
(
    sourceIP String,
    destinationURL String,
    visitDate Date,
    adRevenue Float32,
    UserAgent String,
    cCode FixedString(3),
    lCode FixedString(6),
    searchWord String,
    duration UInt32
) ENGINE = MergeTree(visitDate, visitDate, 8192);

CREATE TABLE rankings_5nodes_on_single
(
    pageURL String,
    pageRank UInt32,
    avgDuration UInt32
) ENGINE = Log;

CREATE TABLE uservisits_5nodes_on_single
(
    sourceIP String,
    destinationURL String,
    visitDate Date,
    adRevenue Float32,
    UserAgent String,
    cCode FixedString(3),
    lCode FixedString(6),
    searchWord String,
    duration UInt32
) ENGINE = MergeTree(visitDate, visitDate, 8192);
```

コンソールに戻って以下を実行します:

```
$ for i in tiny/rankings/*.deflate; do echo $i; zlib-flate -uncompress < $i | clickhouse-client --host=example-perftest01j --query="INSERT INTO rankings_tiny FORMAT CSV"; done
$ for i in tiny/uservisits/*.deflate; do echo $i; zlib-flate -uncompress < $i | clickhouse-client --host=example-perftest01j --query="INSERT INTO uservisits_tiny FORMAT CSV"; done
$ for i in 1node/rankings/*.deflate; do echo $i; zlib-flate -uncompress < $i | clickhouse-client --host=example-perftest01j --query="INSERT INTO rankings_1node FORMAT CSV"; done
$ for i in 1node/uservisits/*.deflate; do echo $i; zlib-flate -uncompress < $i | clickhouse-client --host=example-perftest01j --query="INSERT INTO uservisits_1node FORMAT CSV"; done
$ for i in 5nodes/rankings/*.deflate; do echo $i; zlib-flate -uncompress < $i | clickhouse-client --host=example-perftest01j --query="INSERT INTO rankings_5nodes_on_single FORMAT CSV"; done
$ for i in 5nodes/uservisits/*.deflate; do echo $i; zlib-flate -uncompress < $i | clickhouse-client --host=example-perftest01j --query="INSERT INTO uservisits_5nodes_on_single FORMAT CSV"; done
```

データの取得サンプルクエリ:

```
SELECT pageURL, pageRank FROM rankings_1node WHERE pageRank > 1000
SELECT substring(sourceIP, 1, 8), sum(adRevenue) FROM uservisits_1node GROUP BY substring(sourceIP, 1, 8)
SELECT
    sourceIP,
    sum(adRevenue) AS totalRevenue,
    avg(pageRank) AS pageRank
FROM rankings_1node ALL INNER JOIN
(
    SELECT
        sourceIP,
        destinationURL AS pageURL,
        adRevenue
    FROM uservisits_1node
    WHERE (visitDate > '1980-01-01') AND (visitDate < '1980-04-01')
) USING pageURL
GROUP BY sourceIP
ORDER BY totalRevenue DESC
LIMIT 1
```

## WikiStat

参照: <http://dumps.wikimedia.org/other/pagecounts-raw/>

テーブルの作成:

```
CREATE TABLE wikistat
(
    date Date,
    time DateTime,
    project String,
    subproject String,
    path String,
    hits UInt64,
    size UInt64
) ENGINE = MergeTree(date, (path, time), 8192);
```

データのロード:

```
$ for i in {2007..2016}; do for j in {01..12}; do echo $i-$j >&2; curl -sSL "http://dumps.wikimedia.org/other/pagecounts-raw/$i/$i-$j/" | grep -oE 'pagecounts-[0-9]+-[0-9]+\.\gz'; done; done | sort | uniq | tee links.txt
$ cat links.txt | while read link; do wget http://dumps.wikimedia.org/other/pagecounts-raw/$(echo $link | sed -r 's/pagecounts-[0-9]{4})([0-9]{2})[0-9]{2}-[0-9]+\.\gz\1/')/$(echo $link | sed -r 's/pagecounts-[0-9]{4})([0-9]{2})[0-9]{2}-[0-9]+\.\gz\1-\2/'|$link; done
$ ls -1 /opt/wikistat/ | grep gz | while read i; do echo $i; gzip -cd /opt/wikistat/$i | ./wikistat-loader --time=$(echo -n $i | sed -r 's/pagecounts-[0-9]{4})([0-9]{2})[0-9]{2}-([0-9]{2})([0-9]{2})([0-9]{2})\.\gz\1-\2-\3 \4-00-00/'); clickhouse-client --query="INSERT INTO wikistat FORMAT TabSeparated"; done
```

## Criteo のテラバイトクリックログ

<http://labs.criteo.com/downloads/download-terabyte-click-logs/> から データをダウンロードして下さい。

ログをインポートするテーブルを作成します:

```
CREATE TABLE criteo_log (date Date, clicked UInt8, int1 Int32, int2 Int32, int3 Int32, int4 Int32, int5 Int32, int6 Int32, int7 Int32, int8 Int32, int9 Int32, int10 Int32, int11 Int32, int12 Int32, int13 Int32, cat1 String, cat2 String, cat3 String, cat4 String, cat5 String, cat6 String, cat7 String, cat8 String, cat9 String, cat10 String, cat11 String, cat12 String, cat13 String, cat14 String, cat15 String, cat16 String, cat17 String, cat18 String, cat19 String, cat20 String, cat21 String, cat22 String, cat23 String, cat24 String, cat25 String, cat26 String) ENGINE = Log
```

データダウンロード:

```
$ for i in {00..23}; do echo $i; zcat datasets/criteo/day_${i#0}.gz | sed -r 's/^2000-01-$i/00/24'\t' | clickhouse-client --host=example-perftest01j --query="INSERT INTO criteo_log FORMAT TabSeparated"; done
```

変換されたデータのテーブルを作成します:

```

CREATE TABLE criteo
(
    date Date,
    clicked UInt8,
    int1 Int32,
    int2 Int32,
    int3 Int32,
    int4 Int32,
    int5 Int32,
    int6 Int32,
    int7 Int32,
    int8 Int32,
    int9 Int32,
    int10 Int32,
    int11 Int32,
    int12 Int32,
    int13 Int32,
    icat1 UInt32,
    icat2 UInt32,
    icat3 UInt32,
    icat4 UInt32,
    icat5 UInt32,
    icat6 UInt32,
    icat7 UInt32,
    icat8 UInt32,
    icat9 UInt32,
    icat10 UInt32,
    icat11 UInt32,
    icat12 UInt32,
    icat13 UInt32,
    icat14 UInt32,
    icat15 UInt32,
    icat16 UInt32,
    icat17 UInt32,
    icat18 UInt32,
    icat19 UInt32,
    icat20 UInt32,
    icat21 UInt32,
    icat22 UInt32,
    icat23 UInt32,
    icat24 UInt32,
    icat25 UInt32,
    icat26 UInt32
) ENGINE = MergeTree(date, intHash32(icat1), (date, intHash32(icat1)), 8192)

```

生のログからデータを変換し、第二のテーブルに入れます：

```

INSERT INTO criteo SELECT date, clicked, int1, int2, int3, int4, int5, int6, int7, int8, int9, int10, int11, int12, int13,
reinterpretAsUInt32(unhex(cat1)) AS icat1, reinterpretAsUInt32(unhex(cat2)) AS icat2,
reinterpretAsUInt32(unhex(cat3)) AS icat3, reinterpretAsUInt32(unhex(cat4)) AS icat4,
reinterpretAsUInt32(unhex(cat5)) AS icat5, reinterpretAsUInt32(unhex(cat6)) AS icat6,
reinterpretAsUInt32(unhex(cat7)) AS icat7, reinterpretAsUInt32(unhex(cat8)) AS icat8,
reinterpretAsUInt32(unhex(cat9)) AS icat9, reinterpretAsUInt32(unhex(cat10)) AS icat10,
reinterpretAsUInt32(unhex(cat11)) AS icat11, reinterpretAsUInt32(unhex(cat12)) AS icat12,
reinterpretAsUInt32(unhex(cat13)) AS icat13, reinterpretAsUInt32(unhex(cat14)) AS icat14,
reinterpretAsUInt32(unhex(cat15)) AS icat15, reinterpretAsUInt32(unhex(cat16)) AS icat16,
reinterpretAsUInt32(unhex(cat17)) AS icat17, reinterpretAsUInt32(unhex(cat18)) AS icat18,
reinterpretAsUInt32(unhex(cat19)) AS icat19, reinterpretAsUInt32(unhex(cat20)) AS icat20,
reinterpretAsUInt32(unhex(cat21)) AS icat21, reinterpretAsUInt32(unhex(cat22)) AS icat22,
reinterpretAsUInt32(unhex(cat23)) AS icat23, reinterpretAsUInt32(unhex(cat24)) AS icat24,
reinterpretAsUInt32(unhex(cat25)) AS icat25, reinterpretAsUInt32(unhex(cat26)) AS icat26 FROM criteo_log;

DROP TABLE criteo_log;

```

## Brown University Benchmark

MgBench is a new analytical benchmark for machine-generated log data, [Andrew Crotty](#).

Download the data:

```
 wget https://datasets.clickhouse.com/mgbench{1..3}.csv.xz
```

Unpack the data:

```
xz -v -d mgbench{1..3}.csv.xz
```

Create tables:

```
CREATE DATABASE mgbench;

CREATE TABLE mgbench.logs1 (
    log_time    DateTime,
    machine_name LowCardinality(String),
    machine_group LowCardinality(String),
    cpu_idle    Nullable(Float32),
    cpu_nice    Nullable(Float32),
    cpu_system  Nullable(Float32),
    cpu_user    Nullable(Float32),
    cpu_wio     Nullable(Float32),
    disk_free   Nullable(Float32),
    disk_total  Nullable(Float32),
    part_max_used Nullable(Float32),
    load_fifteen Nullable(Float32),
    load_five   Nullable(Float32),
    load_one    Nullable(Float32),
    mem_buffers Nullable(Float32),
    mem_cached  Nullable(Float32),
    mem_free    Nullable(Float32),
    mem_shared   Nullable(Float32),
    swap_free   Nullable(Float32),
    bytes_in    Nullable(Float32),
    bytes_out   Nullable(Float32)
)
ENGINE = MergeTree()
ORDER BY (machine_group, machine_name, log_time);

CREATE TABLE mgbench.logs2 (
    log_time    DateTime,
    client_ip   IPv4,
    request     String,
    status_code UInt16,
    object_size UInt64
)
ENGINE = MergeTree()
ORDER BY log_time;

CREATE TABLE mgbench.logs3 (
    log_time   DateTime64,
    device_id  FixedString(15),
    device_name LowCardinality(String),
    device_type LowCardinality(String),
    device_floor UInt8,
    event_type  LowCardinality(String),
    event_unit  FixedString(1),
    event_value Nullable(Float32)
)
ENGINE = MergeTree()
ORDER BY (event_type, log_time);
```

Insert data:

```
clickhouse-client --query "INSERT INTO mgbench.logs1 FORMAT CSVWithNames" < mgbench1.csv
clickhouse-client --query "INSERT INTO mgbench.logs2 FORMAT CSVWithNames" < mgbench2.csv
clickhouse-client --query "INSERT INTO mgbench.logs3 FORMAT CSVWithNames" < mgbench3.csv
```

Run benchmark queries:

-- Q1.1: What is the CPU/network utilization for each web server since midnight?

```
SELECT machine_name,
       MIN(cpu) AS cpu_min,
       MAX(cpu) AS cpu_max,
       AVG(cpu) AS cpu_avg,
       MIN(net_in) AS net_in_min,
       MAX(net_in) AS net_in_max,
       AVG(net_in) AS net_in_avg,
       MIN(net_out) AS net_out_min,
       MAX(net_out) AS net_out_max,
       AVG(net_out) AS net_out_avg
  FROM (
    SELECT machine_name,
           COALESCE(cpu_user, 0.0) AS cpu,
           COALESCE(bytes_in, 0.0) AS net_in,
           COALESCE(bytes_out, 0.0) AS net_out
      FROM logs1
     WHERE machine_name IN ('anansi','aragog','urd')
       AND log_time >= TIMESTAMP '2017-01-11 00:00:00'
  ) AS r
 GROUP BY machine_name;
```

-- Q1.2: Which computer lab machines have been offline in the past day?

```
SELECT machine_name,
       log_time
  FROM logs1
 WHERE (machine_name LIKE 'cslab%' OR
       machine_name LIKE 'mslab%')
   AND load_one IS NULL
   AND log_time >= TIMESTAMP '2017-01-10 00:00:00'
 ORDER BY machine_name,
          log_time;
```

-- Q1.3: What are the hourly average metrics during the past 10 days for a specific workstation?

```
SELECT dt,
       hr,
       AVG(load_fifteen) AS load_fifteen_avg,
       AVG(load_five) AS load_five_avg,
       AVG(load_one) AS load_one_avg,
       AVG(mem_free) AS mem_free_avg,
       AVG(swap_free) AS swap_free_avg
  FROM (
    SELECT CAST(log_time AS DATE) AS dt,
           EXTRACT(HOUR FROM log_time) AS hr,
           load_fifteen,
           load_five,
           load_one,
           mem_free,
           swap_free
      FROM logs1
     WHERE machine_name = 'babbage'
       AND load_fifteen IS NOT NULL
       AND load_five IS NOT NULL
       AND load_one IS NOT NULL
       AND mem_free IS NOT NULL
       AND swap_free IS NOT NULL
       AND log_time >= TIMESTAMP '2017-01-01 00:00:00'
  ) AS r
 GROUP BY dt,
          hr
 ORDER BY dt,
          hr;
```

-- Q1.4: Over 1 month, how often was each server blocked on disk I/O?

```
SELECT machine_name,
       COUNT(*) AS spikes
  FROM logs1
```

```

FROM logs1
WHERE machine_group = 'Servers'
AND cpu_wio > 0.99
AND log_time >= TIMESTAMP '2016-12-01 00:00:00'
AND log_time < TIMESTAMP '2017-01-01 00:00:00'
GROUP BY machine_name
ORDER BY spikes DESC
LIMIT 10;

```

-- Q1.5: Which externally reachable VMs have run low on memory?

```

SELECT machine_name,
       dt,
       MIN(mem_free) AS mem_free_min
FROM (
    SELECT machine_name,
           CAST(log_time AS DATE) AS dt,
           mem_free
    FROM logs1
    WHERE machine_group = 'DMZ'
      AND mem_free IS NOT NULL
) AS r
GROUP BY machine_name,
       dt
HAVING MIN(mem_free) < 10000
ORDER BY machine_name,
       dt;

```

-- Q1.6: What is the total hourly network traffic across all file servers?

```

SELECT dt,
       hr,
       SUM(net_in) AS net_in_sum,
       SUM(net_out) AS net_out_sum,
       SUM(net_in) + SUM(net_out) AS both_sum
FROM (
    SELECT CAST(log_time AS DATE) AS dt,
           EXTRACT(HOUR FROM log_time) AS hr,
           COALESCE(bytes_in, 0.0) / 1000000000.0 AS net_in,
           COALESCE(bytes_out, 0.0) / 1000000000.0 AS net_out
    FROM logs1
    WHERE machine_name IN ('allsorts','andes','bigred','blackjack','bonbon',
                           'cadbury','chiclets','cotton','crows','dove','fireball','hearts','huey',
                           'lindt','milkduds','milkyway','mnmm','necco','nerds','orbit','peeps',
                           'poprocks','razzles','runts','smarties','smuggler','spree','stride',
                           'tootsie','trident','wrigley','york')
) AS r
GROUP BY dt,
       hr
ORDER BY both_sum DESC
LIMIT 10;

```

-- Q2.1: Which requests have caused server errors within the past 2 weeks?

```

SELECT *
FROM logs2
WHERE status_code >= 500
  AND log_time >= TIMESTAMP '2012-12-18 00:00:00'
ORDER BY log_time;

```

-- Q2.2: During a specific 2-week period, was the user password file leaked?

```

SELECT *
FROM logs2
WHERE status_code >= 200
  AND status_code < 300
  AND request LIKE '%/etc/passwd%'
  AND log_time >= TIMESTAMP '2012-05-06 00:00:00'
  AND log_time < TIMESTAMP '2012-05-20 00:00:00';

```

-- Q2.3: What was the average path depth for top-level requests in the past month?

~~SELECT top\_level~~

```

SELECT top_level,
       AVG(LENGTH(request) - LENGTH(REPLACE(request, '/', ''))) AS depth_avg
FROM (
    SELECT SUBSTRING(request FROM 1 FOR len) AS top_level,
           request
    FROM (
        SELECT POSITION(SUBSTRING(request FROM 2), '/') AS len,
               request
        FROM logs2
        WHERE status_code >= 200
          AND status_code < 300
          AND log_time >= TIMESTAMP '2012-12-01 00:00:00'
    ) AS r
    WHERE len > 0
) AS s
WHERE top_level IN ('/about','/courses','/degrees','/events',
                     '/grad','/industry','/news','/people',
                     '/publications','/research','/teaching','/ugrad')
GROUP BY top_level
ORDER BY top_level;

```

-- Q2.4: During the last 3 months, which clients have made an excessive number of requests?

```

SELECT client_ip,
       COUNT(*) AS num_requests
FROM logs2
WHERE log_time >= TIMESTAMP '2012-10-01 00:00:00'
GROUP BY client_ip
HAVING COUNT(*) >= 100000
ORDER BY num_requests DESC;

```

-- Q2.5: What are the daily unique visitors?

```

SELECT dt,
       COUNT(DISTINCT client_ip)
FROM (
    SELECT CAST(log_time AS DATE) AS dt,
           client_ip
    FROM logs2
) AS r
GROUP BY dt
ORDER BY dt;

```

-- Q2.6: What are the average and maximum data transfer rates (Gbps)?

```

SELECT AVG(transfer) / 125000000.0 AS transfer_avg,
       MAX(transfer) / 125000000.0 AS transfer_max
FROM (
    SELECT log_time,
           SUM(object_size) AS transfer
    FROM logs2
    GROUP BY log_time
) AS r;

```

-- Q3.1: Did the indoor temperature reach freezing over the weekend?

```

SELECT *
FROM logs3
WHERE event_type = 'temperature'
  AND event_value <= 32.0
  AND log_time >= '2019-11-29 17:00:00.000';

```

-- Q3.4: Over the past 6 months, how frequently were each door opened?

```

SELECT device_name,
       device_floor,
       COUNT(*) AS ct
FROM logs3
WHERE event_type = 'door_open'
  AND log_time >= '2019-06-01 00:00:00.000'
GROUP BY device_name,
       device_floor
ORDER BY ct DESC;

```

```
ORDER BY ct DESC,
```

-- Q3.5: Where in the building do large temperature variations occur in winter and summer?

```
WITH temperature AS (
  SELECT dt,
    device_name,
    device_type,
    device_floor
  FROM (
    SELECT dt,
      hr,
      device_name,
      device_type,
      device_floor,
      AVG(event_value) AS temperature_hourly_avg
    FROM (
      SELECT CAST(log_time AS DATE) AS dt,
        EXTRACT(HOUR FROM log_time) AS hr,
        device_name,
        device_type,
        device_floor,
        event_value
      FROM logs3
      WHERE event_type = 'temperature'
    ) AS r
    GROUP BY dt,
      hr,
      device_name,
      device_type,
      device_floor
  ) AS s
  GROUP BY dt,
    device_name,
    device_type,
    device_floor
  HAVING MAX(temperature_hourly_avg) - MIN(temperature_hourly_avg) >= 25.0
)
SELECT DISTINCT device_name,
  device_type,
  device_floor,
  'WINTER'
FROM temperature
WHERE dt >= DATE '2018-12-01'
  AND dt < DATE '2019-03-01'
UNION
SELECT DISTINCT device_name,
  device_type,
  device_floor,
  'SUMMER'
FROM temperature
WHERE dt >= DATE '2019-06-01'
  AND dt < DATE '2019-09-01';
```

-- Q3.6: For each device category, what are the monthly power consumption metrics?

```
SELECT yr,
  mo,
  SUM(coffee_hourly_avg) AS coffee_monthly_sum,
  AVG(coffee_hourly_avg) AS coffee_monthly_avg,
  SUM(printer_hourly_avg) AS printer_monthly_sum,
  AVG(printer_hourly_avg) AS printer_monthly_avg,
  SUM(projector_hourly_avg) AS projector_monthly_sum,
  AVG(projector_hourly_avg) AS projector_monthly_avg,
  SUM(vending_hourly_avg) AS vending_monthly_sum,
  AVG(vending_hourly_avg) AS vending_monthly_avg
FROM (
  SELECT dt,
    yr,
    mo,
    hr,
    AVG(coffee) AS coffee_hourly_avg,
    AVG(printer) AS printer_hourly_avg,
    AVG(projector) AS projector_hourly_avg,
    AVG(vending) AS vending_hourly_avg
  FROM /
```

```

FROM (
  SELECT CAST(log_time AS DATE) AS dt,
    EXTRACT(YEAR FROM log_time) AS yr,
    EXTRACT(MONTH FROM log_time) AS mo,
    EXTRACT(HOUR FROM log_time) AS hr,
    CASE WHEN device_name LIKE 'coffee%' THEN event_value END AS coffee,
    CASE WHEN device_name LIKE 'printer%' THEN event_value END AS printer,
    CASE WHEN device_name LIKE 'projector%' THEN event_value END AS projector,
    CASE WHEN device_name LIKE 'vending%' THEN event_value END AS vending
  FROM logs3
  WHERE device_type = 'meter'
) AS r
GROUP BY dt,
  yr,
  mo,
  hr
) AS s
GROUP BY yr,
  mo
ORDER BY yr,
  mo;

```

The data is also available for interactive queries in the [Playground](#), example.

## Crowdsourced air traffic data from The OpenSky Network 2020

"The data in this dataset is derived and cleaned from the full OpenSky dataset to illustrate the development of air traffic during the COVID-19 pandemic. It spans all flights seen by the network's more than 2500 members since 1 January 2019. More data will be periodically included in the dataset until the end of the COVID-19 pandemic".

Source: <https://zenodo.org/record/5092942#.YRBCyTpRXYd>

Martin Strohmeier, Xavier Olive, Jannis Lübbe, Matthias Schäfer, and Vincent Lenders  
 "Crowdsourced air traffic data from the OpenSky Network 2019-2020"  
*Earth System Science Data* 13(2), 2021  
<https://doi.org/10.5194/essd-13-357-2021>

## Download the Dataset

Run the command:

```

wget -O- https://zenodo.org/record/5092942 | grep -oP
'https://zenodo.org/record/5092942/files/flightlist_\d+\_\d+\_.csv\.gz' | xargs wget

```

Download will take about 2 minutes with good internet connection. There are 30 files with total size of 4.3 GB.

## Create the Table

```

CREATE TABLE opensky
(
    callsign String,
    number String,
    icao24 String,
    registration String,
    typecode String,
    origin String,
    destination String,
    firstseen DateTime,
    lastseen DateTime,
    day DateTime,
    latitude_1 Float64,
    longitude_1 Float64,
    altitude_1 Float64,
    latitude_2 Float64,
    longitude_2 Float64,
    altitude_2 Float64
) ENGINE = MergeTree ORDER BY (origin, destination, callsign);

```

## Import Data

Upload data into ClickHouse in parallel:

```
ls -1 flightlist_*.csv.gz | xargs -P100 -I{} bash -c 'gzip -c -d "{}" | clickhouse-client --date_time_input_format best_effort --query "INSERT INTO opensky FORMAT CSVWithNames"'
```

- Here we pass the list of files (`ls -1 flightlist_*.csv.gz`) to `xargs` for parallel processing. `xargs -P100` specifies to use up to 100 parallel workers but as we only have 30 files, the number of workers will be only 30.
- For every file, `xargs` will run a script with `bash -c`. The script has substitution in form of `{}` and the `xargs` command will substitute the filename to it (we have asked it for `xargs` with `-I{}`).
- The script will decompress the file (`gzip -c -d "{}"`) to standard output (`-c` parameter) and the output is redirected to `clickhouse-client`.
- We also asked to parse `DateTime` fields with extended parser (`--date_time_input_format best_effort`) to recognize ISO-8601 format with timezone offsets.

Finally, `clickhouse-client` will do insertion. It will read input data in `CSVWithNames` format.

Parallel upload takes 24 seconds.

If you don't like parallel upload, here is sequential variant:

```
for file in flightlist_*.csv.gz; do gzip -c -d "$file" | clickhouse-client --date_time_input_format best_effort --query "INSERT INTO opensky FORMAT CSVWithNames"; done
```

## Validate the Data

Query:

```
SELECT count() FROM opensky;
```

Result:

count()
66010819

The size of dataset in ClickHouse is just 2.66 GiB, check it.

Query:

```
SELECT formatReadableSize(total_bytes) FROM system.tables WHERE name = 'opensky';
```

Result:

```
formatReadableSize(total_bytes)─  
2.66 GiB |
```

## Run Some Queries

Total distance travelled is 68 billion kilometers.

Query:

```
SELECT formatReadableQuantity(sum(geoDistance(longitude_1, latitude_1, longitude_2, latitude_2)) / 1000) FROM opensky;
```

Result:

```
formatReadableQuantity(divide(sum(geoDistance(longitude_1, latitude_1, longitude_2, latitude_2)), 1000))─  
68.72 billion |
```

Average flight distance is around 1000 km.

Query:

```
SELECT avg(geoDistance(longitude_1, latitude_1, longitude_2, latitude_2)) FROM opensky;
```

Result:

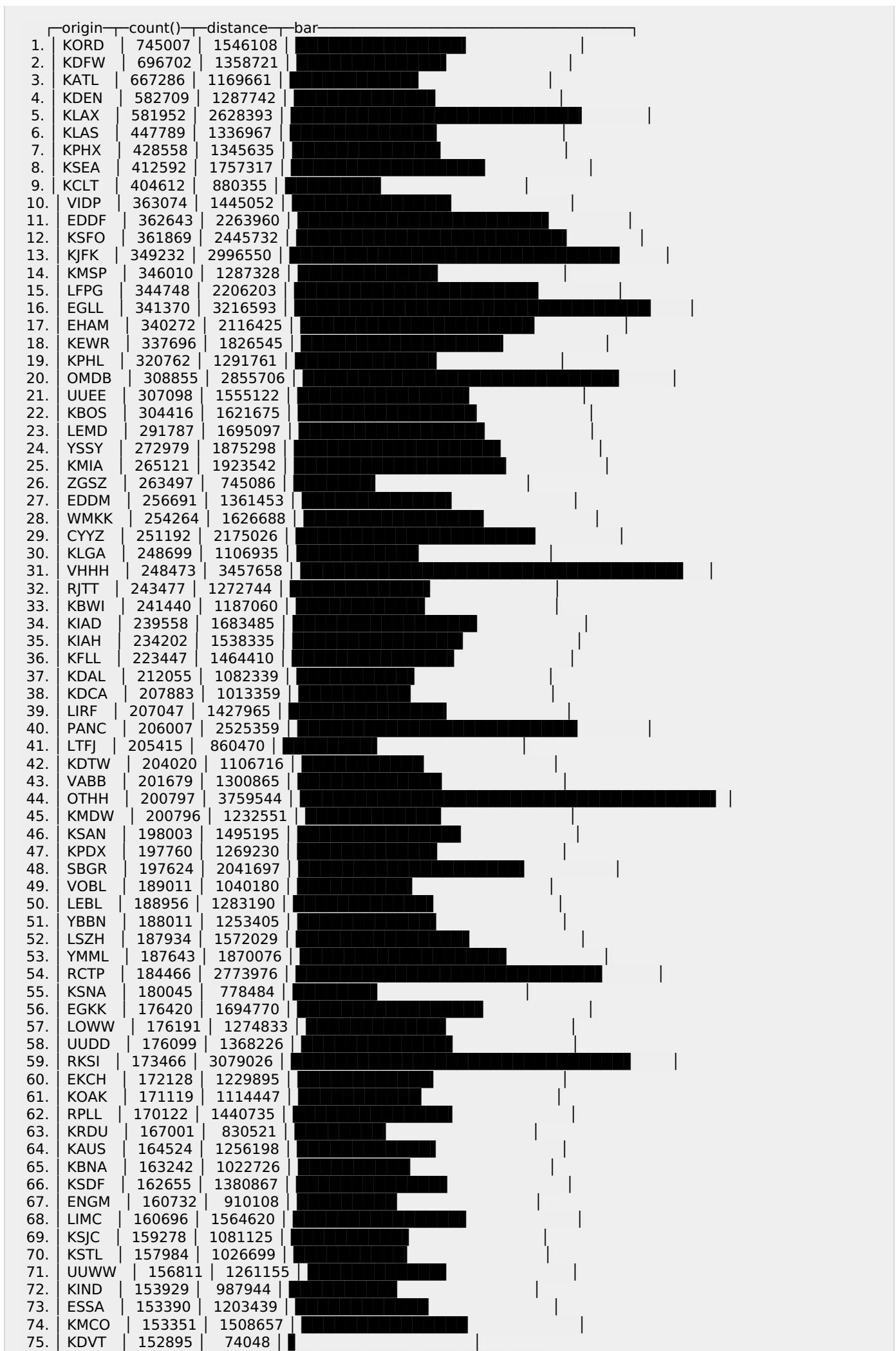
```
avg(geoDistance(longitude_1, latitude_1, longitude_2, latitude_2))─  
1041090.6465708319 |
```

## Most busy origin airports and the average distance seen

Query:

```
SELECT  
origin,  
count(),  
round(avg(geoDistance(longitude_1, latitude_1, longitude_2, latitude_2))) AS distance,  
bar(distance, 0, 10000000, 100) AS bar  
FROM opensky  
WHERE origin != ''  
GROUP BY origin  
ORDER BY count() DESC  
LIMIT 100;
```

Result:



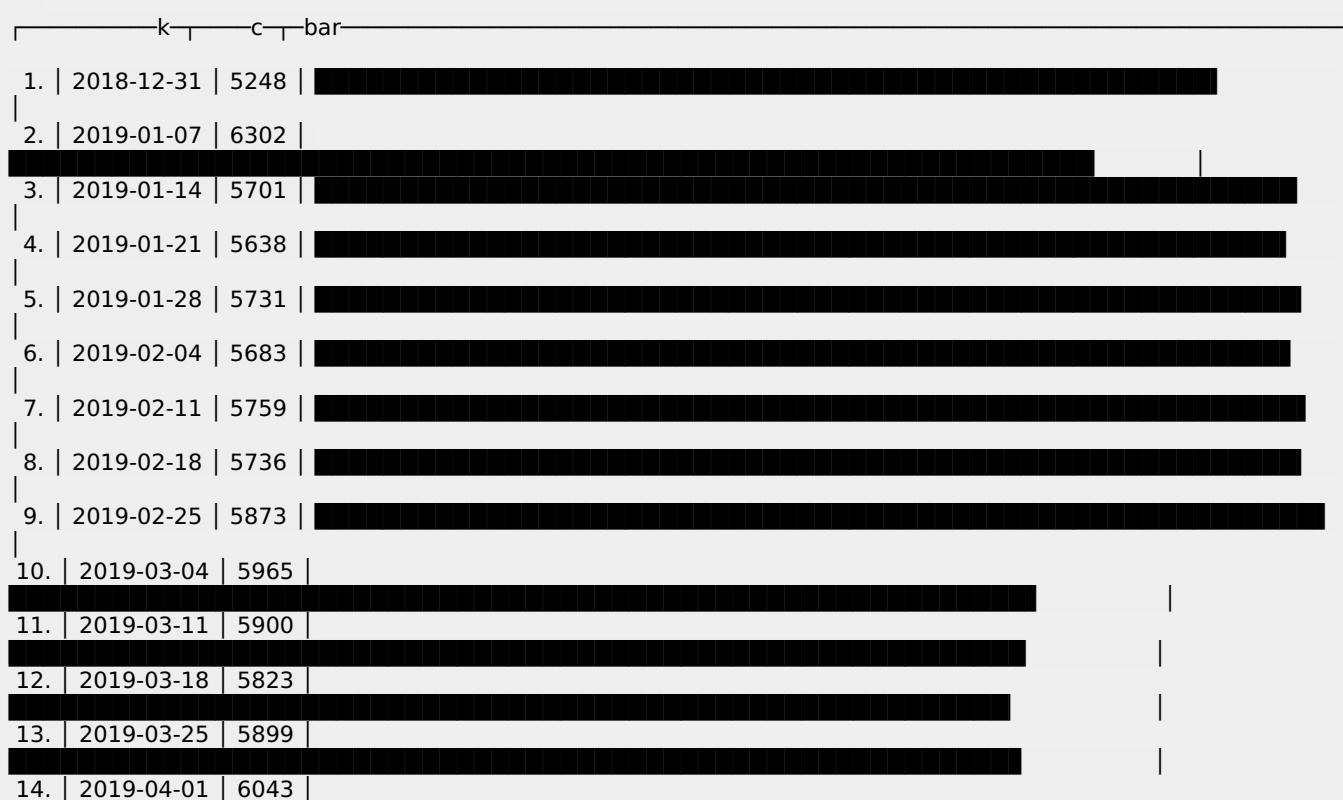
76.	VTBS	152645	2255591	[REDACTED]	[REDACTED]	[REDACTED]
77.	CYVR	149574	2027413	[REDACTED]	[REDACTED]	[REDACTED]
78.	EIDW	148723	1503985	[REDACTED]	[REDACTED]	[REDACTED]
79.	LFPO	143277	1152964	[REDACTED]	[REDACTED]	[REDACTED]
80.	EGSS	140830	1348183	[REDACTED]	[REDACTED]	[REDACTED]
81.	KAPA	140776	420441	[REDACTED]	[REDACTED]	[REDACTED]
82.	KHOU	138985	1068806	[REDACTED]	[REDACTED]	[REDACTED]
83.	KTPA	138033	1338223	[REDACTED]	[REDACTED]	[REDACTED]
84.	KFFZ	137333	55397	[REDACTED]	[REDACTED]	[REDACTED]
85.	NZAA	136092	1581264	[REDACTED]	[REDACTED]	[REDACTED]
86.	YPPH	133916	1271550	[REDACTED]	[REDACTED]	[REDACTED]
87.	RJBB	133522	1805623	[REDACTED]	[REDACTED]	[REDACTED]
88.	EDDL	133018	1265919	[REDACTED]	[REDACTED]	[REDACTED]
89.	ULLI	130501	1197108	[REDACTED]	[REDACTED]	[REDACTED]
90.	KIWA	127195	250876	[REDACTED]	[REDACTED]	[REDACTED]
91.	KTEB	126969	1189414	[REDACTED]	[REDACTED]	[REDACTED]
92.	VOMM	125616	1127757	[REDACTED]	[REDACTED]	[REDACTED]
93.	LSGG	123998	1049101	[REDACTED]	[REDACTED]	[REDACTED]
94.	LPPT	122733	1779187	[REDACTED]	[REDACTED]	[REDACTED]
95.	WSSS	120493	3264122	[REDACTED]	[REDACTED]	[REDACTED]
96.	EBBR	118539	1579939	[REDACTED]	[REDACTED]	[REDACTED]
97.	VTBD	118107	661627	[REDACTED]	[REDACTED]	[REDACTED]
98.	KVNY	116326	692960	[REDACTED]	[REDACTED]	[REDACTED]
99.	EDDT	115122	941740	[REDACTED]	[REDACTED]	[REDACTED]
100.	EFHK	114860	1629143	[REDACTED]	[REDACTED]	[REDACTED]

## Number of flights from three major Moscow airports, weekly

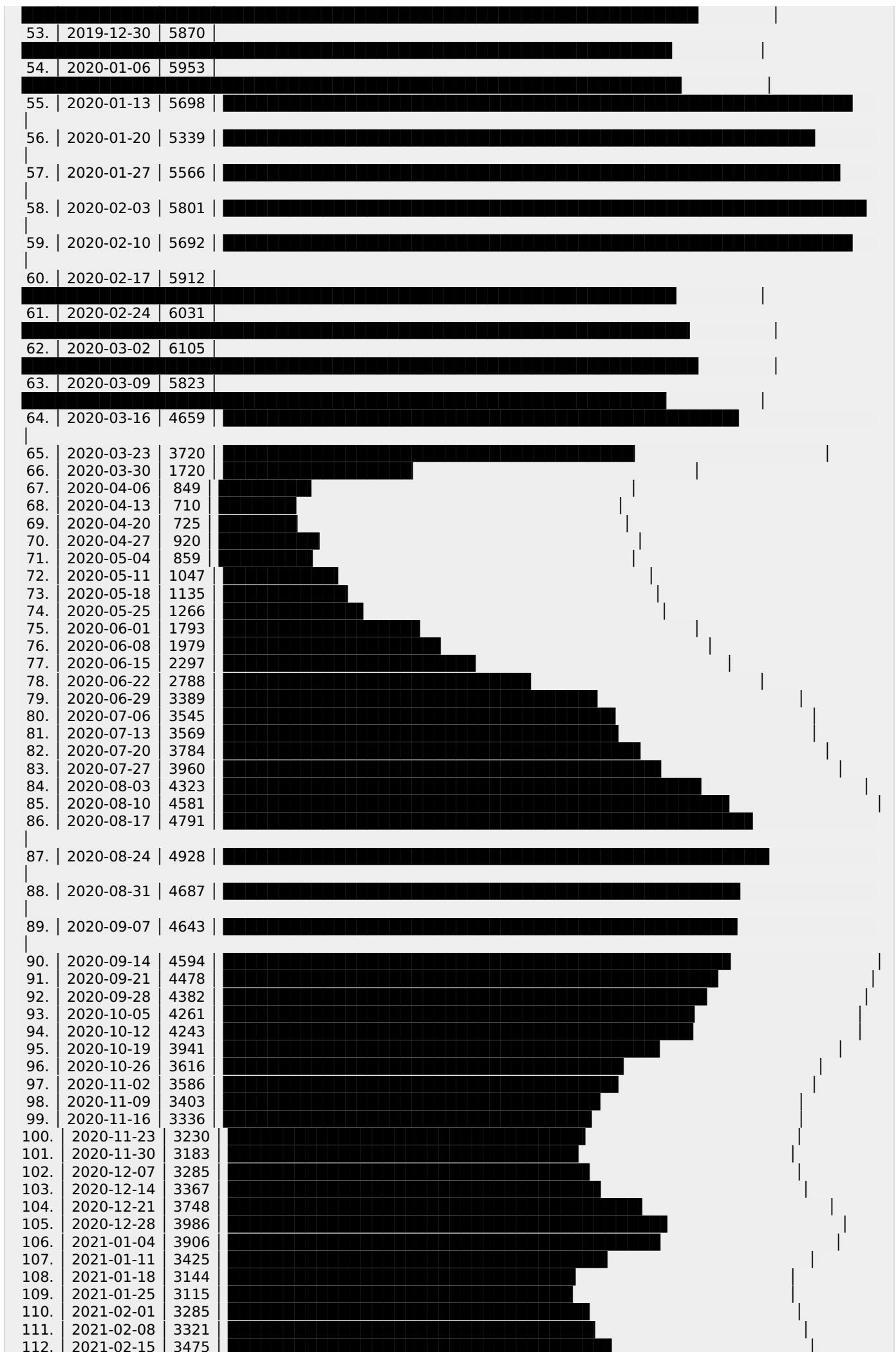
Query:

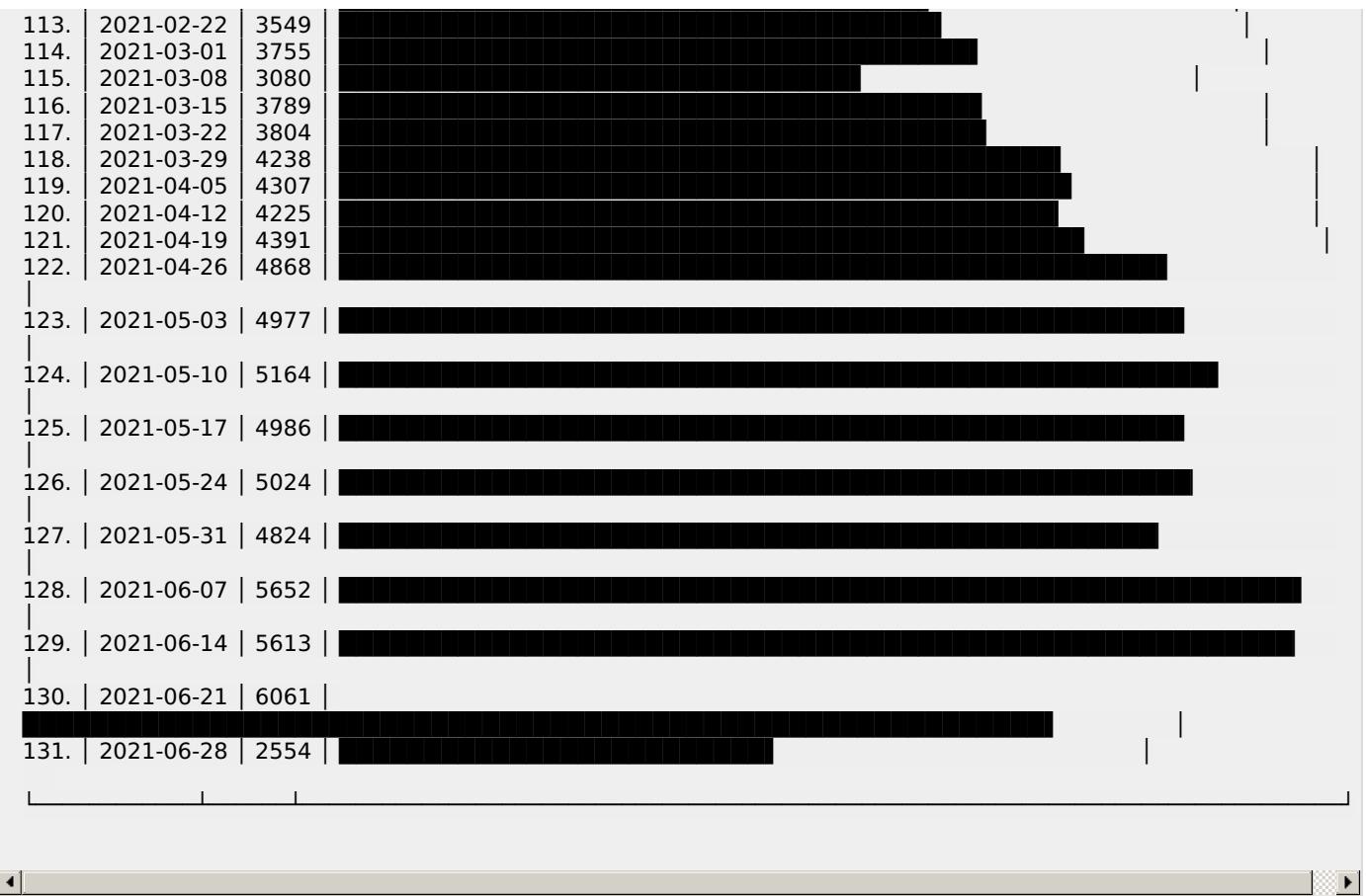
```
SELECT
    toMonday(day) AS k,
    count() AS c,
    bar(c, 0, 10000, 100) AS bar
FROM opensky
WHERE origin IN ('UUUE', 'UUDD', 'UUWW')
GROUP BY k
ORDER BY k ASC;
```

Result:



15.	2019-04-08	6098
16.	2019-04-15	6196
17.	2019-04-22	6486
18.	2019-04-29	6682
19.	2019-05-06	6739
20.	2019-05-13	6600
21.	2019-05-20	6575
22.	2019-05-27	6786
23.	2019-06-03	6872
24.	2019-06-10	7045
25.	2019-06-17	7045
26.	2019-06-24	6852
27.	2019-07-01	7248
28.	2019-07-08	7284
29.	2019-07-15	7142
30.	2019-07-22	7108
31.	2019-07-29	7251
32.	2019-08-05	7403
33.	2019-08-12	7457
34.	2019-08-19	7502
35.	2019-08-26	7540
36.	2019-09-02	7237
37.	2019-09-09	7328
38.	2019-09-16	5566
39.	2019-09-23	7049
40.	2019-09-30	6880
41.	2019-10-07	6518
42.	2019-10-14	6688
43.	2019-10-21	6667
44.	2019-10-28	6303
45.	2019-11-04	6298
46.	2019-11-11	6137
47.	2019-11-18	6051
48.	2019-11-25	5820
49.	2019-12-02	5942
50.	2019-12-09	4891
51.	2019-12-16	5682
52.	2019-12-23	6111





## Online Playground

You can test other queries to this data set using the interactive resource [Online Playground](#). For example, [like this](#). However, please note that you cannot create temporary tables here.

## UK Property Price Paid

The dataset contains data about prices paid for real-estate property in England and Wales. The data is available since year 1995.

The size of the dataset in uncompressed form is about 4 GiB and it will take about 278 MiB in ClickHouse.

Source: <https://www.gov.uk/government/statistical-data-sets/price-paid-data-downloads>

Description of the fields: <https://www.gov.uk/guidance/about-the-price-paid-data>

Contains HM Land Registry data © Crown copyright and database right 2021. This data is licensed under the Open Government Licence v3.0.

## Download the Dataset

Run the command:

```
wget http://prod.publicdata.landregistry.gov.uk.s3-website-eu-west-1.amazonaws.com/pp-complete.csv
```

Download will take about 2 minutes with good internet connection.

## Create the Table

```
CREATE TABLE uk_price_paid
(
    price UInt32,
    date Date,
    postcode1 LowCardinality(String),
    postcode2 LowCardinality(String),
    type Enum8('terraced' = 1, 'semi-detached' = 2, 'detached' = 3, 'flat' = 4, 'other' = 0),
    is_new UInt8,
    duration Enum8('freehold' = 1, 'leasehold' = 2, 'unknown' = 0),
    addr1 String,
    addr2 String,
    street LowCardinality(String),
    locality LowCardinality(String),
    town LowCardinality(String),
    district LowCardinality(String),
    county LowCardinality(String),
    category UInt8
) ENGINE = MergeTree ORDER BY (postcode1, postcode2, addr1, addr2);
```

## Preprocess and Import Data

We will use `clickhouse-local` tool for data preprocessing and `clickhouse-client` to upload it.

In this example, we define the structure of source data from the CSV file and specify a query to preprocess the data with `clickhouse-local`.

The preprocessing is:

- splitting the postcode to two different columns `postcode1` and `postcode2` that is better for storage and queries;
- converting the time field to date as it only contains 00:00 time;
- ignoring the `UUid` field because we don't need it for analysis;
- transforming `type` and `duration` to more readable Enum fields with function `transform`;
- transforming `is_new` and `category` fields from single-character string (Y/N and A/B) to `UInt8` field with 0 and 1.

Preprocessed data is piped directly to `clickhouse-client` to be inserted into ClickHouse table in streaming fashion.

```

clickhouse-local --input-format CSV --structure '
    uuid String,
    price UInt32,
    time DateTime,
    postcode String,
    a String,
    b String,
    c String,
    addr1 String,
    addr2 String,
    street String,
    locality String,
    town String,
    district String,
    county String,
    d String,
    e String
' --query "
WITH splitByChar(' ', postcode) AS p
SELECT
    price,
    toDate(time) AS date,
    p[1] AS postcode1,
    p[2] AS postcode2,
    transform(a, ['T', 'S', 'D', 'F', 'O'], ['terraced', 'semi-detached', 'detached', 'flat', 'other']) AS type,
    b = 'Y' AS is_new,
    transform(c, ['F', 'L', 'U'], ['freehold', 'leasehold', 'unknown']) AS duration,
    addr1,
    addr2,
    street,
    locality,
    town,
    district,
    county,
    d = 'B' AS category
FROM table" --date_time input_format best_effort < pp-complete.csv | clickhouse-client --query "INSERT INTO
uk_price_paid FORMAT TSV"

```

It will take about 40 seconds.

## Validate the Data

Query:

```
SELECT count() FROM uk_price_paid;
```

Result:

count()
26321785

The size of dataset in ClickHouse is just 278 MiB, check it.

Query:

```
SELECT formatReadableSize(total_bytes) FROM system.tables WHERE name = 'uk_price_paid';
```

Result:

formatReadableSize(total_bytes)
278.80 MiB

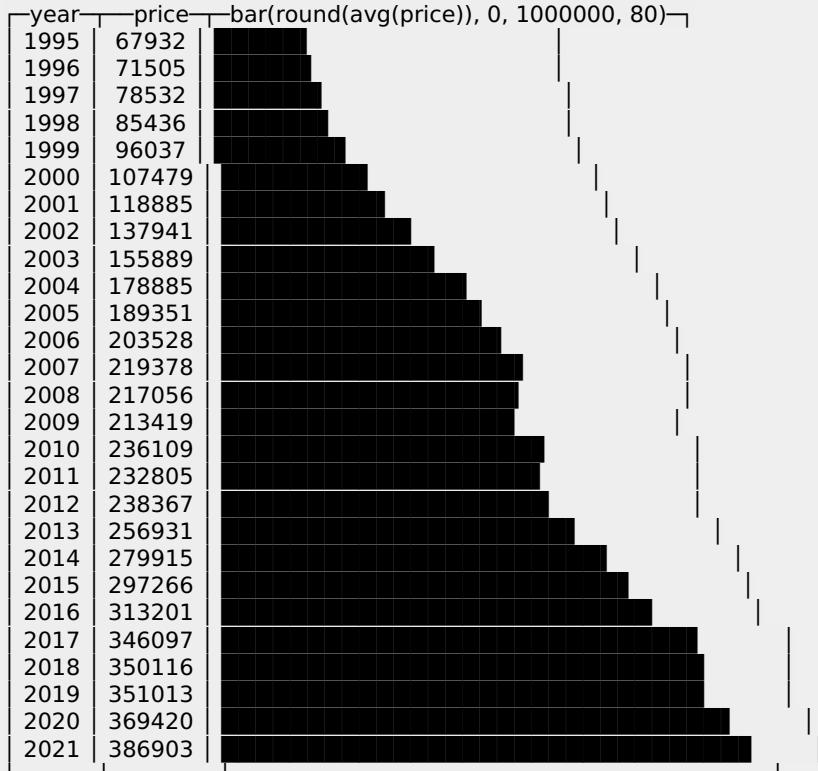
# Run Some Queries

## Query 1. Average Price Per Year

Query:

```
SELECT toYear(date) AS year, round(avg(price)) AS price, bar(price, 0, 1000000, 80) FROM uk_price_paid GROUP BY year ORDER BY year;
```

Result:

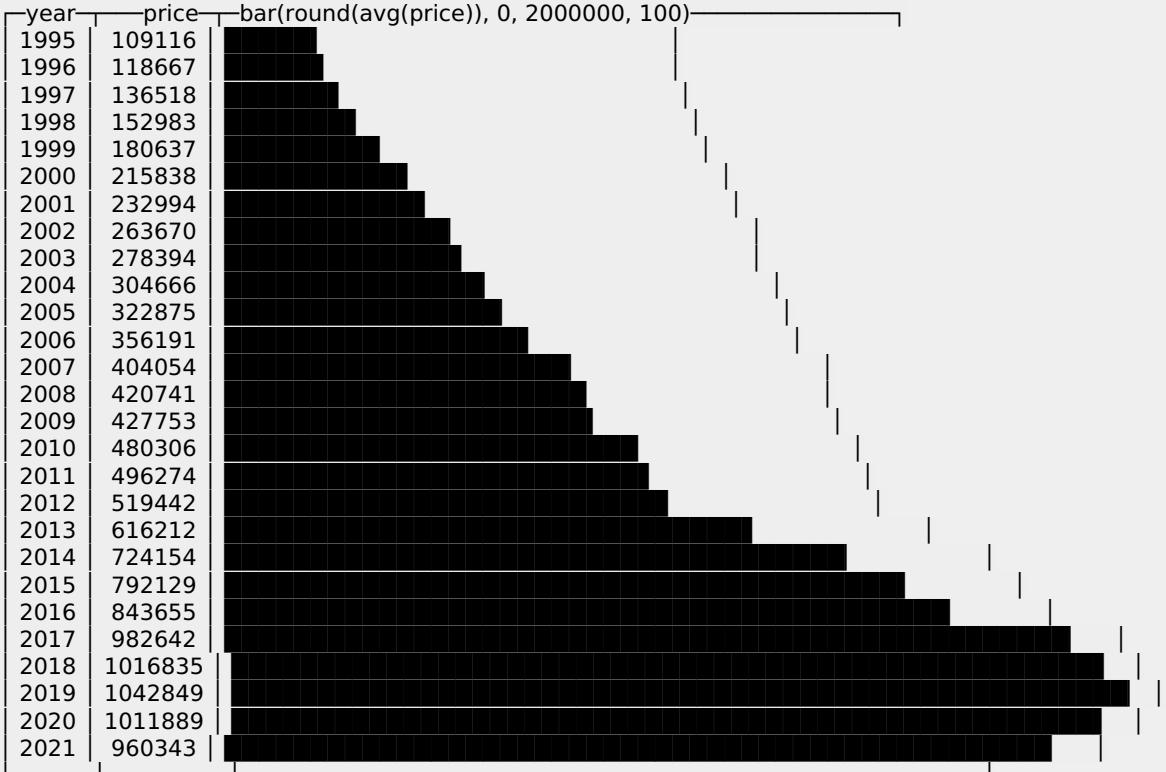


## Query 2. Average Price per Year in London

Query:

```
SELECT toYear(date) AS year, round(avg(price)) AS price, bar(price, 0, 2000000, 100) FROM uk_price_paid WHERE town = 'LONDON' GROUP BY year ORDER BY year;
```

Result:



Something happened in 2013. I don't have a clue. Maybe you have a clue what happened in 2020?

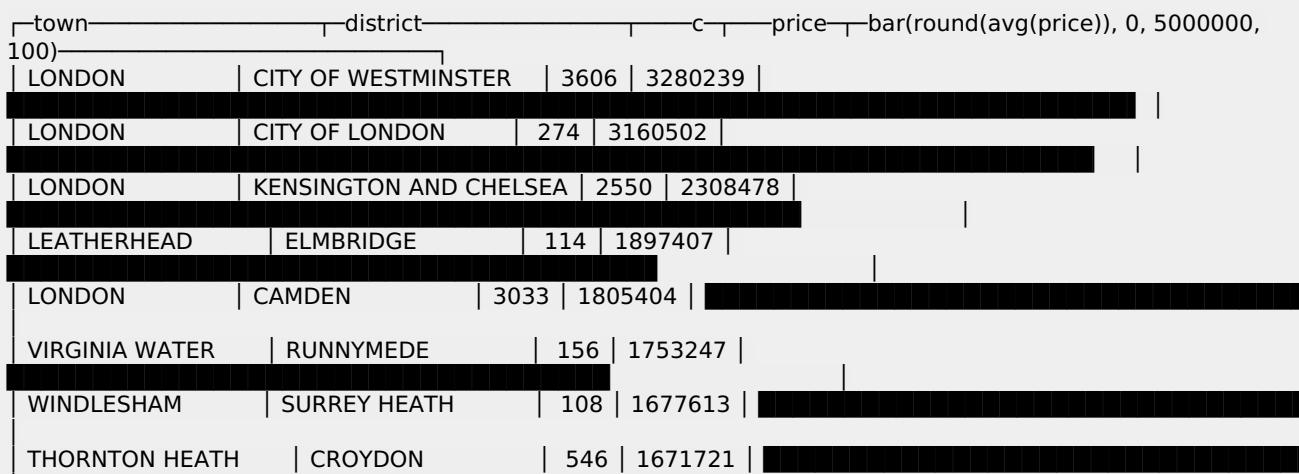
### Query 3. The Most Expensive Neighborhoods

Query:

```

SELECT
  town,
  district,
  count() AS c,
  round(avg(price)) AS price,
  bar(price, 0, 5000000, 100)
FROM uk_price_paid
WHERE date >= '2020-01-01'
GROUP BY
  town,
  district
HAVING c >= 100
ORDER BY price DESC
LIMIT 100;
  
```

Result:



BARNET	ENFIELD	124   1505840   [REDACTED]
COBHAM	ELMBRIDGE	387   1237250   [REDACTED]
LONDON	ISLINGTON	2668   1236980   [REDACTED]
OXFORD	SOUTH OXFORDSHIRE	321   1220907   [REDACTED]
LONDON	RICHMOND UPON THAMES	704   1215551   [REDACTED]
LONDON	HOUNSLOW	671   1207493   [REDACTED]
ASCOT	WINDSOR AND MAIDENHEAD	407   1183299   [REDACTED]
BEACONSFIELD	BUCKINGHAMSHIRE	330   1175615   [REDACTED]
RICHMOND	RICHMOND UPON THAMES	874   1110444   [REDACTED]
LONDON	HAMMERSMITH AND FULHAM	3086   1053983   [REDACTED]
SURBITON	ELMBRIDGE	100   1011800   [REDACTED]
RADLETT	HERTSMERE	283   1011712   [REDACTED]
SALCOMBE	SOUTH HAMS	127   1011624   [REDACTED]
WEYBRIDGE	ELMBRIDGE	655   1007265   [REDACTED]
ESHER	ELMBRIDGE	485   986581   [REDACTED]
LEATHERHEAD	GUILDFORD	202   977320   [REDACTED]
BURFORD	WEST OXFORDSHIRE	111   966893   [REDACTED]
BROCKENHURST	NEW FOREST	129   956675   [REDACTED]
HINDHEAD	WAVERLEY	137   953753   [REDACTED]
GERRARDS CROSS	BUCKINGHAMSHIRE	419   951121   [REDACTED]
EAST MOLESEY	ELMBRIDGE	192   936769   [REDACTED]
CHALFONT ST GILES	BUCKINGHAMSHIRE	146   925515   [REDACTED]
LONDON	TOWER HAMLETS	4388   918304   [REDACTED]
OLNEY	MILTON KEYNES	235   910646   [REDACTED]
HENLEY-ON-THAMES	SOUTH OXFORDSHIRE	540   902418   [REDACTED]
LONDON	SOUTHWARK	3885   892997   [REDACTED]
KINGSTON UPON THAMES	KINGSTON UPON THAMES	960   885969   [REDACTED]
LONDON	EALING	2658   871755   [REDACTED]
CRANBROOK	TUNBRIDGE WELLS	431   862348   [REDACTED]
LONDON	MERTON	2099   859118   [REDACTED]
BELVEDERE	BEXLEY	346   842423   [REDACTED]
GUILDFORD	WAVERLEY	143   841277   [REDACTED]
HARPENDEN	ST ALBANS	657   841216   [REDACTED]
LONDON	HACKNEY	3307   837090   [REDACTED]
LONDON	WANDSWORTH	6566   832663   [REDACTED]
MAIDENHEAD	BUCKINGHAMSHIRE	123   824299   [REDACTED]
KINGS LANGLEY	DACORUM	145   821331   [REDACTED]
BERKHAMSTED	DACORUM	543   818415   [REDACTED]
GREAT MISSENDEN	BUCKINGHAMSHIRE	226   802807   [REDACTED]
BILLINGSHURST	CHICHESTER	144   797829   [REDACTED]
WOKING	GUILDFORD	176   793494   [REDACTED]
STOCKBRIDGE	TEST VALLEY	178   793269   [REDACTED]
EPSOM	REIGATE AND BANSTEAD	172   791862   [REDACTED]

TONBRIDGE	TUNBRIDGE WELLS	360	787876	[REDACTED]
TEDDINGTON	RICHMOND UPON THAMES	595	786492	[REDACTED]
TWICKENHAM	RICHMOND UPON THAMES	1155	786193	[REDACTED]
LYNDHURST	NEW FOREST	102	785593	[REDACTED]
LONDON	LAMBETH	5228	774574	[REDACTED]
LONDON	BARNET	3955	773259	[REDACTED]
OXFORD	VALE OF WHITE HORSE	353	772088	[REDACTED]
TONBRIDGE	MAIDSTONE	305	770740	[REDACTED]
LUTTERWORTH	HARBOROUGH	538	768634	[REDACTED]
WOODSTOCK	WEST OXFORDSHIRE	140	766037	[REDACTED]
MIDHURST	CHICHESTER	257	764815	[REDACTED]
MARLOW	BUCKINGHAMSHIRE	327	761876	[REDACTED]
LONDON	NEWHAM	3237	761784	[REDACTED]
ALDERLEY EDGE	CHESHIRE EAST	178	757318	[REDACTED]
LUTON	CENTRAL BEDFORDSHIRE	212	754283	[REDACTED]
PETWORTH	CHICHESTER	154	754220	[REDACTED]
ALRESFORD	WINCHESTER	219	752718	[REDACTED]
POTTERS BAR	WELWYN HATFIELD	174	748465	[REDACTED]
HASLEMERE	CHICHESTER	128	746907	[REDACTED]
TADWORTH	REIGATE AND BANSTEAD	502	743252	[REDACTED]
THAMES DITTON	ELMBRIDGE	244	741913	[REDACTED]
REIGATE	REIGATE AND BANSTEAD	581	738198	[REDACTED]
BOURNE END	BUCKINGHAMSHIRE	138	735190	[REDACTED]
SEVENOAKS	SEVENOAKS	1156	730018	[REDACTED]
OXTED	TANDRIDGE	336	729123	[REDACTED]
INGATESTONE	BRENTWOOD	166	728103	[REDACTED]
LONDON	BRENT	2079	720605	[REDACTED]
LONDON	HARINGEY	3216	717780	[REDACTED]
PURLEY	CROYDON	575	716108	[REDACTED]
WELWYN	WELWYN HATFIELD	222	710603	[REDACTED]
RICKMANSWORTH	THREE RIVERS	798	704571	[REDACTED]
BANSTEAD	REIGATE AND BANSTEAD	401	701293	[REDACTED]
CHIGWELL	EPPING FOREST	261	701203	[REDACTED]
PINNER	HARROW	528	698885	[REDACTED]
HASLEMERE	WAVERLEY	280	696659	[REDACTED]
SLOUGH	BUCKINGHAMSHIRE	396	694917	[REDACTED]
WALTON-ON-THAMES	ELMBRIDGE	946	692395	[REDACTED]
READING	SOUTH OXFORDSHIRE	318	691988	[REDACTED]
NORTHWOOD	HILLINGDON	271	690643	[REDACTED]
FELTHAM	HOUNSLAW	763	688595	[REDACTED]
ASHTEAD	MOLE VALLEY	303	687923	[REDACTED]
BARNET	BARNET	975	686980	[REDACTED]
WOKING	SURREY HEATH	283	686669	[REDACTED]
MALMESBURY	WILTSHIRE	323	683324	[REDACTED]
AMERSHAM	BUCKINGHAMSHIRE	496	680962	[REDACTED]
CHISLEHURST	BROMLEY	430	680209	[REDACTED]
HYTHE	FOLKESTONE AND HYTHE	490	676908	[REDACTED]
MAYFIELD	WEALDEN	101	676210	[REDACTED]
ASCOT	BRACKNELL FOREST	168	676004	[REDACTED]



# Let's Speed Up Queries Using Projections

Projections allow to improve queries speed by storing pre-aggregated data.

## Build a Projection

Create an aggregate projection by dimensions `toYear(date)`, `district`, `town`:

```
ALTER TABLE uk_price_paid
  ADD PROJECTION projection_by_year_district_town
(
  SELECT
    toYear(date),
    district,
    town,
    avg(price),
    sum(price),
    count()
  GROUP BY
    toYear(date),
    district,
    town
);
```

Populate the projection for existing data (without it projection will be created for only newly inserted data):

```
ALTER TABLE uk_price_paid
  MATERIALIZE PROJECTION projection_by_year_district_town
  SETTINGS mutations_sync = 1;
```

## Test Performance

Let's run the same 3 queries.

Enable projections for selects:

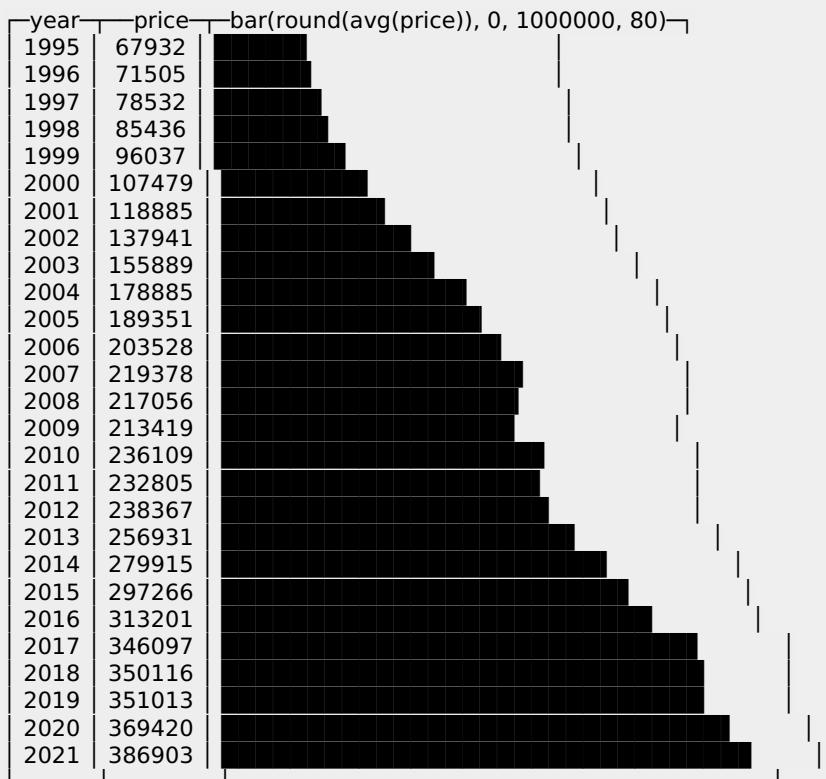
```
SET allow_experimental_projection_optimization = 1;
```

### Query 1. Average Price Per Year

Query:

```
SELECT
  toYear(date) AS year,
  round(avg(price)) AS price,
  bar(price, 0, 1000000, 80)
FROM uk_price_paid
GROUP BY year
ORDER BY year ASC;
```

Result:

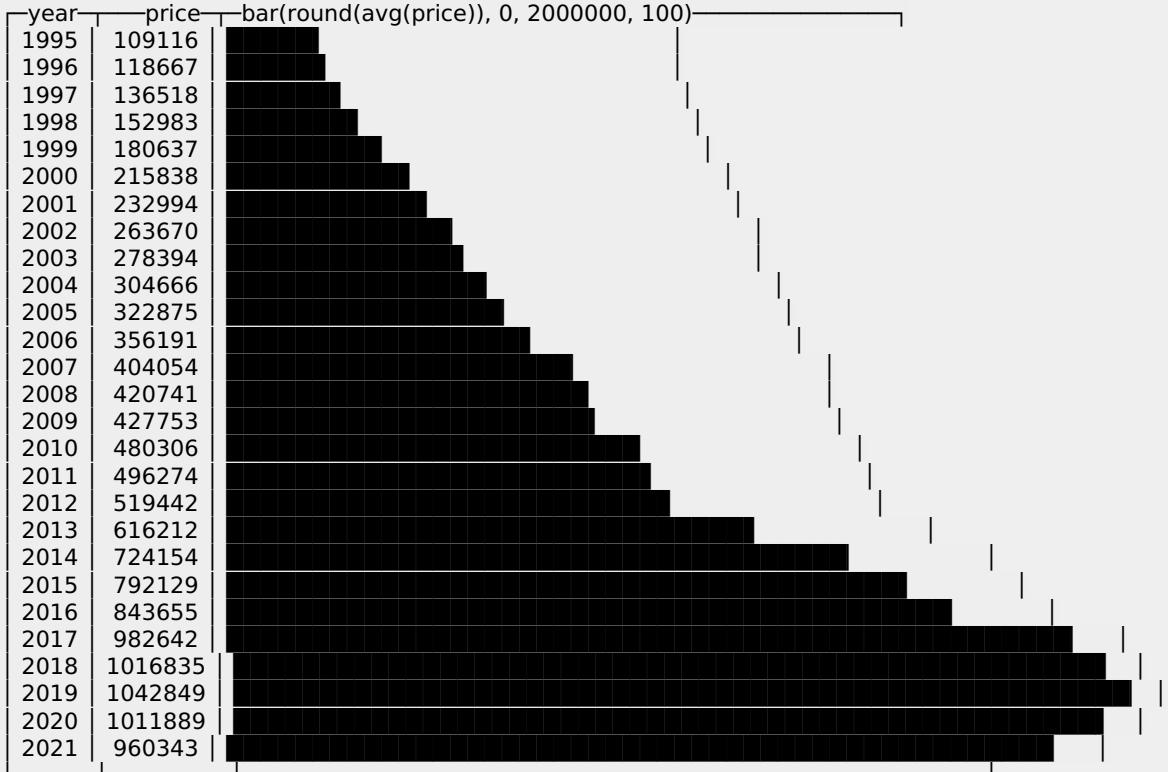


## Query 2. Average Price Per Year in London

Query:

```
SELECT
    toYear(date) AS year,
    round(avg(price)) AS price,
    bar(price, 0, 2000000, 100)
FROM uk_price_paid
WHERE town = 'LONDON'
GROUP BY year
ORDER BY year ASC;
```

Result:



### Query 3. The Most Expensive Neighborhoods

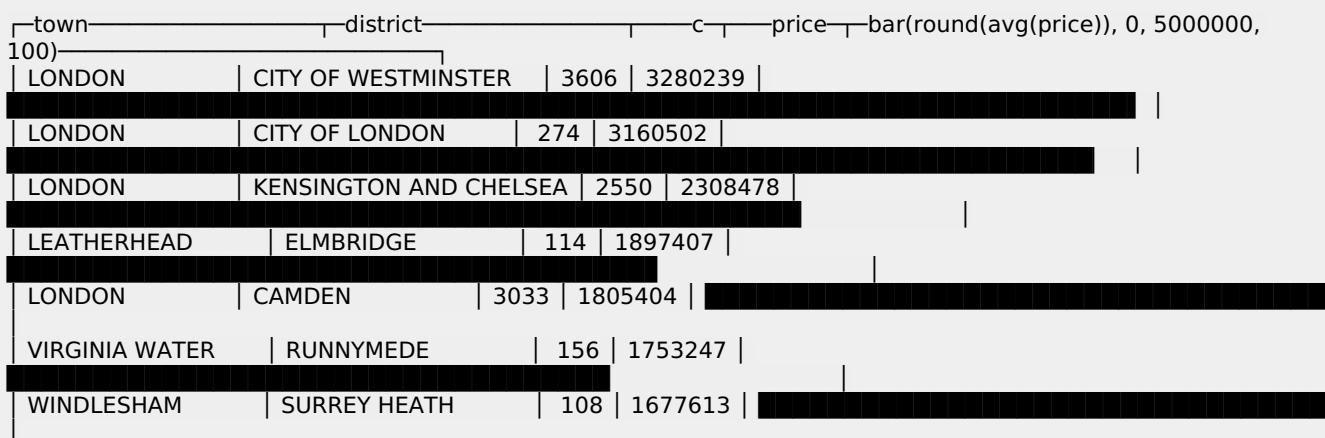
The condition (date >= '2020-01-01') needs to be modified to match projection dimension (toYear(date) >= 2020).

Query:

```

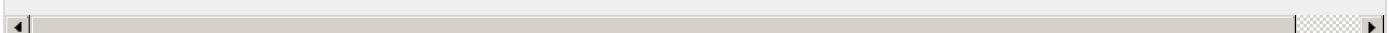
SELECT
  town,
  district,
  count() AS c,
  round(avg(price)) AS price,
  bar(price, 0, 5000000, 100)
FROM uk_price_paid
WHERE toYear(date) >= 2020
GROUP BY
  town,
  district
HAVING c >= 100
ORDER BY price DESC
LIMIT 100;
    
```

Result:



THORNTON HEATH	CROYDON	546   1671721   [REDACTED]
BARNET	ENFIELD	124   1505840   [REDACTED]
COBHAM	ELMBRIDGE	387   1237250   [REDACTED]
LONDON	ISLINGTON	2668   1236980   [REDACTED]
OXFORD	SOUTH OXFORDSHIRE	321   1220907   [REDACTED]
LONDON	RICHMOND UPON THAMES	704   1215551   [REDACTED]
LONDON	HOUNSLOW	671   1207493   [REDACTED]
ASCOT	WINDSOR AND MAIDENHEAD	407   1183299   [REDACTED]
BEACONSFIELD	BUCKINGHAMSHIRE	330   1175615   [REDACTED]
RICHMOND	RICHMOND UPON THAMES	874   1110444   [REDACTED]
LONDON	HAMMERSMITH AND FULHAM	3086   1053983   [REDACTED]
SURBITON	ELMBRIDGE	100   1011800   [REDACTED]
RADLETT	HERTSMERE	283   1011712   [REDACTED]
SALCOMBE	SOUTH HAMS	127   1011624   [REDACTED]
WEYBRIDGE	ELMBRIDGE	655   1007265   [REDACTED]
ESHER	ELMBRIDGE	485   986581   [REDACTED]
LEATHERHEAD	GUILDFORD	202   977320   [REDACTED]
BURFORD	WEST OXFORDSHIRE	111   966893   [REDACTED]
BROCKENHURST	NEW FOREST	129   956675   [REDACTED]
HINDHEAD	WAVERLEY	137   953753   [REDACTED]
GERRARDS CROSS	BUCKINGHAMSHIRE	419   951121   [REDACTED]
EAST MOLESEY	ELMBRIDGE	192   936769   [REDACTED]
CHALFONT ST GILES	BUCKINGHAMSHIRE	146   925515   [REDACTED]
LONDON	TOWER HAMLETS	4388   918304   [REDACTED]
OLNEY	MILTON KEYNES	235   910646   [REDACTED]
HENLEY-ON-THAMES	SOUTH OXFORDSHIRE	540   902418   [REDACTED]
LONDON	SOUTHWARK	3885   892997   [REDACTED]
KINGSTON UPON THAMES	KINGSTON UPON THAMES	960   885969   [REDACTED]
CRANBROOK	EALING	2658   871755   [REDACTED]
CRANBROOK	TUNBRIDGE WELLS	431   862348   [REDACTED]
LONDON	MERTON	2099   859118   [REDACTED]
BELVEDERE	BEXLEY	346   842423   [REDACTED]
GUILDFORD	WAVERLEY	143   841277   [REDACTED]
HARPENDEN	ST ALBANS	657   841216   [REDACTED]
LONDON	HACKNEY	3307   837090   [REDACTED]
LONDON	WANDSWORTH	6566   832663   [REDACTED]
MAIDENHEAD	BUCKINGHAMSHIRE	123   824299   [REDACTED]
KINGS LANGLEY	DACORUM	145   821331   [REDACTED]
BERKHAMSTED	DACORUM	543   818415   [REDACTED]
GREAT MISSENDEN	BUCKINGHAMSHIRE	226   802807   [REDACTED]
BILLINGSHURST	CHICHESTER	144   797829   [REDACTED]
WOKING	GUILDFORD	176   793494   [REDACTED]
STOCKBRIDGE	TEST VALLEY	178   793269   [REDACTED]

EPSOM	REIGATE AND BANSTEAD	172   791862   [REDACTED]
TONBRIDGE	TUNBRIDGE WELLS	360   787876   [REDACTED]
TEDDINGTON	RICHMOND UPON THAMES	595   786492   [REDACTED]
TWICKENHAM	RICHMOND UPON THAMES	1155   786193   [REDACTED]
LYNDHURST	NEW FOREST	102   785593   [REDACTED]
LONDON	LAMBETH	5228   774574   [REDACTED]
LONDON	BARNET	3955   773259   [REDACTED]
OXFORD	VALE OF WHITE HORSE	353   772088   [REDACTED]
TONBRIDGE	MAIDSTONE	305   770740   [REDACTED]
LUTTERWORTH	HARBOROUGH	538   768634   [REDACTED]
WOODSTOCK	WEST OXFORDSHIRE	140   766037   [REDACTED]
MIDHURST	CHICHESTER	257   764815   [REDACTED]
MARLOW	BUCKINGHAMSHIRE	327   761876   [REDACTED]
LONDON	NEWHAM	3237   761784   [REDACTED]
ALDERLEY EDGE	CHESHIRE EAST	178   757318   [REDACTED]
LUTON	CENTRAL BEDFORDSHIRE	212   754283   [REDACTED]
PETWORTH	CHICHESTER	154   754220   [REDACTED]
ALRESFORD	WINCHESTER	219   752718   [REDACTED]
POTTERS BAR	WELWYN HATFIELD	174   748465   [REDACTED]
HASLEMERE	CHICHESTER	128   746907   [REDACTED]
TADWORTH	REIGATE AND BANSTEAD	502   743252   [REDACTED]
THAMES DITTON	ELMBRIDGE	244   741913   [REDACTED]
REIGATE	REIGATE AND BANSTEAD	581   738198   [REDACTED]
BOURNE END	BUCKINGHAMSHIRE	138   735190   [REDACTED]
SEVENOAKS	SEVENOAKS	1156   730018   [REDACTED]
OXTED	TANDRIDGE	336   729123   [REDACTED]
INGATESTONE	BRENTWOOD	166   728103   [REDACTED]
LONDON	BRENT	2079   720605   [REDACTED]
LONDON	HARINGEY	3216   717780   [REDACTED]
PURLEY	CROYDON	575   716108   [REDACTED]
WELWYN	WELWYN HATFIELD	222   710603   [REDACTED]
RICKMANSWORTH	THREE RIVERS	798   704571   [REDACTED]
BANSTEAD	REIGATE AND BANSTEAD	401   701293   [REDACTED]
CHIGWELL	EPPING FOREST	261   701203   [REDACTED]
PINNER	HARROW	528   698885   [REDACTED]
HASLEMERE	WAVERLEY	280   696659   [REDACTED]
SLOUGH	BUCKINGHAMSHIRE	396   694917   [REDACTED]
WALTON-ON-THAMES	ELMBRIDGE	946   692395   [REDACTED]
READING	SOUTH OXFORDSHIRE	318   691988   [REDACTED]
NORTHWOOD	HILLINGDON	271   690643   [REDACTED]
FELTHAM	HOUNSLAW	763   688595   [REDACTED]
ASHTead	MOLE VALLEY	303   687923   [REDACTED]
BARNET	BARNET	975   686980   [REDACTED]
WOKING	SURREY HEATH	283   686669   [REDACTED]
MALMESBURY	WILTSHIRE	323   683324   [REDACTED]
AMERSHAM	BUCKINGHAMSHIRE	496   680962   [REDACTED]
CHISLEHURST	BROMLEY	430   680209   [REDACTED]
HYTHE	FOLKESTONE AND HYTHE	490   676908   [REDACTED]
MAYFIELD	WEALDEN	101   676210   [REDACTED]
ASCOT	BRACKNELL FOREST	168   676004   [REDACTED]



## Summary

All 3 queries work much faster and read fewer rows.

### Query 1

```
no projection: 27 rows in set. Elapsed: 0.158 sec. Processed 26.32 million rows, 157.93 MB (166.57 million rows/s., 999.39 MB/s.)  
projection: 27 rows in set. Elapsed: 0.007 sec. Processed 105.96 thousand rows, 3.33 MB (14.58 million rows/s., 458.13 MB/s.)
```

### Query 2

```
no projection: 27 rows in set. Elapsed: 0.163 sec. Processed 26.32 million rows, 80.01 MB (161.75 million rows/s., 491.64 MB/s.)  
projection: 27 rows in set. Elapsed: 0.008 sec. Processed 105.96 thousand rows, 3.67 MB (13.29 million rows/s., 459.89 MB/s.)
```

### Query 3

```
no projection: 100 rows in set. Elapsed: 0.069 sec. Processed 26.32 million rows, 62.47 MB (382.13 million rows/s., 906.93 MB/s.)  
projection: 100 rows in set. Elapsed: 0.029 sec. Processed 8.08 thousand rows, 511.08 KB (276.06 thousand rows/s., 17.47 MB/s.)
```

## Test It in Playground

The dataset is also available in the [Online Playground](#).

## スタースキーマ ベンチマーク

dbgen のコンパイル:

```
$ git clone git@github.com:vadimtk/ssb-dbgen.git  
$ cd ssb-dbgen  
$ make
```

データの生成:

### 注意

-s 100 をつけると dbgen は 600 万行(67GB)を生成します。 -s 1000 は 6 億行を生成します(これには非常に時間がかかります)

```
$ ./dbgen -s 1000 -T c  
$ ./dbgen -s 1000 -T l  
$ ./dbgen -s 1000 -T p  
$ ./dbgen -s 1000 -T s  
$ ./dbgen -s 1000 -T d
```

ClickHouseでのテーブルの作成:

```

CREATE TABLE customer
(
    C_CUSTKEY      UInt32,
    C_NAME         String,
    C_ADDRESS      String,
    C_CITY          LowCardinality(String),
    C_NATION        LowCardinality(String),
    C_REGION        LowCardinality(String),
    C_PHONE         String,
    C_MKTSEGMENT   LowCardinality(String)
)
ENGINE = MergeTree ORDER BY (C_CUSTKEY);

CREATE TABLE lineorder
(
    LO_ORDERKEY      UInt32,
    LO_LINENUMBER    UInt8,
    LO_CUSTKEY       UInt32,
    LO_PARTKEY       UInt32,
    LO_SUPPKEY       UInt32,
    LO_ORDERDATE     Date,
    LO_ORDERPRIORITY LowCardinality(String),
    LO_SHIPPRIORITY  UInt8,
    LO_QUANTITY      UInt8,
    LO_EXTENDEDPRICE UInt32,
    LO_ORDTOTALPRICE UInt32,
    LO_DISCOUNT      UInt8,
    LO_REVENUE       UInt32,
    LO_SUPPLYCOST    UInt32,
    LO_TAX           UInt8,
    LO_COMMITDATE    Date,
    LO_SHIPMODE      LowCardinality(String)
)
ENGINE = MergeTree PARTITION BY toYear(LO_ORDERDATE) ORDER BY (LO_ORDERDATE, LO_ORDERKEY);

CREATE TABLE part
(
    P_PARTKEY      UInt32,
    P_NAME         String,
    P_MFGR          LowCardinality(String),
    P_CATEGORY      LowCardinality(String),
    P_BRAND         LowCardinality(String),
    P_COLOR         LowCardinality(String),
    P_TYPE          LowCardinality(String),
    P_SIZE          UInt8,
    P_CONTAINER     LowCardinality(String)
)
ENGINE = MergeTree ORDER BY P_PARTKEY;

CREATE TABLE supplier
(
    S_SUPPKEY      UInt32,
    S_NAME         String,
    S_ADDRESS      String,
    S_CITY          LowCardinality(String),
    S_NATION        LowCardinality(String),
    S_REGION        LowCardinality(String),
    S_PHONE         String
)
ENGINE = MergeTree ORDER BY S_SUPPKEY;

```

データの挿入:

```

$ clickhouse-client --query "INSERT INTO customer FORMAT CSV" < customer.tbl
$ clickhouse-client --query "INSERT INTO part FORMAT CSV" < part.tbl
$ clickhouse-client --query "INSERT INTO supplier FORMAT CSV" < supplier.tbl
$ clickhouse-client --query "INSERT INTO lineorder FORMAT CSV" < lineorder.tbl

```

「star schema」を非正規化された「flat schema」に変換します。

```

SET max_memory_usage = 200000000000;

CREATE TABLE lineorder_flat
ENGINE = MergeTree
PARTITION BY toYear(LO_ORDERDATE)
ORDER BY (LO_ORDERDATE, LO_ORDERKEY) AS
SELECT
    I.LO_ORDERKEY AS LO_ORDERKEY,
    I.LO_LINENUMBER AS LO_LINENUMBER,
    I.LO_CUSTKEY AS LO_CUSTKEY,
    I.LO_PARTKEY AS LO_PARTKEY,
    I.LO_SUPPKEY AS LO_SUPPKEY,
    I.LO_ORDERDATE AS LO_ORDERDATE,
    I.LO_ORDERPRIORITY AS LO_ORDERPRIORITY,
    I.LO_SHIPPRIORITY AS LO_SHIPPRIORITY,
    I.LO_QUANTITY AS LO_QUANTITY,
    I.LO_EXTENDEDPRICE AS LO_EXTENDEDPRICE,
    I.LO_ORDTOTALPRICE AS LO_ORDTOTALPRICE,
    I.LO_DISCOUNT AS LO_DISCOUNT,
    I.LO_REVENUE AS LO_REVENUE,
    I.LO_SUPPLYCOST AS LO_SUPPLYCOST,
    I.LO_TAX AS LO_TAX,
    I.LO_COMMITDATE AS LO_COMMITDATE,
    I.LO_SHIPMODE AS LO_SHIPMODE,
    c.C_NAME AS C_NAME,
    c.C_ADDRESS AS C_ADDRESS,
    c.C_CITY AS C_CITY,
    c.C_NATION AS C_NATION,
    c.C_REGION AS C_REGION,
    c.C_PHONE AS C_PHONE,
    c.C_MKTSEGMENT AS C_MKTSEGMENT,
    s.S_NAME AS S_NAME,
    s.S_ADDRESS AS S_ADDRESS,
    s.S_CITY AS S_CITY,
    s.S_NATION AS S_NATION,
    s.S_REGION AS S_REGION,
    s.S_PHONE AS S_PHONE,
    p.P_NAME AS P_NAME,
    p.P_MFGR AS P_MFGR,
    p.P_CATEGORY AS P_CATEGORY,
    p.P_BRAND AS P_BRAND,
    p.P_COLOR AS P_COLOR,
    p.P_TYPE AS P_TYPE,
    p.P_SIZE AS P_SIZE,
    p.P_CONTAINER AS P_CONTAINER
FROM lineorder AS I
INNER JOIN customer AS c ON c.C_CUSTKEY = I.LO_CUSTKEY
INNER JOIN supplier AS s ON s.S_SUPPKEY = I.LO_SUPPKEY
INNER JOIN part AS p ON p.P_PARTKEY = I.LO_PARTKEY;

```

クエリの実行:

Q1.1

```

SELECT sum(LO_EXTENDEDPRICE * LO_DISCOUNT) AS revenue
FROM lineorder_flat
WHERE toYear(LO_ORDERDATE) = 1993 AND LO_DISCOUNT BETWEEN 1 AND 3 AND LO_QUANTITY < 25;

```

Q1.2

```

SELECT sum(LO_EXTENDEDPRICE * LO_DISCOUNT) AS revenue
FROM lineorder_flat
WHERE toYYYYMM(LO_ORDERDATE) = 199401 AND LO_DISCOUNT BETWEEN 4 AND 6 AND LO_QUANTITY BETWEEN 26
AND 35;

```

Q1.3

```
SELECT sum(LO_EXTENDEDPRICE * LO_DISCOUNT) AS revenue
FROM lineorder_flat
WHERE toISOWeek(LO_ORDERDATE) = 6 AND toYear(LO_ORDERDATE) = 1994
AND LO_DISCOUNT BETWEEN 5 AND 7 AND LO_QUANTITY BETWEEN 26 AND 35;
```

Q2.1

```
SELECT
    sum(LO_REVENUE),
    toYear(LO_ORDERDATE) AS year,
    P_BRAND
FROM lineorder_flat
WHERE P_CATEGORY = 'MFGR#12' AND S_REGION = 'AMERICA'
GROUP BY
    year,
    P_BRAND
ORDER BY
    year,
    P_BRAND;
```

Q2.2

```
SELECT
    sum(LO_REVENUE),
    toYear(LO_ORDERDATE) AS year,
    P_BRAND
FROM lineorder_flat
WHERE P_BRAND >= 'MFGR#2221' AND P_BRAND <= 'MFGR#2228' AND S_REGION = 'ASIA'
GROUP BY
    year,
    P_BRAND
ORDER BY
    year,
    P_BRAND;
```

Q2.3

```
SELECT
    sum(LO_REVENUE),
    toYear(LO_ORDERDATE) AS year,
    P_BRAND
FROM lineorder_flat
WHERE P_BRAND = 'MFGR#2239' AND S_REGION = 'EUROPE'
GROUP BY
    year,
    P_BRAND
ORDER BY
    year,
    P_BRAND;
```

Q3.1

```

SELECT
  C_NATION,
  S_NATION,
  toYear(LO_ORDERDATE) AS year,
  sum(LO_REVENUE) AS revenue
FROM lineorder_flat
WHERE C_REGION = 'ASIA' AND S_REGION = 'ASIA' AND year >= 1992 AND year <= 1997
GROUP BY
  C_NATION,
  S_NATION,
  year
ORDER BY
  year ASC,
  revenue DESC;

```

Q3.2

```

SELECT
  C_CITY,
  S_CITY,
  toYear(LO_ORDERDATE) AS year,
  sum(LO_REVENUE) AS revenue
FROM lineorder_flat
WHERE C_NATION = 'UNITED STATES' AND S_NATION = 'UNITED STATES' AND year >= 1992 AND year <= 1997
GROUP BY
  C_CITY,
  S_CITY,
  year
ORDER BY
  year ASC,
  revenue DESC;

```

Q3.3

```

SELECT
  C_CITY,
  S_CITY,
  toYear(LO_ORDERDATE) AS year,
  sum(LO_REVENUE) AS revenue
FROM lineorder_flat
WHERE (C_CITY = 'UNITED KI1' OR C_CITY = 'UNITED KI5') AND (S_CITY = 'UNITED KI1' OR S_CITY = 'UNITED KI5') AND
year >= 1992 AND year <= 1997
GROUP BY
  C_CITY,
  S_CITY,
  year
ORDER BY
  year ASC,
  revenue DESC;

```

Q3.4

```

SELECT
  C_CITY,
  S_CITY,
  toYear(LO_ORDERDATE) AS year,
  sum(LO_REVENUE) AS revenue
FROM lineorder_flat
WHERE (C_CITY = 'UNITED KI1' OR C_CITY = 'UNITED KI5') AND (S_CITY = 'UNITED KI1' OR S_CITY = 'UNITED KI5') AND
toYYYYMM(LO_ORDERDATE) = 199712
GROUP BY
  C_CITY,
  S_CITY,
  year
ORDER BY
  year ASC,
  revenue DESC;

```

Q4.1

```
SELECT
    toYear(LO_ORDERDATE) AS year,
    C_NATION,
    sum(LO_REVENUE - LO_SUPPLYCOST) AS profit
FROM lineorder_flat
WHERE C_REGION = 'AMERICA' AND S_REGION = 'AMERICA' AND (P_MFGR = 'MFGR#1' OR P_MFGR = 'MFGR#2')
GROUP BY
    year,
    C_NATION
ORDER BY
    year ASC,
    C_NATION ASC;
```

Q4.2

```
SELECT
    toYear(LO_ORDERDATE) AS year,
    S_NATION,
    P_CATEGORY,
    sum(LO_REVENUE - LO_SUPPLYCOST) AS profit
FROM lineorder_flat
WHERE C_REGION = 'AMERICA' AND S_REGION = 'AMERICA' AND (year = 1997 OR year = 1998) AND (P_MFGR =
'MFGR#1' OR P_MFGR = 'MFGR#2')
GROUP BY
    year,
    S_NATION,
    P_CATEGORY
ORDER BY
    year ASC,
    S_NATION ASC,
    P_CATEGORY ASC;
```

Q4.3

```
SELECT
    toYear(LO_ORDERDATE) AS year,
    S_CITY,
    P_BRAND,
    sum(LO_REVENUE - LO_SUPPLYCOST) AS profit
FROM lineorder_flat
WHERE S_NATION = 'UNITED STATES' AND (year = 1997 OR year = 1998) AND P_CATEGORY = 'MFGR#14'
GROUP BY
    year,
    S_CITY,
    P_BRAND
ORDER BY
    year ASC,
    S_CITY ASC,
    P_BRAND ASC;
```

## Cell Towers

This dataset is from [OpenCellid](#) - The world's largest Open Database of Cell Towers.

As of 2021, it contains more than 40 million records about cell towers (GSM, LTE, UMTS, etc.) around the world with their geographical coordinates and metadata (country code, network, etc).

OpenCellID Project is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License, and we redistribute a snapshot of this dataset under the terms of the same license. The up-to-date version of the dataset is available to download after sign in.

## Get the Dataset

1. Download the snapshot of the dataset from February 2021:  
[[https://datasets.clickhouse.com/cell\\_towers.csv.xz](https://datasets.clickhouse.com/cell_towers.csv.xz)] (729 MB).

2. Validate the integrity (optional step):

```
md5sum cell_towers.csv.xz  
8cf986f4a0d9f12c6f384a0e9192c908 cell_towers.csv.xz
```

3. Decompress it with the following command:

```
xz -d cell_towers.csv.xz
```

4. Create a table:

```
CREATE TABLE cell_towers  
(  
    radio Enum8(" = 0, 'CDMA' = 1, 'GSM' = 2, 'LTE' = 3, 'NR' = 4, 'UMTS' = 5),  
    mcc UInt16,  
    net UInt16,  
    area UInt16,  
    cell UInt64,  
    unit Int16,  
    lon Float64,  
    lat Float64,  
    range UInt32,  
    samples UInt32,  
    changeable UInt8,  
    created DateTime,  
    updated DateTime,  
    averageSignal UInt8  
)  
ENGINE = MergeTree ORDER BY (radio, mcc, net, created);
```

5. Insert the dataset:

```
clickhouse-client --query "INSERT INTO cell_towers FORMAT CSVWithNames" < cell_towers.csv
```

## Examples

1. A number of cell towers by type:

```
SELECT radio, count() AS c FROM cell_towers GROUP BY radio ORDER BY c DESC
```

radio	c
UMTS	20686487
LTE	12101148
GSM	9931312
CDMA	556344
NR	867

```
5 rows in set. Elapsed: 0.011 sec. Processed 43.28 million rows, 43.28 MB (3.83 billion rows/s., 3.83 GB/s.)
```

2. Cell towers by **mobile country code (MCC)**:

```
SELECT mcc, count() FROM cell_towers GROUP BY mcc ORDER BY count() DESC LIMIT 10
```

mcc	count()
310	5024650
262	2622423
250	1953176
208	1891187
724	1836150
404	1729151
234	1618924
510	1353998
440	1343355
311	1332798

10 rows in set. Elapsed: 0.019 sec. Processed 43.28 million rows, 86.55 MB (2.33 billion rows/s., 4.65 GB/s.)

So, the top countries are: the USA, Germany, and Russia.

You may want to create an [External Dictionary](#) in ClickHouse to decode these values.

## Use case

Using pointInPolygon function.

1. Create a table where we will store polygons:

```
CREATE TEMPORARY TABLE moscow (polygon Array(Tuple(Float64, Float64)));
```

2. This is a rough shape of Moscow (without "new Moscow"):

```
INSERT INTO moscow VALUES ([(37.84172564285271, 55.78000432402266), (37.8381207618713, 55.775874525970494), (37.83979446823122, 55.775626746008065), (37.84243326983639, 55.77446586811748), (37.84262672750849, 55.771974101091104), (37.84153238623039, 55.77114545193181), (37.841124690460184, 55.76722010265554), (37.84239076983644, 55.76654891107098), (37.842283558197025, 55.76258709833121), (37.8421759312134, 55.758073999993734), (37.84198330422974, 55.75381499999371), (37.8416827275085, 55.749277102484484), (37.84157576190186, 55.74794544108413), (37.83897929098507, 55.74525257875241), (37.83739676451868, 55.74404373042019), (37.838732481460525, 55.74298009816793), (37.841183997352545, 55.743060321833575), (37.84097476190185, 55.73938799999373), (37.84048155819702, 55.73570799999372), (37.840095812164286, 55.73228210777237), (37.83983814285274, 55.73080491981639), (37.83846476321406, 55.729799917464675), (37.83835745269769, 55.72919751082619), (37.838636380279524, 55.72859509486539), (37.8395161005249, 55.727705075632784), (37.83897964285276, 55.722727886185154), (37.83862557539366, 55.72034817326636), (37.83559735744853, 55.71944437307499), (37.83537070803126, 55.71831419154461), (37.83738169402022, 55.71765218986692), (37.83823396494291, 55.71691750159089), (37.838056931213345, 55.71547311301385), (37.836812846557606, 55.71221445615604), (37.83522525396725, 55.709331054395555), (37.83269301586908, 55.70953687463627), (37.829667367706236, 55.70903403789297), (37.83311126588435, 55.70552351822608), (37.83058993121339, 55.70041317726053), (37.82983872750851, 55.69883771404813), (37.82934501586913, 55.69718947487017), (37.828926414016685, 55.69504441658371), (37.82876530422971, 55.69287499999378), (37.82894754100031, 55.690759754047335), (37.827697554878185, 55.68951421135665), (37.82447346292115, 55.68965045405069), (37.83136543914793, 55.68322046195302), (37.833554015869154, 55.67814012759211), (37.83544184655761, 55.67295011628339), (37.83748038885474, 55.6672498719639), (37.838960677246064, 55.66316274139358), (37.83926093121332, 55.66046999999383), (37.839025050262435, 55.65869897264431), (37.83670784390257, 55.65794084879904), (37.835656529083245, 55.65694309303843), (37.83704060449217, 55.65689306460552), (37.83696819873806, 55.65550363526252), (37.83760389616388, 55.65487847246661), (37.83687972750851, 55.65356745541324), (37.83515216004943, 55.65155951234079), (37.83312418518067, 55.64979413590619), (37.82801726983639, 55.64640836412121), (37.820614174591, 55.64164525405531), (37.818908190475426, 55.6421883258084), (37.81717543386075, 55.64112490388471), (37.81690987037274, 55.63916106913107), (37.815099354492155, 55.637925371757085), (37.808769150787356, 55.633798276884455), (37.80100123544311, 55.62873670012244), (37.79598013491824, 55.62554336109055), (37.78634567724606, 55.62033499605651), (37.78334147619623, 55.618768681480326), (37.77746201055901, 55.619855533402706), (37.77527329626457, 55.61909966711279), (37.77801986242668, 55.618770300976294), (37.778212973541216, 55.617257701952106), (37.77784818518065, 55.61574504433011), (37.77016867724609, 55.61148576294007), (37.760191219573976, 55.60599579539028), (37.75338926983641, 55.60227892751446), (37.746329965606634, 55.59920577639331), (37.73939925396728, 55.59631430313617), (37.73273665739439, 55.5935318803559), (37.7299954450912, 55.59350760316188), (37.7268679946899, 55.59469840523759), (37.72626726983634, 55.59229549697373), (37.7262673598022, 55.59081598950582), (37.71897193121335, 55.5877595845419), (37.70871550793456, 55.58393177431724), (37.700497489410374, 55.580917323756644), (37.69204305026244, 55.57778089778455), (37.68544477378839, 55.57815154690915), (37.68391050793454, 55.57472945079756), (37.678803592590306, 55.57328235936491), (37.6743402539673, 55.57255251445782), (37.66813862698363, 55.57216388774464), (37.617927457672096, 55.575056918958051), (37.604430999999999)
```

(37.5001300209303, 55.57210300, 77704), (37.50132, 77702, 2093, 55.5730031093003), (37.504433333333333, 55.5757737568051), (37.599683515869145, 55.57749105910326), (37.59754177842709, 55.57796291823627), (37.59625834786988, 55.57906686095235), (37.59501783265684, 55.57746616444403), (37.593090671936025, 55.57671634534502), (37.587018007904, 55.577944600233785), (37.578692203704804, 55.57982895000019), (37.57327546607398, 55.58116294118248), (37.57385012109279, 55.581550362779), (37.57399562266922, 55.5820107079112), (37.5735356072979, 55.58226289171689), (37.57290393054962, 55.582393529795155), (37.57037722355653, 55.581919415056234), (37.5592298306885, 55.584471614867844), (37.54189249206543, 55.58867650795186), (37.5297256269836, 55.59158133551745), (37.517837865081766, 55.59443656218868), (37.51200186508174, 55.59635625174229), (37.506808949737554, 55.59907823904434), (37.49820432275389, 55.6062944994944), (37.494406071441674, 55.60967103463367), (37.494760001358024, 55.61066689753365), (37.49397137107085, 55.61220931698269), (37.49016528606031, 55.613417718449064), (37.48773249206542, 55.61530616333343), (37.47921386508177, 55.622640129112334), (37.470652153442394, 55.62993723476164), (37.46273446298218, 55.6368075123157), (37.46350692265317, 55.64068225239439), (37.46050283203121, 55.640794546982576), (37.457627470916734, 55.64118904154646), (37.450718034393326, 55.64690488145138), (37.44239252645875, 55.65397824729769), (37.434587576721185, 55.66053543155961), (37.43582144975277, 55.661693766520735), (37.43576786245721, 55.662755031737014), (37.430982915344174, 55.664610641628116), (37.428547447097685, 55.66778515273695), (37.42945134592044, 55.668633314343566), (37.42859571562949, 55.66948145750025), (37.4262836402282, 55.670813882451405), (37.418709037048295, 55.6811141674414), (37.41922139651101, 55.6823537785389), (37.419218771842885, 55.68359335082235), (37.417196501327446, 55.684375235224735), (37.41607020370478, 55.68540557585352), (37.415640857147146, 55.68686637150793), (37.414632153442334, 55.68903015131686), (37.413344899475064, 55.690896881757396), (37.41171432275391, 55.69264232162232), (37.40948282275393, 55.69455101638112), (37.40703674603271, 55.69638690385348), (37.39607169577025, 55.70451821283731), (37.38952706878662, 55.70942491932811), (37.387778313491815, 55.71149057784176), (37.39049275399779, 55.71419814298992), (37.385557272491454, 55.7155489617061), (37.38388335714726, 55.71849856042102), (37.378368238098155, 55.7292763261685), (37.37763597123337, 55.730845879211614), (37.37890062088197, 55.73167906388319), (37.37750451918789, 55.734703664681774), (37.375610832015965, 55.734851959522246), (37.3723813571472, 55.74105626086403), (37.37014935714723, 55.746115620904355), (37.36944173016362, 55.750883999993725), (37.36975304365541, 55.76335905525834), (37.37244070571134, 55.76432079697595), (37.3724259757175, 55.76636979670426), (37.369922155757884, 55.76735417953104), (37.369892695770275, 55.76823419316575), (37.370214730163575, 55.782312184391266), (37.370493611114505, 55.78436801120489), (37.37120164550783, 55.78596427165359), (37.37284851456452, 55.7874378183096), (37.37608325135799, 55.7886695054807), (37.3764587460632, 55.78947647305964), (37.37530000265506, 55.79146512926804), (37.38235915344241, 55.79899647809345), (37.384344043655396, 55.80113596939471), (37.38594269577028, 55.80322699999366), (37.38711208598329, 55.804919036911976), (37.3880239841309, 55.806610999993666), (37.38928977249147, 55.81001864976979), (37.39038389947512, 55.81348641242801), (37.39235781481933, 55.81983538336746), (37.393709457672124, 55.82417822811877), (37.394685720901464, 55.82792275755836), (37.39557615344238, 55.830447148154136), (37.39844478226658, 55.83167107969975), (37.40019761214057, 55.83151823557964), (37.400398790382326, 55.83264967594742), (37.39659544313046, 55.83322180909622), (37.39667059524539, 55.83402792148566), (37.39682089947515, 55.83638877400216), (37.39643489154053, 55.83861656112751), (37.3955338994751, 55.84072348043264), (37.392680272491454, 55.84502158126453), (37.39241188227847, 55.84659117913199), (37.392529730163616, 55.84816071336481), (37.39486835714723, 55.85288092980303), (37.39873052645878, 55.859893456073635), (37.40272161111449, 55.86441833633205), (37.40697072750854, 55.867579567544375), (37.410007082016016, 55.868369880337), (37.4120992989502, 55.86920843741314), (37.412668021163924, 55.870553369615854), (37.41482461111453, 55.87170587948249), (37.41862266137694, 55.873183961039565), (37.42413732540892, 55.874879126654704), (37.4312182698669, 55.875614937236705), (37.43111093783558, 55.8762723478417), (37.43332105622856, 55.87706546369396), (37.43385747619623, 55.87790681284802), (37.441303050262405, 55.88027084462084), (37.44747234260555, 55.87942070143253), (37.44716141796871, 55.88072960917233), (37.44769797085568, 55.88121221323979), (37.45204320500181, 55.882080694420715), (37.45673176190186, 55.882346110794586), (37.463383999999984, 55.88252729504517), (37.46682797486874, 55.88294937719063), (37.470014457672086, 55.88361266759345), (37.47751410450743, 55.88546991372396), (37.47860317658232, 55.88534929207307), (37.48165826025772, 55.882563306475106), (37.48316434442331, 55.8815803226785), (37.483831555817645, 55.882427612793315), (37.483182967125686, 55.88372791409729), (37.483092277908824, 55.88495581062434), (37.4855716508179, 55.8875561994203), (37.486440636245746, 55.887827444039566), (37.49014203439328, 55.88897899871799), (37.493210285705544, 55.890208937135604), (37.497512451065035, 55.891342397444696), (37.49780744510645, 55.89174030252967), (37.49940333499519, 55.892397455070797), (37.50018383334346, 55.89339220941865), (37.52421672750851, 55.903869074155224), (37.52977457672118, 55.90564076517974), (37.53503220370484, 55.90661661218259), (37.54042858064267, 55.90714113744566), (37.54320461007303, 55.905645048442985), (37.545686966066306, 55.906608607018505), (37.54743976120755, 55.90788552162358), (37.557969999999999, 55.90901557907218), (37.572711542327866, 55.91059395704873), (37.579427999999998, 55.91073854155573), (37.58502865872187, 55.91009969268444), (37.58739968913264, 55.90794809960554), (37.59131567193598, 55.908713267595054), (37.612687423278814, 55.902866854295375), (37.62348079629517, 55.90041967242986), (37.635797880950896, 55.898141151686396), (37.649487626983664, 55.89639275532968), (37.65619302513125, 55.89572360207488), (37.66294133862307, 55.895295577183965), (37.66874564418033, 55.89505457604897), (37.67375601586915, 55.89254677027454), (37.67744661901856, 55.8947775867987), (37.688347, 55.89450045676125), (37.69480554232789, 55.89422926332761), (37.70107096560668, 55.89322256101114), (37.705962965606716, 55.891763491662616), (37.711885134918205, 55.889110234998974), (37.71682005026245, 55.886577568759876), (37.7199315476074, 55.88458159806678), (37.72234560316464, 55.882281005794134), (37.72364385977171, 55.8809452036196), (37.725371142837474, 55.8809722706006), (37.727870902099546, 55.88037213862385), (37.73394330422971, 55.877941504088696), (37.745339592590376, 55.87208120378722), (37.75525267724611, 55.86703807949492), (37.76919976190188, 55.859821640197474), (37.827835219574, 55.82962968399116), (37.83341438888553, 55.82575289922351), (37.83652584655761, 55.82188784027888), (37.83809213491821, 55.81612575504693), (37.83605359521481, 55.81460347077685), (37.83632178569025, 55.81276696067908), (37.838623105812026, 55.811486181656385), (37.83912198147584, 55.807329380532785), (37.839079078033414, 55.80510270463816), (37.83965844708251, 55.79940712529036), (37.840581150787344, 55.79131399999368), (37.84172564285271, 55.78000432402266)])

3. Check how many cell towers are in Moscow:

```
SELECT count() FROM cell_towers WHERE pointInPolygon((lon, lat), (SELECT * FROM moscow))
count()
310463
```

1 rows in set. Elapsed: 0.067 sec. Processed 43.28 million rows, 692.42 MB (645.83 million rows/s., 10.33 GB/s.)

The data is also available for interactive queries in the [Playground](#), [example](#).

Although you cannot create temporary tables there.

## New York Public Library "What's on the Menu?" Dataset

The dataset is created by the New York Public Library. It contains historical data on the menus of hotels, restaurants and cafes with the dishes along with their prices.

Source: <http://menus.nypl.org/data>

The data is in public domain.

The data is from library's archive and it may be incomplete and difficult for statistical analysis.  
Nevertheless it is also very yummy.

The size is just 1.3 million records about dishes in the menus — it's a very small data volume for ClickHouse, but it's still a good example.

## Download the Dataset

Run the command:

```
wget https://s3.amazonaws.com/menusdata.nypl.org/gzips/2021_08_01_07_01_17_data.tgz
```

Replace the link to the up to date link from <http://menus.nypl.org/data> if needed.  
Download size is about 35 MB.

## Unpack the Dataset

```
tar xvf 2021_08_01_07_01_17_data.tgz
```

Uncompressed size is about 150 MB.

The data is normalized consisted of four tables:

- `Menu` — Information about menus: the name of the restaurant, the date when menu was seen, etc.
- `Dish` — Information about dishes: the name of the dish along with some characteristic.
- `MenuPage` — Information about the pages in the menus, because every page belongs to some menu.
- `MenuItem` — An item of the menu. A dish along with its price on some menu page: links to dish and menu page.

## Create the Tables

We use `Decimal` data type to store prices.

```

CREATE TABLE dish
(
    id UInt32,
    name String,
    description String,
    menus_appeared UInt32,
    times_appeared Int32,
    first_appeared UInt16,
    last_appeared UInt16,
    lowest_price Decimal64(3),
    highest_price Decimal64(3)
) ENGINE = MergeTree ORDER BY id;

CREATE TABLE menu
(
    id UInt32,
    name String,
    sponsor String,
    event String,
    venue String,
    place String,
    physical_description String,
    occasion String,
    notes String,
    call_number String,
    keywords String,
    language String,
    date String,
    location String,
    location_type String,
    currency String,
    currency_symbol String,
    status String,
    page_count UInt16,
    dish_count UInt16
) ENGINE = MergeTree ORDER BY id;

CREATE TABLE menu_page
(
    id UInt32,
    menu_id UInt32,
    page_number UInt16,
    image_id String,
    full_height UInt16,
    full_width UInt16,
    uuid UUID
) ENGINE = MergeTree ORDER BY id;

CREATE TABLE menu_item
(
    id UInt32,
    menu_page_id UInt32,
    price Decimal64(3),
    high_price Decimal64(3),
    dish_id UInt32,
    created_at DateTime,
    updated_at DateTime,
    xpos Float64,
    ypos Float64
) ENGINE = MergeTree ORDER BY id;

```

## Import the Data

Upload data into ClickHouse, run:

```
clickhouse-client --format_csv_allow_single_quotes 0 --input_format_null_as_default 0 --query "INSERT INTO dish
FORMAT CSVWithNames" < Dish.csv
clickhouse-client --format_csv_allow_single_quotes 0 --input_format_null_as_default 0 --query "INSERT INTO menu
FORMAT CSVWithNames" < Menu.csv
clickhouse-client --format_csv_allow_single_quotes 0 --input_format_null_as_default 0 --query "INSERT INTO
menu_page FORMAT CSVWithNames" < MenuPage.csv
clickhouse-client --format_csv_allow_single_quotes 0 --input_format_null_as_default 0 --date_time_input_format
best_effort --query "INSERT INTO menu_item FORMAT CSVWithNames" < MenuItem.csv
```

We use **CSVWithNames** format as the data is represented by CSV with header.

We disable **format\_csv\_allow\_single\_quotes** as only double quotes are used for data fields and single quotes can be inside the values and should not confuse the CSV parser.

We disable **input\_format\_null\_as\_default** as our data does not have **NULL**. Otherwise ClickHouse will try to parse \N sequences and can be confused with \ in data.

The setting **date\_time\_input\_format best\_effort** allows to parse **DateTime** fields in wide variety of formats. For example, ISO-8601 without seconds like '2000-01-01 01:02' will be recognized. Without this setting only fixed DateTime format is allowed.

## Denormalize the Data

Data is presented in multiple tables in **normalized form**. It means you have to perform **JOIN** if you want to query, e.g. dish names from menu items.

For typical analytical tasks it is way more efficient to deal with pre-JOINed data to avoid doing **JOIN** every time. It is called "denormalized" data.

We will create a table `menu_item_denorm` which will contain all the data JOINed together:

```

CREATE TABLE menu_item_denorm
ENGINE = MergeTree ORDER BY (dish_name, created_at)
AS SELECT
    price,
    high_price,
    created_at,
    updated_at,
    xpos,
    ypos,
    dish.id AS dish_id,
    dish.name AS dish_name,
    dish.description AS dish_description,
    dish.menus_appeared AS dish_menus_appeared,
    dish.times_appeared AS dish_times_appeared,
    dish.first_appeared AS dish_first_appeared,
    dish.last_appeared AS dish_last_appeared,
    dish.lowest_price AS dish_lowest_price,
    dish.highest_price AS dish_highest_price,
    menu.id AS menu_id,
    menu.name AS menu_name,
    menu.sponsor AS menu_sponsor,
    menu.event AS menu_event,
    menu.venue AS menu_venue,
    menu.place AS menu_place,
    menu.physical_description AS menu_physical_description,
    menu.occasion AS menu_occasion,
    menu.notes AS menu_notes,
    menu.call_number AS menu_call_number,
    menu.keywords AS menu_keywords,
    menu.language AS menu_language,
    menu.date AS menu_date,
    menu.location AS menu_location,
    menu.location_type AS menu_location_type,
    menu.currency AS menu_currency,
    menu.currency_symbol AS menu_currency_symbol,
    menu.status AS menu_status,
    menu.page_count AS menu_page_count,
    menu.dish_count AS menu_dish_count
FROM menu_item
JOIN dish ON menu_item.dish_id = dish.id
JOIN menu_page ON menu_item.menu_page_id = menu_page.id
JOIN menu ON menu_page.menu_id = menu.id;

```

## Validate the Data

Query:

```
SELECT count() FROM menu_item_denorm;
```

Result:

count()
1329175

## Run Some Queries

Averaged historical prices of dishes

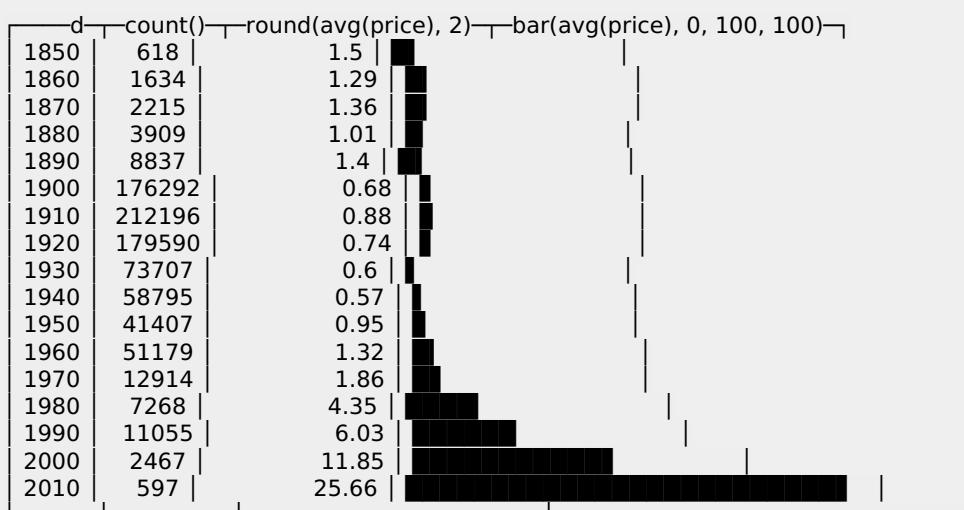
Query:

```

SELECT
    round(toUInt32OrZero(extract(menu_date, '^\\d{4}')), -1) AS d,
    count(),
    round(avg(price), 2),
    bar(avg(price), 0, 100, 100)
FROM menu_item_denorm
WHERE (menu_currency = 'Dollars') AND (d > 0) AND (d < 2022)
GROUP BY d
ORDER BY d ASC;

```

Result:



Take it with a grain of salt.

## Burger Prices

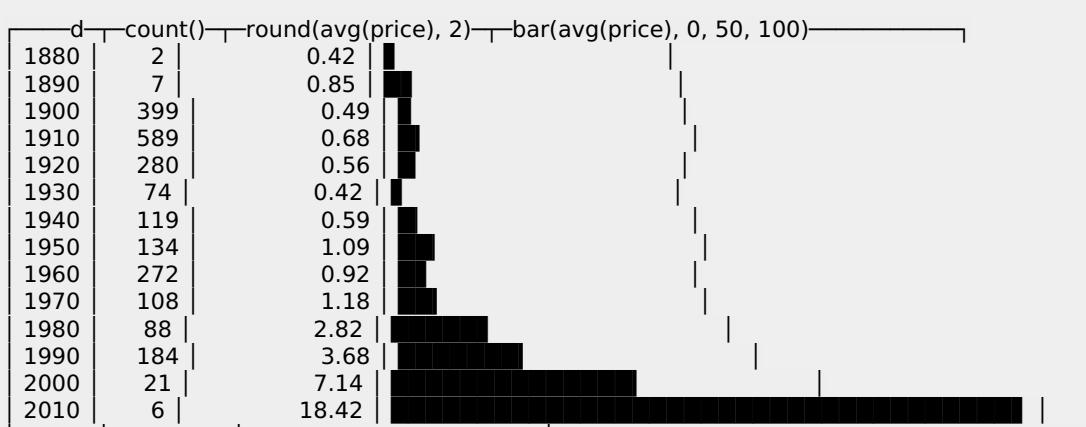
Query:

```

SELECT
    round(toUInt32OrZero(extract(menu_date, '^\\d{4}')), -1) AS d,
    count(),
    round(avg(price), 2),
    bar(avg(price), 0, 50, 100)
FROM menu_item_denorm
WHERE (menu_currency = 'Dollars') AND (d > 0) AND (d < 2022) AND (dish_name ILIKE '%burger%')
GROUP BY d
ORDER BY d ASC;

```

Result:

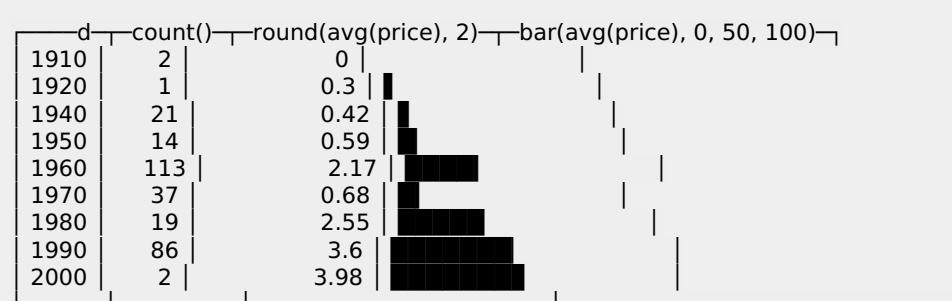


## Vodka

Query:

```
SELECT
    round(toUInt32OrZero(extract(menu_date, '^\\d{4}')), -1) AS d,
    count(),
    round(avg(price), 2),
    bar(avg(price), 0, 50, 100)
FROM menu_item_denorm
WHERE (menu_currency IN ('Dollars', '')) AND (d > 0) AND (d < 2022) AND (dish_name ILIKE '%vodka%')
GROUP BY d
ORDER BY d ASC;
```

Result:



To get vodka we have to write ILIKE '%vodka%' and this definitely makes a statement.

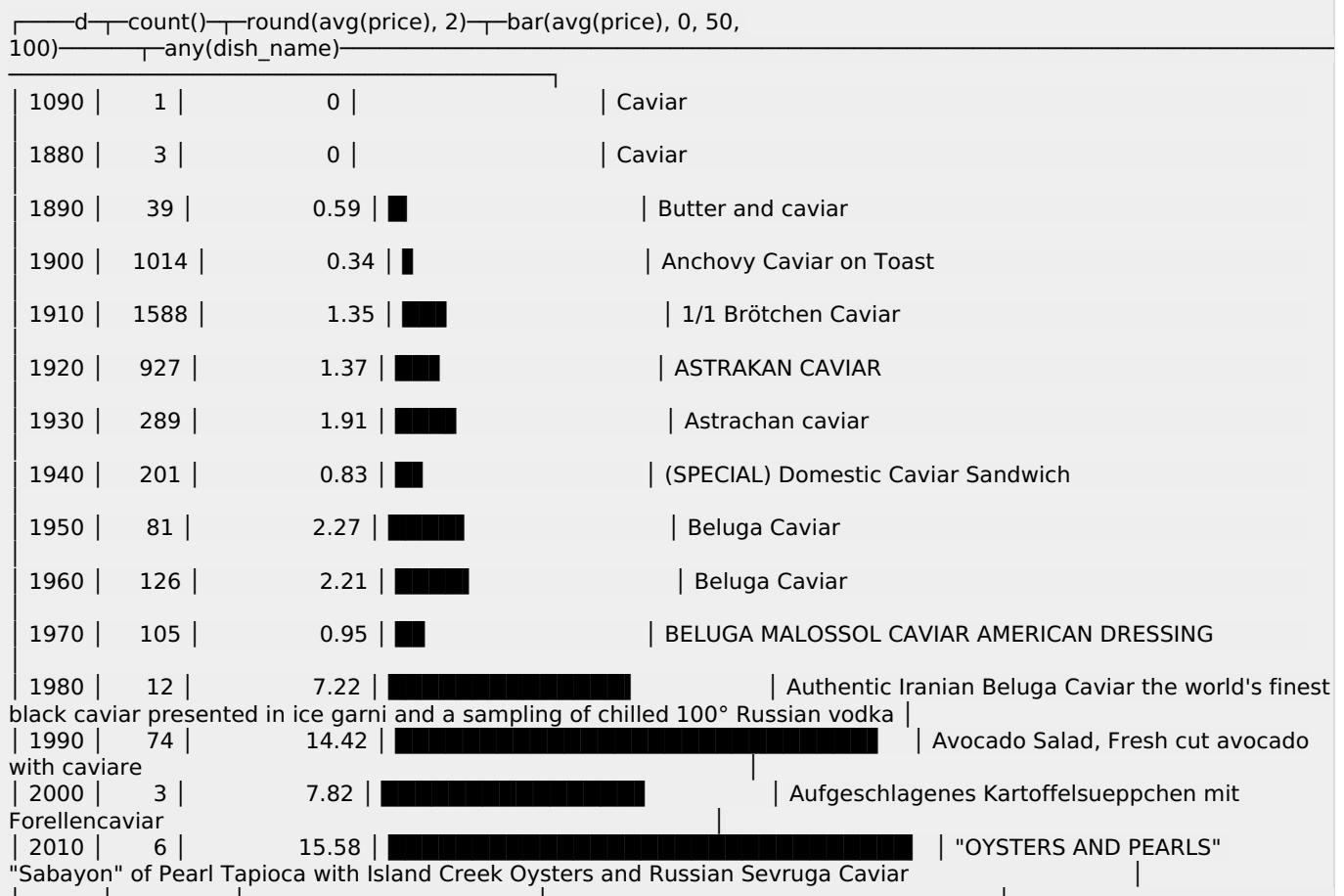
## Caviar

Let's print caviar prices. Also let's print a name of any dish with caviar.

Query:

```
SELECT
    round(toUInt32OrZero(extract(menu_date, '^\\d{4}')), -1) AS d,
    count(),
    round(avg(price), 2),
    bar(avg(price), 0, 50, 100),
    any(dish_name)
FROM menu_item_denorm
WHERE (menu_currency IN ('Dollars', '')) AND (d > 0) AND (d < 2022) AND (dish_name ILIKE '%caviar%')
GROUP BY d
ORDER BY d ASC;
```

Result:



At least they have caviar with vodka. Very nice.

## Online Playground

The data is uploaded to ClickHouse Playground, [example](#).

## ClickHouse Playground

ClickHouse Playground では、サーバーやクラスタを設定することなく、即座にクエリを実行して ClickHouse を試すことができます。

いくつかの例のデータセットは、Playground だけでなく、ClickHouse の機能を示すサンプルクエリとして利用可能です。また、ClickHouse の LTS リリースで試すこともできます。

ClickHouse Playground は、Yandex.Cloud にホストされている m2.small Managed Service for ClickHouse インスタンス(4 vCPU, 32 GB RAM) で提供されています。クラウドプロバイダの詳細情報については[こちら](#)。

任意の HTTP クライアントを使用してプレイグラウンドへのクエリを作成することができます。例えば curl、wget、JDBC または ODBC ドライバを使用して接続を設定します。

ClickHouse をサポートするソフトウェア製品の詳細情報は[こちら](#)をご覧ください。

## 資格情報

パラメータ	値
HTTPS エンドポイント	<a href="https://play-api.clickhouse.com:8443">https://play-api.clickhouse.com:8443</a>

パラメータ	値
ネイティブ TCP エンドポイント	play-api.clickhouse.com:9440
ユーザー名	playgrounnd
パスワード	clickhouse

特定のClickHouseのリリースで試すために、追加のエンドポイントがあります。（ポートとユーザー/パスワードは上記と同じです）。

- 20.3 LTS: play-api-v20-3.clickhouse.com
- 19.14 LTS: play-api-v19-14.clickhouse.com

## 備考

これらのエンドポイントはすべて、安全なTLS接続が必要です。

## 制限事項

クエリは読み取り専用のユーザとして実行されます。これにはいくつかの制限があります。

- DDL クエリは許可されていません。
- INSERT クエリは許可されていません。

また、以下の設定がなされています。

- `max_result_bytes=10485760`
- `max_result_rows=2000`
- `result_overflow_mode=break`
- `max_execution_time=60000`

## 例

curl を用いて HTTPS エンドポイントへ接続する例:

```
curl "https://play-api.clickhouse.com:8443/?query=SELECT+'Play+ClickHouse\!';&user=playground&password=clickhouse&database=datasets"
```

CLI で TCP エンドポイントへ接続する例:

```
clickhouse client --secure -h play-api.clickhouse.com --port 9440 -u playground --password clickhouse -q "SELECT 'Play ClickHouse\!'"
```

## 実装の詳細

ClickHouse PlaygroundのWebインターフェースは、ClickHouse [HTTP API](#)を介してリクエストを行います。Playgroundのパックエンドは、追加のサーバーサイドのアプリケーションを伴わない、ただのClickHouseクラスタです。上記のように、ClickHouse HTTPSとTCP/TLSのエンドポイントは Playground の一部としても公開されており、いずれも、上記の保護とよりよいグローバルな接続のためのレイヤを追加するために、[Cloudflare Spectrum](#) を介してプロキシされています。

## 注意

いかなる場合においても、インターネットにClickHouseサーバを公開することは [非推奨](#)です。  
プライベートネットワーク上でのみ接続を待機し、適切に設定されたファイアウォールによって保護されていることを確認してください。

## インター

ClickHouseについての知識とネットワークインターフェイス（両方できる任意に包まれたTLSのための追加のセキュリティ）：

- [HTTP](#)、文書化され、直接使用するのは簡単です。
- [ネイティブTCP](#) より少ない間接費がある。

るケースがほとんどでの使用をお勧めの適切なツールや図書館での交流と直結。公式にサポートしよYandexは次のとお：

- [コマンド行クライアント](#)
- [JDBC ドライバ](#)
- [ODBC ドライバ](#)
- [C++クライアントライ](#)

ClickHouseを操作するための幅広いサードパーティのライブラリもあります：

- [クライアント](#)
- [統合](#)
- [視界面](#)

## コマンド行クライアント

ClickHouseスネイティブコマンドラインのクライアント：[clickhouse-client](#). クライア 詳細については、[設定](#).

[インスト](#) それから `clickhouse-client` パッケージ化し、コマンドで実行します `clickhouse-client`.

```
$ clickhouse-client
ClickHouse client version 19.17.1.1579 (official build).
Connecting to localhost:9000 as user default.
Connected to ClickHouse server version 19.17.1 revision 54428.
```

:)

異なるクライアントとサーババージョンと互換性が、一部機能が利用できない古いクライアント。お使いになることをお勧めしま同じバージョンのクライアントのサーバーアプリです。古いバージョンのクライアントを使用しようすると、サーバー、`clickhouse-client` メッセージを表示する：

```
ClickHouse client version is older than ClickHouse server. It may lack support for new features.
```

## 使用法

クライアントは、対話型および非対話型(パッチ)モードで使用できます。パッチモードを使用するには、「query」変数は、またはデータをに送ります「stdin」（これは、「stdin」は端末ではない）、またはその両方。HTTPインターフェイスと同様に、「query」変数およびデータをに送ります「stdin」は、リクエストの連結である。「query」パラメータ、改行、およびデータ「stdin」。これは、大規模なINSERTクエリに便利です。

クライアントを使用してデータを挿入する例：

```
$ echo -ne "1, 'some text', '2016-08-14 00:00:00'\n2, 'some more text', '2016-08-14 00:00:01'" | clickhouse-client --database=test --query="INSERT INTO test FORMAT CSV";  
  
$ cat <<_EOF | clickhouse-client --database=test --query="INSERT INTO test FORMAT CSV";  
3, 'some text', '2016-08-14 00:00:00'  
4, 'some more text', '2016-08-14 00:00:01'  
_EOF  
  
$ cat file.csv | clickhouse-client --database=test --query="INSERT INTO test FORMAT CSV";
```

パッチモードでは、デフォルトのデータ形式はTabSeparatedです。クエリのFORMAT句で書式を設定できます。

既定では、パッチモードでのみ単一のクエリを処理できます。複数のクエリを作成するには“script,” 使用する`--multiquery` パラメータ。このため、すべてのクエリを除をサポートしていません。クエリ結果は、追加の区切り文字なしで連続して出力されます。同様に、多数のクエリを処理するには、次のように実行できます`'clickhouse-client'` 各クエリに対して。起動には数十ミリ秒かかることに注意してください。`'clickhouse-client'` プログラム

対話モードでは、クエリを入力できるコマンドラインを取得します。

もし ‘multiline’ 指定されていない(デフォルト):クエリを実行するには、Enterキーを押します。クエリの最後にセミコロンは必要ありません。複数行のクエリを入力するには、円記号を入力します\ ラインフィードの前に。Enterキーを押すと、クエリの次の行を入力するように求められます。

複数行が指定されている場合:クエリを実行するには、セミコロンで終了し、Enterキーを押します。入力した行の最後にセミコロンが省略されている場合は、クエリの次の行を入力するよう求められます。

単一のクエリのみが実行されるため、セミコロンの後のすべてが無視されます。

以下を指定できます\G セミコロンの代わりに、またはその後。これは、垂直形式を示します。この形式では、各値は別々の行に印刷されます。この異常な機能は、MySQL CLIとの互換性のために追加されました。

のコマンドラインに基づく‘replxx’（類似‘readline’）。つまり、使い慣れたキーボードショートカットを使用し、履歴を保持します。歴史はに書かれています`~/.clickhouse-client-history`。

既定では、使用される形式はPrettyCompactです。クエリのFORMAT句で書式を変更するか、次のように指定することができます\G クエリの最後に、`--format` または`--vertical` コマンドラインでの引数、またはクライアント構成ファイルの使用。

クライアントを終了するには、Ctrl+Dを押すか、クエリの代わりに次のいずれかを入力します：“exit”, “quit”, “logout”, “exit;”, “quit;”, “logout;”, “q”, “Q”, “:q”

が処理クエリー、クライアントを示し：

1. (デフォルトでは)毎秒10回を超えないように更新されます。クイッククエリの場合、進行状況を表示する時間がない場合があります。
2. 解析後、デバッグ用に書式設定されたクエリ。
3. 指定された形式の結果。
4. 結果の行数、経過した時間、およびクエリ処理の平均速度。

ただし、サーバーが要求を中止するのを少し待つ必要があります。特定の段階でクエリをキャンセルすることはできません。もう一度待ってCtrl+Cを押さないと、クライアントは終了します。

コマンドライ 詳細については “External data for query processing”.

## パラメータ

を作成でき、クエリパラメータおよびパスの値からのお知らクライアントアプリケーション。これを避けるフォーマットのクエリが特定の動的価値観にクライアント側で行われます。例えば:

```
$ clickhouse-client --param_parName="[1, 2]" -q "SELECT * FROM table WHERE a = {parName:Array(UInt16)}"
```

## クエリ構文

通常どおりにクエリを書式設定し、アプリパラメータからクエリに渡す値を次の形式の中っこで囲みます:

```
{<name>:<data type>}
```

- **name** — Placeholder identifier. In the console client it should be used in app parameters as `--param_<name> = value`.
- **data type** — データ型 アプリのパラメータ値のたとえば、次のようなデータ構造 (`integer, ('string', integer)`) を持つことができ `Tuple(UInt8, Tuple(String, UInt8))` データ型(別のデータ型も使用できます 整数 タイプ)。

## 例

```
$ clickhouse-client --param_tuple_in_tuple="(10, ('dt', 10))" -q "SELECT * FROM table WHERE val = {tuple_in_tuple:Tuple(UInt8, Tuple(String, UInt8))}"
```

## 設定

パラメータを渡すには `clickhouse-client` (すべてのパラメータには既定値があります):

- コマンドラインから  
コマンドラインオプションは、構成ファイルの既定値と設定を上書きします。
- 設定ファイル。  
構成ファイルの設定は、既定値を上書きします。

## コマンドライ

- `--host, -h` — The server name, ‘localhost’ デフォルトでは。名前またはIPv4またはIPv6アドレスを使用できます。
- `--port` — The port to connect to. Default value: 9000. Note that the HTTP interface and the native interface use different ports.
- `--user, -u` — The username. Default value: default.
- `--password` — The password. Default value: empty string.
- `--query, -q` — The query to process when using non-interactive mode.

- `--database, -d` – Select the current default database. Default value: the current database from the server settings ('default' デフォルトでは)。
- `--multiline, -m` – If specified, allow multiline queries (do not send the query on Enter).
- `--multiquery, -n` – If specified, allow processing multiple queries separated by semicolons.
- `--format, -f` – Use the specified default format to output the result.
- `--vertical, -E` – If specified, use the Vertical format by default to output the result. This is the same as '`-format=Vertical`'. この形式では、各値は別々の行に印刷されます。
- `--time, -t` – If specified, print the query execution time to 'stderr' 非対話型モードで。
- `--stacktrace` – If specified, also print the stack trace if an exception occurs.
- `--config-file` – The name of the configuration file.
- `--secure` – If specified, will connect to server over secure connection.
- `--param_<name>` — Value for a パラメー。

## 設定ファイル

`clickhouse-client` 次の最初の既存のファイルを使用します:

- で定義される。`--config-file` パラメータ。
- `./clickhouse-client.xml`
- `~/.clickhouse-client/config.xml`
- `/etc/clickhouse-client/config.xml`

設定ファイルの例:

```
<config>
  <user>username</user>
  <password>password</password>
  <secure>False</secure>
</config>
```

## ネイティブ(

ネイティブプロトコルは [コマンド行クライアント](#)、分散クエリ処理中のサーバー間通信、および他のC++プログラムでも使用できます。残念ながら、ネイティブClickHouseプロトコルは形式的仕様記述を利用できるバージョンからClickHouseソースコードを開始 [この辺り](#) や傍受し、分析すTCPます。

## HTTPインターフェ

HTTPイトのご利用ClickHouseにプラットフォームからゆるプログラミング言語です。JavaやPerl、シェルスクリプトからの作業に使用します。他の部門では、HttpインターフェイスはPerl、Python、Goから使用されます。HTTPイデフォルトでは、`clickhouse-server`はポート8123でHTTPをリッスンします（これは設定で変更できます）。

パラメータなしでGET/requestを行うと、200の応答コードとで定義された文字列が返されます  
`http_server_default_response` デフォルト値 “Ok.” (最後に改行があります)

```
$ curl 'http://localhost:8123/'  
Ok.
```

ヘルスチェックスクリプトでGET/ping要求を使用します。このハンドラは常に“Ok.”（最後にラインフィード付き）。バージョン18.12.13から利用可能。

```
$ curl 'http://localhost:8123/ping'  
Ok.
```

リクエストをURLとして送信する‘query’パラメータ、またはポストとして。または、クエリの先頭を‘query’パラメータ、およびポストの残りの部分（これが必要な理由を後で説明します）。URLのサイズは16KBに制限されているため、大規模なクエリを送信する場合はこの点に注意してください。

成功すると、200の応答コードとその結果が応答本文に表示されます。

エラーが発生すると、応答本文に500応答コードとエラー説明テキストが表示されます。

GETメソッドを使用する場合、‘readonly’設定されています。つまり、データを変更するクエリでは、POSTメソッドのみを使用できます。クエリ自体は、POST本文またはURLパラメータのいずれかで送信できます。

例：

```
$ curl 'http://localhost:8123/?query=SELECT%201'  
1  
  
$ wget -nv -O- 'http://localhost:8123/?query=SELECT 1'  
1  
  
$ echo -ne 'GET /?query=SELECT%201 HTTP/1.0\r\n\r\n' | nc localhost 8123  
HTTP/1.0 200 OK  
Date: Wed, 27 Nov 2019 10:30:18 GMT  
Connection: Close  
Content-Type: text/tab-separated-values; charset=UTF-8  
X-ClickHouse-Server-Display-Name: clickhouse.ru-central1.internal  
X-ClickHouse-Query-Id: 5abe861c-239c-467f-b955-8a201abb8b7f  
X-ClickHouse-Summary:  
{"read_rows":"0","read_bytes":"0","written_rows":"0","written_bytes":"0","total_rows_to_read":"0"}  
1
```

ご覧の通り、カールはやや不便ですがそのスペースするURLが自動的にエスケープされます。

Wgetはすべてそのものをエスケープしますが、keep-aliveとTransfer-Encoding:chunkedを使用する場合、HTTP1.1よりもうまく動作しないため、使用することはお勧めしません。

```
$ echo 'SELECT 1' | curl 'http://localhost:8123/' --data-binary @-  
1  
  
$ echo 'SELECT 1' | curl 'http://localhost:8123/?query=' --data-binary @-  
1  
  
$ echo '1' | curl 'http://localhost:8123/?query=SELECT' --data-binary @-  
1
```

クエリの一部がパラメータで送信され、投稿の一部が送信されると、これらの二つのデータ部分の間に改行が挿入されます。

例（これは動作しません）：

```
$ echo 'ECT 1' | curl 'http://localhost:8123/?query=SEL' --data-binary @-  
Code: 59, e.displayText() = DB::Exception: Syntax error: failed at position 0: SEL  
ECT 1  
, expected One of: SHOW TABLES, SHOW DATABASES, SELECT, INSERT, CREATE, ATTACH, RENAME, DROP, DETACH,  
USE, SET, OPTIMIZE., e.what() = DB::Exception
```

デフォルトでは、データはTabSeparated形式で返されます(詳細については、“Formats”セクション)。他の形式を要求するには、クエリのFORMAT句を使用します。

```
$ echo 'SELECT 1 FORMAT Pretty' | curl 'http://localhost:8123/?' --data-binary @-
```

```
1  
1
```

データを送信するPOSTメソッドは、INSERTクエリに必要です。この場合、URLパラメーターにクエリの先頭を記述し、POSTを使用して挿入するデータを渡すことができます。挿入するデータは、たとえば、MySQLからのタブ区切りのダンプです。このようにして、INSERTクエリはMySQLからのLOAD DATA LOCAL INFILEを置き換えます。

例: テーブルの作成:

```
$ echo 'CREATE TABLE t (a UInt8) ENGINE = Memory' | curl 'http://localhost:8123/' --data-binary @-
```

データ挿入のための使い慣れたINSERTクエリの使用:

```
$ echo 'INSERT INTO t VALUES (1),(2),(3)' | curl 'http://localhost:8123/' --data-binary @-
```

データはクエリとは別に送信できます:

```
$ echo '(4),(5),(6)' | curl 'http://localhost:8123/?query=INSERT%20INTO%20t%20VALUES' --data-binary @-
```

任意のデータ形式を指定できます。その‘Values’書式は、INSERTをt値に書き込むときに使用される書式と同じです:

```
$ echo '(7),(8),(9)' | curl 'http://localhost:8123/?query=INSERT%20INTO%20t%20FORMAT%20Values' --data-binary @-
```

タブ区切りのダンプからデータを挿入するには、対応する形式を指定します:

```
$ echo -ne '10\n11\n12\n' | curl 'http://localhost:8123/?query=INSERT%20INTO%20t%20FORMAT%20TabSeparated' --data-binary @-
```

テーブルの内容を読む。並列クエリ処理により、データがランダムに出力されます:

```
$ curl 'http://localhost:8123/?query=SELECT%20a%20FROM%20t'
```

```
7  
8  
9  
10  
11  
12  
1  
2  
3  
4  
5  
6
```

テーブルの削除。

```
$ echo 'DROP TABLE t' | curl 'http://localhost:8123/' --data-binary @-
```

データテーブルを返さない正常な要求の場合、空の応答本文が返されます。

データを送信するときは、内部ClickHouse圧縮形式を使用できます。圧縮されたデータは非標準形式であり、特別な形式を使用する必要があります `clickhouse-compressor` それで動作するようにプログラム（それは `clickhouse-client` パッケージ）。データ挿入の効率を高めるために、以下を使用してサーバー側のチェックサム検証を無効にできます `http_native_compression_disable_checksumming_on_decompress` 設定。

指定した場合 `compress=1` URLでは、サーバーが送信するデータを圧縮します。

指定した場合 `decompress=1` このURLでは、サーバーは渡すデータと同じデータを解凍します。 `POST` 方法。

また、使用することもできます **HTTP圧縮**。圧縮を送信するには `POST` リクエストヘッダーを追加します `Content-Encoding: compression_method`。ClickHouseが応答を圧縮するには、次のように追加する必要があります `Accept-Encoding: compression_method`。ClickHouseサポート `gzip`, `br`, and `deflate` **圧縮方法**。HTTP圧縮を有効にするには、ClickHouseを使用する必要があります `enable_http_compression` 設定。データ圧縮レベルは、`http_zlib_compression_level` すべての圧縮方法の設定。

利用することができ削減ネットワーク通信の送受信には大量のデータをダンプすると直ちに圧縮されます。

圧縮によるデータ送信の例:

```
## Sending data to the server:  
$ curl -vsS "http://localhost:8123/?enable_http_compression=1" -d 'SELECT number FROM system.numbers LIMIT 10' -H 'Accept-Encoding: gzip'  
  
## Sending data to the client:  
$ echo "SELECT 1" | gzip -c | curl -sS --data-binary @- -H 'Content-Encoding: gzip' 'http://localhost:8123/'
```

## 注

あるHTTPお客様が圧縮解除データからサーバによるデフォルト（`gzip` と `deflate`）圧縮設定を正しく使用しても、圧縮解除されたデータが得られることがあります。

を使用することができます ‘database’ 既定のデータベースを指定するURLパラメータ。

```
$ echo 'SELECT number FROM numbers LIMIT 10' | curl 'http://localhost:8123/?database=system' --data-binary @-  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

既定では、サーバー設定に登録されているデータベースが既定のデータベースとして使用されます。デフォルトでは、このデータベース ‘default’。または、テーブル名の前にドットを使用してデータベースを常に指定できます。

ユーザー名とパスワードは、次のいずれかの方法で指定できます:

1. HTTP基本認証の使用。例:

```
$ echo 'SELECT 1' | curl 'http://user:password@localhost:8123/' -d @-
```

1. で ‘user’ と ‘password’ URLパラメータ。例:

```
$ echo 'SELECT 1' | curl 'http://localhost:8123/?user=user&password=password' -d @-
```

1. を使用して ‘X-ClickHouse-User’ と ‘X-ClickHouse-Key’ ヘッダー。例:

```
$ echo 'SELECT 1' | curl -H 'X-ClickHouse-User: user' -H 'X-ClickHouse-Key: password' 'http://localhost:8123/' -d @-
```

ユーザ名が指定されていない場合、`default` 名前が使用されます。パスワードを指定しない場合は、空のパスワードが使用されます。

にお使いいただけますURLパラメータで指定した設定処理の單一クエリーまたは全体をプロファイルを設定します。

例:[http://localhost:8123/?profile=web&max\\_rows\\_to\\_read=1000000000&query=SELECT+1](http://localhost:8123/?profile=web&max_rows_to_read=1000000000&query=SELECT+1)

詳細については、を参照してください [設定](#) セクション

```
$ echo 'SELECT number FROM system.numbers LIMIT 10' | curl 'http://localhost:8123/?' --data-binary @-
0
1
2
3
4
5
6
7
8
9
```

そのための情報その他のパラメータの項をご参考ください “SET”。

同様に、HttpプロトコルでClickHouseセッションを使用できます。これを行うには、`session_id` 要求のパラメータを取得します。セッションIDとして任意の文字列を使用できます。既定では、セッションは非アクティブの60秒後に終了します。このタイムアウトを変更するには、`default_session_timeout` サーバー構成で設定するか、または`session_timeout` 要求のパラメータを取得します。セッションステータスを確認するには、`session_check=1` パラメータ。単一のセッション内で一度に一つのクエリだけを実行できます。

クエリの進行状況に関する情報を受け取ることができます `X-ClickHouse-Progress` 応答ヘッダー。これを行うには、有効にします `send_progress_in_http_headers` ヘッダシーケンスの例:

```
X-ClickHouse-Progress: {"read_rows": "2752512", "read_bytes": "240570816", "total_rows_to_read": "8880128"}
X-ClickHouse-Progress: {"read_rows": "5439488", "read_bytes": "482285394", "total_rows_to_read": "8880128"}
X-ClickHouse-Progress: {"read_rows": "8783786", "read_bytes": "819092887", "total_rows_to_read": "8880128"}
```

可能なヘッダ項目:

- `read_rows` — Number of rows read.
- `read_bytes` — Volume of data read in bytes.
- `total_rows_to_read` — Total number of rows to be read.
- `written_rows` — Number of rows written.
- `written_bytes` — Volume of data written in bytes.

HTTP接続が失われても、要求の実行は自動的に停止しません。解析とデータの書式設定はサーバー側で実行され、ネットワークを使用すると無効になる可能性があります。

任意 ‘query\_id’ パラメータは、クエリID(任意の文字列)として渡すことができます。詳細については “Settings, replace\_running\_query”。

任意 ‘quota\_key’ パラメータとして渡すことができ、クォーターキー(切文字列)。詳細については “Quotas”。

HTTPイ 詳細については “External data for query processing”。

## 応答バッファリング

サーバー側で応答バッファリングを有効にできます。その `buffer_size` と `wait_end_of_query` URL パラメータを提供しています。

`buffer_size` サーバーメモリ内のバッファーに格納する結果のバイト数を決定します。結果の本文がこのしきい値より大きい場合、バッファは HTTP チャネルに書き込まれ、残りのデータは HTTP チャネルに直接送信されます。

保全の対応はバッファ処理されますが、設定 `wait_end_of_query=1`。この場合、メモリに格納されていないデータは一時サーバーファイルにバッファされます。

例:

```
$ curl -sS 'http://localhost:8123/?max_result_bytes=4000000&buffer_size=3000000&wait_end_of_query=1' -d 'SELECT toUInt8(number) FROM system.numbers LIMIT 9000000 FORMAT RowBinary'
```

使用バッファリングなどでクエリの処理エラーが発生した後は、応答コード、HTTP ヘッダが送信されます。この状況では、応答本文の最後にエラーメッセージが書き込まれ、クライアント側では、エラーは解析段階でのみ検出できます。

## パラメータ

パラメータを使用してクエリを作成し、対応する HTTP 要求パラメータから値を渡すことができます。詳細については、[CLI のパラメータを使用した照会](#)。

例

```
$ curl -sS "<address>?param_id=2&param_phrase=test" -d "SELECT * FROM table WHERE int_column = {id:UInt8} and string_column = {phrase:String}"
```

## 所定の HTTP ス

ClickHouse 支持特定のクエリは HTTP インターフェース。たとえば、次のように表にデータを書き込むことができます:

```
$ echo '(4),(5),(6)' | curl 'http://localhost:8123/?query=INSERT%20INTO%20t%20VALUES' --data-binary @-
```

ClickHouse にも対応した所定の HTTP インターフェースができあがとの統合に第三者のリーディングプロジェクト [プロメテウス](#) 輸出。

例:

- まず、このセクションをサーバー設定ファイルに追加します:

```
<http_handlers>
  <rule>
    <url>/predefined_query</url>
    <methods>POST,GET</methods>
    <handler>
      <type>predefined_query_handler</type>
      <query>SELECT * FROM system.metrics LIMIT 5 FORMAT Template SETTINGS format_template_resultset =
'prometheus_template_output_format_resultset', format_template_row = 'prometheus_template_output_format_row',
format_template_rows_between_delimiter = '\n'</query>
    </handler>
  </rule>
  <rule>...</rule>
  <rule>...</rule>
</http_handlers>
```

- Prometheus 形式のデータの url を直接リクエストできるようになりました:

```

$ curl -v 'http://localhost:8123/predefined_query'
* Trying ::1...
* Connected to localhost (::1) port 8123 (#0)
> GET /predefined_query HTTP/1.1
> Host: localhost:8123
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Tue, 28 Apr 2020 08:52:56 GMT
< Connection: Keep-Alive
< Content-Type: text/plain; charset=UTF-8
< X-ClickHouse-Server-Display-Name: i-mloy5trc
< Transfer-Encoding: chunked
< X-ClickHouse-Query-Id: 96fe0052-01e6-43ce-b12a-6b7370de6e8a
< X-ClickHouse-Format: Template
< X-ClickHouse-Timezone: Asia/Shanghai
< Keep-Alive: timeout=3
< X-ClickHouse-Summary:
{"read_rows":"0","read_bytes":"0","written_rows":"0","written_bytes":"0","total_rows_to_read":"0"}
<
## HELP "Query" "Number of executing queries"
## TYPE "Query" counter
"Query" 1

## HELP "Merge" "Number of executing background merges"
## TYPE "Merge" counter
"Merge" 0

## HELP "PartMutation" "Number of mutations (ALTER DELETE/UPDATE)"
## TYPE "PartMutation" counter
"PartMutation" 0

## HELP "ReplicatedFetch" "Number of data parts being fetched from replica"
## TYPE "ReplicatedFetch" counter
"ReplicatedFetch" 0

## HELP "ReplicatedSend" "Number of data parts being sent to replicas"
## TYPE "ReplicatedSend" counter
"ReplicatedSend" 0

* Connection #0 to host localhost left intact

* Connection #0 to host localhost left intact

```

この例からわかるように、`<http_handlers>` 設定で構成されています。xmlファイルと`<http_handlers>`を含むことができ多くの`<rule>`s. ClickHouseは、受信したHTTP要求を事前定義されたタイプに一致させます`<rule>`最初に一致したハンドラが実行されます。一致が成功すると、ClickHouseは対応する事前定義されたクエリを実行します。

さて `<rule>` 構成できます `<method>`, `<headers>`, `<url>`, `<handler>`:

`<method>` HTTP要求のメソッド部分の照合を担当します。 `<method>` 十分に定義にの合致します **方法** HTTPプロトコルで。 これはオプションの構成です。 構成ファイルで定義されていない場合、HTTP要求のメソッド部分と一致しません。

`<url>` HTTPリクエストのurl部分の照合を担当します。 それはと互換性があります **RE2**の正規表現。 これはオプションの構成です。 構成ファイルで定義されていない場合、HTTP要求のurl部分と一致しません。

`<headers>` HTTPリクエストのヘッダー部分の照合を担当します。 **RE2**の正規表現と互換性があります。 これはオプションの構成です。 構成ファイルで定義されていない場合、HTTP要求のヘッダー部分と一致しません。

`<handler>` 主要な処理の部品を含んでいます。 さて `<handler>` 構成できます `<type>`, `<status>`, `<content_type>`, `<response_content>`, `<query>`, `<query_param_name>`.

> `<type>` 現在サポート : **predefined\_query\_handler**, **dynamic\_query\_handler**, 静的.

>

> `<query>` -`predefined_query_handler`型で使用し、ハンドラが呼び出されたときにクエリを実行します。

>

> `<query_param_name>` -`dynamic_query_handler`型で使用すると、それに対応する値を抽出して実行します。

<`query_param_name`> HTTP要求パラメータの値。

>

> `<status>` -静的タイプ、応答ステータスコードで使用します。

>

> `<content_type>` -静的なタイプ、応答との使用 **コンテンツタイプ**.

>

> `<response_content>` 使用静止型で、応答が送信（発信）したコンテンツをクライアントでご使用になる場合、接頭辞 'file://' または 'config://'、クライアントに送信するファイルまたは構成から内容を検索します。

次に、異なる設定方法を示します `<type>`.

## predefined\_query\_handler

`<predefined_query_handler>` 設定と`query_params`値の設定をサポートします。 設定できます `<query>` のタイプで `<predefined_query_handler>`.

`<query>` 値は以下の定義済みクエリです `<predefined_query_handler>` これは、Http要求が一致し、クエリの結果が返されたときにClickHouseによって実行されます。 これは必須構成です。

次の例では、次の値を定義します `max_threads` と `max_alter_threads` 設定、そしてクエリのテーブルから設定設定します。

例:

```
<http_handlers>
  <rule>
    <url><![CDATA[/query_param_with_url/w+/(?P<name_1>[^/]+)(/(?P<name_2>[^/]+))?]></url>
    <method>GET</method>
    <headers>
      <XXX>TEST_HEADER_VALUE</XXX>
      <PARAMS_XXX><![CDATA[(?P<name_1>[^/]+)(/(?P<name_2>[^/]+))?]></PARAMS_XXX>
    </headers>
    <handler>
      <type>predefined_query_handler</type>
      <query>SELECT value FROM system.settings WHERE name = {name_1:String}</query>
      <query>SELECT name, value FROM system.settings WHERE name = {name_2:String}</query>
    </handler>
  </rule>
</http_handlers>
```

```
$ curl -H 'XXX:TEST_HEADER_VALUE' -H 'PARAMS_XXX:max_threads'
'http://localhost:8123/query_param_with_url/1/max_threads/max_alter_threads?
max_threads=1&max_alter_threads=2'
1
max_alter_threads 2
```

## 注意

一つで `<predefined_query_handler>` 一つだけをサポート `<query>` 挿入タイプ。

## dynamic\_query\_handler

で `<dynamic_query_handler>`、クエリは、HTTP要求のparamの形式で書かれています。違いは、  
`<predefined_query_handler>`、クエリは設定ファイルに書き込まれます。設定できます `<query_param_name>` で  
`<dynamic_query_handler>`。

クリックハウスは、`<query_param_name>` HTTP要求のurlの値。のデフォルト値 `<query_param_name>` は `/query`。これはオプションの構成です。設定ファイルに定義がない場合、paramは渡されません。

この機能を試すために、この例では `max_threads` と `max_alter_threads` の値を定義し、設定が正常に設定されたかどうかを照会します。

例:

```
<http_handlers>
  <rule>
    <headers>
      <XXX>TEST_HEADER_VALUE_DYNAMIC</XXX>  </headers>
    <handler>
      <type>dynamic_query_handler</type>
      <query_param_name>query_param</query_param_name>
    </handler>
  </rule>
</http_handlers>
```

```
$ curl -H 'XXX:TEST_HEADER_VALUE_DYNAMIC' 'http://localhost:8123/own?
max_threads=1&max_alter_threads=2&param_name_1=max_threads&param_name_2=max_alter_threads&query_p
aram=SELECT%20name,value%20FROM%20system.settings%20where%20name%20=%20%7Bname_1:String%7D%
20OR%20name%20=%20%7Bname_2:String%7D'
max_threads 1
max_alter_threads 2
```

## 静的

`<static>` 戻れる `content_type`, 状態 そして `response_content`。`response_content` は、指定された内容を返すことができます

例:

メッセージを返す。

```
<http_handlers>
  <rule>
    <methods>GET</methods>
    <headers><XXX>xxx</XXX></headers>
    <url>/hi</url>
    <handler>
      <type>static</type>
      <status>402</status>
      <content_type>text/html; charset=UTF-8</content_type>
      <response_content>Say Hi!</response_content>
    </handler>
  </rule>
</http_handlers>
```

```
$ curl -vv -H 'XXX:xxx' 'http://localhost:8123/hi'
* Trying ::1...
* Connected to localhost (::1) port 8123 (#0)
> GET /hi HTTP/1.1
> Host: localhost:8123
> User-Agent: curl/7.47.0
> Accept: /*
> XXX:xxx
>
< HTTP/1.1 402 Payment Required
< Date: Wed, 29 Apr 2020 03:51:26 GMT
< Connection: Keep-Alive
< Content-Type: text/html; charset=UTF-8
< Transfer-Encoding: chunked
< Keep-Alive: timeout=3
< X-ClickHouse-Summary:
{ "read_rows": "0", "read_bytes": "0", "written_rows": "0", "written_bytes": "0", "total_rows_to_read": "0" }
<
* Connection #0 to host localhost left intact
Say Hi!%
```

のコンテンツから構成送信します。

```
<get_config_static_handler><![CDATA[<html ng-app="SMI2"><head><base href="http://ui.tabix.io/"></head>
<body><div ui-view="" class="content-ui"></div><script src="http://loader.tabix.io/master.js"></script></body>
</html>]]></get_config_static_handler>

<http_handlers>
  <rule>
    <methods>GET</methods>
    <headers><XXX>xxx</XXX></headers>
    <url>/get_config_static_handler</url>
    <handler>
      <type>static</type>
      <response_content>config://get_config_static_handler</response_content>
    </handler>
  </rule>
</http_handlers>
```

```
$ curl -v -H 'XXX:xxx' 'http://localhost:8123/get_config_static_handler'
* Trying ::1...
* Connected to localhost (::1) port 8123 (#0)
> GET /get_config_static_handler HTTP/1.1
> Host: localhost:8123
> User-Agent: curl/7.47.0
> Accept: */*
> XXX:xxx
>
< HTTP/1.1 200 OK
< Date: Wed, 29 Apr 2020 04:01:24 GMT
< Connection: Keep-Alive
< Content-Type: text/plain; charset=UTF-8
< Transfer-Encoding: chunked
< Keep-Alive: timeout=3
< X-ClickHouse-Summary:
{"read_rows":"0","read_bytes":"0","written_rows":"0","written_bytes":"0","total_rows_to_read":"0"}
<
* Connection #0 to host localhost left intact
<html ng-app="SMI2"><head><base href="http://ui.tabix.io/"></head><body><div ui-view="" class="content-ui"></div><script src="http://loader.tabix.io/master.js"></script></body></html>%
```

クライアントに送信するファイルから内容を検索します。

```
<http_handlers>
  <rule>
    <methods>GET</methods>
    <headers><XXX>xxx</XXX></headers>
    <url>/get_absolute_path_static_handler</url>
    <handler>
      <type>static</type>
      <content_type>text/html; charset=UTF-8</content_type>
      <response_content>file:///absolute_path_file.html</response_content>
    </handler>
  </rule>
  <rule>
    <methods>GET</methods>
    <headers><XXX>xxx</XXX></headers>
    <url>/get_relative_path_static_handler</url>
    <handler>
      <type>static</type>
      <content_type>text/html; charset=UTF-8</content_type>
      <response_content>file:///relative_path_file.html</response_content>
    </handler>
  </rule>
</http_handlers>
```

```
$ user_files_path='/var/lib/clickhouse/user_files'
$ sudo echo "<html><body>Relative Path File</body></html>" > $user_files_path/relative_path_file.html
$ sudo echo "<html><body>Absolute Path File</body></html>" > $user_files_path/absolute_path_file.html
$ curl -vv -H 'XXX:xxx' 'http://localhost:8123/get_absolute_path_static_handler'
* Trying ::1...
* Connected to localhost (::1) port 8123 (#0)
> GET /get_absolute_path_static_handler HTTP/1.1
> Host: localhost:8123
> User-Agent: curl/7.47.0
> Accept: /*
> XXX:xxx
>
< HTTP/1.1 200 OK
< Date: Wed, 29 Apr 2020 04:18:16 GMT
< Connection: Keep-Alive
< Content-Type: text/html; charset=UTF-8
< Transfer-Encoding: chunked
< Keep-Alive: timeout=3
< X-ClickHouse-Summary:
>{"read_rows":"0","read_bytes":"0","written_rows":"0","written_bytes":"0","total_rows_to_read":"0"}
<
<html><body>Absolute Path File</body></html>
* Connection #0 to host localhost left intact
$ curl -vv -H 'XXX:xxx' 'http://localhost:8123/get_relative_path_static_handler'
* Trying ::1...
* Connected to localhost (::1) port 8123 (#0)
> GET /get_relative_path_static_handler HTTP/1.1
> Host: localhost:8123
> User-Agent: curl/7.47.0
> Accept: /*
> XXX:xxx
>
< HTTP/1.1 200 OK
< Date: Wed, 29 Apr 2020 04:18:31 GMT
< Connection: Keep-Alive
< Content-Type: text/html; charset=UTF-8
< Transfer-Encoding: chunked
< Keep-Alive: timeout=3
< X-ClickHouse-Summary:
>{"read_rows":"0","read_bytes":"0","written_rows":"0","written_bytes":"0","total_rows_to_read":"0"}
<
<html><body>Relative Path File</body></html>
* Connection #0 to host localhost left intact
```

## MySQL インタ

ClickHouseはMySQL wire protocolをサポートしています。で有効にすることができます [mysql\\_port](#) 設定ファイルでの設定:

```
<mysql_port>9004</mysql_port>
```

コマンドラインツールを使用した接続例 mysql:

```
$ mysql --protocol tcp -u default -P 9004
```

接続が成功した場合の出力:

```
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 4  
Server version: 20.2.1.1-ClickHouse
```

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
mysql>
```

との互換性を維持するため、すべてのMySQLのお客様におすすめで指定ユーザのパスワード **ダブルSHA1** 設定ファイル。

場合は、ユーザのパスワードが指定 **SHA256** 一部のクライアントは認証できません（mysqljsおよび古いバージョンのコマンドラインツールmysql）。

制限:

- 作成問合せには対応していない
- 一部のデータ型は文字列として送信されます

## 入出力データのフォーマット

ClickHouse受け入れと返信データなフレームワークです。入力用にサポートされている形式を使用して、指定されたデータを解析できます **INSERTs**、実行する **SELECT**ファイル、URL、HDFSなどのファイルバックアップテーブルから、または外部辞書を読み取ることができます。出力でサポートされているフォーマットを使用して、  
aの結果 **SELECT**、および実行する **INSERT**ファイルバックアップテーブルにs。

のサポートされるフォーマットは:

形式	入力	出力
TabSeparated	✓	✓
TabSeparatedRaw	✗	✓
TabSeparatedWithNames	✓	✓
TabSeparatedWithNamesAndTypes	✓	✓
テンプレー	✓	✓
TemplateIgnoreSpaces	✓	✗
CSV	✓	✓
CSVWithNames	✓	✓
カスタム区切り	✓	✓
値	✓	✓

形式	入力	出力
垂直	✗	✓
JSON	✗	✓
JSONCompact	✗	✓
JSONEachRow	✓	✓
TSKV	✓	✓
プリティ	✗	✓
プリティコンパクト	✗	✓
プリティコンパクトモノブロック	✗	✓
プリティノスケープ	✗	✓
PrettySpace	✗	✓
プロトブフ	✓	✓
アヴロ	✓	✓
アプロコンフルエント	✓	✗
寄木細工	✓	✓
ORC	✓	✗
ローバイナリ	✓	✓
RowBinaryWithNamesAndTypes	✓	✓
ネイティブ	✓	✓
Null	✗	✓
XML	✗	✓
CapnProto	✓	✗

ClickHouse設定を使用して、いくつかの形式処理パラメータを制御できます。 詳細については、[設定](#) セクション

## TabSeparated

TabSeparated形式では、データは行ごとに書き込まれます。各行にはタブで区切られた値が含まれます。各値の後にはタブが続きますが、行の最後の値の後には改行が続きます。厳正なUnixラインフィードと仮定します。最後の行には、最後に改行も含める必要があります。値は、引用符を囲まずにテキスト形式で書き込まれ、特殊文字はエスケープされます。

この形式は、名前でも使用できます **TSV**.

その **TabSeparated** 形式は便利な加工データをカスタムプログラムやイントロダクションです。これは、HTTPインターフェイスおよびコマンドラインクライアントのバッチモードで既定で使用されます。この形式は、異なるDbms間でデータを転送することもできます。たとえば、MySQLからダンプを取得してClickHouseにアップロードすることも、その逆も可能です。

その **TabSeparated** 書式では、合計値(合計と共に使用する場合)と極値(場合)の出力がサポートされます ‘extremes’ 1) に設定される。このような場合、合計値と極値がメインデータの後に出力されます。主な結果、合計値、および極値は、空の行で区切られます。例:

```
SELECT EventDate, count() AS c FROM test.hits GROUP BY EventDate WITH TOTALS ORDER BY EventDate FORMAT TabSeparated``
```

```
2014-03-17 1406958
2014-03-18 1383658
2014-03-19 1405797
2014-03-20 1353623
2014-03-21 1245779
2014-03-22 1031592
2014-03-23 1046491

1970-01-01 8873898

2014-03-17 1031592
2014-03-23 1406958
```

## データの書式設定

整数は小数点形式で書かれています。番号を含むことができ追加 “+” 先頭の文字(解析時は無視され、書式設定時は記録されません)。負でない数値には負符号を含めることはできません。読み取り時には、空の文字列をゼロとして解析するか、(符号付き型の場合)マイナス記号だけで構成される文字列をゼロとして解析することができ 対応するデータ型に適合しない数値は、エラーメッセージなしで別の数値として解析される可能性があります。

浮動小数点数は小数点形式で記述されます。ドットは小数点区切り記号として使用されます。指数エントリがサポートされています。‘inf’, ‘+inf’, ‘-inf’, and ‘nan’. 浮動小数点数のエントリは、小数点で始まつたり終わつたりすることができます。

書式設定中に、浮動小数点数の精度が失われる可能性があります。

解析中に、最も近いマシン表現可能な番号を読み取ることは厳密には必要ありません。

日付はYYYY-MM-DD形式で記述され、同じ形式で解析されますが、区切り文字として任意の文字が使用されます。

時刻付きの日付は、次の形式で記述されます YYYY-MM-DD hh:mm:ss 同じ形式で解析されますが、区切り文字として任意の文字で解析されます。

これはすべて、クライアントまたはサーバーが起動したときのシステムタイムゾーンで発生します(データの形式に応じて)。時刻のある日付の場合、夏時間は指定されません。したがって、夏時間の間にダンプに時間がある場合、ダンプはデータと明確に一致しないため、解析は二回のいずれかを選択します。

読み取り操作中に、エラーメッセージなしで、自然なオーバーフローまたはnullの日付と時刻として、不適切な日付と時刻を解析することができます。

例外として、正確に10桁の数字で構成されている場合は、時刻を含む日付の解析もUnixタイムスタンプ形式でサポートされます。結果はタイムゾーンに依存しません。YYYY-MM-DD hh:mm:ssとNNNNNNNNNNの形式は自動的に区別されます。

文字列はバックスラッシュでエスケープされた特殊文字で出力されます。以下のエスケープシーケンスを使用出力: \b, \f, \r, \n, \t, \0, \', \\. 解析にも対応し配列 \a, \v, and \xHH (hexエスケープシーケンス)と \c シーケンス c は任意の文字です(これらのシーケンスは c). このように、データを読み込む形式に対応し、改行して書き込み可能で \n または \ または改行として。たとえば、文字列 Hello world スペースの代わりに単語の間の改行を使用すると、次のいずれかのバリエーションで解析できます:

```
Hello\nworld
```

```
Hello\  
world
```

の変異体でサポートしてMySQLで書く時にタブ区切り捨て場。

TabSeparated形式でデータを渡すときにエスケープする必要がある最小文字セット:tab、改行(LF)、およびバックスラッシュ。

のみの小さなセットの記号は自動的にエスケープされます。あなたの端末が出力で台無しにする文字列値に簡単に遭遇することができます。

配列は、角かっこで囲まれたコンマ区切りの値のリストとして記述されます。配列内のNumber項目は、通常どおりに書式設定されます。DateとDateTime型は单一引quotesで記述されます。文字列は、上記と同じエスケープ規則で一重引quotesで記述されます。

NULLとして書式設定される\N.

の各要素 入れ子構造体は配列として表されます。

例えは:

```
CREATE TABLE nestedt
(
    `id` UInt8,
    `aux` Nested(
        a UInt8,
        b String
    )
)
ENGINE = TinyLog
```

```
INSERT INTO nestedt Values ( 1, [1], ['a'])
```

```
SELECT * FROM nestedt FORMAT TSV
```

```
1 [1] ['a']
```

## TabSeparatedRaw

とは異なる TabSeparated 行がエスケープせずに書き込まれる形式。

この形式は、クエリ結果の出力にのみ適していますが、解析(テーブルに挿入するデータの取得)には適していません。

この形式は、名前でも使用できます TSVRaw.

## TabSeparatedWithNames

とは異なる TabSeparated 列名が最初の行に書き込まれる形式。

解析中、最初の行は完全に無視されます。列名を使用して位置を特定したり、列の正しさを確認したりすることはできません。

(ヘッダー行の解析のサポートは、将来的に追加される可能性があります。)

この形式は、名前でも使用できます TSVWithNames.

## TabSeparatedWithNamesAndTypes

とは異なる **TabSeparated** 列名は最初の行に書き込まれ、列タイプは二番目の行に書き込まれます。解析中、第一行と第二行は完全に無視されます。

この形式は、名前でも使用できます **TSVWithNamesAndTypes**.

## テンプレート

このフォーマットで指定するカスタムフォーマット文字列とプレースホルダーのための値を指定して逃げます。

設定を使用します **format\_template\_resultset**, **format\_template\_row**, **format\_template\_rows\_between\_delimiter** and **some settings of other formats** (e.g. **output\_format\_json\_quote\_64bit\_integers** 使用する場合 **JSON** エスケープ、さらに参照)

設定 **format\_template\_row** 次の構文で行の書式指定文字列を含むファイルへのパスを指定します:

```
delimiter_1${column_1:serializeAs_1}delimiter_2${column_2:serializeAs_2} ... delimiter_N,
```

どこに **delimiter\_i** 値の区切り文字です (\$ シンボルでき逃してい \$\$), **column\_i** 値が選択または挿入される列の名前またはインデックスを指定します(空の場合、列はスキップされます), **serializeAs\_i** 列値のエスケープ規則です。以下の脱出ルールに対応:

- **CSV, JSON, XML** (同じ名前の形式と同様)
- **Escaped** (同様に **TSV**)
- **Quoted** (同様に **Values**)
- **Raw** (エスケープせずに、同様に **TSVRaw**)
- **None** (エスケープルールなし、詳細を参照)

エスケープ規則が省略された場合、**None** 使用されます。 **XML** と **Raw** 出力のためにだけ適しています。

したがって、次の書式文字列の場合:

```
`Search phrase: ${SearchPhrase:Quoted}, count: ${c:Escaped}, ad price: $$$${price:JSON};`
```

の値 **SearchPhrase**, **c** と **price** 列は、次のようにエスケープ **Quoted**, **Escaped** と **JSON** その間に (選択のために) 印刷されるか、または (挿入のために) 期待されます **Search phrase: , count: , ad price: \$** と ; 区切り文字。例えば:

```
Search phrase: 'bathroom interior design', count: 2166, ad price: $3;
```

その **format\_template\_rows\_between\_delimiter** これは、最後の行を除くすべての行の後に印刷されます (または期待されます) 。 (\n 既定では)

設定 **format\_template\_resultset** このパスには、**resultset**の書式指定文字列が含まれます。 **Resultset**の書式指定文字列は、行の書式指定文字列と同じ構文を持ち、接頭辞、接尾辞、および追加情報を出力する方法を指定できます。列名の代わりに次のプレースホルダが含まれます:

- **data** はデータを含む行です **format\_template\_row** で区切られた形式 **format\_template\_rows\_between\_delimiter**. このプレースホルダーの最初のプレースホルダー形式の文字列になります。
- **totals** は、合計値を持つ行です。 **format\_template\_row** 書式(合計で使用する場合)
- **min** は最小値を持つ行です。 **format\_template\_row** 書式(極値が1に設定されている場合)
- **max** は最大値を持つ行です。 **format\_template\_row** 書式(極値が1に設定されている場合)
- **rows** 出力行の合計数です

- `rows_before_limit` そこにあったであろう行の最小数は制限なしです。出力の場合のみを含むクエリを制限します。クエリにGROUP BYが含まれている場合、`rows_before_limit_at_least`は制限なしで行われていた行の正確な数です。
- `time` リクエストの実行時間を秒単位で指定します
- `rows_read` 読み込まれた行数です
- `bytes_read` (圧縮されていない)読み込まれたバイト数です

プレースホルダ `data`, `totals`, `min` と `max` エスケープルールを指定してはいけません `None` 明示的に指定する必要があります)。残りのプレースホルダはあるの脱出ルールを指定します。

もし `format_template_resultset` 設定は空の文字列です,  `${data}`  デフォルト値として使用されます。

`Insert` クエリ形式では、接頭辞または接尾辞の場合は、一部の列または一部のフィールドをスキップできます (例を参照)。

例を選択:

```
SELECT SearchPhrase, count() AS c FROM test.hits GROUP BY SearchPhrase ORDER BY c DESC LIMIT 5 FORMAT
Template SETTINGS
format_template_resultset = '/some/path/resultset.format', format_template_row = '/some/path/row.format',
format_template_rows_between_delimiter = '\n' '
```

/some/path/resultset.format:

```
<!DOCTYPE HTML>
<html> <head> <title>Search phrases</title> </head>
<body>
<table border="1"> <caption>Search phrases</caption>
  <tr> <th>Search phrase</th> <th>Count</th> </tr>
  ${data}
</table>
<table border="1"> <caption>Max</caption>
  ${max}
</table>
<b>Processed ${rows_read:XML} rows in ${time:XML} sec</b>
</body>
</html>
```

/some/path/row.format:

```
<tr> <td>${0:XML}</td> <td>${1:XML}</td> </tr>
```

結果:

```
<!DOCTYPE HTML>
<html> <head> <title>Search phrases</title> </head>
<body>
<table border="1"> <caption>Search phrases</caption>
  <tr> <th>Search phrase</th> <th>Count</th> </tr>
  <tr> <td></td> <td>8267016</td> </tr>
  <tr> <td>bathroom interior design</td> <td>2166</td> </tr>
  <tr> <td>yandex</td> <td>1655</td> </tr>
  <tr> <td>spring 2014 fashion</td> <td>1549</td> </tr>
  <tr> <td>freeform photos</td> <td>1480</td> </tr>
</table>
<table border="1"> <caption>Max</caption>
  <tr> <td></td> <td>8873898</td> </tr>
</table>
<b>Processed 3095973 rows in 0.1569913 sec</b>
</body>
</html>
```

挿入例:

```
Some header
Page views: 5, User id: 4324182021466249494, Useless field: hello, Duration: 146, Sign: -1
Page views: 6, User id: 4324182021466249494, Useless field: world, Duration: 185, Sign: 1
Total rows: 2
```

```
INSERT INTO UserActivity FORMAT Template SETTINGS
format_template_resultset = '/some/path/resultset.format', format_template_row = '/some/path/row.format'
```

```
/some/path/resultset.format:
```

```
Some header\n${data}\nTotal rows: ${:CSV}\n
```

```
/some/path/row.format:
```

```
Page views: ${PageViews:CSV}, User id: ${UserID:CSV}, Useless field: ${:CSV}, Duration: ${Duration:CSV}, Sign: ${Sign:CSV}
```

PageViews, UserID, Duration と Sign 内部のプレースホルダーは、テーブル内の列の名前です。後の値 Useless field 行とその後 \nTotal rows: in suffixは無視されます。

すべての区切り文字の入力データを厳密に等しい区切り文字で指定されたフォーマット文字列です。

## TemplateIgnoreSpaces

この形式は入力にのみ適しています。

に類似した Template しかし、入力ストリーム内の区切り文字と値の間の空白文字をスキップします。ただし、書式指定文字列に空白文字が含まれている場合、これらの文字は入力ストリーム内で使用されます。でも指定空のプレースホルダー \${} または \${:None} 区切り文字を別々の部分に分割して、それらの間のスペースを無視します。などのプレースホルダを使用させていただきますの飛び空白文字です。

それは読むことが可能で JSON 列の値がすべての行で同じ順序を持つ場合は、この形式を使用します。たとえば、次のリクエストは、formatの出力例からデータを挿入するために使用できます JSON:

```
INSERT INTO table_name FORMAT TemplateIgnoreSpaces SETTINGS
format_template_resultset = '/some/path/resultset.format', format_template_row = '/some/path/row.format',
format_template_rows_between_delimiter = ','
```

```
/some/path/resultset.format:
```

```
 ${{}"meta":${}:${:JSON},${}"data":${}:${}
[${data}]${},${}"totals":${}:${:JSON},${}"extremes":${}:${:JSON},${}"rows":${}:${:JSON},${}"rows_before_limit_
at_least":${}:${:JSON}${}}
```

```
/some/path/row.format:
```

```
 ${{}"SearchPhrase":${}:${}{phrase:JSON}${},${}"c":${}:${}{cnt:JSON}${}}
```

## TSKV

TabSeparatedに似ていますが、name=value形式で値を出力します。名前はTabSeparated形式と同じ方法でエスケープされ、=記号もエスケープされます。

```
SearchPhrase= count()=8267016
SearchPhrase=bathroom interior design count()=2166
SearchPhrase=yandex count()=1655
SearchPhrase=2014 spring fashion count()=1549
SearchPhrase=freeform photos count()=1480
SearchPhrase=angelina jolie count()=1245
SearchPhrase=omsk count()=1112
SearchPhrase=photos of dog breeds count()=1091
SearchPhrase=curtain designs count()=1064
SearchPhrase=baku count()=1000
```

**NULL** として書式設定される \N.

```
SELECT * FROM t_null FORMAT TSKV
```

```
x=1 y=\N
```

小さな列が多数ある場合、この形式は無効であり、一般的に使用する理由はありません。それにもかかわらず、それは JSONEachRowよりも悪くありません効率の面で。

Both data output and parsing are supported in this format. For parsing, any order is supported for the values of different columns. It is acceptable for some values to be omitted – they are treated as equal to their default values. In this case, zeros and blank rows are used as default values. Complex values that could be specified in the table are not supported as defaults.

構文解析できるの存在は、追加のフィールド `tskv` 等号または値なし。この項目は無視されます。

## CSV

カンマ区切りの値の形式 ([RFC](#)).

書式設定の場合、行は二重引用符で囲まれます。文字列内の二重引用符は、行の二重引用符として出力されます。文字をエスケープするルールは他にありません。Dateとdate-timeは二重引用符で囲みます。数値は引用符なしで出力されます。値は区切り文字で区切られます。, デフォルトでは。区切り文字は設定で定義されています

**format\_csv\_delimiter**. 行は、Unixラインフィード(LF)を使用して区切られます。最初に、配列はTabSeparated形式のように文字列にシリアル化され、結果の文字列は二重引用符でCSVに出力されます。CSV形式のタプルは、個別の列としてシリアル化されます(つまり、タプル内の入れ子は失われます)。

```
$ clickhouse-client --format_csv_delimiter="|" --query="INSERT INTO test.csv FORMAT CSV" < data.csv
```

\*デフォルトでは、区切り文字は , を参照。[format\\_csv\\_delimiter](#) より多くの情報のための設定。

解析時には、すべての値を引用符の有無にかかわらず解析できます。二重引用符と单一引quotesの両方がサポートされています。行は引用符なしで配置することもできます。この場合、区切り文字または改行 (CRまたはLF) まで解析されます。RFCに違反して、引用符なしで行を解析すると、先頭と末尾のスペースとタブは無視されます。ラインフィードでは、Unix(LF)、Windows(CR LF)、およびMac OS Classic(CR LF)タイプがすべてサポートされています。

空の引用符で囲まれていない入力値は、それぞれの列のデフォルト値に置き換えられます。

[input\\_format\\_defaults\\_for\\_omitted\\_fields](#)

有効です。

**NULL** として書式設定される \N または NULL または空の引用符で囲まれていない文字列(設定を参照

[input\\_format\\_csv\\_unquoted\\_null\\_literal\\_as\\_null](#) と [input\\_format\\_defaults\\_for\\_omitted\\_fields](#)).

CSV形式では、合計と極値の出力は次のようにサポートされます TabSeparated.

## CSVWithNames

また、次のようなヘッダー行も出力します `TabSeparatedWithNames`.

## カスタム区切り

に類似した `テンプレート` ですが、版画を読み込みまたは全てのカラムを使用脱出ルールからの設定

`format_customEscapingRule` 設定から区切り文字 `formatCustomFieldDelimiter`, `formatCustomRowBeforeDelimiter`,  
`formatCustomRowAfterDelimiter`, `formatCustomRowBetweenDelimiter`, `formatCustomResultBeforeDelimiter` と  
`formatCustomResultAfterDelimiter` 書式指定文字列からではありません。

また `CustomSeparatedIgnoreSpaces` 形式は次のようにになります `TemplateIgnoreSpaces`.

## JSON

JSON形式でデータを出力します。データテーブルのほかに、列名と型、および出力行の合計数、制限がない場合に出力される可能性のある行の数などの追加情報も出力します。例:

```
SELECT SearchPhrase, count() AS c FROM test.hits GROUP BY SearchPhrase WITH TOTALS ORDER BY c DESC LIMIT 5
FORMAT JSON
```

```
{
  "meta": [
    {
      "name": "SearchPhrase",
      "type": "String"
    },
    {
      "name": "c",
      "type": "UInt64"
    }
  ],
  "data": [
    {
      "SearchPhrase": "",
      "c": "8267016"
    },
    {
      "SearchPhrase": "bathroom interior design",
      "c": "2166"
    },
    {
      "SearchPhrase": "yandex",
      "c": "1655"
    },
    {
      "SearchPhrase": "spring 2014 fashion",
      "c": "1549"
    },
    {
      "SearchPhrase": "freeform photos",
      "c": "1480"
    }
  ],
  "totals": [
    {
      "SearchPhrase": "",
      "c": "8873898"
    }
  ],
  "extremes": [
    {
      "min": [
        {
          "SearchPhrase": "",
          "c": "1480"
        }
      ],
      "max": [
        {
          "SearchPhrase": "",
          "c": "8267016"
        }
      ]
    }
  ],
  "rows": 5,
  "rows_before_limit_at_least": 141137
}
```

JSONはJavaScriptと互換性があります。これを確実にするために、一部の文字は追加でエスケープされます。/としてエスケープ \;代替の改行 U+2028 と U+2029 ブラウザによってはエスケープされます \uXXXX. ASCII制御文字はエスケープされます。 \b, \f, \n, \r, \t を使用して、00-1F範囲の残りのバイトと同様に \uXXXX sequences. Invalid UTF-8 sequences are changed to the replacement character \uFFFD so the output text will consist of valid UTF-8 sequences. For compatibility with JavaScript, Int64 and UInt64 integers are enclosed in double-quotes by default. To remove the quotes, you can set the configuration parameter [output\\_format\\_json\\_quote\\_64bit\\_integers](#) 0にする。

`rows` – The total number of output rows.

`rows_before_limit_at_least` そこにある行の最小数は制限なしであったでしょう。出力の場合のみを含むクエリを制限します。

クエリにGROUP BYが含まれている場合、`rows_before_limit_at_least`は制限なしで行われていた行の正確な数です。

`totals` – Total values (when using WITH TOTALS).

`extremes` – Extreme values (when extremes are set to 1).

この形式は、クエリ結果の出力にのみ適していますが、解析(テーブルに挿入するデータの取得)には適していません。

ClickHouseサポート `NULL` として表示されます。`null` JSON出力で。

も参照。`JSONEachRow` 形式。

## JSONCompact

JSONとは異なるのは、データ行がオブジェクトではなく配列で出力される点だけです。

例:

```
{  
    "meta": [  
        {  
            "name": "SearchPhrase",  
            "type": "String"  
        },  
        {  
            "name": "c",  
            "type": "UInt64"  
        }  
    ],  
    "data": [  
        ["", "8267016"],  
        ["bathroom interior design", "2166"],  
        ["yandex", "1655"],  
        ["fashion trends spring 2014", "1549"],  
        ["freeform photo", "1480"]  
    ],  
    "totals": ["", "8873898"],  
    "extremes": {  
        "min": ["", "1480"],  
        "max": ["", "8267016"]  
    },  
    "rows": 5,  
    "rows_before_limit_at_least": 141137  
}
```

この形式は、クエリ結果の出力にのみ適していますが、解析(テーブルに挿入するデータの取得)には適していません。  
も参照。`JSONEachRow` 形式。

## JSONEachRow

この形式を使用する場合、ClickHouseは行を区切り、改行区切りのJSONオブジェクトとして出力しますが、データ全体は有効なJSONではありません。

```
{"SearchPhrase": "curtain designs", "count()": "1064"}  
{"SearchPhrase": "baku", "count()": "1000"}  
{"SearchPhrase": "", "count()": "8267016"}
```

データを挿入するときは、各行に個別のJSONオブジェクトを指定する必要があります。

## データの挿入

```
INSERT INTO UserActivity FORMAT JSONEachRow {"PageViews":5,"UserID":"4324182021466249494","Duration":146,"Sign":-1} {"UserID":"4324182021466249494","PageViews":6,"Duration":185,"Sign":1}
```

クリックハウス：

- オブジェクト内のキーと値のペアの任意の順序。
- いくつかの値を省略します。

ClickHouseを無視した空間要素には、カンマの後にオブジェクト。すべてのオブジェクトを一行で渡すことができます。改行で区切る必要はありません。

### 省略された値の処理

ClickHouseは省略された値を対応するデフォルト値に置き換えます [データ型](#)。

もし `DEFAULT expr` に応じて異なる置換規則を使用します。[input\\_format\\_defaults\\_for\\_omitted\\_fields](#) 設定。

次の表を考えます:

```
CREATE TABLE IF NOT EXISTS example_table
(
    x UInt32,
    a DEFAULT x * 2
) ENGINE = Memory;
```

- もし `input_format_defaults_for_omitted_fields = 0` のデフォルト値 `x` と `a` 等しい `0` のデフォルト値として `UInt32` データ型)。
- もし `input_format_defaults_for_omitted_fields = 1` のデフォルト値 `x` 等しい `0` しかし、デフォルト値は `a` 等しい `x * 2`.

### 警告

データを挿入するとき `input_format_defaults_for_omitted_fields = 1`, ClickHouseは、挿入と比較して、より多くの計算リソースを消費します `input_format_defaults_for_omitted_fields = 0`.

## データの選択

を考える `UserActivity` 例としての表:

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	-1
4324182021466249494	6	185	1

クエリ `SELECT * FROM UserActivity FORMAT JSONEachRow` ツヅケ。

```
{"UserID":"4324182021466249494","PageViews":5,"Duration":146,"Sign":-1}
{"UserID":"4324182021466249494","PageViews":6,"Duration":185,"Sign":1}
```

とは異なり `JSON` 無効なUTF-8シーケンスの置換はありません。値は、次の場合と同じ方法でエスケープされます `JSON`.

## 注

任意のバイトセットを文字列に出力できます。 使用する `JSONEachRow` テーブル内のデータが情報を失うことなく JSONとしてフォーマットできることが確実な場合は、`format`。

## 入れ子構造の使用

あなたがテーブルを持っている場合 入れ子 データ型の列には、同じ構造でJSONデータを挿入することができます。 この機能を有効にするには `input_format_import_nested_json` 設定。

たとえば、次の表を考えてみましょう：

```
CREATE TABLE json_each_row_nested (n Nested (s String, i Int32) ) ENGINE = Memory
```

で見ることができるように Nested ClickHouseは、入れ子構造の各コンポーネントを個別の列として扱います (`n.s` と `n.i` 私達のテーブルのため)。 次の方法でデータを挿入できます:

```
INSERT INTO json_each_row_nested FORMAT JSONEachRow {"n.s": ["abc", "def"], "n.i": [1, 23]}
```

挿入データとしての階層JSONオブジェクト `input_format_import_nested_json=1`.

```
{
  "n": {
    "s": ["abc", "def"],
    "i": [1, 23]
  }
}
```

この設定がない場合、ClickHouseは例外をスローします。

```
SELECT name, value FROM system.settings WHERE name = 'input_format_import_nested_json'
```

name	value
input_format_import_nested_json	0

```
INSERT INTO json_each_row_nested FORMAT JSONEachRow {"n": {"s": ["abc", "def"], "i": [1, 23]}}
```

```
Code: 117. DB::Exception: Unknown field found while parsing JSONEachRow format: n: (at row 1)
```

```
SET input_format_import_nested_json=1
INSERT INTO json_each_row_nested FORMAT JSONEachRow {"n": {"s": ["abc", "def"], "i": [1, 23]}}
SELECT * FROM json_each_row_nested
```

n.s	n.i
['abc','def']	[1,23]

ネイティブ

最も効率的な形式。データ書き込みおよび読み込みをブロックのバイナリ形式です。ブロックごとに、このブロック内の行数、列数、列名と型、および列の一部が次々と記録されます。つまり、この形式は “columnar” – it doesn't convert columns to rows. This is the format used in the native interface for interaction between servers, for using the command-line client, and for C++ clients.

この形式を使用すると、ClickHouse DBMSでのみ読み取ることができるダンプをすばやく生成できます。この形式で自分で作業するのは理にかなっていません。

## Null

何も出力されません。ただし、クエリは処理され、コマンドラインクライアントを使用する場合、データはクライアントに送信されます。この使用のための試験、性能試験をします。

明らかに、この形式は出力にのみ適しており、解析には適していません。

## プリティ

出力データとしてのUnicodeトテーブルも用ANSI-エスケープシーケンス設定色の端子です。

テーブルの完全なグリッドが描画され、各行は、端末内の二つの行を占めています。

各結果ブロックは、個別のテーブルとして出力されます。これは、結果をパッファリングせずにブロックを出力できるようにするために必要です（すべての値の可視幅を事前に計算するためにはパッファ

**NULL** として出力されます `NULL`.

例（以下に示す プリティコンパクト 形式）：

```
SELECT * FROM t_null
```

x		y
1	NULL	

行はきれいな\*形式ではエスケープされません。例はのために示されています **プリティコンパクト** 形式：

```
SELECT 'String with \'quotes\' and \t character' AS Escaping_test
```

```
Escaping_test
String with 'quotes' and      character |
```

端末に大量のデータをダンプするのを避けるために、最初の10,000行だけが印刷されます。行数が10,000以上の場合、メッセージ “Showed first 10 000” 印刷されます。

この形式は、クエリ結果の出力にのみ適していますが、解析(テーブルに挿入するデータの取得)には適していません。

Pretty形式は、合計値(合計と共に使用する場合)および極値(場合)の出力をサポートします ‘extremes’ 1)に設定される。このような場合、合計値と極値は、メインデータの後に個別のテーブルで出力されます。例（以下に示す **プリティコンパクト** 形式）：

```
SELECT EventDate, count() AS c FROM test.hits GROUP BY EventDate WITH TOTALS ORDER BY EventDate FORMAT PrettyCompact
```

EventDate	c
2014-03-17	1406958
2014-03-18	1383658
2014-03-19	1405797
2014-03-20	1353623
2014-03-21	1245779
2014-03-22	1031592
2014-03-23	1046491

Totals:

EventDate	c
1970-01-01	8873898

Extremes:

EventDate	c
2014-03-17	1031592
2014-03-23	1406958

## プリティコンパクト

とは異なる [プリティ グリッド](#)が行の間に描画され、結果がよりコンパクトになるという点で。  
この形式は、対話モードのコマンドラインクライアントで既定で使用されます。

## プリティコンパクトモノブロック

とは異なる [プリティコンパクト](#) 最大10,000行がバッファリングされ、ブロックではなく単一のテーブルとして出力されます。

## プリティノスケープ

Ansiエスケープシーケンスが使用されない点でPrettyとは異なります。これは、プラウザでこの形式を表示するために必要です。‘watch’ コマンドライン

例:

```
$ watch -n1 "clickhouse-client --query='SELECT event, value FROM system.events FORMAT PrettyCompactNoEscapes'"
```

HTTPインターフェイスを使用して、ブラウザーで表示できます。

## プリティコンパクトノスケープ

前の設定と同じです。

## プリティスペースノスケープス

前の設定と同じです。

## PrettySpace

とは異なる [プリティコンパクト](#) その中で、グリッドの代わりに空白（空白文字）が使用されます。

## ローバイナリ

バイナリ形式で行ごとにデータを書式設定および解析します。行と値は、区切り文字なしで連続して表示されます。  
この形式は行ベースであるため、ネイティブ形式よりも効率的ではありません。

整数は固定長のリトルエンディアン表現を使用します。たとえば、UInt64は8バイトを使用します。

DateTimeは、Unixタイムスタンプを値として含むUInt32として表されます。

Dateは、値として1970-01-01以降の日数を含むUInt16オブジェクトとして表されます。

文字列はvarintの長さ(符号なし)で表されます **LEB128**)の後に文字列のバイトが続きます。

FixedStringは、単にバイトのシーケンスとして表されます。

配列はvarintの長さとして表されます(符号なし **LEB128**)の後に、配列の連続する要素が続きます。

のために **NULL** 1または0を含む追加のバイトがそれぞれの前に追加されます **Null可能** 値。1の場合、値は **NULL** このバイトは別の値として解釈されます。0の場合、バイトの後の値は **NULL**.

## RowBinaryWithNamesAndTypes

に類似した **ローバイナリ**,しかし、追加されたヘッダ:

- **LEB128**-エンコードされた列数(N)
- N **Strings**列名の指定
- N **Strings**列タイプの指定

## 値

すべての行をかっこで表示します。行はカンマで区切られます。最後の行の後にコンマはありません。角かっこ内の値もコンマ区切りです。数値は、引用符なしで小数点形式で出力されます。配列は角かっこで出力されます。文字列、日付、および時刻付きの日付は引用符で出力されます。ルールのエスケープと解析は **TabSeparated** 形式。フォーマット中に余分なスペースは挿入されませんが、解析中には許可され、スキップされます（配列値内のスペースは許可されません）。**NULL** と表される。**NULL**.

The minimum set of characters that you need to escape when passing data in Values format: single quotes and backslashes.

これはで使用される形式です **INSERT INTO t VALUES ...** ただし、クエリ結果の書式設定にも使用できます。

も参照。**:input\_format\_values\_interpret\_expressions** と  
**input\_format\_values\_deduce\_templates\_of\_expressions** 設定。

## 垂直

列名を指定して、それぞれの値を別々の行に出力します。この形式は、各行が多数の列で構成されている場合、一つまたは数つの行だけを印刷するのに便利です。

**NULL** として出力されます **NULL**.

例:

```
SELECT * FROM t_null FORMAT Vertical
```

Row 1:

x: 1  
y: **NULL**

行は縦書き式でエスケープされません:

```
SELECT 'string with \'quotes\' and \'t with some special \n characters' AS test FORMAT Vertical
```

Row 1:

test: string with 'quotes' and      with some special  
characters

この形式は、クエリ結果の出力にのみ適していますが、解析(テーブルに挿入するデータの取得)には適していません。

## XML

XML形式は出力にのみ適しており、解析には適していません。例:

```
<?xml version='1.0' encoding='UTF-8' ?>
<result>
  <meta>
    <columns>
      <column>
        <name>SearchPhrase</name>
        <type>String</type>
      </column>
      <column>
        <name>count()</name>
        <type>UInt64</type>
      </column>
    </columns>
  </meta>
  <data>
    <row>
      <SearchPhrase></SearchPhrase>
      <field>8267016</field>
    </row>
    <row>
      <SearchPhrase>bathroom interior design</SearchPhrase>
      <field>2166</field>
    </row>
    <row>
      <SearchPhrase>yandex</SearchPhrase>
      <field>1655</field>
    </row>
    <row>
      <SearchPhrase>2014 spring fashion</SearchPhrase>
      <field>1549</field>
    </row>
    <row>
      <SearchPhrase>freeform photos</SearchPhrase>
      <field>1480</field>
    </row>
    <row>
      <SearchPhrase>angelina jolie</SearchPhrase>
      <field>1245</field>
    </row>
    <row>
      <SearchPhrase>omsk</SearchPhrase>
      <field>1112</field>
    </row>
    <row>
      <SearchPhrase>photos of dog breeds</SearchPhrase>
      <field>1091</field>
    </row>
    <row>
      <SearchPhrase>curtain designs</SearchPhrase>
      <field>1064</field>
    </row>
    <row>
      <SearchPhrase>baku</SearchPhrase>
      <field>1000</field>
    </row>
  </data>
  <rows>10</rows>
  <rows_before_limit_at_least>141137</rows_before_limit_at_least>
</result>
```

列名に許容可能な形式がない場合は、次のようにします 'field' 要素名として使用されます。一般に、XML構造はJSON構造に従います。

Just as for JSON, invalid UTF-8 sequences are changed to the replacement character `☒` so the output text will consist of valid UTF-8 sequences.

文字列値では、文字 `<` と `&` としてエスケープ `<` と `&`.

配列は次のように出力される `<array><elem>Hello</elem><elem>World</elem>...</array>`、およびタプルとして `<tuple><elem>Hello</elem><elem>World</elem>...</tuple>`.

## CapnProto

Cap'n Protoは、プロトコルバッファや儕約に似たバイナリメッセージ形式ですが、JSONやMessagePackのようなものではありません。

Cap'n Protoメッセージは厳密に型指定され、自己記述ではないため、外部スキーマ記述が必要です。スキーマはその場で適用され、クエリごとにキャッシュされます。

```
$ cat capnproto_messages.bin | clickhouse-client --query "INSERT INTO test.hits FORMAT CapnProto SETTINGS format_schema='schema:Message'"
```

どこに `schema.capnp` このように見えます:

```
struct Message {
    SearchPhrase @0 :Text;
    c @1 :Uint64;
}
```

逆シリアル化は効果的であり、通常はシステム負荷を増加させません。

も参照。 [スキーマの書式](#).

## プロトブフ

Protobufは-です [プロトコル](#) 形式。

この形式には、外部形式スキーマが必要です。このスキーマをキャッシュ間のクエリ。

ClickHouseは両方をサポート `proto2` と `proto3` 構文。繰り返/オプションに必要な分野に対応しています。

使用例:

```
SELECT * FROM test.table FORMAT Protobuf SETTINGS format_schema = 'schemafile:MessageType'
```

```
cat protobuf_messages.bin | clickhouse-client --query "INSERT INTO test.table FORMAT Protobuf SETTINGS format_schema='schemafile:MessageType'"
```

ここで、ファイル `schemafile.proto` このように見えます:

```
syntax = "proto3";

message MessageType {
    string name = 1;
    string surname = 2;
    uint32 birthDate = 3;
    repeated string phoneNumbers = 4;
};
```

の対応関係はテーブル列-分野のプロトコルバッファのメッセージタイプClickHouseを比較しつけられた名前が使われている。

この比較では、大文字と小文字は区別されません。\_ (アンダースコア)と. (ドット) は等しいと見なされます。列の型とプロトコルバッファのメッセージのフィールドが異なる場合は、必要な変換が適用されます。

ネストしたメッセージに対応します。たとえば、フィールドの場合 z 次のメッセージの種類

```
message MessageType {  
    message XType {  
        message YType {  
            int32 z;  
        };  
        repeated YType y;  
    };  
    XType x;  
};
```

ClickHouseは、名前のある列を検索しようとします x.y.z (または x\_y\_z または X.y\_Z というように)。入れ子になったメッセージは、[入れ子データ構造](#)。

次のようにprotobufスキーマで定義されたデフォルト値

```
syntax = "proto2";  
  
message MessageType {  
    optional int32 result_per_page = 3 [default = 10];  
}
```

は適用されない。[テーブルの既定値](#) それらの代わりに使用されます。

ClickHouseの入力および出力protobufのメッセージ [length-delimited](#) 形式。

これは、すべてのメッセージがその長さを書き込まれる前に [varint](#).

も参照。[一般的な言語で長さ区切りのprotobufメッセージを読み書きする方法](#).

## アヴロ

[Apache Avro](#) ApacheのHadoopプロジェクト内で開発された行指向のデータ直列化フレームワークです。

ClickHouse Avro形式は読み書きをサポートします [Avroデータファイル](#).

### データ型の一致

次の表に、サポートされているデータ型とClickHouseとの一致を示します [データ型](#) で [INSERT](#) と [SELECT](#) クエリ。

Avroデータ型 <a href="#">INSERT</a>	ClickHouseデータ型	Avroデータ型 <a href="#">SELECT</a>
boolean, int, long, float, double	<a href="#">Int(8/16/32)</a> , <a href="#">UInt(8/16/32)</a>	int
boolean, int, long, float, double	<a href="#">Int64</a> , <a href="#">UInt64</a>	long
boolean, int, long, float, double	<a href="#">Float32</a>	float
boolean, int, long, float, double	<a href="#">Float64</a>	double
bytes, string, fixed, enum	文字列	bytes
bytes, string, fixed	固定文字列(N)	fixed(N)

Avroデータ型 <b>INSERT</b>	ClickHouseデータ型	Avroデータ型 <b>SELECT</b>
enum	Enum (8月16日)	enum
array(T)	配列(T)	array(T)
union(null, T), union(T, null)	Nullable(T)	union(null, T)
null	Nullable(Nothing)	null
int (date) *	日付	int (date) *
long (timestamp-millis) *	DateTime64(3)	long (timestamp-millis) *
long (timestamp-micros) *	DateTime64(6)	long (timestamp-micros) *

\* Avro論理タイプ

未サポートのAvroデータ型: record (非ルート), map

サポートされないAvro論理データ型: uuid, time-millis, time-micros, duration

## データの挿入

AvroファイルからClickHouseテーブルにデータを挿入するには:

```
$ cat file.avro | clickhouse-client --query="INSERT INTO {some_table} FORMAT Avro"
```

入力Avroファ record タイプ。

の対応関係はテーブル列分野のアプロスキーマClickHouseを比較しつけられた名前が使われている。この比較では、大文字と小文字が区別されます。

未使用の項目はスキップされます。

データの種類ClickHouseテーブルの列ができ、対応する分野においてアプロのデータを挿入します。データを挿入するとき、ClickHouseは上記の表に従ってデータ型を解釈し、次に キャスト 対応する列タイプのデータ。

## データの選択

ClickHouseテーブルからAvroファイルにデータを選択するには:

```
$ clickhouse-client --query="SELECT * FROM {some_table} FORMAT Avro" > file.avro
```

列名は必須です:

- で始まる [A-Za-z\_]
- その後のみ [A-Za-z0-9\_]

出力Avroファイルの圧縮および同期間隔は以下で設定できます `output_format_avro_codec` と `output_format_avro_sync_interval` それぞれ。

## アプロコンフルエント

AvroConfluent支援復号单一のオブジェクトアプロのメッセージを使用する カフカ と Confluentスキーマレジストリ。

各Avroメッセージには、スキーマレジストリを使用して実際のスキーマに解決できるスキーマidが埋め込まれます。

スキーマがキャッシュ一度に解決されます。

スキーマのレジストリのURLは設定され `format_avro_schema_registry_url`

## データ型の一致

同じ `アヴロ`

## 使用法

迅速な検証スキーマ分解能を使用でき `kafkacat` と `ツツイツ姪ツ債ツツケ`:

```
$ kafkacat -b kafka-broker -C -t topic1 -o beginning -f '%s' -c 3 | clickhouse-local --input-format AvroConfluent --format_avro_schema_registry_url 'http://schema-registry' -S "field1 Int64, field2 String" -q 'select * from table'  
1 a  
2 b  
3 c
```

使用するには `AvroConfluent` と `カフカ`:

```
CREATE TABLE topic1_stream  
(  
    field1 String,  
    field2 String  
)  
ENGINE = Kafka()  
SETTINGS  
kafka_broker_list = 'kafka-broker',  
kafka_topic_list = 'topic1',  
kafka_group_name = 'group1',  
kafka_format = 'AvroConfluent';  
  
SET format_avro_schema_registry_url = 'http://schema-registry';  
  
SELECT * FROM topic1_stream;
```

## 警告

設定 `format_avro_schema_registry_url` で構成する必要があります `users.xml` 再起動後にその値を維持する。

## 寄木細工

アパッチの寄木細工 Hadoopエコシステムに広く普及している柱状ストレージ形式です。ClickHouse支援を読み取りと書き込みの操作のためにこの形式です。

## データ型の一致

次の表に、サポートされているデータ型とClickHouseとの一致を示します `データ型` で `INSERT` と `SELECT` クエリ。

Parquetデータ型 ( <code>INSERT</code> )	ClickHouseデータ型	Parquetデータ型 ( <code>SELECT</code> )
<code>UINT8, BOOL</code>	<code>UInt8</code>	<code>UINT8</code>
<code>INT8</code>	<code>Int8</code>	<code>INT8</code>
<code>UINT16</code>	<code>UInt16</code>	<code>UINT16</code>

Parquetデータ型 ( <code>INSERT</code> )	ClickHouseデータ型	Parquetデータ型 ( <code>SELECT</code> )
<code>INT16</code>	<code>Int16</code>	<code>INT16</code>
<code>UINT32</code>	<code>UInt32</code>	<code>UINT32</code>
<code>INT32</code>	<code>Int32</code>	<code>INT32</code>
<code>UINT64</code>	<code>UInt64</code>	<code>UINT64</code>
<code>INT64</code>	<code>Int64</code>	<code>INT64</code>
<code>FLOAT, HALF_FLOAT</code>	<code>Float32</code>	<code>FLOAT</code>
<code>DOUBLE</code>	<code>Float64</code>	<code>DOUBLE</code>
<code>DATE32</code>	日付	<code>UINT16</code>
<code>DATE64, TIMESTAMP</code>	<code>DateTime</code>	<code>UINT32</code>
<code>STRING, BINARY</code>	文字列	<code>STRING</code>
—	<code>FixedString</code>	<code>STRING</code>
<code>DECIMAL</code>	小数点	<code>DECIMAL</code>

ClickHouseは構成可能な精密を支えます `Decimal` タイプ。その `INSERT` クエリは、寄木細工を扱います `DECIMAL` ClickHouseとして入力します `Decimal128` タイプ。

対応している寄木細工のデータ種類: `DATE32`, `TIME32`, `FIXED_SIZE_BINARY`, `JSON`, `UUID`, `ENUM`.

データの種類ClickHouseテーブルの列ができ、対応する分野の寄木細工のデータを挿入します。データを挿入するとき、ClickHouseは上記の表に従ってデータ型を解釈し、次に キャスト ClickHouseテーブル列に設定されているそのデータ型のデータ。

## データの挿入と選択

次のコマンドで、ファイルからパーケットデータをClickHouseテーブルに挿入できます:

```
$ cat {filename} | clickhouse-client --query="INSERT INTO {some_table} FORMAT Parquet"
```

ClickHouseテーブルからデータを選択し、次のコマンドでParquet形式でファイルに保存できます:

```
$ clickhouse-client --query="SELECT * FROM {some_table} FORMAT Parquet" > {some_file.pq}
```

Hadoopとデータを交換するには、以下を使用できます [HDFSテーブルエンジン](#)。

## ORC

[Apache ORC](#) Hadoopエコシステムに広く普及している柱状ストレージ形式です。この形式のデータはClickHouseにのみ挿入できます。

## データ型の一致

次の表に、サポートされているデータ型とClickHouseとの一致を示します **データ型** で **INSERT** クエリ。

ORCデータ型 ( <b>INSERT</b> )	ClickHouseデータ型
UINT8, BOOL	UInt8
INT8	Int8
UINT16	UInt16
INT16	Int16
UINT32	UInt32
INT32	Int32
UINT64	UInt64
INT64	Int64
FLOAT, HALF_FLOAT	Float32
DOUBLE	Float64
DATE32	日付
DATE64, TIMESTAMP	DateTime
STRING, BINARY	文字列
DECIMAL	小数点

ClickHouseは構成可能な精密を支えます **Decimal** タイプ。その **INSERT** クエリはORCを扱います **DECIMAL** ClickHouseとして入力します **Decimal128** タイプ。

未サポートのORCデータ型: **DATE32**, **TIME32**, **FIXED\_SIZE\_BINARY**, **JSON**, **UUID**, **ENUM**.

ClickHouseテーブル列のデータ型は、対応するORCデータフィールドと一致する必要はありません。データを挿入するとき、ClickHouseは上記の表に従ってデータ型を解釈し、次に **キャスト** ClickHouseテーブル列のデータ型セットへのデータ。

## データの挿入

次のコマンドで、ファイルからOrcデータをClickHouseテーブルに挿入できます:

```
$ cat filename.orc | clickhouse-client --query="INSERT INTO some_table FORMAT ORC"
```

Hadoopとデータを交換するには、以下を使用できます **HDFS** テーブルエンジン.

## スキーマの書式

形式スキーマを含むファイル名は、この設定によって設定されます `format_schema`。  
いずれかの形式を使用する場合は、この設定を設定する必要があります `Cap'n Proto` と `Protobuf`。  
形式スキーマは、ファイル名とこのファイル内のメッセージ型の名前の組み合わせで、コロンで区切られます，  
e.g. `schemafile.proto:MessageType`。  
ファイルが形式の標準拡張子を持っている場合（たとえば, `.proto` のために `Protobuf`），  
この場合、形式スキーマは次のようにになります `schemafile:MessageType`。

を介してデータを入力または出力する場合 **クライアント** で **対話モード**、形式スキーマで指定されたファイル名  
を含むことができ、絶対パス名は相対パスを現在のディレクトリのクライアント  
クライアントを使用する場合 **バッチモード** は、パスのスキーマ"相対的"に指定する必要があります。

を介してデータを入力または出力する場合 **HTTPインターフェ** 形式スキーマで指定されたファイル名  
指定されたディレクトリにあるはずです。 `format_schema_path`  
サーバー構成で。

## スキップエラー

次のような形式があります `CSV`, `TabSeparated`, `TSKV`, `JSONEachRow`, `Template`, `CustomSeparated` と `Protobuf` 解析エ  
ラーが発生した場合に壊れた行をスキップし、次の行の先頭から解析を続行できます。 見る  
`input_format_allow_errors_num` と  
`input_format_allow_errors_ratio` 設定。

制限:

- 解析エラーの場合 `JSONEachRow` 新しい行(またはEOF)まですべてのデータをスキップします。 `\n` エラーを正しく数え  
る。
- `Template` と `CustomSeparated` 最後の列の後に `delimiter` を使用し、行の間に `delimiter` を使用すると、次の行の先頭を  
見つけることができます。

## JDBCドライバ

- **公式ドライバー**
- サードパーティドライバ:
  - [ClickHouse-Native-JDBC](#)
  - [clickhouse4j](#)

## ODBCドライバ

- **公式ドライバー**.

## C++クライアントライ

のREADMEを参照してください [クリックハウス-cpp](#) リポジトリ。

## サードパーティ

### 免責事項

Yandexのはない 以下のライブラリを維持し、その品質を保証するための広範なテストを行っていません。

- Python
  - インフィ [clickhouse\\_orm](#)
  - [clickhouse-](#) ドライバ
  - [clickhouse-](#) クライアント
  - [aiochclient](#)
  - [asynch](#)
- PHP
  - [smi2/phpclickhouse](#)
  - [8bitov/clickhouse-php](#)- クライアント
  - [ツツイツ姪"ツツ"ツ債ツづツツケ](#)
  - [simpod/clickhouse-](#) クライアント
  - [seva-code/php-click-house-client](#)
  - [SeasClick C++](#) クライアント
- 行け
  - [クリックハウス](#)
  - [ゴークリックハウス](#)
  - [メールルーゴ-](#) クリックハウス
  - [ゴラン-](#) クリックハウス
- NodeJs
  - [クリックハウス\(曖昧さ回避\)](#)
  - [ノード-](#) クリックハウス
- Perl
  - [perl-DBD-](#) クリックハウス
  - [HTTP-](#) クリックハウス
  - [AnyEvent-](#) クリックハウス
- Ruby
  - [クリックハウス\(Ruby\)](#)
  - [クリックハウス-activerecord](#)
- R
  - [クリックハウス-r](#)
  - [RClickHouse](#)
- Java
  - [clickhouse-](#) クライアント -java
  - [clickhouse-](#) クライアント
- Scala
  - [clickhouse-scala-](#) クライアント

- コトリン
  - AORM
- C#
  - Octonica.ClickHouseClient
  - クリックハウスAdo
  - クリックハウスクライアン
  - ClickHouse.Net
- エリクサー
  - クリックハウス
  - 柱
- Nim
  - ニム-クリックハウス

## サードパーティ開発者からの統合ライブラリ

### 免責事項

Yandexのはない以下のツールとライブラリを維持し、その品質を確保するための広範なテストを行っています。

## インフラ製品

- リレーションナルデータベース管理システム
  - MySQL
    - mysql2ch
    - ProxySQL
    - clickhouse-mysql-データリーダー
    - horgh-レプリケーター
  - PostgreSQL
    - clickhousedb\_fdw
    - インフィ clickhouse\_fdw (用途 インフィ clickhouse\_orm)
    - pg2ch
    - clickhouse\_fdw
  - MSSQL
    - ClickHouseMigrator
- メッセージキュー
  - カフカ
    - clickhouse\_sinker (用途 Goクライアント)
    - stream-loader-clickhouse

- ストリーム処理
  - フリンク
    - フリンク-クリックハウス-シンク
- オブジェクト保管
  - S3
    - clickhouse-バックアップ
- 容器の協奏
  - Kubernetes
    - clickhouse-演算子
- 構成管理
  - パペット
    - イノゲームズ/クリックハウス
  - mfedorov/クリックハウス
- 監視
  - 黒鉛
    - グラファウス
    - カーボンクリックハウス
    - グラファイト-クリック
    - 黒鉛-ch-オティマイザ -staled仕切りを最大限に活用する \*GraphiteMergeTree からのルールの場合 ロールアップ構成 応用できます
  - グラファナ
    - クリックハウス-グラファナ
  - プロメテウス
    - clickhouse\_exporter
    - プロムハウス
    - clickhouse\_exporter (用途 Goクライアント)
  - ナギオス
    - check\_clickhouse
    - check\_clickhouse.py
  - Zabbix
    - clickhouse-zabbix-テンプレート
  - Sematext
    - clickhouseの統合
- ロギング
  - rsyslog
    - オムクリックハウス
  - フルエント
    - ログハウス (のために Kubernetes)
  - logagent
    - logagent output-plugin-clickhouse

- Geo
  - MaxMind
    - クリックハウス-maxmind-geoip

## プログラミ

- Python
  - SQLAlchemy
    - sqlalchemy-クリックハウス（用途 インフィ clickhouse\_orm）
  - パンダ
    - パンダハウス
- PHP
  - 教義
    - dbal-クリックハウス
- R
  - dplyr
    - RClickHouse（用途 クリックハウス-cpp）
- Java
  - Hadoop
    - clickhouse-hdfs-loader（用途 JDBC）
- Scala
  - アッカ
    - clickhouse-scala-クライアント
- C#
  - ADO.NET
    - クリックハウスAdo
    - クリックハウスクライアン
    - ClickHouse.Net
    - ClickHouse.Net.Migrations
- エリクサー
  - Ecto
    - clickhouse\_ecto
- Ruby
  - Ruby on rails
    - activecube
    - ActiveRecord
  - GraphQL
    - activecube-graphql

---

サードパーティ

オープンソース

Tabix

## のClickHouseのためのWebインターフェイス **Tabix** プロジェクト

特徴:

- 追加のソフトウェアをインストールする必要なしに、ブラウザから直接ClickHouseで動作します。
- 構文の強調表示とクエリエディタ。
- コマンドの自動補完。
- ツールのためのグラフィカルに分析クエリを実行します。
- 配色オプション。

## Tabixドキュメント .

### ハウスオップ

ハウスオップ OSX、Linux、Windows用のUI/IDEです。

特徴:

- 構文の強調表示とクエリビルダー。テーブルまたはJSONビューで応答を表示します。
- CSVまたはJSONとしてエクスポートクエリ結果。
- 説明付きのプロセスのリスト。書き込みモード。停止する能力 (KILL) プロセス。
- データベース すべてのテーブルとその列に追加情報が表示されます。
- 列サイズのクイックビュー。
- サーバー構成。

以下の機能の開発が計画されています:

- データベース管理。
- ユーザー管理。
- リアルタイムデータ解析。
- クラスター監視。
- クラスタ管理。
- 複製されたテーブルとカフカテーブルの監視。

### 灯台

灯台 ClickHouseのための軽量なwebインターフェイスです。

特徴:

- テーブルのリストのフィルタリング、メタデータを指すものとします。
- テーブルのプレビューとフィルタです。
- 読み取り専用クエリの実行。

### レダッシュ

レダッシュ めるためのプラットフォームのデータを可視化する。

サポート、多数のデータソースを含むClickHouse、Redash参加できる結果のクエリからデータソースへの最終データセットである。

特徴:

- クエリの強力なエディタ。
- エクスプローラ
- さまざまな形でデータを表現できる視覚化ツール。

## DBeaver

**DBeaver** - ユニバーサルデスクトップのデータベースのクライアント ClickHouse ます。

特徴:

- 構文の強調表示と自動補完によるクエリ開発。
- テーブルリフィルとメタデータを検索する
- 表データプレビュー。
- 全文検索。

## clickhouse-cli

**clickhouse-cli** Python3で書かれたClickHouseの代替コマンドラインクライアントです。

特徴:

- 自動補完。
- クエリとデータ出力の構文強調表示。
- データ出力のためのポケットベルサポート。
- カスタム PostgreSQL のようなコマンド。

## クリックハウス-グラフ

**クリックハウス-グラフ** このツールは `system.trace_log` として **グラフ**。

## クリックハウス-プランタム

**クリックハウス-プランタム** 生成するスクリプトです **プランタム** テーブルのスキームの図。

## 商業

## DataGrip

**DataGrip** ClickHouse専用のサポートを持つJetBrainsのデータベースIDEです。 PyCharm、IntelliJ IDEA、GoLand、PhpStormなど、他のIntelliJベースのツールにも埋め込まれています。

特徴:

- 非常に高速なコード補完。
- ClickHouse構文の強調表示。
- ネストされた列、テーブルエンジンなど、ClickHouse固有の機能のサポート。
- データエディタ。

- リファクタリング。

- 検索とナビゲーション。

## Yandexデータレンズ

Yandexデータレンズ データの可視化と分析のサービスです。

特徴:

- シンプルな棒グラフから複雑なダッシュボードまで、幅広い視覚化が可能です。
- ダッシュボードを公開できます。
- ClickHouseを含む複数のデータソースのサポート。
- ClickHouseに基づく実体化されたデータのストレージ。

データレンズは **自由のために利用できる** 商業使用の低負荷プロジェクトのため。

- DataLensドキュメント。

- チュートリアル 上の可視化するデータからClickHouseデータベースです。

## Holisticsソフトウェア

ホリスティック フルスタックのデータプラットフォームは、ビジネスインツールです。

特徴:

- 自動メール、Slackやグーグルシートのスケジュール。
- Visualizations、バージョン管理、自動補完、再利用可能なクエリコンポーネントと動的フィルタとSQLエディタ。
- Iframeによるレポートとダッシュボードの組み込み分析。
- データ準備とETL機能。
- SQLデータモデリング支援のためのリレーションナルマッピングのデータです。

## ルッカー

ルッカー はデータプラットフォームは、ビジネスインツールをサポート 50+データベースの方言を含むClickHouse。LookerはSaaSプラットフォームとして利用可能で、セルフホスト型です。ユーザーが利用できLookerの場合は、VPNクライアントの直接探索、データの構築の可視化とダッシュボード、スケジュール、農場管理について学んでいます。Lookerのツールを埋め込むためのチャプターでは、これらの機能の他のアプリケーション、およびAPI統合データを、他のアプリケーション

特徴:

- LookMLを使った簡単でアジャイルな開発  
**データモデル化** レポート作成者とエンドユーザーをサポートする。
- 見物人による強力なワークフローの統合 **データ操作**。

LookerでClickHouseを設定する方法。

---

## サードパーティ

## chproxy

**chproxy**,は、ClickHouseデータベース用のHTTPプロキシとロードバランサです。

特徴:

- ユーザーごとのルーティング。
- 適用範囲が広い限界。
- SSL証明書の自動renewal。

Goで実装。

## キッテンハウス

**キッテンハウス** することが重要である現地代理人とClickHouse、アプリケーションサーバの場合でもバッファに挿入データとお客様側です。

特徴:

- メモリ内およびディスク上のデータバッファリング。
- テーブル単位のルーティング。
- 負荷分散と正常性チェック。

Goで実装。

## クリックハウス-バルク

**クリックハウス-バルク** 単純なClickHouse挿入コレクターです。

特徴:

- グループの要求送信によるしきい値または間隔で出ています。
- 複数のリモートサーバー。
- 基本認証。

Goで実装。

## 表エンジン

のテーブルエンジン型式の表を行います。:

- データの格納方法と場所、データの書き込み先、およびデータの読み取り先。
- サポートされているクエリと方法。
- 同時データアクセス。
- インデックスが存在する場合の使用。
- マルチスレッド要求の実行が可能かどうか。
- データ複製パラメーター。

## エンジン家族

### メルゲツリー

高負荷仕事のための最も普遍的な、機能テーブルエンジン。本物件の共有によるこれらのエンジンには迅速にデータを挿入とその後のバックグラウンドデータを処となります。**MergeTree** 家族のエンジンの支援データレプリケーション（**複製\*** バージョンのエンジン）分割、その他の機能で対応していない他のエンジンです。

家族のエンジン：

- **メルゲツリー**
- 置換マージツリー
- サミングマーゲツリー
- **AggregatingMergeTree**
- 折りたたみマージツリー
- バージョニングコラプシングマーゲットリー
- **GraphiteMergeTree**

## ログ

軽量 エンジン 最低の機能性を使って。多くの小さなテーブル（最大約1万行）をすばやく書き込み、後で全体として読み込む必要がある場合に最も効果的です。

家族のエンジン：

- **TinyLog**
- ストリップログ
- ログ

## 統合エンジン

エンジン用プリケーションデータストレージと処理システム。

家族のエンジン：

- カフカ
- **MySQL**
- **ODBC**
- **JDBC**
- **HDFS**

## 特殊エンジン

家族のエンジン：

- 分散
- マテリアライズドビュー
- 辞書
- [Merge](special/merge.md#merge
- ファイル
- Null

- セット
- 参加
- URL
- 表示
- メモリ
- パッファ

## 仮想列

仮想列は、エンジンのソースコードで定義されている整数テーブルエンジン属性です。

仮想列を指定するべきではありません。CREATE TABLE あなたはそれらを見ることができません SHOW CREATE TABLE と DESCRIBE TABLE クエリ結果。仮想列も読み取り専用であるため、仮想列にデータを挿入することはできません。

仮想列からデータを選択するには、仮想列の名前を指定する必要があります。SELECT クエリ。SELECT \* 仮想列から値を返しません。

テーブル仮想列のいずれかと同じ名前の列を持つテーブルを作成すると、仮想列にアクセスできなくなります。これはお勧めしません。競合を避けるために、仮想列名には通常、アンダースコアが付けられます。

## メルゲツリー

その MergeTree この家族のエンジンそして他のエンジン (\*MergeTree) は、最も堅牢なClickHouseテーブルエンジンです。

のエンジン MergeTree ファミリは、非常に大量のデータをテーブルに挿入するために設計されています。のデータが書き込まれ、テーブル部、そのルール適用のための統合のパートです。この方法は効率よく継続的に書き換えのデータストレージ中をサポートしていません。

主な特長:

- 主キ

これを作成できる小型疎指標を見つけるデータを高速に行います。

- この場合、分割キー が指定される。

ClickHouseは、同じ結果を持つ同じデータに対する一般的な操作よりも効果的なパーティションを持つ特定の操作をサポートします。ClickHouseも自動的に遮断すると、パーティションデータのパーティショニングキーで指定されたクエリ。この改善するためのクエリ。

- データ複製のサポート。

の家族 ReplicatedMergeTree テーブルを提供データレプリケーション。詳細については、データ複製。

- データサンプリングです。

必要に応じて、テーブル内でデータサンプリング方法を設定できます。

## 情報

その マージ エンジンはに属しません \*MergeTree 家族だ

# テーブルの作成

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1] [TTL expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2] [TTL expr2],
    ...
    INDEX index_name1 expr1 TYPE type1(...) GRANULARITY value1,
    INDEX index_name2 expr2 TYPE type2(...) GRANULARITY value2
) ENGINE = MergeTree()
[PARTITION BY expr]
[ORDER BY expr]
[PRIMARY KEY expr]
[SAMPLE BY expr]
[TTL expr [DELETE|TO DISK 'xxx'|TO VOLUME 'xxx'], ...]
[SETTINGS name=value, ...]
```

パラメータの説明については、[クエリの説明の作成](#)。

## 注

**INDEX** 実験的な機能です。データスキップ索引。

## クエリ句

- **ENGINE** — Name and parameters of the engine. `ENGINE = MergeTree()`. その `MergeTree` エンジンにはパラメータがない。
- **PARTITION BY** — The [分割キー](#)。

月によるパーティション分割の場合は、`toYYYYMM(date_column)` 式、ここで `date_column` 型の日付を持つ列です [日付](#)。ここでのパーティション名は `"YYYYMM"` 形式。

- **ORDER BY** — The sorting key.

列または任意の式のタプル。例: `ORDER BY (CounterID, EventDate)`.

- **PRIMARY KEY** — The primary key if it [ソートキーとは異なります](#)。

デフォルトでは、プライマリキーはソートキーと同じです。`ORDER BY` 節)。したがって、ほとんどの場合、別の `PRIMARY KEY` 句。

- **SAMPLE BY** — An expression for sampling.

サンプリング式を使用する場合は、主キーにそれを含める必要があります。例: `SAMPLE BY intHash32(UserID) ORDER BY (CounterID, EventDate, intHash32(UserID))`.

- **TTL** — A list of rules specifying storage duration of rows and defining logic of automatic parts movement [ディスクとボリューム間](#)。

式には次のものが必要です `Date` または `DateTime` 結果としての列。例:

`TTL date + INTERVAL 1 DAY`

ルールのタイプ `DELETE|TO DISK 'xxx'|TO VOLUME 'xxx'` 式が満たされた場合(現在の時刻に達した場合)、パートに対して実行されるアクションを指定します;期限切れの行の削除、パートの移動(パート内のすべての `(TO DISK 'xxx')` またはボリュームに `(TO VOLUME 'xxx')`)。ルールの既定の種類は削除です (`DELETE`)。リストに複数のルール指定がありませんよ `DELETE` ルール

詳細は、を参照してください [列および表のTTL](#)

- SETTINGS — Additional parameters that control the behavior of the MergeTree:
  - `index_granularity` — Maximum number of data rows between the marks of an index. Default value: 8192. See [データ保存](#).
  - `index_granularity_bytes` — Maximum size of data granules in bytes. Default value: 10Mb. To restrict the granule size only by number of rows, set to 0 (not recommended). See [データ保存](#).
  - `enable_mixed_granularity_parts` — Enables or disables transitioning to control the granule size with the `index_granularity_bytes` setting. In version 19.11 and earlier, this setting controls the size of the granules in bytes. It is recommended to use `index_granularity` instead. This setting improves ClickHouse performance by selecting a large row-based granule (several dozen to several hundred megabytes) for tables with many rows. It also improves the efficiency of large `SELECT` queries.
  - `use_minimalistic_part_header_in_zookeeper` — Storage method of the data parts headers in ZooKeeper. If `use_minimalistic_part_header_in_zookeeper=1` その飼育係の店が少ない。 詳細については、を参照してください [設定の説明](#) で “Server configuration parameters”.
  - `min_merge_bytes_to_use_direct_io` — The minimum data volume for merge operation that is required for using direct I/O access to the storage disk. When merging data parts, ClickHouse calculates the total storage volume of all the data to be merged. If the volume exceeds `min_merge_bytes_to_use_direct_io` バイト、ClickHouseは直接入出力インターフェイスを使用して記憶域ディスクにデータを読み、書きます (`O_DIRECT` オプション)。もし `min_merge_bytes_to_use_direct_io = 0` その後、直接I/Oが無効になります。 デフォルト値: `10 * 1024 * 1024 * 1024` バイト数。
  - `merge_with_ttl_timeout` — Minimum delay in seconds before repeating a merge with TTL. Default value: 86400 (1 day).
  - `write_final_mark` — Enables or disables writing the final index mark at the end of data part (after the last byte). Default value: 1. Don't turn it off.
  - `merge_max_block_size` — Maximum number of rows in block for merge operations. Default value: 8192.
  - `storage_policy` — Storage policy. See [複数のブロックデバイスのためのデータ保存](#).

## セクション設定例

```
ENGINE MergeTree() PARTITION BY toYYYYMM(EventDate) ORDER BY (CounterID, EventDate, intHash32(UserID))
SAMPLE BY intHash32(UserID) SETTINGS index_granularity=8192
```

この例では、月別にパーティション分割を設定します。

また、ユーザIDによってハッシュとしてサンプリングする式を設定します。これにより、各テーブルのデータを擬似乱数化することができます `CounterID` と `EventDate`. を定義すると `SAMPLE` 勅データを選択すると、ClickHouseはユーザーのサブセットに対して均等な擬似乱数データサンプルを返します。

その `index_granularity` 8192がデフォルト値であるため、設定を省略できます。

▶ 推奨されていません法テーブルを作成する

## データ保存

テーブルのデータ部品の分別によりその有効なタイプを利用します。

データがテーブルに挿入されると、個別のデータパートが作成され、それぞれが主キーで辞書順に並べ替えられます。たとえば、主キーが `(CounterID, Date)` パーツ内のデータは次のようにソートされます `CounterID`、およびそれぞれの中 `CounterID`、それは順序付けられます `Date`.

データに属する別のパーティションが分離の異なる部品です。背景では、ClickHouseはより有効な貯蔵のためのデータ部分を併合する。異なる区画に属する部分はマージされません。マージ機構では、同じ主キーを持つすべての行が同じデータ部分に含まれることは保証されません。

各データ部分は論理的にグラニュールに分割されます。顆粒は、データを選択するときにClickHouseが読み取る最小の不可分データセットです。ClickHouseは行や値を分割しないため、各グラニュールには常に整数の行が含まれます。顆粒の最初の行には、行の主キーの値がマークされます。各データ、ClickHouseを作成しインデックスファイルを格納するのです。各列について、主キーにあるかどうかにかかわらず、ClickHouseにも同じマークが格納されます。これらのマークまたはデータを見つける直列のファイルです。

微粒のサイズはによって制限されます `index_granularity` と `index_granularity_bytes` テーブルエンジンの設定。微粒の列の数はで置きます `[1, index_granularity]` 行のサイズに応じて、範囲。微粒のサイズは超過できます `index_granularity_bytes` 単一行のサイズが設定の値より大きい場合。この場合、顆粒のサイズは行のサイズに等しい。

## クエリ内の主キーとインデックス

を取る (`CounterID, Date`) 例として主キー。この場合、ソートとインデックスは次のように示すことができます:

```
Whole data:  [-----]
CounterID:  [aaaaaaaaaaaaaaaaaaaaabbbbcdeeeeeeeeeeeefgggggggghhhhhhhhiiiiiiik|||||||]
Date:        [11111112222222333123321111222222332111111212222223111122231112233]
Marks:       |   |   |   |   |   |   |   |   |   |
              a,1  a,2  a,3  b,3  e,2  e,3  g,1  h,2  i,1  i,3  l,3
Marks numbers: 0   1   2   3   4   5   6   7   8   9   10
```

データクエリが指定する場合:

- `CounterID in ('a', 'h')`、サーバーはマークの範囲のデータを読み取ります `[0, 3)` と `[6, 8)`.
- `CounterID IN ('a', 'h') AND Date = 3`、サーバーはマークの範囲のデータを読み取ります `[1, 3)` と `[7, 8)`.
- `Date = 3`、サーバーは、マークの範囲内のデータを読み取ります `[1, 10]`.

上記の例としては常に使用するのがより効果的指標により、フルスキャン！

に乏指数で追加するデータを読み込みます。プライマリキーの単一の範囲を読み取るとき `index_granularity * 2` 余分な列の各データブロック読み取ることができます。

疎指標できる作業は非常に多くのテーブル行において、多くの場合、指標はコンピュータのアドレスです。

ClickHouseでは一意の主キーは必要ありません。同じ主キーで複数の行を挿入できます。

## 主キーの選択

主キーの列の数は明示的に制限されません。データ構造によっては、主キーに多かれ少なかれ含めることができます。これは:

- 索引のパフォーマンスを向上させます。

主キーが `(a, b)` 次に、別の列を追加します `c` 以下の条件を満たすと、パフォーマンスが向上します:

- 列に条件を持つクエリがあります `c`.
- 長いデータ範囲(長いデータ範囲より数倍長い `index_granularity`)と同一の値を持つ。`(a, b)` 一般的です。つまり、別の列を追加すると、かなり長いデータ範囲をスキップすることができます。
- データ圧縮を改善します。

ClickHouseは主キーによってデータを並べ替えるので、一貫性が高いほど圧縮が良くなります。

■ 追加的なロジックが統合データ部分の **折りたたみマージツリー** と **サミングマーゲツリー** エンジンだ

この場合、ソートキーこれは主キーとは異なります。

長い主キーは挿入のパフォーマンスとメモリ消費に悪影響を及ぼしますが、主キーの追加の列はClickHouseのパフォーマンスには影響しません **SELECT** クエリ。

## 並べ替えキーとは異なる主キーの選択

ソートキー(データ部分の行をソートする式)とは異なる主キー(マークごとにインデックスファイルに書き込まれる値を持つ式)を指定することができます。この場合、主キー式タプルは、並べ替えキー式タプルのプレフィックスでなければなりません。

この機能は、**サミングマーゲツリー** と

**AggregatingMergeTree** テーブルエンジン。共通の場合はご利用になられる場合はエンジンのテーブルは、二種類のカラム: 寸法と対策。典型的なクエリは、任意のメジャー列の値を集計します **GROUP BY** そして次元によるろ過。

**Summingmergetree** と **AggregatingMergeTree** は並べ替えキーの値が同じ行を集計するため、すべてのディメンションを追加するのは自然です。その結果、キー式は列の長いリストで構成され、このリストは頻繁に新しく追加されたディメンションで更新する必要があります。

この場合、効率的な範囲スキャンを提供し、残りのディメンション列を並べ替えキータプルに追加する主キーには数列のみを残すことが理にかなって

**ALTER** 新しい列がテーブルとソートキーに同時に追加されると、既存のデータ部分を変更する必要がないため、ソートキーの軽量な操作です。古い並べ替えキーは新しい並べ替えキーのプレフィックスであり、新しく追加された列にデータがないため、テーブルの変更の瞬間に、データは新旧の並べ替え

## クエリでの索引とパーティションの使用

のために **SELECT** ク ClickHouse を分析するかどうかの指標を使用できます。インデックスは、次の場合に使用できます **WHERE/PREWHERE** 句には、等価比較演算または不等比較演算を表す式 (連結要素のいずれかとして、または完全に) があります。 **IN** または **LIKE** 主キーまたはパーティショニングキーにある列または式、またはこれらの列の特定的部分的に反復機能、またはこれらの式の論理的関係に固定プレフィッ

したがって、主キーの一つまたは多くの範囲でクエリを迅速に実行することができます。この例では、特定の追跡タグ、特定のタグと日付範囲、特定のタグと日付、日付範囲を持つ複数のタグなどに対してクエリを実行すると、クエリが高速

次のように構成されたエンジンを見てみましょう:

```
ENGINE MergeTree() PARTITION BY toYYYYMM(EventDate) ORDER BY (CounterID, EventDate) SETTINGS  
index_granularity=8192
```

この場合、クエリで:

```
SELECT count() FROM table WHERE EventDate = toDate(now()) AND CounterID = 34  
SELECT count() FROM table WHERE EventDate = toDate(now()) AND (CounterID = 34 OR CounterID = 42)  
SELECT count() FROM table WHERE ((EventDate >= toDate('2014-01-01') AND EventDate <= toDate('2014-01-31'))  
OR EventDate = toDate('2014-05-01')) AND CounterID IN (101500, 731962, 160656) AND (CounterID = 101500 OR  
EventDate != toDate('2014-05-01'))
```

ClickHouseの主キー指標のトリムで不正なデータを毎月パーティショニングキーパンフレット、ホームページの間仕切りする不適切な日。

上記のクエリのインデックスが使用されるときにも複雑な表現です。テーブルからの読み取りがいを使用した指標できないっぱいたします。

以下の例では、インデックスは使用できません。

```
SELECT count() FROM table WHERE CounterID = 34 OR URL LIKE '%upyachka%'
```

クエリの実行時にClickHouseがインデックスを使用できるかどうかを確認するには、設定を使用します **force\_index\_by\_date** と **force\_primary\_key**.

の分割による月で読み込みのみこれらのデータブロックを含むからスピーチへのマークの範囲内で適切に取扱います。この場合、データブロックには多くの日付(月全体まで)のデータが含まれる場合があります。ブロック内では、データは主キーによってソートされます。このため、主キープレフィックスを指定しない日付条件のみのクエリを使用すると、単一の日付よりも多くのデータが読み取られます。

## 部分的に単調な主キーのインデックスの使用

例えば、月の日を考えてみましょう。それらはaを形作る **単調系列** 一ヶ月のために、しかし、より長期間単調ではありません。これは部分的に単調なシーケンスです。ユーザーが部分的に単調な主キーでテーブルを作成する場合、ClickHouseは通常どおりスパースインデックスを作成します。ユーザーがこの種のテーブルからデータを選択すると、ClickHouseはクエリ条件を分析します。これは、クエリのパラメータとインデックスマークの間の距離を計算できるためです。

クエリパラメータ範囲内の主キーの値が単調なシーケンスを表していない場合、ClickHouseはインデックスを使用できません。この場合、ClickHouseはフルスキヤン方法を使用します。

ClickHouseは、このロジックを月の日数シーケンスだけでなく、部分的に単調なシーケンスを表す主キーにも使用します。

## データスキップインデックス(実験)

インデックス宣言は、`CREATE` クエリ。

```
INDEX index_name expr TYPE type(...) GRANULARITY granularity_value
```

からのテーブルのため \*MergeTree 家族データの飛び指標を指定できます。

これらのインデックスは、指定された式に関する情報をブロックに集約します。`granularity_value` 微粒（微粒のサイズはを使用して指定されます `index_granularity` テーブルエンジンでの設定）。次に、これらの集計は `SELECT` クエリを削減量のデータから読み込むディスクにより操大きなブロックのデータを `where` クエリが満たされません。

例

```
CREATE TABLE table_name
(
    u64 UInt64,
    i32 Int32,
    s String,
    ...
    INDEX a (u64 * i32, s) TYPE minmax GRANULARITY 3,
    INDEX b (u64 * length(s)) TYPE set(1000) GRANULARITY 4
) ENGINE = MergeTree()
...
```

この例のインデックスをClickHouseで使用すると、次のクエリでディスクから読み取るデータ量を減らすことができます:

```
SELECT count() FROM table WHERE s < 'z'
SELECT count() FROM table WHERE u64 * i32 == 10 AND u64 * length(s) >= 1234
```

使用可能なインデックスタイプ

#### ■ minmax

指定された式の極値を格納します(式が `tuple` の各要素の極値を格納します。 `tuple`)を使用して保存情報の飛びプロックのようなデータは、その有効なタイプを利用します。

#### ■ set(max\_rows)

指定された式の一意の値を格納します。`max_rows` 行, `max_rows=0` つまり “no limits”). 値を使用して、`WHERE` 式はデータのプロックでは満足できません。

#### ■ ngrambf\_v1(n, size\_of\_bloom\_filter\_in\_bytes, number\_of\_hash\_functions, random\_seed)

ストア a **Bloom フィルタ** これによりngramsからプロックのデータです。 文字列でのみ動作します。 の最適化に使用することができます `equals`, `like` と `in` 式。

- `n` — ngram size,

- `size_of_bloom_filter_in_bytes` — Bloom filter size in bytes (you can use large values here, for example, 256 or 512, because it can be compressed well).

- `number_of_hash_functions` — The number of hash functions used in the Bloom filter.

- `random_seed` — The seed for Bloom filter hash functions.

#### ■ tokenbf\_v1(size\_of\_bloom\_filter\_in\_bytes, number\_of\_hash\_functions, random\_seed)

と同じ `ngrambf_v1` しかし、ngramsの代わりにトークンを格納します。 トークンは英数字以外の文字で区切られた配列です。

#### ■ bloom\_filter([false\_positive]) — Stores a **Bloom フィルタ** 指定された列の場合。

任意 `false_positive` パラメータは、フィルタから偽陽性応答を受信する確率です。 可能な値:(0,1)。 既定値は0.025です。

対応するデータ型: `Int*`, `UInt*`, `Float*`, `Enum`, `Date`, `DateTime`, `String`, `FixedString`, `Array`, `LowCardinality`, `Nullable`.

以下の関数が使用できます: 等しい, `notEquals`, で, ノティン, は.

```
INDEX sample_index (u64 * length(s)) TYPE minmax GRANULARITY 4
INDEX sample_index2 (u64 * length(str), i32 + f64 * 100, date, str) TYPE set(100) GRANULARITY 4
INDEX sample_index3 (lower(str), str) TYPE ngrambf_v1(3, 256, 2, 0) GRANULARITY 4
```

#### 機能サポート

の条件 `WHERE` 句には、列で動作する関数の呼び出しが含まれます。 列がインデックスの一部である場合、ClickHouseは関数の実行時にこのインデックスを使用しようとします。 ClickHouse支援の異なるサブセットの機能を使用。

その `set` 索引はすべての機能と使用することができる。 その他のインデックスの関数サブセットを以下の表に示します。

関数(演算子)/インデックス	主キー	<code>minmax</code>	<code>ngrambf_v1</code>	<code>tokenbf_v1</code>	<code>bloom_filter name</code>
等しい( <code>=, ==</code> )	✓	✓	✓	✓	✓
<code>notEquals(!=, \&lt;&gt;)</code>	✓	✓	✓	✓	✓
のよう	✓	✓	✓	✗	✗
<code>notLike</code>	✓	✓	✓	✗	✗

関数(演算子)/インデックス	主キー	minmax	ngrambf_v1	tokenbf_v1	bloom_filter name
スタート	✓	✓	✓	✓	✗
endsWith	✗	✗	✓	✓	✗
マルチサーチ	✗	✗	✓	✗	✗
で	✓	✓	✓	✓	✓
ノティン	✓	✓	✓	✓	✓
less(\<)	✓	✓	✗	✗	✗
グレーター(>)	✓	✓	✗	✗	✗
lessOrEquals(\<=)	✓	✓	✗	✗	✗
greaterOrEquals(>=)	✓	✓	✗	✗	✗
空	✓	✓	✗	✗	✗
ノーテンプティ	✓	✓	✗	✗	✗
ハイストケン	✗	✗	✗	✓	✗

Ngramサイズよりも小さい定数引数を持つ関数は、次のように使用できません ngrambf\_v1 クエリ最適化のため。

フルでは偽陽性一致なので、ngrambf\_v1, tokenbf\_v1, and bloom\_filter インデックスは、関数の結果がfalseになると予想されるクエリの最適化には使用できません。:

- 最適化できます:
  - s LIKE '%test%'
  - NOT s NOT LIKE '%test%'
  - s = 1
  - NOT s != 1
  - startsWith(s, 'test')
- 最適化できない:
  - NOT s LIKE '%test%'
  - s NOT LIKE '%test%'
  - NOT s = 1
  - s != 1
  - NOT startsWith(s, 'test')

## 同時データアクセス

同時テーブルアクセスには、マルチバージョン管理を使用します。つまり、テーブルの読み取りと更新が同時に行われると、クエリの時点で現在の部分のセットからデータが読み取られます。長いロックはありません。挿入は読み取り操作の邪魔になりません。

テーブルからの読み取りは自動的に並列化されます。

## 列および表のTTL

値の有効期間を決定します。

その `TTL` 句は、テーブル全体および個々の列ごとに設定できます。テーブルレベルのTTLで指定した論理の自動移動のデータディスクの間とします。

表現を行い、評価しなければならぬ `日付` または `DateTime` データ型。

例:

```
TTL time_column  
TTL time_column + interval
```

定義するには `interval,use 時間間隔` 演算子。

```
TTL date_time + INTERVAL 1 MONTH  
TTL date_time + INTERVAL 15 HOUR
```

## 列TTL

列の値が期限切れになると、ClickHouseは列のデータ型の既定値に置き換えます。データ部分のすべての列の値が期限切れになった場合、ClickHouseはファイルシステム内のデータ部分からこの列を削除します。

その `TTL` 句はキー列には使用できません。

例:

TTLを使用したテーブルの作成

```
CREATE TABLE example_table  
(  
    d DateTime,  
    a Int TTL d + INTERVAL 1 MONTH,  
    b Int TTL d + INTERVAL 1 MONTH,  
    c String  
)  
ENGINE = MergeTree  
PARTITION BY toYYYYMM(d)  
ORDER BY d;
```

既存のテーブルの列へのTTLの追加

```
ALTER TABLE example_table  
MODIFY COLUMN  
c String TTL d + INTERVAL 1 DAY;
```

列のTTLを変更する

```
ALTER TABLE example_table  
MODIFY COLUMN  
c String TTL d + INTERVAL 1 MONTH;
```

## テーブルTTL

テーブルでの表現の除去に終了しました列、複数の表現を自動で部品の移動と **ディスク** また、表の行が期限切れになると、ClickHouseは対応するすべての行を削除します。 パーツ移動フィーチャの場合、パーツのすべての行が移動式基準を満たす必要があります。

```
TTL expr [DELETE|TO DISK 'aaa'|TO VOLUME 'bbb'], ...
```

TTL規則のタイプは、各TTL式に従うことができます。これは、式が満たされた後（現在の時刻に達する）に実行されるアクションに影響します）：

- **DELETE** 削除行を終了しました（デフォルトアクション）；
- **TO DISK 'aaa'** -ディスクへの部分の移動 aaa；
- **TO VOLUME 'bbb'** -ディスクへの部分の移動 bbb.

例：

TTLを使用したテーブルの作成

```
CREATE TABLE example_table
(
    d DateTime,
    a Int
)
ENGINE = MergeTree
PARTITION BY toYYYYMM(d)
ORDER BY d
TTL d + INTERVAL 1 MONTH [DELETE],
        d + INTERVAL 1 WEEK TO VOLUME 'aaa',
        d + INTERVAL 2 WEEK TO DISK 'bbb';
```

テーブルのTTLの変更

```
ALTER TABLE example_table
    MODIFY TTL d + INTERVAL 1 DAY;
```

データの削除

期限切れのTTLのデータはClickHouseがデータ部分を結合するとき取除かれる。

時ClickHouseるデータの期間は終了しましたので、行offスケジュール内スケジュールする必要がありません。このようなマージの頻度を制御するには、次のように設定できます `merge_with_ttl_timeout`. 値が小さすぎると、大量のリソースを消費する可能性のある、スケジュール外のマージが多数実行されます。

を実行する場合 **SELECT** クエリと合併はありませんが、できれば終了しました。それを避けるには、**OPTIMIZE** クエリの前に **SELECT**.

## 複数のブロックデバイスのためのデータ保存

### はじめに

`MergeTree` 家族のテーブルエンジンでデータを複数のブロックデバイス たとえば、特定のテーブルのデータが暗黙的に次のように分割される場合に便利です “hot” と “cold”. 最新のデータは定期的に要求されますが、わずかなスペースしか必要ありません。逆に、`fat-tailed`履歴データはほとんど要求されません。複数のディスクが利用できれば、“hot” データは高速ディスク(例えば、NVMe Ssdまたはメモリ内)に配置されることがあります。“cold” データ-比較的遅いもの (例えば、HDD)。

データ部分は最低の移動可能な単位のための MergeTree-エンジンテーブル。ある部分に属するデータは、一つのディスクに保存されます。データパーティットは、バックグラウンドで(ユーザー設定に従って)ディスク間で移動することができます。ALTER クエリ。

## 用語

- Disk — Block device mounted to the filesystem.
- Default disk — Disk that stores the path specified in the パス サーバー設定。
- Volume — Ordered set of equal disks (similar to JBOD).
- Storage policy — Set of volumes and the rules for moving data between them.

の名称を記載することから、システムテーブル、システムストレージポリシーとシステムディスク。適用の設定を保存政策のためのテーブルを使用 storage\_policy 設定の MergeTree-エンジン家族のテーブル。

## 設定

ディスク、ボリューム、およびストレージポリシーは、`<storage_configuration>` メインファイルにタグを付ける config.xml または、config.d ディレクトリ。

構成構造:

```
<storage_configuration>
  <disks>
    <disk_name_1> <!-- disk name -->
      <path>/mnt/fast_ssdclickhouse/</path>
    </disk_name_1>
    <disk_name_2>
      <path>/mnt/hdd1clickhouse/</path>
      <keep_free_space_bytes>10485760</keep_free_space_bytes>
    </disk_name_2>
    <disk_name_3>
      <path>/mnt/hdd2clickhouse/</path>
      <keep_free_space_bytes>10485760</keep_free_space_bytes>
    </disk_name_3>

    ...
  </disks>

  ...
</storage_configuration>
```

タグ:

- `<disk_name_N>` — Disk name. Names must be different for all disks.
- `path` — path under which a server will store data (`data` と `shadow` フォルダ) で終了する必要があります。'/'.
- `keep_free_space_bytes` — the amount of free disk space to be reserved.

ディスク定義の順序は重要ではありません。

保管方針の設定のマークアップ:

```

<storage_configuration>
...
<policies>
  <policy_name_1>
    <volumes>
      <volume_name_1>
        <disk>disk_name_from_disks_configuration</disk>
        <max_data_part_size_bytes>1073741824</max_data_part_size_bytes>
      </volume_name_1>
      <volume_name_2>
        <!-- configuration -->
      </volume_name_2>
      <!-- more volumes -->
    </volumes>
    <move_factor>0.2</move_factor>
  </policy_name_1>
  <policy_name_2>
    <!-- configuration -->
  </policy_name_2>

  <!-- more policies -->
</policies>
...
</storage_configuration>

```

タグ:

- `policy_name_N` — Policy name. Policy names must be unique.
- `volume_name_N` — Volume name. Volume names must be unique.
- `disk` — a disk within a volume.
- `max_data_part_size_bytes` — the maximum size of a part that can be stored on any of the volume's disks.
- `move_factor` — when the amount of available space gets lower than this factor, data automatically start to move on the next volume if any (by default, 0.1).

構成の例:

```

<storage_configuration>
...
<policies>
  <hdd_in_order> <!-- policy name -->
    <volumes>
      <single> <!-- volume name -->
        <disk>disk1</disk>
        <disk>disk2</disk>
      </single>
    </volumes>
  </hdd_in_order>

  <moving_from_ss_to_hdd>
    <volumes>
      <hot>
        <disk>fast_ss</disk>
        <max_data_part_size_bytes>1073741824</max_data_part_size_bytes>
      </hot>
      <cold>
        <disk>disk1</disk>
      </cold>
    </volumes>
    <move_factor>0.2</move_factor>
  </moving_from_ss_to_hdd>
</policies>
...
</storage_configuration>

```

与えられた例では、`hdd_in_order` ポリシーは、ラウンドロビン アプローチ このようにこの方針を定義しみ量 (`single`) 、データ部分は円形の順序ですべてのディスクに格納されています。こうした政策ぞれの知見について学ぶとともに有が複数ある場合は同様のディスク搭載のシステムがRAIDな設定を行います。個々のディスクドライブは信頼できないため、複製係数が3以上で補う必要がある場合があります。

システムで使用可能なディスクの種類が異なる場合、`moving_from_ssd_to_hdd` 代わりにポリシーを使用できます。ボリューム `hot` SSDディスクで構成されています (`fast_ssd`) 、このボリュームに格納できるパートの最大サイズは1GBです。1GBより大きいサイズのすべての部品は直接貯えられます `cold` HDDディスクを含むボリューム `disk1`。また、一度ディスク `fast_ssd` 80%以上によって満たされて、データはに移ります得ます `disk1` 背景プロセスによって。

ストレージポリシー内のボリューム列挙の順序は重要です。ボリュームが満杯になると、データは次のボリュームに移動されます。データは順番に格納されるため、ディスク列挙の順序も重要です。

作成時にテーブルは、適用の設定を保存方針で:

```
CREATE TABLE table_with_non_default_policy (
    EventDate Date,
    OrderID UInt64,
    BannerID UInt64,
    SearchPhrase String
) ENGINE = MergeTree
ORDER BY (OrderID, BannerID)
PARTITION BY toYYYYMM(EventDate)
SETTINGS storage_policy = 'moving_from_ssd_to_hdd'
```

その `default` ストレージポリシーは、一つのボリュームのみを使用することを意味します。<path>. テーブルを作成すると、そのストレージポリシーは変更できません。

## 詳細

の場合 `MergeTree` テーブル、データがあるディスクには、異なる方法:

- 插入の結果として (`INSERT` クエリ)。
- バックグラウンドマージ中 突然変異.
- 別のレプリカからダウンロード
- パーティションの凍結の結果として `ALTER TABLE ... FREEZE PARTITION`.

すべてのこれらの場合を除き、突然変異とパーティションの凍結は、一部が保存され、大量のディスクに保存政策:

1. 部品を格納するのに十分なディスク領域を持つ最初のボリューム(定義順) (`unreserved_space > current_part_size`) と指定されたサイズの部分を格納することができます (`max_data_part_size_bytes > current_part_size`) が選ばれる。
2. このボリューム内では、データの前のチャinkを格納するために使用され、パートサイズよりも空き領域が多いディスクが選択されます (`unreserved_space - keep_free_space_bytes > current_part_size`).

フードの下で、突然変異および仕切りの凍結は利用します ハードリンク。ハードリンクとディスクには対応していないため、この場合、パートの保管と同じディスクの初期ます。

バックグラウンドでは、部品は空き領域の量に基づいてボリューム間で移動されます (`move_factor` パラメータ) ボリュームが設定ファイルで宣言されている順序に従って。

データは、最後のデータから最初のデータに転送されることはありません。システムテーブル `システム part_log` (フィールド `type = MOVE_PART`) と `システム部品` (フィールド `path` と `disk`) 背景の動きを監視します。また、詳細情報はサーバーログに記載されています。

ユーザーの力で移動中的一部またはパーティションから量別のクエリ **ALTER TABLE ... MOVE PART|PARTITION ... TO VOLUME|DISK ...** パックグラウンド操作のすべての制限が考慮されます。クエリは単独で移動を開始し、パックグラウンド操作が完了するまで待機しません。十分な空き領域がない場合、または必要な条件のいずれかが満たされていない場合、ユーザーはエラーメッセージを表示します。

データの移動は、データの複製を妨げません。そのため、異なる保管方針を指定することができ、同じテーブルの異なるレプリカ。

パックグラウンドマージと突然変異の完了後、古い部分は一定時間後にのみ削除されます (`old_parts_lifetime`)。この間、他のボリュームまたはディスクには移動されません。したがって、部品が最終的に除去されるまで、それらは占有されたディスク領域の評価のために考慮される。

## データ複製

複製がサポートされる唯一のためのテーブルのMergeTree家族:

- 複製マージツリー
- 複製されたサミングマージツリー
- レプリケートリプレースマージツリー
- 複製された集合マージツリー
- レプリケートコラピングマージツリー
- ReplicatedVersionedCollapsingMergetree
- レプリケートグラフィティマージツリー

複製の作品のレベルを個別のテーブルではなく、全体のサーバーです。サーバーでの店舗も複製、非複製のテーブルでも同時に行います。

複製はシャーディングに依存しません。各シャードには、独自の独立した複製があります。

圧縮されたデータ `INSERT` と `ALTER` クエリを複製(詳細については、ドキュメンテーションに [ALTER](#))。

`CREATE`, `DROP`, `ATTACH`, `DETACH` と `RENAME` クエリは单一サーバーで実行され、レプリケートされません:

- その `CREATE TABLE` クエリは、クエリが実行されるサーバー上に新しい複製可能テーブルを作成します。このテーブルが既にあるその他のサーバーを加え新たなレプリカ。
- その `DROP TABLE` クエリは、クエリが実行されるサーバー上にあるレプリカを削除します。
- その `RENAME query` は、レプリカのいずれかのテーブルの名前を変更します。つまり、複製のテーブルでの異なる名称の異なるレプリカ。

ClickHouseの使用 [アパッチの飼育係](#) レプリカのメタ情報を格納するため。ZooKeeperバージョン3.4.5以降を使用します。

レプリケーションを使用するには、[飼育係](#) サーバー構成セクション。

### 注意

セキュリティ設定を無視しないでください クリックハウスは `digest ACL` キーム ZooKeeperセキュリティサブシステムの。

ZooKeeperクラスタのアドレスを設定する例:

```

<zookeeper>
  <node index="1">
    <host>example1</host>
    <port>2181</port>
  </node>
  <node index="2">
    <host>example2</host>
    <port>2181</port>
  </node>
  <node index="3">
    <host>example3</host>
    <port>2181</port>
  </node>
</zookeeper>

```

既存のZooKeeperクラスタを指定すると、システムはそのクラスタ上のディレクトリを独自のデータとして使用します（複製可能なテーブルを作成するときに）

場合飼育係な設定コンフィグファイルを創り上げられないんで再現しテーブル、および既存の複製のテーブル読み取り専用になります。

飼育係はで使用されていません `SELECT` レプリケーションのパフォーマンス `SELECT` また、クエリは非レプリケートテーブルの場合と同様に高速に実行されます。時の照会に配布再現し、テーブルClickHouse行動制御の設定 `max_replica_delay_for_distributed_queries` と `フォールバック_to_stale_replicas_for_distributed_queries`.

それぞれのため `INSERT` クエリー、契約時に応募を追加飼育係を務取引等（より正確には、これは挿入されたデータの各ブロックに対するものです。`max_insert_block_size = 1048576` 行。）これは、`INSERT` 非レプリケートテーブルと比較します。しかし、推奨事項に従ってデータを複数のバッチで挿入する場合 `INSERT` 毎秒、それは問題を作成しません。一つのZooKeeperクラスターを調整するために使用されるClickHouseクラスター全体の合計は数百です `INSERTs` 毎秒データ挿入のスループット(秒あたりの行数)は、レプリケートされていないデータの場合と同じくらい高くなります。

非常に大きなクラスタの場合、異なるシャードに異なるZooKeeperクラスタを使用できます。しかし、これはYandexの上で必要な証明されていません。メトリカクラスタ（約300台）。

複製は非同期でマルチマスターです。`INSERT` クエリ(および `ALTER`) 利用可能な任意のサーバに送信することができます。データは、クエリが実行されるサーバーに挿入され、他のサーバーにコピーされます。ですので非同期であるため、最近では挿入されたデータが表示され、その他のレプリカとの待ち時間をゼロにすることに レプリカの一部が使用できない場合、データは使用可能になったときに書き込まれます。レプリカが使用可能な場合、待機時間は、圧縮されたデータのブロックをネットワーク経由で転送するのにかかる時間です。

既定では、挿入クエリは、单一のレプリカからのデータの書き込みの確認を待機します。データが得られない場合には成功した記述につのレプリカのサーバーのこのレプリカが消滅し、保存したデータは失われます。複数のレプリカからのデータ書き込みの確認の取得を有効にするには、`insert_quorum` オプション

データの各ブロックは原子的に書き込まれます。挿入クエリは、次のブロックに分割されます `max_insert_block_size = 1048576` 行。言い換えれば、`INSERT` クエリには1048576行未満があり、アトミックに作成されます。

データブロックは重複排除されます。同じデータブロック（同じ順序で同じ行を含む同じサイズのデータブロック）の複数の書き込みの場合、ブロックは一度だけ書き込まれます。この理由は、クライアントアプリケーションがデータがDBに書き込まれたかどうかを知らないときにネットワーク障害が発生した場合です。`INSERT` クエリーするだけで簡単に繰り返します。どのレプリカ挿入が同一のデータで送信されたかは関係ありません。`INSERTs` 署等である。重複排除圧縮パラメータの制御 `merge_tree` サーバー設定。

複製時に、元のデータの挿入には転送されます。さらなるデータ変換(マージ)は、すべてのレプリカで同じ方法で調整され、実行されます。つまり、レプリカが異なるデータセンターに存在する場合、レプリケーションは適切に機能します。レプリケーションの主な目的は、異なるデータセンターでデータを複製することです。)

同じデータの任意の数のレプリカを持つことができます。Yandex.Metricaは本番環境で二重複製を使用します。各サーバーはRAID-5またはRAID-6を使用し、場合によってはRAID-10を使用します。これは比較的の信頼性が高く便利な解決策です。

システムは、レプリカのデータ同期を監視し、障害発生後に回復することができます。フェールオーバーは、自動(データのわずかな違いの場合)または半自動(データの異なりが多すぎる場合、構成エラーを示す可能性があります)です。

## 複製テーブルの作成

その `Replicated` テーブルエンジン名に接頭辞が追加されます。例えば:`ReplicatedMergeTree`.

### 複製\*`MergeTree`パラメータ

- `zoo_path` — The path to the table in ZooKeeper.
- `replica_name` — The replica name in ZooKeeper.

例:

```
CREATE TABLE table_name
(
    EventDate DateTime,
    CounterID UInt32,
    UserID UInt32
) ENGINE = ReplicatedMergeTree('/clickhouse/tables/{layer}-{shard}/table_name', '{replica}')
PARTITION BY toYYYYMM(EventDate)
ORDER BY (CounterID, EventDate, intHash32(UserID))
SAMPLE BY intHash32(UserID)
```

### ▶ 非推奨の構文の例

その例としては、これらのパラメータを含むことができ換巻きていただけるボディーです。置き換えられた値は、「macros」設定ファイルのセクション。例:

```
<macros>
    <layer>05</layer>
    <shard>02</shard>
    <replica>example05-02-1.yandex.ru</replica>
</macros>
```

の表の飼育係るべきで機能していませんが将来的には再現します。テーブルの異なる資料は異なる。  
この場合、パスは次の部分で構成されます:

`/clickhouse/tables/` 共通の接頭辞です。の使用をお勧めします。

`{layer}-{shard}` シャード識別子です。この例では、Yandexでの、二つの部分で構成されています。Metricaクラスターの使用インターネット上のファイル転送sharding. ほとんどのタスクでは、`{shard}`置換だけを残すことができます。

`table_name` ZooKeeper内のテーブルのノード名です。テーブル名と同じにすることをお勧めします。これは、テーブル名とは対照的に、名前変更クエリの後に変更されないため、明示的に定義されています。

**HINT:** データベース名を追加することができます `table_name` 同様に。例えば `db_name.table_name`

のレプリカの名前を識別のレプリカと同じ。これには、例のようにサーバー名を使用できます。名前は、各シャード内で一意である必要があります。

置換を使用する代わりに、パラメーターを明示的に定義できます。これは、テストや小さなクラスターの構成に便利です。ただし、分散DDLクエリは使用できません (ON CLUSTER) この場合。

大規模なクラスターで作業する場合は、エラーの可能性を減らすため、置換を使用することをお勧めします。

実行 `CREATE TABLE` 各レプリカのクエリ。このクエリを複製テーブルを追加し、新たなレプリカは、既存します。

テーブルに他のレプリカのデータがすでに含まれている後に新しいレプリカを追加すると、クエリの実行後にデータが他のレプリカから新しいレプリカ つまり、新しいレプリカは他のレプリカと同期します。

レプリカを削除するには、`DROP TABLE`. However, only one replica is deleted – the one that resides on the server where you run the query.

## 障害後の回復

場合によっては、サーバは、複製のテーブルスイッチ読み取り専用モードになります。システムは定期的に ZooKeeperへの接続を試みます。

飼育係が中に使用できない場合 `INSERT`、またはZooKeeperとの対話時にエラーが発生すると、例外がスローされます。 ZooKeeperに接続すると、ローカルファイルシステム内のデータセットが予想されるデータセットと一致するかどうかがチェックされます(ZooKeeperはこの情報を格納し がある場合は軽微な不整合の解消による同期データのレプリカ)。

システムが壊れたデータ部分(ファイルのサイズが間違っている)または認識できない部分(ファイルシステムに書き込まれているが、ZooKeeperに記録されて `detached` サブディレクトリ(削除されません)。他の部分がコピーからのレプリカ)。

ClickHouseは、大量のデータを自動的に削除するなどの破壊的な操作は実行しません。

サーバが起動時に表示されます(または新たに設立し、セッションとの飼育係)でのみチェックの量やサイズのすべてのファイルです。ファイルサイズは一致するが、途中でバイトが変更された場合、これはすぐには検出されず、データを読み取ろうとしたときにのみ検出されます。 `SELECT` クエリ。 クエリが例外をスローしつつ、非マッチングのチェックサムはサイズに圧縮されたブロックです。この場合、データパートは検証キューに追加され、必要に応じてレプリカからコピーされます。

場合には地元のデータセットが異なりすぎとされ、安全機構を起動します。サーバーはこれをログに入力し、起動を拒否します。この理由は、シャード上のレプリカが別のシャード上のレプリカのように誤って構成された場合など、このケースが構成エラーを示す可能性があるためです。しかし、しきい値をこの機構の設定かなり低く、こうした状況が起る中で、失敗を回復しました。この場合、データは半自動的に復元されます。“pushing a button”。

回復を開始するには、ノードを作成します `/path_to_table/replica_name flags/force_restore_data` で飼育係とコンテンツ、またはコマンドを実行し復元すべての複製のテーブル:

```
sudo -u clickhouse touch /var/lib/clickhouse/flags/force_restore_data
```

次に、サーバーを再起動します。起動時に、サーバーはこれらのフラグを削除し、回復を開始します。

## 完全なデータ損失の後の回復

すべてのデータやメタデータ消えたらサーバには、次の手順に従ってください復興:

1. ClickHouseをサーバにインストールします。シャード識別子とレプリカを使用する場合は、設定ファイルで置換を正しく定義します。
2. サーバー上で手動で複製する必要がある未複製のテーブルがある場合は、レプリカからデータをコピーします(ディレクトリ内)。`/var/lib/clickhouse/data/db_name/table_name/`.
3. テーブル定義のコピー `/var/lib/clickhouse/metadata/` レプリカから。シャード識別子またはレプリカ識別子がテーブル定義で明示的に定義されている場合は、このレプリカに対応するように修正します。(または、サーバーを起動し、すべての `ATTACH TABLE` にあったはずのクエリ.sqlファイル `/var/lib/clickhouse/metadata/.`)
4. 回復を開始するには、ZooKeeperノードを作成します `/path_to_table/replica_name flags/force_restore_data` コマンドを実行してレプリケートされたテーブルをすべて復元します: `sudo -u clickhouse touch /var/lib/clickhouse/flags/force_restore_data`

次に、サーバーを起動します(すでに実行されている場合は再起動します)。データダウンロードからのレプリカ。

代替の回復オプションは削除に関する情報は失われたレプリカから飼育係 (`/path_to_table/replica_name`)で説明したように、レプリカを再度作成します “[複製テーブルの作成](#)”。

復旧時のネットワーク帯域幅に制限はありません。一度に多数のレプリカを復元する場合は、この点に注意してください。

## MergeTreeからReplicatedMergeTreeへの変換

我々はこの用語を使用する `MergeTree` すべてのテーブルエンジンを参照するには `MergeTree family` の場合と同じです。`ReplicatedMergeTree`。

あなたが持っていた場合 `MergeTree` 手動で複製されたテーブルは、複製されたテーブルに変換することができます。すでに大量のデータを収集している場合は、これを行う必要があるかもしれません。`MergeTree` レプリケーションを有効にします。

さまざまなレプリカでデータが異なる場合は、最初に同期するか、このデータを除くすべてのレプリカで削除します。

既存の`MergeTree`テーブルの名前を変更し、次に `ReplicatedMergeTree` 古い名前のテーブル。

データを古いテーブルから `detached` サブディレクトリ内のディレクトリを新しいテーブルデータ (`/var/lib/clickhouse/data/db_name/table_name/`)。

その後、実行 `ALTER TABLE ATTACH PARTITION` これらのデータパーティットを作業セットに追加するレプリカのいずれかで。

## ReplicatedMergeTreeからMergeTreeへの変換

別の名前の`MergeTree`テーブルを作成します。ディレクトリからすべてのデータを移動します。`ReplicatedMergeTree` テーブルデータを新しいテーブルのデータディレクトリです。その後、削除 `ReplicatedMergeTree` サーバーを表にして再起動します。

あなたが取り除きたい場合は、`ReplicatedMergeTree` サーバーを起動しないテーブル:

- 対応するものを削除する `.sql` ファイルのメタデータディレクトリ (`/var/lib/clickhouse/metadata/`)。
- ZooKeeperで対応するパスを削除します (`/path_to_table/replica_name`)。

この後、サーバーを起動し、`MergeTree` データをそのディレクトリに移動し、サーバーを再起動します。

## Zookeeperクラスター内のメタデータが紛失または破損した場合の復旧

ZooKeeperのデータが紛失または破損している場合は、上記のように再生されていないテーブルに移動することでデータを保存できます。

## カスタム分割キー

パーティション分割は、[メルゲツリー](#) 家族テーブル（含む 複製 テーブル）。[実体化ビュー](#)に基づく`MergeTree`テーブル支援を分割します。

パーティションは、指定された条件によるテーブル内のレコードの論理的な組合せです。パーティションは、月別、日別、イベントタイプ別など、任意の条件で設定できます。各パーティションは別に保存される簡単操作のデータです。アクセス時のデータClickHouseの最小サブセットのパーティションは可能です。

パーティションは `PARTITION BY expr` 句とき [テーブルの作成](#)。これはパーティションキーにすることはでき表現からのテーブル列あります。例えば、指定ヨ月の表現を使用 `toYYYYMM(date_column)`:

```

CREATE TABLE visits
(
    VisitDate Date,
    Hour UInt8,
    ClientID UUID
)
ENGINE = MergeTree()
PARTITION BY toYYYYMM(VisitDate)
ORDER BY Hour;

```

パーティションキーは、式のタプルにすることもできます。[主キー](#))。例えば:

```

ENGINE = ReplicatedCollapsingMergeTree('/clickhouse/tables/name', 'replica1', Sign)
PARTITION BY (toMonday(StartDate), EventType)
ORDER BY (CounterID, StartDate, intHash32(userID));

```

この例では、現在の週に発生したイベントの種類によってパーティション分割を設定します。

挿入する際に新しいデータテーブルにこのデータを保存することで別パートとして（個）[-field-list](#)順にソートその有効なタイプを利用します。挿入後10-15分で、同じパーティションの部分が部分全体にマージされます。

## 情報

マージは、パーティション分割式の値が同じデータパートに対してのみ機能します。つまり なんかを過度に粒状仕切り（千約以上のパーティション）。それ以外の場合は、[SELECT](#) ファイルシステムおよびオープンファイル記述子に不当に多数のファイルがあるため、クエリの実行が不十分です。

使用する [システム部品](#) 表パートとパーティションを表示する表。たとえば、のは、我々が持っていると仮定しましょう `visits` テーブルを分割する。のは、実行してみましょう [SELECT](#) のクエリ `system.parts` テーブル:

```

SELECT
    partition,
    name,
    active
FROM system.parts
WHERE table = 'visits'

```

partition	name	active
201901	201901_1_3_1	0
201901	201901_1_9_2	1
201901	201901_8_8_0	0
201901	201901_9_9_0	0
201902	201902_4_6_1	1
201902	201902_10_10_0	1
201902	201902_11_11_0	1

その `partition` 列にはパーティションの名前が含まれます。あるパーティション例: `201901` と `201902`。この列の値を使用して、パーティション名を指定できます [ALTER ... PARTITION](#) クエリ。

その `name` カラムの名前を格納して、パーティションのデータ部品です。この列を使用して、パートの名前を指定することができます。 [ALTER ATTACH PART](#) クエリ。

最初の部分の名前を分解しましょう: `201901_1_3_1`:

- `201901` パーティション名です。
- `1` データブロックの最小数です。

- 3 データブロックの最大数です。
- 1 チャンクレベル(形成されるマージツリーの深さ)です。

## 情報

古いタイプのテーブルの部分には名前があります: 20190117\_20190123\_2\_2\_0 (最小日-最大日-最小ブロック番号-最大ブロック番号-レベル)。

その `active` 列は部品の状態を示します。1 アクティブです; 0 非アクティブです。非アクティブな部分は、たとえば、より大きな部分にマージした後に残るソース部分です。破損したデータ部分も非アクティブとして示されます。

この例でわかるように、同じパーティションにはいくつかの分離された部分があります (たとえば, 201901\_1\_3\_1 と 201901\_1\_9\_2)。つまり、これらの部分はまだマージされていません。ClickHouseは、データの挿入された部分を定期的にマージし、挿入の約15分後にマージします。また、スケジュールされていないマージを実行するには `OPTIMIZE` クエリ。例:

```
OPTIMIZE TABLE visits PARTITION 201902;
```

partition	name	active
201901	201901_1_3_1	0
201901	201901_1_9_2	1
201901	201901_8_8_0	0
201901	201901_9_9_0	0
201902	201902_4_6_1	0
201902	201902_4_11_2	1
201902	201902_10_10_0	0
201902	201902_11_11_0	0

非アクティブな部分は、マージ後約10分で削除されます。

パートとパーティションのセットを表示する別 の方法は、テーブルのディレクトリに移動します:  
`/var/lib/clickhouse/data/<database>/<table>/`. 例えば:

```
/var/lib/clickhouse/data/default/visits$ ls -l
total 40
drwxr-xr-x 2 clickhouse clickhouse 4096 Feb 1 16:48 201901_1_3_1
drwxr-xr-x 2 clickhouse clickhouse 4096 Feb 5 16:17 201901_1_9_2
drwxr-xr-x 2 clickhouse clickhouse 4096 Feb 5 15:52 201901_8_8_0
drwxr-xr-x 2 clickhouse clickhouse 4096 Feb 5 15:52 201901_9_9_0
drwxr-xr-x 2 clickhouse clickhouse 4096 Feb 5 16:17 201902_10_10_0
drwxr-xr-x 2 clickhouse clickhouse 4096 Feb 5 16:17 201902_11_11_0
drwxr-xr-x 2 clickhouse clickhouse 4096 Feb 5 16:19 201902_4_11_2
drwxr-xr-x 2 clickhouse clickhouse 4096 Feb 5 12:09 201902_4_6_1
drwxr-xr-x 2 clickhouse clickhouse 4096 Feb 1 16:48 detached
```

フォルダ '201901\_1\_1\_0', '201901\_1\_7\_1' そして、部品のディレクトリです。各部に関する対応する分割データが含まれまで一定の月のテーブルこの例では、分割による。

その `detached` ディレクトリに含まれる部品のことがあったかを使って、テーブル `DETACH` クエリ。破損した部分も、削除されるのではなく、このディレクトリに移動されます。サーバーは、サーバーからの部品を使用しません。  
`detached directory. You can add, delete, or modify the data in this directory at any time – the server will not know about this until you run the ATTACH` クエリ。

オペレーティングサーバーでは、ファイルシステム上の部品のセットまたはそのデータを手動で変更することはできません。非複製のテーブル、これを実行する事ができます。サーバが停止中でないお勧めします。そのための複製のテーブルはパーティのセットの変更はできません。

ClickHouseでは、パーティションの削除、テーブル間のコピー、またはバックアップの作成などの操作を実行できます。セクションのすべての操作の一覧を参照してください [パーティションとパーティツの操作](#)。

## 置換マージツリー

エンジンは [メルゲツリー](#) 同じ主キーを持つ重複したエントリを削除するという点で、より正確には同じです [ソートキー](#) 値)。

データ重複除外は、マージ中にのみ発生します。マージは未知の間にバックグラウンドで発生するため、計画することはできません。一部のデータは未処理のままになる場合があります。スケジュールされていないマージを実行するには [OPTIMIZE](#) クエリは、それを使用してカウントされませんので、[OPTIMIZE](#) クエリは大量のデータを読み書きします。

従って、[ReplacingMergeTree](#) に適した清算出重複データを背景に保存するための空間が保証するものではありませんが重複している。

## テーブルの作成

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = ReplacingMergeTree([ver])
[PARTITION BY expr]
[ORDER BY expr]
[PRIMARY KEY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

説明リクエストパラメータの参照 [要求の説明](#)。

### ReplacingMergeTreeパラメータ

- `ver` — column with version. Type `UInt*`, `Date` または `DateTime`. 任意パラメータ。

マージ時、[ReplacingMergeTree](#) 同じ主キーを持つすべての行から、一つだけを残します:

- 選択範囲の最後にある場合 `ver` 設定されていません。
- 最大バージョンでは、次の場合 `ver` 指定。

### クエリ句

を作成するとき [ReplacingMergeTree](#) 同じテーブル `句` を作成するときのように必要です。[MergeTree](#) テーブル。

▶ 推奨されていません法テーブルを作成する

## サミングマージツリー

エンジンはから継承します [メルゲツリー](#)。違いは、データ部分をマージするときに [SummingMergeTree](#) テーブル ClickHouseは、すべての行を同じ主キー（またはより正確には同じキー）で置き換えます [ソートキー](#)）数値データ型の列の集計値を含む行。並べ替えキーが単一のキーが多数の行に対応するように構成されている場合、ストレージ容量が大幅に削減され、データ選択が高速化されます。

エンジンと一緒に使用することをお勧めします [MergeTree](#)。完全なデータを格納する [MergeTree](#) テーブルおよび使用 [SummingMergeTree](#) たとえば、レポートの準備時など、集計データを格納する場合。このようなアプローチを防ぎまら貴重なデータにより正しく構成しその有効なタイプを利用します。

# テーブルの作成

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = SummingMergeTree([columns])
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

説明リクエストパラメータの参照 [要求の説明](#).

## Summingmergetreeのパラメータ

- `columns` - 値が集計される列の名前を持つタプル。任意パラメータ。  
列は数値型である必要があります、主キーには含まれていない必要があります。

もし `columns` 指定されていないClickHouseは、主キーにない数値データ型を持つすべての列の値を集計します。

## クエリ句

を作成するとき `SummingMergeTree` 同じテーブル [句](#) を作成するときのように必要です。`MergeTree` テーブル。

▶ 推奨されていません法テーブルを作成する

## 使用例

次の表を考えます:

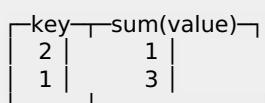
```
CREATE TABLE summtt
(
    key UInt32,
    value UInt32
)
ENGINE = SummingMergeTree()
ORDER BY key
```

データを挿入する:

```
INSERT INTO summtt Values(1,1),(1,2),(2,1)
```

ClickHouseは完全ではないすべての行を合計可能性があります ([以下を参照](#)) ので、集計関数を使用します `sum` と `GROUP BY` クエリ内の句。

```
SELECT key, sum(value) FROM summtt GROUP BY key
```



## データ処理

データがテーブルに挿入されると、そのまま保存されます。ClickHouseは、データの挿入された部分を定期的にマージし、これは、同じ主キーを持つ行が合計され、結果として得られるデータの各部分に対して行に置き換えられる

ClickHouse can merge the data parts so that different resulting parts of data cat consist rows with the same primary key, i.e. the summation will be incomplete. Therefore (SELECT)集計関数 **和()** と GROUP BY 上記の例で説明したように、クエリで句を使用する必要があります。

## 合計の共通ルール

数値データ型の列の値が集計されます。列のセットは、パラメータによって定義されます **columns**.

合計のすべての列の値が0の場合、行は削除されます。

Columnが主キーになく、集計されていない場合は、既存の値から任意の値が選択されます。

主キーの列の値は集計されません。

## Aggregatefunction列の合計

の列に対して **AggregateFunction** タイプ ClickHouseとして振る舞うと考えられてい **AggregatingMergeTree** 機能に従って集計するエンジン。

### 入れ子構造

テーブルでネストしたデータ構造と加工"と言われています。

入れ子になったテーブルの名前が **Map** また、以下の条件を満たす少なくとも二つの列が含まれています:

- 最初の列は数値です (**\*Int\***, Date, DateTime) または文字列 (String, FixedString) それを呼びましょう **key**,
- 他の列は算術演算です (**\*Int\***, Float32/64) それを呼びましょう **(values...)**,

次に、この入れ子になったテーブルは **key => (values...)** その行をマージするとき、二つのデータセットの要素は **key** 対応するの合計と **(values...)**.

例:

```
[(1, 100)] + [(2, 150)] -> [(1, 100), (2, 150)]
[(1, 100)] + [(1, 150)] -> [(1, 250)]
[(1, 100)] + [(1, 150), (2, 150)] -> [(1, 250), (2, 150)]
[(1, 100), (2, 150)] + [(1, -100)] -> [(2, 150)]
```

データを要求するときは、**sumMap(キー、値)** の集合のための関数 **Map**.

入れ子になったデータ構造の場合、合計のために列のタプルにその列を指定する必要はありません。

## Aggregatingmergetree

エンジンはから継承します **マルゲツリー**、データ部分のマージのロジックを変更する。ClickHouseは、すべての行を同じ主キー（またはより正確には同じキー）で置き換えます **ソートキー**）集計関数の状態の組み合わせを格納する单一の行（一つのデータ部分内）を持つ。

以下を使用できます **AggregatingMergeTree** 集計されたマテリアライズドビューを含む、増分データ集計用の表。

エンジンは、次の型のすべての列を処理します:

- AggregateFunction**
- SimpleAggregateFunction**

使用することは適切です `AggregatingMergeTree` 注文によって行数を減らす場合。

## テーブルの作成

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = AggregatingMergeTree()
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[TTL expr]
[SETTINGS name=value, ...]
```

説明リクエストパラメータの参照 [要求の説明](#).

### クエリ句

を作成するとき `AggregatingMergeTree` 同じテーブル `句` を作成するときのように必要です。`MergeTree` テーブル。

▶ 推奨されていません法テーブルを作成する

## 選択と挿入

データを挿入するには、`INSERT SELECT aggregate-State-functions`を使用したクエリ。

データを選択するとき `AggregatingMergeTree` テーブル、使用 `GROUP BY` データを挿入するときと同じ集計関数ですが、`-Merge` 接尾辞。

の結果 `SELECT` クエリ、の値 `AggregateFunction type` は、すべての ClickHouse 出力形式に対して実装固有のバイナリ表現を持ちます。たとえば、データをダンプする場合、`TabSeparated` フォーマット `SELECT` 次に、このダンプを次のようにロードします `INSERT` クエリ。

## 集約マテリアライズドビューの例

`AggregatingMergeTree` これは、`test.visits` テーブル:

```
CREATE MATERIALIZED VIEW test.basic
ENGINE = AggregatingMergeTree() PARTITION BY toYYYYMM(StartDate) ORDER BY (CounterID, StartDate)
AS SELECT
    CounterID,
    StartDate,
    sumState(Sign) AS Visits,
    uniqState(UserID) AS Users
FROM test.visits
GROUP BY CounterID, StartDate;
```

にデータを挿入する `test.visits` テーブル。

```
INSERT INTO test.visits ...
```

データはテーブルとビューの両方に挿入されます `test.basic` それは集計を実行します。

集計データを取得するには、次のようなクエリを実行する必要があります `SELECT ... GROUP BY ...` ビューから `test.basic`:

```
SELECT
    StartDate,
    sumMerge(Visits) AS Visits,
    uniqMerge(Users) AS Users
FROM test.basic
GROUP BY StartDate
ORDER BY StartDate;
```

## 折りたたみマージツリー

エンジンはから継承します **マルゲツリー** 加算の論理行の崩壊データ部品の統合アルゴリズムです。

**CollapsingMergeTree** 並べ替えキー内のすべてのフィールドの場合、行のペアを非同期に削除(折りたたみ)します (**ORDER BY**) は、特定のフィールドを除いて同等です **Sign** これは **1** と **-1** 値。ペアのない行は保持されます。 詳細については、**崩壊** 文書のセクション。

エンジンはかなり貯蔵の容積を減らし、効率をの高めるかもしません **SELECT** 結果としてのクエリ。

## テーブルの作成

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = CollapsingMergeTree(sign)
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

クエリパラメータの説明については、[クエリの説明](#).

### **CollapsingMergeTree**パラメータ

- **sign** — Name of the column with the type of row: **1** は “state” 行, **-1** は “cancel” ロウ  
Column data type — **Int8**.

### クエリ句

を作成するとき **CollapsingMergeTree** テーブル、同じ **クエリ句** を作成するときのように必要です。**MergeTree** テーブル。

▶ 推奨されていません法テーブルを作成する

## 崩壊

### データ

考える必要がある状況などが保存継続的に変化するデータのオブジェクトです。オブジェクトの行を一つ持ち、変更時に更新するのは論理的ですが、DBMSではストレージ内のデータを書き換える必要があるため、更新操作はコストがかかる場合にデータを書き込むには、迅速に更新できませんが、きの変化をオブジェクトの順にしております。

特定の列を使用する **Sign**. もし **Sign = 1** これは、行がオブジェクトの状態であることを意味します。“state” ロウもし **Sign = -1** これは、同じ属性を持つオブジェクトの状態の取り消しを意味します。“cancel” ロウ

たとえば、ユーザーがあるサイトでチェックしたページ数と、そこにいた時間を計算したいとします。ある時点で、ユーザーアクティビティの状態で次の行を書きます:

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	1

ある時点で、後で我々は、ユーザーの活動の変更を登録し、次の二つの行でそれを書きます。

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	-1
4324182021466249494	6	185	1

最初の行は、オブジェクト(ユーザー)の以前の状態を取り消します。キャンセルされた状態の並べ替えキーのフィールドをコピーします。Sign。

次の行には現在の状態が含まれます。

ユーザーアクティビティの最後の状態だけが必要なので、行

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	1
4324182021466249494	5	146	-1

オブジェクトの無効な(古い)状態を折りたたんで削除することができます。CollapsingMergeTreeデータパーティのマージ中にこれを行います。

なぜ私たちは、読み込まれた各変更のための2行が必要ですアルゴリズム段落。

このようなアプローチ

1. プログラムを書き込み、データ意のオブジェクトをキャンセルはできます。“Cancel”文字列はコピーのソートキーの分野において“state”文字列とその逆Sign。この増加の初期サイズでの保存が可能なデータを書き込む。
2. 列の長い成長配列は、書き込みの負荷によるエンジンの効率を低下させます。より簡単なデータ、より高い効率。
3. そのSELECT結果は、オブジェクト変更履歴の一貫性に強く依存します。挿入のためのデータを準備するときは正確です。たとえば、セッションの深さなど、負でない指標の負の値など、不整合なデータで予測不可能な結果を得ることができます。

## アルゴリズム

ClickHouseがデータパーティをマージすると、同じ並べ替えキーを持つ連続した行の各グループ(ORDER BY)は、以下の二つの行に縮小されます。Sign = 1 (“state”行) と別のSign = -1 (“cancel”行)。つまり、エントリは崩壊します。

各々、その結果得られたデータは部分ClickHouse省:

1. 最初の“cancel”そして最後の“state”行の数が“state”と“cancel”行は一致し、最後の行は“state”ロウ
2. 最後の“state”より多くがあれば、行“state”行より“cancel”行。
3. 最初の“cancel”より多くがあれば、行“cancel”行より“state”行。
4. 他のすべての場合には、行のいずれも。

また、少なくとも2以上がある場合“state”行より“cancel”行、または少なくとも2以上“cancel”次の行“state”マージは続行されますが、ClickHouseはこの状況を論理エラーとして扱い、サーバーログに記録します。同じデータが複数回挿入された場合、このエラーが発生する可能性があります。

したがって、崩壊は統計の計算結果を変えてはならない。

変更は徐々に崩壊し、最終的にはほぼすべてのオブジェクトの最後の状態だけが残った。

その `Sign` マージアルゴリズムでは、同じ並べ替えキーを持つすべての行が同じ結果データ部分にあり、同じ物理サーバー上にすることを保証するものではないため、必 ClickHouseプロセス `SELECT` 複数のスレッドでクエリを実行し、結果の行の順序を予測することはできません。完全に取得する必要がある場合は、集計が必要です “collapsed” データから `CollapsingMergeTree` テーブル。

折りたたみを終了するには、次のクエリを記述します `GROUP BY` 符号を説明する句および集計関数。たとえば、数量を計算するには、次を使用します `sum(Sign)` 代わりに `count()`. 何かの合計を計算するには、次のようにします `sum(Sign * x)` 代わりに `sum(x)`、というように、また、追加 `HAVING sum(Sign) > 0`.

集計 `count`, `sum` と `avg` この方法で計算できます。集計 `uniq` 算出できる場合にはオブジェクトは、少なくとも一つの状態はまだ崩れていない。集計 `min` と `max` 計算できませんでした。`CollapsingMergeTree` 折りたたまれた状態の値の履歴は保存されません。

集計なしでデータを抽出する必要がある場合(たとえば、特定の条件に一致する最新の値を持つ行が存在するかどうかをチェックする場合)には、次の `FINAL` モディファイア `FROM` 句。このアプローチは、大幅に低効率です。

## 使用例

データ例:

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	1
4324182021466249494	5	146	-1
4324182021466249494	6	185	1

テーブルの作成:

```
CREATE TABLE UAct
(
    UserID UInt64,
    PageViews UInt8,
    Duration UInt8,
    Sign Int8
)
ENGINE = CollapsingMergeTree(Sign)
ORDER BY UserID
```

データの挿入:

```
INSERT INTO UAct VALUES (4324182021466249494, 5, 146, 1)
```

```
INSERT INTO UAct VALUES (4324182021466249494, 5, 146, -1), (4324182021466249494, 6, 185, 1)
```

我々は二つを使用します `INSERT` 二つの異なるデータ部分を作成するクエリ。一つのクエリでデータを挿入すると、ClickHouseは一つのデータ部分を作成し、マージを実行しません。

データの取得:

```
SELECT * FROM UAct
```

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	-1
4324182021466249494	6	185	1

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	1

私たちは何を見て、どこで崩壊していますか？

二つと `INSERT` クエリーを作成し、2つのデータ部品です。その `SELECT` クエリは2つのスレッドで実行され、行のランダムな順序が得られました。データパートのマージがまだないため、折りたたみは発生しませんでした。ClickHouseは予測できない未知の瞬間にデータ部分をマージします。

したがって、集約が必要です：

```
SELECT
  UserID,
  sum(PageViews * Sign) AS PageViews,
  sum(Duration * Sign) AS Duration
FROM UAct
GROUP BY UserID
HAVING sum(Sign) > 0
```

UserID	PageViews	Duration
4324182021466249494	6	185

集約を必要とせず、強制的に崩壊させたい場合は、以下を使用できます `FINAL` 修飾子のための `FROM` 句。

```
SELECT * FROM UAct FINAL
```

UserID	PageViews	Duration	Sign
4324182021466249494	6	185	1

データを選択するこの方法は非常に非効率的です。大きなテーブルには使わないでください。

## 別のアプローチの例

データ例：

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	1
4324182021466249494	-5	-146	-1
4324182021466249494	6	185	1

アイデアは、マージが考慮にキーフィールドのみを取ることです。そして、“Cancel”行我々は、符号列を使用せずに合計するときに行の以前のバージョンを等しく負の値を指定することができます。この方法では、データ型を変更する必要があります `PageViews, Duration uint8->Int16` の負の値を格納する。

```

CREATE TABLE UAct
(
    UserID UInt64,
    PageViews Int16,
    Duration Int16,
    Sign Int8
)
ENGINE = CollapsingMergeTree(Sign)
ORDER BY UserID

```

ましょう試験のアプローチ:

```

insert into UAct values(4324182021466249494, 5, 146, 1);
insert into UAct values(4324182021466249494, -5, -146, -1);
insert into UAct values(4324182021466249494, 6, 185, 1);

select * from UAct final; // avoid using final in production (just for a test or small tables)

```

UserID	PageViews	Duration	Sign
4324182021466249494	6	185	1

```

SELECT
    UserID,
    sum(PageViews) AS PageViews,
    sum(Duration) AS Duration
FROM UAct
GROUP BY UserID
``text

```

UserID	PageViews	Duration
4324182021466249494	6	185

```
select count() FROM UAct
```

count()
3

```
optimize table UAct final;
```

```
select * FROM UAct
```

UserID	PageViews	Duration	Sign
4324182021466249494	6	185	1

## バージョニングコラプシングマーケットリー

このエンジン:

- では迅速書き込みオブジェクトとは常に変化しています。
- 削除古いオブジェクト状態の背景になります。これにより、保管量が大幅に削減されます。

セクションを参照 [崩壊](#) 詳細については.

エンジンはから継承します **マルゲツリー** 追加した論理崩壊行のアルゴリズムのための統合データ部品です。 **VersionedCollapsingMergeTree** と同じ目的を果たしています **折りたたみマージツリー** が異なる崩壊のアルゴリズムを挿入し、データを任意の順番で複数のスレッド）。特に、**Version** 列は、間違った順序で挿入されても、行を適切に折りたたむのに役立ちます。対照的に、**CollapsingMergeTree** 厳密に連続した挿入のみを許可します。

## テーブルの作成

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = VersionedCollapsingMergeTree(sign, version)
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

クエリパラメータの説明については、[クエリの説明](#)。

### エンジン変数

**VersionedCollapsingMergeTree(sign, version)**

- **sign** — Name of the column with the type of row: **1** は “state” 行, **-1** は “cancel” ロウ

列のデータ型は次のとおりです **Int8**.

- **version** — Name of the column with the version of the object state.

列のデータ型は次のとおりです **UInt\***.

### クエリ句

を作成するとき **VersionedCollapsingMergeTree** テーブル、同じ **句** を作成するときに必要です。**MergeTree** テーブル。

▶ 推奨されていません法テーブルを作成する

## 崩壊

### データ

うな状況を考える必要がある場合の保存継続的に変化するデータのオブジェクトです。オブジェクトに対して一つの行を持ち、変更があるたびにその行を更新するのが妥当です。ただし、DBMSではストレージ内のデータを書き換える必要があるため、更新操作はコストがかかり、時間がかかります。更新はできませんが必要な場合は書き込みデータはすでに書き込み、変更オブジェクトの順にしておきます。

使用する **Sign** 行を書き込むときの列。もし **Sign = 1** これは、行がオブジェクトの状態であることを意味します (“state” 行)。もし **Sign = -1** これは、同じ属性を持つオブジェクトの状態の取り消しを示します (“cancel” 行)。また、**Version** オブジェクトの各状態を個別の番号で識別する必要があります。

たとえば、ユーザーがいくつのサイトにアクセスしたのか、どのくらいの時間があったのかを計算します。ある時点で、次の行をユーザーアクティビティの状態で記述します:

UserID	PageViews	Duration	Sign	Version
4324182021466249494	5	146	1	1

後のある時点で、ユーザーアクティビティの変更を登録し、次の二つの行でそれを書きます。

UserID	PageViews	Duration	Sign	Version
4324182021466249494	5	146	-1	1
4324182021466249494	6	185	1	2

最初の行は、オブジェクト(ユーザー)の以前の状態を取り消します。キャンセルされた状態のすべてのフィールドをコピーします。 **Sign**.

次の行には現在の状態が含まれます。

ユーザーアクティビティの最後の状態だけが必要なので、行

UserID	PageViews	Duration	Sign	Version
4324182021466249494	5	146	1	1
4324182021466249494	5	146	-1	1

オブジェクトの無効な(古い)状態を折りたたんで、削除することができます。 **VersionedCollapsingMergeTree** このことを融合させたデータの部品です。

変更ごとに二つの行が必要な理由については、以下を参照してください [アルゴリズム](#).

#### 使用上の注意

1. プログラムを書き込み、データ意のオブジェクトを解除します。その“cancel”のコピーでなければなりません。“state”反対の文字列 **Sign**. この増加の初期サイズでの保存が可能なデータを書き込む。
2. 列の長い成長配列は、書き込みの負荷のためにエンジンの効率を低下させる。より簡単なデータ、より良い効率。
3. **SELECT** 結果は、オブジェクト変更の履歴の一貫性に強く依存します。挿入のためのデータを準備するときは正確です。セッションの深さなど、負でない指標の負の値など、不整合なデータで予測不可能な結果を得ることができます。

#### アルゴリズム

ClickHouseは、データパーティをマージするときに、同じ主キーとバージョンと異なる行の各ペアを削除します **Sign**. 行の順序は重要ではありません。

ClickHouseはデータを挿入するとき、主キーによって行を並べ替えます。もし **Version** 列が主キーにない場合、ClickHouseはそれを最後のフィールドとして暗黙的に主キーに追加し、順序付けに使用します。

#### データの選択

ClickHouseは、同じ主キーを持つすべての行が同じ結果のデータ部分または同じ物理サーバー上にあることを保証するものではありません。これは、データの書き込みとその後のデータ部分のマージの両方に当てはまります。さらに、ClickHouseプロセス **SELECT** 複数のスレッドを使用してクエリを実行し、結果の行の順序を予測することはできません。これは、完全に取得する必要がある場合に集約が必要であることを意味します “collapsed” aからのデータ **VersionedCollapsingMergeTree** テーブル。

折りたたみを終了するには、**GROUP BY** 符号を説明する句および集計関数。たとえば、数量を計算するには、次を使用します **sum(Sign)** 代わりに **count()**. 何かの合計を計算するには、次のようにします **sum(Sign \* x)** 代わりに **sum(x)** を追加します。 **HAVING sum(Sign) > 0.**

集計 **count**, **sum** と **avg** この方法で計算できます。集計 **uniq** オブジェクトが少なくとも一つの非折りたたみ状態を持つ場合に計算できます。集計 **min** と **max** 計算できないのは **VersionedCollapsingMergeTree** 折りたたまれた状態の値の履歴は保存されません。

データを抽出する必要がある場合は “collapsing” な集計(例えば、確認列が存在する最新の値に一致条件)を使用できます **FINAL** モディファイア **FROM** 句。このアプローチは非効率で利用すべきではありませんの大きさです。

# 使用例

データ例:

UserID	PageViews	Duration	Sign	Version
4324182021466249494	5	146	1	1
4324182021466249494	5	146	-1	1
4324182021466249494	6	185	1	2

テーブルの作成:

```
CREATE TABLE UAct
(
    UserID UInt64,
    PageViews UInt8,
    Duration UInt8,
    Sign Int8,
    Version UInt8
)
ENGINE = VersionedCollapsingMergeTree(Sign, Version)
ORDER BY UserID
```

データの挿入:

```
INSERT INTO UAct VALUES (4324182021466249494, 5, 146, 1, 1)
```

```
INSERT INTO UAct VALUES (4324182021466249494, 5, 146, -1, 1),(4324182021466249494, 6, 185, 1, 2)
```

我々は二つを使用します `INSERT` 二つの異なるデータ部分を作成するクエリ。 単一のクエリでデータを挿入すると、ClickHouseは一つのデータパーティットを作成し、マージを実行することはありません。

データの取得:

```
SELECT * FROM UAct
```

UserID	PageViews	Duration	Sign	Version
4324182021466249494	5	146	1	1
4324182021466249494	5	146	-1	1
4324182021466249494	6	185	1	2

私たちがここで何を見て、崩壊した部分はどこにありますか？

二つのデータパーティットを作成しました `INSERT` クエリ。 その `SELECT` クエリは二つのスレッドで実行され、結果は行のランダムな順序です。

崩壊の発生はありませんので、データのパーティットがなされております。 ClickHouseは、予測できない未知の時点でのデータパーティットをマージします。

これが集約が必要な理由です:

```

SELECT
    UserID,
    sum(PageViews * Sign) AS PageViews,
    sum(Duration * Sign) AS Duration,
    Version
FROM UAct
GROUP BY UserID, Version
HAVING sum(Sign) > 0

```

UserID	PageViews	Duration	Version
4324182021466249494	6	185	2

集約を必要とせず、強制的に折りたたみたいなら、以下を使うことができます **FINAL** モディファイア **FROM** 句。

```
SELECT * FROM UAct FINAL
```

UserID	PageViews	Duration	Sign	Version
4324182021466249494	6	185	1	2

これは、データを選択する非常に非効率的な方法です。大きなテーブルには使用しないでください。

## GraphiteMergeTree

このエンジン) **黒鉛** データ これは、GraphiteのデータストアとしてClickHouseを使用したい開発者にとって役立つかもしれません。

を利用できますClickHouseテーブルエンジンの黒鉛のデータが必要ない場合rollupが必要な場合は、rollupを使用 GraphiteMergeTree. エンジンは貯蔵量を減らし、グラフアイトからの照会の効率を高めます。

エンジンはプロパティを **メルゲツリー**.

### テーブルの作成

```

CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    Path String,
    Time DateTime,
    Value <Numeric_type>,
    Version <Numeric_type>
    ...
) ENGINE = GraphiteMergeTree(config_section)
[PARTITION BY expr]
[ORDER BY expr]
[SAMPLE BY expr]
[SETTINGS name=value, ...]

```

の詳細な説明を参照してください **CREATE TABLE** クエリ。

グラフアイトデータのテーブルには、次のデータの次の列が必要です:

- ミリ規格名 (グラフアイトセンサ) データ型: **String**.
- メトリックを測定する時間。 データ型: **DateTime**.
- メトリックの値。 データ型:任意の数値。

- 指標のバージョン。 データ型:任意の数値。

ClickHouseは、バージョンが同じ場合は、最も高いバージョンまたは最後に書かれた行を保存します。他の行は、データパートのマージ中に削除されます。

これらの列の名前は、ロールアップ構成で設定する必要があります。

## GraphiteMergeTreeパラメータ

- `config_section` — Name of the section in the configuration file, where are the rules of rollup set.

クエリ句

を作成するとき `GraphiteMergeTree` テーブル、同じ `句` を作成するときのように必要です。`MergeTree` テーブル。

▶ 推奨されていません法テーブルを作成する

## ロールアップ構成

ロールアップの設定は、`graphite_rollup` サーバー構成のパラメータ。パラメーターの名前は任意です。複数の構成を作成し、異なるテーブルに使用できます。

ロールアップ構成構造:

```
required-columns  
patterns
```

## 必要な列

- `path_column_name` — The name of the column storing the metric name (Graphite sensor). Default value: `Path`.
- `time_column_name` — The name of the column storing the time of measuring the metric. Default value: `Time`.
- `value_column_name` — The name of the column storing the value of the metric at the time set in `time_column_name`. デフォルト値: `Value`.
- `version_column_name` — The name of the column storing the version of the metric. Default value: `Timestamp`.

## パターン

の構造 `patterns` セクション:

```
pattern
  regexp
  function
pattern
  regexp
  age + precision
...
pattern
  regexp
  function
  age + precision
...
pattern
...
default
  function
  age + precision
...
```

## 注意

パターンは厳密に注文する必要が:

1. Patterns without **function** or **retention**.
2. Patterns with both **function** and **retention**.
3. Pattern **default**.

行を処理するとき、ClickHouseは **pattern** セクション それぞれの **pattern** (含む **default**)セクションには **function** 集計のパラメータ, **retention** 変数または両方。メトリック名が **regexp**、からのルール **pattern** セクション(またはセクション)が適用されます。 **default** セクションを使用します。

のフィールド **pattern** と **default** セクション:

- **regexp**- A pattern for the metric name.
- **age** - The minimum age of the data in seconds.
- **precision**- How precisely to define the age of the data in seconds. Should be a divisor for 86400 (seconds in a day).
- **function** - The name of the aggregating function to apply to data whose age falls within the range [age, age + precision].

## 設定例

```

<graphite_rollup>
  <version_column_name>Version</version_column_name>
  <pattern>
    <regexp>click_cost</regexp>
    <function>any</function>
    <retention>
      <age>0</age>
      <precision>5</precision>
    </retention>
    <retention>
      <age>86400</age>
      <precision>60</precision>
    </retention>
  </pattern>
  <default>
    <function>max</function>
    <retention>
      <age>0</age>
      <precision>60</precision>
    </retention>
    <retention>
      <age>3600</age>
      <precision>300</precision>
    </retention>
    <retention>
      <age>86400</age>
      <precision>3600</precision>
    </retention>
  </default>
</graphite_rollup>

```

## ログエンジン家族

これらのエンジンは、多くの小さなテーブル（最大1万行）をすばやく書き込み、後で全体として読み込む必要があるシナリオ用に開発されました。

家族のエンジン：

- **ストリップログ**
- **ログ**
- **TinyLog**

## 共通プロパティ

エンジン：

- ディスクにデータを格納します。
- 書き込み時にファイルの末尾にデータを追加します。
- 同時データアクセスのサポートロック。

中 **INSERT** クエリのテーブルがロックされ、その他の質問を読み込みおよび書き込みデータの両方のテーブルを作成する データ書き込みクエリがない場合は、任意の数のデータ読み込みクエリを同時に実行できます。

- サポートしない **突然変異** 作戦だ
- 索引をサポートしません。

つまり **SELECT** データ範囲のクエリは効率的ではありません。

- 書くわけではありませんデータを原子的に。

取得できるテーブルデータが破損した場合も破れ、書き込み操作は、例えば、異常サーバをシャットダウンしました。

## 違い

その `TinyLog` エンジンは家族の最も簡単、最も悪い機能性および最も低い効率を提供する。その `TinyLog` エンジンをサポートしていない並列データの読み取りによる複数のスレッド）。でデータを読み込む代わりに、各エンジンの家族を支援する並列読みでの使用がほとんど同じになりました記述子としての `Log` エンジンは、各列を別々のファイルに格納するためです。単純な低負荷のシナリオで使用します。

その `Log` と `StripeLog` エンジンの支援並列データです。読み込み時にデータ `ClickHouse` 使複数のスレッド）。各スレッドプロセス別データブロックです。その `Log` エンジンは、テーブルの各列に個別のファイルを使用します。`StripeLog` すべてのデータファイルです。その結果、`StripeLog` エンジンは、オペレーティングシステム `Log` エンジンはデータを読むとき高性能を提供する。

## ストリップログ

このエンジンはログエンジンの系列に属します。ログエンジンの共通のプロパティとその違いを参照してください [ログエンジン家族](#) 記事だ

少量のデータ(1万行未満)で多数のテーブルを記述する必要がある場合に、このエンジンを使用します。

### テーブルの作成

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    column1_name [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    column2_name [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = StripeLog
```

の詳細な説明を見て下さい [CREATE TABLE](#) クエリ。

### データの書き込み

その `StripeLog` engineはすべての列を一つのファイルに格納します。それそのため `INSERT query`, `ClickHouse` は、データブロックをテーブルファイルの最後に追加し、列を一つずつ書き込みます。

各テーブル `ClickHouse` に書き込み中のファイル:

- `data.bin` — Data file.
- `index.mrk` — File with marks. Marks contain offsets for each column of each data block inserted.

その `StripeLog` エンジンはサポートしません `ALTER UPDATE` と `ALTER DELETE` 作戦だ

### データの読み取り

ファイルをマークで `ClickHouse` を並列化したデータです。これは、`SELECT query` は、予測不可能な順序で行を返します。使用する `ORDER BY` 行をソートする句。

### 使用例

テーブルの作成:

```
CREATE TABLE stripe_log_table
(
    timestamp DateTime,
    message_type String,
    message String
)
ENGINE = StripeLog
```

データの挿入:

```
INSERT INTO stripe_log_table VALUES (now(),'REGULAR','The first regular message')
INSERT INTO stripe_log_table VALUES (now(),'REGULAR','The second regular message'),(now(),'WARNING','The first warning message')
```

我々は二つを使用 `INSERT` データブロックを作成するためのクエリ `data.bin` ファイル

ClickHouse利用は、複数のスレッド選択時のデータです。各スレッドは、個別のデータブロックを読み取り、終了時に結果の行を個別に返します その結果、出力内の行のブロックの順序は、ほとんどの場合、入力内の同じブロックの順序と一致しません。例えば:

```
SELECT * FROM stripe_log_table
```

timestamp	message_type	message
2019-01-18 14:27:32	REGULAR	The second regular message
2019-01-18 14:34:53	WARNING	The first warning message
timestamp	message_type	message
2019-01-18 14:23:43	REGULAR	The first regular message

結果の並べ替え(デフォルトでは昇順):

```
SELECT * FROM stripe_log_table ORDER BY timestamp
```

timestamp	message_type	message
2019-01-18 14:23:43	REGULAR	The first regular message
2019-01-18 14:27:32	REGULAR	The second regular message
2019-01-18 14:34:53	WARNING	The first warning message

## ログ

エンジンはログエンジンの系列に属します。ログエンジンの共通のプロパティとその違いを参照してください [ログエンジン家族](#) 記事だ

ログとは異なります [TinyLog](#) その中の小さなファイルの“marks”列ファイルに存在します。これらのマークはすべてのデータブロックに書き込まれ、指定された行数をスキップするためにファイルの読み取りを開始する場所を示すオフセット この読み取りを可能にする機能がありテーブルデータを複数のスレッド)。

同時データアクセスの場合、読み取り操作は同時に実行できますが、書き込み操作は読み取りをロックします。ログエンジン 同様に、テーブルへの書き込みが失敗した場合、テーブルは壊れ、そこから読み込むとエラーが返されます。ログエンジンは、一時データ、書き込み一度テーブル、およびテストまたはデモの目的に適しています。

## TinyLog

エンジンはログエンジンファミリに属します。見る [ログエンジン家族](#) ログエンジンの共通プロパティとその違い。

このテーブルエンジンは、通常、write-once メソッドで使用されます。たとえば、次のようにします TinyLog-小さなバッチで処理される中間データのテーブルを入力します。多数の小さなテーブルにデータを格納するのは非効率的です。

クエリは単一のストリームで実行されます。言い換えれば、このエンジンは比較的小さなテーブル（最大約1,000,000行）を対象としています。小さなテーブルが多い場合は、このテーブルエンジンを使用するのが理にかなっています。  
ログ エンジン(開く必要の少ないファイル)。

## MongoDB

MongoDB engine is read-only table engine which allows to read data (SELECT queries) from remote MongoDB collection. Engine supports only non-nested data types. INSERT queries are not supported.

### Creating a Table

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name
(
    name1 [type1],
    name2 [type2],
    ...
) ENGINE = MongoDB(host:port, database, collection, user, password [, options]);
```

#### Engine Parameters

- `host:port` — MongoDB server address.
- `database` — Remote database name.
- `collection` — Remote collection name.
- `user` — MongoDB user.
- `password` — User password.
- `options` — MongoDB connection string options (optional parameter).

### Usage Example

Create a table in ClickHouse which allows to read data from MongoDB collection:

```
CREATE TABLE mongo_table
(
    key UInt64,
    data String
) ENGINE = MongoDB('mongo1:27017', 'test', 'simple_table', 'testuser', 'clickhouse');
```

To read from an SSL secured MongoDB server:

```
CREATE TABLE mongo_table_ssl
(
    key UInt64,
    data String
) ENGINE = MongoDB('mongo2:27017', 'test', 'simple_table', 'testuser', 'clickhouse', 'ssl=true');
```

Query:

```
SELECT COUNT() FROM mongo_table;
```

```
count()
  4 |
```

# S3 Table Engine

This engine provides integration with [Amazon S3](#) ecosystem. This engine is similar to the [HDFS](#) engine, but provides S3-specific features.

## Create Table

```
CREATE TABLE s3_engine_table (name String, value UInt32)
ENGINE = S3(path, [aws_access_key_id, aws_secret_access_key], format, [compression])
```

### Engine parameters

- `path` — Bucket url with path to file. Supports following wildcards in readonly mode: `*`, `?`, `{abc,def}` and `{N..M}` where `N, M` — numbers, `'abc'`, `'def'` — strings. For more information see [below](#).
- `format` — The [format](#) of the file.
- `aws_access_key_id`, `aws_secret_access_key` - Long-term credentials for the [AWS](#) account user. You can use these to authenticate your requests. Parameter is optional. If credentials are not specified, they are used from the configuration file. For more information see [Using S3 for Data Storage](#).
- `compression` — Compression type. Supported values: `none`, `gzip/gz`, `brotli/br`, `xz/LZMA`, `zstd/zst`. Parameter is optional. By default, it will autodetect compression by file extension.

### Example

1. Set up the `s3_engine_table` table:

```
CREATE TABLE s3_engine_table (name String, value UInt32) ENGINE=S3('https://storage.yandexcloud.net/my-test-
bucket-768/test-data.csv.gz', 'CSV', 'gzip');
```

2. Fill file:

```
INSERT INTO s3_engine_table VALUES ('one', 1), ('two', 2), ('three', 3);
```

3. Query the data:

```
SELECT * FROM s3_engine_table LIMIT 2;
```

```
name    value
one     1
two     2
```

## Virtual columns

- `_path` — Path to the file.
- `_file` — Name of the file.

For more information about virtual columns see [here](#).

## Implementation Details

- Reads and writes can be parallel
- **Zero-copy** replication is supported.
- Not supported:
  - `ALTER` and `SELECT...SAMPLE` operations.
  - Indexes.

## Wildcards In Path

`path` argument can specify multiple files using bash-like wildcards. For being processed file should exist and match to the whole path pattern. Listing of files is determined during `SELECT` (not at `CREATE` moment).

- `*` — Substitutes any number of any characters except/ including empty string.
- `?` — Substitutes any single character.
- `{some_string,another_string,yet_another_one}` — Substitutes any of strings `'some_string'`, `'another_string'`, `'yet_another_one'`.
- `{N..M}` — Substitutes any number in range from N to M including both borders. N and M can have leading zeroes e.g. `000..078`.

Constructions with `{}` are similar to the **remote** table function.

### Example

1. Suppose we have several files in CSV format with the following URIs on S3:

- '`https://storage.yandexcloud.net/my-test-bucket-768/some_prefix/some_file_1.csv`'
- '`https://storage.yandexcloud.net/my-test-bucket-768/some_prefix/some_file_2.csv`'
- '`https://storage.yandexcloud.net/my-test-bucket-768/some_prefix/some_file_3.csv`'
- '`https://storage.yandexcloud.net/my-test-bucket-768/another_prefix/some_file_1.csv`'
- '`https://storage.yandexcloud.net/my-test-bucket-768/another_prefix/some_file_2.csv`'
- '`https://storage.yandexcloud.net/my-test-bucket-768/another_prefix/some_file_3.csv`'

There are several ways to make a table consisting of all six files:

The first way:

```
CREATE TABLE table_with_range (name String, value UInt32) ENGINE = S3('https://storage.yandexcloud.net/my-test-bucket-768/{some,another}_prefix/some_file_{1..3}', 'CSV');
```

Another way:

```
CREATE TABLE table_with_question_mark (name String, value UInt32) ENGINE =
S3('https://storage.yandexcloud.net/my-test-bucket-768/{some,another}_prefix/some_file_?', 'CSV');
```

Table consists of all the files in both directories (all files should satisfy format and schema described in query):

```
CREATE TABLE table_with_asterisk (name String, value UInt32) ENGINE = S3('https://storage.yandexcloud.net/my-test-bucket-768/{some,another}_prefix/*', 'CSV');
```

If the listing of files contains number ranges with leading zeros, use the construction with braces for each digit separately or use `?`.

## Example

Create table with files named `file-000.csv`, `file-001.csv`, ... , `file-999.csv`:

```
CREATE TABLE big_table (name String, value UInt32) ENGINE = S3('https://storage.yandexcloud.net/my-test-bucket-768/big_prefix/file-{000..999}.csv', 'CSV');
```

## Virtual Columns

- `_path` — Path to the file.
- `_file` — Name of the file.

### See Also

- [Virtual columns](#)

## S3-related settings

The following settings can be set before query execution or placed into configuration file.

- `s3_max_single_part_upload_size` — The maximum size of object to upload using singlepart upload to S3. Default value is `64Mb`.
- `s3_min_upload_part_size` — The minimum size of part to upload during multipart upload to [S3 Multipart upload](#). Default value is `512Mb`.
- `s3_max_redirects` — Max number of S3 redirects hops allowed. Default value is `10`.
- `s3_single_read_retries` — The maximum number of attempts during single read. Default value is `4`.

Security consideration: if malicious user can specify arbitrary S3 URLs, `s3_max_redirects` must be set to zero to avoid [SSRF](#) attacks; or alternatively, `remote_host_filter` must be specified in server configuration.

## Endpoint-based Settings

The following settings can be specified in configuration file for given endpoint (which will be matched by exact prefix of a URL):

- `endpoint` — Specifies prefix of an endpoint. Mandatory.
- `access_key_id` and `secret_access_key` — Specifies credentials to use with given endpoint. Optional.
- `use_environment_credentials` — If set to `true`, S3 client will try to obtain credentials from environment variables and [Amazon EC2](#) metadata for given endpoint. Optional, default value is `false`.
- `region` — Specifies S3 region name. Optional.
- `use_insecure_imds_request` — If set to `true`, S3 client will use insecure IMDS request while obtaining credentials from Amazon EC2 metadata. Optional, default value is `false`.
- `header` — Adds specified HTTP header to a request to given endpoint. Optional, can be specified multiple times.

- `server_side_encryption_customer_key_base64` — If specified, required headers for accessing S3 objects with SSE-C encryption will be set. Optional.
- `max_single_read_retries` — The maximum number of attempts during single read. Default value is 4. Optional.

### Example:

```
<s3>
  <endpoint-name>
    <endpoint>https://storage.yandexcloud.net/my-test-bucket-768/</endpoint>
    <!-- <access_key_id>ACCESS_KEY_ID</access_key_id> -->
    <!-- <secret_access_key>SECRET_ACCESS_KEY</secret_access_key> -->
    <!-- <region>us-west-1</region> -->
    <!-- <use_environment_credentials>false</use_environment_credentials> -->
    <!-- <use_insecure_imds_request>false</use_insecure_imds_request> -->
    <!-- <header>Authorization: Bearer SOME-TOKEN</header> -->
    <!-- <server_side_encryption_customer_key_base64>BASE64-ENCODED-
KEY</server_side_encryption_customer_key_base64> -->
    <!-- <max_single_read_retries>4</max_single_read_retries> -->
  </endpoint-name>
</s3>
```

## Usage

Suppose we have several files in CSV format with the following URIs on S3:

- '[https://storage.yandexcloud.net/my-test-bucket-768/some\\_prefix/some\\_file\\_1.csv](https://storage.yandexcloud.net/my-test-bucket-768/some_prefix/some_file_1.csv)'
- '[https://storage.yandexcloud.net/my-test-bucket-768/some\\_prefix/some\\_file\\_2.csv](https://storage.yandexcloud.net/my-test-bucket-768/some_prefix/some_file_2.csv)'
- '[https://storage.yandexcloud.net/my-test-bucket-768/some\\_prefix/some\\_file\\_3.csv](https://storage.yandexcloud.net/my-test-bucket-768/some_prefix/some_file_3.csv)'
- '[https://storage.yandexcloud.net/my-test-bucket-768/another\\_prefix/some\\_file\\_1.csv](https://storage.yandexcloud.net/my-test-bucket-768/another_prefix/some_file_1.csv)'
- '[https://storage.yandexcloud.net/my-test-bucket-768/another\\_prefix/some\\_file\\_2.csv](https://storage.yandexcloud.net/my-test-bucket-768/another_prefix/some_file_2.csv)'
- '[https://storage.yandexcloud.net/my-test-bucket-768/another\\_prefix/some\\_file\\_3.csv](https://storage.yandexcloud.net/my-test-bucket-768/another_prefix/some_file_3.csv)'

1. There are several ways to make a table consisting of all six files:

```
CREATE TABLE table_with_range (name String, value UInt32)
ENGINE = S3('https://storage.yandexcloud.net/my-test-bucket-768/{some,another}_prefix/some_file_{1..3}', 'CSV');
```

2. Another way:

```
CREATE TABLE table_with_question_mark (name String, value UInt32)
ENGINE = S3('https://storage.yandexcloud.net/my-test-bucket-768/{some,another}_prefix/some_file_?', 'CSV');
```

3. Table consists of all the files in both directories (all files should satisfy format and schema described in query):

```
CREATE TABLE table_with_asterisk (name String, value UInt32)
ENGINE = S3('https://storage.yandexcloud.net/my-test-bucket-768/{some,another}_prefix/*', 'CSV');
```

## Warning

If the listing of files contains number ranges with leading zeros, use the construction with braces for each digit separately or use `?`.

4. Create table with files named `file-000.csv`, `file-001.csv`, ... , `file-999.csv`:

```
CREATE TABLE big_table (name String, value UInt32)
ENGINE = S3('https://storage.yandexcloud.net/my-test-bucket-768/big_prefix/file-{000..999}.csv', 'CSV');
```

## See also

- [s3 table function](#)

## SQLite

The engine allows to import and export data to SQLite and supports queries to SQLite tables directly from ClickHouse.

### Creating a Table

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name
(
    name1 [type1],
    name2 [type2], ...
) ENGINE = SQLite('db_path', 'table')
```

#### Engine Parameters

- `db_path` — Path to SQLite file with a database.
- `table` — Name of a table in the SQLite database.

### Usage Example

Shows a query creating the SQLite table:

```
SHOW CREATE TABLE sqlite_db.table2;
```

```
CREATE TABLE SQLite.table2
(
    `col1` Nullable(Int32),
    `col2` Nullable(String)
)
ENGINE = SQLite('sqlite.db','table2');
```

Returns the data from the table:

```
SELECT * FROM sqlite_db.table2 ORDER BY col1;
```

col1	col2
1	text1
2	text2
3	text3

## See Also

- [SQLite engine](#)
- [sqlite table function](#)

# EmbeddedRocksDB Engine

This engine allows integrating ClickHouse with [rocksdb](#).

## Creating a Table

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = EmbeddedRocksDB PRIMARY KEY(primary_key_name)
```

Required parameters:

- `primary_key_name` – any column name in the column list.
- `primary key` must be specified, it supports only one column in the primary key. The primary key will be serialized in binary as a `rocksdb` key.
- columns other than the primary key will be serialized in binary as `rocksdb` value in corresponding order.
- queries with key `equals` or `in` filtering will be optimized to multi keys lookup from `rocksdb`.

Example:

```
CREATE TABLE test
(
    `key` String,
    `v1` UInt32,
    `v2` String,
    `v3` Float32,
)
ENGINE = EmbeddedRocksDB
PRIMARY KEY key
```

## Metrics

There is also `system.rocksdb` table, that expose rocksdb statistics:

```
SELECT
    name,
    value
FROM system.rocksdb
```

name	value
no.file.opens	1
number.block.decompressed	1

## Configuration

You can also change any [rocksdb options](#) using config:

```

<rocksdb>
  <options>
    <max_background_jobs>8</max_background_jobs>
  </options>
  <column_family_options>
    <num_levels>2</num_levels>
  </column_family_options>
  <tables>
    <table>
      <name>TABLE</name>
      <options>
        <max_background_jobs>8</max_background_jobs>
      </options>
      <column_family_options>
        <num_levels>2</num_levels>
      </column_family_options>
    </table>
  </tables>
</rocksdb>

```

## RabbitMQ Engine

This engine allows integrating ClickHouse with [RabbitMQ](#).

[RabbitMQ](#) lets you:

- Publish or subscribe to data flows.
- Process streams as they become available.

## Creating a Table

```

CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
  name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
  name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
  ...
) ENGINE = RabbitMQ SETTINGS
  rabbitmq_host_port = 'host:port' [or rabbitmq_address = 'amqp(s)::guest:guest@localhost/vhost'],
  rabbitmq_exchange_name = 'exchange_name',
  rabbitmq_format = 'data_format'[,]
  [rabbitmq_exchange_type = 'exchange_type',]
  [rabbitmq_routing_key_list = 'key1,key2,...',]
  [rabbitmq_secure = 0,]
  [rabbitmq_row_delimiter = 'delimiter_symbol',]
  [rabbitmq_schema = ","]
  [rabbitmq_num_consumers = N,]
  [rabbitmq_num_queues = N,]
  [rabbitmq_queue_base = 'queue',]
  [rabbitmq_deadletter_exchange = 'dl-exchange',]
  [rabbitmq_persistent = 0,]
  [rabbitmq_skip_broken_messages = N,]
  [rabbitmq_max_block_size = N,]
  [rabbitmq_flush_interval_ms = N]

```

Required parameters:

- `rabbitmq_host_port` – host:port (for example, `localhost:5672`).
- `rabbitmq_exchange_name` – RabbitMQ exchange name.
- `rabbitmq_format` – Message format. Uses the same notation as the SQL `FORMAT` function, such as `JSONEachRow`. For more information, see the [Formats](#) section.

Optional parameters:

- `rabbitmq_exchange_type` – The type of RabbitMQ exchange: `direct`, `fanout`, `topic`, `headers`, `consistent_hash`. Default: `fanout`.
- `rabbitmq_routing_key_list` – A comma-separated list of routing keys.
- `rabbitmq_row_delimiter` – Delimiter character, which ends the message.
- `rabbitmq_schema` – Parameter that must be used if the format requires a schema definition. For example, `Cap'n Proto` requires the path to the schema file and the name of the `rootschema.capnp:Message` object.
- `rabbitmq_num_consumers` – The number of consumers per table. Default: `1`. Specify more consumers if the throughput of one consumer is insufficient.
- `rabbitmq_num_queues` – Total number of queues. Default: `1`. Increasing this number can significantly improve performance.
- `rabbitmq_queue_base` - Specify a hint for queue names. Use cases of this setting are described below.
- `rabbitmq_deadletter_exchange` - Specify name for a **dead letter exchange**. You can create another table with this exchange name and collect messages in cases when they are republished to dead letter exchange. By default dead letter exchange is not specified.
- `rabbitmq_persistent` - If set to `1` (true), in insert query delivery mode will be set to `2` (marks messages as 'persistent'). Default: `0`.
- `rabbitmq_skip_broken_messages` – RabbitMQ message parser tolerance to schema-incompatible messages per block. Default: `0`. If `rabbitmq_skip_broken_messages = N` then the engine skips `N` RabbitMQ messages that cannot be parsed (a message equals a row of data).
- `rabbitmq_max_block_size`
- `rabbitmq_flush_interval_ms`

SSL connection:

Use either `rabbitmq_secure = 1` or `amqps` in connection address: `rabbitmq_address = 'amqps://guest:guest@localhost/vhost'`.

The default behaviour of the used library is not to check if the created TLS connection is sufficiently secure. Whether the certificate is expired, self-signed, missing or invalid: the connection is simply permitted. More strict checking of certificates can possibly be implemented in the future.

Also format settings can be added along with rabbitmq-related settings.

Example:

```
CREATE TABLE queue (
    key UInt64,
    value UInt64,
    date DateTime
) ENGINE = RabbitMQ SETTINGS rabbitmq_host_port = 'localhost:5672',
    rabbitmq_exchange_name = 'exchange1',
    rabbitmq_format = 'JSONEachRow',
    rabbitmq_num_consumers = 5,
    date_time_input_format = 'best_effort';
```

The RabbitMQ server configuration should be added using the ClickHouse config file.

Required configuration:

```
<rabbitmq>
  <username>root</username>
  <password>clickhouse</password>
</rabbitmq>
```

Additional configuration:

```
<rabbitmq>
  <vhost>clickhouse</vhost>
</rabbitmq>
```

## Description

`SELECT` is not particularly useful for reading messages (except for debugging), because each message can be read only once. It is more practical to create real-time threads using **materialized views**. To do this:

1. Use the engine to create a RabbitMQ consumer and consider it a data stream.
2. Create a table with the desired structure.
3. Create a materialized view that converts data from the engine and puts it into a previously created table.

When the `MATERIALIZED VIEW` joins the engine, it starts collecting data in the background. This allows you to continually receive messages from RabbitMQ and convert them to the required format using `SELECT`. One RabbitMQ table can have as many materialized views as you like.

Data can be channeled based on `rabbitmq_exchange_type` and the specified `rabbitmq_routing_key_list`. There can be no more than one exchange per table. One exchange can be shared between multiple tables - it enables routing into multiple tables at the same time.

Exchange type options:

- `direct` - Routing is based on the exact matching of keys. Example table key list: `key1,key2,key3,key4,key5`, message key can equal any of them.
- `fanout` - Routing to all tables (where exchange name is the same) regardless of the keys.
- `topic` - Routing is based on patterns with dot-separated keys. Examples: `*.logs, records.*.*.2020, *.2018,*.2019,*.2020`.
- `headers` - Routing is based on `key=value` matches with a setting `x-match=all` or `x-match=any`. Example table key list: `x-match=all,format=logs,type=report,year=2020`.
- `consistent_hash` - Data is evenly distributed between all bound tables (where the exchange name is the same). Note that this exchange type must be enabled with RabbitMQ plugin: `rabbitmq-plugins enable rabbitmq_consistent_hash_exchange`.

Setting `rabbitmq_queue_base` may be used for the following cases:

- to let different tables share queues, so that multiple consumers could be registered for the same queues, which makes a better performance. If using `rabbitmq_num_consumers` and/or `rabbitmq_num_queues` settings, the exact match of queues is achieved in case these parameters are the same.
- to be able to restore reading from certain durable queues when not all messages were successfully consumed. To resume consumption from one specific queue - set its name in `rabbitmq_queue_base` setting and do not specify `rabbitmq_num_consumers` and `rabbitmq_num_queues` (defaults to 1). To resume consumption from all queues, which were declared for a specific table - just specify the same settings: `rabbitmq_queue_base`, `rabbitmq_num_consumers`, `rabbitmq_num_queues`. By default, queue names will be unique to tables.

- to reuse queues as they are declared durable and not auto-deleted. (Can be deleted via any of RabbitMQ CLI tools.)

To improve performance, received messages are grouped into blocks the size of `max_insert_block_size`. If the block wasn't formed within `stream_flush_interval_ms` milliseconds, the data will be flushed to the table regardless of the completeness of the block.

If `rabbitmq_num_consumers` and/or `rabbitmq_num_queues` settings are specified along with `rabbitmq_exchange_type`, then:

- `rabbitmq-consistent-hash-exchange` plugin must be enabled.
- `message_id` property of the published messages must be specified (unique for each message/batch).

For insert query there is message metadata, which is added for each published message: `messageID` and republished flag (true, if published more than once) - can be accessed via message headers.

Do not use the same table for inserts and materialized views.

Example:

```
CREATE TABLE queue (
    key UInt64,
    value UInt64
) ENGINE = RabbitMQ SETTINGS rabbitmq_host_port = 'localhost:5672',
    rabbitmq_exchange_name = 'exchange1',
    rabbitmq_exchange_type = 'headers',
    rabbitmq_routing_key_list = 'format=logs,type=report,year=2020',
    rabbitmq_format = 'JSONEachRow',
    rabbitmq_num_consumers = 5;

CREATE TABLE daily (key UInt64, value UInt64)
ENGINE = MergeTree() ORDER BY key;

CREATE MATERIALIZED VIEW consumer TO daily
AS SELECT key, value FROM queue;

SELECT key, value FROM daily ORDER BY key;
```

## Virtual Columns

- `_exchange_name` - RabbitMQ exchange name.
- `_channel_id` - ChannelID, on which consumer, who received the message, was declared.
- `_delivery_tag` - DeliveryTag of the received message. Scoped per channel.
- `_redelivered` - redelivered flag of the message.
- `_message_id` - `messageID` of the received message; non-empty if was set, when message was published.
- `_timestamp` - timestamp of the received message; non-empty if was set, when message was published.

## PostgreSQL

The PostgreSQL engine allows to perform `SELECT` and `INSERT` queries on data that is stored on a remote PostgreSQL server.

## Creating a Table

```

CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1] [TTL expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2] [TTL expr2],
    ...
) ENGINE = PostgreSQL('host:port', 'database', 'table', 'user', 'password'[, `schema`]);

```

See a detailed description of the [CREATE TABLE](#) query.

The table structure can differ from the original PostgreSQL table structure:

- Column names should be the same as in the original PostgreSQL table, but you can use just some of these columns and in any order.
- Column types may differ from those in the original PostgreSQL table. ClickHouse tries to [cast](#) values to the ClickHouse data types.
- The [external\\_table\\_functions\\_use\\_nulls](#) setting defines how to handle Nullable columns. Default value: 1. If 0, the table function does not make Nullable columns and inserts default values instead of nulls. This is also applicable for NULL values inside arrays.

## Engine Parameters

- `host:port` — PostgreSQL server address.
- `database` — Remote database name.
- `table` — Remote table name.
- `user` — PostgreSQL user.
- `password` — User password.
- `schema` — Non-default table schema. Optional.
- `on conflict ...` — example: `ON CONFLICT DO NOTHING`. Optional. Note: adding this option will make insertion less efficient.

## Implementation Details

`SELECT` queries on PostgreSQL side run as `COPY (SELECT ...)` TO STDOUT inside read-only PostgreSQL transaction with commit after each `SELECT` query.

Simple `WHERE` clauses such as `=`, `!=`, `>`, `>=`, `<`, `<=`, and `IN` are executed on the PostgreSQL server.

All joins, aggregations, sorting, `IN [ array ]` conditions and the `LIMIT` sampling constraint are executed in ClickHouse only after the query to PostgreSQL finishes.

`INSERT` queries on PostgreSQL side run as `COPY "table_name" (field1, field2, ... fieldN)` FROM STDIN inside PostgreSQL transaction with auto-commit after each `INSERT` statement.

PostgreSQL Array types are converted into ClickHouse arrays.

## Note

Be careful - in PostgreSQL an array data, created like a `type_name[]`, may contain multi-dimensional arrays of different dimensions in different table rows in same column. But in ClickHouse it is only allowed to have multidimensional arrays of the same count of dimensions in all table rows in same column.

Supports multiple replicas that must be listed by ]. For example:

```
CREATE TABLE test_replicas (id UInt32, name String) ENGINE = PostgreSQL(`postgres{2|3|4}:5432`, 'clickhouse', 'test_replicas', 'postgres', 'mysecretpassword');
```

Replicas priority for PostgreSQL dictionary source is supported. The bigger the number in map, the less the priority. The highest priority is 0.

In the example below replica `example01-1` has the highest priority:

```
<postgresql>
  <port>5432</port>
  <user>clickhouse</user>
  <password>qwerty</password>
  <replica>
    <host>example01-1</host>
    <priority>1</priority>
  </replica>
  <replica>
    <host>example01-2</host>
    <priority>2</priority>
  </replica>
  <db>db_name</db>
  <table>table_name</table>
  <where>id=10</where>
  <invalidate_query>SQL_QUERY</invalidate_query>
</postgresql>
</source>
```

## Usage Example

Table in PostgreSQL:

```
postgres=# CREATE TABLE "public"."test" (
  "int_id" SERIAL,
  "int_nullable" INT NULL DEFAULT NULL,
  "float" FLOAT NOT NULL,
  "str" VARCHAR(100) NOT NULL DEFAULT '',
  "float_nullable" FLOAT NULL DEFAULT NULL,
  PRIMARY KEY (int_id));

CREATE TABLE

postgres=# INSERT INTO test (int_id, str, "float") VALUES (1,'test',2);
INSERT 0 1

postgres@:~$ SELECT * FROM test;
 int_id | int_nullable | float | str  | float_nullable
-----+-----+-----+-----+
  1   |      |    2 | test |
(1 row)
```

Table in ClickHouse, retrieving data from the PostgreSQL table created above:

```
CREATE TABLE default.postgresql_table
(
  `float_nullable` Nullable(Float32),
  `str` String,
  `int_id` Int32
)
ENGINE = PostgreSQL('localhost:5432', 'public', 'test', 'postges_user', 'postres_password');
```

```
SELECT * FROM postgresql_table WHERE str IN ('test');
```

float_nullable	str	int_id
NULL	test	1

Using Non-default Schema:

```
postgres=# CREATE SCHEMA "nice.schema";
postgres=# CREATE TABLE "nice.schema"."nice.table" (a integer);
postgres=# INSERT INTO "nice.schema"."nice.table" SELECT i FROM generate_series(0, 99) as t(i)
```

```
CREATE TABLE pg_table_schema_with_dots (a UInt32)
    ENGINE PostgreSQL('localhost:5432', 'clickhouse', 'nice.table', 'postgrsql_user', 'password', 'nice.schema');
```

## See Also

- [The postgresql table function](#)
- [Using PostgreSQL as a source of external dictionary](#)

# ExternalDistributed

The `ExternalDistributed` engine allows to perform `SELECT` queries on data that is stored on a remote servers MySQL or PostgreSQL. Accepts [MySQL](#) or [PostgreSQL](#) engines as an argument so sharding is possible.

## Creating a Table

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1] [TTL expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2] [TTL expr2],
    ...
) ENGINE = ExternalDistributed('engine', 'host:port', 'database', 'table', 'user', 'password');
```

See a detailed description of the [CREATE TABLE](#) query.

The table structure can differ from the original table structure:

- Column names should be the same as in the original table, but you can use just some of these columns and in any order.
- Column types may differ from those in the original table. ClickHouse tries to [cast](#) values to the ClickHouse data types.

## Engine Parameters

- `engine` — The table engine [MySQL](#) or [PostgreSQL](#).
- `host:port` — MySQL or PostgreSQL server address.
- `database` — Remote database name.
- `table` — Remote table name.
- `user` — User name.
- `password` — User password.

# Implementation Details

Supports multiple replicas that must be listed by | and shards must be listed by . For example:

```
CREATE TABLE test_shards (id UInt32, name String, age UInt32, money UInt32) ENGINE = ExternalDistributed('MySQL', `mysql{1|2}:3306,mysql{3|4}:3306`, 'clickhouse', 'test_replicas', 'root', 'clickhouse');
```

When specifying replicas, one of the available replicas is selected for each of the shards when reading. If the connection fails, the next replica is selected, and so on for all the replicas. If the connection attempt fails for all the replicas, the attempt is repeated the same way several times.

You can specify any number of shards and any number of replicas for each shard.

## See Also

- [MySQL table engine](#)
- [PostgreSQL table engine](#)
- [Distributed table engine](#)

# MaterializedPostgreSQL

Creates ClickHouse table with an initial data dump of PostgreSQL table and starts replication process, i.e. executes background job to apply new changes as they happen on PostgreSQL table in the remote PostgreSQL database.

If more than one table is required, it is highly recommended to use the [MaterializedPostgreSQL](#) database engine instead of the table engine and use the [materialized\\_postgresql\\_tables\\_list](#) setting, which specifies the tables to be replicated. It will be much better in terms of CPU, fewer connections and fewer replication slots inside the remote PostgreSQL database.

## Creating a Table

```
CREATE TABLE postgresql_db.postgresql_replica (key UInt64, value UInt64)
ENGINE = MaterializedPostgreSQL('postgres1:5432', 'postgres_database', 'postgresql_replica', 'postgres_user',
'postgres_password')
PRIMARY KEY key;
```

## Engine Parameters

- `host:port` — PostgreSQL server address.
- `database` — Remote database name.
- `table` — Remote table name.
- `user` — PostgreSQL user.
- `password` — User password.

## Requirements

1. The `wal_level` setting must have a value `logical` and `max_replication_slots` parameter must have a value at least 2 in the PostgreSQL config file.
2. A table with `MaterializedPostgreSQL` engine must have a primary key — the same as a replica identity index (by default: primary key) of a PostgreSQL table (see [details on replica identity index](#)).

3. Only database **Atomic** is allowed.

## Virtual columns

- `_version` — Transaction counter. Type: **UInt64**.
- `_sign` — Deletion mark. Type: **Int8**. Possible values:
  - **1** — Row is not deleted,
  - **-1** — Row is deleted.

These columns do not need to be added when a table is created. They are always accessible in SELECT query.

`_version` column equals LSN position in WAL, so it might be used to check how up-to-date replication is.

```
CREATE TABLE postgresql_db.postgresql_replica (key UInt64, value UInt64)
ENGINE = MaterializedPostgreSQL('postgres1:5432', 'postgres_database', 'postgresql_replica', 'postgres_user',
'postgres_password')
PRIMARY KEY key;

SELECT key, value, _version FROM postgresql_db.postgresql_replica;
```

## Warning

Replication of **TOAST** values is not supported. The default value for the data type will be used.

## カフカ

このエンジンは **アバッチ-カフカ**.

カフカはあなたをできます:

- データフローを公開または購読する。
- 整理-フォールトトレラント保管します。
- ストリームが使用可能になったら処理します。

## テーブルの作成

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1][DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2][DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = Kafka()
SETTINGS
    kafka_broker_list = 'host:port',
    kafka_topic_list = 'topic1,topic2,...',
    kafka_group_name = 'group_name',
    kafka_format = 'data_format'[,]
    [kafka_row_delimiter = 'delimiter_symbol',]
    [kafka_schema = ","]
    [kafka_num_consumers = N,]
    [kafka_max_block_size = 0,]
    [kafka_skip_broken_messages = N,]
    [kafka_commit_every_batch = 0]
```

必須パラメータ:

- `kafka_broker_list` – A comma-separated list of brokers (for example, `localhost:9092`).
- `kafka_topic_list` – A list of Kafka topics.
- `kafka_group_name` – A group of Kafka consumers. Reading margins are tracked for each group separately. If you don't want messages to be duplicated in the cluster, use the same group name everywhere.
- `kafka_format` – Message format. Uses the same notation as the SQL `FORMAT` 機能、のような `JSONEachRow`. 詳細については、を参照してください 形式 セクション

任意変数:

- `kafka_row_delimiter` – Delimiter character, which ends the message.
- `kafka_schema` – Parameter that must be used if the format requires a schema definition. For example, `Cap'N Proto` スキーマファイルへのパスとルートの名前が必要です `schema.capnp:Message` オブジェクト
- `kafka_num_consumers` – The number of consumers per table. Default: 1. 指定しこれからも、多くの消費者の場合、スループットの消費が不足しています。の総数消費者を超えることはできませんパーティションの数の問題から一つだけの消費者割り当てることができた。
- `kafka_max_block_size` - ポーリングの最大バッチサイズ(メッセージ)(デフォルト: `max_block_size`).
- `kafka_skip_broken_messages` – Kafka message parser tolerance to schema-incompatible messages per block. Default: 0. もし `kafka_skip_broken_messages = N` その後、エンジンがスキップ `N` 解析できないKafka メッセージ(メッセージはデータの行に等しい)。
- `kafka_commit_every_batch` コミット毎に消費され、取り扱うバッチの代わりに単一のコミットし、全体をロック(デフォルト: 0).

例:

```
CREATE TABLE queue (
    timestamp UInt64,
    level String,
    message String
) ENGINE = Kafka('localhost:9092', 'topic', 'group1', 'JSONEachRow');

SELECT * FROM queue LIMIT 5;

CREATE TABLE queue2 (
    timestamp UInt64,
    level String,
    message String
) ENGINE = Kafka SETTINGS kafka_broker_list = 'localhost:9092',
    kafka_topic_list = 'topic',
    kafka_group_name = 'group1',
    kafka_format = 'JSONEachRow',
    kafka_num_consumers = 4;

CREATE TABLE queue2 (
    timestamp UInt64,
    level String,
    message String
) ENGINE = Kafka('localhost:9092', 'topic', 'group1')
    SETTINGS kafka_format = 'JSONEachRow',
    kafka_num_consumers = 4;
```

▶ 推奨されていません法テーブルを作成する

## 説明

届いたメッセージは自動的に追跡で、それぞれのメッセージグループでは数えます。データを二度取得したい場合は、別のグループ名を持つテーブルのコピーを作成します。

グループは柔軟で、クラスター上で同期されます。例えば10テーマの5冊のテーブルにクラスターでは、そのコピーを取得します。コピーが変更されると、トピックは自動的にコピー間で再配布されます。もっと読むことで <http://kafka.apache.org/intro>.

`SELECT` は特に役立つメッセージを読む(以外のデバッグ)では、それぞれのメッセージでしか読み込むことができます。ではより実践的な創出の実時間スレッドを実現します。これを行うには:

1. エンジンを使用してKafkaコンシューマーを作成し、データストリームとみなします。
2. 目的の構造を持つテーブルを作成します。
3. エンジンからデータを変換し、以前に作成したテーブルに格納するマテリアライズドビューを作成します。

ときに `MATERIALIZED VIEW` 入、エンジンでデータを収集しあげます。これにより、kafkaからのメッセージを継続的に受信し、必要な形式に変換することができます `SELECT`.

一つのカフカテーブルは、あなたが好きなだけ多くのマテリアライズドビューを持つことができ、彼らは直接カフカテーブルからデータを読み取るのではな

例:

```
CREATE TABLE queue (
    timestamp UInt64,
    level String,
    message String
) ENGINE = Kafka('localhost:9092', 'topic', 'group1', 'JSONEachRow');

CREATE TABLE daily (
    day Date,
    level String,
    total UInt64
) ENGINE = SummingMergeTree(day, (day, level), 8192);

CREATE MATERIALIZED VIEW consumer TO daily
    AS SELECT toDate(toDateTime(timestamp)) AS day, level, count() as total
    FROM queue GROUP BY day, level;

SELECT level, sum(total) FROM daily GROUP BY level;
```

パフォーマンスを向上させるために、受信したメッセージは `max_insert_block_size`. ブロックが内に形成されていない場合 `stream_flush_interval_ms` ミリ秒、データは関係なく、ブロックの完全性のテーブルにフラッシュされます。

リクエストを受けた話題のデータは変更に変換ロジック、切り離しを実現ビュー:

```
DETACH TABLE consumer;
ATTACH TABLE consumer;
```

を使用してターゲットテーブルを変更する場合 `ALTER` マテリアルビューを無効にすると、ターゲットテーブルとビューのデータとの間の不一致を回避できます。

## 設定

`GraphiteMergeTree`と同様に、KafkaエンジンはClickHouse設定ファイルを使用した拡張構成をサポートします。使用できる設定キーは次の二つです。`(kafka)`とトピックレベル (`kafka_*`). 最初にグローバル構成が適用され、次にトピックレベルの構成が適用されます(存在する場合)。

```

<!-- Global configuration options for all tables of Kafka engine type -->
<kafka>
  <debug>cgrp</debug>
  <auto_offset_reset>smallest</auto_offset_reset>
</kafka>

<!-- Configuration specific for topic "logs" -->
<kafka_logs>
  <retry_backoff_ms>250</retry_backoff_ms>
  <fetch_min_bytes>100000</fetch_min_bytes>
</kafka_logs>

```

可能な構成オプションのリストについては、[librdkafka設定リファレンス](#)。アンダースコアを使用する（\_）ClickHouse 設定のドットの代わりに。例えば, `check.crcs=true` になります `<check_crcs>true</check_crcs>`.

## 仮想列

- `_topic` — Kafka topic.
  - `_key` — Key of the message.
  - `_offset` — Offset of the message.
  - `_timestamp` — Timestamp of the message.
  - `_partition` — Partition of Kafka topic.
- も参照。
- [仮想列](#)

## Mysql

MySQLエンジンでは、次の操作を実行できます `SELECT` リモート MySQLサーバーに格納されているデータに対するクエリ。

### テーブルの作成

```

CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
  name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1] [TTL expr1],
  name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2] [TTL expr2],
  ...
) ENGINE = MySQL('host:port', 'database', 'table', 'user', 'password'[, replace_query, 'on_duplicate_clause']);

```

の詳細な説明を参照してください [CREATE TABLE](#) クエリ。

テーブル構造は、元のMySQLテーブル構造と異なる場合があります:

- カラム名は元のMySQLテーブルと同じでなければなりませんが、これらのカラムの一部だけを任意の順序で使用できます。
- 列の型は、元のMySQLテーブルの型とは異なる場合があります。ClickHouseは [キャスト](#) ClickHouseデータ型の値。

### エンジン変数

- `host:port` — MySQL server address.
- `database` — Remote database name.

- `table` — Remote table name.
- `user` — MySQL user.
- `password` — User password.
- `replace_query` — Flag that converts `INSERT INTO`へのクエリ `REPLACE INTO`. もし `replace_query=1`、クエリが代入されます。
- `on_duplicate_clause` — The `ON DUPLICATE KEY on_duplicate_clause`に追加される式 `INSERT` クエリ。

例: `INSERT INTO t (c1,c2) VALUES ('a', 2) ON DUPLICATE KEY UPDATE c2 = c2 + 1`ここで `on_duplicate_clause` は `UPDATE c2 = c2 + 1.` を参照。 MySQLドキュメント これを見つけるには `on_duplicate_clause` と使用できます `ON DUPLICATE KEY` 句。

指定するには `on_duplicate_clause` 合格する必要があります 0 に `replace_query` パラメータ。あなたが同時に合格した場合 `replace_query = 1` と `on_duplicate_clause`, ClickHouseは例外を生成します。

シンプル `WHERE` 次のような句 `=, !=, >, >=, <, <=` MySQLサーバー上で実行されます。

残りの条件と `LIMIT` サンプリング制約は、MySQLへのクエリが終了した後にのみClickHouseで実行されます。

## 使用例

MySQLのテーブル:

```
mysql> CREATE TABLE `test`.`test` (
->   `int_id` INT NOT NULL AUTO_INCREMENT,
->   `int_nullable` INT NULL DEFAULT NULL,
->   `float` FLOAT NOT NULL,
->   `float_nullable` FLOAT NULL DEFAULT NULL,
->   PRIMARY KEY (`int_id`));
Query OK, 0 rows affected (0,09 sec)

mysql> insert into test (`int_id`, `float`) VALUES (1,2);
Query OK, 1 row affected (0,00 sec)

mysql> select * from test;
+-----+-----+-----+
| int_id | int_nullable | float | float_nullable |
+-----+-----+-----+
|    1 |      NULL |    2 |        NULL |
+-----+-----+-----+
1 row in set (0,00 sec)
```

ClickHouseのテーブル、上で作成したMySQLテーブルからデータを取得する:

```
CREATE TABLE mysql_table
(
  `float_nullable` Nullable(Float32),
  `int_id` Int32
)
ENGINE = MySQL('localhost:3306', 'test', 'test', 'bayonet', '123')
```

```
SELECT * FROM mysql_table
```

float_nullable	int_id
NULL	1

も参照。

- その ‘mysql’ テーブル関数
- 外部辞書のソースとしてMySQLを使用する

## JDBC

ClickHouseが外部データベースに接続できるようにする [JDBC](#).

JDBC接続を実装するには、ClickHouseは別のプログラムを使用します [clickhouse-jdbc-bridge](#) うにしてくれました。

このエンジンは Null 可能 データ型。

### テーブルの作成

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name
(
    columns list...
)
ENGINE = JDBC(datasource_uri, external_database, external_table)
```

#### エンジン変数

- `datasource_uri` — URI or name of an external DBMS.

URI形式: `jdbc:<driver_name>://<host_name>:<port>/?user=<username>&password=<password>`.

MySQLの例: `jdbc:mysql://localhost:3306/?user=root&password=root`.

- `external_database` — Database in an external DBMS.
- `external_table` — Name of the table in `external_database` or a select query like `select * from table1 where column1=1`.

### 使用例

コンソールクライアントに直接接続してMySQL serverでテーブルを作成する:

```
mysql> CREATE TABLE `test`.`test` (
->   `int_id` INT NOT NULL AUTO_INCREMENT,
->   `int_nullable` INT NULL DEFAULT NULL,
->   `float` FLOAT NOT NULL,
->   `float_nullable` FLOAT NULL DEFAULT NULL,
->   PRIMARY KEY (`int_id`));
Query OK, 0 rows affected (0,09 sec)

mysql> insert into test (`int_id`, `float`) VALUES (1,2);
Query OK, 1 row affected (0,00 sec)

mysql> select * from test;
+-----+-----+-----+
| int_id | int_nullable | float | float_nullable |
+-----+-----+-----+
|    1 |      NULL |    2 |        NULL |
+-----+-----+-----+
1 row in set (0,00 sec)
```

ClickHouse serverでテーブルを作成し、そこからデータを選択する:

```

CREATE TABLE jdbc_table
(
    `int_id` Int32,
    `int_nullable` Nullable(Int32),
    `float` Float32,
    `float_nullable` Nullable(Float32)
)
ENGINE JDBC('jdbc:mysql://localhost:3306/?user=root&password=root', 'test', 'test')

```

```

SELECT *
FROM jdbc_table

```

int_id	int_nullable	float	float_nullable
1	NULL	2	NULL

```

INSERT INTO jdbc_table(`int_id`, `float`)
SELECT toInt32(number), toFloat32(number * 1.0)
FROM system.numbers

```

も参照。

- JDBCテーブル関数.

## ODBC

ClickHouseが外部データベースに接続できるようにする **ODBC**.

ODBC接続を安全に実装するために、ClickHouseは別のプログラムを使用します `clickhouse-odbc-bridge`. ODBCドライバーが直接ロードされる場合 `clickhouse-server` ドライバの問題でクラッシュのClickHouseサーバーです。ClickHouseは自動的に起動します `clickhouse-odbc-bridge` それが必要なとき。ODBC bridgeプログラムは、`clickhouse-server`.

このエンジンは **Null可能** データ型。

### テーブルの作成

```

CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1],
    name2 [type2],
    ...
)
ENGINE = ODBC(connection_settings, external_database, external_table)

```

の詳細な説明を参照してください **CREATE TABLE** クエリ。

表構造は、ソース表構造とは異なる場合があります:

- 列名はソーステーブルと同じにする必要がありますが、これらの列の一部だけを任意の順序で使用できます。
- 列の型は、ソーステーブルの型と異なる場合があります。ClickHouseは **キャスト** ClickHouseデータ型の値。

#### エンジン変数

- `connection_settings` — Name of the section with connection settings in the `odbc.ini` ファイル
- `external_database` — Name of a database in an external DBMS.

- `external_table` — Name of a table in the `external_database`.

## 使用例

取得データから地元のMySQLのインストール目盛

この例では、Ubuntu Linux18.04およびMySQL server5.7がチェックされています。

UnixODBCとMySQL Connectorがインストールされていることを確認します。

デフォルトでインストールされた場合、パッケージから),ClickHouse開始してユーザー `clickhouse`. したがって、MySQLサーバーでこのユーザーを作成して構成する必要があります。

```
$ sudo mysql
```

```
mysql> CREATE USER 'clickhouse'@'localhost' IDENTIFIED BY 'clickhouse';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'clickhouse'@'clickhouse' WITH GRANT OPTION;
```

次に、接続を設定します /etc/odbc.ini.

```
$ cat /etc/odbc.ini
[mysqlconn]
DRIVER = /usr/local/lib/libmyodbc5w.so
SERVER = 127.0.0.1
PORT = 3306
DATABASE = test
USERNAME = clickhouse
PASSWORD = clickhouse
```

接続を確認するには `isql` unixODBCの取付けからの実用性。

```
$ isql -v mysqlconn
+-----+
| Connected!
|                               |
...
```

MySQLのテーブル:

```
mysql> CREATE TABLE `test`.`test` (
    -> `int_id` INT NOT NULL AUTO_INCREMENT,
    -> `int_nullable` INT NULL DEFAULT NULL,
    -> `float` FLOAT NOT NULL,
    -> `float_nullable` FLOAT NULL DEFAULT NULL,
    -> PRIMARY KEY (`int_id`));
Query OK, 0 rows affected (0,09 sec)
```

```
mysql> insert into test(`int_id`, `float`) VALUES (1,2);
Query OK, 1 row affected (0,00 sec)
```

```
mysql> select * from test;
+-----+-----+-----+
| int_id | int_nullable | float | float_nullable |
+-----+-----+-----+
|     1 |        NULL |     2 |        NULL |
+-----+-----+-----+
1 row in set (0,00 sec)
```

ClickHouseのテーブル、MySQLテーブルからデータを取得する:

```
CREATE TABLE odbc_t
(
    `int_id` Int32,
    `float_nullable` Nullable(Float32)
)
ENGINE = ODBC('DSN=mysqlconn', 'test', 'test')
```

```
SELECT * FROM odbc_t
```

int_id	float_nullable
1	NULL

も参照。

- [ODBC外部辞書](#)
- [ODBCテーブル関数](#)

## HDFS

このエンジンは、[Apache Hadoop](#) 生態系および管理データ [HDFS](#) クリックハウス経由。 このエンジンは同様ですに [ファイル](#) と [URL](#) エンジンが、Hadoop固有の機能を提供します。

### 使用法

```
ENGINE = HDFS(URI, format)
```

その [URI](#) パラメータは、HDFS内のファイルURI全体です。

その [format](#) パラメータを指定するか、ファイルのファイルフォーマット 実行するには

[SELECT](#) この形式は、入力と実行のためにサポートされている必要があります

[INSERT queries – for output. The available formats are listed in the](#)

[形式 セクション](#)

のパス部分 [URI globs](#)を含むことができます。 この場合、テーブルは[readonly](#)になります。

例:

1. セットアップ [hdfs\\_engine\\_table](#) テーブル:

```
CREATE TABLE hdfs_engine_table (name String, value UInt32) ENGINE=HDFS('hdfs://hdfs1:9000/other_storage', 'TSV')
```

2. Fill ファイル:

```
INSERT INTO hdfs_engine_table VALUES ('one', 1), ('two', 2), ('three', 3)
```

3. データの照会:

```
SELECT * FROM hdfs_engine_table LIMIT 2
```

name	value
one	1
two	2

## 実装の詳細

- 読み書きできる並列
- 対応していません:
  - ALTER と SELECT...SAMPLE 作戦だ
  - インデックス。
  - 複製。

### パス内のグローブ

複数のパスコンポーネン のための処理中のファイルが存在するマッチのパスのパターンです。 ファイルのリストは `SELECT` ( ではない `CREATE` 瞬間 ) 。

- `*` — Substitutes any number of any characters except/ 空の文字列を含む。
- `?` — Substitutes any single character.
- `{some_string,another_string,yet_another_one}` — Substitutes any of strings 'some\_string', 'another\_string', 'yet\_another\_one'.
- `{N..M}` — Substitutes any number in range from N to M including both borders.

構造との `{}` に類似しています [リモート](#) テーブル関数。

### 例

1. HDFS上に次のUriを持つTSV形式のファイルがいくつかあるとします:

- 'hdfs://hdfs1:9000/some\_dir/some\_file\_1'
- 'hdfs://hdfs1:9000/some\_dir/some\_file\_2'
- 'hdfs://hdfs1:9000/some\_dir/some\_file\_3'
- 'hdfs://hdfs1:9000/another\_dir/some\_file\_1'
- 'hdfs://hdfs1:9000/another\_dir/some\_file\_2'
- 'hdfs://hdfs1:9000/another\_dir/some\_file\_3'

1. あはいくつかの方法が考えられているテーブルの構成は、すべてのファイル:

```
CREATE TABLE table_with_range (name String, value UInt32) ENGINE =
HDFS('hdfs://hdfs1:9000/{some,another}_dir/some_file_{1..3}', 'TSV')
```

別 の 方法:

```
CREATE TABLE table_with_question_mark (name String, value UInt32) ENGINE =
HDFS('hdfs://hdfs1:9000/{some,another}_dir/some_file_?', 'TSV')
```

テーブルはすべてのファイルの両方のディレクトリ(すべてのファイルが満たすべき書式は、スキーマに記載のクエリ):

```
CREATE TABLE table_with_asterisk (name String, value UInt32) ENGINE =  
HDFS('hdfs://hdfs1:9000/{some,another}_dir/*', 'TSV')
```

## 警告

ファイル ?.

## 例

このように作成されたテーブルとファイル名 file000, file001, ..., file999:

```
CREATE TABLE big_table (name String, value UInt32) ENGINE = HDFS('hdfs://hdfs1:9000/big_dir/file{0..9}{0..9}  
{0..9}', 'CSV')
```

## 仮想列

- `_path` — Path to the file.
- `_file` — Name of the file.  
も参照。
- **仮想列**

## 分散

分散エンジンを備えたテーブルは、データ自体を格納しません しかし、複数のサーバーで分散クエリ処理を許可します。

読み取りは自動的に並列化されます。読み取り中に、リモートサーバー上のテーブルインデックスが存在する場合に使用されます。

分散エンジ:

- サーバーの設定ファイル内のクラスタ名
- リモートデータベースの名前
- リモートテーブルの名前
- (オプション)shardingキー
- (必要に応じて)ポリシーネームは、非同期送信の一時ファイルを格納するために使用されます

も参照。:

- `insert_distributed_sync` 設定
- **メルゲツリー** 例については

例:

```
Distributed(logs, default, hits[, sharding_key[, policy_name]])
```

すべてのサーバからデータが読み取られます。'logs' デフォルトからのクラスター。ヒットテーブルに位置毎にサーバのクラスター

データは読み取りだけでなく、リモートサーバーで部分的に処理されます（可能な限り）。

たとえば、GROUP BYを使用したクエリの場合、リモートサーバーでデータが集計され、集計関数の中間状態が要求元サーバーに送信されます。その後、データはさらに集計されます。

データベース名の代わりに、文字列を返す定数式を使用できます。たとえば、currentDatabase()です。

logs – The cluster name in the server's config file.

クラスターがセットのようななこ：

```
<remote_servers>
  <logs>
    <shard>
      <!-- Optional. Shard weight when writing data. Default: 1. -->
      <weight>1</weight>
      <!-- Optional. Whether to write data to just one of the replicas. Default: false (write data to all replicas). -->
      <internal_replication>false</internal_replication>
      <replica>
        <host>example01-01-1</host>
        <port>9000</port>
      </replica>
      <replica>
        <host>example01-01-2</host>
        <port>9000</port>
      </replica>
    </shard>
    <shard>
      <weight>2</weight>
      <internal_replication>false</internal_replication>
      <replica>
        <host>example01-02-1</host>
        <port>9000</port>
      </replica>
      <replica>
        <host>example01-02-2</host>
        <secure>1</secure>
        <port>9440</port>
      </replica>
    </shard>
  </logs>
</remote_servers>
```

ここで、クラスターは名前で定義されます 'logs' これは二つの破片で構成され、それぞれに二つの複製が含まれています。

シャードは、データの異なる部分を含むサーバーを指します(すべてのデータを読み取るには、すべてのシャードにアクセスする必要があります)。

レプリカはサーバーを複製しています (すべてのデータを読み取るために、レプリカのいずれかのデータにアクセスできます)。

クラスターの名前などを含めないでくださいませんでした。

パラメータ `host`, `port`、および必要に応じて `user`, `password`, `secure`, `compression` サーバーごとに指定されます:

- `host` – The address of the remote server. You can use either the domain or the IPv4 or IPv6 address. If you specify the domain, the server makes a DNS request when it starts, and the result is stored as long as the server is running. If the DNS request fails, the server doesn't start. If you change the DNS record, restart the server.

- `port` – The TCP port for messenger activity ('`tcp_port`' 設定では、通常9000に設定されています)。 `Http_port` と混同しないでください。

- `user` – Name of the user for connecting to a remote server. Default value: `default`. This user must have access to connect to the specified server. Access is configured in the `users.xml` file. For more information, see the section [アクセス権](#).

- `password` – The password for connecting to a remote server (not masked). Default value: empty string.

- `secure` -接続にsslを使用します。 `port = 9440`. サーバーがリッスンする `<tcp_port_secure>9440</tcp_port_secure>` 正しい証明書を持ってています。

- `compression` -データ圧縮を使用します。 デフォルト値:true。

When specifying replicas, one of the available replicas will be selected for each of the shards when reading. You can configure the algorithm for load balancing (the preference for which replica to access) – see the [load\\_balancing](#) 設定。

サーバーとの接続が確立されていない場合は、短いタイムアウトで接続しようとします。 接続に失敗した場合は、すべてのレプリカに対して次のレプリカが選択されます。 すべてのレプリカで接続試行が失敗した場合、その試行は同じ方法で何度か繰り返されます。

リモートサーバーは接続を受け入れるかもしれません、動作しないか、または不十分に動作する可能性があります。

いずれかのシャードのみを指定することができます(この場合、クエリ処理は分散ではなくリモートと呼ばれる必要があります)。 各シャードでは、いずれかから任意の数のレプリカを指定できます。 シャードごとに異なる数のレプリカを指定できます。

設定では、必要な数のクラスターを指定できます。

クラスターを表示するには、'system.clusters' テーブル。

の分散型エンジン能にすることで、社会とクラスターのように現地サーバーです。 サーバー設定ファイルにその設定を書き込む必要があります(すべてのクラスターのサーバーでさらに優れています)。

The Distributed engine requires writing clusters to the config file. Clusters from the config file are updated on the fly, without restarting the server. If you need to send a query to an unknown set of shards and replicas each time, you don't need to create a Distributed table – use the 'remote' 代わりにテーブル関数。 セクションを参照 [テーブル関数](#).

クラスターにデータを書き込む方法は二つあります:

まず、どのサーバーにどのデータを書き込むかを定義し、各シャードで直接書き込みを実行できます。 つまり、分散テーブルのテーブルに挿入を実行します "looks at". これは、対象領域の要件のために自明ではない可能性のあるシャーディングスキームを使用できるので、最も柔軟なソリューションです。 データは完全に独立して異なるシャードに書き込むことができるので、これも最適な解決策です。

次に、分散テーブルで`INSERT`を実行できます。 この場合、テーブルは挿入されたデータをサーバー自体に分散します。 分散テーブルに書き込むには、シャーディングキー(最後のパラメーター)が必要です。 さらに、シャードがひとつしかない場合、この場合は何も意味しないため、シャーディングキーを指定せずに書き込み操作が機能します。

各シャードには、設定ファイルで定義された重みを設定できます。 デフォルトでは、重みは重みと等しくなります。 データは、シャードの重みに比例する量でシャードに分散されます。 たとえば、シャードが二つあり、最初のシャードが9の重みを持ち、次のシャードが10の重みを持つ場合、最初のシャードは行の9/19部分に送信され、次のシャー

各シャードは、'internal\_replication' 設定ファイルで定義されたパラメータ。

このパラメータが ‘true’ 書き込み操作は、最初の正常なレプリカを選択し、それにデータを書き込みます。分散テーブルの場合は、この代替を使用します “looks at” 複製されたテーブル。言い換れば、データが書き込まれるテーブルがそれ自身を複製する場合。

に設定されている場合 ‘false’ データはすべてのレプリカに書き込まれます。本質的に、これは分散テーブルがデータ自身を複製することを意味します。これは、レプリケートされたテーブルを使用するよりも悪いことです。

データの行が送信されるシャードを選択するために、シャーディング式が分析され、残りの部分がシャードの総重量で除算されます。この行は、残りの半分の間隔に対応するシャードに送信されます。‘prev\_weight’ に ‘prev\_weights + weight’,ここで ‘prev\_weights’ は、最小の数を持つ破片の総重量です。‘weight’ この破片の重量です。たとえば、シャードが二つあり、最初のシャードの重みが9で、次のシャードの重みが10である場合、行は範囲[0,9)の残りのシャードの最初のシャードに送信され、

シャーディング式には、整数を返す定数およびテーブル列の任意の式を使用できます。たとえば、次の式を使用できます ‘rand()’ データのランダム分布の場合、または ‘UserID’ ユーザーのIDを分割することからの残りの部分による分配のために（単一のユーザーのデータは単一のシャード上に存在し、ユーザーによる実行と結合を簡素化する）。いずれかの列が十分に均等に分散されていない場合は、ハッシュ関数intHash64(UserID)でラップできます。

簡単なリマインダからの限定シshardingんを常に適しています。これは、データの中規模および大規模なボリューム（サーバーの数十）ではなく、データの非常に大規模なボリューム（サーバーの数百以上）のために動作します。後者の場合、分散テーブルのエントリを使用するのではなく、サブジェクト領域で必要なシャーディングスキームを使用します。

SELECT queries are sent to all the shards and work regardless of how data is distributed across the shards (they can be distributed completely randomly). When you add a new shard, you don't have to transfer the old data to it. You can write new data with a heavier weight - the data will be distributed slightly unevenly, but queries will work correctly and efficiently.

次の場合は、シャーディングスキームについて心配する必要があります:

- 特定のキーによるデータの結合(INまたはJOIN)を必要とするクエリが使用されます。このキーによってデータがシャードされる場合は、GLOBAL INまたはGLOBAL JOINの代わりにlocal INまたはJOINを使用できます。
- 多数のサーバーが、多数の小さなクエリ（個々のクライアント-ウェブサイト、広告主、またはパートナーのクエリ）で使用されます（数百以上）。小さなクエリがクラスタ全体に影響を与えないようにするには、単一のシャード上の单一のクライアントのデータを検索することが理にかなっていません。我々はYandexでのやったように。Metricaでは、biレベルのシャーディングを設定できます：クラスタ全体を次のように分割します “layers” ここで、レイヤーは複数のシャードで構成されます。単一のクライアントのデータは単一のレイヤー上にありますが、必要に応じてシャードをレイヤーに追加することができ、データはランダムに分散されます。分散テーブルはレイヤごとに作成され、グローバルクエリ用に単一の共有分散テーブルが作成されます。

データは非同期に書き込まれます。テーブルに挿入すると、データブロックはローカルファイルシステムに書き込まれます。データはできるだけ早くバックグラウンドでリモートサーバーに送信されます。データを送信するための期間は、[distributed\\_directory\\_monitor\\_sleep\\_time\\_ms](#) と [distributed\\_directory\\_monitor\\_max\\_sleep\\_time\\_ms](#) 設定。その Distributed エンジンは、挿入されたデータを含む各ファイルを別々に送信しますが、[distributed\\_directory\\_monitor\\_batch\\_inserts](#) 設定。この設定の改善にクラスターの性能をより一層の活用地域のサーバやネットワーク資源です。を確認しておきましょうか否かのデータが正常に送信されるチェックリストファイル（データまたは間に-をはさんだ）はテーブルディレクトリ：`/var/lib/clickhouse/data/database/table/`。

分散テーブルへの挿入後にサーバーが存在しなくなった場合、またはデバイスの障害後などに大まかな再起動が行われた場合は、挿入されたデータが失われ テーブルディレクトリで破損したデータ部分が検出されると、その部分は ‘broken’ 使用されなくなりました。

`Max_parallel_replicas`オプションを有効にすると、単一のシャード内のすべてのレプリカでクエリ処理が並列化されます。詳細については [max\\_parallel\\_replicas](#).

## 仮想列

- `_shard_num` — Contains the `shard_num` (from `system.clusters`). Type: `UInt32`.

## 注

以来 `remote/cluster` 表関数は内部的に同じ分散エンジンの一時インスタンスを作成します, `_shard_num` あまりにもそこに利用可能です。

も参照。

- **仮想列**

## クエリ処理の外部データ

ClickHouseでは、クエリの処理に必要なデータをSELECTクエリと共にサーバーに送信できます。このデータは一時テーブルに格納されます(セクションを参照 “Temporary tables”)とクエリで使用できます(たとえば、in演算子)。

たとえば、重要なユーザー識別子を持つテキストファイルがある場合は、このリストによるフィルタリングを使用するクエリと共にサーバーにアップロードで

大量の外部データを含む複数のクエリを実行する必要がある場合は、この機能を使用しないでください。事前にDBにデータをアップロードする方が良いです。

外部データは、コマンドラインクライアント(非対話モード)またはHTTPインターフェイスを使用してアップロードできます。

コマンドラインクライアントでは、parametersセクションを次の形式で指定できます

```
--external --file=... [--name=...] [--format=...] [--types=...|--structure=...]
```

して複数の部分のように、このテーブルが送信されます。

**-external** – Marks the beginning of a clause.

**-file** – Path to the file with the table dump, or -, which refers to stdin.  
Stdinから取得できるのは単一のテーブルのみです。

次のパラメータは省略可能です: **-name** – Name of the table. If omitted, `_data` is used.

**-format** – Data format in the file. If omitted, TabSeparated is used.

次のいずれかのパラメータが必要です:**-types** – A list of comma-separated column types. For example: `UInt64, String`. The columns will be named `_1, _2, ...`

**-structure** – The table structure in the format `userID UInt64, URL String`. 列名と型を定義します。

で指定されたファイル ‘file’ に指定された形式で解析されます。‘format’ で指定されたデータ型を使用します ‘types’ または ‘structure’ のテーブルがアップロードサーバへのアクセスが一時テーブルの名前 ‘name’.

例:

```
$ echo -ne "1\n2\n3\n" | clickhouse-client --query="SELECT count() FROM test.visits WHERE TraficSourceID IN _data" --external --file=- --types=Int8
849897
$ cat /etc/passwd | sed 's/:/\t/g' | clickhouse-client --query="SELECT shell, count() AS c FROM passwd GROUP BY shell ORDER BY c DESC" --external --file=- --name=passwd --structure='login String, unused String, uid UInt16, gid UInt16, comment String, home String, shell String'
/bin/sh 20
/bin/false 5
/bin/bash 4
/usr/sbin/nologin 1
/bin/sync 1
```

HTTPインターフェイスを使用する場合、外部データはmultipart/form-data形式で渡されます。各テーブルは別々のファイルとして送信されます。テーブル名は、ファイル名から取得されます。その‘query\_string’パラメータが渡されます‘name\_format’, ‘name\_types’, and ‘name\_structure’,ここで‘name’これらのパラメーターが対応するテーブルの名前を指定します。パラメーターの意味は、コマンドラインクライアントを使用する場合と同じです。

例:

```
$ cat /etc/passwd | sed 's/:/\t/g' > passwd.tsv

$ curl -F 'passwd=@passwd.tsv;' 'http://localhost:8123/?query=SELECT+shell,+count()+AS+c+FROM+passwd+GROUP+BY+shell+ORDER+BY+c+DESC&passwd_structure=login+String,+unused+String,+uid+UInt16,+gid+UInt16,+comment+String,+home+String,+shell+String'
/bin/sh 20
/bin/false 5
/bin/bash 4
/usr/sbin/nologin 1
/bin/sync 1
```

分散クエリ処理では、一時テーブルがすべてのリモートサーバーに送信されます。

## 辞書

その Dictionary エンジンは表示します 辞書 ClickHouseテーブルとしてのデータ。

例として、の辞書を考えてみましょう products 以下の構成で:

```

<dictionaries>
<dictionary>
    <name>products</name>
    <source>
        <odbc>
            <table>products</table>
            <connection_string>DSN=some-db-server</connection_string>
        </odbc>
    </source>
    <lifetime>
        <min>300</min>
        <max>360</max>
    </lifetime>
    <layout>
        <flat/>
    </layout>
    <structure>
        <id>
            <name>product_id</name>
        </id>
        <attribute>
            <name>title</name>
            <type>String</type>
            <null_value></null_value>
        </attribute>
    </structure>
</dictionary>
</dictionaries>

```

辞書データの照会:

```

SELECT
    name,
    type,
    key,
    attribute.names,
    attribute.types,
    bytes_allocated,
    element_count,
    source
FROM system.dictionaries
WHERE name = 'products'

```

name	type	key	attribute.names	attribute.types	bytes_allocated	element_count	source
products	Flat	UInt64	['title']	['String']	23065376	175032	ODBC: .products

を使用することができます [dictGet\\*](#) この形式の辞書データを取得する関数。

このビューは、生データを取得する必要がある場合や、JOIN 作戦だこれらのケースでは、Dictionary ディクショナリデータをテーブルに表示するエンジン。

構文:

```
CREATE TABLE %table_name% (%fields%) engine = Dictionary(%dictionary_name%)`
```

使用例:

```
create table products (product_id UInt64, title String) Engine = Dictionary(products);
```

Ok

テーブルにあるものを見てみなさい。

```
select * from products limit 1;
```

product_id	title
152689	Some item

## マージ

その `Merge` エンジン（と混同しないように `MergeTree`）を格納していないデータそのものができるから、他のテーブルを同時に

読み取りは自動的に並列化されます。表への書き込みはサポートされません。読み取り時には、実際に読み取られているテーブルのインデックスが存在する場合に使用されます。

その `Merge` データベース名とテーブルの正規表現です。

例：

```
Merge(hits, '^WatchLog')
```

データはのテーブルから読みれます `hits` 正規表現に一致する名前を持つデータベース '`^WatchLog`'.

データベース名の代わりに、文字列を返す定数式を使用できます。例えば, `currentDatabase()`.

Regular expressions — `re2` (PCREのサブセットをサポート)、大文字と小文字を区別します。

正規表現におけるシンボルのエスケープに関する注意 “match” セクション

読み取るテーブルを選択するとき、`Merge` 正規表現と一致しても、テーブル自体は選択されません。これはループを避けるためです。

二つを作成することができます `Merge` 無限にお互いのデータを読み込もうとするテーブルですが、これは良い考えではありません。

使用する典型的な方法 `Merge` エンジンは多数を使用のためです `TinyLog` 単一のテーブルを持つかのようにテーブル。

例2：

古いテーブル (`WatchLog_old`) があり、データを新しいテーブル (`WatchLog_new`) に移動せずにパーティション分割を変更することにしたとしましょう。

```
CREATE TABLE WatchLog_old(date Date, UserId Int64, EventType String, Cnt UInt64)
ENGINE=MergeTree(date, (UserId, EventType), 8192);
INSERT INTO WatchLog_old VALUES ('2018-01-01', 1, 'hit', 3);

CREATE TABLE WatchLog_new(date Date, UserId Int64, EventType String, Cnt UInt64)
ENGINE=MergeTree PARTITION BY date ORDER BY (UserId, EventType) SETTINGS index_granularity=8192;
INSERT INTO WatchLog_new VALUES ('2018-01-02', 2, 'hit', 3);

CREATE TABLE WatchLog as WatchLog_old ENGINE=Merge(currentDatabase(), '^WatchLog');

SELECT *
FROM WatchLog
```

date	User Id	Event Type	Cnt
2018-01-01	1	hit	3
2018-01-02	2	hit	3

## 仮想列

- `_table` — Contains the name of the table from which data was read. Type: 文字列.

定数条件は次のように設定できます `_table` で WHERE/PREWHERE 句(例えば, `WHERE _table='xyz'`)。この場合、読み取り操作は、条件がオンのテーブルに対してのみ実行されます `_table` は満足しているので、`_table` 列は索引として機能します。

も参照。

- [仮想列](#)

## ファイル

ファイルにテーブルエンジンのデータをファイルを使ったり、[ファイル形式](#) (TabSeparated、Nativeなど)。

使用例:

- ClickHouseからファイルにデータエクスポート。
- ある形式から別の形式にデータを変換します。
- データ更新にClickHouse経由で編集ファイルディスク。

## ClickHouseサーバーでの使用状況

### File(Format)

その Format パラメータを指定するか、ファイルのファイルフォーマット 実行するには  
`SELECT` この形式は、入力と実行のためにサポートされている必要があります  
`INSERT queries - for output. The available formats are listed in the`  
[形式](#) セクション

ClickHouseはファイルシステムのパスを `File` で定義されたフォルダを使用します [パス](#) サーバー構成の設定。

を使用して表を作成する場合 `File(Format)` で空のサブディレクトリとフォルダにまとめた。データがそのテーブルに書き込まれると、`data.Format` そのサブディレ

お手動で作成このサブフォルダやファイルサーバのファイルシステムとし `ATTACH` 一致する名前で情報を表すので、そのファイルからデータを照会することができます。

### 警告

ClickHouseはそのようなファイルへの外部の変更を追跡しないので、この機能に注意してください。ClickHouseとClickHouseの外部での同時書き込みの結果は未定義です。

例:

## 1. セットアップ file\_engine\_table テーブル:

```
CREATE TABLE file_engine_table (name String, value UInt32) ENGINE=File(TabSeparated)
```

デフォルトでClickHouse フォルダを作成します /var/lib/clickhouse/data/default/file\_engine\_table.

## 2. 手動で作成 /var/lib/clickhouse/data/default/file\_engine\_table/data.TabSeparated 含む:

```
$ cat data.TabSeparated
one 1
two 2
```

## 3. データの照会:

```
SELECT * FROM file_engine_table
```

name	value
one	1
two	2

## ClickHouseでの使用-ローカル

で **ツツイ姪ツ債ツケ** ファイルエンジンは、以下に加えて **Format**. デフォルトの入出力ストリームは、次のような数値または人間が読める名前で指定できます `0` または `stdin`, `1` または `stdout`.

例:

```
$ echo -e "1,2\n3,4" | clickhouse-local -q "CREATE TABLE table (a Int64, b Int64) ENGINE = File(CSV, stdin); SELECT a, b FROM table; DROP TABLE table"
```

## 実施内容

- 複数 **SELECT** クエリは同時に実行できますが **INSERT** クエリは互いに待機します。
- 新しいファイルを作成する **INSERT** クエリ。
- ファイルが存在する場合、**INSERT** それに新しい値を追加します。
- 対応していません:
  - ALTER**
  - SELECT ... SAMPLE**
  - 指標
  - 複製

## Null

Nullテーブルに書き込む場合、データは無視されます。 Nullテーブルから読み取る場合、応答は空です。

ただし、Nullテーブルにマテリアライズドビューを作成できます。 したがって、テーブルに書き込まれたデータはビュー内で終了します。

## セット

常にRAM内にあるデータ-セット。これは、in演算子の右側での使用を意図しています（セクションを参照してください“IN operators”）。

INSERTを使用すると、テーブルにデータを挿入できます。新しい要素がデータセットに追加され、重複は無視されます。

ができない行から選択します。データを取得する唯一の方法は、in演算子の右半分でデータを使用することです。

データは常にRAMにあります。INSERTでは、挿入されたデータのブロックもディスク上のテーブルのディレクトリに書き込まれます。サーバーを起動すると、このデータはRAMにロードされます。つまり、再起動後、データは所定の位置に残ります。

ラサーバを再起動し、ブロックのデータのディスクが失われることも想定されます。後者の場合、破損したデータを含むファイルを手動で削除する必要がある場合があります。

## 参加

で使用するための準備されたデータ構造 JOIN 作戦だ

## テーブルの作成

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1] [TTL expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2] [TTL expr2],
) ENGINE = Join(join_strictness, join_type, k1[, k2, ...])
```

の詳細な説明を見て下さい CREATE TABLE クエリ。

### エンジン変数

- join\_strictness – 厳密に結合する。
- join\_type – 結合タイプ。
- k1[, k2, ...] – Key columns from the USING 句は、JOIN 操作はでなされる。

入力 join\_strictness と join\_type 引用符のないパラメータ, Join(ANY, LEFT, col1). も一致しなければならない JOIN テーブルが使用される操作。パラメーターが一致しない場合、ClickHouseは例外をスローせず、誤ったデータを返す可能性があります。

## テーブルの使用法

### 例

左側のテーブルの作成:

```
CREATE TABLE id_val(`id` UInt32, `val` UInt32) ENGINE = TinyLog
```

```
INSERT INTO id_val VALUES (1,11)(2,12)(3,13)
```

右側の作成 Join テーブル:

```
CREATE TABLE id_val_join(`id` UInt32, `val` UInt8) ENGINE = Join(ANY, LEFT, id)
```

```
INSERT INTO id_val_join VALUES (1,21)(1,22)(3,23)
```

テーブルの結合:

```
SELECT * FROM id_val ANY LEFT JOIN id_val_join USING (id) SETTINGS join_use_nulls = 1
```

id	val	id_val	join.val
1	11		21
2	12		NULL
3	13		23

別 の方法として、Join 結合キーを指定するテーブル:

```
SELECT joinGet('id_val_join', 'val', toUInt32(1))
```

```
joinGet('id_val_join', 'val', toUInt32(1))  
21 |
```

## データの選択と挿入

以下を使用できます `INSERT` にデータを追加するクエリ `Join`-エンジンテーブル。 テーブルが作成された場合 `ANY` 厳密さ、重複キーのデータは無視されます。 と `ALL` 厳密さは、すべての行が追加されます。

を実行できません。 `SELECT` テーブルから直接クエリ。 代わりに、次のいずれかの方法を使用します:

- Aの右側にテーブルを置いて下さい `JOIN` 句。
- コール `joinGet` 辞書と同じ方法でテーブルからデータを抽出できる関数。

## 制限事項と設定

テーブルを作成するときは、次の設定が適用されます:

- `join_use_nulls`
- `max_rows_in_join`
- `max_bytes_in_join`
- `join_overflow_mode`
- `join_any_take_last_row`

その `Join`-エンジンテーブルは使用できません `GLOBAL JOIN` 作戦だ

その `Join`-エンジンは使用を許可する `join_use_nulls` の設定 `CREATE TABLE` 声明。 と `SELECT` クエリを使用できる `join_use_nulls` あまりにも。 あなたが違う場合 `join_use_nulls` 設定することができるエラー入社。 それは結合の種類に依存します。 使用するとき `joinGet` 関数、あなたは同じを使用する必要があります `join_use_nulls` 設定 `CREATE TABLE` と `SELECT` 声明。

## データ保存

`Join` テーブルデータは常にRAMにあります。 を挿入する際、リストClickHouseに書き込みデータブロックのディレクトリのディスクできるように復元され、サーバが再起動してしまいます。

場合はサーバが再起動誤り、データブロックのディスクがいます。この場合、破損したデータを含むファイルを手動で削除する必要があります。

## URL(URL,形式)

リモートHTTP/HTTPSサーバー上のデータを管理します。このエンジンは同様です  
に **ファイル** エンジン

### ClickHouseサーバーでのエンジンの使用

その `format ClickHouse` が使用できるものでなければなりません

`SELECT` クエリと、必要に応じて、`INSERTs.` サポートされている形式の完全な一覧については、  
**形式**.

その URL Uniform Resource Locator の構造に準拠している必要があります。指定した URL はサーバーを指す必要があります

HTTP または HTTPS を使用します。これは必要ありません  
サーバーからの応答を取得するための追加ヘッダー。

`INSERT` と `SELECT` 質問への `POST` と `GET` 要求、  
それぞれ。処理のため `POST` 要求は、リモートサーバーが  
**チャンク転送エンコード**.

HTTP GET リダイレクトホップの最大数を制限するには `max_http_get_redirects` 設定。

例：

1. 作成 `url_engine_table` サーバー上のテーブル：

```
CREATE TABLE url_engine_table (word String, value UInt64)
ENGINE=URL('http://127.0.0.1:12345/', CSV)
```

2. 標準の Python3 ツールを使用して基本的な HTTP サーバーを作成し、  
それを開始：

```
from http.server import BaseHTTPRequestHandler, HTTPServer

class CSVHTTPServer(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/csv')
        self.end_headers()

        self.wfile.write(bytes('Hello,1\nWorld,2\n', "utf-8"))

if __name__ == "__main__":
    server_address = ('127.0.0.1', 12345)
    HTTPServer(server_address, CSVHTTPServer).serve_forever()
```

```
$ python3 server.py
```

3. 要求データ：

```
SELECT * FROM url_engine_table
```

word	value
Hello	1
World	2

## 実施内容

- 読み書きできる並列
- 対応していません:
  - ALTER と SELECT...SAMPLE 作戦だ
  - インデックス。
  - 複製。

## 表示

ビューの実装に使用されます(詳細については、[CREATE VIEW query](#))。これはデータを格納せず、指定されたデータのみを格納します `SELECT` クエリ。テーブルから読み取るときに、このクエリが実行されます(クエリから不要な列がすべて削除されます)。

## マテリアライズドビュー

マテリアライズドビューの実装に使用されます(詳細については、[CREATE TABLE](#))。データを格納するために、ビューの作成時に指定された別のエンジンを使用します。読み込み時にテーブルから、使用してこのエンジンです。

## メモリ

メモリエンジンは、非圧縮形式でRAMにデータを格納します。データは、読み取り時に受信されるのとまったく同じ形式で格納されます。言い換えれば、この表からの読書は完全に無料です。

同時データアクセスは同期されます。ロックは短く、読み取り操作と書き込み操作は互いにロックしません。索引はサポートされません。読み取りは並列化されます。

単純なクエリでは、ディスクからの読み取り、データの解凍、または逆シリアル化が行われないため、最大生産性(10GB/秒以上)に達します。(多くの場合、MergeTreeエンジンの生産性はほぼ同じくらい高いことに注意してください。)

サーバーを再起動すると、テーブルからデータが消え、テーブルが空になります。

通常、このテーブルエンジンの使用は正当化されません。ただし、テストや、比較的少数の行(最大約100,000,000)で最大速度が必要なタスクに使用できます。

メモリーのエンジンを使用するシステムの一時テーブルの外部クエリデータの項をご参照ください “External data for processing a query”グローバルでの実装については、セクションを参照 “IN operators”).

## バッファ

バッファのデータを書き込RAM、定期的にフラッシングで別表に示す。読み取り動作中、データはバッファと他のテーブルから同時に読み取られます。

```
Buffer(database, table, num_layers, min_time, max_time, min_rows, max_rows, min_bytes, max_bytes)
```

エンジン変数:

- `database` – Database name. Instead of the database name, you can use a constant expression that returns a string.
- `table` – Table to flush data to.
- `num_layers` – Parallelism layer. Physically, the table will be represented as `num_layers` 独立した緩衝の。推奨値:16.
- `min_time, max_time, min_rows, max_rows, min_bytes, and max_bytes` – Conditions for flushing data from the buffer.

データはバッファからフラッシュされ、宛先テーブルに書き込まれます。`min*` 条件または少なくとも一つ `max*` 条件が満たされる。

- `min_time, max_time` – Condition for the time in seconds from the moment of the first write to the buffer.
- `min_rows, max_rows` – Condition for the number of rows in the buffer.
- `min_bytes, max_bytes` – Condition for the number of bytes in the buffer.

書き込み操作の間、データはaに挿入されます `num_layers` ランダムバッファの数。または、挿入するデータ部分が十分に大きい（より大きい）場合 `max_rows` または `max_bytes`）、それはバッファを省略して、宛先テーブルに直接書き込まれます。

データをフラッシュするための条件は、`num_layers` バッファ たとえば、`num_layers = 16` と `max_bytes = 100000000`、最大RAM消費量は1.6GBです。

例:

```
CREATE TABLE merge.hits_buffer AS merge.hits ENGINE = Buffer(merge, hits, 16, 10, 100, 10000, 1000000, 10000000, 100000000)
```

作成 ‘merge.hits\_buffer’ と同じ構造を持つテーブル ‘merge.hits’ そして、バッファエンジンを使用。このテーブルに書き込むとき、データはRAMにバッファされ、後で ‘merge.hits’ テーブル。16のバッファが作成されます。それぞれのデータは、100秒が経過した場合、または百万行が書き込まれた場合、または100MBのデータが書き込まれた場合、または10秒が経過して10,000行と10 たとえば、ただ一つの行が書き込まれている場合、100秒後には何があってもフラッシュされます。しかし、多くの行が書き込まれている場合、データは早くフラッシュされます。

`DROP TABLE` または `DETACH TABLE` を使用してサーバーを停止すると、バッファーデータも宛先テーブルにフラッシュされます。

データベース名とテーブル名には、单一引 quotation で空の文字列を設定できます。これは、宛先テーブルがないことを示します。この場合、データフラッシュ条件に達すると、バッファは単純にクリアされます。これは、データのウィンドウをメモリに保持する場合に便利です。

バッファーテーブルから読み取るとき、データはバッファと宛先テーブルの両方から処理されます(存在する場合)。バッファーテーブルはインデックスをサポートしません。つまり、バッファ内の中のデータは完全にスキャンされます。(下位テーブルのデータについては、サポートするインデックスが使用されます。)

バッファーテーブル内の列のセットが下位テーブル内の列のセットと一致しない場合、両方のテーブルに存在する列のサブセットが挿入されます。

バッファーテーブル内のいずれかの列と下位テーブルで型が一致しない場合、サーバログにエラーメッセージが入力され、バッファがクリアされます。

バッファがフラッシュされたときに下位テーブルが存在しない場合も同じことが起こります。

下位テーブルとバッファーテーブルに対して `ALTER` を実行する必要がある場合は、まずバッファーテーブルを削除し、下位テーブルに対して `ALTER` を実行してから、バッ

サーバーが異常に再起動されると、バッファー内のデータは失われます。

最終サンプルな正常に動作しないためのバッファです。これらの条件は宛先テーブルに渡されますが、バッファー内のデータの処理には使用されません。これらの特徴が必要のみを使用することをお勧めしますバッファのテーブルを書き込み、読みの行先表に示す。

を追加する場合にデータをバッファのバッファがロックされています。これにより、テーブルから読み取り操作が同時に実行される場合に遅延が発生します。

バッファテーブルに挿入されるデータは、下位テーブル内で異なる順序と異なるブロックになる場合があります。このため、CollapsingMergeTreeに正しく書き込むためにバッファテーブルを使用することは困難です。問題を回避するには、以下を設定できます ‘num\_layers’ に1.

の場合は先テーブルがそのまま再現され、期待の特徴を再現でテーブルが失われた書き込みバッファへの表に示す。行の順序とデータ部分のサイズがランダムに変更されると、データ重複排除が動作を終了します。‘exactly once’ 書く再現します。

これらの欠点があるため、まれにしかバッファテーブルの使用を推奨できません。

バッファテーブルは、単位時間にわたって多数のサーバーから多数の挿入を受信し、挿入の前にデータをバッファリングできない場合に使用されます。

バッファテーブルであっても、一度にデータ行を挿入することは意味がありません。これにより、毎秒数千行の速度が生成されますが、より大きなデータブロックを挿入すると、毎秒百万行を超える速度が生成されます（セクションを参照 “Performance”）。

## Generaterandom

のGenerateRandomテーブルエンジンの生産ランダムなデータが与えられたテーブルのスキーマ。

使用例:

- 再現可能で大きいテーブルを移入するテストの使用。
- ファジングテストのランダム入力を生成します。

## ClickHouseサーバーでの使用状況

```
ENGINE = GenerateRandom(random_seed, max_string_length, max_array_length)
```

その `max_array_length` と `max_string_length` パラメータを指定し最大限の長さのすべて生成されたデータに対応する配列の列と文字列。

Generate table engineのサポートのみ SELECT クエリ。

それはすべて **データ型** を除いてテーブルに格納することができます `LowCardinality` と `AggregateFunction`.

例:

1. セットアップ `generate_engine_table` テーブル:

```
CREATE TABLE generate_engine_table (name String, value UInt32) ENGINE = GenerateRandom(1, 5, 3)
```

2. データの照会:

```
SELECT * FROM generate_engine_table LIMIT 3
```

name	value
c4xj	1412771199
r	1791099446
7#\$	124312908

## 実施内容

- 対応していません:
  - ALTER
  - SELECT ... SAMPLE
  - INSERT
  - 指数
  - 複製

## データベース

データベースエ

既定では、ClickHouseはネイティブのデータベースエンジンを使用します。表エンジン そして SQL方言。

次のデータベースエンジンも使用できます:

- MySQL
- 急げ者

## [experimental] MaterializedMySQL

### Warning

This is an experimental feature that should not be used in production.

Creates ClickHouse database with all the tables existing in MySQL, and all the data in those tables.

ClickHouse server works as MySQL replica. It reads binlog and performs DDL and DML queries.

## Creating a Database

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

### Engine Parameters

- host:port — MySQL server endpoint.
- database — MySQL database name.
- user — MySQL user.
- password — User password.

## Engine Settings

- `max_rows_in_buffer` — Maximum number of rows that data is allowed to cache in memory (for single table and the cache data unable to query). When this number is exceeded, the data will be materialized. Default: 65 505.
- `max_bytes_in_buffer` — Maximum number of bytes that data is allowed to cache in memory (for single table and the cache data unable to query). When this number is exceeded, the data will be materialized. Default: 1 048 576.
- `max_rows_in_buffers` — Maximum number of rows that data is allowed to cache in memory (for database and the cache data unable to query). When this number is exceeded, the data will be materialized. Default: 65 505.
- `max_bytes_in_buffers` — Maximum number of bytes that data is allowed to cache in memory (for database and the cache data unable to query). When this number is exceeded, the data will be materialized. Default: 1 048 576.
- `max_flush_data_time` — Maximum number of milliseconds that data is allowed to cache in memory (for database and the cache data unable to query). When this time is exceeded, the data will be materialized. Default: 1000.
- `max_wait_time_when_mysql_unavailable` — Retry interval when MySQL is not available (milliseconds). Negative value disables retry. Default: 1000.
- `allows_query_when_mysql_lost` — Allows to query a materialized table when MySQL is lost. Default: 0 (false).

```
CREATE DATABASE mysql ENGINE = MaterializedMySQL('localhost:3306', 'db', 'user', '***')
SETTINGS
    allows_query_when_mysql_lost=true,
    max_wait_time_when_mysql_unavailable=10000;
```

## Settings on MySQL-server Side

For the correct work of `MaterializedMySQL`, there are few mandatory MySQL-side configuration settings that must be set:

- `default_authentication_plugin = mysql_native_password` since `MaterializedMySQL` can only authorize with this method.
- `gtid_mode = on` since GTID based logging is a mandatory for providing correct `MaterializedMySQL` replication.

## Attention

While turning on `gtid_mode` you should also specify `enforce_gtid_consistency = on`.

## Virtual Columns

When working with the `MaterializedMySQL` database engine, `ReplacingMergeTree` tables are used with virtual `_sign` and `_version` columns.

- `_version` — Transaction counter. Type `UInt64`.
- `_sign` — Deletion mark. Type `Int8`. Possible values:
  - 1 — Row is not deleted,
  - -1 — Row is deleted.

# Data Types Support

MySQL	ClickHouse
TINY	Int8
SHORT	Int16
INT24	Int32
LONG	UInt32
LONGLONG	UInt64
FLOAT	Float32
DOUBLE	Float64
DECIMAL, NEWDECIMAL	Decimal
DATE, NEWDATE	Date
DATETIME, TIMESTAMP	DateTime
DATETIME2, TIMESTAMP2	DateTime64
ENUM	Enum
STRING	String
VARCHAR, VAR_STRING	String
BLOB	String
BINARY	FixedString

Nullable is supported.

Other types are not supported. If MySQL table contains a column of such type, ClickHouse throws exception "Unhandled data type" and stops replication.

## Specifics and Recommendations

### Compatibility Restrictions

Apart of the data types limitations there are few restrictions comparing to MySQL databases, that should be resolved before replication will be possible:

- Each table in MySQL should contain PRIMARY KEY.
- Replication for tables, those are containing rows with ENUM field values out of range (specified in ENUM signature) will not work.

### DDL Queries

MySQL DDL queries are converted into the corresponding ClickHouse DDL queries ([ALTER](#), [CREATE](#), [DROP](#), [RENAME](#)). If ClickHouse cannot parse some DDL query, the query is ignored.

## Data Replication

MaterializedMySQL does not support direct `INSERT`, `DELETE` and `UPDATE` queries. However, they are supported in terms of data replication:

- MySQL `INSERT` query is converted into `INSERT` with `_sign=1`.
- MySQL `DELETE` query is converted into `INSERT` with `_sign=-1`.
- MySQL `UPDATE` query is converted into `INSERT` with `_sign=-1` and `INSERT` with `_sign=1`.

## Selecting from MaterializedMySQL Tables

`SELECT` query from MaterializedMySQL tables has some specifics:

- If `_version` is not specified in the `SELECT` query, [FINAL](#) modifier is used. So only rows with `MAX(_version)` are selected.
- If `_sign` is not specified in the `SELECT` query, `WHERE _sign=1` is used by default. So the deleted rows are not included into the result set.
- The result includes columns comments in case they exist in MySQL database tables.

## Index Conversion

MySQL `PRIMARY KEY` and `INDEX` clauses are converted into `ORDER BY` tuples in ClickHouse tables.

ClickHouse has only one physical order, which is determined by `ORDER BY` clause. To create a new physical order, use [materialized views](#).

## Notes

- Rows with `_sign=-1` are not deleted physically from the tables.
- Cascade `UPDATE/DELETE` queries are not supported by the MaterializedMySQL engine.
- Replication can be easily broken.
- Manual operations on database and tables are forbidden.
- MaterializedMySQL is influenced by [optimize\\_on\\_insert](#) setting. The data is merged in the corresponding table in the MaterializedMySQL database when a table in the MySQL server changes.

## Examples of Use

Queries in MySQL:

```
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
1	11	
2	22	Wow!

Database in ClickHouse, exchanging data with the MySQL server:

The database and the table created:

```
CREATE DATABASE mysql ENGINE = MaterializedMySQL('localhost:3306', 'db', 'user', '***');
SHOW TABLES FROM mysql;
```

name
test

After inserting data:

```
SELECT * FROM mysql.test;
```

a	b
1	11
2	22

After deleting data, adding the column and updating:

```
SELECT * FROM mysql.test;
```

a	b	c
2	222	Wow!

## [experimental] MaterializedPostgreSQL

Creates ClickHouse database with an initial data dump of PostgreSQL database tables and starts replication process, i.e. executes background job to apply new changes as they happen on PostgreSQL database tables in the remote PostgreSQL database.

ClickHouse server works as PostgreSQL replica. It reads WAL and performs DML queries. DDL is not replicated, but can be handled (described below).

### Creating a Database

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

#### Engine Parameters

- `host:port` — PostgreSQL server endpoint.
- `database` — PostgreSQL database name.

- `user` — PostgreSQL user.
- `password` — User password.

## Dynamically adding new tables to replication

```
ATTACH TABLE postgres_database.new_table;
```

It will work as well if there is a setting `materialized_postgresql_tables_list`.

## Dynamically removing tables from replication

```
DETACH TABLE postgres_database.table_to_remove;
```

## Settings

- `materialized_postgresql_max_block_size`
- `materialized_postgresql_tables_list`
- `materialized_postgresql_allow_automatic_update`
- `materialized_postgresql_replication_slot`
- `materialized_postgresql_snapshot`

```
CREATE DATABASE database1
ENGINE = MaterializedPostgreSQL('postgres1:5432', 'postgres_database', 'postgres_user', 'postgres_password')
SETTINGS materialized_postgresql_max_block_size = 65536,
materialized_postgresql_tables_list = 'table1,table2,table3';
```

```
SELECT * FROM database1.table1;
```

It is also possible to change settings at run time.

```
ALTER DATABASE postgres_database MODIFY SETTING materialized_postgresql_max_block_size = <new_size>;
```

## Requirements

1. The `wal_level` setting must have a value `logical` and `max_replication_slots` parameter must have a value at least `2` in the PostgreSQL config file.
2. Each replicated table must have one of the following `replica identity`:
  - primary key (by default)
  - index

```
postgres# CREATE TABLE postgres_table (a Integer NOT NULL, b Integer, c Integer NOT NULL, d Integer, e Integer NOT NULL);
postgres# CREATE unique INDEX postgres_table_index on postgres_table(a, c, e);
postgres# ALTER TABLE postgres_table REPLICA IDENTITY USING INDEX postgres_table_index;
```

The primary key is always checked first. If it is absent, then the index, defined as replica identity index, is checked.

If the index is used as a replica identity, there has to be only one such index in a table.

You can check what type is used for a specific table with the following command:

```
postgres# SELECT CASE relreplicant
    WHEN 'd' THEN 'default'
    WHEN 'n' THEN 'nothing'
    WHEN 'f' THEN 'full'
    WHEN 'i' THEN 'index'
END AS replica_identity
FROM pg_class
WHERE oid = 'postgres_table'::regclass;
```

## Warning

Replication of **TOAST** values is not supported. The default value for the data type will be used.

## Example of Use

```
CREATE DATABASE postgresql_db
ENGINE = MaterializedPostgreSQL('postgres1:5432', 'postgres_database', 'postgres_user', 'postgres_password');

SELECT * FROM postgresql_db.postgres_table;
```

## Notes

### Failover of the logical replication slot

Logical Replication Slots which exist on the primary are not available on standby replicas.

So if there is a failover, new primary (the old physical standby) won't be aware of any slots which were existing with old primary. This will lead to a broken replication from PostgreSQL.

A solution to this is to manage replication slots yourself and define a permanent replication slot (some information can be found [here](#)). You'll need to pass slot name via `materialized_postgresql_replication_slot` setting, and it has to be exported with `EXPORT SNAPSHOT` option. The snapshot identifier needs to be passed via `materialized_postgresql_snapshot` setting.

Please note that this should be used only if it is actually needed. If there is no real need for that or full understanding why, then it is better to allow the table engine to create and manage its own replication slot.

### Example (from [@bchrobot](#))

1. Configure replication slot in PostgreSQL.

```
apiVersion: "acid.zalan.do/v1"
kind: postgresql
metadata:
  name: acid-demo-cluster
spec:
  numberOflInstances: 2
  postgresql:
    parameters:
      wal_level: logical
    patroni:
      slots:
        clickhouse_sync:
          type: logical
          database: demodb
          plugin: pgoutput
```

2. Wait for replication slot to be ready, then begin a transaction and export the transaction snapshot identifier:

```
BEGIN;  
SELECT pg_export_snapshot();
```

### 3. In ClickHouse create database:

```
CREATE DATABASE demodb  
ENGINE = MaterializedPostgreSQL('postgres1:5432', 'postgres_database', 'postgres_user', 'postgres_password')  
SETTINGS  
    materialized_postgresql_replication_slot = 'clickhouse_sync',  
    materialized_postgresql_snapshot = '0000000A-0000023F-3',  
    materialized_postgresql_tables_list = 'table1,table2,table3';
```

### 4. End the PostgreSQL transaction once replication to ClickHouse DB is confirmed. Verify that replication continues after failover:

```
kubectl exec acid-demo-cluster-0 -c postgres -- su postgres -c 'patronictl failover --candidate acid-demo-cluster-1 --force'
```

## MySQL

で接続するデータベースのリモート MySQL サーバを実行 `INSERT` と `SELECT` ClickHouse と MySQL の間でデータを交換するためのクエリ。

その MySQL データベースエンジ `SHOW TABLES` または `SHOW CREATE TABLE`.

次のクエリは実行できません:

- `RENAME`
- `CREATE TABLE`
- `ALTER`

## データベースの作成

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

### エンジン変数

- `host:port` — MySQL server address.
- `database` — Remote database name.
- `user` — MySQL user.
- `password` — User password.

## データ型のサポート

MySQL	クリックハウス
UNSIGNED TINYINT	UInt8
TINYINT	Int8

UNSIGNED SMALLINT	UInt16
SMALLINT	Int16
UNSIGNED INT, UNSIGNED MEDIUMINT	UInt32
INT, MEDIUMINT	Int32
UNSIGNED BIGINT	UInt64
BIGINT	Int64
FLOAT	Float32
DOUBLE	Float64
DATE	日付
DATETIME, TIMESTAMP	DateTime
BINARY	FixedString

他のすべてのMySQLデータ型に変換され 文字列.

Null可能 サポートされます。

## 使用例

MySQLのテーブル:

```
mysql> USE test;
Database changed

mysql> CREATE TABLE `mysql_table` (
->   `int_id` INT NOT NULL AUTO_INCREMENT,
->   `float` FLOAT NOT NULL,
->   PRIMARY KEY (`int_id`));
Query OK, 0 rows affected (0,09 sec)

mysql> insert into mysql_table (`int_id`, `float`) VALUES (1,2);
Query OK, 1 row affected (0,00 sec)

mysql> select * from mysql_table;
+-----+-----+
| int_id | value |
+-----+-----+
|     1 |    2 |
+-----+-----+
1 row in set (0,00 sec)
```

ClickHouseのデータベース、MySQLサーバとのデータ交換:

```
CREATE DATABASE mysql_db ENGINE = MySQL('localhost:3306', 'test', 'my_user', 'user_password')
```

```
SHOW DATABASES
```

```
name
default |
mysql_db |
system |
```

```
SHOW TABLES FROM mysql_db
```

```
name
mysql_table |
```

```
SELECT * FROM mysql_db.mysql_table
```

```
int_id   value
1       2 |
```

```
INSERT INTO mysql_db.mysql_table VALUES (3,4)
```

```
SELECT * FROM mysql_db.mysql_table
```

```
int_id   value
1       2 |
3       4 |
```

## 急け者

RAM内のテーブルのみを保持 `expiration_time_in_seconds` 最後のアクセスの秒後。 \*Logテーブルでのみ使用できます。

これは、アクセス間に長い時間間隔がある多くの小さな\*ログテーブルを格納するために最適化されています。

## データベースの作成

```
CREATE DATABASE testlazy ENGINE = Lazy(expiration_time_in_seconds);
```

## Atomic

It supports non-blocking DROP and RENAME TABLE queries and atomic EXCHANGE TABLES t1 AND t2 queries.  
Atomic database engine is used by default.

## Creating a Database

```
CREATE DATABASE test ENGINE = Atomic;
```

## SQLite

Allows to connect to **SQLite** database and perform **INSERT** and **SELECT** queries to exchange data between ClickHouse and SQLite.

## Creating a Database

```
CREATE DATABASE sqlite_database  
ENGINE = SQLite('db_path')
```

### Engine Parameters

- `db_path` — Path to a file with SQLite database.

## Data Types Support

SQLite	ClickHouse
INTEGER	Int32
REAL	Float32
TEXT	String
BLOB	String

## Specifics and Recommendations

SQLite stores the entire database (definitions, tables, indices, and the data itself) as a single cross-platform file on a host machine. During writing SQLite locks the entire database file, therefore write operations are performed sequentially. Read operations can be multitasked.

SQLite does not require service management (such as startup scripts) or access control based on GRANT and passwords. Access control is handled by means of file-system permissions given to the database file itself.

## Usage Example

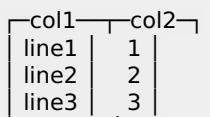
Database in ClickHouse, connected to the SQLite:

```
CREATE DATABASE sqlite_db ENGINE = SQLite('sqlite.db');  
SHOW TABLES FROM sqlite_db;
```



Shows the tables:

```
SELECT * FROM sqlite_db.table1;
```



Inserting data into SQLite table from ClickHouse table:

```
CREATE TABLE clickhouse_table(`col1` String,`col2` Int16) ENGINE = MergeTree() ORDER BY col2;
INSERT INTO clickhouse_table VALUES ('text',10);
INSERT INTO sqlite_db.table1 SELECT * FROM clickhouse_table;
SELECT * FROM sqlite_db.table1;
```

col1	col2
line1	1
line2	2
line3	3
text	10

## PostgreSQL

Allows to connect to databases on a remote [PostgreSQL](#) server. Supports read and write operations ([SELECT](#) and [INSERT](#) queries) to exchange data between ClickHouse and PostgreSQL.

Gives the real-time access to table list and table structure from remote PostgreSQL with the help of [SHOW TABLES](#) and [DESCRIBE TABLE](#) queries.

Supports table structure modifications ([ALTER TABLE ... ADD|DROP COLUMN](#)). If [use\\_table\\_cache](#) parameter (see the Engine Parameters below) is set to [1](#), the table structure is cached and not checked for being modified, but can be updated with [DETACH](#) and [ATTACH](#) queries.

## Creating a Database

```
CREATE DATABASE test_database
ENGINE = PostgreSQL('host:port', 'database', 'user', 'password'[, `schema`, `use_table_cache`]);
```

### Engine Parameters

- `host:port` — PostgreSQL server address.
- `database` — Remote database name.
- `user` — PostgreSQL user.
- `password` — User password.
- `schema` — PostgreSQL schema.
- `use_table_cache` — Defines if the database table structure is cached or not. Optional. Default value: `0`.

## Data Types Support

PostgreSQL	ClickHouse
DATE	Date
TIMESTAMP	DateTime
REAL	Float32
DOUBLE	Float64

PostgreSQL	ClickHouse
DECIMAL, NUMERIC	Decimal
SMALLINT	Int16
INTEGER	Int32
BIGINT	Int64
SERIAL	UInt32
BIGSERIAL	UInt64
TEXT, CHAR	String
INTEGER	Nullable(Int32)
ARRAY	Array

## Examples of Use

Database in ClickHouse, exchanging data with the PostgreSQL server:

```
CREATE DATABASE test_database
ENGINE = PostgreSQL('postgres1:5432', 'test_database', 'postgres', 'mysecretpassword', 1);
```

```
SHOW DATABASES;
```

name
default
test_database
system

```
SHOW TABLES FROM test_database;
```

name
test_table

Reading data from the PostgreSQL table:

```
SELECT * FROM test_database.test_table;
```

id	value
1	2

Writing data to the PostgreSQL table:

```
INSERT INTO test_database.test_table VALUES (3,4);
SELECT * FROM test_database.test_table;
```

int_id	value
1	2
3	4

Consider the table structure was modified in PostgreSQL:

```
postgres> ALTER TABLE test_table ADD COLUMN data Text
```

As the `use_table_cache` parameter was set to `1` when the database was created, the table structure in ClickHouse was cached and therefore not modified:

```
DESCRIBE TABLE test_database.test_table;
```

name	type
id	Nullable(Integer)
value	Nullable(Integer)

After detaching the table and attaching it again, the structure was updated:

```
DETACH TABLE test_database.test_table;
ATTACH TABLE test_database.test_table;
DESCRIBE TABLE test_database.test_table;
```

name	type
id	Nullable(Integer)
value	Nullable(Integer)
data	Nullable(String)

## [experimental] Replicated

The engine is based on the [Atomic](#) engine. It supports replication of metadata via DDL log being written to ZooKeeper and executed on all of the replicas for a given database.

One ClickHouse server can have multiple replicated databases running and updating at the same time. But there can't be multiple replicas of the same replicated database.

## Creating a Database

```
CREATE DATABASE testdb ENGINE = Replicated('zoo_path', 'shard_name', 'replica_name') [SETTINGS ...]
```

### Engine Parameters

- `zoo_path` — ZooKeeper path. The same ZooKeeper path corresponds to the same database.
- `shard_name` — Shard name. Database replicas are grouped into shards by `shard_name`.
- `replica_name` — Replica name. Replica names must be different for all replicas of the same shard.

## Warning

For **ReplicatedMergeTree** tables if no arguments provided, then default arguments are used: `/clickhouse` and `{replica}`. These can be changed in the server settings **default\_replica\_path** and **default\_replica\_name**. Macro `{uuid}` is unfolded to table's `uuid`, `{shard}` and `{replica}` are unfolded to values from server config, not from database engine arguments. But in the future, it will be possible to use `shard_name` and `replica_name` of Replicated database.

## Specifics and Recommendations

DDL queries with `Replicated` database work in a similar way to `ON CLUSTER` queries, but with minor differences.

First, the DDL request tries to execute on the initiator (the host that originally received the request from the user). If the request is not fulfilled, then the user immediately receives an error, other hosts do not try to fulfill it. If the request has been successfully completed on the initiator, then all other hosts will automatically retry until they complete it. The initiator will try to wait for the query to be completed on other hosts (no longer than `distributed_ddl_task_timeout`) and will return a table with the query execution statuses on each host.

The behavior in case of errors is regulated by the `distributed_ddl_output_mode` setting, for a `Replicated` database it is better to set it to `null_status_on_timeout` — i.e. if some hosts did not have time to execute the request for `distributed_ddl_task_timeout`, then do not throw an exception, but show the `NULL` status for them in the table.

The `system.clusters` system table contains a cluster named like the replicated database, which consists of all replicas of the database. This cluster is updated automatically when creating/deleting replicas, and it can be used for `Distributed` tables.

When creating a new replica of the database, this replica creates tables by itself. If the replica has been unavailable for a long time and has lagged behind the replication log — it checks its local metadata with the current metadata in ZooKeeper, moves the extra tables with data to a separate non-replicated database (so as not to accidentally delete anything superfluous), creates the missing tables, updates the table names if they have been renamed. The data is replicated at the `ReplicatedMergeTree` level, i.e. if the table is not replicated, the data will not be replicated (the database is responsible only for metadata).

`ALTER TABLE ATTACH|FETCH|DROP|DROP DETACHED|DETACH PARTITION|PART` queries are allowed but not replicated. The database engine will only add/fetch/remove the partition/part to the current replica. However, if the table itself uses a `Replicated` table engine, then the data will be replicated after using `ATTACH`.

## Usage Example

Creating a cluster with three hosts:

```
node1 :) CREATE DATABASE r ENGINE=Replicated('some/path/r','shard1','replica1');
node2 :) CREATE DATABASE r ENGINE=Replicated('some/path/r','shard1','other_replica');
node3 :) CREATE DATABASE r ENGINE=Replicated('some/path/r','other_shard','{replica}');
```

Running the DDL-query:

```
CREATE TABLE r.rmt (n UInt64) ENGINE=ReplicatedMergeTree ORDER BY n;
```

hosts		status	error	num_hosts_remaining	num_hosts_active
shard1 replica1	0		2	0	
shard1 other_replica	0		1	0	
other_shard r1	0		0	0	

Showing the system table:

```
SELECT cluster, shard_num, replica_num, host_name, host_address, port, is_local
FROM system.clusters WHERE cluster='r';
```

cluster	shard_num	replica_num	host_name	host_address	port	is_local
r	1	1	node3	127.0.0.1	9002	0
r	2	1	node2	127.0.0.1	9001	0
r	2	2	node1	127.0.0.1	9000	1

Creating a distributed table and inserting the data:

```
node2 :) CREATE TABLE r.d (n UInt64) ENGINE=Distributed('r','r','rmt', n % 2);
node3 :) INSERT INTO r SELECT * FROM numbers(10);
node1 :) SELECT materialize(hostName()) AS host, groupArray(n) FROM r.d GROUP BY host;
```

hosts	groupArray(n)
node1	[1,3,5,7,9]
node2	[0,2,4,6,8]

Adding replica on the one more host:

```
node4 :) CREATE DATABASE r ENGINE=Replicated('some/path/r','other_shard','r2');
```

The cluster configuration will look like this:

cluster	shard_num	replica_num	host_name	host_address	port	is_local
r	1	1	node3	127.0.0.1	9002	0
r	1	2	node4	127.0.0.1	9003	0
r	2	1	node2	127.0.0.1	9001	0
r	2	2	node1	127.0.0.1	9000	1

The distributed table also will get data from the new host:

```
node2 :) SELECT materialize(hostName()) AS host, groupArray(n) FROM r.d GROUP BY host;
```

hosts	groupArray(n)
node2	[1,3,5,7,9]
node4	[0,2,4,6,8]

## SQLリファレンス

ClickHouseは次の種類のクエリをサポートします:

- **SELECT**
  - **INSERT INTO**
  - **CREATE**
  - **ALTER**
  - その他の種類のクエリ
- 

## SELECT Query

**SELECT** queries perform data retrieval. By default, the requested data is returned to the client, while in conjunction with **INSERT INTO** it can be forwarded to a different table.

### Syntax

```
[WITH expr_list|(subquery)]
SELECT [DISTINCT] expr_list
[FROM [db.]table | (subquery) | table_function] [FINAL]
[SAMPLE sample_coeff]
[ARRAY JOIN ...]
[GLOBAL] [ANY|ALL|ASOF] [INNER|LEFT|RIGHT|FULL|CROSS] [OUTER|SEMI|ANTI] JOIN (subquery)|table (ON
<expr_list>)|(USING <column_list>)
[PREWHERE expr]
[WHERE expr]
[GROUP BY expr_list] [WITH ROLLUP|WITH CUBE] [WITH TOTALS]
[HAVING expr]
[ORDER BY expr_list] [WITH FILL] [FROM expr] [TO expr] [STEP expr]
[LIMIT offset_value, ]n BY columns]
[LIMIT [n, ]m] [WITH TIES]
[SETTINGS ...]
[UNION ...]
[INTO OUTFILE filename]
[FORMAT format]
```

All clauses are optional, except for the required list of expressions immediately after **SELECT** which is covered in more detail [below](#).

Specifics of each optional clause are covered in separate sections, which are listed in the same order as they are executed:

- **WITH clause**
- **FROM clause**
- **SAMPLE clause**
- **JOIN clause**
- **PREWHERE clause**
- **WHERE clause**
- **GROUP BY clause**
- **LIMIT BY clause**
- **HAVING clause**
- **SELECT clause**

- [DISTINCT clause](#)
- [LIMIT clause](#)
- [OFFSET clause](#)
- [UNION clause](#)
- [INTO OUTFILE clause](#)
- [FORMAT clause](#)

## SELECT Clause

[Expressions](#) specified in the `SELECT` clause are calculated after all the operations in the clauses described above are finished. These expressions work as if they apply to separate rows in the result. If expressions in the `SELECT` clause contain aggregate functions, then ClickHouse processes aggregate functions and expressions used as their arguments during the [GROUP BY](#) aggregation.

If you want to include all columns in the result, use the asterisk (\*) symbol. For example, `SELECT * FROM ...`

### COLUMNS expression

To match some columns in the result with a [re2](#) regular expression, you can use the `COLUMNS` expression.

```
COLUMNS('regexp')
```

For example, consider the table:

```
CREATE TABLE default.col_names (aa Int8, ab Int8, bc Int8) ENGINE = TinyLog
```

The following query selects data from all the columns containing the `a` symbol in their name.

```
SELECT COLUMNS('a') FROM col_names
```

aa	ab
1	1

The selected columns are returned not in the alphabetical order.

You can use multiple `COLUMNS` expressions in a query and apply functions to them.

For example:

```
SELECT COLUMNS('a'), COLUMNS('c'), toTypeName(COLUMNS('c')) FROM col_names
```

aa	ab	bc	toTypeName(bc)
1	1	1	Int8

Each column returned by the `COLUMNS` expression is passed to the function as a separate argument. Also you can pass other arguments to the function if it supports them. Be careful when using functions. If a function doesn't support the number of arguments you have passed to it, ClickHouse throws an exception.

For example:

```
SELECT COLUMNS('a') + COLUMNS('c') FROM col_names
```

Received exception from server (version 19.14.1):  
Code: 42. DB::Exception: Received from localhost:9000. DB::Exception: Number of arguments for function plus doesn't match: passed 3, should be 2.

In this example, `COLUMNS('a')` returns two columns: `aa` and `ab`. `COLUMNS('c')` returns the `bc` column. The `+` operator can't apply to 3 arguments, so ClickHouse throws an exception with the relevant message.

Columns that matched the `COLUMNS` expression can have different data types. If `COLUMNS` doesn't match any columns and is the only expression in `SELECT`, ClickHouse throws an exception.

## Asterisk

You can put an asterisk in any part of a query instead of an expression. When the query is analyzed, the asterisk is expanded to a list of all table columns (excluding the `MATERIALIZED` and `ALIAS` columns). There are only a few cases when using an asterisk is justified:

- When creating a table dump.
- For tables containing just a few columns, such as system tables.
- For getting information about what columns are in a table. In this case, set `LIMIT 1`. But it is better to use the `DESC TABLE` query.
- When there is strong filtration on a small number of columns using `PREDWHERE`.
- In subqueries (since columns that aren't needed for the external query are excluded from subqueries).

In all other cases, we don't recommend using the asterisk, since it only gives you the drawbacks of a columnar DBMS instead of the advantages. In other words using the asterisk is not recommended.

## Extreme Values

In addition to results, you can also get minimum and maximum values for the results columns. To do this, set the `extremes` setting to 1. Minimums and maximums are calculated for numeric types, dates, and dates with times. For other columns, the default values are output.

An extra two rows are calculated – the minimums and maximums, respectively. These extra two rows are output in `JSON*`, `TabSeparated*`, and `Pretty*` `formats`, separate from the other rows. They are not output for other formats.

In `JSON*` formats, the extreme values are output in a separate 'extremes' field. In `TabSeparated*` formats, the row comes after the main result, and after 'totals' if present. It is preceded by an empty row (after the other data). In `Pretty*` formats, the row is output as a separate table after the main result, and after `totals` if present.

Extreme values are calculated for rows before `LIMIT`, but after `LIMIT BY`. However, when using `LIMIT offset, size`, the rows before `offset` are included in extremes. In stream requests, the result may also include a small number of rows that passed through `LIMIT`.

## Notes

You can use synonyms (AS aliases) in any part of a query.

The `GROUP BY` and `ORDER BY` clauses do not support positional arguments. This contradicts MySQL, but conforms to standard SQL. For example, `GROUP BY 1, 2` will be interpreted as grouping by constants (i.e. aggregation of all rows into one).

# Implementation Details

If the query omits the `DISTINCT`, `GROUP BY` and `ORDER BY` clauses and the `IN` and `JOIN` subqueries, the query will be completely stream processed, using  $O(1)$  amount of RAM. Otherwise, the query might consume a lot of RAM if the appropriate restrictions are not specified:

- `max_memory_usage`
- `max_rows_to_group_by`
- `max_rows_to_sort`
- `max_rows_in_distinct`
- `max_bytes_in_distinct`
- `max_rows_in_set`
- `max_bytes_in_set`
- `max_rows_in_join`
- `max_bytes_in_join`
- `max_bytes_before_external_sort`
- `max_bytes_before_external_group_by`

For more information, see the section “Settings”. It is possible to use external sorting (saving temporary tables to a disk) and external aggregation.

## SELECT modifiers

You can use the following modifiers in `SELECT` queries.

### APPLY

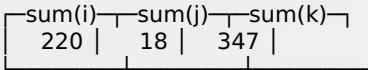
Allows you to invoke some function for each row returned by an outer table expression of a query.

#### Syntax:

```
SELECT <expr> APPLY( <func> ) FROM [db.]table_name
```

#### Example:

```
CREATE TABLE columns_transformers (i Int64, j Int16, k Int64) ENGINE = MergeTree ORDER by (i);
INSERT INTO columns_transformers VALUES (100, 10, 324), (120, 8, 23);
SELECT * APPLY(sum) FROM columns_transformers;
```



### EXCEPT

Specifies the names of one or more columns to exclude from the result. All matching column names are omitted from the output.

#### Syntax:

```
SELECT <expr> EXCEPT ( col_name1 [, col_name2, col_name3, ...] ) FROM [db.]table_name
```

### Example:

```
SELECT * EXCEPT (i) from columns_transformers;
```

j	k
10	324
8	23

## REPLACE

Specifies one or more **expression aliases**. Each alias must match a column name from the `SELECT *` statement. In the output column list, the column that matches the alias is replaced by the expression in that `REPLACE`.

This modifier does not change the names or order of columns. However, it can change the value and the value type.

### Syntax:

```
SELECT <expr> REPLACE( <expr> AS col_name) from [db.]table_name
```

### Example:

```
SELECT * REPLACE(i + 1 AS i) from columns_transformers;
```

i	j	k
101	10	324
121	8	23

## Modifier Combinations

You can use each modifier separately or combine them.

### Examples:

Using the same modifier multiple times.

```
SELECT COLUMNS('ijk') APPLY(toString) APPLY(length) APPLY(max) from columns_transformers;
```

max(length(toString(j)))	max(length(toString(k)))
2	3

Using multiple modifiers in a single query.

```
SELECT * REPLACE(i + 1 AS i) EXCEPT (j) APPLY(sum) from columns_transformers;
```

sum(plus(i, 1))	sum(k)
222	347

## SETTINGS in SELECT Query

You can specify the necessary settings right in the `SELECT` query. The setting value is applied only to this query and is reset to default or previous value after the query is executed.

Other ways to make settings see [here](#).

### Example

```
SELECT * FROM some_table SETTINGS optimize_read_in_order=1, cast_keep_nullable=1;
```

## ALL Clause

If there are multiple matching rows in the table, then `ALL` returns all of them. `SELECT ALL` is identical to `SELECT` without `DISTINCT`. If both `ALL` and `DISTINCT` specified, exception will be thrown.

`ALL` can also be specified inside aggregate function with the same effect(noop), for instance:

```
SELECT sum(ALL number) FROM numbers(10);
```

equals to

```
SELECT sum(number) FROM numbers(10);
```

## ARRAY JOIN Clause

It is a common operation for tables that contain an array column to produce a new table that has a column with each individual array element of that initial column, while values of other columns are duplicated. This is the basic case of what `ARRAY JOIN` clause does.

Its name comes from the fact that it can be looked at as executing `JOIN` with an array or nested data structure. The intent is similar to the [arrayJoin](#) function, but the clause functionality is broader.

Syntax:

```
SELECT <expr_list>
FROM <left_subquery>
[LEFT] ARRAY JOIN <array>
[WHERE|PREWHERE <expr>]
...
...
```

You can specify only one `ARRAY JOIN` clause in a `SELECT` query.

Supported types of `ARRAY JOIN` are listed below:

- `ARRAY JOIN` - In base case, empty arrays are not included in the result of `JOIN`.
- `LEFT ARRAY JOIN` - The result of `JOIN` contains rows with empty arrays. The value for an empty array is set to the default value for the array element type (usually 0, empty string or NULL).

# Basic ARRAY JOIN Examples

The examples below demonstrate the usage of the `ARRAY JOIN` and `LEFT ARRAY JOIN` clauses. Let's create a table with an `Array` type column and insert values into it:

```
CREATE TABLE arrays_test
(
    s String,
    arr Array(UInt8)
) ENGINE = Memory;

INSERT INTO arrays_test
VALUES ('Hello', [1,2]), ('World', [3,4,5]), ('Goodbye', []);
```

s	arr
Hello	[1,2]
World	[3,4,5]
Goodbye	[]

The example below uses the `ARRAY JOIN` clause:

```
SELECT s, arr
FROM arrays_test
ARRAY JOIN arr;
```

s	arr
Hello	1
Hello	2
World	3
World	4
World	5

The next example uses the `LEFT ARRAY JOIN` clause:

```
SELECT s, arr
FROM arrays_test
LEFT ARRAY JOIN arr;
```

s	arr
Hello	1
Hello	2
World	3
World	4
World	5
Goodbye	0

## Using Aliases

An alias can be specified for an array in the `ARRAY JOIN` clause. In this case, an array item can be accessed by this alias, but the array itself is accessed by the original name. Example:

```
SELECT s, arr, a
FROM arrays_test
ARRAY JOIN arr AS a;
```

s	arr	a
Hello	[1,2]	1
Hello	[1,2]	2
World	[3,4,5]	3
World	[3,4,5]	4
World	[3,4,5]	5

Using aliases, you can perform `ARRAY JOIN` with an external array. For example:

```
SELECT s, arr_external
FROM arrays_test
ARRAY JOIN [1, 2, 3] AS arr_external;
```

s	arr_external
Hello	1
Hello	2
Hello	3
World	1
World	2
World	3
Goodbye	1
Goodbye	2
Goodbye	3

Multiple arrays can be comma-separated in the `ARRAY JOIN` clause. In this case, `JOIN` is performed with them simultaneously (the direct sum, not the cartesian product). Note that all the arrays must have the same size. Example:

```
SELECT s, arr, a, num, mapped
FROM arrays_test
ARRAY JOIN arr AS a, arrayEnumerate(arr) AS num, arrayMap(x -> x + 1, arr) AS mapped;
```

s	arr	a	num	mapped
Hello	[1,2]	1	1	2
Hello	[1,2]	2	2	3
World	[3,4,5]	3	1	4
World	[3,4,5]	4	2	5
World	[3,4,5]	5	3	6

The example below uses the `arrayEnumerate` function:

```
SELECT s, arr, a, num, arrayEnumerate(arr)
FROM arrays_test
ARRAY JOIN arr AS a, arrayEnumerate(arr) AS num;
```

s	arr	a	num	arrayEnumerate(arr)
Hello	[1,2]	1	1	[1,2]
Hello	[1,2]	2	2	[1,2]
World	[3,4,5]	3	1	[1,2,3]
World	[3,4,5]	4	2	[1,2,3]
World	[3,4,5]	5	3	[1,2,3]

## ARRAY JOIN with Nested Data Structure

ARRAY JOIN also works with `nested data structures`:

```

CREATE TABLE nested_test
(
    s String,
    nest Nested(
        x UInt8,
        y UInt32)
) ENGINE = Memory;

INSERT INTO nested_test
VALUES ('Hello', [1,2], [10,20]), ('World', [3,4,5], [30,40,50]), ('Goodbye', [], []);

```

s	nest.x	nest.y
Hello	[1,2]	[10,20]
World	[3,4,5]	[30,40,50]
Goodbye	[]	[]

```

SELECT s, `nest.x`, `nest.y`
FROM nested_test
ARRAY JOIN nest;

```

s	nest.x	nest.y
Hello	1	10
Hello	2	20
World	3	30
World	4	40
World	5	50

When specifying names of nested data structures in ARRAY JOIN, the meaning is the same as ARRAY JOIN with all the array elements that it consists of. Examples are listed below:

```

SELECT s, `nest.x`, `nest.y`
FROM nested_test
ARRAY JOIN `nest.x`, `nest.y`;

```

s	nest.x	nest.y
Hello	1	10
Hello	2	20
World	3	30
World	4	40
World	5	50

This variation also makes sense:

```

SELECT s, `nest.x`, `nest.y`
FROM nested_test
ARRAY JOIN `nest.x`;

```

s	nest.x	nest.y
Hello	1	[10,20]
Hello	2	[10,20]
World	3	[30,40,50]
World	4	[30,40,50]
World	5	[30,40,50]

An alias may be used for a nested data structure, in order to select either the `JOIN` result or the source array. Example:

```
SELECT s, `n.x`, `n.y`, `nest.x`, `nest.y`
FROM nested_test
ARRAY JOIN nest AS n;
```

s	n.x	n.y	nest.x	nest.y
Hello	1	10	[1,2]	[10,20]
Hello	2	20	[1,2]	[10,20]
World	3	30	[3,4,5]	[30,40,50]
World	4	40	[3,4,5]	[30,40,50]
World	5	50	[3,4,5]	[30,40,50]

Example of using the `arrayEnumerate` function:

```
SELECT s, `n.x`, `n.y`, `nest.x`, `nest.y`, num
FROM nested_test
ARRAY JOIN nest AS n, arrayEnumerate(`nest.x`) AS num;
```

s	n.x	n.y	nest.x	nest.y	num
Hello	1	10	[1,2]	[10,20]	1
Hello	2	20	[1,2]	[10,20]	2
World	3	30	[3,4,5]	[30,40,50]	1
World	4	40	[3,4,5]	[30,40,50]	2
World	5	50	[3,4,5]	[30,40,50]	3

## Implementation Details

The query execution order is optimized when running `ARRAY JOIN`. Although `ARRAY JOIN` must always be specified before the `WHERE/PREWHERE` clause in a query, technically they can be performed in any order, unless result of `ARRAY JOIN` is used for filtering. The processing order is controlled by the query optimizer.

## DISTINCT Clause

If `SELECT DISTINCT` is specified, only unique rows will remain in a query result. Thus only a single row will remain out of all the sets of fully matching rows in the result.

You can specify the list of columns that must have unique values: `SELECT DISTINCT ON (column1, column2,...)`. If the columns are not specified, all of them are taken into consideration.

Consider the table:

a	b	c
1	1	1
1	1	1
2	2	2
2	2	2
1	1	2
1	2	2

Using `DISTINCT` without specifying columns:

```
SELECT DISTINCT * FROM t1;
```

a	b	c
1	1	1
2	2	2
1	1	2
1	2	2

Using `DISTINCT` with specified columns:

```
SELECT DISTINCT ON (a,b) * FROM t1;
```

a	b	c
1	1	1
2	2	2
1	2	2

## DISTINCT and ORDER BY

ClickHouse supports using the `DISTINCT` and `ORDER BY` clauses for different columns in one query. The `DISTINCT` clause is executed before the `ORDER BY` clause.

Consider the table:

a	b
2	1
1	2
3	3
2	4

Selecting data:

```
SELECT DISTINCT a FROM t1 ORDER BY b ASC;
```

a
2
1
3

Selecting data with the different sorting direction:

```
SELECT DISTINCT a FROM t1 ORDER BY b DESC;
```

a
3
1
2

Row 2, 4 was cut before sorting.

Take this implementation specificity into account when programming queries.

## Null Processing

`DISTINCT` works with `NULL` as if `NULL` were a specific value, and `NULL==NULL`. In other words, in the `DISTINCT` results, different combinations with `NULL` occur only once. It differs from `NULL` processing in most other contexts.

## Alternatives

It is possible to obtain the same result by applying `GROUP BY` across the same set of values as specified as `SELECT` clause, without using any aggregate functions. But there are few differences from `GROUP BY` approach:

- `DISTINCT` can be applied together with `GROUP BY`.
- When `ORDER BY` is omitted and `LIMIT` is defined, the query stops running immediately after the required number of different rows has been read.
- Data blocks are output as they are processed, without waiting for the entire query to finish running.

---

## FORMAT Clause

ClickHouse supports a wide range of [serialization formats](#) that can be used on query results among other things. There are multiple ways to choose a format for `SELECT` output, one of them is to specify `FORMAT` format at the end of query to get resulting data in any specific format.

Specific format might be used either for convenience, integration with other systems or performance gain.

### Default Format

If the `FORMAT` clause is omitted, the default format is used, which depends on both the settings and the interface used for accessing the ClickHouse server. For the [HTTP interface](#) and the [command-line client](#) in batch mode, the default format is `TabSeparated`. For the command-line client in interactive mode, the default format is `PrettyCompact` (it produces compact human-readable tables).

### Implementation Details

When using the command-line client, data is always passed over the network in an internal efficient format (`Native`). The client independently interprets the `FORMAT` clause of the query and formats the data itself (thus relieving the network and the server from the extra load).

---

## FROM Clause

The `FROM` clause specifies the source to read data from:

- [Table](#)
- [Subquery](#)
- [Table function](#)

`JOIN` and `ARRAY JOIN` clauses may also be used to extend the functionality of the `FROM` clause.

`Subquery` is another `SELECT` query that may be specified in parenthesis inside `FROM` clause.

`FROM` clause can contain multiple data sources, separated by commas, which is equivalent of performing `CROSS JOIN` on them.

## FINAL Modifier

When `FINAL` is specified, ClickHouse fully merges the data before returning the result and thus performs all data transformations that happen during merges for the given table engine.

It is applicable when selecting data from tables that use the `MergeTree`-engine family. Also supported for:

- `Replicated` versions of `MergeTree` engines.
- `View`, `Buffer`, `Distributed`, and `MaterializedView` engines that operate over other engines, provided they were created over `MergeTree`-engine tables.

Now `SELECT` queries with `FINAL` are executed in parallel and slightly faster. But there are drawbacks (see below). The `max_final_threads` setting limits the number of threads used.

## Drawbacks

Queries that use `FINAL` are executed slightly slower than similar queries that do not, because:

- Data is merged during query execution.
- Queries with `FINAL` read primary key columns in addition to the columns specified in the query.

**In most cases, avoid using `FINAL`.** The common approach is to use different queries that assume the background processes of the `MergeTree` engine have't happened yet and deal with it by applying aggregation (for example, to discard duplicates).

## Implementation Details

If the `FROM` clause is omitted, data will be read from the `system.one` table.

The `system.one` table contains exactly one row (this table fulfills the same purpose as the `DUAL` table found in other DBMSs).

To execute a query, all the columns listed in the query are extracted from the appropriate table. Any columns not needed for the external query are thrown out of the subqueries.

If a query does not list any columns (for example, `SELECT count() FROM t`), some column is extracted from the table anyway (the smallest one is preferred), in order to calculate the number of rows.

## GROUP BY Clause

`GROUP BY` clause switches the `SELECT` query into an aggregation mode, which works as follows:

- `GROUP BY` clause contains a list of expressions (or a single expression, which is considered to be the list of length one). This list acts as a “grouping key”, while each individual expression will be referred to as a “key expression”.
- All the expressions in the `SELECT`, `HAVING`, and `ORDER BY` clauses **must** be calculated based on key expressions **or** on `aggregate functions` over non-key expressions (including plain columns). In other words, each column selected from the table must be used either in a key expression or inside an aggregate function, but not both.
- Result of aggregating `SELECT` query will contain as many rows as there were unique values of “grouping key” in source table. Usually this significantly reduces the row count, often by orders of magnitude, but not necessarily: row count stays the same if all “grouping key” values were distinct.

When you want to group data in the table by column numbers instead of column names, enable the setting `enable_positional_arguments`.

### Note

There's an additional way to run aggregation over a table. If a query contains table columns only inside aggregate functions, the `GROUP BY` clause can be omitted, and aggregation by an empty set of keys is assumed. Such queries always return exactly one row.

## NULL Processing

For grouping, ClickHouse interprets `NULL` as a value, and `NULL==NULL`. It differs from `NULL` processing in most other contexts.

Here's an example to show what this means.

Assume you have this table:

x	y
1	2
2	NULL
3	2
3	3
3	NULL

The query `SELECT sum(x), y FROM t_null_big GROUP BY y` results in:

sum(x)	y
4	2
3	3
5	NULL

You can see that `GROUP BY` for `y = NULL` summed up `x`, as if `NULL` is this value.

If you pass several keys to `GROUP BY`, the result will give you all the combinations of the selection, as if `NULL` were a specific value.

## WITH ROLLUP Modifier

`WITH ROLLUP` modifier is used to calculate subtotals for the key expressions, based on their order in the `GROUP BY` list. The subtotals rows are added after the result table.

The subtotals are calculated in the reverse order: at first subtotals are calculated for the last key expression in the list, then for the previous one, and so on up to the first key expression.

In the subtotals rows the values of already "grouped" key expressions are set to 0 or empty line.

### Note

Mind that `HAVING` clause can affect the subtotals results.

### Example

Consider the table `t`:

year	month	day
2019	1	5
2019	1	15
2020	1	5
2020	1	15
2020	10	5
2020	10	15

Query:

```
SELECT year, month, day, count(*) FROM t GROUP BY year, month, day WITH ROLLUP;
```

As `GROUP BY` section has three key expressions, the result contains four tables with subtotals "rolled up" from right to left:

- `GROUP BY year, month, day`;
- `GROUP BY year, month` (and `day` column is filled with zeros);
- `GROUP BY year` (now `month, day` columns are both filled with zeros);
- and totals (and all three key expression columns are zeros).

year	month	day	count()
2020	10	15	1
2020	1	5	1
2019	1	5	1
2020	1	15	1
2019	1	15	1
2020	10	5	1

year	month	day	count()
2019	1	0	2
2020	1	0	2
2020	10	0	2

year	month	day	count()
2019	0	0	2
2020	0	0	4

year	month	day	count()
0	0	0	6

## WITH CUBE Modifier

`WITH CUBE` modifier is used to calculate subtotals for every combination of the key expressions in the `GROUP BY` list. The subtotals rows are added after the result table.

In the subtotals rows the values of all "grouped" key expressions are set to 0 or empty line.

### Note

Mind that **HAVING** clause can affect the subtotals results.

### Example

Consider the table t:

year	month	day
2019	1	5
2019	1	15
2020	1	5
2020	1	15
2020	10	5
2020	10	15

Query:

```
SELECT year, month, day, count(*) FROM t GROUP BY year, month, day WITH CUBE;
```

As `GROUP BY` section has three key expressions, the result contains eight tables with subtotals for all key expression combinations:

- `GROUP BY year, month, day`
- `GROUP BY year, month`
- `GROUP BY year, day`
- `GROUP BY year`
- `GROUP BY month, day`
- `GROUP BY month`
- `GROUP BY day`
- and totals.

Columns, excluded from `GROUP BY`, are filled with zeros.

year	month	day	count()
2020	10	15	1
2020	1	5	1
2019	1	5	1
2020	1	15	1
2019	1	15	1
2020	10	5	1

year	month	day	count()
2019	1	0	2
2020	1	0	2
2020	10	0	2

year	month	day	count()
2020	0	5	2
2019	0	5	1
2020	0	15	2
2019	0	15	1

year	month	day	count()
2019	0	0	2
2020	0	0	4

year	month	day	count()
0	1	5	2
0	10	15	1
0	10	5	1
0	1	15	2

year	month	day	count()
0	1	0	4
0	10	0	2

year	month	day	count()
0	0	5	3
0	0	15	3

year	month	day	count()
0	0	0	6

## WITH TOTALS Modifier

If the `WITH TOTALS` modifier is specified, another row will be calculated. This row will have key columns containing default values (zeros or empty lines), and columns of aggregate functions with the values calculated across all the rows (the “total” values).

This extra row is only produced in `JSON*`, `TabSeparated*`, and `Pretty*` formats, separately from the other rows:

- In `JSON*` formats, this row is output as a separate ‘totals’ field.
- In `TabSeparated*` formats, the row comes after the main result, preceded by an empty row (after the other data).
- In `Pretty*` formats, the row is output as a separate table after the main result.
- In the other formats it is not available.

`WITH TOTALS` can be run in different ways when `HAVING` is present. The behavior depends on the `totals_mode` setting.

## Configuring Totals Processing

By default, `totals_mode = 'before_having'`. In this case, ‘totals’ is calculated across all rows, including the ones that do not pass through `HAVING` and `max_rows_to_group_by`.

The other alternatives include only the rows that pass through `HAVING` in ‘totals’, and behave differently with the setting `max_rows_to_group_by` and `group_by_overflow_mode = 'any'`.

`after_having_exclusive` – Don't include rows that didn't pass through `max_rows_to_group_by`. In other words, 'totals' will have less than or the same number of rows as it would if `max_rows_to_group_by` were omitted.

`after_having_inclusive` – Include all the rows that didn't pass through 'max\_rows\_to\_group\_by' in 'totals'. In other words, 'totals' will have more than or the same number of rows as it would if `max_rows_to_group_by` were omitted.

`after_having_auto` – Count the number of rows that passed through HAVING. If it is more than a certain amount (by default, 50%), include all the rows that didn't pass through 'max\_rows\_to\_group\_by' in 'totals'. Otherwise, do not include them.

`totals_auto_threshold` – By default, 0.5. The coefficient for `after_having_auto`.

If `max_rows_to_group_by` and `group_by_overflow_mode = 'any'` are not used, all variations of `after_having` are the same, and you can use any of them (for example, `after_having_auto`).

You can use `WITH TOTALS` in subqueries, including subqueries in the `JOIN` clause (in this case, the respective total values are combined).

## Examples

Example:

```
SELECT
    count(),
    median(FetchTiming > 60 ? 60 : FetchTiming),
    count() - sum(Refresh)
FROM hits
```

As opposed to MySQL (and conforming to standard SQL), you can't get some value of some column that is not in a key or aggregate function (except constant expressions). To work around this, you can use the 'any' aggregate function (get the first encountered value) or 'min/max'.

Example:

```
SELECT
    domainWithoutWWW(URL) AS domain,
    count(),
    any>Title) AS title -- getting the first occurred page header for each domain.
FROM hits
GROUP BY domain
```

For every different key value encountered, `GROUP BY` calculates a set of aggregate function values.

## Implementation Details

Aggregation is one of the most important features of a column-oriented DBMS, and thus it's implementation is one of the most heavily optimized parts of ClickHouse. By default, aggregation is done in memory using a hash-table. It has 40+ specializations that are chosen automatically depending on "grouping key" data types.

## GROUP BY Optimization Depending on Table Sorting Key

The aggregation can be performed more effectively, if a table is sorted by some key, and `GROUP BY` expression contains at least prefix of sorting key or injective functions. In this case when a new key is read from table, the in-between result of aggregation can be finalized and sent to client. This behaviour is switched on by the `optimize_aggregation_in_order` setting. Such optimization reduces memory usage during aggregation, but in some cases may slow down the query execution.

## GROUP BY in External Memory

You can enable dumping temporary data to the disk to restrict memory usage during GROUP BY. The `max_bytes_before_external_group_by` setting determines the threshold RAM consumption for dumping GROUP BY temporary data to the file system. If set to 0 (the default), it is disabled.

When using `max_bytes_before_external_group_by`, we recommend that you set `max_memory_usage` about twice as high. This is necessary because there are two stages to aggregation: reading the data and forming intermediate data (1) and merging the intermediate data (2). Dumping data to the file system can only occur during stage 1. If the temporary data wasn't dumped, then stage 2 might require up to the same amount of memory as in stage 1.

For example, if `max_memory_usage` was set to 10000000000 and you want to use external aggregation, it makes sense to set `max_bytes_before_external_group_by` to 10000000000, and `max_memory_usage` to 20000000000. When external aggregation is triggered (if there was at least one dump of temporary data), maximum consumption of RAM is only slightly more than `max_bytes_before_external_group_by`.

With distributed query processing, external aggregation is performed on remote servers. In order for the requester server to use only a small amount of RAM, set `distributed_aggregation_memory_efficient` to 1.

When merging data flushed to the disk, as well as when merging results from remote servers when the `distributed_aggregation_memory_efficient` setting is enabled, consumes up to  $1/256 * \text{the\_number\_of\_threads}$  from the total amount of RAM.

When external aggregation is enabled, if there was less than `max_bytes_before_external_group_by` of data (i.e. data was not flushed), the query runs just as fast as without external aggregation. If any temporary data was flushed, the run time will be several times longer (approximately three times).

If you have an ORDER BY with a LIMIT after GROUP BY, then the amount of used RAM depends on the amount of data in LIMIT, not in the whole table. But if the ORDER BY does not have LIMIT, do not forget to enable external sorting (`max_bytes_before_external_sort`).

## HAVING Clause

Allows filtering the aggregation results produced by GROUP BY. It is similar to the WHERE clause, but the difference is that WHERE is performed before aggregation, while HAVING is performed after it.

It is possible to reference aggregation results from SELECT clause in HAVING clause by their alias. Alternatively, HAVING clause can filter on results of additional aggregates that are not returned in query results.

## Limitations

HAVING can't be used if aggregation is not performed. Use WHERE instead.

## INTO OUTFILE Clause

Add the INTO OUTFILE filename clause (where filename is a string literal) to SELECT query to redirect its output to the specified file on the client-side.

## Implementation Details

- This functionality is available in the command-line client and clickhouse-local. Thus a query sent via HTTP interface will fail.
- The query will fail if a file with the same filename already exists.

- The default **output format** is **TabSeparated** (like in the command-line client batch mode).

# JOIN Clause

Join produces a new table by combining columns from one or multiple tables by using values common to each. It is a common operation in databases with SQL support, which corresponds to **relational algebra** join. The special case of one table join is often referred to as “self-join”.

## Syntax

```
SELECT <expr_list>
FROM <left_table>
[GLOBAL] [INNER|LEFT|RIGHT|FULL|CROSS] [OUTER|SEMI|ANTI|ANY|ASOF] JOIN <right_table>
(ON <expr_list>)|(USING <column_list>) ...
```

Expressions from **ON** clause and columns from **USING** clause are called “join keys”. Unless otherwise stated, join produces a **Cartesian product** from rows with matching “join keys”, which might produce results with much more rows than the source tables.

## Supported Types of JOIN

All standard **SQL JOIN** types are supported:

- INNER JOIN**, only matching rows are returned.
- LEFT OUTER JOIN**, non-matching rows from left table are returned in addition to matching rows.
- RIGHT OUTER JOIN**, non-matching rows from right table are returned in addition to matching rows.
- FULL OUTER JOIN**, non-matching rows from both tables are returned in addition to matching rows.
- CROSS JOIN**, produces cartesian product of whole tables, “join keys” are **not** specified.

**JOIN** without specified type implies **INNER**. Keyword **OUTER** can be safely omitted. Alternative syntax for **CROSS JOIN** is specifying multiple tables in **FROM clause** separated by commas.

Additional join types available in ClickHouse:

- LEFT SEMI JOIN** and **RIGHT SEMI JOIN**, a whitelist on “join keys”, without producing a cartesian product.
- LEFT ANTI JOIN** and **RIGHT ANTI JOIN**, a blacklist on “join keys”, without producing a cartesian product.
- LEFT ANY JOIN**, **RIGHT ANY JOIN** and **INNER ANY JOIN**, partially (for opposite side of **LEFT** and **RIGHT**) or completely (for **INNER** and **FULL**) disables the cartesian product for standard **JOIN** types.
- ASOF JOIN** and **LEFT ASOF JOIN**, joining sequences with a non-exact match. **ASOF JOIN** usage is described below.

## Note

When **join\_algorithm** is set to **partial\_merge**, **RIGHT JOIN** and **FULL JOIN** are supported only with **ALL** strictness (**SEMI**, **ANTI**, **ANY**, and **ASOF** are not supported).

## Settings

The default join type can be overridden using **join\_default\_strictness** setting.

The behavior of ClickHouse server for ANY JOIN operations depends on the [any\\_join\\_distinct\\_right\\_table\\_keys](#) setting.

## See also

- [join\\_algorithm](#)
- [join\\_any\\_take\\_last\\_row](#)
- [join\\_use\\_nulls](#)
- [partial\\_merge\\_join\\_optimizations](#)
- [partial\\_merge\\_join\\_rows\\_in\\_right\\_blocks](#)
- [join\\_on\\_disk\\_max\\_files\\_to\\_merge](#)
- [any\\_join\\_distinct\\_right\\_table\\_keys](#)

## ON Section Conditions

An ON section can contain several conditions combined using the AND operator. Conditions specifying join keys must refer both left and right tables and must use the equality operator. Other conditions may use other logical operators but they must refer either the left or the right table of a query.

Rows are joined if the whole complex condition is met. If the conditions are not met, still rows may be included in the result depending on the JOIN type. Note that if the same conditions are placed in a WHERE section and they are not met, then rows are always filtered out from the result.

### Note

The OR operator inside an ON section is not supported yet.

### Note

If a condition refers columns from different tables, then only the equality operator (=) is supported so far.

## Example

Consider table\_1 and table\_2:

Id	name	
1	A	
2	B	
3	C	

Id	text	scores
1	Text A	10
1	Another text A	12
2	Text B	15

Query with one join key condition and an additional condition for table\_2:

```
SELECT name, text FROM table_1 LEFT OUTER JOIN table_2  
    ON table_1.Id = table_2.Id AND startsWith(table_2.text, 'Text');
```

Note that the result contains the row with the name C and the empty text column. It is included into the result because an OUTER type of a join is used.

name	text
A	Text A
B	Text B
C	

Query with `INNER` type of a join and multiple conditions:

```
SELECT name, text, scores FROM table_1 INNER JOIN table_2
    ON table_1.id = table_2.id AND table_2.scores > 10 AND startsWith(table_2.text, 'Text');
```

Result:

name	text	scores
B	Text B	15

## ASOF JOIN Usage

`ASOF JOIN` is useful when you need to join records that have no exact match.

Algorithm requires the special column in tables. This column:

- Must contain an ordered sequence.
- Can be one of the following types: `Int`, `UInt`, `Float`, `Date`, `DateTime`, `Decimal`.
- Can't be the only column in the `JOIN` clause.

Syntax `ASOF JOIN ... ON`:

```
SELECT expressions_list
FROM table_1
ASOF LEFT JOIN table_2
ON equi_cond AND closest_match_cond
```

You can use any number of equality conditions and exactly one closest match condition. For example,

```
SELECT count() FROM table_1 ASOF LEFT JOIN table_2 ON table_1.a == table_2.b AND table_2.t <= table_1.t
```

Conditions supported for the closest match: `>`, `>=`, `<`, `<=`.

Syntax `ASOF JOIN ... USING`:

```
SELECT expressions_list
FROM table_1
ASOF JOIN table_2
USING (equi_column1, ... equi_columnN, asof_column)
```

`ASOF JOIN` uses `equi_columnX` for joining on equality and `asof_column` for joining on the closest match with the `table_1.asof_column >= table_2.asof_column` condition. The `asof_column` column is always the last one in the `USING` clause.

For example, consider the following tables:

table_1			table_2		
event	ev_time	user_id	event	ev_time	user_id
event_1_1	12:00	42	event_2_1	11:59	42
...	...	...	event_2_2	12:30	42
event_1_2	13:00	42	event_2_3	13:00	42
...	...	...	...	...	...

ASOF JOIN can take the timestamp of a user event from `table_1` and find an event in `table_2` where the timestamp is closest to the timestamp of the event from `table_1` corresponding to the closest match condition. Equal timestamp values are the closest if available. Here, the `user_id` column can be used for joining on equality and the `ev_time` column can be used for joining on the closest match. In our example, `event_1_1` can be joined with `event_2_1` and `event_1_2` can be joined with `event_2_3`, but `event_2_2` can't be joined.

## Note

ASOF join is **not** supported in the **Join** table engine.

## Distributed JOIN

There are two ways to execute join involving distributed tables:

- When using a normal `JOIN`, the query is sent to remote servers. Subqueries are run on each of them in order to make the right table, and the join is performed with this table. In other words, the right table is formed on each server separately.
- When using `GLOBAL ... JOIN`, first the requestor server runs a subquery to calculate the right table. This temporary table is passed to each remote server, and queries are run on them using the temporary data that was transmitted.

Be careful when using `GLOBAL`. For more information, see the [Distributed subqueries](#) section.

## Implicit Type Conversion

`INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`, and `FULL JOIN` queries support the implicit type conversion for "join keys". However the query can not be executed, if join keys from the left and the right tables cannot be converted to a single type (for example, there is no data type that can hold all values from both `UInt64` and `Int64`, or `String` and `Int32`).

### Example

Consider the table `t_1`:

a	b	toTypeName(a)	toTypeName(b)
1	1	UInt16	UInt8
2	2	UInt16	UInt8

and the table `t_2`:

a	b	toTypeName(a)	toTypeName(b)
-1	1	Int16	Nullable(Int64)
1	-1	Int16	Nullable(Int64)
1	1	Int16	Nullable(Int64)

The query

```
SELECT a, b, toTypeName(a), toTypeName(b) FROM t_1 FULL JOIN t_2 USING (a, b);
```

returns the set:

a	b	toTypeName(a)	toTypeName(b)
1	1	Int32	Nullable(Int64)
2	2	Int32	Nullable(Int64)
-1	1	Int32	Nullable(Int64)
1	-1	Int32	Nullable(Int64)

## Usage Recommendations

### Processing of Empty or NULL Cells

While joining tables, the empty cells may appear. The setting `join_use_nulls` define how ClickHouse fills these cells.

If the `JOIN` keys are `Nullable` fields, the rows where at least one of the keys has the value `NULL` are not joined.

### Syntax

The columns specified in `USING` must have the same names in both subqueries, and the other columns must be named differently. You can use aliases to change the names of columns in subqueries.

The `USING` clause specifies one or more columns to join, which establishes the equality of these columns. The list of columns is set without brackets. More complex join conditions are not supported.

### Syntax Limitations

For multiple `JOIN` clauses in a single `SELECT` query:

- Taking all the columns via `*` is available only if tables are joined, not subqueries.
- The `PREWHERE` clause is not available.

For `ON`, `WHERE`, and `GROUP BY` clauses:

- Arbitrary expressions cannot be used in `ON`, `WHERE`, and `GROUP BY` clauses, but you can define an expression in a `SELECT` clause and then use it in these clauses via an alias.

### Performance

When running a `JOIN`, there is no optimization of the order of execution in relation to other stages of the query. The join (a search in the right table) is run before filtering in `WHERE` and before aggregation.

Each time a query is run with the same `JOIN`, the subquery is run again because the result is not cached. To avoid this, use the special `Join` table engine, which is a prepared array for joining that is always in RAM.

In some cases, it is more efficient to use `IN` instead of `JOIN`.

If you need a `JOIN` for joining with dimension tables (these are relatively small tables that contain dimension properties, such as names for advertising campaigns), a `JOIN` might not be very convenient due to the fact that the right table is re-accessed for every query. For such cases, there is an “external dictionaries” feature that you should use instead of `JOIN`. For more information, see the [External dictionaries](#) section.

## Memory Limitations

By default, ClickHouse uses the `hash join` algorithm. ClickHouse takes the `right_table` and creates a hash table for it in RAM. If `join_algorithm = 'auto'` is enabled, then after some threshold of memory consumption, ClickHouse falls back to `merge` join algorithm. For `JOIN` algorithms description see the [join\\_algorithm](#) setting.

If you need to restrict `JOIN` operation memory consumption use the following settings:

- `max_rows_in_join` — Limits number of rows in the hash table.
- `max_bytes_in_join` — Limits size of the hash table.

When any of these limits is reached, ClickHouse acts as the [join\\_overflow\\_mode](#) setting instructs.

## Examples

Example:

```
SELECT
    CounterID,
    hits,
    visits
FROM
(
    SELECT
        CounterID,
        count() AS hits
    FROM test.hits
    GROUP BY CounterID
) ANY LEFT JOIN
(
    SELECT
        CounterID,
        sum(Sign) AS visits
    FROM test.visits
    GROUP BY CounterID
) USING CounterID
ORDER BY hits DESC
LIMIT 10
```

CounterID	hits	visits
1143050	523264	13665
731962	475698	102716
722545	337212	108187
722889	252197	10547
2237260	196036	9522
23057320	147211	7689
722818	90109	17847
48221	85379	4652
19762435	77807	7026
722884	77492	11056

## LIMIT Clause

`LIMIT m` allows to select the first `m` rows from the result.

`LIMIT n, m` allows to select the `m` rows from the result after skipping the first `n` rows. The `LIMIT m OFFSET n` syntax is equivalent.

`n` and `m` must be non-negative integers.

If there is no **ORDER BY** clause that explicitly sorts results, the choice of rows for the result may be arbitrary and non-deterministic.

## Note

The number of rows in the result set can also depend on the **limit** setting.

## LIMIT ... WITH TIES Modifier

When you set **WITH TIES** modifier for **LIMIT n[,m]** and specify **ORDER BY expr\_list**, you will get in result first **n** or **n,m** rows and all rows with same **ORDER BY** fields values equal to row at position **n** for **LIMIT n** and **m** for **LIMIT n,m**.

This modifier also can be combined with **ORDER BY ... WITH FILL** modifier.

For example, the following query

```
SELECT * FROM (
    SELECT number%50 AS n FROM numbers(100)
) ORDER BY n LIMIT 0,5
```

returns

n
0
0
1
1
2

but after apply **WITH TIES** modifier

```
SELECT * FROM (
    SELECT number%50 AS n FROM numbers(100)
) ORDER BY n LIMIT 0,5 WITH TIES
```

it returns another rows set

n
0
0
1
1
2
2

cause row number 6 have same value "2" for field n as row number 5

## LIMIT BY Clause

A query with the **LIMIT n BY expressions** clause selects the first **n** rows for each distinct value of **expressions**. The key for **LIMIT BY** can contain any number of **expressions**.

ClickHouse supports the following syntax variants:

- `LIMIT [offset_value, ]n BY expressions`
- `LIMIT n OFFSET offset_value BY expressions`

During query processing, ClickHouse selects data ordered by sorting key. The sorting key is set explicitly using an `ORDER BY` clause or implicitly as a property of the table engine. Then ClickHouse applies `LIMIT n BY expressions` and returns the first `n` rows for each distinct combination of `expressions`. If `OFFSET` is specified, then for each data block that belongs to a distinct combination of `expressions`, ClickHouse skips `offset_value` number of rows from the beginning of the block and returns a maximum of `n` rows as a result. If `offset_value` is bigger than the number of rows in the data block, ClickHouse returns zero rows from the block.

## Note

`LIMIT BY` is not related to `LIMIT`. They can both be used in the same query.

If you want to use column numbers instead of column names in the `LIMIT BY` clause, enable the setting `enable_positional_arguments`.

## Examples

Sample table:

```
CREATE TABLE limit_by(id Int, val Int) ENGINE = Memory;
INSERT INTO limit_by VALUES (1, 10), (1, 11), (1, 12), (2, 20), (2, 21);
```

Queries:

```
SELECT * FROM limit_by ORDER BY id, val LIMIT 2 BY id
```

id	val
1	10
1	11
2	20
2	21

```
SELECT * FROM limit_by ORDER BY id, val LIMIT 1, 2 BY id
```

id	val
1	11
1	12
2	21

The `SELECT * FROM limit_by ORDER BY id, val LIMIT 2 OFFSET 1 BY id` query returns the same result.

The following query returns the top 5 referrers for each `domain, device_type` pair with a maximum of 100 rows in total (`LIMIT n BY + LIMIT`).

```
SELECT
    domainWithoutWWW(URL) AS domain,
    domainWithoutWWW(REFERRER_URL) AS referrer,
    device_type,
    count() cnt
FROM hits
GROUP BY domain, referrer, device_type
ORDER BY cnt DESC
LIMIT 5 BY domain, device_type
LIMIT 100
```

## OFFSET FETCH Clause

`OFFSET` and `FETCH` allow you to retrieve data by portions. They specify a row block which you want to get by a single query.

```
OFFSET offset_row_count {ROW | ROWS} [FETCH {FIRST | NEXT} fetch_row_count {ROW | ROWS} {ONLY | WITH TIES}]
```

The `offset_row_count` or `fetch_row_count` value can be a number or a literal constant. You can omit `fetch_row_count`; by default, it equals to 1.

`OFFSET` specifies the number of rows to skip before starting to return rows from the query result set.

The `FETCH` specifies the maximum number of rows that can be in the result of a query.

The `ONLY` option is used to return rows that immediately follow the rows omitted by the `OFFSET`. In this case the `FETCH` is an alternative to the `LIMIT` clause. For example, the following query

```
SELECT * FROM test_fetch ORDER BY a OFFSET 1 ROW FETCH FIRST 3 ROWS ONLY;
```

is identical to the query

```
SELECT * FROM test_fetch ORDER BY a LIMIT 3 OFFSET 1;
```

The `WITH TIES` option is used to return any additional rows that tie for the last place in the result set according to the `ORDER BY` clause. For example, if `fetch_row_count` is set to 5 but two additional rows match the values of the `ORDER BY` columns in the fifth row, the result set will contain seven rows.

### Note

According to the standard, the `OFFSET` clause must come before the `FETCH` clause if both are present.

### Note

The real offset can also depend on the `offset` setting.

## Examples

Input table:

a	b
1	1
2	1
3	4
1	3
5	4
0	6
5	7

Usage of the `ONLY` option:

```
SELECT * FROM test_fetch ORDER BY a OFFSET 3 ROW FETCH FIRST 3 ROWS ONLY;
```

Result:

a	b
2	1
3	4
5	4

Usage of the `WITH TIES` option:

```
SELECT * FROM test_fetch ORDER BY a OFFSET 3 ROW FETCH FIRST 3 ROWS WITH TIES;
```

Result:

a	b
2	1
3	4
5	4
5	7

## ORDER BY Clause

The `ORDER BY` clause contains a list of expressions, which can each be attributed with `DESC` (descending) or `ASC` (ascending) modifier which determine the sorting direction. If the direction is not specified, `ASC` is assumed, so it's usually omitted. The sorting direction applies to a single expression, not to the entire list.  
Example: `ORDER BY Visits DESC, SearchPhrase`.

If you want to sort by column numbers instead of column names, enable the setting `enable_positional_arguments`.

Rows that have identical values for the list of sorting expressions are output in an arbitrary order, which can also be non-deterministic (different each time).

If the `ORDER BY` clause is omitted, the order of the rows is also undefined, and may be non-deterministic as well.

## Sorting of Special Values

There are two approaches to `Nan` and `NULL` sorting order:

- By default or with the `NULLS LAST` modifier: first the values, then `Nan`, then `NULL`.
- With the `NULLS FIRST` modifier: first `NULL`, then `Nan`, then other values.

## Example

For the table

x	y
1	NULL
2	2
1	nan
2	2
3	4
5	6
6	nan
7	NULL
6	7
8	9

Run the query `SELECT * FROM t_null_nan ORDER BY y NULLS FIRST` to get:

x	y
1	NULL
7	NULL
1	nan
6	nan
2	2
2	2
3	4
5	6
6	7
8	9

When floating point numbers are sorted, NaNs are separate from the other values. Regardless of the sorting order, NaNs come at the end. In other words, for ascending sorting they are placed as if they are larger than all the other numbers, while for descending sorting they are placed as if they are smaller than the rest.

## Collation Support

For sorting by **String** values, you can specify collation (comparison). Example: `ORDER BY SearchPhrase COLLATE 'tr'` - for sorting by keyword in ascending order, using the Turkish alphabet, case insensitive, assuming that strings are UTF-8 encoded. `COLLATE` can be specified or not for each expression in `ORDER BY` independently. If `ASC` or `DESC` is specified, `COLLATE` is specified after it. When using `COLLATE`, sorting is always case-insensitive.

Collate is supported in **LowCardinality**, **Nullable**, **Array** and **Tuple**.

We only recommend using `COLLATE` for final sorting of a small number of rows, since sorting with `COLLATE` is less efficient than normal sorting by bytes.

## Collation Examples

Example only with **String** values:

Input table:

X	S
1	bca
2	ABC
3	123a
4	abc
5	BCA

Query:

```
SELECT * FROM collate_test ORDER BY s ASC COLLATE 'en';
```

Result:

X	S
3	123a
4	abc
2	ABC
1	bca
5	BCA

Example with **Nullable**:

Input table:

X	S
1	bca
2	NULL
3	ABC
4	123a
5	abc
6	NULL
7	BCA

Query:

```
SELECT * FROM collate_test ORDER BY s ASC COLLATE 'en';
```

Result:

X	S
4	123a
5	abc
3	ABC
1	bca
7	BCA
6	NULL
2	NULL

Example with **Array**:

Input table:

X	S
1	['Z']
2	['z']
3	['a']
4	['A']
5	['z','a']
6	['z','a','a']
7	[]

Query:

```
SELECT * FROM collate_test ORDER BY s ASC COLLATE 'en';
```

Result:

X	S
7	[]
3	['a']
4	['A']
2	['z']
5	['z','a']
6	['z','a','a']
1	['Z']

Example with **LowCardinality** string:

Input table:

X	S
1	Z
2	z
3	a
4	A
5	za
6	zaa
7	

Query:

```
SELECT * FROM collate_test ORDER BY s ASC COLLATE 'en';
```

Result:

X	S
7	
3	a
4	A
2	z
1	Z
5	za
6	zaa

Example with **Tuple**:

X	S
1	(1,'Z')
2	(1,'z')
3	(1,'a')
4	(2,'z')
5	(1,'A')
6	(2,'Z')
7	(2,'A')

Query:

```
SELECT * FROM collate_test ORDER BY s ASC COLLATE 'en';
```

Result:

X	S
3	(1,'a')
5	(1,'A')
2	(1,'z')
1	(1,'Z')
7	(2,'A')
4	(2,'z')
6	(2,'Z')

## Implementation Details

Less RAM is used if a small enough **LIMIT** is specified in addition to **ORDER BY**. Otherwise, the amount of memory spent is proportional to the volume of data for sorting. For distributed query processing, if **GROUP BY** is omitted, sorting is partially done on remote servers, and the results are merged on the requestor server. This means that for distributed sorting, the volume of data to sort can be greater than the amount of memory on a single server.

If there is not enough RAM, it is possible to perform sorting in external memory (creating temporary files on a disk). Use the setting `max_bytes_before_external_sort` for this purpose. If it is set to 0 (the default), external sorting is disabled. If it is enabled, when the volume of data to sort reaches the specified number of bytes, the collected data is sorted and dumped into a temporary file. After all data is read, all the sorted files are merged and the results are output. Files are written to the `/var/lib/clickhouse/tmp/` directory in the config (by default, but you can use the `tmp_path` parameter to change this setting).

Running a query may use more memory than `max_bytes_before_external_sort`. For this reason, this setting must have a value significantly smaller than `max_memory_usage`. As an example, if your server has 128 GB of RAM and you need to run a single query, set `max_memory_usage` to 100 GB, and `max_bytes_before_external_sort` to 80 GB.

External sorting works much less effectively than sorting in RAM.

## Optimization of Data Reading

If **ORDER BY** expression has a prefix that coincides with the table sorting key, you can optimize the query by using the **optimize\_read\_in\_order** setting.

When the `optimize_read_in_order` setting is enabled, the ClickHouse server uses the table index and reads the data in order of the **ORDER BY** key. This allows to avoid reading all data in case of specified **LIMIT**. So queries on big data with small limit are processed faster.

Optimization works with both **ASC** and **DESC** and does not work together with **GROUP BY** clause and **FINAL** modifier.

When the `optimize_read_in_order` setting is disabled, the ClickHouse server does not use the table index while processing `SELECT` queries.

Consider disabling `optimize_read_in_order` manually, when running queries that have `ORDER BY` clause, large `LIMIT` and `WHERE` condition that requires to read huge amount of records before queried data is found.

Optimization is supported in the following table engines:

- `MergeTree`
- `Merge`, `Buffer`, and `MaterializedView` table engines over `MergeTree`-engine tables

In `MaterializedView`-engine tables the optimization works with views like `SELECT ... FROM merge_tree_table ORDER BY pk`. But it is not supported in the queries like `SELECT ... FROM view ORDER BY pk` if the view query does not have the `ORDER BY` clause.

## ORDER BY Expr WITH FILL Modifier

This modifier also can be combined with `LIMIT ... WITH TIES` modifier.

`WITH FILL` modifier can be set after `ORDER BY expr` with optional `FROM expr`, `TO expr` and `STEP expr` parameters. All missed values of `expr` column will be filled sequentially and other columns will be filled as defaults.

To fill multiple columns, add `WITH FILL` modifier with optional parameters after each field name in `ORDER BY` section.

```
ORDER BY expr [WITH FILL] [FROM const_expr] [TO const_expr] [STEP const_numeric_expr], ... exprN [WITH FILL]
[FROM expr] [TO expr] [STEP numeric_expr]
```

`WITH FILL` can be applied for fields with Numeric (all kinds of float, decimal, int) or Date/DateTime types.

When applied for String fields, missed values are filled with empty strings.

When `FROM const_expr` not defined sequence of filling use minimal `expr` field value from `ORDER BY`.

When `TO const_expr` not defined sequence of filling use maximum `expr` field value from `ORDER BY`.

When `STEP const_numeric_expr` defined then `const_numeric_expr` interprets as is for numeric types as days for Date type and as seconds for DateTime type.

When `STEP const_numeric_expr` omitted then sequence of filling use 1.0 for numeric type, 1 day for Date type and 1 second for DateTime type.

Example of a query without `WITH FILL`:

```
SELECT n, source FROM (
    SELECT toFloat32(number % 10) AS n, 'original' AS source
    FROM numbers(10) WHERE number % 3 = 1
) ORDER BY n;
```

Result:

n	source
1	original
4	original
7	original

Same query after applying `WITH FILL` modifier:

```

SELECT n, source FROM (
    SELECT toFloat32(number % 10) AS n, 'original' AS source
    FROM numbers(10) WHERE number % 3 = 1
) ORDER BY n WITH FILL FROM 0 TO 5.51 STEP 0.5;

```

Result:

n	source
0	
0.5	
1	original
1.5	
2	
2.5	
3	
3.5	
4	original
4.5	
5	
5.5	
7	original

For the case with multiple fields `ORDER BY field2 WITH FILL, field1 WITH FILL` order of filling will follow the order of fields in the `ORDER BY` clause.

Example:

```

SELECT
    toDate((number * 10) * 86400) AS d1,
    toDate(number * 86400) AS d2,
    'original' AS source
FROM numbers(10)
WHERE (number % 3) = 1
ORDER BY
    d2 WITH FILL,
    d1 WITH FILL STEP 5;

```

Result:

d1	d2	source
1970-01-11	1970-01-02	original
1970-01-01	1970-01-03	
1970-01-01	1970-01-04	
1970-02-10	1970-01-05	original
1970-01-01	1970-01-06	
1970-01-01	1970-01-07	
1970-03-12	1970-01-08	original

Field `d1` does not fill in and use the default value cause we do not have repeated values for `d2` value, and the sequence for `d1` can't be properly calculated.

The following query with the changed field in `ORDER BY`:

```

SELECT
    toDate((number * 10) * 86400) AS d1,
    toDate(number * 86400) AS d2,
    'original' AS source
FROM numbers(10)
WHERE (number % 3) = 1
ORDER BY
    d1 WITH FILL STEP 5,
    d2 WITH FILL;

```

Result:

d1	d2	source
1970-01-11	1970-01-02	original
1970-01-16	1970-01-01	
1970-01-21	1970-01-01	
1970-01-26	1970-01-01	
1970-01-31	1970-01-01	
1970-02-05	1970-01-01	
1970-02-10	1970-01-05	original
1970-02-15	1970-01-01	
1970-02-20	1970-01-01	
1970-02-25	1970-01-01	
1970-03-02	1970-01-01	
1970-03-07	1970-01-01	
1970-03-12	1970-01-08	original

## PREWHERE Clause

Prewhere is an optimization to apply filtering more efficiently. It is enabled by default even if `PREWHERE` clause is not specified explicitly. It works by automatically moving part of `WHERE` condition to prewhere stage. The role of `PREWHERE` clause is only to control this optimization if you think that you know how to do it better than it happens by default.

With prewhere optimization, at first only the columns necessary for executing prewhere expression are read. Then the other columns are read that are needed for running the rest of the query, but only those blocks where the prewhere expression is `true` at least for some rows. If there are a lot of blocks where prewhere expression is `false` for all rows and prewhere needs less columns than other parts of query, this often allows to read a lot less data from disk for query execution.

## Controlling Prewhere Manually

The clause has the same meaning as the `WHERE` clause. The difference is in which data is read from the table. When manually controlling `PREWHERE` for filtration conditions that are used by a minority of the columns in the query, but that provide strong data filtration. This reduces the volume of data to read.

A query may simultaneously specify `PREWHERE` and `WHERE`. In this case, `PREWHERE` precedes `WHERE`.

If the `optimize_move_to_prewhere` setting is set to 0, heuristics to automatically move parts of expressions from `WHERE` to `PREWHERE` are disabled.

If query has `FINAL` modifier, the `PREWHERE` optimization is not always correct. It is enabled only if both settings `optimize_move_to_prewhere` and `optimize_move_to_prewhere_if_final` are turned on.

### Attention

The `PREWHERE` section is executed before `FINAL`, so the results of `FROM ... FINAL` queries may be skewed when using `PREWHERE` with fields not in the `ORDER BY` section of a table.

## Limitations

`PREWHERE` is only supported by tables from the `*MergeTree` family.

## SAMPLE Clause

The `SAMPLE` clause allows for approximated `SELECT` query processing.

When data sampling is enabled, the query is not performed on all the data, but only on a certain fraction of data (sample). For example, if you need to calculate statistics for all the visits, it is enough to execute the query on the 1/10 fraction of all the visits and then multiply the result by 10.

Approximated query processing can be useful in the following cases:

- When you have strict latency requirements (like below 100ms) but you can't justify the cost of additional hardware resources to meet them.
- When your raw data is not accurate, so approximation does not noticeably degrade the quality.
- Business requirements target approximate results (for cost-effectiveness, or to market exact results to premium users).

## Note

You can only use sampling with the tables in the **MergeTree** family, and only if the sampling expression was specified during table creation (see **MergeTree engine**).

The features of data sampling are listed below:

- Data sampling is a deterministic mechanism. The result of the same `SELECT .. SAMPLE` query is always the same.
- Sampling works consistently for different tables. For tables with a single sampling key, a sample with the same coefficient always selects the same subset of possible data. For example, a sample of user IDs takes rows with the same subset of all the possible user IDs from different tables. This means that you can use the sample in subqueries in the `IN` clause. Also, you can join samples using the `JOIN` clause.
- Sampling allows reading less data from a disk. Note that you must specify the sampling key correctly. For more information, see [Creating a MergeTree Table](#).

For the `SAMPLE` clause the following syntax is supported:

SAMPLE Clause Syntax	Description
<code>SAMPLE k</code>	Here <code>k</code> is the number from 0 to 1.

The query is executed on `k` fraction of data. For example, `SAMPLE 0.1` runs the query on 10% of data. [Read more](#) `SAMPLE n` Here `n` is a sufficiently large integer. The query is executed on a sample of at least `n` rows (but not significantly more than this). For example, `SAMPLE 10000000` runs the query on a minimum of 10,000,000 rows. [Read more](#) `SAMPLE k OFFSET m` Here `k` and `m` are the numbers from 0 to 1. The query is executed on a sample of `k` fraction of the data. The data used for the sample is offset by `m` fraction. [Read more](#)

## SAMPLE K

Here `k` is the number from 0 to 1 (both fractional and decimal notations are supported). For example, `SAMPLE 1/2` or `SAMPLE 0.5`.

In a `SAMPLE k` clause, the sample is taken from the `k` fraction of data. The example is shown below:

```
SELECT
    Title,
    count() * 10 AS PageViews
FROM hits_distributed
SAMPLE 0.1
WHERE
    CounterID = 34
GROUP BY Title
ORDER BY PageViews DESC LIMIT 1000
```

In this example, the query is executed on a sample from 0.1 (10%) of data. Values of aggregate functions are not corrected automatically, so to get an approximate result, the value `count()` is manually multiplied by 10.

## SAMPLE N

Here `n` is a sufficiently large integer. For example, `SAMPLE 10000000`.

In this case, the query is executed on a sample of at least `n` rows (but not significantly more than this). For example, `SAMPLE 10000000` runs the query on a minimum of 10,000,000 rows.

Since the minimum unit for data reading is one granule (its size is set by the `index_granularity` setting), it makes sense to set a sample that is much larger than the size of the granule.

When using the `SAMPLE n` clause, you do not know which relative percent of data was processed. So you do not know the coefficient the aggregate functions should be multiplied by. Use the `_sample_factor` virtual column to get the approximate result.

The `_sample_factor` column contains relative coefficients that are calculated dynamically. This column is created automatically when you `create` a table with the specified sampling key. The usage examples of the `_sample_factor` column are shown below.

Let's consider the table `visits`, which contains the statistics about site visits. The first example shows how to calculate the number of page views:

```
SELECT sum(PageViews * _sample_factor)
FROM visits
SAMPLE 10000000
```

The next example shows how to calculate the total number of visits:

```
SELECT sum(_sample_factor)
FROM visits
SAMPLE 10000000
```

The example below shows how to calculate the average session duration. Note that you do not need to use the relative coefficient to calculate the average values.

```
SELECT avg(Duration)
FROM visits
SAMPLE 10000000
```

## SAMPLE K OFFSET M

Here `k` and `m` are numbers from 0 to 1. Examples are shown below.

### Example 1

```
SAMPLE 1/10
```

In this example, the sample is 1/10th of all data:

```
[++-----]
```

### Example 2

```
SAMPLE 1/10 OFFSET 1/2
```

Here, a sample of 10% is taken from the second half of the data.

```
[-----+-----]
```

## WHERE Clause

`WHERE` clause allows to filter the data that is coming from `FROM` clause of `SELECT`.

If there is a `WHERE` clause, it must contain an expression with the `UInt8` type. This is usually an expression with comparison and logical operators. Rows where this expression evaluates to 0 are excluded from further transformations or result.

`WHERE` expression is evaluated on the ability to use indexes and partition pruning, if the underlying table engine supports that.

### Note

There's a filtering optimization called `prewhere`.

## WITH Clause

ClickHouse supports Common Table Expressions (`CTE`), that provides to use results of `WITH` clause in the rest of `SELECT` query. Named subqueries can be included to the current and child query context in places where table objects are allowed. Recursion is prevented by hiding the current level CTEs from the `WITH` expression.

## Syntax

```
WITH <expression> AS <identifier>
```

or

```
WITH <identifier> AS <subquery expression>
```

## Examples

**Example 1:** Using constant expression as “variable”

```
WITH '2019-08-01 15:23:00' as ts_upper_bound
SELECT *
FROM hits
WHERE
    EventDate = toDate(ts_upper_bound) AND
    EventTime <= ts_upper_bound;
```

### Example 2: Evicting a sum(bytes) expression result from the SELECT clause column list

```
WITH sum(bytes) as s
SELECT
    formatReadableSize(s),
    table
FROM system.parts
GROUP BY table
ORDER BY s;
```

### Example 3: Using results of a scalar subquery

```
/* this example would return TOP 10 of most huge tables */
WITH
(
    SELECT sum(bytes)
    FROM system.parts
    WHERE active
) AS total_disk_usage
SELECT
    (sum(bytes) / total_disk_usage) * 100 AS table_disk_usage,
    table
FROM system.parts
GROUP BY table
ORDER BY table_disk_usage DESC
LIMIT 10;
```

### Example 4: Reusing expression in a subquery

```
WITH test1 AS (SELECT i + 1, j + 1 FROM test1)
SELECT * FROM test1;
```

../../../../en/sql-reference/statements/select/union.md

## クエリの作成

### CREATE DATABASE

データベースの作成

```
CREATE DATABASE [IF NOT EXISTS] db_name [ON CLUSTER cluster] [ENGINE = engine(...)]
```

#### 句

##### ■ IF NOT EXISTS

もし `db_name` データベースが既に存在する場合、ClickHouseは新しいデータベースを作成せず、：

- 句が指定されている場合は例外をスローしません。
- 句が指定されていない場合、例外をスローします。

#### ■ ON CLUSTER

クリックハウスは `db_name` データベースの全てのサーバーの指定されたクラスター

#### ■ ENGINE

##### ■ MySQL

リモートMySQLサーバーからデータを取得できます。

既定では、ClickHouseは独自のものを使用します [データベース](#)。

## CREATE TABLE

その `CREATE TABLE` クエリには複数の形式があります。

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1] [compression_codec] [TTL expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2] [compression_codec] [TTL expr2],
    ...
) ENGINE = engine
```

テーブルを作成します ‘name’ で ‘db’ データベース ‘db’ が設定されていない。‘engine’ エンジン  
テーブルの構造は、列の説明のリストです。た場合の指標については、エンジンとして表示していパラメータテーブル  
のエンジンです。

列の説明は次のとおりです `name type` 最も単純なケースでは。例: `RegionID UInt32`.  
デフォルト値に対して式を定義することもできます(下記参照)。

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name AS [db2.]name2 [ENGINE = engine]
```

別のテーブルと同じ構造を持つテーブルを作成します。テーブルに別のエンジンを指定できます。エンジンが指定され  
ていない場合、同じエンジンが使用されます。`db2.name2` テーブル。

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name AS table_function()
```

テーブルを作成しますの構造やデータによって返される [テーブル関数](#).

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name ENGINE = engine AS SELECT ...
```

の結果のような構造体を持つテーブルを作成します。`SELECT` クエリは、‘engine’ エンジン充填でのデータを選択しま  
す。

すべての場合において、`IF NOT EXISTS` テーブルが既に存在する場合、クエリはエラーを返しません。この場合、クエ  
リは何もしません。

の後に他の句がある場合もあります。`ENGINE` クエリ内の句。テーブルの作成方法に関する詳細なドキュメントを参照  
してください [表エンジン](#).

## デフォルト値

列の説明では、次のいずれかの方法で既定値の式を指定できます:`DEFAULT expr`, `MATERIALIZED expr`, `ALIAS expr`.

例: `URLDomain String DEFAULT domain(URL)`.

既定値の式が定義されていない場合、既定値は数値の場合はゼロ、文字列の場合は空の文字列、配列の場合は空の配列、  
および `1970-01-01` 日付または `zero unix timestamp` 時間と日付のために。Nullはサポートされていません。

既定の式が定義されている場合、列の型は省略可能です。明示的に定義された型がない場合は、既定の式の型が使用さ  
れます。例: `EventDate DEFAULT toDate(EventTime)` - the ‘Date’ タイプは ‘EventDate’ 列。

データ型と既定の式が明示的に定義されている場合、この式は型キャスト関数を使用して指定された型にキャストされます。例: `Hits UInt32 DEFAULT 0`と同じことを意味します `Hits UInt32 DEFAULT toUInt32(0)`.

Default expressions may be defined as an arbitrary expression from table constants and columns. When creating and changing the table structure, it checks that expressions don't contain loops. For INSERT, it checks that expressions are resolvable – that all columns they can be calculated from have been passed.

#### DEFAULT expr

標準のデフォルト値。INSERTクエリで対応する列が指定されていない場合は、対応する式を計算して入力されます。

#### MATERIALIZED expr

マテリアライズ式。このような列は常に計算されるため、INSERTには指定できません。

列のリストがない挿入の場合、これらの列は考慮されません。

また、SELECTクエリでアスタリスクを使用する場合、この列は置換されません。これは、ダンプが取得した不变量を保持するためです `SELECT * 列のリストを指定せずにINSERT`を使用してテーブルに挿入することができます。

#### ALIAS expr

シノニム このような列は、テーブルにまったく格納されません。

また、SELECTクエリでアスタリスクを使用する場合は、その値は置換されません。

クエリの解析中にエイリアスが展開される場合は、Selectで使用できます。

ALTERクエリを使用して新しい列を追加する場合、これらの列の古いデータは書き込まれません。代わりに、新しい列の値を持たない古いデータを読み取るときは、デフォルトで式がその場で計算されます。ただし、式を実行するには、クエリで指定されていない異なる列が必要な場合は、これらの列はさらに読み取られますが、必要なデータブロックに対し

テーブルに新しい列を追加し、後でその既定の式を変更すると、古いデータに使用される値が変更されます（値がディスクに格納されていないデータの場合）。ご注意実行する場合背景が合併のデータ列を欠損の合流部に書かれて合併します。

入れ子になったデータ構造の要素に既定値を設定することはできません。

## 制約

列記述制約と一緒に定義することができます:

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1] [compression_codec] [TTL expr1],
    ...
    CONSTRAINT constraint_name_1 CHECK boolean_expr_1,
    ...
) ENGINE = engine
```

`boolean_expr_1` 任意の布尔式で可能です。場合に制約の定義のテーブルのそれぞれチェック毎に行 `INSERT query`. If any constraint is not satisfied — server will raise an exception with constraint name and checking expression.

追加大量の制約になる可能性の性能を大 `INSERT` クエリ。

## TTL式

値の保存時間を定義します。MergeTree-familyテーブルに対してのみ指定できます。詳細な説明については、[列および表のTTL](#)。

## 列圧縮コーデック

デフォルトでは、ClickHouseは `Lz4` 圧縮方法。のために MergeTree-エンジンファミリでデフォルトの圧縮方法を変更できます。圧縮 サーバー構成のセクション。また、圧縮方法を定義することもできます。CREATE TABLE クエリ。

```
CREATE TABLE codec_example
(
    dt Date CODEC(ZSTD),
    ts DateTime CODEC(LZ4HC),
    float_value Float32 CODEC(NONE),
    double_value Float64 CODEC(LZ4HC(9))
    value Float32 CODEC(Delta, ZSTD)
)
ENGINE = <Engine>
...
```

## 警告

できない解凍ClickHouseデータベースファイルを外部の事のように `Lz4`。代わりに、特別な **clickhouse-コンプレッサー** ユーティリティ

圧縮は、次の表エンジンでサポートされます:

- **メルゲツリー** 家族だ 支柱の圧縮コーデックとの選択のデフォルトの圧縮メソッドによる 圧縮 設定。
- **ログ** 家族だ を使用して `Lz4` 圧縮メソッドはデフォルト対応カラムの圧縮コーデック。
- **セット** 既定の圧縮のみをサポートしました。
- **参加** 既定の圧縮のみをサポートしました。

ClickHouse支援共通の目的コーデックや専門のコーデック。

## 特殊なコーデック

これらのコーデックしていただくための圧縮により効果的な利用の特徴データです。これらのコーデックの一部は、データ自体を圧縮しません。代わりに、共通の目的のコーデックのためにデータを準備し、この準備がない場合よりも圧縮します。

特殊なコーデック:

- **Delta(delta\_bytes)** — Compression approach in which raw values are replaced by the difference of two neighboring values, except for the first value that stays unchanged. Up to `delta_bytes` デルタ値を格納するために使用されます。`delta_bytes` 生の値の最大サイズです。可能 `delta_bytes` 値:1,2,4,8。のデフォルト値 `delta_bytes` は `sizeof(type)` 1、2、4、または8に等しい場合。他のすべてのケースでは、それは1です。
- **DoubleDelta** — Calculates delta of deltas and writes it in compact binary form. Optimal compression rates are achieved for monotonic sequences with a constant stride, such as time series data. Can be used with any fixed-width type. Implements the algorithm used in Gorilla TSDB, extending it to support 64-bit types. Uses 1 extra bit for 32-byte deltas: 5-bit prefixes instead of 4-bit prefixes. For additional information, see Compressing Time Stamps in **Gorilla:高速でスケーラブルなメモリ内の時系列データベース**.
- **Gorilla** — Calculates XOR between current and previous value and writes it in compact binary form. Efficient when storing a series of floating point values that change slowly, because the best compression rate is achieved when neighboring values are binary equal. Implements the algorithm used in Gorilla TSDB, extending it to support 64-bit types. For additional information, see Compressing Values in **Gorilla:高速でスケーラブルなメモリ内の時系列データベース**.

- **T64** — Compression approach that crops unused high bits of values in integer data types (including **Enum**, **Date** と **DateTime**). そのアルゴリズムの各ステップで、**codec**は64の値のブロックを取り、それらを64x64ビット行列に入れ、転置し、未使用の値のビットを切り取り、残りをシーウェイビットは、圧縮が使用されるデータ部分全体の最大値と最小値の間で異なるビットです。

**DoubleDelta** と **Gorilla Gorilla TSDB**では圧縮アルゴリズムの構成要素としてコーデックが使用されている。 **Gorilla**のアプローチは、タイムスタンプにゆっくりと変化する一連の値がある場合のシナリオで効果的です。 タイムスタンプは **DoubleDelta** によって効果的に圧縮されます **Gorilla** コーデック。 たとえば、効果的に格納されたテーブルを取得するには、次の構成で作成できます:

```
CREATE TABLE codec_example
(
    timestamp DateTime CODEC(DoubleDelta),
    slow_values Float32 CODEC(Gorilla)
)
ENGINE = MergeTree()
```

汎用コーデック

コーデック:

- **NONE** — No compression.
- **LZ4** — Lossless **データ圧縮** 既定で使用されます。 LZ4高速圧縮を適用します。
- **LZ4HC[(level)]** — LZ4 HC (high compression) algorithm with configurable level. Default level: 9. Setting **level <= 0** 既定のレベルを適用します。 可能なレベル : [1、12]。 推奨レベル範囲 : [4、9]。
- **ZSTD[(level)]** — **ZSTD圧縮アルゴリズム** 構成可能を使って **level**. 可能なレベル : [1、22]。 デフォルト値:1。

高い圧縮レベルは、一度の圧縮、繰り返しの解凍などの非対称シナリオに役立ちます。 高いレベルは、より良い圧縮と高いCPU使用率を意味します。

## 一時テーブル

**ClickHouse**は、次の特性を持つ一時テーブルをサポートします:

- 一時テーブルは、接続が失われた場合を含め、セッションが終了すると消滅します。
- 一時テーブルはメモリエンジンのみを使用します。
- 一時テーブルに**DB**を指定することはできません。 データベースの外部に作成されます。
- すべてのクラスタサーバーで分散**DDL**クエリを使用して一時テーブルを作成することはできません **ON CLUSTER**):このテーブルは現在のセッションにのみ存在します。
- 一時テーブルの名前が別のテーブルと同じで、クエリが**DB**を指定せずにテーブル名を指定した場合、一時テーブルが使用されます。
- 分散クエリ処理では、クエリで使用される一時テーブルがリモートサーバーに渡されます。

一時テーブルを作成するには、次の構文を使用します:

```
CREATE TEMPORARY TABLE [IF NOT EXISTS] table_name
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
)
```

ほとんどの場合、一時テーブルを手動で作成され、外部データを利用するためのクエリに対して、または配布 (GLOBAL) IN. 詳細については、該当する項を参照してください

テーブルを使用することは可能です エンジン=メモリ 一時テーブルの代わりに。

## 分散DDLクエリ(ON CLUSTER句)

その CREATE, DROP, ALTER, and RENAME クエリの支援の分散実行クラスター  
たとえば、次のクエリは、 all\_hits Distributed 各ホストのテーブル cluster:

```
CREATE TABLE IF NOT EXISTS all_hits ON CLUSTER cluster (p Date, i Int32) ENGINE = Distributed(cluster, default, hits)
```

これらのクエリを正しく実行するには、各ホストが同じクラスタ定義を持っている必要があります（設定の同期を簡単にするために、ZooKeeperからの置換 また、ZooKeeperサーバーに接続する必要があります。

一部のホストが現在利用できない場合でも、ローカルバージョンのクエリは最終的にクラスタ内の各ホストに実装されます。 単一のホスト内でクエリを実行する順序は保証されます。

## CREATE VIEW

```
CREATE [MATERIALIZED] VIEW [IF NOT EXISTS] [db.]table_name [TO[db.]name] [ENGINE = engine] [POPULATE] AS  
SELECT ...
```

ビューを作成します。 通常と実体化：ビューの二つのタイプがあります。

通常のビューはデータを格納せず、別のテーブルからの読み取りを実行するだけです。 つまり、通常のビューは保存されたクエリに過ぎません。 ビューから読み取るとき、この保存されたクエリはFROM句のサブクエリとして使用されます。

例として、ビューを作成したとします：

```
CREATE VIEW view AS SELECT ...
```

そして、クエリを書いた：

```
SELECT a, b, c FROM view
```

このクエリは、サブクエリの使用と完全に同等です：

```
SELECT a, b, c FROM (SELECT ...)
```

マテリアライズドビュー

マテリアライズドビューを作成するとき TO [db].[table], you must specify ENGINE – the table engine for storing data.

マテリアライズドビューを作成するとき TO [db].[table]、使用してはいけません POPULATE.

SELECTで指定されたテーブルにデータを挿入すると、挿入されたデータの一部がこのSELECTクエリによって変換され、結果がビューに挿入されます。

POPULATEを指定すると、既存のテーブルデータが作成時にビューに挿入されます。 CREATE TABLE ... AS SELECT .... そうしないと、クエリーを含み、データを挿入し、表の作成後、作成した。 ビューの作成中にテーブルに挿入されたデータは挿入されないため、POPULATEを使用することはお勧めしません。

A `SELECT` クエリを含むことができ `DISTINCT, GROUP BY, ORDER BY, LIMIT...` Note that the corresponding conversions are performed independently on each block of inserted data. For example, if `GROUP BY` データは挿入中に集計されますが、挿入されたデータの単一パケット内でのみ集計されます。データはそれ以上集計されません。例外は、次のように独立してデータ集計を実行するエンジンを使用する場合です `SummingMergeTree`.

の実行 `ALTER` マテリアライズドビューのクエリは完全に開発されていないため、不便な場合があります。マテリアライズドビューで構成を使用する場合 `TO [db.]name`、できます `DETACH` ビュー、実行 `ALTER` ターゲットテーブルの場合、次に `ATTACH` 以前に切り離された (`DETACH`) ビュー。

ビューの外観は通常のテーブルと同じです。たとえば、それらは次の結果にリストされます `SHOW TABLES` クエリ。

ビューを削除するための別のクエリはありません。ビューを削除するには `DROP TABLE`.

## CREATE DICTIONARY

```
CREATE DICTIONARY [IF NOT EXISTS] [db.]dictionary_name [ON CLUSTER cluster]
(
    key1 type1 [DEFAULT|EXPRESSION expr1] [HIERARCHICAL|INJECTIVE|IS_OBJECT_ID],
    key2 type2 [DEFAULT|EXPRESSION expr2] [HIERARCHICAL|INJECTIVE|IS_OBJECT_ID],
    attr1 type2 [DEFAULT|EXPRESSION expr3],
    attr2 type2 [DEFAULT|EXPRESSION expr4]
)
PRIMARY KEY key1, key2
SOURCE(SOURCE_NAME([param1 value1 ... paramN valueN]))
LAYOUT(LAYOUT_NAME([param_name param_value]))
LIFETIME({MIN min_val MAX max_val | max_val})
```

作成 [外部辞書](#) 与えられたと 構造, ソース, レイアウト と 生涯.

外部辞書構造は属性で構成されます。辞書属性は、表の列と同様に指定されます。必要な属性プロパティはその型だけで、他のすべてのプロパティには既定値があります。

辞書によっては [レイアウト](#) 一つ以上の属性を辞書キーとして指定できます。

詳細については、[外部辞書](#) セクション

## CREATE USER

を作成します。[ユーザー](#).

### 構文

```
CREATE USER [IF NOT EXISTS | OR REPLACE] name [ON CLUSTER cluster_name]
[IDENTIFIED [WITH
{NO_PASSWORD|PLAINTEXT_PASSWORD|SHA256_PASSWORD|SHA256_HASH|DOUBLE_SHA1_PASSWORD|DOUBLE_SH
A1_HASH}] BY {'password'|'hash'}]
[HOST {LOCAL | NAME 'name' | REGEXP 'name_regexp' | IP 'address' | LIKE 'pattern'} [...]| ANY | NONE]
[DEFAULT ROLE role [...]]
[SETTINGS variable [= value] [MIN [=] min_value] [MAX [=] max_value] [READONLY|WRITABLE] | PROFILE
'profile_name'] [...]
```

### 識別情報

ユーザー識別には複数の方法があります:

- `IDENTIFIED WITH no_password`
- `IDENTIFIED WITH plaintext_password BY 'qwerty'`
- `IDENTIFIED WITH sha256_password BY 'qwerty'` または `IDENTIFIED BY 'password'`
- `IDENTIFIED WITH sha256_hash BY 'hash'`

- IDENTIFIED WITH double\_sha1\_password BY 'qwertys'

- IDENTIFIED WITH double\_sha1\_hash BY 'hash'

## ユーザホスト

ユーザー ホストは、ClickHouse サーバーへの接続を確立できるホストです。ホストを指定することができます。HOST 次の方法によるクエリのセクション：

- HOST IP 'ip\_address\_or\_subnetwork' — User can connect to ClickHouse server only from the specified IP address or a サブネットワーク. 例: HOST IP '192.168.0.0/16', HOST IP '2001:DB8::/32'. の使用、生産を指定するだけでいいのです。HOST IP 要素 (IPアドレスとそのマスク) を使用しているため host と host\_regex が原因別の待ち時間をゼロにすることに
- HOST ANY — User can connect from any location. This is default option.
- HOST LOCAL — User can connect only locally.
- HOST NAME 'fqdn' — User host can be specified as FQDN. For example, HOST NAME 'mysite.com'.
- HOST NAME REGEXP 'regexp' — You can use pcre ユーザー ホストを指定するときの正規表現。例えば, HOST NAME REGEXP '.\*\mysite\.com'.
- HOST LIKE 'template' — Allows you use the LIKE ユーザー ホストをフィルタする演算子。例えば, HOST LIKE '%' に等しい。HOST ANY, HOST LIKE '%.mysite.com' すべてのホストをフィルタする。mysite.com ドメイン。

Hostを指定する別 の方法は次のとおりです @ ユーザー名の構文。例:

- CREATE USER mira@'127.0.0.1' — Equivalent to the HOST IP 構文。
- CREATE USER mira@'localhost' — Equivalent to the HOST LOCAL 構文。
- CREATE USER mira@'192.168.%.%' — Equivalent to the HOST LIKE 構文。

## 警告

クリックハウス user\_name@'address' 全体としてのユーザー名として。このように、技術的に作成できます複数のユーザー user\_name そして後の異なる構造 @. 私たちはそうすることをお勧めしません。

## 例

ユーザー アカウントの作成 `mira` パスワードで保護 `qwertys`:

```
CREATE USER mira HOST IP '127.0.0.1' IDENTIFIED WITH sha256_password BY 'qwertys'
```

`mira` 開始すべきお客様のアプリをホストにClickHouse サーバー運行しています。

ユーザー アカウントの作成 `john` ロールを割り当て、このロールをデフォルトにする:

```
CREATE USER john DEFAULT ROLE role1, role2
```

ユーザー アカウントの作成 `john` 彼の将来の役割をすべてデフォルトにする:

```
ALTER USER user DEFAULT ROLE ALL
```

いくつかの役割が割り当てられるとき `john` 将来的には自動的にデフォルトになります。

ユーザー アカウント の作成 `john` く 全ての 彼の 今後の 役割 は デフォルト を除く `role1` と `role2`:

```
ALTER USER john DEFAULT ROLE ALL EXCEPT role1, role2
```

## CREATE ROLE

を作成します。 役割.

### 構文

```
CREATE ROLE [IF NOT EXISTS | OR REPLACE] name  
[SETTINGS variable [= value] [MIN [=] min_value] [MAX [=] max_value] [READONLY|WRITABLE] | PROFILE  
'profile_name'] [...]
```

### 説明

役割は、特権. ロールで付与されたユーザーは、このロールのすべての権限を取得します。

ユーザーの割り当てることができ複数の役割です。ユーザーに適用できるのを助の役割の任意の組み合わせによる SET ROLE 声明。特権の最終的なスコープは、適用されたすべてのロールのすべての特権の組み合わせです。ユーザーが権限を付与に直接かつてのユーザー アカウント、または権限を付与する。

ユーザーがデフォルトの役割を応用したユーザーログインします。デフォルトのロールを設定するには、SET DEFAULT ROLE 文または ALTER USER 声明。

ロールを取り消すには、REVOKE 声明。

ロールを削除するには、DROP ROLE 声明。削除されたロールは、付与されたすべてのユーザーおよびロールから自動的に取り消されます。

### 例

```
CREATE ROLE accountant;  
GRANT SELECT ON db.* TO accountant;
```

この一連のクエリは、ロールを作成します `accountant` それはからデータを読み取る権限を持っています `accounting` データベース。

ユーザーへのロールの付与 `mira`:

```
GRANT accountant TO mira;
```

後の役割が付与されたユーザーが利用できる実行させます。例えば:

```
SET ROLE accountant;  
SELECT * FROM db.*;
```

## CREATE ROW POLICY

を作成します。行のフィルタ ユーザーがテーブルから読み取ることができます。

### 構文

```
CREATE [ROW] POLICY [IF NOT EXISTS | OR REPLACE] policy_name [ON CLUSTER cluster_name] ON [db.]table  
[AS {PERMISSIVE | RESTRICTIVE}]  
[FOR SELECT]  
[USING condition]  
[TO {role [...] | ALL | ALL EXCEPT role [...]}]
```

## セクションAS

このセクショ

確認方針にアクセスが付与されています。同じテーブルに適用されるパーミッシブポリシーは、booleanを使用して結合されます OR オペレーター ポリシーは既定では許可されています。

制限的な政策アクセス制限を行います。同じテーブルに適用される制限ポリシーは、ブール値を使用して結合されます AND オペレーター

制限ポリシーは、許可フィルターを通過した行に適用されます。制限的なポリシーを設定しても、許可的なポリシーを設定しない場合、ユーザーはテーブルから行を取得できません。

## セクションへ

セクション内 TO ロールとユーザーの混在リストを指定できます。, CREATE ROW POLICY ... TO accountant, john@localhost.

キーワード ALL 現在のユーザを含むすべてのClickHouseユーザを意味します。キーワード ALL EXCEPT allow toは、すべてのユーザーリストから一部のユーザーを除外します。CREATE ROW POLICY ... TO ALL EXCEPT accountant, john@localhost

## 例

- CREATE ROW POLICY filter ON mydb.mytable FOR SELECT USING a<1000 TO accountant, john@localhost
- CREATE ROW POLICY filter ON mydb.mytable FOR SELECT USING a<1000 TO ALL EXCEPT mira

## CREATE QUOTA

を作成します。 クオータ ユーザーまたはロールに割り当てることができます。

## 構文

```
CREATE QUOTA [IF NOT EXISTS | OR REPLACE] name [ON CLUSTER cluster_name]  
[KEYED BY {'none' | 'user name' | 'ip address' | 'client key' | 'client key or user name' | 'client key or ip address'}]  
[FOR [RANDOMIZED] INTERVAL number {SECOND | MINUTE | HOUR | DAY | WEEK | MONTH | QUARTER | YEAR}  
{MAX { {QUERIES | ERRORS | RESULT ROWS | RESULT BYTES | READ ROWS | READ BYTES | EXECUTION TIME} =  
number } [...] }  
NO LIMITS | TRACKING ONLY} [...] ]  
[TO {role [...] | ALL | ALL EXCEPT role [...]}]
```

## 例

現在のユーザーに対するクエリの最大数を123か月以内に制限する制約:

```
CREATE QUOTA qA FOR INTERVAL 15 MONTH MAX QUERIES 123 TO CURRENT_USER
```

## CREATE SETTINGS PROFILE

を作成します。 設定プロファイル ユーザーまたはロールに割り当てることができます。

## 構文

```
CREATE SETTINGS PROFILE [IF NOT EXISTS | OR REPLACE] name [ON CLUSTER cluster_name]
    [SETTINGS variable [= value] [MIN [=] min_value] [MAX [=] max_value] [READONLY|WRITABLE] | INHERIT
    'profile_name'] [...]
```

## 例

を作成します。 `max_memory_usage_profile` の値と制約を持つ設定プロファイル `max_memory_usage` 設定。それを割り当てる `robin`:

```
CREATE SETTINGS PROFILE max_memory_usage_profile SETTINGS max_memory_usage = 100000001 MIN 90000000
MAX 110000000 TO robin
```

## INSERT

データの追加。

基本的なクエリ形式:

```
INSERT INTO [db.]table [(c1, c2, c3)] VALUES (v11, v12, v13), (v21, v22, v23), ...
```

クエリでは、挿入する列のリストを指定できます `[(c1, c2, c3)]`. この場合、残りの列は:

- から計算される値 `DEFAULT` テーブル定義で指定された式。
- ゼロと空の文字列 `DEFAULT` 式は定義されていません。

もし `strict_insert_defaults=1`,を持たない列 `DEFAULT` 定義され記載されていることを返します。

データは、任意の場所で `INSERT` に渡すことができます **形式** ClickHouseがサポートしています。この形式は、クエリで明示的に指定する必要があります:

```
INSERT INTO [db.]table [(c1, c2, c3)] FORMAT format_name data_set
```

For example, the following query format is identical to the basic version of `INSERT ... VALUES`:

```
INSERT INTO [db.]table [(c1, c2, c3)] FORMAT Values (v11, v12, v13), (v21, v22, v23), ...
```

ClickHouseは、データの前にスペースと改行がある場合をすべて削除します。クエリを作成するときは、クエリ演算子の後に新しい行にデータを置くことをお勧めします（データがスペースで始まる場合は重要です）。

例:

```
INSERT INTO t FORMAT TabSeparated
11 Hello, world!
22 Qwerty
```

挿入することができます。データから別のクエリのコマンドラインクライアント、HTTPインターフェース。詳細については “[インター](#)”。

## 制約

テーブルが **制約**, their expressions will be checked for each row of inserted data. If any of those constraints is not satisfied — server will raise an exception containing constraint name and expression, the query will be stopped.

## の結果を挿入する SELECT

```
INSERT INTO [db.]table [(c1, c2, c3)] SELECT ...
```

列は、SELECT句内の位置に従ってマップされます。ただし、SELECT式とINSERTテーブルの名前は異なる場合があります。必要に応じて、型铸造が行われる。

値以外のデータ形式では、次のような式に値を設定できません `now()`, `1 + 2`、というように。Values形式では式の使用は制限されますが、この場合は非効率的なコードが実行に使用されるため、これは推奨されません。

その他のクエリをデータ部品に対応していない: UPDATE, DELETE, REPLACE, MERGE, UPSERT, INSERT UPDATE. ただし、古いデータを削除するには `ALTER TABLE ... DROP PARTITION`.

FORMAT 次の場合、クエリの最後に句を指定する必要があります `SELECT` 句は、表関数を含みます [入力\(\)](#).

## パフォーマンス

`INSERT` 入力データを主キーでソートし、パーティションキーでパーティションに分割します。た場合のデータを挿入し複数の仕切りは一度で大幅に低減できることの `INSERT` クエリ。これを避けるには:

- 一度に100,000行など、かなり大きなバッチでデータを追加します。
- グループによるデータのパーティション鍵のアップロード前にでClickHouse.

パフォーマンスが低下しない場合:

- データはリアルタイムで追加されます。
- アップロードしたデータとは、通常はソートされました。

## ALTER

その `ALTER` クエリーのみ対応して \*MergeTree テーブルだけでなく、MergeとDistributed. クエリに複数のバリエーションがあります。

### 列の操作

テーブル構造の変更。

```
ALTER TABLE [db].name [ON CLUSTER cluster] ADD|DROP|CLEAR|COMMENT|MODIFY COLUMN ...
```

クエリでは、コンマ区切りのアクションのリストを指定します。

各アクションは、列に対する操作です。

次の操作がサポートされます:

- ADD COLUMN** — Adds a new column to the table.
- DROP COLUMN** — Deletes the column.
- CLEAR COLUMN** — Resets column values.
- COMMENT COLUMN** — Adds a text comment to the column.
- MODIFY COLUMN** — Changes column's type, default expression and TTL.

これらの行動を詳細に説明します。

### ADD COLUMN

```
ADD COLUMN [IF NOT EXISTS] name [type] [default_expr] [codec] [AFTER name_after]
```

指定されたテーブルに新しい列を追加します `name`, `type`, `codec` と `default_expr` (節を参照 [既定の式](#)).

もし `IF NOT EXISTS` 列が既に存在する場合、クエリはエラーを返しません。指定した場合 `AFTER name_after` (別の列の名前)、列は表の列のリスト内で指定された列の後に追加されます。それ以外の場合は、列がテーブルの最後に追加されます。テーブルの先頭に列を追加する方法はないことに注意してください。一連の行動のために, `name_after` 前の操作のいずれかで追加された列の名前を指定できます。

列を追加すると、データでアクションを実行せずにテーブル構造が変更されます。その後、データはディスクに表示されません `ALTER`. テーブルから読み取るときに列のデータがない場合は、デフォルト値が入力されます (デフォルトの式がある場合はデフォルトの式を実行するか、ゼロ 列の表示のディスクと統合データ部品 ([メルゲツリー](#))).

このアプローチにより、`ALTER` 古いデータの量を増やすことなく、瞬時にクエリ。

例:

```
ALTER TABLE visits ADD COLUMN browser String AFTER user_id
```

## DROP COLUMN

```
DROP COLUMN [IF EXISTS] name
```

名前の列を削除します `name`. もし `IF EXISTS` 列が存在しない場合、クエリはエラーを返しません。

コンピュータのデータを削除するファイルシステム。この削除全ファイル、クエリーがほぼ完了します。

例:

```
ALTER TABLE visits DROP COLUMN browser
```

## CLEAR COLUMN

```
CLEAR COLUMN [IF EXISTS] name IN PARTITION partition_name
```

すべてリセットデータ列の指定されたパーティション セクションのパーティション名の設定の詳細を読む [パーティション式の指定方法](#).

もし `IF EXISTS` 列が存在しない場合、クエリはエラーを返しません。

例:

```
ALTER TABLE visits CLEAR COLUMN browser IN PARTITION tuple()
```

## COMMENT COLUMN

```
COMMENT COLUMN [IF EXISTS] name 'comment'
```

列にコメントを追加します。もし `IF EXISTS` 列が存在しない場合、クエリはエラーを返しません。

各列には一つのコメントがあります。列にコメントが既に存在する場合、新しいコメントは前のコメントを上書きします。

コメントは `comment_expression` によって返される列 [DESCRIBE TABLE](#) クエリ。

例:

```
ALTER TABLE visits COMMENT COLUMN browser 'The table shows the browser used for accessing the site.'
```

## MODIFY COLUMN

```
MODIFY COLUMN [IF EXISTS] name [type] [default_expr] [TTL]
```

このクエリは、`name` 列のプロパティ:

- タイプ
- 既定の式
- TTL

For examples of columns TTL modifying, see [Column TTL]  
(#sql-reference-engines-table\_engines-mergetree\_family-mergetree-md).

もし `IF EXISTS` 列が存在しない場合、クエリはエラーを返しません。

型を変更するとき、値は次のように変換されます **トタイプ** 機能をそれらに適用した。既定の式のみが変更された場合、クエリは複雑な処理を行わず、ほぼ即座に完了します。

例:

```
ALTER TABLE visits MODIFY COLUMN browser Array(String)
```

Changing the column type is the only complex action – it changes the contents of files with data. For large tables, this may take a long time.

いくつかの処理段階があります:

- 準備一時(新しいファイルが修正データです)。
- 古いファイルの名前を変更する。
- 一時(新しい)ファイルの名前を古い名前に変更します。
- 古いファイルを削除します。

最初の段階だけ時間がかかります。この段階で障害が発生した場合、データは変更されません。

連続するいずれかの段階で障害が発生した場合、データを手動で復元することができます。例外は、古いファイルがファイルシステムから削除されたが、新しいファイルのデータがディスクに書き込まれず、失われた場合です。

その `ALTER` 列を変更するクエリが複製されます。命令はZooKeeperに保存され、各レプリカはそれらを適用します。すべて `ALTER` クエリは同じ順序で実行されます。クエリは、他のレプリカで適切なアクションが完了するのを待機します。ただし、レプリケートされたテーブル内の列を変更するクエリは中断され、すべてのアクションが非同期に実行されます。

## クエリの制限の変更

その `ALTER` クエリを作成および削除個別要素（カラム）をネストしたデータ構造が全体に入れ子データ構造です。入れ子になったデータ構造を追加するには、次のような名前の列を追加できます `name.nested_name` そしてタイプ `Array(T)`。入れ子になったデータ構造は、ドットの前に同じプレフィックスを持つ名前を持つ複数の配列列と同等です。

主キーまたはサンプリングキーの列を削除することはサポートされていません。ENGINE 式)。主キーに含まれる列の型を変更することは、この変更によってデータが変更されない場合にのみ可能です(たとえば、列挙型に値を追加したり、型を変更 DateTime に UInt32)。

もし ALTER クエリでは、必要なテーブルの変更を行うのに十分ではありません。INSERT SELECT を使用してテーブルを切り替えます RENAME 古いテーブルを照会して削除します。を使用することができます クリックハウス-複写機 の代わりとして INSERT SELECT クエリ。

その ALTER クエリーのロックすべてを読み込みと書き込んでいます。言い換えれば、SELECT の時に実行されている ALTER クエリは、ALTER クエリは、それが完了するのを待ちます。同時に、同じテーブルに対するすべての新しいクエリは、この間待機します ALTER 走ってる

データ自体を格納しないテーブルの場合(以下のように Merge と Distributed), ALTER テーブル構造を変更するだけで、下位テーブルの構造は変更されません。たとえば、Distributed テーブル、あなたも実行する必要があります ALTER テーブルのすべてすることができます。

## キー式による操作

次のコマン:

```
MODIFY ORDER BY new_expression
```

これは、MergeTree 家族 (含む  
複製 テーブル)。このコマンドは  
ソートキー テーブルの  
に new\_expression (式または式のタプル)。主キーは同じままです。

このコマンドは、メタデータのみを変更するという意味で軽量です。データ部分のプロパティを保持するには  
行は並べ替えキー式で並べ替えられます既存の列を含む式は追加できません  
によって追加された列のみ ADD COLUMN 同じコマンド ALTER クエリ)。

## 操作データを飛指標

これは、\*MergeTree 家族 (含む  
複製 テーブル)。次の操作  
利用できます:

- ALTER TABLE [db].name ADD INDEX name expression TYPE type GRANULARITY value [FIRST|AFTER name]-付加価指数の説明をテーブルメタデータを指すものとします。
- ALTER TABLE [db].name DROP INDEX name -除去す指標の説明からテーブルメタデータを削除を行指数のファイルからディスク。

これらのコマンドは軽量でいるという意味においてのみ変化メタデータの削除ファイルです。  
また、それらは複製されます (ZooKeeperを介して索引メタデータを同期)。

## 制約による操作

詳細はこちら [制約](#)

制約は、次の構文を使用して追加または削除できます:

```
ALTER TABLE [db].name ADD CONSTRAINT constraint_name CHECK expression;
ALTER TABLE [db].name DROP CONSTRAINT constraint_name;
```

クエリに追加または削除約メタデータの制約からテーブルで、速やかに処理します。

制約チェック 実行されません 既存のデータが追加された場合。

変更後の内容の複製のテーブル放送への飼育係で適用されますその他のレプリカ。

## パーティションとパートの操作

次の操作は **仕切り** 利用できます:

- **DETACH PARTITION** – Moves a partition to the **detached** ディレクトリとそれを忘れ。
- **DROP PARTITION** – Deletes a partition.
- **ATTACH PART|PARTITION** – Adds a part or partition from the **detached** テーブルへのディレクトリ。
- **ATTACH PARTITION FROM** – Copies the data partition from one table to another and adds.
- **REPLACE PARTITION** - コピーするデータを仕切りからテーブルにも置き換え。
- **MOVE PARTITION TO TABLE**(#alter\_move\_to\_table-partition)-データ-パーティションのあるテーブルから別のテーブルに移動します。
- **CLEAR COLUMN IN PARTITION** - パーティション内の指定された列の値をリセットします。
- **CLEAR INDEX IN PARTITION** - リセットの指定された二次インデックス、パーティション
- **FREEZE PARTITION** – Creates a backup of a partition.
- **FETCH PARTITION** – Downloads a partition from another server.
- **MOVE PARTITION|PART** – Move partition/data part to another disk or volume.

### DETACH PARTITION

```
ALTER TABLE table_name DETACH PARTITION partition_expr
```

指定されたパーティションのすべてのデータを **detached** ディレクトリ。 サーバーのを忘れているのは、一戸建てのデータを分配していない場合は存在します。 サーバーはこのデータについて知りません。 **ATTACH** クエリ。

例:

```
ALTER TABLE visits DETACH PARTITION 201901
```

セクションでのパーティション式の設定について **パーティション式の指定方法**.

クエリが実行された後、データを使用して好きなことを行うことができます。 **detached directory — delete it from the file system, or just leave it.**

This query is replicated – it moves the data to the **detached** すべてのレプリカ上の このクエリは、リーダーレプリカでのみ実行できます。 する場合は、レプリカは、オーソドックスなアプローチを行う **SELECT** クエリを実行する **システムレプリカ** テーブル。 あるいは、**DETACH** クエリはすべてのレプリカ-すべてのレプリカ、例外をスロー以外のリーダーレプリカ。

### DROP PARTITION

```
ALTER TABLE table_name DROP PARTITION partition_expr
```

指定したパーティションを表から削除します。 このクエリのタグの仕切りとして休止または消去いたしますデータを完全に約10分です。

セクションでのパーティション式の設定について **パーティション式の指定方法**.

The query is replicated – it deletes data on all replicas.

## DROP DETACHED PARTITION|PART

```
ALTER TABLE table_name DROP DETACHED PARTITION|PART partition_expr
```

指定された部分または指定されたパーティションのすべての部分を `detached`.  
セクションでのパーティション式の設定の詳細 [パーティション式の指定方法](#).

## ATTACH PARTITION|PART

```
ALTER TABLE table_name ATTACH PARTITION|PART partition_expr
```

テーブルにデータを追加します。 `detached` ディレクトリ。 パーティション全体または別の部分のデータを追加することができます。 例:

```
ALTER TABLE visits ATTACH PARTITION 201901;  
ALTER TABLE visits ATTACH PART 201901_2_2_0;
```

セクションでのパーティション式の設定の詳細 [パーティション式の指定方法](#).

このクエリは複製されます。 のレプリカ-イニシエータチェックがあるか否かのデータを `detached` ディレクトリ。 データが存在する場合、クエリはその整合性を確認します。 すべてが正しい場合、クエリはデータをテーブルに追加します。 他のすべてのレプリカをダウンロードからデータのレプリカ-イニシエータです。

だから、データを置くことができます `detached` ディレクトリを一つのレプリカ上に置き、 `ALTER ... ATTACH` すべてのレプリカのテーブルに追加するクエリ。

## ATTACH PARTITION FROM

```
ALTER TABLE table2 ATTACH PARTITION partition_expr FROM table1
```

このクエリは、データパーティションを `table1` に `table2` にデータを追加します。 `table2`. データは削除されないことに注意してください `table1`.

クエリを正常に実行するには、次の条件を満たす必要があります:

- 両方のテーブルを作成するときに必要となる構造です。
- 両方のテーブルを作成するときに必要となる分割。

## REPLACE PARTITION

```
ALTER TABLE table2 REPLACE PARTITION partition_expr FROM table1
```

このクエリは、データパーティションを `table1` に `table2` の既存のパーティションを置き換え `table2`. データは削除されないことに注意してください `table1`.

クエリを正常に実行するには、次の条件を満たす必要があります:

- 両方のテーブルを作成するときに必要となる構造です。
- 両方のテーブルを作成するときに必要となる分割。

## MOVE PARTITION TO TABLE

```
ALTER TABLE table_source MOVE PARTITION partition_expr TO TABLE table_dest
```

このクエリは、データパーティションを `table_source` に `table_dest` からデータを削除すると `table_source`.

クエリを正常に実行するには、次の条件を満たす必要があります:

- 両方のテーブルを作成するときに必要となる構造です。
- 両方のテーブルを作成するときに必要となる分割。
- 両方のテーブルと同じでなければならエンジンです。 (複製または非複製)
- 両方のテーブルを作成するときに必要となる貯ます。

## CLEAR COLUMN IN PARTITION

```
ALTER TABLE table_name CLEAR COLUMN column_name IN PARTITION partition_expr
```

すべてリセット値で指定されたカラムがありました。もし `DEFAULT` 句が決定されたテーブルを作成するときに、このクエリは、指定された既定値に列の値を設定します。

例:

```
ALTER TABLE visits CLEAR COLUMN hour in PARTITION 201902
```

## FREEZE PARTITION

```
ALTER TABLE table_name FREEZE [PARTITION partition_expr]
```

このクエリーを作成し、地元のバックアップの指定されたパーティション もし `PARTITION` 条項を省略して、クエリーを作成し、バックアップの仕切れます。

### 注

バックアップ処理全体は、サーバーを停止せずに実行されます。

古いスタイルのテーブルの場合、パーティション名のプレフィックスを指定できます(例えば, '2019')のクエリーを作成し、バックアップのためのすべてに対応する隔壁セクションでのパーティション式の設定について [パーティション式の指定方法](#).

実行時に、データスナップショットの場合、クエリはテーブルデータへのハードリンクを作成します。Hardlinksに設置されているディレクトリ `/var/lib/clickhouse/shadow/N/...`, ここで:

- `/var/lib/clickhouse/` 設定で指定された作業ClickHouseディレクトリです。
- `N` バックアップの増分数です。

### 注

を使用する場合 [テーブル内のデータ格納用のディスクのセット](#) は、`shadow/N` ディレクトリが表示される毎にディスクデータを格納する部品と合わせによる `PARTITION` 式。

バックアップの内部と同じディレクトリ構造が作成されます `/var/lib/clickhouse/`. クエリの実行 '`chmod`' すべてのファイルの禁止に対して書き込みます。

バックアップの作成後、データをコピーするには `/var/lib/clickhouse/shadow/` ローカルサーバーから削除します。なお、`ALTER t FREEZE PARTITION` クエリは複製されません。するための地元のバックアップ、現地サーバーです。

クエリをバックアップで最初でお待ちしておりますので、現在のクエリーに対応するテーブルに仕上げた。

`ALTER TABLE t FREEZE PARTITION` コピーのみのデータのないテーブルメタデータを指すものとします。をバックアップ テーブルメタデータ、コピー、ファイル `/var/lib/clickhouse/metadata/database/table.sql`

バックアップからデータを復元するには:

1. テーブルが存在しない場合はテーブルを作成します。クエリを表示するには、を使用します。`.sql` ファイル(置換 `ATTACH` それで `CREATE`)。
2. からデータをコピーします `data/database/table/` バックアップ内のディレクトリ `/var/lib/clickhouse/data/database/table/detached/` ディレクトリ。
3. 走れ。 `ALTER TABLE t ATTACH PARTITION` テーブルにデータを追加するクエリ。

復元からのバックアップを必要としないの停止、サーバーにコピーします。

バックアップおよびデータの復元の詳細については、[データバックア セクション](#)

## CLEAR INDEX IN PARTITION

```
ALTER TABLE table_name CLEAR INDEX index_name IN PARTITION partition_expr
```

クエリは次のように動作します `CLEAR COLUMN` しかし、列データではなくインデックスをリセットします。

## FETCH PARTITION

```
ALTER TABLE table_name FETCH PARTITION partition_expr FROM 'path-in-zookeeper'
```

ダウンロードパーティションから別のサーバーです。このクエリーだけを再現します。

クエリは、次の操作を実行します:

1. ダウンロードパーティションから、指定されたザ-シャー。で `'path-in-zookeeper'` ZooKeeperでシャードへのパスを指定する必要があります。
2. 次に、クエリはダウンロードしたデータを `detached` のディレクトリ `table_name` テーブル。使用する `ATTACH PARTITION|PART` テーブルにデータを追加するクエリ。

例えば:

```
ALTER TABLE users FETCH PARTITION 201902 FROM '/clickhouse/tables/01-01/visits';
ALTER TABLE users ATTACH PARTITION 201902;
```

なお:

- その `ALTER ... FETCH PARTITION` クエリは複製されません。それは仕切りをに置きます `detached` ディレクトリの現地サーバーです。
- その `ALTER TABLE ... ATTACH` クエリが複製されます。すべてのレプリカにデータを追加します。データはレプリカのいずれかに追加されます。`detached` ディレクトリ、および他の人に-隣接するレプリカから。

ダウンロードする前に、システムかどうかをチェックすると、パーティションが存在するとテーブル構造。最も適切なレプリカは、正常なレプリカから自動的に選択されます。

クエリは呼び出されますが `ALTER TABLE` テーブル構造を変更せず、テーブル内で使用可能なデータを直ちに変更することはできません。

## MOVE PARTITION|PART

パーティションまたは MergeTree-エンジンテーブル。見る 複数のロックデバイスのためのデータ保存。

```
ALTER TABLE table_name MOVE PARTITION|PART partition_expr TO DISK|VOLUME 'disk_name'
```

その ALTER TABLE t MOVE クエリ:

- な再現が異なるレプリカで保管。
- 指定されたディスクまたはボリュームストレージポリシーで指定されたデータ移動の条件を適用できない場合、Queryはエラーを返します。
- 復帰できるエラーの場合、データの移動に移行している背景には、同時に ALTER TABLE t MOVE クエリまたはバックグラウンドデータマージの結果。この場合、ユーザーは追加の操作を実行しないでください。

例:

```
ALTER TABLE hits MOVE PART '20190301_14343_16206_438' TO VOLUME 'slow'  
ALTER TABLE hits MOVE PARTITION '2019-09-01' TO DISK 'fast_ssds'
```

## パーティション式の設定方法

パーティション式を指定するには ALTER ... PARTITION 異なる方法でのクエリ:

- からの値として `partition` の列 `system.parts` テーブル。例えば, `ALTER TABLE visits DETACH PARTITION 201901`
- テーブル列からの式として。定数と定数式がサポートされています。例えば, `ALTER TABLE visits DETACH PARTITION toYYYYMM(toDate('2019-01-25'))`.
- パーティションIDの使用。Partition IDは、ファイルシステムおよびZooKeeperのパーティションの名前として使用されるパーティションの文字列識別子(可能であれば人間が読める)です。パーティションIDは、`PARTITION ID` 一重引用符で囲まれた句。例えば, `ALTER TABLE visits DETACH PARTITION ID '201901'`.
- で `ALTER ATTACH PART` と `DROP DETACHED PART` パーツの名前を指定するには、文字列リテラルを使用します。`name` の列 `システムdetached_parts` テーブル。例えば, `ALTER TABLE visits ATTACH PART '201901_1_1_0'`.

ご利用の引用符を指定する場合、パーティションのエントランスは目を引く壁面緑化を表現。例えば、`String` その名前を引用符で指定する必要があります ('). のために `Date` と `Int*` 型引用符は必要ありません。

古いスタイルのテーブルの場合は、パーティションを数値として指定できます `201901` または文字列 `'201901'`. 新しいスタイルのテーブルの構文は、型によってより厳密になります（値の入力形式のパーサーと同様）。

上記のすべてのルールは、`OPTIMIZE` クエリ。を指定する場合にのみ分配時の最適化、非仕切られたテーブルセットの表現 `PARTITION tuple()`. 例えば:

```
OPTIMIZE TABLE table_not_partitioned PARTITION tuple() FINAL;
```

の例 `ALTER ... PARTITION` クエリは、テストで示されます `00502_custom_partitioning_local` と `00502_custom_partitioning_replicated_zookeeper`.

## テーブルTTLによる操作

変更できます `テーブルTTL` 以下のフォームのリクエストで:

```
ALTER TABLE table-name MODIFY TTL ttl-expression
```

## ALTER クエリの同期性

複製不可能なテーブルの場合、すべて `ALTER` クエリは同期的に実行されます。そのための `replicable` テーブル、クエリだけで追加指示のための適切な行動を `ZooKeeper` そして、アクション自体はできるだけ早く実行されます。ただし、クエリは、すべてのレプリカでこれらの操作が完了するまで待機できます。

のために `ALTER ... ATTACH|DETACH|DROP` クエリを使用することができます `replication_alter_partitions_sync` 待機を設定する設定。

可能な値: 0 – do not wait; 1 – only wait for own execution (default); 2 – wait for all.

## 突然変異

突然変異は、テーブル内の行を変更または削除できる `ALTER query variant` です。標準とは対照的に `UPDATE` と `DELETE` ポイントデータの変更、突然変異を目的としたクエリは、テーブル内の多くの行を変更する重い操作を目的としています。のために支えられる `MergeTree` 家族のテーブルエンジンなどのエンジンの複製です。

既存のテーブルはそのままで(変換は必要ありません)突然変異の準備ができていますが、最初の突然変異がテーブルに適用されると、そのメタデータ形式

現在利用可能なコマンド:

```
ALTER TABLE [db.]table DELETE WHERE filter_expr
```

その `filter_expr` 型である必要があります `UInt8`。このクエリは、この式がゼロ以外の値をとるテーブル内の行を削除します。

```
ALTER TABLE [db.]table UPDATE column1 = expr1 [, ...] WHERE filter_expr
```

その `filter_expr` 型である必要があります `UInt8`。このクエリは、指定された列の値を、指定された列の行の対応する式の値に更新します。`filter_expr` ゼロ以外の値をとります。値は、列型にキャストされます。`CAST` オペレーター プライマリキーまたはパーティションキーの計算で使用される列の更新はサポートされません。

```
ALTER TABLE [db.]table MATERIALIZE INDEX name IN PARTITION partition_name
```

クエリを再建の二次指標 `name` パーティション内 `partition_name`。

一つのクエリを含むことができ複数のコマンドをカンマで区切られています。

\*`MergeTree`表の場合、突然変異はデータ部分全体を書き換えることによって実行されます。原子性がない部品は準備ができているとすぐ変異させた部品のために置き換えられ、a `SELECT` 突然変異中に実行を開始したクエリには、すでに変異している部分のデータと、まだ変異していない部分のデータが表示されます。

突然変異は完全にそれらの作成順序によって順序付けられ、その順序で各部分に適用される。突然変異が送信される前にテーブルに挿入されたデータは変異され、その後に挿入されたデータは変異されません。突然変異は挿入を決してブロックしないことに注意してください。

突然変異クエリは、突然変異エントリが追加された直後に返されます(複製されたテーブルが `ZooKeeper` に、複製されていないテーブルがファイルシステムに)。の突然変異の実行を非同利用システムの概要を設定します。あなたが使用することができ、突然変異の進行を追跡するために `system.mutations` テーブル。正常に送信された突然変異は、`ClickHouse`サーバーが再起動されても実行され続けます。それが提出された後、突然変異をロールバックする方法はありませんが、突然変異が何らかの理由で立ち往生している場合、それは `KILL MUTATION` クエリ。

完了した突然変異のエントリはすぐに削除されません(保存されたエントリの数は、`finished_mutations_to_keep` ストレージエンジ 古い突然変異エントリは削除されます)。

## ALTER USER

`ClickHouse`ユーザーアカウントの変更。

## 構文

```
ALTER USER [IF EXISTS] name [ON CLUSTER cluster_name]
  [RENAME TO new_name]
  [IDENTIFIED [WITH {PLAINTEXT_PASSWORD|SHA256_PASSWORD|DOUBLE_SHA1_PASSWORD}] BY
  {'password'|'hash'}]
  [[ADD|DROP] HOST {LOCAL | NAME 'name' | REGEXP 'name_regex' | IP 'address' | LIKE 'pattern'} [,....] | ANY |
  NONE]
  [DEFAULT ROLE role [,....] | ALL | ALL EXCEPT role [,....] ]
  [SETTINGS variable [= value] [MIN [=] min_value] [MAX [=] max_value] [READONLY|WRITABLE] | PROFILE
  'profile_name'] [,....]
```

## 説明

使用するには `ALTER USER` あなたは、`ALTER USER` 特権だ

### 例

付与されたロールを既定に設定する:

```
ALTER USER user DEFAULT ROLE role1, role2
```

ロールが以前にユーザーに付与されていない場合、ClickHouseは例外をスローします。

付与されたすべての役割を `default` に設定します:

```
ALTER USER user DEFAULT ROLE ALL
```

将来ユーザーにロールが付与されると、自動的にデフォルトになります。

セットの付与の役割をデフォルトを除く `role1` と `role2`:

```
ALTER USER user DEFAULT ROLE ALL EXCEPT role1, role2
```

## ALTER ROLE

役割を変更します。

## 構文

```
ALTER ROLE [IF EXISTS] name [ON CLUSTER cluster_name]
  [RENAME TO new_name]
  [SETTINGS variable [= value] [MIN [=] min_value] [MAX [=] max_value] [READONLY|WRITABLE] | PROFILE
  'profile_name'] [,....]
```

## ALTER ROW POLICY

行ポリシーを変更します。

## 構文

```
ALTER [ROW] POLICY [IF EXISTS] name [ON CLUSTER cluster_name] ON [database.]table
  [RENAME TO new_name]
  [AS {PERMISSIVE | RESTRICTIVE}]
  [FOR SELECT]
  [USING {condition | NONE}] [,....]
  [TO {role [,....] | ALL | ALL EXCEPT role [,....]}]
```

# ALTER QUOTA

クオータを変更します。

## 構文

```
ALTER QUOTA [IF EXISTS] name [ON CLUSTER cluster_name]
  [RENAME TO new_name]
  [KEYED BY {'none' | 'user name' | 'ip address' | 'client key' | 'client key or user name' | 'client key or ip address'}]
  [FOR [RANDOMIZED] INTERVAL number {SECOND | MINUTE | HOUR | DAY | WEEK | MONTH | QUARTER | YEAR}
    {MAX { {QUERIES | ERRORS | RESULT ROWS | RESULT BYTES | READ ROWS | READ BYTES | EXECUTION TIME} =
    number } [,...]}]
    NO LIMITS | TRACKING ONLY} [,...]]
  [TO {role [,...]} | ALL | ALL EXCEPT role [,...]}]
```

# ALTER SETTINGS PROFILE

クオータを変更します。

## 構文

```
ALTER SETTINGS PROFILE [IF EXISTS] name [ON CLUSTER cluster_name]
  [RENAME TO new_name]
  [SETTINGS variable [= value] [MIN [=] min_value] [MAX [=] max_value] [READONLY|WRITABLE] | INHERIT
  'profile_name'] [,...]]
```

## システムクエリ

- RELOAD DICTIONARIES
- RELOAD DICTIONARY
- DROP DNS CACHE
- DROP MARK CACHE
- FLUSH LOGS
- RELOAD CONFIG
- SHUTDOWN
- KILL
- STOP DISTRIBUTED SENDS
- FLUSH DISTRIBUTED
- START DISTRIBUTED SENDS
- STOP MERGES
- START MERGES

## RELOAD DICTIONARIES

以前に正常に読み込まれたすべての辞書を再読み込みします。

デフォルトでは、辞書は遅延して読み込まれます **dictionaries\_lazy\_load**) したがって、起動時に自動的にロードされるのではなく、dictGet関数による最初のアクセス時に初期化されるか、ENGINE=Dictionaryを使用してテーブルから

選択されます。その SYSTEM RELOAD DICTIONARIES クエリなどの辞書(ロード)。  
常に戻ります Ok. 辞書の更新の結果に関係なく。

## 辞書Dictionary\_nameを再読み込み

辞書を完全に再読み込みします dictionary\_name ディクショナリの状態に関係なく(LOADED/NOT\_LOADED/FAILED)。  
常に戻ります Ok. 辞書の更新の結果に関係なく。  
ディクショナリのステータスは、 system.dictionaries テーブル。

```
SELECT name, status FROM system.dictionaries;
```

## DROP DNS CACHE

ClickHouseの内部DNSキャッシュをリセットします。場合によっては（古いClickHouseバージョンの場合）、インフラストラクチャを変更するとき（別のClickHouseサーバーまたは辞書で使用されているサーバーのIPアドレスを変更するより便利な（自動）キャッシング管理については、"disable\_internal\_dns\_cache,dns\_cache\_update\_period"パラメーター"を参照してください。

## DROP MARK CACHE

リセットをマークします。ClickHouseおよび性能試験の開発で使用される。

## FLUSH LOGS

Flushes buffers of log messages to system tables (e.g. system.query\_log). Allows you to not wait 7.5 seconds when debugging.

## RELOAD CONFIG

ClickHouse構成を再読み込みします。設定がZookeeperに格納されている場合に使用されます。

## SHUTDOWN

通常はClickHouseをシャットダウンします service clickhouse-server stop / kill {\$pid\_clickhouse-server})

## KILL

クリックハウスプロセスを中止します kill -9 {\$pid\_clickhouse-server})

## 分散テーブルの管理

ClickHouseは管理できます 分散 テーブル ユーザーがこれらのテーブルにデータを挿入すると、ClickHouseはまずクラスターノードに送信するデータのキューを作成し、それを非同期に送信します。キュー処理を管理するには STOP DISTRIBUTED SENDS, FLUSH DISTRIBUTED, and START DISTRIBUTED SENDS クエリ。分散データを同期して挿入することもできます。insert\_distributed\_sync 設定。

## STOP DISTRIBUTED SENDS

を無効にした背景データの分布を挿入する際、データを配布します。

```
SYSTEM STOP DISTRIBUTED SENDS [db.]<distributed_table_name>
```

## FLUSH DISTRIBUTED

ClickHouseが強制的にクラスターノードにデータを同期的に送信します。使用できないノードがある場合、ClickHouseは例外をスローし、クエリの実行を停止します。これは、すべてのノードがオンラインに戻ったときに発生します。

```
SYSTEM FLUSH DISTRIBUTED [db.]<distributed_table_name>
```

## START DISTRIBUTED SENDS

を背景データの分布を挿入する際、データを配布します。

```
SYSTEM START DISTRIBUTED SENDS [db.]<distributed_table_name>
```

## STOP MERGES

MergeTreeファミリ内のテーブルのバックグラウンドマージを停止できます:

```
SYSTEM STOP MERGES [[db.]merge_tree_family_table_name]
```

### 注

**DETACH / ATTACH** 以前にすべてのMergeTreeテーブルに対してマージが停止された場合でも、tableはテーブルのバックグラウンドマージを開始します。

## START MERGES

MergeTreeファミリ内のテーブルのバックグラウンドマージを開始できます:

```
SYSTEM START MERGES [[db.]merge_tree_family_table_name]
```

## クエリを表示

## SHOW CREATE TABLE

```
SHOW CREATE [TEMPORARY] [TABLE|DICTIONARY] [db.]table [INTO OUTFILE filename] [FORMAT format]
```

单一を返します String-タイプ ‘statement’ column, which contains a single value – the CREATE 指定したオブジェクトの作成に使用するクエリ。

## SHOW DATABASES

```
SHOW DATABASES [INTO OUTFILE filename] [FORMAT format]
```

一覧の全てのデータベースです。

このクエリは次と同じです `SELECT name FROM system.databases [INTO OUTFILE filename] [FORMAT format]`

## SHOW PROCESSLIST

```
SHOW PROCESSLIST [INTO OUTFILE filename] [FORMAT format]
```

の内容を出力します。 システムプロセス 現在処理されているクエリのリストを含むテーブル。 SHOW PROCESSLIST クエリ。

その `SELECT * FROM system.processes` クエリを返しますデータに現在のすべてのクエリ。

Tip(コンソールで実行):

```
$ watch -n1 "clickhouse-client --query='SHOW PROCESSLIST'"
```

## SHOW TABLES

テーブルの一覧を表示します。

```
SHOW [TEMPORARY] TABLES [{FROM | IN} <db>] [LIKE '<pattern>' | WHERE expr] [LIMIT <N>] [INTO OUTFILE <filename>] [FORMAT <format>]
```

もし `FROM` 句が指定されていない場合、クエリは現在のデータベースからテーブルの一覧を返します。

と同じ結果を得ることができます `SHOW TABLES` 次の方法でクエリを実行します:

```
SELECT name FROM system.tables WHERE database = <db> [AND name LIKE <pattern>] [LIMIT <N>] [INTO OUTFILE <filename>] [FORMAT <format>]
```

例

次のクエリでは、テーブルのリストから最初の二つの行を選択します。`system` 名前が含まれるデータベース `co.`

```
SHOW TABLES FROM system LIKE '%co%' LIMIT 2
```

name
aggregate_function_combinators
collations

## SHOW DICTIONARIES

のリストを表示します [外部辞書](#)。

```
SHOW DICTIONARIES [FROM <db>] [LIKE '<pattern>'] [LIMIT <N>] [INTO OUTFILE <filename>] [FORMAT <format>]
```

もし `FROM` 句が指定されていない場合、クエリは現在のデータベースから辞書のリストを返します。

同じ結果を得ることができます `SHOW DICTIONARIES` 次の方法でクエリを実行します:

```
SELECT name FROM system.dictionaries WHERE database = <db> [AND name LIKE <pattern>] [LIMIT <N>] [INTO OUTFILE <filename>] [FORMAT <format>]
```

例

次のクエリでは、テーブルのリストから最初の二つの行を選択します。`system` 名前が含まれるデータベース `reg.`

```
SHOW DICTIONARIES FROM db LIKE '%reg%' LIMIT 2
```

```
name  
regions |  
region_names |
```

## SHOW GRANTS

ユーザーの権限を表示します。

構文

```
SHOW GRANTS [FOR user]
```

Userが指定されていない場合、クエリは現在のユーザーの特権を返します。

## SHOW CREATE USER

Aで使用されたパラメータを示します **ユーザー作成**。

SHOW CREATE USER ユーザーパスワードを出力しません。

構文

```
SHOW CREATE USER [name | CURRENT_USER]
```

## SHOW CREATE ROLE

Aで使用されたパラメータを示します **ロールの作成**

構文

```
SHOW CREATE ROLE name
```

## SHOW CREATE ROW POLICY

Aで使用されたパラメータを示します **行ポリシーの作成**

構文

```
SHOW CREATE [ROW] POLICY name ON [database.]table
```

## SHOW CREATE QUOTA

Aで使用されたパラメータを示します **クオータの作成**

構文

```
SHOW CREATE QUOTA [name | CURRENT]
```

## SHOW CREATE SETTINGS PROFILE

Aで使用されたパラメータを示します **設定プロファイルの作成**

構文

```
SHOW CREATE [SETTINGS] PROFILE name
```

## EXPLAIN Statement

Shows the execution plan of a statement.

Syntax:

```
EXPLAIN [AST | SYNTAX | PLAN | PIPELINE] [setting = value, ...] SELECT ... [FORMAT ...]
```

Example:

```
EXPLAIN SELECT sum(number) FROM numbers(10) UNION ALL SELECT sum(number) FROM numbers(10) ORDER BY sum(number) ASC FORMAT TSV;
```

```
Union
Expression (Projection)
Expression (Before ORDER BY and SELECT)
Aggregating
Expression (Before GROUP BY)
SettingQuotaAndLimits (Set limits and quota after reading from storage)
ReadFromStorage (SystemNumbers)
Expression (Projection)
MergingSorted (Merge sorted streams for ORDER BY)
MergeSorting (Merge sorted blocks for ORDER BY)
PartialSorting (Sort each block for ORDER BY)
Expression (Before ORDER BY and SELECT)
Aggregating
Expression (Before GROUP BY)
SettingQuotaAndLimits (Set limits and quota after reading from storage)
ReadFromStorage (SystemNumbers)
```

## EXPLAIN Types

- **AST** — Abstract syntax tree.
- **SYNTAX** — Query text after AST-level optimizations.
- **PLAN** — Query execution plan.
- **PIPELINE** — Query execution pipeline.

## EXPLAIN AST

Dump query AST. Supports all types of queries, not only `SELECT`.

Examples:

```
EXPLAIN AST SELECT 1;
```

```
SelectWithUnionQuery (children 1)
ExpressionList (children 1)
SelectQuery (children 1)
ExpressionList (children 1)
Literal UInt64_1
```

```
EXPLAIN AST ALTER TABLE t1 DELETE WHERE date = today();
```

```
explain
AlterQuery t1 (children 1)
ExpressionList (children 1)
AlterCommand 27 (children 1)
Function equals (children 1)
ExpressionList (children 2)
Identifier date
Function today (children 1)
ExpressionList
```

## EXPLAIN SYNTAX

Returns query after syntax optimizations.

Example:

```
EXPLAIN SYNTAX SELECT * FROM system.numbers AS a, system.numbers AS b, system.numbers AS c;
```

```
SELECT
`--a.number` AS `a.number`,
`--b.number` AS `b.number`,
number AS `c.number`
FROM
(
  SELECT
    number AS `--a.number`,
    b.number AS `--b.number`
  FROM system.numbers AS a
  CROSS JOIN system.numbers AS b
) AS `--.s`
CROSS JOIN system.numbers AS c
```

## EXPLAIN PLAN

Dump query plan steps.

Settings:

- `header` — Prints output header for step. Default: 0.
- `description` — Prints step description. Default: 1.
- `indexes` — Shows used indexes, the number of filtered parts and the number of filtered granules for every index applied. Default: 0. Supported for **MergeTree** tables.
- `actions` — Prints detailed information about step actions. Default: 0.
- `json` — Prints query plan steps as a row in **JSON** format. Default: 0. It is recommended to use **TSVRaw** format to avoid unnecessary escaping.

Example:

```
EXPLAIN SELECT sum(number) FROM numbers(10) GROUP BY number % 4;
```

```
Union
Expression (Projection)
Expression (Before ORDER BY and SELECT)
Aggregating
Expression (Before GROUP BY)
SettingQuotaAndLimits (Set limits and quota after reading from storage)
ReadFromStorage (SystemNumbers)
```

## Note

Step and query cost estimation is not supported.

When `json = 1`, the query plan is represented in JSON format. Every node is a dictionary that always has the keys `Node Type` and `Plans`. `Node Type` is a string with a step name. `Plans` is an array with child step descriptions. Other optional keys may be added depending on node type and settings.

Example:

```
EXPLAIN json = 1, description = 0 SELECT 1 UNION ALL SELECT 2 FORMAT TSVRaw;
```

```
[{"Plan": {"Node Type": "Union", "Plans": [{"Node Type": "Expression", "Plans": [{"Node Type": "SettingQuotaAndLimits", "Plans": [{"Node Type": "ReadFromStorage"}]}]}, {"Node Type": "Expression", "Plans": [{"Node Type": "SettingQuotaAndLimits", "Plans": [{"Node Type": "ReadFromStorage"}]}]}]}]
```

With `description = 1`, the `Description` key is added to the step:

```
{ "Node Type": "ReadFromStorage",
  "Description": "SystemOne"
}
```

With `header = 1`, the `Header` key is added to the step as an array of columns.

Example:

```
EXPLAIN json = 1, description = 0, header = 1 SELECT 1, 2 + dummy;
```

```
[  
  {  
    "Plan": {  
      "Node Type": "Expression",  
      "Header": [  
        {  
          "Name": "1",  
          "Type": "UInt8"  
        },  
        {  
          "Name": "plus(2, dummy)",  
          "Type": "UInt16"  
        }  
      ],  
      "Plans": [  
        {  
          "Node Type": "SettingQuotaAndLimits",  
          "Header": [  
            {  
              "Name": "dummy",  
              "Type": "UInt8"  
            }  
          ],  
          "Plans": [  
            {  
              "Node Type": "ReadFromStorage",  
              "Header": [  
                {  
                  "Name": "dummy",  
                  "Type": "UInt8"  
                }  
              ]  
            }  
          ]  
        }  
      ]  
    }  
  ]
```

With `indexes = 1`, the `Indexes` key is added. It contains an array of used indexes. Each index is described as JSON with `Type` key (a string `MinMax`, `Partition`, `PrimaryKey` or `Skip`) and optional keys:

- `Name` — An index name (for now, is used only for `Skip` index).
- `Keys` — An array of columns used by the index.
- `Condition` — A string with condition used.
- `Description` — An index (for now, is used only for `Skip` index).
- `Initial Parts` — A number of parts before the index is applied.
- `Selected Parts` — A number of parts after the index is applied.
- `Initial Granules` — A number of granules before the index is applied.
- `Selected Granules` — A number of granules after the index is applied.

Example:

```

"Node Type": "ReadFromMergeTree",
"Indexes": [
{
  "Type": "MinMax",
  "Keys": ["y"],
  "Condition": "(y in [1, +inf))",
  "Initial Parts": 5,
  "Selected Parts": 4,
  "Initial Granules": 12,
  "Selected Granules": 11
},
{
  "Type": "Partition",
  "Keys": ["y", "bitAnd(z, 3)"],
  "Condition": "and((bitAnd(z, 3) not in [1, 1]), and((y in [1, +inf)), (bitAnd(z, 3) not in [1, 1])))",
  "Initial Parts": 4,
  "Selected Parts": 3,
  "Initial Granules": 11,
  "Selected Granules": 10
},
{
  "Type": "PrimaryKey",
  "Keys": ["x", "y"],
  "Condition": "and((x in [11, +inf)), (y in [1, +inf)))",
  "Initial Parts": 3,
  "Selected Parts": 2,
  "Initial Granules": 10,
  "Selected Granules": 6
},
{
  "Type": "Skip",
  "Name": "t_minmax",
  "Description": "minmax GRANULARITY 2",
  "Initial Parts": 2,
  "Selected Parts": 1,
  "Initial Granules": 6,
  "Selected Granules": 2
},
{
  "Type": "Skip",
  "Name": "t_set",
  "Description": "set GRANULARITY 2",
  "Initial Parts": 1,
  "Selected Parts": 1,
  "Initial Granules": 2,
  "Selected Granules": 1
}
]

```

With `actions = 1`, added keys depend on step type.

Example:

```
EXPLAIN json = 1, actions = 1, description = 0 SELECT 1 FORMAT TSVRaw;
```

```
[
  {
    "Plan": {
      "Node Type": "Expression",
      "Expression": {
        "Inputs": [],
        "Actions": [
          {
            "Node Type": "Column",
            "Result Type": "UInt8",
            "Result Type": "Column",
            "Column": "Const(UInt8)",
            "Arguments": [],
            "Removed Arguments": [],
            "Result": 0
          }
        ],
        "Outputs": [
          {
            "Name": "1",
            "Type": "UInt8"
          }
        ],
        "Positions": [0],
        "Project Input": true
      },
      "Plans": [
        {
          "Node Type": "SettingQuotaAndLimits",
          "Plans": [
            {
              "Node Type": "ReadFromStorage"
            }
          ]
        }
      ]
    }
  ]
]
```

## EXPLAIN PIPELINE

Settings:

- `header` — Prints header for each output port. Default: 0.
- `graph` — Prints a graph described in the [DOT](#) graph description language. Default: 0.
- `compact` — Prints graph in compact mode if `graph` setting is enabled. Default: 1.

Example:

```
EXPLAIN PIPELINE SELECT sum(number) FROM numbers_mt(100000) GROUP BY number % 4;
```

```
(Union)
(Expression)
ExpressionTransform
(Expression)
ExpressionTransform
(Aggregating)
Resize 2 → 1
AggregatingTransform × 2
(Expression)
ExpressionTransform × 2
(SettingQuotaAndLimits)
(ReadFromStorage)
NumbersMt × 2 0 → 1
```

## EXPLAIN ESTIMATE

Shows the estimated number of rows, marks and parts to be read from the tables while processing the query. Works with tables in the **MergeTree** family.

### Example

Creating a table:

```
CREATE TABLE ttt (i Int64) ENGINE = MergeTree() ORDER BY i SETTINGS index_granularity = 16, write_final_mark = 0;
INSERT INTO ttt SELECT number FROM numbers(128);
OPTIMIZE TABLE ttt;
```

Query:

```
EXPLAIN ESTIMATE SELECT * FROM ttt;
```

Result:

database	table	parts	rows	marks
default	ttt	1	128	8

## GRANT

- 助成金 **特権** ClickHouseユーザー アカウントまたはロールへ。
- 員の役割をユーザー アカウントまたはその他の役割です。

権限を取り消すには **REVOKE** 声明。また、付与された権限を **SHOW GRANTS** 声明。

### 権限構文の付与

```
GRANT [ON CLUSTER cluster_name] privilege[(column_name [,....])] [,....] ON {db.table|db.*|*.*|table|*} TO {user | role | CURRENT_USER} [,....] [WITH GRANT OPTION] [WITH REPLACE OPTION]
```

- privilege** — Type of privilege.
- role** — ClickHouse user role.
- user** — ClickHouse user account.

この **WITH GRANT OPTION** 句の付与 **user** または **role** 実行する許可を得て **GRANT** クエリ。ユーザーは、持っているスコープとそれ以下の権限を付与できます。

この **WITH REPLACE OPTION** 句は **user** または **role** の新しい特権で古い特権を置き換えます、指定しない場合は、古い特権を古いものに追加してください

### ロール構文の割り当て

```
GRANT [ON CLUSTER cluster_name] role [,....] TO {user | another_role | CURRENT_USER} [,....] [WITH ADMIN OPTION] [WITH REPLACE OPTION]
```

- role** — ClickHouse user role.
- user** — ClickHouse user account.

この `WITH ADMIN OPTION` 勅の付与 `ADMIN OPTION` への特権 `user` または `role`.

この `WITH REPLACE OPTION` 勅は `user` または `role` の新しい役割によって古い役割を置き換えます、指定しない場合は、古い特権を古いものに追加してください

## 使用法

使用するには `GRANT` アカウントには `GRANT OPTION` 特権だ 権限を付与できるのは、アカウント権限の範囲内でのみです。

たとえば、管理者は `john` クエリによるアカウント：

```
GRANT SELECT(x,y) ON db.table TO john WITH GRANT OPTION
```

つまり `john` 実行する権限があります：

- `SELECT x,y FROM db.table.`
- `SELECT x FROM db.table.`
- `SELECT y FROM db.table.`

`john` 実行できない `SELECT z FROM db.table.` この `SELECT * FROM db.table` また、利用できません。このクエリを処理すると、ClickHouseはデータを返しません。`x` そして `y`. 唯一の例外は、テーブルにのみ含まれている場合です `x` そして `y` 列。この場合、ClickHouseはすべてのデータを返します。

また `john` は、`GRANT OPTION` そのため、同じまたはより小さいスコープの特権を持つ他のユーザーに付与できます。

アスタリスクを使用できる権限の指定 (\*) テーブルまたはデータベース名の代わりに。例えば、`GRANT SELECT ON db.* TO john` クエリ許可 `john` 実行するには `SELECT` すべてのテーブルに対するクエリ `db` データベース。また、データベース名を省略できます。この場合、現在のデータベースに特権が付与されます。例えば、`GRANT SELECT ON * TO john` 現在のデータベース、`GRANT SELECT ON mytable TO john` の権限を付与します。`mytable` 現在のデータベース内のテーブル。

へのアクセス `system database` は常に許可されます(このデータベースはクエリの処理に使用されるため)。

一つのクエリで複数のアカウントに複数の権限を付与できます。クエリ `GRANT SELECT, INSERT ON *.* TO john, robin` アカウントを許可 `john` そして `robin` 実行するには `INSERT` そして `SELECT` クエリのテーブルのすべてのデータベースに、サーバーにコピーします。

## 特権

特権は、特定の種類のクエリを実行する権限です。

権限には階層構造があります。許可されるクエリのセットは、特権スコープに依存します。

特権の階層：

- `SELECT`
- `INSERT`

- **ALTER**

- **ALTER TABLE**
  - **ALTER UPDATE**
  - **ALTER DELETE**
- **ALTER COLUMN**
  - **ALTER ADD COLUMN**
  - **ALTER DROP COLUMN**
  - **ALTER MODIFY COLUMN**
  - **ALTER COMMENT COLUMN**
  - **ALTER CLEAR COLUMN**
  - **ALTER RENAME COLUMN**
- **ALTER INDEX**
  - **ALTER ORDER BY**
  - **ALTER ADD INDEX**
  - **ALTER DROP INDEX**
  - **ALTER MATERIALIZE INDEX**
  - **ALTER CLEAR INDEX**
- **ALTER CONSTRAINT**
  - **ALTER ADD CONSTRAINT**
  - **ALTER DROP CONSTRAINT**
- **ALTER TTL**
- **ALTER MATERIALIZE TTL**
- **ALTER SETTINGS**
- **ALTER MOVE PARTITION**
- **ALTER FETCH PARTITION**
- **ALTER FREEZE PARTITION**
- **ALTER VIEW**
  - **ALTER VIEW REFRESH**
  - **ALTER VIEW MODIFY QUERY**

- **CREATE**

- **CREATE DATABASE**
- **CREATE TABLE**
- **CREATE VIEW**
- **CREATE DICTIONARY**
- **CREATE TEMPORARY TABLE**

- **DROP**
  - `DROP DATABASE`
  - `DROP TABLE`
  - `DROP VIEW`
  - `DROP DICTIONARY`
- **TRUNCATE**
- **OPTIMIZE**
- **SHOW**
  - `SHOW DATABASES`
  - `SHOW TABLES`
  - `SHOW COLUMNS`
  - `SHOW DICTIONARIES`
- **KILL QUERY**

- **ACCESS MANAGEMENT**

- CREATE USER
- ALTER USER
- DROP USER
- CREATE ROLE
- ALTER ROLE
- DROP ROLE
- CREATE ROW POLICY
- ALTER ROW POLICY
- DROP ROW POLICY
- CREATE QUOTA
- ALTER QUOTA
- DROP QUOTA
- CREATE SETTINGS PROFILE
- ALTER SETTINGS PROFILE
- DROP SETTINGS PROFILE
- SHOW ACCESS
  - SHOW\_USERS
  - SHOW\_ROLES
  - SHOW\_ROW\_POLICIES
  - SHOW\_QUOTAS
  - SHOW\_SETTINGS\_PROFILES
- ROLE ADMIN

- **SYSTEM**

- SYSTEM SHUTDOWN
  - SYSTEM DROP CACHE
    - SYSTEM DROP DNS CACHE
    - SYSTEM DROP MARK CACHE
    - SYSTEM DROP UNCOMPRESSED CACHE
  - SYSTEM RELOAD
    - SYSTEM RELOAD CONFIG
    - SYSTEM RELOAD DICTIONARY
    - SYSTEM RELOAD EMBEDDED DICTIONARIES
  - SYSTEM MERGES
  - SYSTEM TTL MERGES
  - SYSTEM FETCHES
  - SYSTEM MOVES
  - SYSTEM SENDS
    - SYSTEM DISTRIBUTED SENDS
    - SYSTEM REPLICATED SENDS
  - SYSTEM REPLICATION QUEUES
  - SYSTEM SYNC REPLICA
  - SYSTEM RESTART REPLICA
  - SYSTEM FLUSH
    - SYSTEM FLUSH DISTRIBUTED
    - SYSTEM FLUSH LOGS
- **INTROSPECTION**
    - addressToLine
    - addressToSymbol
    - demangle

- SOURCES

- FILE
  - URL
  - REMOTE
  - YSQL
  - ODBC
  - JDBC
  - HDFS
  - S3
- dictGet

この階層がどのように扱われるかの例:

- この ALTER 特典を含む他のすべての ALTER\* 特権だ
- ALTER CONSTRAINT 含む ALTER ADD CONSTRAINT そして ALTER DROP CONSTRAINT 特権だ

特権は異なるレベルで適用されます。 レベルを知ることは、特権に利用可能な構文を示唆しています。

レベル（下位から上位へ）:

- COLUMN — Privilege can be granted for column, table, database, or globally.
- TABLE — Privilege can be granted for table, database, or globally.
- VIEW — Privilege can be granted for view, database, or globally.
- DICTIONARY — Privilege can be granted for dictionary, database, or globally.
- DATABASE — Privilege can be granted for database or globally.
- GLOBAL — Privilege can be granted only globally.
- GROUP — Groups privileges of different levels. When GROUP-レベルの特権が付与され、使用される構文に対応するグループからの特権のみが付与されます。

許可される構文の例:

- GRANT SELECT(x) ON db.table TO user
- GRANT SELECT ON db.\* TO user

禁止された構文の例:

- GRANT CREATE USER(x) ON db.table TO user
- GRANT CREATE USER ON db.\* TO user

特別特典 ALL ユーザーアカウントまたはロールにすべての権限を付与します。

既定では、ユーザーアカウントまたはロールには特権はありません。

ユーザーまたはロールに権限がない場合は、次のように表示されます NONE 特権だ

実装によるクエリには、一連の特権が必要です。たとえば、**RENAME** クエリには次の権限が必要です: **SELECT**, **CREATE TABLE**, **INSERT** そして **DROP TABLE**.

## SELECT

実行を許可する **SELECT** クエリ。

特権レベル: COLUMN.

### 説明

ユーザーの交付を受けるこの権限での実行 **SELECT** 指定した表およびデータベース内の指定した列のリストに対するクエリ。ユーザーが他の列を含む場合、クエリはデータを返しません。

次の特権を考慮してください:

```
GRANT SELECT(x,y) ON db.table TO john
```

この特権は **john** 実行するには **SELECT** データを含むクエリ **x** および/または **y** の列 **db.table** 例えば, **SELECT x FROM db.table. john** 実行できない **SELECT z FROM db.table.** この **SELECT \* FROM db.table** また、利用できません。このクエリを処理すると、ClickHouseはデータを返しません。**x** そして **y**. 唯一の例外は、テーブルにのみ含まれている場合です **x** そして **y** この場合、ClickHouseはすべてのデータを返します。

## INSERT

実行を許可する **INSERT** クエリ。

特権レベル: COLUMN.

### 説明

ユーザーの交付を受けるこの権限での実行 **INSERT** 指定した表およびデータベース内の指定した列のリストに対するクエリ。ユーザーが他の列を含む場合、指定されたクエリはデータを挿入しません。

### 例

```
GRANT INSERT(x,y) ON db.table TO john
```

許可された特権は **john** にデータを挿入するには **x** および/または **y** の列 **db.table.**

## ALTER

実行を許可する **ALTER** 次の特権の階層に従ってクエリを実行します:

■ ALTER. レベル: COLUMN.

- ALTER TABLE. レベル: GROUP
  - ALTER UPDATE. レベル: COLUMN. 別名: UPDATE
  - ALTER DELETE. レベル: COLUMN. 別名: DELETE
- ALTER COLUMN. レベル: GROUP
  - ALTER ADD COLUMN. レベル: COLUMN. 別名: ADD COLUMN
  - ALTER DROP COLUMN. レベル: COLUMN. 別名: DROP COLUMN
  - ALTER MODIFY COLUMN. レベル: COLUMN. 别名: MODIFY COLUMN
  - ALTER COMMENT COLUMN. レベル: COLUMN. 别名: COMMENT COLUMN
  - ALTER CLEAR COLUMN. レベル: COLUMN. 别名: CLEAR COLUMN
  - ALTER RENAME COLUMN. レベル: COLUMN. 别名: RENAME COLUMN
- ALTER INDEX. レベル: GROUP. 别名: INDEX
  - ALTER ORDER BY. レベル: TABLE. 别名: ALTER MODIFY ORDER BY, MODIFY ORDER BY
  - ALTER ADD INDEX. レベル: TABLE. 别名: ADD INDEX
  - ALTER DROP INDEX. レベル: TABLE. 别名: DROP INDEX
  - ALTER MATERIALIZE INDEX. レベル: TABLE. 别名: MATERIALIZE INDEX
  - ALTER CLEAR INDEX. レベル: TABLE. 别名: CLEAR INDEX
- ALTER CONSTRAINT. レベル: GROUP. 别名: CONSTRAINT
  - ALTER ADD CONSTRAINT. レベル: TABLE. 别名: ADD CONSTRAINT
  - ALTER DROP CONSTRAINT. レベル: TABLE. 别名: DROP CONSTRAINT
- ALTER TTL. レベル: TABLE. 别名: ALTER MODIFY TTL, MODIFY TTL
- ALTER MATERIALIZE TTL. レベル: TABLE. 别名: MATERIALIZE TTL
- ALTER SETTINGS. レベル: TABLE. 别名: ALTER SETTING, ALTER MODIFY SETTING, MODIFY SETTING
- ALTER MOVE PARTITION. レベル: TABLE. 别名: ALTER MOVE PART, MOVE PARTITION, MOVE PART
- ALTER FETCH PARTITION. レベル: TABLE. 别名: FETCH PARTITION
- ALTER FREEZE PARTITION. レベル: TABLE. 别名: FREEZE PARTITION
- ALTER VIEW レベル: GROUP
  - ALTER VIEW REFRESH. レベル: VIEW. 别名: ALTER LIVE VIEW REFRESH, REFRESH VIEW
  - ALTER VIEW MODIFY QUERY. レベル: VIEW. 别名: ALTER TABLE MODIFY QUERY

この階層がどのように扱われるかの例:

- この ALTER 特権を含む他のすべての ALTER\* 特権だ
- ALTER CONSTRAINT 含む ALTER ADD CONSTRAINT そして ALTER DROP CONSTRAINT 特権だ

#### ノート

- この MODIFY SETTING 権限を変更できるテーブルエンジンを設定します。 設定やサーバー構成パラメーターには影響しません。
- この ATTACH 操作は必要とします CREATE 特権だ

- この **DETACH** 操作は必要とします **DROP** 特権だ
- によって突然変異を停止するには **KILL MUTATION** クエリ、あなたはこの突然変異を開始する権限を持っている必要があります。たとえば、**ALTER UPDATE** 問い合わせ、必要とします **ALTER UPDATE**, **ALTER TABLE**, または **ALTER** 特権だ

## CREATE

実行を許可する **CREATE** そして **ATTACH** DDL-次の特権の階層に従ったクエリ：

- **CREATE**. レベル: **GROUP**
  - **CREATE DATABASE**. レベル: **DATABASE**
  - **CREATE TABLE**. レベル: **TABLE**
  - **CREATE VIEW**. レベル: **VIEW**
  - **CREATE DICTIONARY**. レベル: **DICTIONARY**
  - **CREATE TEMPORARY TABLE**. レベル: **GLOBAL**

ノート

- 削除し、作成したテーブルは、ユーザーニーズ **DROP**.

## DROP

実行を許可する **DROP** そして **DETACH** 次の特権の階層に従ってクエリを実行します:

- **DROP**. レベル:
  - **DROP DATABASE**. レベル: **DATABASE**
  - **DROP TABLE**. レベル: **TABLE**
  - **DROP VIEW**. レベル: **VIEW**
  - **DROP DICTIONARY**. レベル: **DICTIONARY**

## TRUNCATE

実行を許可する **TRUNCATE** クエリ。

特権レベル: **TABLE**.

## OPTIMIZE

実行を許可する **OPTIMIZE TABLE** クエリ。

特権レベル: **TABLE**.

## SHOW

実行を許可する **SHOW**, **DESCRIBE**, **USE**, and **EXISTS** 次の特権の階層に従ってクエリを実行します:

- SHOW. レベル: GROUP
- SHOW DATABASES. レベル: DATABASE. 実行を許可する SHOW DATABASES, SHOW CREATE DATABASE, USE <database> クエリ。
- SHOW TABLES. レベル: TABLE. 実行を許可する SHOW TABLES, EXISTS <table>, CHECK <table> クエリ。
- SHOW COLUMNS. レベル: COLUMN. 実行を許可する SHOW CREATE TABLE, DESCRIBE クエリ。
- SHOW DICTIONARIES. レベル: DICTIONARY. 実行を許可する SHOW DICTIONARIES, SHOW CREATE DICTIONARY, EXISTS <dictionary> クエリ。

ノート

ユーザは、SHOW 特典の場合は、その他の特典に関する指定されたテーブル、辞書やデータベースです。

## KILL QUERY

実行を許可する KILL 次の特権の階層に従ってクエリを実行します:

特権レベル: GLOBAL.

ノート

KILL QUERY 特典でユーザを殺すクエリの他のユーザー

## ACCESS MANAGEMENT

ユーザー、ロール、および行ポリシーを管理するクエリを実行できます。

- ACCESS MANAGEMENT. レベル: GROUP
  - CREATE USER. レベル: GLOBAL
  - ALTER USER. レベル: GLOBAL
  - DROP USER. レベル: GLOBAL
  - CREATE ROLE. レベル: GLOBAL
  - ALTER ROLE. レベル: GLOBAL
  - DROP ROLE. レベル: GLOBAL
  - ROLE ADMIN. レベル: GLOBAL
- CREATE ROW POLICY. レベル: GLOBAL. 別名: CREATE POLICY
- ALTER ROW POLICY. レベル: GLOBAL. 別名: ALTER POLICY
- DROP ROW POLICY. レベル: GLOBAL. 別名: DROP POLICY
- CREATE QUOTA. レベル: GLOBAL
- ALTER QUOTA. レベル: GLOBAL
- DROP QUOTA. レベル: GLOBAL
- CREATE SETTINGS PROFILE. レベル: GLOBAL. 別名: CREATE PROFILE
- ALTER SETTINGS PROFILE. レベル: GLOBAL. 別名: ALTER PROFILE
- DROP SETTINGS PROFILE. レベル: GLOBAL. 別名: DROP PROFILE
- SHOW ACCESS. レベル: GROUP
  - SHOW\_USERS. レベル: GLOBAL. 別名: SHOW CREATE USER
  - SHOW\_ROLES. レベル: GLOBAL. 別名: SHOW CREATE ROLE
  - SHOW\_ROW\_POLICIES. レベル: GLOBAL. 別名: SHOW POLICIES, SHOW CREATE ROW POLICY, SHOW CREATE POLICY
  - SHOW\_QUOTAS. レベル: GLOBAL. 別名: SHOW CREATE QUOTA
  - SHOW\_SETTINGS\_PROFILES. レベル: GLOBAL. 別名: SHOW PROFILES, SHOW CREATE SETTINGS PROFILE, SHOW CREATE PROFILE

この **ROLE ADMIN** 特典できるユーザーに割り当ておよび取り消す為の役割を含めるものではありませんので、ユーザーの管理のオプションです。

## SYSTEM

ユーザーに実行を許可する **SYSTEM** 次の特権の階層に従ってクエリを実行します。

- SYSTEM. レベル: GROUP
  - SYSTEM SHUTDOWN. レベル: GLOBAL. 別名: SYSTEM KILL, SHUTDOWN
  - SYSTEM DROP CACHE. 別名: DROP CACHE
    - SYSTEM DROP DNS CACHE. レベル: GLOBAL. 別名: SYSTEM DROP DNS, DROP DNS CACHE, DROP DNS
    - SYSTEM DROP MARK CACHE. レベル: GLOBAL. 別名: SYSTEM DROP MARK, DROP MARK CACHE, DROP MARKS
    - SYSTEM DROP UNCOMPRESSED CACHE. レベル: GLOBAL. 別名: SYSTEM DROP UNCOMPRESSED, DROP UNCOMPRESSED CACHE, DROP UNCOMPRESSED
  - SYSTEM RELOAD. レベル: GROUP
    - SYSTEM RELOAD CONFIG. レベル: GLOBAL. 別名: RELOAD CONFIG
    - SYSTEM RELOAD DICTIONARY. レベル: GLOBAL. 別名: SYSTEM RELOAD DICTIONARIES, RELOAD DICTIONARY, RELOAD DICTIONARIES
    - SYSTEM RELOAD EMBEDDED DICTIONARIES. レベル: GLOBAL. 別名: RELOAD EMBEDDED DICTIONARIES
  - SYSTEM MERGES. レベル: TABLE. 別名: SYSTEM STOP MERGES, SYSTEM START MERGES, STOP MERGES, START MERGES
  - SYSTEM TTL MERGES. レベル: TABLE. 別名: SYSTEM STOP TTL MERGES, SYSTEM START TTL MERGES, STOP TTL MERGES, START TTL MERGES
  - SYSTEM FETCHES. レベル: TABLE. 別名: SYSTEM STOP FETCHES, SYSTEM START FETCHES, STOP FETCHES, START FETCHES
  - SYSTEM MOVES. レベル: TABLE. 別名: SYSTEM STOP MOVES, SYSTEM START MOVES, STOP MOVES, START MOVES
  - SYSTEM SENDS. レベル: GROUP. 別名: SYSTEM STOP SENDS, SYSTEM START SENDS, STOP SENDS, START SENDS
    - SYSTEM DISTRIBUTED SENDS. レベル: TABLE. 别名: SYSTEM STOP DISTRIBUTED SENDS, SYSTEM START DISTRIBUTED SENDS, STOP DISTRIBUTED SENDS, START DISTRIBUTED SENDS
    - SYSTEM REPLICATED SENDS. レベル: TABLE. 别名: SYSTEM STOP REPLICATED SENDS, SYSTEM START REPLICATED SENDS, STOP REPLICATED SENDS, START REPLICATED SENDS
  - SYSTEM REPLICATION QUEUES. レベル: TABLE. 别名: SYSTEM STOP REPLICATION QUEUES, SYSTEM START REPLICATION QUEUES, STOP REPLICATION QUEUES, START REPLICATION QUEUES
  - SYSTEM SYNC REPLICA. レベル: TABLE. 别名: SYNC REPLICA
  - SYSTEM RESTART REPLICA. レベル: TABLE. 别名: RESTART REPLICA
  - SYSTEM FLUSH. レベル: GROUP
    - SYSTEM FLUSH DISTRIBUTED. レベル: TABLE. 别名: FLUSH DISTRIBUTED
    - SYSTEM FLUSH LOGS. レベル: GLOBAL. 别名: FLUSH LOGS

この SYSTEM RELOAD EMBEDDED DICTIONARIES によって暗黙的に付与される特権 SYSTEM RELOAD DICTIONARY ON \*.\* 特権だ

## INTROSPECTION

使用を許可する 内省 機能。

- INTROSPECTION. レベル: GROUP. 别名: INTROSPECTION FUNCTIONS
  - addressToLine. レベル: GLOBAL
  - addressToSymbol. レベル: GLOBAL
  - demangle. レベル: GLOBAL

## SOURCES

外部データソースの使用を許可します。に適用されます **表エンジン** そして **テーブル関数**。

- SOURCES. レベル: GROUP

- FILE. レベル: GLOBAL

- URL. レベル: GLOBAL

- REMOTE. レベル: GLOBAL

- YSQL. レベル: GLOBAL

- ODBC. レベル: GLOBAL

- JDBC. レベル: GLOBAL

- HDFS. レベル: GLOBAL

- S3. レベル: GLOBAL

この **SOURCES** 特権は、すべてのソースの使用を可能にします。また、各ソースに個別に権限を付与することもできます。ソースを使用するには、追加の権限が必要です。

例:

- テーブルを作成するには MySQL テーブルエンジン、必要とします CREATE TABLE (ON db.table\_name) そして MySQL 特権だ
- を使用するには mysql テーブル関数、必要とします CREATE TEMPORARY TABLE そして MySQL 特権だ

## dictGet

- dictGet. 別名: dictHas, dictGetHierarchy, dictIsIn

ユーザーに実行を許可する **dictGet**, **ディクタス**, **dictGetHierarchy**, **ジクチシン** 機能。

特権レベル: DICTIONARY.

例

- GRANT dictGet ON mydb.mydictionary TO john

- GRANT dictGet ON mydictionary TO john

## ALL

助成金の全ての権限を規制組織のユーザー アカウント または役割を担う。

## NONE

権限は付与されません。

## ADMIN OPTION

この **ADMIN OPTION** 特典できるユーザー補助金の役割を他のユーザーです。

# REVOKE Statement

Revokes privileges from users or roles.

# Syntax

## Revoking privileges from users

```
REVOKE [ON CLUSTER cluster_name] privilege[(column_name [...])] [...] ON {db.table|db.*|*.*|table*} FROM {user | CURRENT_USER} [...] | ALL | ALL EXCEPT {user | CURRENT_USER} [...]
```

## Revoking roles from users

```
REVOKE [ON CLUSTER cluster_name] [ADMIN OPTION FOR] role [...] FROM {user | role | CURRENT_USER} [...] | ALL | ALL EXCEPT {user_name | role_name | CURRENT_USER} [...]
```

# Description

To revoke some privilege you can use a privilege of a wider scope than you plan to revoke. For example, if a user has the `SELECT (x,y)` privilege, administrator can execute `REVOKE SELECT(x,y) ...`, or `REVOKE SELECT * ...`, or even `REVOKE ALL PRIVILEGES ...` query to revoke this privilege.

## Partial Revokes

You can revoke a part of a privilege. For example, if a user has the `SELECT *.*` privilege you can revoke from it a privilege to read data from some table or a database.

# Examples

Grant the `john` user account with a privilege to select from all the databases, excepting the `accounts` one:

```
GRANT SELECT ON *.* TO john;
REVOKE SELECT ON accounts.* FROM john;
```

Grant the `mira` user account with a privilege to select from all the columns of the `accounts.staff` table, excepting the `wage` one.

```
GRANT SELECT ON accounts.staff TO mira;
REVOKE SELECT(wage) ON accounts.staff FROM mira;
```

# ATTACH Statement

Attaches a table or a dictionary, for example, when moving a database to another server.

## Syntax

```
ATTACH TABLE|DICTIONARY [IF NOT EXISTS] [db.]name [ON CLUSTER cluster] ...
```

The query does not create data on the disk, but assumes that data is already in the appropriate places, and just adds information about the table or the dictionary to the server. After executing the `ATTACH` query, the server will know about the existence of the table or the dictionary.

If a table was previously detached (`DETACH` query), meaning that its structure is known, you can use shorthand without defining the structure.

# Attach Existing Table

## Syntax

```
ATTACH TABLE [IF NOT EXISTS] [db.]name [ON CLUSTER cluster]
```

This query is used when starting the server. The server stores table metadata as files with `ATTACH` queries, which it simply runs at launch (with the exception of some system tables, which are explicitly created on the server).

If the table was detached permanently, it won't be reattached at the server start, so you need to use `ATTACH` query explicitly.

## Create New Table And Attach Data

### With Specified Path to Table Data

The query creates a new table with provided structure and attaches table data from the provided directory in `user_files`.

#### Syntax

```
ATTACH TABLE name FROM 'path/to/data/' (col1 Type1, ...)
```

#### Example

Query:

```
DROP TABLE IF EXISTS test;
INSERT INTO TABLE FUNCTION file('01188_attach/test/data.TSV', 'TSV', 's String, n UInt8') VALUES ('test', 42);
ATTACH TABLE test FROM '01188_attach/test' (s String, n UInt8) ENGINE = File(TSV);
SELECT * FROM test;
```

Result:

s	n
test	42

### With Specified Table UUID

This query creates a new table with provided structure and attaches data from the table with the specified UUID.

It is supported by the [Atomic](#) database engine.

#### Syntax

```
ATTACH TABLE name UUID '<uuid>' (col1 Type1, ...)
```

## Attach Existing Dictionary

Attaches a previously detached dictionary.

#### Syntax

```
ATTACH DICTIONARY [IF NOT EXISTS] [db.]name [ON CLUSTER cluster]
```

## CHECK TABLE Statement

Checks if the data in the table is corrupted.

```
CHECK TABLE [db.]name
```

The `CHECK TABLE` query compares actual file sizes with the expected values which are stored on the server. If the file sizes do not match the stored values, it means the data is corrupted. This can be caused, for example, by a system crash during query execution.

The query response contains the `result` column with a single row. The row has a value of **Boolean** type:

- 0 - The data in the table is corrupted.
- 1 - The data maintains integrity.

The `CHECK TABLE` query supports the following table engines:

- `Log`
- `TinyLog`
- `StripeLog`
- `MergeTree family`

Performed over the tables with another table engines causes an exception.

Engines from the `*Log` family do not provide automatic data recovery on failure. Use the `CHECK TABLE` query to track data loss in a timely manner.

## Checking the MergeTree Family Tables

For `MergeTree family` engines, if `check_query_single_value_result` = 0, the `CHECK TABLE` query shows a check status for every individual data part of a table on the local server.

```
SET check_query_single_value_result = 0;
CHECK TABLE test_table;
```

part_path	is_passed	message
all_1_4_1	1	
all_1_4_2	1	

If `check_query_single_value_result` = 0, the `CHECK TABLE` query shows the general table check status.

```
SET check_query_single_value_result = 1;
CHECK TABLE test_table;
```

result
1

## If the Data Is Corrupted

If the table is corrupted, you can copy the non-corrupted data to another table. To do this:

1. Create a new table with the same structure as damaged table. To do this execute the query `CREATE TABLE <new_table_name> AS <damaged_table_name>`.
2. Set the `max_threads` value to 1 to process the next query in a single thread. To do this run the query `SET max_threads = 1`.
3. Execute the query `INSERT INTO <new_table_name> SELECT * FROM <damaged_table_name>`. This request copies the non-corrupted data from the damaged table to another table. Only the data before the corrupted part will be copied.
4. Restart the clickhouse-client to reset the `max_threads` value.

## その他のクエリ

### ATTACH

このクエリはまったく同じです `CREATE` でも

- 言葉の代わりに `CREATE` それは単語を使用します `ATTACH`.
- クエリはディスク上にデータを作成するのではなく、データがすでに適切な場所にあると仮定し、テーブルに関する情報をサーバーに追加するだけです。  
アタッチクエリを実行すると、サーバーはテーブルの存在を知ります。

テーブルが以前にデタッチされた場合 (`DETACH`) 、その構造が知られていることを意味する、あなたは構造を定義せずに省略形を使用することができます。

```
ATTACH TABLE [IF NOT EXISTS] [db.]name [ON CLUSTER cluster]
```

このクエリは、サーバーの起動時に使用されます。サーバーに店舗のテーブルメタデータとしてファイル `ATTACH` 単に起動時に実行されるクエリ（サーバー上で明示的に作成されるシステムテーブルを除く）。

### CHECK TABLE

表のデータが破損しているかどうかを確認します。

```
CHECK TABLE [db.]name
```

その `CHECK TABLE` queryは、実際のファイルサイズと、サーバーに格納されている期待値を比較します。ファイルサイズが格納された値と一致しない場合は、データが破損していることを意味します。これが発生する可能性があります、例えは、システムがクラッシュ時のクエリを実行します。

クエリ応答には、`result` 単一行の列。行の値は次のとおりです

**ブール値** タイプ:

- 0-テーブル内のデータが破損しています。

- 1-データは整合性を維持します。

その `CHECK TABLE` クエリは以下のテーブルエンジン:

- ログ
- TinyLog
- ストリップログ
- メルゲツリー族

これは、テーブルが別のテーブルエンジンの原因となる例外です。

からのエンジン \*Log 家族は失敗の自動データ回復を提供しない。使用する `CHECK TABLE` タイムリーにデータ損失を追跡するためのクエリ。

のために `MergeTree` ファミリーエンジン `CHECK TABLE` クエリを示すステータス確認のための個人データのテーブルに現地サーバーです。

### データが破損している場合

テーブルが破損している場合は、破損していないデータを別のテーブルにコピーできます。これを行うには:

1. 破損したテーブルと同じ構造を持つ新しいテーブルを作成します。これを行うにはクエリを実行します `CREATE TABLE <new_table_name> AS <damaged_table_name>`.
2. セット `max_threads` 単一のスレッドで次のクエリを処理するには、値を `1` に設定します。このクエリ `SET max_threads = 1`.
3. クエリの実行 `INSERT INTO <new_table_name> SELECT * FROM <damaged_table_name>`. この要求により、破損していないデータが破損した表から別の表にコピーされます。破損した部分の前のデータのみがコピーされます。
4. 再起動 `clickhouse-client` リセットするには `max_threads` 値。

## DESCRIBE TABLE

```
DESC|DESCRIBE TABLE [db.]table [INTO OUTFILE filename] [FORMAT format]
```

次の値を返します `String` タイプ列:

- `name` — Column name.
- `type` — Column type.
- `default_type` — Clause that is used in **既定の式** (`DEFAULT`, `MATERIALIZED` または `ALIAS`). 既定の式が指定されていない場合、Columnには空の文字列が含まれます。
- `default_expression` — Value specified in the `DEFAULT` 句。
- `comment_expression` — Comment text.

入れ子になったデータ構造は “`expanded`” 形式。各列は、ドットの後に名前を付けて別々に表示されます。

## DETACH

に関する情報を削除します。‘`name`’ サーバーからのテーブル。サーバーは、テーブルの存在を知ることを停止します。

```
DETACH TABLE [IF EXISTS] [db.]name [ON CLUSTER cluster]
```

テーブルのデータまたはメタデータは削除されません。次のサーバー起動時に、サーバーはメタデータを読み取り、テーブルについて再度確認します。

同様に、“`detached`” テーブルはを使用して再付することができます `ATTACH` クエリ(メタデータが格納されていないシステムテーブルを除く)。

ありません `DETACH DATABASE` クエリ。

## DROP

このクエリ慮して、調教メニューを組み立て: `DROP DATABASE` と `DROP TABLE`.

```
DROP DATABASE [IF EXISTS] db [ON CLUSTER cluster]
```

内部のすべてのテーブルを削除 ‘db’ データベースを削除します。‘db’ データベース自体。  
もし **IF EXISTS** データベースが存在しない場合、エラーは返されません。

```
DROP [TEMPORARY] TABLE [IF EXISTS] [db.]name [ON CLUSTER cluster]
```

テーブルを削除します。

もし **IF EXISTS** テーブルが存在しない場合、またはデータベースが存在しない場合、エラーは返されません。

```
DROP DICTIONARY [IF EXISTS] [db.]name
```

辞書を削除します。

もし **IF EXISTS** テーブルが存在しない場合、またはデータベースが存在しない場合、エラーは返されません。

## DROP USER

ユーザーを削除します。

### 構文

```
DROP USER [IF EXISTS] name [,....] [ON CLUSTER cluster_name]
```

## DROP ROLE

ロールを削除します。

削除された役割は、付与されたすべてのエンティティから取り消されます。

### 構文

```
DROP ROLE [IF EXISTS] name [,....] [ON CLUSTER cluster_name]
```

## DROP ROW POLICY

行ポリシーを削除します。

削除行の政策が取り消すべての主体で割り当てられます。

### 構文

```
DROP [ROW] POLICY [IF EXISTS] name [,....] ON [database.]table [,....] [ON CLUSTER cluster_name]
```

## DROP QUOTA

クオータを削除します。

削除枠が取り消すべての主体で割り当てられます。

### 構文

```
DROP QUOTA [IF EXISTS] name [,....] [ON CLUSTER cluster_name]
```

## DROP SETTINGS PROFILE

クオータを削除します。

削除権が取り消すべての主体で割り当てられます。

## 構文

```
DROP [SETTINGS] PROFILE [IF EXISTS] name [...] [ON CLUSTER cluster_name]
```

## EXISTS

```
EXISTS [TEMPORARY] [TABLE|DICTIONARY] [db.]name [INTO OUTFILE filename] [FORMAT format]
```

单一を返します UInt8-单一の値を含む列を入力します 0 テーブルまたはデータベースが存在しない場合、または 1 指定されたデータベースにテーブルが存在する場合。

## KILL QUERY

```
KILL QUERY [ON CLUSTER cluster]
  WHERE <where expression to SELECT FROM system.processes query>
  [SYNC|ASYNC|TEST]
  [FORMAT format]
```

現在実行中のクエリを強制的に終了しようとします。

終了するクエリがシステムから選択されます。で定義された基準を使用してテーブルを処理します WHERE の節 KILL クエリ。

例:

```
-- Forcibly terminates all queries with the specified query_id:
KILL QUERY WHERE query_id='2-857d-4a57-9ee0-327da5d60a90'

-- Synchronously terminates all queries run by 'username':
KILL QUERY WHERE user='username' SYNC
```

読み取り専用ユーザーは、独自のクエリのみを停止できます。

既定では、非同期バージョンのクエリが使用されます (ASYNC)、クエリが停止したことの確認を待たない。

同期バージョン (SYNC) すべてのクエリが停止するのを待機し、停止すると各プロセスに関する情報を表示します。応答には、kill\_status 列は、次の値を取ることができます:

1. 'finished' – The query was terminated successfully.
2. 'waiting' – Waiting for the query to end after sending it a signal to terminate.
3. The other values explain why the query can't be stopped.

テストクエリ (TEST) ユーザーの権限のみをチェックし、停止するクエリのリストを表示します。

## KILL MUTATION

```
KILL MUTATION [ON CLUSTER cluster]
  WHERE <where expression to SELECT FROM system.mutations query>
  [TEST]
  [FORMAT format]
```

取り消しと削除を試みます 突然変異 現在実行中です 取り消すべき突然変異はから選ばれます system.mutations によって指定されたフィルタを使用する表 WHERE の節 KILL クエリ。

テストクエリ (TEST) ユーザーの権限のみをチェックし、停止するクエリのリストを表示します。

例:

```
-- Cancel and remove all mutations of the single table:  
KILL MUTATION WHERE database = 'default' AND table = 'table'  
  
-- Cancel the specific mutation:  
KILL MUTATION WHERE database = 'default' AND table = 'table' AND mutation_id = 'mutation_3.txt'
```

The query is useful when a mutation is stuck and cannot finish (e.g. if some function in the mutation query throws an exception when applied to the data contained in the table).

突然変異によって既に行われた変更はロールバックされません。

## OPTIMIZE

```
OPTIMIZE TABLE [db.]name [ON CLUSTER cluster] [PARTITION partition | PARTITION ID 'partition_id'] [FINAL]  
[DEDUPLICATE]
```

このクエリは、テーブルエンジンを使用してテーブルのデータ部分の予定外のマージを初期化しようとします。 **マルゲツリー** 家族だ

その `OPTIMIZE` クエリは、**マテリアライズドビュー** そして、**パッファ** エンジンだ その他のテーブルエンジンなサポート。

とき `OPTIMIZE` と共に使用されます **複製マージツリー** テーブルエンジンのファミリでは、ClickHouseはマージ用のタスクを作成し、すべてのノードで実行を待機します。`replication_alter_partitions_sync` 設定が有効になっています)。

- もし `OPTIMIZE` 何らかの理由でマージを実行せず、クライアントに通知しません。通知を有効にするには、`optimize_throw_if_noop` 設定。
- を指定した場合 `PARTITION` 指定したパーティションのみが最適化されます。 **パーティション式の設定方法**。
- 指定した場合 `FINAL`、最適化は、すべてのデータが一つの部分に既にある場合でも実行されます。
- 指定した場合 `DEDUPLICATE` その後、完全に同一の行が重複除外されます（すべての列が比較されます）。

### 警告

`OPTIMIZE` できない修正 “Too many parts” エラー

## RENAME

テーブルの名前を変更します。

```
RENAME TABLE [db11.]name11 TO [db12.]name12, [db21.]name21 TO [db22.]name22, ... [ON CLUSTER cluster]
```

すべてのテーブル名変更"グローバルチェンジにおけるロックしなければなりません。テーブルの名前を変更することは簡単な操作です。TOの後に別のデータベースを指定した場合、表はこのデータベースに移動されます。しかし、そのディレクトリのデータベースに格納してある必要がある同一ファイルシステム(それ以外の場合、エラーを返す)。

## SET

```
SET param = value
```

割り当て `value` に `param` 設定 現在のセッションの場合。 変更できません **サーバー設定** こっちだ

指定した設定プロファイルのすべての値を单一のクエリで設定することもできます。

```
SET profile = 'profile-name-from-the-settings-file'
```

詳細については、[設定](#)。

## SET ROLE

現在のユーザーのロールを有効にします。

### 構文

```
SET ROLE {DEFAULT | NONE | role [...] | ALL | ALL EXCEPT role [...]}
```

## SET DEFAULT ROLE

既定のロールをユーザーに設定します。

デフォルトの役割を自動的に起動されたユーザーログインします。既定として設定できるのは、以前に付与されたロールのみです。ロールがユーザーに付与されていない場合、ClickHouseは例外をスローします。

### 構文

```
SET DEFAULT ROLE {NONE | role [...] | ALL | ALL EXCEPT role [...]} TO {user|CURRENT_USER} [...]
```

### 例

複数の既定のロールをユーザーに設定する:

```
SET DEFAULT ROLE role1, role2, ... TO user
```

付与されたすべてのロールを既定のユーザーに設定します:

```
SET DEFAULT ROLE ALL TO user
```

ユーザーからの既定の役割の削除:

```
SET DEFAULT ROLE NONE TO user
```

セットの付与の役割としてデフォルトの例外を除き、きっと:

```
SET DEFAULT ROLE ALL EXCEPT role1, role2 TO user
```

## TRUNCATE

```
TRUNCATE TABLE [IF EXISTS] [db.]name [ON CLUSTER cluster]
```

表からすべてのデータを削除します。とき句 `IF EXISTS` テーブルが存在しない場合、クエリはエラーを返します。

その `TRUNCATE` クエリはサポートされません [表示](#), [ファイル](#), [URL](#) と [Null](#) テーブルエンジン。

## USE

```
USE db
```

セッションの現在のデータベースを設定できます。

現在のデータベースは、データベースがクエリで明示的に定義されていない場合、テーブルの検索に使用されます。

セッションの概念がないため、HTTPプロトコルを使用する場合は、このクエリを実行できません。

## DESCRIBE TABLE

Returns information about table columns.

### Syntax

```
DESC|DESCRIBE TABLE [db.]table [INTO OUTFILE filename] [FORMAT format]
```

The `DESCRIBE` statement returns a row for each table column with the following **String** values:

- `name` — A column name.
- `type` — A column type.
- `default_type` — A clause that is used in the column **default expression**: `DEFAULT`, `MATERIALIZED` or `ALIAS`. If there is no default expression, then empty string is returned.
- `default_expression` — An expression specified after the `DEFAULT` clause.
- `comment` — A **column comment**.
- `codec_expression` — A **codec** that is applied to the column.
- `ttl_expression` — A **TTL** expression.
- `is_subcolumn` — A flag that equals `1` for internal subcolumns. It is included into the result only if subcolumn description is enabled by the `describe_include_subcolumns` setting.

All columns in **Nested** data structures are described separately. The name of each column is prefixed with a parent column name and a dot.

To show internal subcolumns of other data types, use the `describe_include_subcolumns` setting.

### Example

Query:

```
CREATE TABLE describe_example (
    id UInt64, text String DEFAULT 'unknown' CODEC(ZSTD),
    user Tuple (name String, age UInt8)
) ENGINE = MergeTree() ORDER BY id;

DESCRIBE TABLE describe_example;
DESCRIBE TABLE describe_example SETTINGS describe_include_subcolumns=1;
```

Result:

name	type	default_type	default_expression	comment	codec_expression
	ttl_expression				
id	UInt64	DEFAULT	'unknown'		ZSTD(1)
text	String				
user	Tuple(name String, age UInt8)				

The second query additionally shows subcolumns:

name	type	default_type	default_expression	comment	codec_expre
ssession	ttl_expression	is_subcolumn			
id	UInt64	DEFAULT	'unknown'		ZSTD(1)
text	String				
user	Tuple(name String, age UInt8)				
user.name	String			0	
user.age	UInt8			0	

## See Also

- [describe\\_include\\_subcolumns](#) setting.

# DETACH Statement

Makes the server "forget" about the existence of a table, a materialized view, or a dictionary.

## Syntax

```
DETACH TABLE|VIEW|DICTIONARY [IF EXISTS] [db.]name [ON CLUSTER cluster] [PERMANENTLY]
```

Detaching does not delete the data or metadata of a table, a materialized view or a dictionary. If an entity was not detached PERMANENTLY, on the next server launch the server will read the metadata and recall the table/view/dictionary again. If an entity was detached PERMANENTLY, there will be no automatic recall.

Whether a table or a dictionary was detached permanently or not, in both cases you can reattach them using the [ATTACH](#) query.

System log tables can be also attached back (e.g. `query_log`, `text_log`, etc). Other system tables can't be reattached. On the next server launch the server will recall those tables again.

`ATTACH MATERIALIZED VIEW` does not work with short syntax (without `SELECT`), but you can attach it using the `ATTACH TABLE` query.

Note that you can not detach permanently the table which is already detached (temporary). But you can attach it back and then detach permanently again.

Also you can not [DROP](#) the detached table, or [CREATE TABLE](#) with the same name as detached permanently, or replace it with the other table with [RENAME TABLE](#) query.

## Example

Creating a table:

Query:

```
CREATE TABLE test ENGINE = Log AS SELECT * FROM numbers(10);
SELECT * FROM test;
```

Result:

number
0
1
2
3
4
5
6
7
8
9

Detaching the table:

Query:

```
DETACH TABLE test;
SELECT * FROM test;
```

Result:

```
Received exception from server (version 21.4.1):
Code: 60. DB::Exception: Received from localhost:9000. DB::Exception: Table default.test does not exist.
```

## See Also

- [Materialized View](#)
- [Dictionaries](#)

# DROP Statements

Deletes existing entity. If the `IF EXISTS` clause is specified, these queries do not return an error if the entity does not exist.

## DROP DATABASE

Deletes all tables inside the `db` database, then deletes the `db` database itself.

Syntax:

```
DROP DATABASE [IF EXISTS] db [ON CLUSTER cluster]
```

## DROP TABLE

Deletes the table.

Syntax:

```
DROP [TEMPORARY] TABLE [IF EXISTS] [db.]name [ON CLUSTER cluster]
```

## DROP DICTIONARY

Deletes the dictionary.

Syntax:

```
DROP DICTIONARY [IF EXISTS] [db.]name
```

## DROP USER

Deletes a user.

Syntax:

```
DROP USER [IF EXISTS] name [,....] [ON CLUSTER cluster_name]
```

## DROP ROLE

Deletes a role. The deleted role is revoked from all the entities where it was assigned.

Syntax:

```
DROP ROLE [IF EXISTS] name [,....] [ON CLUSTER cluster_name]
```

## DROP ROW POLICY

Deletes a row policy. Deleted row policy is revoked from all the entities where it was assigned.

Syntax:

```
DROP [ROW] POLICY [IF EXISTS] name [,....] ON [database.]table [,....] [ON CLUSTER cluster_name]
```

## DROP QUOTA

Deletes a quota. The deleted quota is revoked from all the entities where it was assigned.

Syntax:

```
DROP QUOTA [IF EXISTS] name [,....] [ON CLUSTER cluster_name]
```

## DROP SETTINGS PROFILE

Deletes a settings profile. The deleted settings profile is revoked from all the entities where it was assigned.

Syntax:

```
DROP [SETTINGS] PROFILE [IF EXISTS] name [,....] [ON CLUSTER cluster_name]
```

## DROP VIEW

Deletes a view. Views can be deleted by a `DROP TABLE` command as well but `DROP VIEW` checks that `[db.]name` is a view.

Syntax:

```
DROP VIEW [IF EXISTS] [db.]name [ON CLUSTER cluster]
```

## DROP FUNCTION

Deletes a user defined function created by [CREATE FUNCTION](#).

System functions can not be dropped.

### Syntax

```
DROP FUNCTION [IF EXISTS] function_name
```

### Example

```
CREATE FUNCTION linear_equation AS (x, k, b) -> k*x + b;  
DROP FUNCTION linear_equation;
```

## EXISTS Statement

```
EXISTS [TEMPORARY] [TABLE|DICTIONARY] [db.]name [INTO OUTFILE filename] [FORMAT format]
```

Returns a single UInt8-type column, which contains the single value 0 if the table or database does not exist, or 1 if the table exists in the specified database.

## KILL Statements

There are two kinds of kill statements: to kill a query and to kill a mutation

### KILL QUERY

```
KILL QUERY [ON CLUSTER cluster]  
WHERE <where expression to SELECT FROM system.processes query>  
[SYNC|ASYNC|TEST]  
[FORMAT format]
```

Attempts to forcibly terminate the currently running queries.

The queries to terminate are selected from the system.processes table using the criteria defined in the WHERE clause of the KILL query.

Examples:

```
-- Forcibly terminates all queries with the specified query_id:  
KILL QUERY WHERE query_id='2-857d-4a57-9ee0-327da5d60a90'  
  
-- Synchronously terminates all queries run by 'username':  
KILL QUERY WHERE user='username' SYNC
```

Read-only users can only stop their own queries.

By default, the asynchronous version of queries is used (ASYNC), which does not wait for confirmation that queries have stopped.

The synchronous version (SYNC) waits for all queries to stop and displays information about each process as it stops.

The response contains the `kill_status` column, which can take the following values:

1. `finished` – The query was terminated successfully.
2. `waiting` – Waiting for the query to end after sending it a signal to terminate.
3. The other values explain why the query can't be stopped.

A test query (TEST) only checks the user's rights and displays a list of queries to stop.

## KILL MUTATION

```
KILL MUTATION [ON CLUSTER cluster]
  WHERE <where expression to SELECT FROM system.mutations query>
  [TEST]
  [FORMAT format]
```

Tries to cancel and remove **mutations** that are currently executing. Mutations to cancel are selected from the `system.mutations` table using the filter specified by the `WHERE` clause of the `KILL` query.

A test query (TEST) only checks the user's rights and displays a list of mutations to stop.

Examples:

```
-- Cancel and remove all mutations of the single table:
KILL MUTATION WHERE database = 'default' AND table = 'table'

-- Cancel the specific mutation:
KILL MUTATION WHERE database = 'default' AND table = 'table' AND mutation_id = 'mutation_3.txt'
```

The query is useful when a mutation is stuck and cannot finish (e.g. if some function in the mutation query throws an exception when applied to the data contained in the table).

Changes already made by the mutation are not rolled back.

## RENAME Statement

Renames databases, tables, or dictionaries. Several entities can be renamed in a single query.

Note that the `RENAME` query with several entities is non-atomic operation. To swap entities names atomically, use the `EXCHANGE` statement.

### Note

The `RENAME` query is supported by the **Atomic** database engine only.

### Syntax

```
RENAME DATABASE|TABLE|DICTIONARY name TO new_name [,...] [ON CLUSTER cluster]
```

## RENAME DATABASE

Renames databases.

### Syntax

```
RENAME DATABASE atomic_database1 TO atomic_database2 [,...] [ON CLUSTER cluster]
```

## RENAME TABLE

Renames one or more tables.

Renaming tables is a light operation. If you pass a different database after `TO`, the table will be moved to this database. However, the directories with databases must reside in the same file system. Otherwise, an error is returned.

If you rename multiple tables in one query, the operation is not atomic. It may be partially executed, and queries in other sessions may get `Table ... does not exist ...` error.

### Syntax

```
RENAME TABLE [db1.]name1 TO [db2.]name2 [,...] [ON CLUSTER cluster]
```

### Example

```
RENAME TABLE table_A TO table_A_bak, table_B TO table_B_bak;
```

## RENAME DICTIONARY

Renames one or several dictionaries. This query can be used to move dictionaries between databases.

### Syntax

```
RENAME DICTIONARY [db0.]dict_A TO [db1.]dict_B [,...] [ON CLUSTER cluster]
```

### See Also

- [Dictionaries](#)

## EXCHANGE Statement

Exchanges the names of two tables or dictionaries atomically.

This task can also be accomplished with a [RENAME](#) query using a temporary name, but the operation is not atomic in that case.

### Note

The `EXCHANGE` query is supported by the **Atomic** database engine only.

### Syntax

```
EXCHANGE TABLES|DICTIONARIES [db0.]name_A AND [db1.]name_B
```

## EXCHANGE TABLES

Exchanges the names of two tables.

### Syntax

```
EXCHANGE TABLES [db0.]table_A AND [db1.]table_B
```

## EXCHANGE DICTIONARIES

Exchanges the names of two dictionaries.

### Syntax

```
EXCHANGE DICTIONARIES [db0.]dict_A AND [db1.]dict_B
```

### See Also

- [Dictionaries](#)

## OPTIMIZE Statement

```
OPTIMIZE TABLE [db.]name [ON CLUSTER cluster] [PARTITION partition | PARTITION ID 'partition_id'] [FINAL]  
[DEDUPLICATE]
```

This query tries to initialize an unscheduled merge of data parts for tables with a table engine from the [MergeTree](#) family.

The `OPTIMIZE` query is also supported for the [MaterializedView](#) and the [Buffer](#) engines. Other table engines aren't supported.

When `OPTIMIZE` is used with the [ReplicatedMergeTree](#) family of table engines, ClickHouse creates a task for merging and waits for execution on all nodes (if the `replication_alter_partitions_sync` setting is enabled).

- If `OPTIMIZE` doesn't perform a merge for any reason, it doesn't notify the client. To enable notifications, use the [optimize\\_throw\\_if\\_noop](#) setting.
- If you specify a `PARTITION`, only the specified partition is optimized. [How to set partition expression](#).
- If you specify `FINAL`, optimization is performed even when all the data is already in one part.
- If you specify `DEDUPLICATE`, then completely identical rows will be deduplicated (all columns are compared), it makes sense only for the MergeTree engine.

### Warning

`OPTIMIZE` can't fix the "Too many parts" error.

## SET Statement

```
SET param = value
```

Assigns `value` to the param `setting` for the current session. You cannot change [server settings](#) this way.

You can also set all the values from the specified settings profile in a single query.

```
SET profile = 'profile-name-from-the-settings-file'
```

For more information, see [Settings](#).

## SET ROLE Statement

Activates roles for the current user.

```
SET ROLE {DEFAULT | NONE | role [...] | ALL | ALL EXCEPT role [...]}
```

## SET DEFAULT ROLE

Sets default roles to a user.

Default roles are automatically activated at user login. You can set as default only the previously granted roles. If the role isn't granted to a user, ClickHouse throws an exception.

```
SET DEFAULT ROLE {NONE | role [...] | ALL | ALL EXCEPT role [...]} TO {user|CURRENT_USER} [...]
```

## Examples

Set multiple default roles to a user:

```
SET DEFAULT ROLE role1, role2, ... TO user
```

Set all the granted roles as default to a user:

```
SET DEFAULT ROLE ALL TO user
```

Purge default roles from a user:

```
SET DEFAULT ROLE NONE TO user
```

Set all the granted roles as default excepting some of them:

```
SET DEFAULT ROLE ALL EXCEPT role1, role2 TO user
```

## TRUNCATE Statement

```
TRUNCATE TABLE [IF EXISTS] [db.]name [ON CLUSTER cluster]
```

Removes all data from a table. When the clause `IF EXISTS` is omitted, the query returns an error if the table does not exist.

The TRUNCATE query is not supported for [View](#), [File](#), [URL](#), [Buffer](#) and [Null](#) table engines.

You can use the `replication_alter_partitions_sync` setting to set up waiting for actions to be executed on replicas.

You can specify how long (in seconds) to wait for inactive replicas to execute TRUNCATE queries with the `replication_wait_for_inactive_replica_timeout` setting.

## Note

If the `replication_alter_partitions_sync` is set to 2 and some replicas are not active for more than the time, specified by the `replication_wait_for_inactive_replica_timeout` setting, then an exception `UNFINISHED` is thrown.

## USE Statement

```
USE db
```

Lets you set the current database for the session.

The current database is used for searching for tables if the database is not explicitly defined in the query with a dot before the table name.

This query can't be made when using the HTTP protocol, since there is no concept of a session.

## WATCH Statement (Experimental)

### Important

This is an experimental feature that may change in backwards-incompatible ways in the future releases.

Enable live views and `WATCH` query using `set allow_experimental_live_view = 1`.

```
WATCH [db.]live_view  
[EVENTS]  
[LIMIT n]  
[FORMAT format]
```

The `WATCH` query performs continuous data retrieval from a `LIVE VIEW` table. Unless the `LIMIT` clause is specified it provides an infinite stream of query results from a `LIVE VIEW`.

```
WATCH [db.]live_view [EVENTS] [LIMIT n] [FORMAT format]
```

## Virtual columns

The virtual `_version` column in the query result indicates the current result version.

### Example:

```
CREATE LIVE VIEW lv WITH REFRESH 5 AS SELECT now();  
WATCH lv;
```

```

now()---version
2021-02-21 09:17:21 |   1 |
                   |
now()---version
2021-02-21 09:17:26 |   2 |
                   |
now()---version
2021-02-21 09:17:31 |   3 |
                   |
...

```

By default, the requested data is returned to the client, while in conjunction with **INSERT INTO** it can be forwarded to a different table.

#### **Example:**

```
INSERT INTO [db.]table WATCH [db.]live_view ...
```

## EVENTS Clause

The **EVENTS** clause can be used to obtain a short form of the **WATCH** query where instead of the query result you will just get the latest query result version.

```
WATCH [db.]live_view EVENTS;
```

#### **Example:**

```
CREATE LIVE VIEW lv WITH REFRESH 5 AS SELECT now();
WATCH lv EVENTS;
```

```

version
 1 |
version
 2 |
...

```

## LIMIT Clause

The **LIMIT n** clause specifies the number of updates the **WATCH** query should wait for before terminating. By default there is no limit on the number of updates and therefore the query will not terminate. The value of 0 indicates that the **WATCH** query should not wait for any new query results and therefore will return immediately once query result is evaluated.

```
WATCH [db.]live_view LIMIT 1;
```

#### **Example:**

```
CREATE LIVE VIEW lv WITH REFRESH 5 AS SELECT now();
WATCH lv EVENTS LIMIT 1;
```

```

version
 1 |

```

# FORMAT Clause

The `FORMAT` clause works the same way as for the `SELECT`.

## Note

The `JSONEachRowWithProgress` format should be used when watching `LIVE VIEW` tables over the HTTP interface. The progress messages will be added to the output to keep the long-lived HTTP connection alive until the query result changes. The interval between progress messages is controlled using the `live_view_heartbeat_interval` setting.

## 構文

システムのパーサーには、完全SQLパーサー(再帰的降下パーサー)とデータ形式パーサー(高速ストリームパーサー)の二つのタイプがあります。

を除くすべてのケースで `INSERT` 完全なSQLパーサーのみが使用されます。

その `INSERT` クエリの両方を使用のパーサ:

```
INSERT INTO t VALUES (1, 'Hello, world'), (2, 'abc'), (3, 'def')
```

その `INSERT INTO t VALUES` フラグメントは完全なパーサーによって解析され、データは `(1, 'Hello, world'), (2, 'abc'), (3, 'def')` 高速ストリームパーサーによって解析されます。また、データの完全なパーサーをオンにするには `input_format_values_interpret_expressions` 設定。とき `input_format_values_interpret_expressions = 1`, ClickHouse はまず、高速ストリームパーサーで値を解析しようとします。失敗した場合、ClickHouseはデータに対して完全なパーサーを使用し、SQLのように扱います 式.

データの形式は任意です。クエリが受信されると、サーバーは以下の値を計算しません `max_query_size` 要求のバイトはRAM(デフォルトでは1MB)で、残りはストリーム解析されます。

これを回避する問題の大き `INSERT` クエリ。

を使用する場合 `Values` フォーマット `INSERT` これは、データがaの式と同じように解析されるように見えるかもしれません `SELECT` クエリが、これは真実ではありません。その `Values` 形式ははるかに限られています。

この記事の残りの部分は完全なパーサーをカバーします。フォーマットパーサーの詳細については、[形式セクション](#)

## スペース

構文構成（クエリの開始と終了を含む）の間には、任意の数のスペースシンボルがあります。スペース記号には、スペース、タブ、改行、CR、およびフォームフィードがあります。

## コメント

ClickHouse支援のいずれかのSQL型は、Cスタイルのコメント。

SQLスタイルのコメントで始まる -- そして、行の終わり、後のスペースに進みます -- 省略できる。

Cスタイルは /\* に \*/ そして複数行にすることができ、スペースも必要ありません。

## キーワード

キーワードでは、大文字と小文字が区別されません:

- SQL標準。例えば, `SELECT`, `select` と `SeLeCt` すべて有効です。
- いくつかの一般的なDBMS (MySQLまたはPostgres) での実装。例えば, `DateTime` と同じです `datetime`.

データ型名で大文字と小文字が区別されるかどうかは、`system.data_type_families` テーブル。

標準SQLとは対照的に、他のすべてのキーワード(関数名を含む)は次のとおりです 大文字と小文字を区別。

キーワードは予約されていません。を使用する場合 識別子 キーワードと同じ名前で、二重引用符またはバッククオートで囲みます。たとえば、次のクエリです `SELECT "FROM" FROM table_name` テーブルの場合は有効です `table_name` 名前を持つ列があります `"FROM"`。

## 識別子

識別子は:

- クラスターデータベース、テーブル、パーティション、カラム名になってしまいます
- 機能。
- データ型。
- 式エイリアス.

識別子で引用することは非引用されます。後者が好ましい。

非引用識別子に一致しなければならな`regex ^[a-zA-Z_][0-9a-zA-Z_]*$` と等しくすることはできません キーワード. 例: `x, _1, X_y_Z123_`.

キーワードと同じ識別子を使用する場合や、識別子に他の記号を使用する場合は、二重引用符またはバッククオートを使用して引用符で囲みます。`, "id", `id``.

## リテラル

数値、文字列、複合、および `NULL` リテラル。

### 数値

数値リテラルが解析されようとなります:

- まず、64ビット符号付きの数値として、`ストルトゥール` 機能。
- 失敗した場合は、64ビット符号なしの数値として、`strtoll` 機能。
- 失敗した場合は、浮動小数点数として `strtod` 機能。
- それ以外の場合は、エラーを返します。

リテラル値は、値が収まる最小の型を持ちます。

たとえば、1は次のように解析されます `UInt8` しかし、256は次のように解析されます `UInt16`. 詳細については、データ型。

例: `1, 18446744073709551615, 0xDEADBEEF, 01, 0.1, 1e100, -1e-100, inf, nan.`

### 文字列

单一引`quotes`の文字列リテラルのみがサポートされます。囲まれた文字はバックスラッシュエスケープできます。以下のエスケープシーケンスに対応する特殊な値: `\b, \f, \r, \n, \t, \0, \a, \v, \xHH`. それ以外の場合は、エスケープシーケンスの形式で `\c`, ここで `c` は任意の文字であり、`c` つまり、シーケンスを使用できます `'\`` と `\\\``. この値は 文字列 タイプ。

文字列リテラルでは、少なくとも ' と `\``. 单一引`quotes`は、单一引`quote`、リテラルでエスケープできます `'It\'s'` と `'It\"s'` 等しい。

### 化合物

配列は角括弧で構成されます [1, 2, 3]. Nuplesは丸括弧で構成されています (1, 'Hello, world!', 2).

技術的には、これらはリテラルではなく、それぞれ配列作成演算子とタプル作成演算子を持つ式です。

配列は少なくとも一つの項目で構成され、組は少なくとも二つの項目を持つ必要があります。

タプルが表示される場合は別のケースがあります IN a の節 SELECT クエリ。クエリ結果には組を含めることができますが、組をデータベースに保存することはできません メモリ エンジン) )。

## NULL

値が欠落していることを示します。

保存するために NULL テーブルフィールドでは、Null可能 タイプ。

データ形式（入力または出力）に応じて), NULL 異なる表現を有していてもよい。 詳細については、以下の文書を参照してください [データ形式](#).

処理に多くのニュアンスがあります NULL. たとえば、比較演算の引数のうち少なくともいずれかが次のようになっている場合 NULL この操作の結果も NULL. 乗算、加算、およびその他の演算についても同様です。 詳細については、各操作のドキュメントを参照してください。

クエリでは、以下を確認できます NULL を使用して IS NULL と IS NOT NULL 演算子と関連する関数isNull と isNotNull.

## 関数

関数呼び出しは、引数のリスト（空の場合もあります）を丸括弧で囲んだ識別子のように書かれます。標準のSQLとは対照的に、空の引数リストであっても角かっこが必要です。例: now().

正規関数と集計関数があります（セクションを参照 “Aggregate functions”）。一部の集計関数を含むことができ二つのリストの引数セットに固定して使用します。例: quantile (0.9) (x). これらの集計関数は “parametric” 関数と最初のリストの引数が呼び出されます “parameters”. パラメータのない集計関数の構文は、通常の関数と同じです。

## 演算子

演算子は、クエリの解析中に、優先順位と連想を考慮して、対応する関数に変換されます。

たとえば、次の式は `1 + 2 * 3 + 4` に変換されます `plus(plus(1, multiply(2, 3)), 4)`.

## データの種類とデータベースのテーブルエンジン

のデータ型とテーブルエンジン CREATE クエリは、識別子または関数と同じ方法で記述されます。つまり、かっこ内に引数リストを含むことも、含まないこともできます。 詳細については “Data types,” “Table engines,” と “CREATE”.

## 式エイリアス

別名は、クエリ内の式のユーザ一定義名です。

### expr AS alias

- AS — The keyword for defining aliases. You can define the alias for a table name or a column name in a SELECT を使用せずに句 AS キーワード。

For example, `SELECT table\_name\_alias.column\_name FROM table\_name table\_name\_alias`.

In the [CAST](#sql-reference-sql\_reference-functions-type\_conversion\_functions-md) function, the `AS` keyword has another meaning. See the description of the function.

- expr — Any expression supported by ClickHouse.

For example, `SELECT column\_name \* 2 AS double FROM some\_table`.

- alias — Name for expr. エイリアスは 識別子 構文。

```
For example, `SELECT "table t".column_name FROM table_name AS "table t"`. 
```

## 使用上の注意

エイリアスは、クエリまたはサブクエリのグローバルであり、任意の式のクエリの任意の部分でエイリアスを定義できます。例えば, `SELECT (1 AS n) + 2, n`

エイリアスは、サブクエリおよびサブクエリ間では表示されません。たとえば、クエリの実行中に `SELECT (SELECT sum(b.a) + num FROM b) - a.a AS num FROM a` ClickHouseは例外を生成します `Unknown identifier: num.`

の結果列に対してエイリアスが定義されている場合 `SELECT` サブクエリの句は、これらの列は、外部クエリで表示されます。例えば, `SELECT n + m FROM (SELECT 1 AS n, 2 AS m)`

列名またはテーブル名と同じ別名に注意してください。次の例を考えてみましょう:

```
CREATE TABLE t
(
    a Int,
    b Int
)
ENGINE = TinyLog()
```

```
SELECT
    argMax(a, b),
    sum(b) AS b
FROM t
```

```
Received exception from server (version 18.14.17):
Code: 184. DB::Exception: Received from localhost:9000, 127.0.0.1. DB::Exception: Aggregate function sum(b) is
found inside another aggregate function in query.
```

この例では、テーブルを宣言しました `t` 列付き `b`. 次に、データを選択するときに、`sum(b) AS b` 別名だとしてエイリアスは、グローバルClickHouse置換されているリテラル `b` 式では `argMax(a, b)` 式を使って `sum(b)`. この置換により例外が発生しました。

## アスタリスク

で `SELECT` クエリー、アスタリスクで置き換え異なるアイコンで表示されます。詳細については “`SELECT`”.

## 式

式は、関数、識別子、リテラル、演算子の適用、角かっこ内の式、サブクエリ、またはアスタリスクです。別名を含めることもできます。

式のリストは、カンマで区切られた一つ以上の式です。

関数と演算子は、引数として式を持つことができます。

# Distributed DDL Queries (ON CLUSTER Clause)

By default the `CREATE`, `DROP`, `ALTER`, and `RENAME` queries affect only the current server where they are executed. In a cluster setup, it is possible to run such queries in a distributed manner with the `ON CLUSTER` clause.

For example, the following query creates the `all_hits` Distributed table on each host in `cluster`:

```
CREATE TABLE IF NOT EXISTS all_hits ON CLUSTER cluster (p Date, i Int32) ENGINE = Distributed(cluster, default, hits)
```

In order to run these queries correctly, each host must have the same cluster definition (to simplify syncing configs, you can use substitutions from ZooKeeper). They must also connect to the ZooKeeper servers.

The local version of the query will eventually be executed on each host in the cluster, even if some hosts are currently not available.

## Warning

The order for executing queries within a single host is guaranteed.

## 関数

通常の関数（それらは単に呼び出されます）- 関数の少なくとも\*二つの種類があります “functions” ) and aggregate functions. These are completely different concepts. Regular functions work as if they are applied to each row separately (for each row, the result of the function doesn't depend on the other rows). Aggregate functions accumulate a set of values from various rows (i.e. they depend on the entire set of rows).

本項では通常の関数について説明する。そのための集計関数の項をご参照ください “Aggregate functions”.

\*-機能の第三のタイプがあります ‘arrayJoin’ テーブル関数は別にも言及することができます。\*

## 強力なタイピング

標準のSQLとは対照的に、ClickHouseには強力な型指定があります。言い換えれば、型間で暗黙的な変換は行われません。各関数は、特定の型セットに対して機能します。つまり、型変換関数を使用する必要があることがあります。

## 共通の部分式の除去

同じAST(同じレコードまたは構文解析結果)を持つクエリ内のすべての式は、同じ値を持つと見なされます。このような式は連結され、一度実行されます。同一のサブクエリもこの方法で削除されます。

## 結果のタイプ

すべての関数は、結果として单一の戻り値を返します（複数の値ではなく、ゼロ値ではありません）。結果の型は、通常、値ではなく、引数の型によってのみ定義されます。例外は、tupleElement関数(a.N演算子)とtoFixedString関数です。

## 定数

簡単にするために、特定の関数は、いくつかの引数の定数のみで動作できます。たとえば、LIKE演算子の右引数は定数でなければなりません。

ほとんどすべての関数は、定数引数の定数を返します。例外は、乱数を生成する関数です。

その ‘now’ 関数は、異なる時間に実行されたクエリに対して異なる値を返しますが、定数は单一のクエリ内でのみ重要であるため、結果は定数と見なされます。

定数式は定数とも見なされます（たとえば、LIKE演算子の右半分は複数の定数から構成できます）。

関数は、定数と非定数引数のために異なる方法で実装することができます（異なるコードが実行されます）。しかし、定数と同じ値だけを含む真の列の結果は、お互いに一致する必要があります。

## ヌル処理

関数の動作は次のとおりです:

- 関数の引数の少なくとも一つが `NULL`、関数の結果もあります `NULL`.
- 各関数の説明で個別に指定される特殊な動作。ClickHouseのソースコードでは、これらの関数は `UseDefaultImplementationForNulls=false`.

## 不变性

Functions can't change the values of their arguments – any changes are returned as the result. Thus, the result of calculating separate functions does not depend on the order in which the functions are written in the query.

## エラー処理

データが無効な場合、一部の関数は例外をスローすることができます。この場合、クエリはキャンセルされ、エラーテキストがクライアントに返されます。分散処理の場合、いずれかのサーバーで例外が発生すると、他のサーバーもクエリを中止しようとします。

## 引数式の評価

ほぼすべてのプログラミング言語の一つの引数が評価される。これは通常、演算子です `&&`, `||`, `and` `?:`。  
しかし、ClickHouseでは、関数（演算子）の引数は常に評価されます。これは、各行を別々に計算するのではなく、列の部分全体が一度に評価されるためです。

## 分散クエリ処理のための関数の実行

分散クエリ処理の場合、リモートサーバーではできるだけ多くのステージのクエリ処理が実行され、残りのステージ（中間結果とそれ以降のすべてをマージ）はリクエ

つまり、異なるサーバーで機能を実行できます。

たとえば、クエリでは `SELECT f(sum(g(x))) FROM distributed_table GROUP BY h(y),`

- もし `distributed_table` 少なくとも二つの破片を持っている、機能 ‘g’ と ‘h’ は、リモートサーバ上で実行され、‘f’ 要求元サーバー上で実行されます。
- もし `distributed_table` 一つだけの破片、すべてを持っています ‘f’, ‘g’, and ‘h’ 機能は、このシャードのサーバー上で実行されます。

通常、関数の結果は、実行されるサーバーに依存しません。しかし、時にはこれが重要です。

たとえば、辞書を操作する関数は、それらが実行されているサーバー上に存在する辞書を使用します。

別の例は、`hostName` 実行しているサーバーの名前を返す関数 `GROUP BY a` のサーバーによって `SELECT` クエリ。

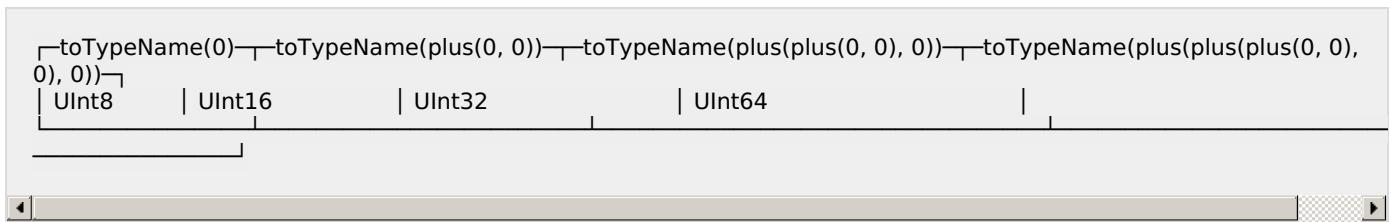
クエリ内の関数がリクエスタサーバで実行されているが、リモートサーバで実行する必要がある場合は、リクエスタサーバにラップすることができます。‘any’ 関数を集計するか、キーに追加します `GROUP BY`.

## 算術関数

すべての算術関数について、結果の型は、そのような型がある場合、結果が収まる最小の数値型として計算されます。最小値は、ビット数、署名されているかどうか、および浮動小数点数に基づいて同時に取得されます。十分なビットがない場合は、最も高いビットタイプが取られます。

例:

```
SELECT toTypeName(0), toTypeName(0 + 0), toTypeName(0 + 0 + 0), toTypeName(0 + 0 + 0 + 0)
```



算術関数は、UInt8、UInt16、UInt32、UInt64、Int8、Int16、Int32、Int64、Float32、Float64のいずれかの型のペアに対しても機能します。

オーバーフローは、C++と同じ方法で生成されます。

## プラス (a、b) 、a+b演算子

数値の合計を計算します。

また、日付または日付と時刻を持つ整数を追加することもできます。日付の場合、整数を追加することは、対応する日数を追加することを意味します。日付と時刻の場合は、対応する秒数を加算することを意味します。

## マイナス(a,b),a-b演算子

差を計算します。結果は常に署名されます。

You can also calculate integer numbers from a date or date with time. The idea is the same – see above for ‘plus’.

## 乗算(a,b),a\*b演算子

数値の積を計算します。

## 除算(a,b)、a/b演算子

数値の商を計算します。結果の型は常に浮動小数点型です。

整数除算ではありません。整数除算の場合は、‘intDiv’ 機能。

ゼロで割ると ‘inf’, ‘-inf’, または ‘nan’.

## intDiv(a,b)

数値の商を計算します。整数に分割し、（絶対値で）丸めます。

例外は、ゼロで除算するとき、または最小の負の数をマイナスで除算するときにスローされます。

## intDivOrZero(a,b)

とは異なる ‘intDiv’ ゼロで除算するとき、または最小の負の数をマイナスで除算するときにゼロを返します。

## modulo(a,b),a%b演算子

除算後の剰余を計算します。

引数が浮動小数点数の場合は、小数点部分を削除することによって整数に事前変換されます。

残りはC++と同じ意味で取られます。切り捨て除算は、負の数に使用されます。

例外は、ゼロで除算するとき、または最小の負の数をマイナスで除算するときにスローされます。

## moduloOrZero(a,b)

とは異なる ‘modulo’ 除数がゼロのときにゼロを返すという点で。

## 否定(a),-演算子

逆符号で数値を計算します。結果は常に署名されます。

## **abs(a)**

数値 ( a ) の絶対値を算出する。つまり、 $a < 0$  の場合、 $-a$  を返します。符号付き整数型の場合、符号なしの数値を返します。

## **gcd(a,b)**

数値の最大公約数を返します。

例外は、ゼロで除算するとき、または最小の負の数をマイナスで除算するときにスローされます。

## **lcm(a,b)**

数値の最小公倍数を返します。

例外は、ゼロで除算するとき、または最小の負の数をマイナスで除算するときにスローされます。

## **比較関数**

比較関数は常に 0 または 1 (UInt8) を返します。

次のタイプを比較できます:

- 数字
- 文字列と固定文字列
- 日付
- 日付と時刻

各グループ内ではなく、異なるグループ間。

たとえば、日付と文字列を比較することはできません。文字列を日付に変換するには関数を使用する必要があります。

文字列はバイト単位で比較されます。短い文字列は、それで始まり、少なくとも一つ以上の文字を含むすべての文字列よりも小さくなります。

**等しい、 $a=b$  および  $a==b$  演算子**

ノートイコライザー、 $a!$  演算子  $=b$  および  $a \neq b$

**less,  $\lt$  演算子**

**より大きい、 $\gt$  演算子**

**lessOrEquals,  $\leq$  演算子**

**greaterOrEquals,  $\geq$  演算子**

## **論理関数**

論理関数は任意の数値型を受け入れますが、UInt8 は 0 または 1 に等しい数値を返します。

引数としてのゼロが考慮されます “false,” ゼロ以外の値は考慮されますが “true”.

**and, and 演算子**

**or, OR 演算子**

# ない、ない演算子

## xor

## 型変換関数

### 数値変換の一般的な問題

あるデータ型から別のデータ型に値を変換する場合、一般的なケースでは、データ損失につながる危険な操作であることを覚えておく必要があります。大きいデータ型から小さいデータ型に値を近似しようとする場合、または異なるデータ型間で値を変換する場合、データ損失が発生する可能性があります。

クリックハウスは C++ プログラムと同じ動作。

### トイント (8/16/32/64)

入力値を Int データ型。この関数ファミ：

- `toInt8(expr)` — Results in the `Int8` データ型。
- `toInt16(expr)` — Results in the `Int16` データ型。
- `toInt32(expr)` — Results in the `Int32` データ型。
- `toInt64(expr)` — Results in the `Int64` データ型。

パラメータ

- `expr` — 式 数値または数値の十進表現を持つ文字列を返します。数値の二進表現、八進表現、進表現はサポートされていません。先頭のゼロは削除されます。

戻り値

の整数値 `Int8`, `Int16`, `Int32`, または `Int64` データ型。

関数の使用 ゼロへの丸め つまり、数字の小数部の数字を切り捨てます。

に対する関数の振る舞い `Nan` および `Inf` 引数は未定義です。覚えておいて [数値変換の問題](#)、関数を使用する場合。

例

```
SELECT toInt64(nan), toInt32(32), toInt16('16'), toInt8(8.8)
```

The diagram shows the number -9223372036854775808 being broken down into four parts corresponding to the output sizes of the conversion functions:

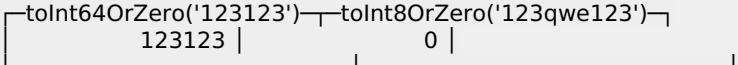
- ToInt64(nan) covers the entire 64-bit range.
- ToInt32(32) covers the 32-bit range from -2^31 to 2^31 - 1.
- ToInt16('16') covers the 16-bit range from -2^15 to 2^15 - 1.
- ToInt8(8.8) covers the 8-bit range from -2^7 to 2^7 - 1.

### toInt(8/16/32/64)OrZero

String型の引数を取り、それをIntに解析しようとします(8 | 16 | 32 | 64)。失敗した場合は0を返します。

例

```
select toInt64OrZero('123123'), toInt8OrZero('123qwe123')
```

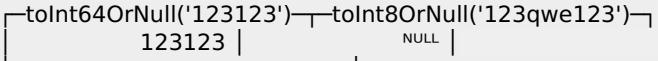


## toInt(8/16/32/64)OrNull

String型の引数を取り、それをIntに解析しようとします(8 | 16 | 32 | 64). 失敗した場合はNULLを返します。

例

```
select toInt64OrNull('123123'), toInt8OrNull('123qwe123')
```



## トウイント (8/16/32/64)

入力値を UInt データ型。この関数ファミ:

- `toUInt8(expr)` — Results in the UInt8 データ型。
- `toUInt16(expr)` — Results in the UInt16 データ型。
- `toUInt32(expr)` — Results in the UInt32 データ型。
- `toUInt64(expr)` — Results in the UInt64 データ型。

パラメータ

- `expr` — 式 数値または数値の十進表現を持つ文字列を返します。数値の二進表現、八進表現、進表現はサポートされていません。先頭のゼロは削除されます。

戻り値

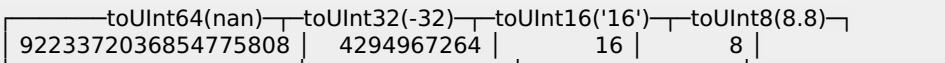
の整数値 UInt8, UInt16, UInt32, または UInt64 データ型。

関数の使用 ゼロへの丸め つまり、数字の小数部の数字を切り捨てます。

負の関数に対する関数の振る舞いと `Nan` および `Inf` 引数は未定義です。負の数の文字列を渡すと、次のようにになります '`-32`', ClickHouseは例外を発生させます。覚えておいて [数値変換の問題](#)、関数を使用する場合。

例

```
SELECT toUInt64(nan), toUInt32(-32), toUInt16('16'), toUInt8(8.8)
```



## トウイント (8/16/32/64)オルゼロ

## トウイント (8/16/32/64)OrNull

## トフロア (32/64)

`toFloat(32/64)OrZero`

`toFloat(32/64)OrNull`

東立（とうだて

トダテオルゼロ

`toDateOrNull`

`dateTime`

トダティメオルゼロ

`dateTimeOrNull`

トデシマル(32/64/128)

変換 `value` に 小数点 精度の高いデータ型 `S`. その `value` 数値または文字列を指定できます。その `S (scale)` パラメータは、小数点以下の桁数を指定します。

- `toDecimal32(value, S)`
- `toDecimal64(value, S)`
- `toDecimal128(value, S)`

`decimal(32/64/128)OrNull`

入力文字列を `a` に変換します `Nullable(Decimal(P,S))` データ型の値。このファミリの機能など:

- `toDecimal32OrNull(expr, S)` — Results in `Nullable(Decimal32(S))` データ型。
- `toDecimal64OrNull(expr, S)` — Results in `Nullable(Decimal64(S))` データ型。
- `toDecimal128OrNull(expr, S)` — Results in `Nullable(Decimal128(S))` データ型。

これらの関数は、`toDecimal*()` あなたが得ることを好むならば、関数 `NULL` 入力値の解析エラーが発生した場合の例外ではなく、値。

パラメータ

- `expr` — 式 の値を返します。文字列 データ型。ClickHouseは、十進数のテキスト表現を想定しています。例えば, '`1.111`'.
- `S` — Scale, the number of decimal places in the resulting value.

戻り値

の値 `Nullable(Decimal(P,S))` データ型。この値を含む:

- との数 `S` ClickHouseが入力文字列を数値として解釈する場合は、小数点以下の桁数。
- `NULL`、ClickHouseが入力文字列を数値として解釈できない場合、または入力番号に以下のものが含まれている場合 `S` 小数点以下の桁数。

例

```
SELECT toDecimal32OrNull(toString(-1.111), 5) AS val, toTypeName(val)
```

```
val ->toTypeName(toDecimal32OrNull(toString(-1.111), 5)) ->
-1.11100 | Nullable(Decimal(9, 5))
```

```
SELECT toDecimal32OrNull(toString(-1.111), 2) AS val, toTypeName(val)
```

```
val ->toTypeName(toDecimal32OrNull(toString(-1.111), 2)) ->
NULL | Nullable(Decimal(9, 2))
```

## toDecimal(32/64/128)OrZero

入力値を 小数(P,S) データ型。このファミリの機能など:

- `toDecimal32OrZero(expr, S)` — Results in `Decimal32(S)` データ型。
- `toDecimal64OrZero(expr, S)` — Results in `Decimal64(S)` データ型。
- `toDecimal128OrZero(expr, S)` — Results in `Decimal128(S)` データ型。

これらの関数は、`toDecimal*`() あなたが得ることを好むならば、関数 0 入力値の解析エラーが発生した場合の例外ではなく、値。

パラメータ

- `expr` — 式 の値を返します。文字列 データ型。ClickHouseは、十進数のテキスト表現を想定しています。例えば、'1.111'。
- `S` — Scale, the number of decimal places in the resulting value.

戻り値

の値 `Nullable(Decimal(P,S))` データ型。この値を含む:

- との数 `S` ClickHouseが入力文字列を数値として解釈する場合は、小数点以下の桁数。
- 0とともに `S` ClickHouseが入力文字列を数値として解釈できない場合、または入力番号に以下のものが含まれている場合は、小数点以下の桁数 `S` 小数点以下の桁数。

例

```
SELECT toDecimal32OrZero(toString(-1.111), 5) AS val, toTypeName(val)
```

```
val ->toTypeName(toDecimal32OrZero(toString(-1.111), 5)) ->
-1.11100 | Decimal(9, 5)
```

```
SELECT toDecimal32OrZero(toString(-1.111), 2) AS val, toTypeName(val)
```

```
val ->toTypeName(toDecimal32OrZero(toString(-1.111), 2)) ->
0.00 | Decimal(9, 2)
```

## toString

数値、文字列（ただし、固定文字列ではありません）、日付、および時刻を持つ日付の間で変換するための関数。これら全ての機能を受け入れを一つの引数。

文字列との間で変換する場合、値は**TabSeparated**形式(およびほとんどすべての他のテキスト形式)と同じ規則を使用して書式設定または解析されます。文字列を解析できない場合は、例外がスローされ、要求がキャンセルされます。

日付を数値に変換する場合、またはその逆の場合、日付はUnixエポックの開始からの日数に対応します。

時刻付きの日付を数値に変換する場合、またはその逆の場合、時刻付きの日付はUnixエポックの開始からの秒数に対応します。

**ToDate/toDateTime**関数の日付および日付と時刻の形式は、次のように定義されます:

```
YYYY-MM-DD  
YYYY-MM-DD hh:mm:ss
```

例外として、**uint32**、**Int32**、**UInt64**、または**Int64**の数値型から**Date**に変換する場合、数値が**65536**以上の場合、数値はUnixタイムスタンプとして解釈され(日数ではなく)、日付 この支援のための共通の発生を書く'**toDate(unix\_timestamp)**' それ以外の場合はエラーになり、より面倒な書き込みが必要になります '**toDate(toDateTime(unix\_timestamp))**'.

日付と時刻の間の変換は、**null**時間を追加するか、時間を削除することによって、自然な方法で実行されます。

数値型間の変換では、C++の異なる数値型間の代入と同じ規則が使用されます。

さらに、**DateTime**引数の**toString**関数は、タイムゾーンの名前を含む第二の文字列引数を取ることができます。例:  
**Asia/Yekaterinburg** この場合、時刻は指定されたタイムゾーンに従って書式設定されます。

```
SELECT  
now() AS now_local,  
toString(now(), 'Asia/Yekaterinburg') AS now_yekat
```

now_local	now_yekat
2016-06-15 00:11:21	2016-06-15 02:11:21

また、**toUnixTimestamp** 機能。

## toFixedString(s,N)

文字列型の引数を**FixedString(N)**型(固定長Nの文字列)に変換します。 Nは定数でなければなりません。

文字列のバイト数がNより少い場合は、右側に**null**バイトが埋め込まれます。 文字列のバイト数がNより多い場合は、例外がスローされます。

## トストリングカットゼロ(s)

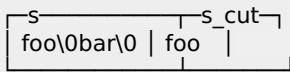
文字列または**FixedString**引数を受け取ります。 見つかった最初のゼロバイトで切り捨てられた内容の文字列を返します。

例:

```
SELECT toFixedString('foo', 8) AS s, toStringCutToZero(s) AS s_cut
```

s	s_cut
foo\0\0\0\0\0   foo	

```
SELECT toFixedString('foo\0bar', 8) AS s, toStringCutToZero(s) AS s_cut
```



## 再解釈 (8/16/32/64)

### 再解釈(8/16/32/64)

#### reinterpretAsFloat(32/64)

#### 再解釈日

#### データの再解釈

これらの関数は、文字列を受け入れ、文字列の先頭に置かれたバイトをホスト順(リトルエンディアン)の数値として解釈します。文字列が十分な長さでない場合、関数は文字列がnullバイトの必要な数で埋め込まれているかのように動作します。文字列が必要以上に長い場合、余分なバイトは無視されます。日付はUnixエポックの開始からの日数として解釈され、時刻を持つ日付はUnixエポックの開始からの秒数として解釈されます。

## 再解釈

この関数は、数値または日付または日付と時刻を受け入れ、対応する値をホスト順(リトルエンディアン)で表すバイトを含む文字列を返します。Nullバイトは末尾から削除されます。たとえば、UInt32型の値255は、バイト長の文字列です。

## reinterpretAsFixedString

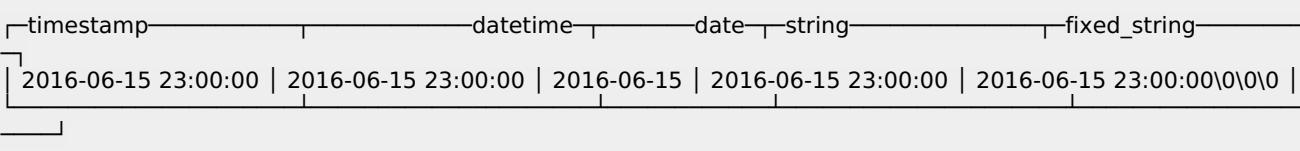
この関数は、数値または日付または日付と時刻を受け取り、ホスト順(リトルエンディアン)で対応する値を表すバイトを含むFixedStringを返します。Nullバイトは末尾から削除されます。たとえば、UInt32型の値255は、バイト長のFixedStringです。

## キヤスト (x,T)

変換 'x' に 't' データ型。構文CAST(x AS t)もサポートされています。

例:

```
SELECT
  '2016-06-15 23:00:00' AS timestamp,
  CAST(timestamp AS DateTime) AS datetime,
  CAST(timestamp AS Date) AS date,
  CAST(timestamp, 'String') AS string,
  CAST(timestamp, 'FixedString(22)') AS fixed_string
```



FixedString(N)への変換は、String型またはFixedString(N)型の引数に対してのみ機能します。

タイプ変換 Null可能 そして背部は支えられる。例:

```
SELECT toTypeName(x) FROM t_null
```

```
toTypeName(x)
└─ Int8
  └─ Int8
```

```
SELECT toTypeName(CAST(x, 'Nullable(UInt16)')) FROM t_null
```

```
toTypeName(CAST(x, 'Nullable(UInt16)'))
└─ Nullable(UInt16)
  └─ Nullable(UInt16)
```

## toInterval(年/四半期/月/週/日/時/分/秒)

数値型引数を **間隔** データ型。

### 構文

```
toIntervalSecond(number)
toIntervalMinute(number)
toIntervalHour(number)
toIntervalDay(number)
toIntervalWeek(number)
toIntervalMonth(number)
toIntervalQuarter(number)
toIntervalYear(number)
```

### パラメータ

- **number** — Duration of interval. Positive integer number.

### 戻り値

- の値 **Interval** データ型。

### 例

```
WITH
  toDate('2019-01-01') AS date,
  INTERVAL 1 WEEK AS interval_week,
  toIntervalWeek(1) AS interval_to_week
SELECT
  date + interval_week,
  date + interval_to_week
```

```
plus(date, interval_week) ── plus(date, interval_to_week) ──
  2019-01-08 |           2019-01-08 |
```

## parseDateTimeBestEffort

の日付と時刻を変換します。 文字列への表現 **DateTime** データ型。

関数は解析します **ISO 8601**, **RFC1123-5.2.14RFC-822日付と時刻の仕様**、ClickHouseのいくつかの他の日付と時刻の形式。

## 構文

```
parseDateTimeBestEffort(time_string [, time_zone]);
```

### パラメータ

- `time_string` — String containing a date and time to convert. 文字列.
- `time_zone` — Time zone. The function parses `time_string` タイムゾーンによると。 文字列.

### サポートされている非標準形式

- 9を含む文字列。.10桁 unixタイムスタン.
- 日付と時刻コンポーネントを持つ文字列: YYYYMMDDhhmmss, DD/MM/YYYY hh:mm:ss, DD-MM-YY hh:mm, YYYY-MM-DD hh:mm:ss など。
- 日付を持つ文字列ですが、時刻コンポーネントはありません: YYYY, YYYYMM, YYYY\*MM, DD/MM/YYYY, DD-MM-YY 等。
- 日付と時刻の文字列: DD, DD hh, DD hh:mm. この場合 YYYY-MM と置き換えられる。 2000-01.
- 日付と時刻とタイムゾーンのオフセット情報を含む文字列: YYYY-MM-DD hh:mm:ss ±h:mm など。 例えば, 2020-12-12 17:36:00 -5:00.

区切り文字を持つすべての形式について、関数は月名をフルネームまたは月名の最初の三文字で表したものを探します。 例: 24/DEC/18, 24-Dec-18, 01-September-2018.

### 戻り値

- `time_string` に変換されます。 `DateTime` データ型。

### 例

クエリ:

```
SELECT parseDateTimeBestEffort('12/12/2020 12:12:57')
AS parseDateTimeBestEffort;
```

結果:

```
parseDateTimeBestEffort→
2020-12-12 12:12:57 |
```

クエリ:

```
SELECT parseDateTimeBestEffort('Sat, 18 Aug 2018 07:22:16 GMT', 'Europe/Moscow')
AS parseDateTimeBestEffort
```

結果:

```
parseDateTimeBestEffort→
2018-08-18 10:22:16 |
```

クエリ:

```
SELECT parseDateTimeBestEffort('1284101485')
AS parseDateTimeBestEffort
```

結果:

```
parseDateTimeBestEffort→
2015-07-07 12:04:41 |
```

クエリ:

```
SELECT parseDateTimeBestEffort('2018-12-12 10:12:12')
AS parseDateTimeBestEffort
```

結果:

```
parseDateTimeBestEffort→
2018-12-12 10:12:12 |
```

クエリ:

```
SELECT parseDateTimeBestEffort('10 20:19')
```

結果:

```
parseDateTimeBestEffort('10 20:19')→
2000-01-10 20:19:00 |
```

も参照。

- [ISO8601による発表 @xkcd] (<https://xkcd.com/1179/>)
- [RFC 1123](#)
- [東立（とうだて](#)
- [toDateTime](#)

## parseDateTimeBestEffortOrNull

同じように `parseDateTimeBestEffort` ただし、処理できない日付形式が検出されると `null` が返されます。

## parseDateTimeBestEffortOrZero

同じように `parseDateTimeBestEffort` ただし、処理できない日付形式が検出されたときにゼロの日付またはゼロの日付時刻を返します。

# 日付と時刻を操作するための関数

タイムゾーンのサポート

タイムゾーンに論理的に使用される日付と時刻を操作するためのすべての関数は、オプションのタイムゾーン引数を受け入れることができます。例：アジア/エカテリンブルク。この場合、ローカル(既定)のタイムゾーンではなく、指定されたタイムゾーンを使用します。

```
SELECT
    toDateTime('2016-06-15 23:00:00') AS time,
    toDate(time) AS date_local,
    toDate(time, 'Asia/Yekaterinburg') AS date_yekat,
    toString(time, 'US/Samoa') AS time_samoa
```

time	date_local	date_yekat	time_samoa
2016-06-15 23:00:00	2016-06-15	2016-06-16	2016-06-15 09:00:00

UTCと時間数が異なるタイムゾーンのみがサポートされます。

## ト タイムゾーン

時刻または日付と時刻を指定したタイムゾーンに変換します。

## toYear

日付または日付と時刻を、年番号(AD)を含むUInt16番号に変換します。

## ト クアーター

Dateまたはdate with timeを、四半期番号を含むUInt8番号に変換します。

## ト モンス

Dateまたはdate with timeを、月番号(1-12)を含むUInt8番号に変換します。

## 今日の年

日付または日付と時刻を、年の日の番号を含むUInt16番号(1-366)に変換します。

## 今日の月

日付または日付と時刻を、月の日の番号を含むUInt8番号(1-31)に変換します。

## 今日の週

Dateまたはdate with timeを、曜日の番号を含むUInt8番号(月曜日は1、日曜日は7)に変換します。

## toHour

時刻を持つ日付を、24時間(0-23)の時間数を含むUInt8数値に変換します。

This function assumes that if clocks are moved ahead, it is by one hour and occurs at 2 a.m., and if clocks are moved back, it is by one hour and occurs at 3 a.m. (which is not always true - even in Moscow the clocks were twice changed at a different time).

## ト ミヌテ

時刻を持つ日付を、時間の分の数を含むUInt8番号(0-59)に変換します。

## toSecond

日付と時刻を、分(0-59)の秒数を含むUInt8数値に変換します。  
うるう秒は考慮されません。

## toUnixTimestamp

DateTime引数の場合: 値を内部の数値表現(Unixタイムスタンプ)に変換します。  
文字列引数の場合: タイムゾーンに従って文字列からdatetimeを解析し（オプションの第二引数、サーバーのタイムゾーンがデフォルトで使用されます）、対応するunixタ  
Date引数の場合: 動作は指定されていません。

### 構文

```
toUnixTimestamp(datetime)
toUnixTimestamp(str, [timezone])
```

### 戻り値

- Unixタイムスタンプを返す。

タイプ: UInt32.

### 例

クエリ:

```
SELECT toUnixTimestamp('2017-11-05 08:07:47', 'Asia/Tokyo') AS unix_timestamp
```

結果:

```
unix_timestamp
1509836867 |
```

## トスタート

日付または日付を年の最初の日に切り捨てます。  
日付を返します。

## トスタートフィソイヤー

日付または日付をiso年の最初の日に切り捨てます。  
日付を返します。

## toStartOfQuarter

日付または日付を四半期の最初の日に切り捨てます。  
四半期の最初の日は、1月、1月、1月、または1月のいずれかです。  
日付を返します。

## トスタートモンス

日付または日付を月の最初の日に切り捨てます。  
日付を返します。

### 注意

不正な日付を解析する動作は、実装固有です。ClickHouseはゼロの日付を返すか、例外をスローするか  
“natural” オーバーフロー

## トモンディ

日付または日付を時刻とともに最も近い月曜日に切り捨てます。  
日付を返します。

## トスタートフィーク(`t[, モード]`)

日付または時刻の日付を、最も近い日曜日または月曜日にモードで切り捨てます。  
日付を返します。

Mode引数は、`toWeek()`のmode引数とまったく同じように動作します。 単一引数構文では、モード値0が使用されます。

## トスタートフディ

日付を時刻とともに日の始まりまで切り捨てます。

## トスタートフル

日付と時刻を時間の開始まで切り捨てます。

## トスタートフミニュート

日付と時刻を分の開始まで切り捨てます。

## トスタートオフィブミニュート

日付と時刻を切り捨てます。

## toStartOfTenMinutes

日付と時刻を切り捨てて、十分間隔の開始まで切り捨てます。

## トスタートフィフテンミニュート

日付を時刻とともに切り捨てて、十分間隔の開始まで切り捨てます。

## トスタートオフィンターバル(`time_or_data, 区間x単位 [,time_zone]`)

これは、名前付きの他の関数の一般化です `toStartOf*`。 例えば,  
`toStartOfInterval(t, INTERVAL 1 year)` と同じを返します `toStartOfYear(t)`,  
`toStartOfInterval(t, INTERVAL 1 month)` と同じを返します `toStartOfMonth(t)`,  
`toStartOfInterval(t, INTERVAL 1 day)` と同じを返します `toStartOfDay(t)`,  
`toStartOfInterval(t, INTERVAL 15 minute)` と同じを返します `toStartOfFifteenMinutes(t)` 等。

## トータイム

時刻を保持しながら、時刻を持つ日付を特定の固定日付に変換します。

## toRelativeYearNum

時刻または日付を持つ日付を、過去の特定の固定点から始まる年の番号に変換します。

## toRelativeQuarterNum

時刻または日付を持つ日付を、過去の特定の固定点から始まる四半期の数値に変換します。

## トレラティブモンスヌム

時刻または日付を持つ日付を、過去の特定の固定点から始まる月の番号に変換します。

## toRelativeWeekNum

時刻または日付を持つ日付を、過去の特定の固定点から始まる週の数に変換します。

## toRelativeDayNum

時刻または日付を持つ日付を、過去の特定の固定点から始まる日付の数値に変換します。

## toRelativeHourNum

時刻または日付を持つ日付を、過去の特定の固定点から始まる時間の数に変換します。

## トレラティブミノテン

時刻または日付を持つ日付を、過去の特定の固定点から始まる分の数に変換します。

## toRelativeSecondNum

時刻または日付を持つ日付を、過去の特定の固定点から始まる秒数に変換します。

## toISOYear

Dateまたはdate with timeをISO年番号を含むUInt16番号に変換します。

## toISOWeek

日付または日付と時刻をISO週番号を含むUInt8番号に変換します。

## トウイーク(日付[, モード])

この関数は、dateまたはdatetimeの週番号を返します。二引数形式のtoWeek()を使用すると、週が日曜日か月曜日か、戻り値が0から53または1から53の範囲であるかどうかを指定できます。引数modeを省略すると、デフォルトのモードは0になります。

toISOWeek()と等価な互換性関数です `toWeek(date,3)`.

次の表は、mode引数の動作方法を示しています。

モード	週の最初の日	範囲	Week 1 is the first week ...
0	日曜日	0-53	今年の日曜日と
1	月曜日	0-53	今年は4日以上
2	日曜日	1-53	今年の日曜日と
3	月曜日	1-53	今年は4日以上
4	日曜日	0-53	今年は4日以上

モード	週の最初の日	範囲	Week 1 is the first week ...
5	月曜日	0-53	今年の月曜日と
6	日曜日	1-53	今年は4日以上
7	月曜日	1-53	今年の月曜日と
8	日曜日	1-53	1を含む
9	月曜日	1-53	1を含む

の意味を持つモード値の場合 “with 4 or more days this year,” 週はISO8601:1988に従って番号が付けられます:

- 1月を含む週が新年に4日以上ある場合、それは1週である。
- それ以外の場合は、前年の最後の週であり、次の週は1週です。

の意味を持つモード値の場合 “contains January 1”、1月を含む週は1週である。たとえそれが一日だけ含まれていても、その週が含まれている新年の日数は問題ではありません。

`toWeek(date, [, mode][, Timezone])`

#### パラメータ

- `date` – Date or DateTime.
- `mode` – Optional parameter, Range of values is [0,9], default is 0.
- `Timezone` – Optional parameter, it behaves like any other conversion function.

#### 例

```
SELECT toDate('2016-12-27') AS date, toWeek(date) AS week0, toWeek(date,1) AS week1, toWeek(date,9) AS week9;
```

date	week0	week1	week9
2016-12-27	52	52	1

## toYearWeek(日付[, モード])

日付の年と週を返します。結果の年は、年の最初と最後の週のdate引数の年とは異なる場合があります。

Mode引数は、toWeek()のmode引数とまったく同じように動作します。単一引数構文では、モード値0が使用されます。

`toISOYear()`と等価な互換性関数です `intDiv(toYearWeek(date,3),100)`.

#### 例

```
SELECT toDate('2016-12-27') AS date, toYearWeek(date) AS yearWeek0, toYearWeek(date,1) AS yearWeek1, toYearWeek(date,9) AS yearWeek9;
```

date	yearWeek0	yearWeek1	yearWeek9
2016-12-27	201652	201652	201701

## さて

ゼロの引数を受け取り、要求の実行のいずれかの瞬間に現在の時刻を返します。  
この関数は、要求が完了するまでに時間がかかった場合でも、定数を返します。

## 今日

ゼロの引数を受け取り、要求の実行のいずれかの瞬間に現在の日付を返します。  
と同じ ‘toDate(now())’.

## 昨日

ゼロの引数を受け取り、要求の実行のいずれかの瞬間に昨日の日付を返します。  
と同じ ‘today() - 1’.

## タイムスロット

時間を半分の時間に丸めます。

この機能はYandexに固有のものです。Metricaは、トラッキングタグが単一のユーザーの連続したページビューを表示する場合、セッションを二つのセッションに分割するための最小時間であるため、こつまり、タプル(タグID、ユーザーID、およびタイムスロット)を使用して、対応するセッションに含まれるページビューを検索できます。

## トイヤイム

日付または日付と時刻を、年と月の番号(YYYY\*100+MM)を含むUInt32番号に変換します。

## トイヤイム

日付または日付と時刻を、年と月の番号(YYYY\*10000+MM\*100+DD)を含むUInt32番号に変換します。

## トイヤイム

日付または日付と時刻を、年と月の番号

(YYYY\*10000000000+MM\*100000000+DD\*1000000+hh\*10000+mm\*100+ss)を含むUInt64番号に変換します。

## addYears,addMonths,addWeeks,addDays,addHours,addMinute

関数は、日付/日付時刻に日付/日付時刻の間隔を追加し、日付/日付時刻を返します。 例えば:

```
WITH
    toDate('2018-01-01') AS date,
    toDateTime('2018-01-01 00:00:00') AS date_time
SELECT
    addYears(date, 1) AS add_years_with_date,
    addYears(date_time, 1) AS add_years_with_date_time
```

add_years_with_date	add_years_with_date_time
2019-01-01	2019-01-01 00:00:00

# subtruttyears、subtrutmonths、subtrutweeks、subtrutdays、 subtrutthours、subtrutminutes、subtrutseconds、 subtrutquarters

関数Date/DateTime間隔をDate/DateTimeに減算し、Date/DateTimeを返します。 例えば:

```
WITH
    toDate('2019-01-01') AS date,
    toDate('2019-01-01 00:00:00') AS date_time
SELECT
    subtractYears(date, 1) AS subtract_years_with_date,
    subtractYears(date_time, 1) AS subtract_years_with_date_time
```

```
subtract_years_with_date | subtract_years_with_date_time |
2018-01-01 | 2018-01-01 00:00:00 |
```

## dateDiff

Date値またはDateTime値の差を返します。

構文

```
dateDiff('unit', startdate, enddate, [timezone])
```

パラメータ

- `unit` — Time unit, in which the returned value is expressed. 文字列.

Supported values:

unit
----
second
minute
hour
day
week
month
quarter
year

- `startdate` — The first time value to compare. 日付 または DateTime.
- `enddate` — The second time value to compare. 日付 または DateTime.
- `timezone` — Optional parameter. If specified, it is applied to both `startdate` と `enddate`. 指定されていない場合、`startdate` と `enddate` 使用されます。 それらが同じでない場合、結果は未指定です。

戻り値

の違い `startdate` と `enddate` で表される。 `unit`.

タイプ: int.

例

クエリ:

```
SELECT dateDiff('hour', toDateTime('2018-01-01 22:00:00'), toDateTime('2018-01-02 23:00:00'));
```

結果:

```
dateDiff('hour', toDateTime('2018-01-01 22:00:00'), toDateTime('2018-01-02 23:00:00'))—  
25 |
```

## タイムスロット (StartTime,Duration,[,Size])

で始まる時間間隔の場合 ‘StartTime’ そして続けるために ‘Duration’ これは、この区間の点で構成される時間内の瞬間の配列を返します。‘Size’ 秒で。‘Size’ 定数 UInt32で、既定では1800に設定されます。

例えば, timeSlots(toDateTime('2012-01-01 12:20:00'), 600) = [toDateTime('2012-01-01 12:00:00'), toDateTime('2012-01-01 12:30:00')].

これは、対応するセッションでページビューを検索するために必要です。

## formatDateTime(時刻,Format[,タイムゾーン])

Function formats a Time according given Format string. N.B.: Format is a constant expression, e.g. you can not have multiple formats for single result column.

サポートされている形式の修飾子:

(“Example” 列に時間の書式設定の結果が表示されます 2018-01-02 22:33:44)

修飾子	説明	例
%C	年を100で割り、整数に切り捨てます(00-99)	20
%d	月の日、ゼロパッド(01-31)	02
%D	%M/%d/%yに相当する短いMM/DD/YYの日付	01/02/18
%e	月の日、スペース埋め(1-31)	2
%F	%Y-%m-%dに相当します	2018-01-02
%H	24時間形式 (00-23時)	22
%I	12時間形式 (01-12)	10
%j	年の日(001-366)	002
%m	十進数としての月(01-12)	01
%M	分(00-59)	33
%n	改行文字(“)	
%p	AMまたはPMの指定	PM
%R	24時間HH:MM時間、%H:%Mに相当	22:33
%S	第二 (00-59)	44

修飾子	説明	例
%t	横タブ文字('')	
%T	ISO8601時間形式(HH:MM:SS)、%H:%M:%Sに相当します	22:33:44
%u	ISO8601平日as番号、月曜日as1(1-7)	2
%V	ISO8601週番号(01-53)	01
%w	曜日を十進数とし、日曜日を0(0-6)とします	2
%y	年,最後の二桁(00-99)	18
%Y	年	2018
%%	%記号	%

## 文字列を操作するための関数

### 空

空の文字列の場合は1、空でない文字列の場合は0を返します。

結果の型はUInt8です。

文字列が空白またはnullバイトであっても、少なくとも一つのバイトが含まれている場合、文字列は空ではないと見なされます。

この関数は配列に対しても機能します。

### ノーテンプティ

空の文字列の場合は0、空でない文字列の場合は1を返します。

結果の型はUInt8です。

この関数は配列に対しても機能します。

### 長さ

文字列の長さをバイト単位で返します(文字ではなく、コードポイントではありません)。

結果の型はUInt64です。

この関数は配列に対しても機能します。

### lengthUTF8

文字列にUTF-8でエンコードされたテキストを構成するバイトのセットが含まれていると仮定して、Unicodeコードポイント(文字ではない)で文字列の長さを返す。この仮定が満たされない場合、結果が返されます(例外はスローされません)。

結果の型はUInt64です。

### char\_length,CHAR\_LENGTH

文字列にUTF-8でエンコードされたテキストを構成するバイトのセットが含まれていると仮定して、Unicodeコードポイント(文字ではない)で文字列の長さを返す。この仮定が満たされない場合、結果が返されます(例外はスローされません)。

結果の型はUInt64です。

## **character\_length,CHARACTER\_LENGTH**

文字列にUTF-8でエンコードされたテキストを構成するバイトのセットが含まれていると仮定して、Unicodeコードポイント(文字ではない)で文字列の長さを返す。この仮定が満たされない場合、結果が返されます(例外はスローされません)。

結果の型はUInt64です。

## **lower,lcase**

文字列内のASCIIラテン文字記号を小文字に変換します。

## **upper,ucase**

文字列内のASCIIラテン文字記号を大文字に変換します。

## **lowerUTF8**

文字列にUTF-8でエンコードされたテキストを構成するバイトのセットが含まれていると仮定して、文字列を小文字に変換します。

それは言語を検出しません。そのためトルコに結果が正確に正しい。

UTF-8バイト-シーケンスの長さがコード-ポイントの大文字と小文字で異なる場合、このコード-ポイントの結果が正しくない可能性があります。

文字列にUTF-8以外のバイトセットが含まれている場合、動作は未定義です。

## **upperUTF8**

文字列にUTF-8でエンコードされたテキストを構成するバイトのセットが含まれていると仮定して、文字列を大文字に変換します。

それは言語を検出しません。そのためトルコに結果が正確に正しい。

UTF-8バイト-シーケンスの長さがコード-ポイントの大文字と小文字で異なる場合、このコード-ポイントの結果が正しくない可能性があります。

文字列にUTF-8以外のバイトセットが含まれている場合、動作は未定義です。

## **isValidUTF8**

バイトセットが有効なUTF-8エンコードの場合は1を返し、それ以外の場合は0を返します。

## **toValidUTF8**

無効なUTF-8文字を◆(U+FFFFD)文字。行で実行されているすべての無効な文字は、一つの置換文字に折りたたまれます。

```
toValidUTF8( input_string )
```

パラメータ:

- **input\_string** — Any set of bytes represented as the [文字列](#) データ型オブジェクト。

戻り値:有効なUTF-8文字列。

例

```
SELECT toValidUTF8('\x61\xF0\x80\x80\x80b')
```

```
toValidUTF8('a◆◆◆◆b')
```

a◆b

|

## 繰り返し

指定した回数だけ文字列を繰り返し、複製された値を单一の文字列として連結します。

### 構文

```
repeat(s, n)
```

パラメータ

- **s** — The string to repeat. 文字列.
- **n** — The number of times to repeat the string. UInt.

### 戻り値

文字列を含む单一の文字列 **s** 繰り返し **n** タイムズ もし **n** < 1、関数は空の文字列を返します。

タイプ: String.

### 例

クエリ:

```
SELECT repeat('abc', 10)
```

結果:

```
repeat('abc', 10)——————|  
abcabcaabcabcaabcabca|
```

## 逆

文字列を(バイトのシーケンスとして)反転します。

## reverseUTF8

文字列にUTF-8テキストを表すバイトのセットが含まれていると仮定して、Unicodeコードポイントのシーケンスを反転します。それ以外の場合は、何か他のことをします(例外はスローされません)。

## format(pattern, s0, s1, ...)

引数にリストされた文字列で定数パターンを書式設定します。**pattern** 単純化されたPython形式のパターンです。書式指定文字列 “replacement fields” 中括弧で囲む **{}**. 中かっこに含まれていないものはリテラルテキストと見なされ、そのまま出力にコピーされます。リテラルテキストに中かっこ文字を含める必要がある場合は、倍にすることでエスケープできます: **{} と {}**. フィールド名には、数値(ゼロから始まる)または空(結果の数値として扱われます)を指定できます。

```
SELECT format('{1} {0} {1}', 'World', 'Hello')
```

```
format('{1} {0} {1}', 'World', 'Hello')  
Hello World Hello |
```

```
SELECT format('{} {}', 'Hello', 'World')
```

```
format('{} {}', 'Hello', 'World')  
Hello World |
```

## コンカ

引数にリストされている文字列を、区切り記号なしで連結します。

### 構文

```
concat(s1, s2, ...)
```

### パラメータ

String型またはFixedString型の値。

### 戻り値

引数を連結した結果の文字列を返します。

引数値のいずれかが NULL, concat ツヅケ。 NULL.

### 例

クエリ:

```
SELECT concat('Hello, ', 'World!')
```

結果:

```
concat('Hello, ', 'World!')  
Hello, World! |
```

## concatAssumeInjective

同じ コンカ 違いは、次のことを確認する必要があるということです `concat(s1, s2, ...)` → `sn` は injective であり、GROUP BY の最適化に使用されます。

関数の名前は次のとおりです “injective” 引数の異なる値に対して常に異なる結果を返す場合。 言い換えれば異なる引数のない利回り同一の結果です。

### 構文

```
concatAssumeInjective(s1, s2, ...)
```

### パラメータ

String型またはFixedString型の値。

## 戻り値

引数を連結した結果の文字列を返します。

引数値のいずれかが `NULL`, `concatAssumeInjective` ツヅケ。 `NULL`.

### 例

入力テーブル:

```
CREATE TABLE key_val(`key1` String, `key2` String, `value` UInt32) ENGINE = TinyLog;
INSERT INTO key_val VALUES ('Hello', 'World', 1), ('Hello', 'World', 2), ('Hello', 'World!', 3), ('Hello', 'World!', 2);
SELECT * from key_val;
```

key1	key2	value
Hello,	World	1
Hello,	World	2
Hello,	World!	3
Hello	, World!	2

クエリ:

```
SELECT concat(key1, key2), sum(value) FROM key_val GROUP BY concatAssumeInjective(key1, key2)
```

結果:

concat(key1, key2)	sum(value)
Hello, World!	3
Hello, World!	2
Hello, World	3

## substring(s,offset,length),mid(s,offset,length),substr(s,offset,le

バイトで始まる部分文字列を返します。'offset' であるインデックス 'length' バイト長。 文字割り出し開始から一つとしての標準SQL). その 'offset' と 'length' 引数は定数である必要があります。

## substringUTF8(s,オフセット,長さ)

と同じ 'substring' しかし、Unicodeコードポイントの場合。 作品は、この文字列が含まれるセットを表すバイトのUTF-8で符号化されます。 この仮定が満たされない場合、結果が返されます（例外はスローされません）。

## appendTrailingCharIfAbsent(s,c)

もし 's' 文字列は空ではなく、'c' 最後に文字を追加します。'c' 最後に文字。

## convertCharset(s,from,to)

文字列を返します 's' それはで符号化から変換された 'from' のエンコーディングに 'to'.

## base64Encode(s)

エンコード 's' base64への文字列

## base64Decode(s)

Base64エンコードされた文字列をデコード 's' 元の文字列に。 障害が発生した場合には例外を発生させます。

## tryBase64Decode(s)

Base64Decodeに似ていますが、エラーの場合は空の文字列が返されます。

## endsWith(s,接尾辞)

指定した接尾辞で終わるかどうかを返します。 文字列が指定された接尾辞で終わる場合は1を返し、それ以外の場合は0を返します。

## startsWith(str, プレフィックス)

文字列が指定された接頭辞で始まるかどうかは1を返します。

```
SELECT startsWith('Spider-Man', 'Spi');
```

戻り値

- 文字列が指定された接頭辞で始まる場合は、1。
- 文字列が指定された接頭辞で始まらない場合は0。

例

クエリ:

```
SELECT startsWith('Hello, world!', 'He');
```

結果:

```
startsWith('Hello, world!', 'He')  
1 |
```

トリム

指定したすべての文字を文字列の先頭または末尾から削除します。

デフォルトでは、文字列の両端から共通の空白(ASCII文字32)が連続して出現するすべてを削除します。

構文

```
trim([[LEADING|TRAILING|BOTH] trim_character FROM] input_string)
```

パラメータ

- `trim_character` — specified characters for trim. **文字列**.
- `input_string` — string for trim. **文字列**.

戻り値

先頭および(または)末尾に指定された文字を含まない文字列。

タイプ: **String**.

例

クエリ:

```
SELECT trim(BOTH '()' FROM '(Hello, world! )')
```

結果:

```
trim(BOTH '()' FROM '(Hello, world! )')  
|  
Hello, world!
```

## トリムレフト

文字列の先頭から連続して出現する共通の空白(ASCII文字32)をすべて削除します。他の種類の空白文字(タブ、改行なしスペースなど)は削除されません。).

構文

```
trimLeft(input_string)
```

別名: ltrim(input\_string).

パラメータ

- `input_string` — string to trim. 文字列.

戻り値

共通の空白の先頭を持たない文字列。

タイプ: String.

例

クエリ:

```
SELECT trimLeft('Hello, world! ')
```

結果:

```
trimLeft('Hello, world! ')  
|  
Hello, world!
```

## トリムライト

文字列の末尾から連続して出現するすべての共通空白(ASCII文字32)を削除します。他の種類の空白文字(タブ、改行なしスペースなど)は削除されません。).

構文

```
trimRight(input_string)
```

別名: rtrim(input\_string).

パラメータ

- `input_string` — string to trim. 文字列.

## 戻り値

共通の空白を末尾に付けない文字列。

タイプ: String.

## 例

クエリ:

```
SELECT trimRight('Hello, world!')
```

結果:

```
trimRight('Hello, world!')  
Hello, world!
```

## トリンボス

文字列の両端から共通の空白(ASCII文字32)が連続して出現するすべてを削除します。他の種類の空白文字(タブ、改行なしスペースなど)は削除されません。).

## 構文

```
trimBoth(input_string)
```

別名: trim(input\_string).

パラメータ

- `input_string` — string to trim. 文字列.

## 戻り値

共通の空白文字の先頭と末尾を持たない文字列。

タイプ: String.

## 例

クエリ:

```
SELECT trimBoth('Hello, world!')
```

結果:

```
trimBoth('Hello, world!')  
Hello, world!
```

## CRC32(s)

CRC-32-IEEE802.3多項式と初期値を使用して、文字列のCRC32チェックサムを返します 0xffffffff ( zlibの実装 )。

結果の型は UInt32 です。

## CRC32IEEE(s)

CRC-32-IEEE802.3多項式を使用して、文字列のCRC32チェックサムを返します。

結果の型はUInt32です。

## CRC64(s)

CRC-64-ECMA多項式を使用して、文字列のCRC64チェックサムを返します。

結果の型はUInt64です。

## 文字列を検索する関数

これらのすべての関数では、デフォルトで大文字と小文字が区別されます。あるvariantのための大文字と小文字を区別しません。

### 位置（干し草の山、針）、位置（干し草の山、針）

文字列内で見つかった部分文字列の位置(バイト単位)を1から始めて返します。

作品は、この文字列が含まれるセットを表すバイトの单一のバイトの符号化されます。この仮定が満たされず、文字を单一バイトで表すことができない場合、関数は例外をスローせず、予期しない結果を返します。文字を二つのバイトで表現できる場合は、二つのバイトなどを使用します。

大文字と小文字を区別しない検索では、次の関数を使用します [ポジションカースインセンティブ](#)。

#### 構文

```
position(haystack, needle[, start_pos])
```

別名: `locate(haystack, needle[, start_pos]).`

#### パラメータ

- `haystack` — string, in which substring will be searched. [文字列](#).
- `needle` — substring to be searched. [文字列](#).
- `start_pos` – Optional parameter, position of the first character in the string to start search. [UInt](#)

#### 戻り値

- 部分文字列が見つかった場合は、開始位置(1から数える)をバイト単位で指定します。
- 部分文字列が見つからなかった場合は0。

タイプ: `Integer`.

#### 例

フレーズ “Hello, world!” を含むの設定を表すバイトの单一のバイトの符号化されます。この関数は、期待される結果を返します:

クエリ:

```
SELECT position('Hello, world!', '!')
```

結果:

```
position('Hello, world!', '!')  
13 |
```

ロシア語の同じ句には、单一バイトで表現できない文字が含まれています。この関数は予期しない結果を返します(使用 positionUTF8 マルチバイトエンコードテキストの関数):

クエリ:

```
SELECT position('Привет, мир!', '!')
```

結果:

```
position('Привет, мир!', '!')  
21 |
```

## ポジションカースインセンティブ

と同じ 位置 文字列内で見つかった部分文字列の位置(バイト単位)を1から始めて返します。大文字と小文字を区別しない検索には、この関数を使用します。

作品は、この文字列が含まれるセットを表すバイトの単一のバイトの符号化されます。この仮定が満たされず、文字を单一バイトで表すことができない場合、関数は例外をスローせず、予期しない結果を返します。文字を二つのバイトで表現できる場合は、二つのバイトなどを使用します。

構文

```
positionCaseInsensitive(haystack, needle[, start_pos])
```

パラメータ

- **haystack** — string, in which substring will be searched. [文字列](#).
- **needle** — substring to be searched. [文字列](#).
- **start\_pos** – Optional parameter, position of the first character in the string to start search. [UInt](#)

戻り値

- 部分文字列が見つかった場合は、開始位置(1から数える)をバイト単位で指定します。
- 部分文字列が見つからなかった場合は0。

タイプ: [Integer](#).

例

クエリ:

```
SELECT positionCaseInsensitive('Hello, world!', 'hello')
```

結果:

```
positionCaseInsensitive('Hello, world!', 'hello')  
1 |
```

## positionUTF8

文字列内で見つかった部分文字列の位置(Unicodeポイント)を1から始めて返します。

作品は、この文字列が含まれるセットを表すバイトのUTF-8で符号化されます。この仮定が満たされない場合、関数は例外をスローせず、予期しない結果を返します。文字が二つのUnicodeポイントを使用して表すことができる場合、それはように二つを使用します。

大文字と小文字を区別しない検索では、次の関数を使用します [positionCaseInsensitiveUTF8](#).

### 構文

```
positionUTF8(haystack, needle[, start_pos])
```

### パラメータ

- `haystack` — string, in which substring will be searched. [文字列](#).
- `needle` — substring to be searched. [文字列](#).
- `start_pos` – Optional parameter, position of the first character in the string to start search. [UInt](#)

### 戻り値

- 部分文字列が見つかった場合、Unicodeポイントの開始位置（1から数える）。
- 部分文字列が見つからなかった場合は0。

タイプ: [Integer](#).

### 例

フレーズ “Hello, world!” ロシア語のUnicodeのポイントを表すシングルポイントで符号化されます。この関数は、期待される結果を返します:

クエリ:

```
SELECT positionUTF8('Привет, мир!', '!')
```

結果:

```
positionUTF8('Привет, мир!', '!')  
12 |
```

フレーズ “Salut, étudiante!”, ここで文字 é 一点を使って表現することができます (U+00E9) または二点 (U+0065U+0301) 関数は予期しない結果を返すことができます:

手紙のクエリ é これは一つのUnicodeポイントで表されます U+00E9:

```
SELECT positionUTF8('Salut, étudiante!', '!')
```

結果:

```
positionUTF8('Salut, étudiante!', '!')
```

17 |

手紙のクエリ é これは二つのUnicodeポイントで表されます U+0065U+0301:

```
SELECT positionUTF8('Salut, étudiante!', '!')
```

結果:

```
positionUTF8('Salut, étudiante!', '!')
```

18 |

## positionCaseInsensitiveUTF8

と同じ [positionUTF8](#) しかし、大文字と小文字は区別されません。文字列内で見つかった部分文字列の位置(Unicodeポイント)を1から始めて返します。

作品は、この文字列が含まれるセットを表すバイトのUTF-8で符号化されます。この仮定が満たされない場合、関数は例外をスローせず、予期しない結果を返します。文字が二つのUnicodeポイントを使用して表すことができる場合、それはように二つを使用します。

### 構文

```
positionCaseInsensitiveUTF8(haystack, needle[, start_pos])
```

### パラメータ

- `haystack` — string, in which substring will be searched. [文字列](#).
- `needle` — substring to be searched. [文字列](#).
- `start_pos` – Optional parameter, position of the first character in the string to start search. [UInt](#)

### 戻り値

- 部分文字列が見つかった場合、Unicodeポイントの開始位置（1から数える）。
- 部分文字列が見つからなかった場合は0。

タイプ: [Integer](#).

### 例

クエリ:

```
SELECT positionCaseInsensitiveUTF8('Привет, мир!', 'Мир')
```

結果:

```
positionCaseInsensitiveUTF8('Привет, мир!', 'Мир')
```

9 |

## マルチサーチアルポジション

と同じ **位置** しかし、戻り **Array** 文字列内で見つかった対応する部分文字列の位置(バイト単位)。位置は1から始まる索引付けです。

の検索を行い、配列のバイトを尊重することなく文字列エンコーディングと照合。

- 大文字と小文字を区別しないASCII検索では、次の関数を使用します **multiSearchAllPositionsCaseInsensitive**.
- UTF-8で検索するには、次の関数を使用します **multiSearchAllPositionsUTF8**.
- 大文字と小文字を区別しないUTF-8検索では、関数**multiSearchAllPositionsCaseInsensitiveutf8**を使用します。

## 構文

```
multiSearchAllPositions(haystack, [needle1, needle2, ..., needlen])
```

### パラメータ

- **haystack** — string, in which substring will to be searched. **文字列**.
- **needle** — substring to be searched. **文字列**.

### 戻り値

- 対応する部分文字列が見つかった場合は1から数えます。

### 例

クエリ:

```
SELECT multiSearchAllPositions('Hello, World!', ['hello', '!', 'world'])
```

結果:

```
multiSearchAllPositions('Hello, World!', ['hello', '!', 'world']) →  
[0,13,0]
```

## multiSearchAllPositionsUTF8

見る **multiSearchAllPositions**.

マルチサーチファーストポジション(ヘイスタック、[針<sub>1</sub>、針<sub>2</sub>, ..., needle<sub>n</sub>])

と同じ **position** しかし、文字列の左端のオフセットを返します **haystack** それは針のいくつかに一致します。

大文字と小文字を区別しない検索または/およびUTF-8形式の場合は、関数を使用します  
**multiSearchFirstPositionCaseInsensitive**, **multiSearchFirstPositionUTF8**, **multiSearchFirstPositionCaseInsensitiveUTF8**.

マルチサーチファーストイソデックス(**haystack**,[needle<sub>1</sub>、針<sub>2</sub>, ..., needle<sub>n</sub>])

インデックスを返す i一番左にある針の1から始まります私は 文字列の中で **haystack** それ以外の場合は0です。

大文字と小文字を区別しない検索または/およびUTF-8形式の場合は、関数を使用します  
**multiSearchFirstIndexCaseInsensitive**, **multiSearchFirstIndexUTF8**, **multiSearchFirstIndexCaseInsensitiveUTF8**.

マルチサーチチャニー (干し草、[針<sub>1</sub>、針<sub>2</sub>, ..., needle<sub>n</sub>])

1を返します。私は 文字列と一致します haystack それ以外の場合は0です。

大文字と小文字を区別しない検索または/およびUTF-8形式の場合は、関数を使用します multiSearchAnyCaseInsensitive, multiSearchAnyUTF8, multiSearchAnyCaseInsensitiveUTF8.

## 注

すべて multiSearch\* 機能針の数は2よりより少しひべきです<sup>8</sup> 実装仕様のため。

## マッチ(曖昧さ回避)

文字列が pattern 正規表現。 A re2 正規表現。 その 構文 の re2 正規表現は、Perl正規表現の構文よりも制限されています。

一致しない場合は0、一致する場合は1を返します。

パックスラッシュ記号は (\)は正規表現でのエスケープに使用されます。 文字列リテラルのエスケープには同じ記号が使用されます。 したがって、正規表現のシンボルをエスケープするには、文字列リテラルに二つの円記号(\)を記述する必要があります。

正規表現は、文字列がバイトのセットであるかのように動作します。 正規表現にはnullバイトを含めることはできません。

文字列内の部分文字列を検索するパターンの場合は、LIKEまたはを使用する方が良いです ‘position’ 彼らははるかに高速に動作するので。

## マルチマッチャニー(曖昧さ回避<sub>1</sub>, パターン<sub>2</sub>, ..., pattern<sub>n</sub>])

と同じ match しかし、正規表現が一致しない場合は0を返し、パターンが一致する場合は1を返します。 それは使用します hyperscan 図書館 パターンが文字列内の部分文字列を検索するには、以下を使用する方が良いです multiSearchAny それははるかに高速に動作するので。

## 注

のいずれかの長さ haystack 文字列は2未満でなければなりません<sup>32</sup> それ以外の場合は例外がスローされます。 この制限はhyperscan APIのために行われます。

## multiMatchAnyIndex(ヘイスタック,[パターン<sub>1</sub>, パターン<sub>2</sub>, ..., pattern<sub>n</sub>])

と同じ multiMatchAny しかし、干し草の山に一致する任意のインデックスを返します。

## multiMatchAllIndices(干し草の山、[パターン<sub>1</sub>, パターン<sub>2</sub>, ..., pattern<sub>n</sub>])

と同じ multiMatchAny しかし、干し草の山に一致するすべての指標の配列を任意の順序で返します。

## マルチフジマッチャニー(干し草、距離、[パターン<sub>1</sub>, パターン<sub>2</sub>, ..., pattern<sub>n</sub>])

と同じ multiMatchAny しかし、定数内の干し草に一致するパターンがあれば1を返します 距離を編集. この機能は実験モードでもあり、非常に遅くなる可能性があります。 詳細については、 hyperscanドキュメント.

## multiFuzzyMatchAnyIndex(干し草の山, 距離,[パターン<sub>1</sub>, パターン<sub>2</sub>, ..., pattern<sub>n</sub>])

と同じ `multiFuzzyMatchAny` しかし、一定の編集距離内の干し草の山に一致するインデックスを返します。

## multiFuzzyMatchAllIndices(干し草の山, 距離,[パターン<sub>1</sub>, パターン<sub>2</sub>, ..., pattern<sub>n</sub>])

と同じ `multiFuzzyMatchAny`, but は、一定の編集距離内の干し草の山に一致する任意の順序ですべてのインデックスの配列を返します。

### 注

`multiFuzzyMatch*` 関数はUTF-8正規表現をサポートしておらず、ハイパースキャンの制限によりこのような式はバイトとして扱われます。

### 注

Hyperscanを使用するすべての機能をオフにするには、設定を使用します `SET allow_hyperscan = 0;`

## 抽出(干し草、パターン)

正規表現を使用して文字列の断片を抽出します。もし ‘haystack’ 一致しない ‘pattern’ 正規表現では、空の文字列が返されます。正規表現にサブパターンが含まれていない場合は、正規表現全体に一致するフラグメントを取ります。それ以外の場合は、最初のサブパターンに一致するフラグメントを取ります。

## extractAll(干し草の山、パターン)

正規表現を使用して文字列のすべてのフラグメントを抽出します。もし ‘haystack’ 一致しない ‘pattern’ 正規表現では、空の文字列が返されます。正規表現に一致するすべての文字列からなる配列を返します。一般に、動作は ‘extract’ 関数（最初のサブパターンを取り、サブパターンがない場合は式全体を取ります）。

## like(干し草の山, パターン), 干し草の山のようなパターン演算子

文字列が単純な正規表現に一致するかどうかを確認します。

正規表現には、メタシンボルを含めることができます % と \_.

% 任意のバイト数(ゼロ文字を含む)を示します。

\_ 任意のバイトを示します。

バックスラッシュを使用する (\) メタシンボルを脱出するため。の説明のエスケープに関する注意を参照してください。‘match’ 機能。

次のような正規表現の場合 %needle% コードは、より最適であり、同じくらい速く動作します `position` 機能。  
他の正規表現の場合、コードは ‘match’ 機能。

## notLike(haystack,pattern), haystack NOT LIKE pattern 演算子

同じことと ‘like’ しかし、否定的。

## グラムディスタンス(曖昧さ回避)

間の4グラムの距離を計算します `haystack` と `needle`: counts the symmetric difference between two multisets of 4-grams and normalizes it by the sum of their cardinalities. Returns float number from 0 to 1 – the closer to zero, the more strings are similar to each other. If the constant `needle` または `haystack` 32Kbを超える場合は、例外をスローします。 非定数のいくつかの場合 `haystack` または `needle` 文字列は32Kbを超え、距離は常に一つです。

大文字と小文字を区別しない検索や、UTF-8形式の場合は関数を使用します `ngramDistanceCaseInsensitive`, `ngramDistanceUTF8`, `ngramDistanceCaseInsensitiveUTF8`.

## グラムサーチ(曖昧さ回避)

同じ `ngramDistance` しかし、の間の非対称差を計算します `needle` と `haystack` – the number of n-grams from `needle` minus the common number of n-grams normalized by the number of `needle` nグラム 近いほど、可能性が高くなります `needle` にある `haystack`. あいまい文字列検索に役立ちます。

大文字と小文字を区別しない検索や、UTF-8形式の場合は関数を使用します `ngramSearchCaseInsensitive`, `ngramSearchUTF8`, `ngramSearchCaseInsensitiveUTF8`.

### 注

For UTF-8 case we use 3-gram distance. All these are not perfectly fair n-gram distances. We use 2-byte hashes to hash n-grams and then calculate the (non-)symmetric difference between these hash tables – collisions may occur. With UTF-8 case-insensitive format we do not use fair `tolower` function – we zero the 5-th bit (starting from zero) of each codepoint byte and first bit of zeroth byte if bytes more than one – this works for Latin and mostly for all Cyrillic letters.

## 文字列の検索と置換のための関数

### replaceOne(干し草、パターン、置換)

それが存在する場合は、最初の出現を置き換えます。‘pattern’ サブストリング ‘haystack’ と ‘replacement’ 部分文字列。

以降, ‘pattern’ と ‘replacement’ 定数である必要があります。

### replaceAll(干し草, パターン, 置換), replace(干し草, パターン, 置換)

すべての出現を置き換えます。‘pattern’ サブストリング ‘haystack’ と ‘replacement’ 部分文字列。

### replaceRegexpOne(干し草、パターン、置換)

を使用して交換 ‘pattern’ 正規表現。 Re2正規表現。

存在する場合は、最初の出現のみを置き換えます。

パターンは次のように指定できます ‘replacement’. このパター \0-\9.

置換 \0 正規表現全体を含みます。置換 \1-\9 サブパターンに対応するnumbers.To 使用する \ テンプレート\.

また、文字列リテラルには余分なエスケープが必要です。

例1 日付をアメリカ形式に変換する:

```
SELECT DISTINCT
  EventDate,
  replaceRegexpOne(toString(EventDate), '(\d{4})-(\d{2})-(\d{2})', '$1/$2/$1') AS res
FROM test.hits
LIMIT 7
FORMAT TabSeparated
```

2014-03-17	03/17/2014
2014-03-18	03/18/2014
2014-03-19	03/19/2014
2014-03-20	03/20/2014
2014-03-21	03/21/2014
2014-03-22	03/22/2014
2014-03-23	03/23/2014

例2。文字列を十回コピーする：

```
SELECT replaceRegexpOne('Hello, World!', '*', '\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0') AS res
```

res  
Hello, World!Hello, World!Hello, World!Hello, World!Hello, World!Hello, World!Hello, World!Hello, World!Hello, World!Hello, World!

## replaceRegexpAll(干し草の山, パターン, 置換)

これは同じことを行いますが、すべての出現を置き換えます。例：

```
SELECT replaceRegexpAll('Hello, World!', '.', '\\0\\0') AS res
```

res  
HHeelllloo,, WWoorrlldd!! |

例外として、正規表現が空の部分文字列に対して機能した場合、置換は複数回行われません。

例：

```
SELECT replaceRegexpAll('Hello, World!', '^', 'here: ') AS res
```

res  
here: Hello, World! |

## regexpQuoteMeta(s)

関数は、文字列内のいくつかの定義済み文字の前に円記号を追加します。

定義済み文字: '0', 'V', '|', '(', ')', '^', '\$', '.', '[', ']', '?', '\*', '+', '{', '}', '-'.

この実装はre2::RE2::QuoteMetaとは若干異なります。ゼロバイトは00の代わりに\0としてエスケープされ、必要な文字のみがエスケープされます。

詳細は、リンクを参照してください: [RE2](#)

# Functions for Working with Files

## file

Reads file as a String. The file content is not parsed, so any information is read as one string and placed into the specified column.

## Syntax

```
file(path)
```

## Arguments

- `path` — The relative path to the file from [user\\_files\\_path](#). Path to file support following wildcards: `*`, `?`, `{abc,def}` and `{N..M}` where `N, M` — numbers, `'abc'`, `'def'` — strings.

## Example

Inserting data from files `a.txt` and `b.txt` into a table as strings:

Query:

```
INSERT INTO table SELECT file('a.txt'), file('b.txt');
```

## See Also

- [user\\_files\\_path](#)
- [file](#)

# 条件関数

## もし

条件分岐を制御します。と異なりほとんどのシステムClickHouse常に評価さの両方表現 `then` と `else`.

### 構文

```
SELECT if(cond, then, else)
```

条件が `cond` ゼロ以外の値に評価され、式の結果を返します `then` 式の結果 `else`、存在する場合は、スキップされます。もし `cond` ゼロまたは `NULL` の結果は `then` 式はスキップされ、`else expression`が存在する場合は、`expression`が返されます。

### パラメータ

- `cond` – The condition for evaluation that can be zero or not. The type is `UInt8`, `Nullable(UInt8)` or `NULL`.
- `then` - 条件が満たされた場合に返される式。
- `else` - 条件が満たされていない場合に返される式。

### 戻り値

この関数は実行されます `then` と `else` 式とは、条件かどうかに応じて、その結果を返します `cond` ゼロかどうかになってしましました。

### 例

クエリ:

```
SELECT if(1, plus(2, 2), plus(2, 6))
```

結果:

```
plus(2, 2)
  4 |
```

クエリ:

```
SELECT if(0, plus(2, 2), plus(2, 6))
```

結果:

```
plus(2, 6)
  8 |
```

- `then` と `else` 共通タイプが最も低い必要があります。

例:

これを取る `LEFT_RIGHT` テーブル:

```
SELECT *
FROM LEFT_RIGHT
```

left	right
NULL	4
1	3
2	2
3	1
4	NULL

次のクエリは比較します `left` と `right` 値:

```
SELECT
  left,
  right,
  if(left < right, 'left is smaller than right', 'right is greater or equal than left') AS is_smaller
FROM LEFT_RIGHT
WHERE isNotNull(left) AND isNotNull(right)
```

left	right	is_smaller
1	3	left is smaller than right
2	2	right is greater or equal than left
3	1	right is greater or equal than left

注: `NULL` この例では値は使用されません。 [条件付きのNULL値](#) セクション

## 三項演算子

それは同じように働く `if` 機能。

構文: `cond ? then : else`

ツヅツ。 `then` もし `cond true`(ゼロより大きい)と評価されます。 `else`.

- `cond` の型でなければなりません `UInt8`, and `then` と `else` 共通タイプが最も低い必要があります。
- `then` と `else` ことができます `NULL`

も参照。

- **ifNotFinite**.

## multilf

あなたが書くことができます **CASE** よりコンパクトにクエリ内の演算子。

構文: `multilf(cond_1, then_1, cond_2, then_2, ..., else)`

パラメータ:

- `cond_N` — The condition for the function to return `then_N`.
- `then_N` — The result of the function when executed.
- `else` — The result of the function if none of the conditions is met.

この関数は `2N+1` 変数。

### 戻り値

関数は、次のいずれかの値を返します `then_N` または `else`、条件によって `cond_N`.

例

再び使用する `LEFT_RIGHT` テーブル。

```
SELECT
    left,
    right,
    multilf(left < right, 'left is smaller', left > right, 'left is greater', left = right, 'Both equal', 'Null value') AS result
FROM LEFT_RIGHT
```

left	right	result
NULL	4	Null value
1	3	left is smaller
2	2	Both equal
3	1	left is greater
4	NULL	Null value

## 条件付き結果を直接使用する

条件は常に次のようにになります `0, 1` または `NULL`. できますので使用条件と結果が直接このような:

```
SELECT left < right AS is_small
FROM LEFT_RIGHT
```

is_small
NULL
1
0
0
NULL

## 条件付きのNULL値

とき `NULL` 値は条件に含まれ、結果も次のようにになります `NULL`.

```

SELECT
    NULL < 1,
    2 < NULL,
    NULL < NULL,
    NULL = NULL

    less(NULL, 1)---less(2, NULL)---less(NULL, NULL)---equals(NULL, NULL)---|
    | NULL          | NULL          | NULL          | NULL          |

```

その構築お問合せくの場合はタイプ Nullable.

次の例では、equals条件の追加に失敗してこれを示します multilf.

```

SELECT
    left,
    right,
    multilf(left < right, 'left is smaller', left > right, 'right is smaller', 'Both equal') AS faulty_result
FROM LEFT_RIGHT

    left---right---faulty_result
    | NULL | 4 | Both equal |
    | 1   | 3 | left is smaller |
    | 2   | 2 | Both equal |
    | 3   | 1 | right is smaller |
    | 4   | NULL | Both equal |

```

## 数学関数

すべての関数はFloat64番号を返します。 結果の精度は可能な最大精度に近いですが、結果は対応する実数に最も近い機械表現可能な数と一致しない可能性があります。

### e()

数値eに近いFloat64番号を返します。

### pi()

Returns a Float64 number that is close to the number π.

### exp(x)

数値引数を受け取り、引数の指数に近いFloat64番号を返します。

### ログ(x),ln(x)

数値引数を受け取り、引数の自然対数に近いFloat64数を返します。

### exp2(x)

数値引数を受け取り、Float64をxの累乗に近い2に近い数値を返します。

### log2(x)

数値引数を受け取り、引数のバイナリ対数に近いFloat64数値を返します。

### exp10(x)

数値引数を受け取り、Float64のxの累乗に近い10の数値を返します。

## **log10(x)**

数値引数を受け取り、引数の小数対数に近いFloat64数値を返します。

## **sqrt(x)**

数値引数を受け取り、引数の平方根に近いFloat64番号を返します。

## **cbrt(x)**

数値引数を受け取り、引数の三次根に近いFloat64番号を返します。

## **erf(x)**

もし ‘x’ 負でない場合、`erf(x / σ√2)` 確率変数が標準偏差を持つ正規分布を持つ確率です ‘σ’ より多くの期待値から分離された値を取ります ‘x’.

例 (三シグマ則):

```
SELECT erf(3 / sqrt(2))
```

```
└─erf(divide(3, sqrt(2)))─┐
  0.9973002039367398 |
```

## **erfc(x)**

数値引数を受け取り、Float64の $1 - \text{erf}(x)$ に近い数値を返します。‘x’ 値。

## **lgamma(x)**

ガンマ関数の対数。

## **tgamma(x)**

ガンマ関数

## **sin(x)**

正弦。

## **cos(x)**

コサイン

## **タン(x)**

接線。

## **asin(x)**

アークサイン。

## **acos(x)**

アークコサイン。

## **atan(x)**

円弧正接。

## pow(x,y),power(x,y)

Yの累乗に近いFloat64の数値を返します。

## intExp2

数値引数を受け取り、xの累乗に近いUInt64数値を返します。

## intExp10

数値引数を受け取り、xの累乗に10に近いUInt64の数値を返します。

# 丸め関数

## フロア(x[,N])

以下の最大ラウンド数を返します x. ラウンド数は $1/10^N$ の倍数、または $1/10^N$ が正確でない場合は適切なデータ型の最も近い数値です。

'N' 整数定数、オプションのパラメーターです。これは整数に丸めることを意味します。

'N' 負の場合があります。

例: `floor(123.45, 1) = 123.4, floor(123.45, -1) = 120.`

x 任意の数値型です。結果は同じ型の数です。

整数引数の場合、負の値で丸めるのが理にかなっています N 値(負でない場合 N、関数は何もしません)。

丸めによってオーバーフローが発生した場合(たとえば、`floor(-128,-1)`)、実装固有の結果が返されます。

## ceil(x[,N]),ceiling(x[,N])

以上の最小の丸め数を返します x. 他のすべての方法では、それはと同じです `floor` 関数 (上記参照)。

## trunc(x[,N]),truncate(x[,N])

絶対値が以下の最大絶対値を持つ丸め数を返します x's. In every other way, it is the same as the 'floor' 関数 (上記参照)。

## round(x[,N])

指定した小数点以下の桁数に値を丸めます。

この関数は、指定された順序の最も近い番号を返します。指定された数値が周囲の数値と等しい距離を持つ場合、この関数は浮動小数点数型に対してバンカーの丸めを使用し、他の数値型に対してはゼロから

```
round(expression [, decimal_places])
```

パラメータ:

- expression — A number to be rounded. Can be any 式 数値を返す データ型.
- decimal-places — An integer value.
  - もし `decimal-places > 0` 次に、関数は値を小数点の右側に丸めます。
  - もし `decimal-places < 0` 次に、この関数は値を小数点の左側に丸めます。
  - もし `decimal-places = 0` 次に、この関数は値を整数に丸めます。この場合、引数は省略できます。

## 戻り値:

入力番号と同じタイプの丸められた数値。

## 例

### 使用例

```
SELECT number / 2 AS x, round(x) FROM system.numbers LIMIT 3
```

x	round(divide(number, 2))
0	0
0.5	0
1	1

### 丸めの例

最も近い数値に丸めます。

```
round(3.2, 0) = 3  
round(4.1267, 2) = 4.13  
round(22,-1) = 20  
round(467,-2) = 500  
round(-467,-2) = -500
```

パンカーの丸め。

```
round(3.5) = 4  
round(4.5) = 4  
round(3.55, 1) = 3.6  
round(3.65, 1) = 3.6
```

も参照。

### ■ ラウンドパンカー

## ラウンドパンカー

数値を指定した小数点以下の位置に丸めます。

### ■ 丸め数が二つの数値の中間にある場合、関数は銀行家の丸めを使用します。

Banker's rounding is a method of rounding fractional numbers. When the rounding number is halfway between two numbers, it's rounded to the nearest even digit at the specified decimal position. For example: 3.5 rounds up to 4, 2.5 rounds down to 2.

It's the default rounding method for floating point numbers defined in [IEEE 754] ([https://en.wikipedia.org/wiki/IEEE\\_754#Roundings\\_to\\_nearest](https://en.wikipedia.org/wiki/IEEE_754#Roundings_to_nearest)). The [round](#rounding\_functions-round) function performs the same rounding for floating point numbers. The `roundBankers` function also rounds integers the same way, for example, `roundBankers(45, -1) = 40`.

### ■ それ以外の場合、関数は数値を最も近い整数に丸めます。

銀行家の丸めを使用すると、丸めの数値がこれらの数値の合計または減算の結果に与える影響を減らすことができます。

たとえば、丸めが異なる合計値1.5、2.5、3.5、4.5などです:

### ■ 丸めなし: $1.5 + 2.5 + 3.5 + 4.5 = 12$ .

- パンカーの丸め:  $2 + 2 + 4 + 4 = 12$ .
- 最も近い整数への丸め:  $2 + 3 + 4 + 5 = 14$ .

## 構文

```
roundBankers(expression [, decimal_places])
```

## パラメータ

- **expression** — A number to be rounded. Can be any 式 数値を返す データ型.
- **decimal-places** — Decimal places. An integer number.
  - **decimal-places > 0** — The function rounds the number to the given position right of the decimal point.  
Example: `roundBankers(3.55, 1) = 3.6`.
  - **decimal-places < 0** — The function rounds the number to the given position left of the decimal point.  
Example: `roundBankers(24.55, -1) = 20`.
  - **decimal-places = 0** — The function rounds the number to an integer. In this case the argument can be omitted. Example: `roundBankers(2.5) = 2`.

## 戻り値

パンカーの丸め法によって丸められた値。

## 例

### 使用例

クエリ:

```
SELECT number / 2 AS x, roundBankers(x, 0) AS b fROM system.numbers limit 10
```

結果:

x	b
0	0
0.5	0
1	1
1.5	2
2	2
2.5	2
3	3
3.5	4
4	4
4.5	4

## 銀行家の丸めの例

```
roundBankers(0.4) = 0
roundBankers(-3.5) = -4
roundBankers(4.5) = 4
roundBankers(3.55, 1) = 3.6
roundBankers(3.65, 1) = 3.6
roundBankers(10.35, 1) = 10.4
roundBankers(10.755, 2) = 11.76
```

も参照。

## ■ 九

### roundToExp2(num)

番号を受け入れます。 数値が一つより小さい場合は、0を返します。 それ以外の場合は、数値を最も近い（負でない全体の）次数まで切り捨てます。

### roundDuration(num)

番号を受け入れます。 数値が一つより小さい場合は、0を返します。 それ以外の場合は、数値をセットの数値に切り捨てます: 1, 10, 30, 60, 120, 180, 240, 300, 600, 1200, 1800, 3600, 7200, 18000, 36000. この機能はYandexに固有のものです。Metricaとセッションの長さに関するレポートの実装に使用されます。

### roundAge(num)

番号を受け入れます。 数値が18未満の場合は、0を返します。 それ以外の場合は、数値をセットの数値に切り捨てます: 18, 25, 35, 45, 55. この機能はYandexに固有のものです。ユーザー年齢に関するレポートを実装するために使用されます。

## ラウンドダウン(num,arr)

数値を受け取り、指定された配列内の要素に切り捨てます。 値が最低限界より小さい場合は、最低限界が返されます。

## Functions for maps

### map

Arranges key:value pairs into **Map(key, value)** data type.

#### Syntax

```
map(key1, value1[, key2, value2, ...])
```

#### Arguments

- **key** — The key part of the pair. **String** or **Integer**.
- **value** — The value part of the pair. **String**, **Integer** or **Array**.

#### Returned value

- Data structure as key:value pairs.

Type: **Map(key, value)**.

#### Examples

Query:

```
SELECT map('key1', number, 'key2', number * 2) FROM numbers(3);
```

Result:

```
map('key1', number, 'key2', multiply(number, 2))  
{'key1':0,'key2':0}  
{'key1':1,'key2':2}  
{'key1':2,'key2':4}
```

Query:

```
CREATE TABLE table_map (a Map(String, UInt64)) ENGINE = MergeTree() ORDER BY a;  
INSERT INTO table_map SELECT map('key1', number, 'key2', number * 2) FROM numbers(3);  
SELECT a['key2'] FROM table_map;
```

Result:

```
arrayElement(a, 'key2')  
0  
2  
4
```

## See Also

- [Map\(key, value\)](#) data type

## mapAdd

Collect all the keys and sum corresponding values.

### Syntax

```
mapAdd(arg1, arg2 [, ...])
```

### Arguments

Arguments are [maps](#) or [tuples](#) of two [arrays](#), where items in the first array represent keys, and the second array contains values for each key. All key arrays should have same type, and all value arrays should contain items which are promoted to the one type ([Int64](#), [UInt64](#) or [Float64](#)). The common promoted type is used as a type for the result array.

### Returned value

- Depending on the arguments returns one [map](#) or [tuple](#), where the first array contains the sorted keys and the second array contains values.

### Example

Query with a tuple:

```
SELECT mapAdd(([toUInt8(1), 2], [1, 1]), ([toUInt8(1), 2], [1, 1])) as res, toTypeName(res) as type;
```

Result:

```
res-----type-----  
([1,2],[2,2]) | Tuple(Array(UInt8), Array(UInt64)) |
```

Query with Map type:

```
SELECT mapAdd(map(1,1), map(1,1));
```

Result:

```
mapAdd(map(1, 1), map(1, 1))  
 {1:2} |
```

## mapSubtract

Collect all the keys and subtract corresponding values.

### Syntax

```
mapSubtract(Tuple(Array, Array), Tuple(Array, Array) [, ...])
```

### Arguments

Arguments are **maps** or **tuples** of two **arrays**, where items in the first array represent keys, and the second array contains values for each key. All key arrays should have same type, and all value arrays should contain items which are promote to the one type (**Int64**, **UInt64** or **Float64**). The common promoted type is used as a type for the result array.

### Returned value

- Depending on the arguments returns one **map** or **tuple**, where the first array contains the sorted keys and the second array contains values.

### Example

Query with a tuple map:

```
SELECT mapSubtract(([toUInt8(1), 2], [toInt32(1), 1]), ([toUInt8(1), 2], [toInt32(2), 1])) as res, toTypeName(res) as type;
```

Result:

```
res type  
{[1,2],[-1,0]} | Tuple(Array(UInt8), Array(Int64)) |
```

Query with **Map** type:

```
SELECT mapSubtract(map(1,1), map(1,1));
```

Result:

```
mapSubtract(map(1, 1), map(1, 1))  
 {1:0} |
```

## mapPopulateSeries

Fills missing keys in the maps (key and value array pair), where keys are integers. Also, it supports specifying the max key, which is used to extend the keys array.

## Syntax

```
mapPopulateSeries(keys, values[, max])
mapPopulateSeries(map[, max])
```

Generates a map (a tuple with two arrays or a value of Map type, depending on the arguments), where keys are a series of numbers, from minimum to maximum keys (or max argument if it specified) taken from the map with a step size of one, and corresponding values. If the value is not specified for the key, then it uses the default value in the resulting map. For repeated keys, only the first value (in order of appearing) gets associated with the key.

For array arguments the number of elements in keys and values must be the same for each row.

## Arguments

Arguments are maps or two arrays, where the first array represent keys, and the second array contains values for the each key.

Mapped arrays:

- keys — Array of keys. [Array\(Int\)](#).
- values — Array of values. [Array\(Int\)](#).
- max — Maximum key value. Optional. [Int8](#), [Int16](#), [Int32](#), [Int64](#), [Int128](#), [Int256](#).

or

- map — Map with integer keys. [Map](#).

## Returned value

- Depending on the arguments returns a map or a tuple of two arrays: keys in sorted order, and values the corresponding keys.

## Example

Query with mapped arrays:

```
SELECT mapPopulateSeries([1,2,4], [11,22,44], 5) AS res, toTypeName(res) AS type;
```

Result:

res	type
[1,2,3,4,5],[11,22,0,44,0]	Tuple(Array(UInt8), Array(UInt8))

Query with Map type:

```
SELECT mapPopulateSeries(map(1, 10, 5, 20), 6);
```

Result:

mapPopulateSeries(map(1, 10, 5, 20), 6)	—
{1:10,2:0,3:0,4:0,5:20,6:0}	

# mapContains

Determines whether the `map` contains the `key` parameter.

## Syntax

```
mapContains(map, key)
```

## Parameters

- `map` — Map. [Map](#).
- `key` — Key. Type matches the type of keys of `map` parameter.

## Returned value

- 1 if `map` contains `key`, 0 if not.

Type: [UInt8](#).

## Example

Query:

```
CREATE TABLE test (a Map(String,String)) ENGINE = Memory;
INSERT INTO test VALUES ( {'name':'eleven','age':'11'}), ( {'number':'twelve','position':'6.0'});
SELECT mapContains(a, 'name') FROM test;
```

Result:

```
mapContains(a, 'name')
1
0
```

# mapKeys

Returns all keys from the `map` parameter.

Can be optimized by enabling the [optimize\\_functions\\_to\\_subcolumns](#) setting. With `optimize_functions_to_subcolumns = 1` the function reads only `keys` subcolumn instead of reading and processing the whole column data. The query `SELECT mapKeys(m) FROM table` transforms to `SELECT m.keys FROM table`.

## Syntax

```
mapKeys(map)
```

## Parameters

- `map` — Map. [Map](#).

## Returned value

- Array containing all keys from the `map`.

Type: [Array](#).

## Example

Query:

```
CREATE TABLE test (a Map(String,String)) ENGINE = Memory;  
INSERT INTO test VALUES ( {'name':'eleven','age':'11'}), ( {'number':'twelve','position':'6.0'});  
SELECT mapKeys(a) FROM test;
```

Result:

```
mapKeys(a)  
[ 'name', 'age' ] |  
[ 'number', 'position' ] |
```

## mapValues

Returns all values from the `map` parameter.

Can be optimized by enabling the [optimize\\_functions\\_to\\_subcolumns](#) setting. With `optimize_functions_to_subcolumns = 1` the function reads only `values` subcolumn instead of reading and processing the whole column data. The query `SELECT mapValues(m)` FROM table transforms to `SELECT m.values` FROM table.

## Syntax

```
mapKeys(map)
```

## Parameters

- `map` — Map. [Map](#).

## Returned value

- Array containing all the values from `map`.

Type: [Array](#).

## Example

Query:

```
CREATE TABLE test (a Map(String,String)) ENGINE = Memory;  
INSERT INTO test VALUES ( {'name':'eleven','age':'11'}), ( {'number':'twelve','position':'6.0'});  
SELECT mapValues(a) FROM test;
```

Result:

```
mapValues(a)  
[ 'eleven', '11' ] |  
[ 'twelve', '6.0' ] |
```

## mapContainsKeyLike

## Syntax

```
mapContainsKeyLike(map, pattern)
```

## Parameters

- `map` — Map. [Map](#).
- `pattern` - String pattern to match.

## Returned value

- 1 if `map` contains key like specified pattern, 0 if not.

## Example

Query:

```
CREATE TABLE test (a Map(String,String)) ENGINE = Memory;  
INSERT INTO test VALUES ({{'abc':'abc','def':'def'}}), ({{'hij':'hij','klm':'klm'}});  
SELECT mapContainsKeyLike(a, 'a%') FROM test;
```

Result:

```
mapContainsKeyLike(a, 'a%')  
1  
0
```

## mapExtractKeyLike

## Syntax

```
mapExtractKeyLike(map, pattern)
```

## Parameters

- `map` — Map. [Map](#).
- `pattern` - String pattern to match.

## Returned value

- A map contained elements the key of which matchs the specified pattern. If there are no elements matched the pattern, it will return an empty map.

## Example

Query:

```
CREATE TABLE test (a Map(String,String)) ENGINE = Memory;  
INSERT INTO test VALUES ({{'abc':'abc','def':'def'}}), ({{'hij':'hij','klm':'klm'}});  
SELECT mapExtractKeyLike(a, 'a%') FROM test;
```

Result:

```
mapExtractKeyLike(a, 'a%')  
{'abc':'abc'}  
{}  
]
```

## 配列を操作するための関数

### 空

空の配列の場合は1、空でない配列の場合は0を返します。

結果の型はUInt8です。

この関数は文字列に対しても機能します。

### ノーテンプティ

空の配列の場合は0、空でない配列の場合は1を返します。

結果の型はUInt8です。

この関数は文字列に対しても機能します。

### 長さ

配列内の項目の数を返します。

結果の型はUInt64です。

この関数は文字列に対しても機能します。

**emptyArrayUInt8,emptyArrayUInt16,emptyArrayUInt32,empty, emptyArrayInt8,emptyArrayInt16,emptyArrayInt32,emptyArrayFloat32,emptyArrayFloat64  
emptyArrayDate,emptyArrayDateTime  
emptyArrayString**

ゼロの引数を受け取り、適切な型の空の配列を返します。

### エンプティアライツングル

空の配列を受け取り、既定値と等しい一要素配列を返します。

## 範囲(終了)、範囲(開始、終了[、ステップ])

開始から終了-1までの数値の配列をステップごとに返します。

引数が `start` 既定値は0です。

引数が `step` 既定値は1です。

それはpythonicのように動作します `range`. しかし、違いは、すべての引数の型は UInt 数字だ

場合によっては、合計長が100,000,000を超える要素の配列がデータブロックに作成されると、例外がスローされます。

## array(x1, ...), operator [x1, ...]

関数の引数から配列を作成します。

引数は定数であり、最小の共通型を持つ型を持つ必要があります。それ以外の場合は、作成する配列の型が明確ではありません。つまり、この関数を使用して空の配列を作成することはできません（それを行うには、「emptyArray\*」以上の機能）。

を返す「Array(T) "result"」と入力します。「T」渡された引数のうち最小の共通型です。

## arrayConcat

引数として渡された配列を結合します。

```
arrayConcat(arrays)
```

パラメータ

- arrays – Arbitrary number of arguments of 配列 タイプ。

例

```
SELECT arrayConcat([1, 2], [3, 4], [5, 6]) AS res
```

```
res  
[1,2,3,4,5,6] |
```

## arrayElement(arr,n), 演算子arr[n]

インデックスを持つ要素を取得します n 配列から arr. n 任意の整数型である必要があります。

配列のインデックスは、配列から始まります。

負の索引がサポートされます。この場合、末尾から番号付きの対応する要素を選択する。例えば, arr[-1] 配列内の最後の項目です。

インデックスが配列の境界の外にある場合、デフォルト値（数値の場合は0、文字列の場合は空の文字列など）を返します。非定数配列と定数インデックス0の場合を除きます（この場合はエラーが発生します Array indices are 1-based).

## has(arr, elem)

チェックかどうか 'arr' 配列は 'elem' 要素。

要素が配列内にない場合は0を返し、要素が配列内にない場合は1を返します。

NULL 値として処理されます。

```
SELECT has([1, 2, NULL], NULL)
```

```
has([1, 2, NULL], NULL)  
1 |
```

## hasAll

ある配列が別の配列のサブセットかどうかを確認します。

```
hasAll(set, subset)
```

パラメータ

- **set** – Array of any type with a set of elements.
- **subset** – Array of any type with elements that should be tested to be a subset of **set**.

戻り値

- **1**,if **set** すべての要素が含まれています **subset**.
- **0** そうでなければ

独特の特性

- 空の配列は、任意の配列のサブセットです。
- **Null** 値として処理されます。
- 両方の配列の値の順序は重要ではありません。

例

`SELECT hasAll([], [])` 1を返します。

`SELECT hasAll([1, Null], [Null])` 1を返します。

`SELECT hasAll([1.0, 2, 3, 4], [1, 3])` 1を返します。

`SELECT hasAll(['a', 'b'], ['a'])` 1を返します。

`SELECT hasAll([1], ['a'])` 0を返します。

`SELECT hasAll([[1, 2], [3, 4]], [[1, 2], [3, 5]])` 0を返します。

ハサニ一

るかどうかを判二つの配列が互いの交差点にある。

`hasAny(array1, array2)`

パラメータ

- **array1** – Array of any type with a set of elements.
- **array2** – Array of any type with a set of elements.

戻り値

- **1**,if **array1** と **array2** 少なくとも一つの同様の要素があります。
- **0** そうでなければ

独特の特性

- **Null** 値として処理されます。
- 両方の配列の値の順序は重要ではありません。

例

`SELECT hasAny([1], [])` ツヅケツ。 0.

`SELECT hasAny([Null], [Null, 1])` ツヅケツ。 1.

`SELECT hasAny([-128, 1., 512], [1])` ツヅケツ。 1.

```
SELECT hasAny([[1, 2], [3, 4]], ['a', 'c']) ツヅケツ。 0.
```

```
SELECT hasAll([[1, 2], [3, 4]], [[1, 2], [1, 2]]) ツヅケツ。 1.
```

## インデックス(arr,x)

最初のインデックスを返します ‘x’ 配列内にある場合は要素（1から始まる） 、そうでない場合は0。

例:

```
SELECT indexOf([1, 3, NULL, NULL], NULL)
```

```
indexOf([1, 3, NULL, NULL], NULL)─  
    3 |
```

に設定された要素 **NULL** 通常の値として処理されます。

## カウントイカル(arr,x)

Xと等しい配列内の要素の数を返します。arrayCount(elem->elem=x,arr)と同等です。

**NULL** 要素は個別の値として処理されます。

例:

```
SELECT countEqual([1, 2, NULL, NULL], NULL)
```

```
countEqual([1, 2, NULL, NULL], NULL)─  
    2 |
```

## アレイン(arr)

Returns the array [1, 2, 3, ..., length (arr) ]

この関数は通常、配列結合で使用されます。 配列結合を適用した後、配列ごとに一度だけ何かを数えることができます。 例:

```
SELECT  
  count() AS Reaches,  
  countIf(num = 1) AS Hits  
FROM test.hits  
ARRAY JOIN  
  GoalsReached,  
  arrayEnumerate(GoalsReached) AS num  
WHERE CounterID = 160656  
LIMIT 10
```

```
Reaches ── Hits  
95606 | 31406 |
```

この例では、Reachはコンバージョン数(配列結合を適用した後に受信した文字列)、Hitsはページビュー数(配列結合の前の文字列)です。 この特定のケースでは、同じ結果を得ることができます:

```

SELECT
    sum(length(GoalsReached)) AS Reaches,
    count() AS Hits
FROM test.hits
WHERE (CounterID = 160656) AND notEmpty(GoalsReached)

```

Reaches	Hits
95606	31406

この関数は、高次関数でも使用できます。たとえば、条件に一致する要素の配列インデックスを取得するために使用できます。

## arrayEnumerateUniq(arr, ...)

ソース配列と同じサイズの配列を返します。

たとえば、`arrayEnumerateUniq([10, 20, 10, 30]) = [1, 1, 2, 1]`.

この関数は、配列結合と配列要素の集計を使用する場合に便利です。

例:

```

SELECT
    Goals.ID AS GoalID,
    sum(Sign) AS Reaches,
    sumIf(Sign, num = 1) AS Visits
FROM test.visits
ARRAY JOIN
    Goals,
    arrayEnumerateUniq(Goals.ID) AS num
WHERE CounterID = 160656
GROUP BY GoalID
ORDER BY Reaches DESC
LIMIT 10

```

GoalID	Reaches	Visits
53225	3214	1097
2825062	3188	1097
56600	2803	488
1989037	2401	365
2830064	2396	910
1113562	2372	373
3270895	2262	812
1084657	2262	345
56599	2260	799
3271094	2256	812

この例では、各ゴールIDには、コンバージョン数(ゴールの入れ子になったデータ構造の各要素は、到達したゴールであり、これをコンバージョンと呼びます)とセッション数を`sum(Sign)`として数えました。しかし、この特定のケースでは、行にネストされたGoals構造体が乗算されているため、この後に各セッションを一度カウントするために、`arrayEnumerateUniq(Goals.ID)` 機能。

`ArrayEnumerateUniq`関数は、引数と同じサイズの複数の配列を取ることができます。この場合、すべての配列の同じ位置にある要素のタプルに対して一意性が考慮されます。

```

SELECT arrayEnumerateUniq([1, 1, 1, 2, 2, 2], [1, 1, 2, 1, 1, 2]) AS res

```

```
res  
[1,2,1,1,2,1] |
```

これは、入れ子になったデータ構造で配列結合を使用し、この構造体内の複数の要素間でさらに集計する場合に必要です。

## arrayPopBack

配列から最後の項目を削除します。

```
arrayPopBack(array)
```

パラメータ

- `array` – Array.

例

```
SELECT arrayPopBack([1, 2, 3]) AS res
```

```
res  
[1,2] |
```

## arrayPopFront

配列から最初の項目を削除します。

```
arrayPopFront(array)
```

パラメータ

- `array` – Array.

例

```
SELECT arrayPopFront([1, 2, 3]) AS res
```

```
res  
[2,3] |
```

## arrayPushBack

配列の最後に項目を追加します。

```
arrayPushBack(array, single_value)
```

パラメータ

- `array` – Array.

- `single_value` – A single value. Only numbers can be added to an array with numbers, and only strings can be added to an array of strings. When adding numbers, ClickHouse automatically sets the `single_value` 配列のデータ型の型。ClickHouseのデータの種類については、次を参照してください “データ型”. ことができます `NULL`. この関数は `NULL` 要素を配列に変換し、配列要素の型を次のように変換します `Nullable`.

例

```
SELECT arrayPushBack(['a'], 'b') AS res
```

```
res  
['a','b'] |
```

## アレイ プッシュフロント

配列の先頭に要素を追加します。

```
arrayPushFront(array, single_value)
```

パラメータ

- `array` – Array.
- `single_value` – A single value. Only numbers can be added to an array with numbers, and only strings can be added to an array of strings. When adding numbers, ClickHouse automatically sets the `single_value` 配列のデータ型の型。ClickHouseのデータの種類については、次を参照してください “データ型”. することができます `NULL`. この関数は `NULL` 要素を配列に変換し、配列要素の型を次のように変換します `Nullable`.

例

```
SELECT arrayPushFront(['b'], 'a') AS res
```

```
res  
['a','b'] |
```

## arrayResize

配列の長さを変更します。

```
arrayResize(array, size[, extender])
```

パラメータ:

- `array` — Array.
- `size` — Required length of the array.
  - もし `size` 配列の元のサイズよりも小さい場合、配列は右から切り捨てられます。
- もし `size` 配列の初期サイズよりも大きい場合、配列は次のように右に拡張されます `extender` 配列項目のデータ型の値または既定値。
- `extender` — Value for extending an array. Can be `NULL`.

戻り値:

長さの配列 size.

コールの例

```
SELECT arrayResize([1], 3)
```

```
arrayResize([1], 3)  
[1,0,0] |
```

```
SELECT arrayResize([1], 3, NULL)
```

```
arrayResize([1], 3, NULL)  
[1,NULL,NULL] |
```

## アレイスライス

配列のスライスを返します。

```
arraySlice(array, offset[, length])
```

パラメータ

- **array** – Array of data.
- **offset** – Indent from the edge of the array. A positive value indicates an offset on the left, and a negative value is an indent on the right. Numbering of the array items begins with 1.
- **length** - 必要なスライスの長さ。 負の値を指定すると、関数は開いているスライスを返します [offset, array\_length - length]. 値を省略すると、関数はスライスを返します [offset, the\_end\_of\_array].

例

```
SELECT arraySlice([1, 2, NULL, 4, 5], 2, 3) AS res
```

```
res  
[2,NULL,4] |
```

配列要素に設定 **NULL** 通常の値として処理されます。

## arraySort([func,] arr, ...)

の要素を並べ替えます **arr** 昇順の配列。もし **func** 関数が指定されると、並べ替え順序は **func** 配列の要素に適用される関数。もし **func** 複数の引数を受け入れる。**arraySort** 関数には、引数の配列が複数渡されます。**func** に対応する。詳しい例はの終わりに示されています **arraySort** 説明。

整数値ソートの例:

```
SELECT arraySort([1, 3, 3, 0]);
```

```
arraySort([1, 3, 3, 0])  
[0,1,3,3]
```

文字列値のソートの例:

```
SELECT arraySort(['hello', 'world', '!']);
```

```
arraySort(['hello', 'world', '!'])  
['!', 'hello', 'world']
```

次の並べ替え順序を考えてみましょう `NULL`, `NaN` と `Inf` 値:

```
SELECT arraySort([1, nan, 2, NULL, 3, nan, -4, NULL, inf, -inf]);
```

```
arraySort([1, nan, 2, NULL, 3, nan, -4, NULL, inf, -inf])  
[-inf, -4, 1, 2, 3, inf, nan, nan, NULL, NULL]
```

- `-Inf` 値は配列の最初にあります。
- `NULL` 値は配列の最後にあります。
- `NaN` 値は直前です `NULL`.
- `Inf` 値は直前です `NaN`.

なお `arraySort` は [高次関数](#). `Lambda` 関数を最初の引数として渡すことができます。この場合、並べ替え順序は、配列の要素に適用された `lambda` 関数の結果によって決定されます。

次の例を考えてみましょう:

```
SELECT arraySort((x) -> -x, [1, 2, 3]) as res;
```

```
res  
[3,2,1]
```

For each element of the source array, the lambda function returns the sorting key, that is,  $[1 \rightarrow -1, 2 \rightarrow -2, 3 \rightarrow -3]$ . Since the `arraySort` 関数は昇順にキーをソートし、結果は  $[3, 2, 1]$  です。したがって、 $(x) \rightarrow -x$  ラムダ関数は [降順](#) ソートで。

`Lambda` 関数は複数の引数を受け取ることができます。この場合、`arraySort` 関数 `lambda` 関数の引数が対応する同じ長さのいくつかの配列。結果の配列は最初の入力配列の要素で構成され、次の入力配列の要素は並べ替えキーを指定します。例えば:

```
SELECT arraySort((x, y) -> y, ['hello', 'world'], [2, 1]) as res;
```

```
res  
['world', 'hello']
```

ここでは、二番目の配列([2,1])に渡される要素は、ソース配列の対応する要素の並べ替えキーを定義します (['hello', 'world'])、それは, ['hello' -> 2, 'world' -> 1]. Since the lambda function doesn't use `x`、ソース配列の実際の値は結果の順序に影響しません。だから, 'hello' 結果の第二の要素になります。'world' 最初になります。

その他の例を以下に示す。

```
SELECT arraySort((x, y) -> y, [0, 1, 2], ['c', 'b', 'a']) as res;
```

```
res  
[2,1,0] |
```

```
SELECT arraySort((x, y) -> -y, [0, 1, 2], [1, 2, 3]) as res;
```

```
res  
[2,1,0] |
```

## 注

効率を分類することを改善するため、**シュワルツ変換** が使用される。

## arrayReverseSort([func,] arr, ...)

の要素を並べ替えます `arr` 降順の配列。もし `func` 関数を指定します, `arr` の結果に従ってソートされます。`func` 関数は、配列の要素に適用され、その後、ソートされた配列が逆になります。もし `func` 複数の引数を受け入れる。`arrayReverseSort` 関数には、引数の配列が複数渡されます。`func` に対応する。詳しい例はの終わりに示されています `arrayReverseSort` 説明。

整数値ソートの例:

```
SELECT arrayReverseSort([1, 3, 3, 0]);
```

```
arrayReverseSort([1, 3, 3, 0])—  
[3,3,1,0] |
```

文字列値のソートの例:

```
SELECT arrayReverseSort(['hello', 'world', '!']);
```

```
arrayReverseSort(['hello', 'world', '!'])—  
['world','hello','!'] |
```

次の並べ替え順序を考えてみましょう `NULL`, `NaN` と `Inf` 値:

```
SELECT arrayReverseSort([1, nan, 2, NULL, 3, nan, -4, NULL, inf, -inf]) as res;
```

```
res  
[inf,3,2,1,-4,-inf,NaN,NaN,NULL,NULL] |
```

- Inf 値は配列の最初にあります。
- NULL 値は配列の最後にあります。
- NaN 値は直前です NULL.
- -Inf 値は直前です NaN.

なお、`arrayReverseSort` は **高次関数**。Lambda関数を最初の引数として渡すことができます。以下に例を示す。

```
SELECT arrayReverseSort((x) -> -x, [1, 2, 3]) AS res;
```

```
res  
[1,2,3] |
```

配列は次のように並べ替えられます:

1. 最初に、ソース配列([1,2,3])は、配列の要素に適用されたlambda関数の結果に従ってソートされます。結果は配列[3,2,1]です。
2. 前のステップで取得された配列は、逆になります。したがって、最終的な結果は[1,2,3]です。

Lambda関数は複数の引数を受け取ることができます。この場合、`arrayReverseSort` 関数lambda関数の引数が対応する同じ長さのいくつかの配列。結果の配列は最初の入力配列の要素で構成され、次の入力配列の要素は並べ替えキーを指定します。例えば:

```
SELECT arrayReverseSort((x, y) -> y, ['hello', 'world'], [2, 1]) AS res;
```

```
res  
['hello','world'] |
```

この例では、配列は次のように並べ替えられます:

1. 最初は、ソース配列(['hello', 'world'])は、配列の要素に適用されたlambda関数の結果に従ってソートされます。二番目の配列 ([2,1]) に渡される要素は、ソース配列の対応する要素の並べ替えキーを定義します。結果は配列です ['world', 'hello']。
2. 前のステップでソートされた配列は、逆になります。だから、最終的な結果は ['hello', 'world']。

その他の例を以下に示す。

```
SELECT arrayReverseSort((x, y) -> y, [4, 3, 5], ['a', 'b', 'c']) AS res;
```

```
res  
[5,3,4] |
```

```
SELECT arrayReverseSort((x, y) -> -y, [4, 3, 5], [1, 2, 3]) AS res;
```

```
res
[4,3,5] |
```

## arrayUniq(arr, ...)

一つの引数が渡されると、配列内の異なる要素の数がカウントされます。

複数の引数が渡された場合、複数の配列内の対応する位置にある要素の異なるタプルの数をカウントします。

配列内の一意の項目のリストを取得する場合は、arrayReduceを使用できます('groupUniqArray',arr)。

## アレイジョイン(arr)

特別な機能。セクションを参照 [“ArrayJoin function”](#).

## arrayDifference

隣接する配列要素の差を計算します。最初の要素が0になる配列を返します。  $a[1] - a[0]$ , etc. The type of elements in the resulting array is determined by the type inference rules for subtraction (e.g. UInt8 - UInt8 = Int16).

構文

```
arrayDifference(array)
```

パラメータ

- **array** – 配列.

戻り値

隣接する要素間の差分の配列を返します。

タイプ: **UInt\***, **Int\***, **フロート\***.

例

クエリ:

```
SELECT arrayDifference([1, 2, 3, 4])
```

結果:

```
arrayDifference([1, 2, 3, 4]) |
[0,1,1,1]
```

結果の型**Int64**によるオーバーフローの例:

クエリ:

```
SELECT arrayDifference([0, 1000000000000000000000000])
```

結果:

```
arrayDifference([0, 1000000000000000000000000])─  
[0,-8446744073709551616] | ]
```

## アレイディスト

配列を受け取り、別個の要素のみを含む配列を返します。

構文

```
arrayDistinct(array)
```

パラメータ

- array – 配列。

戻り値

個別の要素を含む配列を返します。

例

クエリ:

```
SELECT arrayDistinct([1, 2, 2, 3, 1])
```

結果:

```
arrayDistinct([1, 2, 2, 3, 1])─  
[1,2,3] | ]
```

## アレイニュメラテンセ(arr)

ソース配列と同じサイズの配列を返します。

例:

```
SELECT arrayEnumerateDense([10, 20, 10, 30])
```

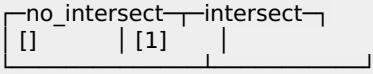
```
arrayEnumerateDense([10, 20, 10, 30])─  
[1,2,1,3] | ]
```

## アレイインターセクト(arr)

複数の配列を取り、すべてのソース配列に存在する要素を含む配列を返します。結果の配列内の要素の順序は、最初の配列と同じです。

例:

```
SELECT  
    arrayIntersect([1, 2], [1, 3], [2, 3]) AS no_intersect,  
    arrayIntersect([1, 2], [1, 3], [1, 4]) AS intersect
```



## arrayReduce

集計関数を配列要素に適用し、その結果を返します。集計関数の名前は、单一引 quotes で文字列として渡されます 'max', 'sum'。パラメトリック集計関数を使用する場合、パラメーターは関数名の後に括弧で示されます 'uniqUpTo(6)'。

### 構文

```
arrayReduce(agg_func, arr1, arr2, ..., arrN)
```

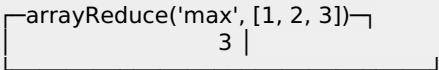
### パラメータ

- `agg_func` — The name of an aggregate function which should be a constant 文字列.
- `arr` — Any number of 配列 集計関数のパラメーターとして列を入力します。

### 戻り値

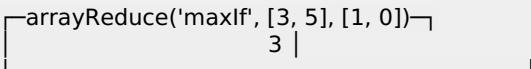
#### 例

```
SELECT arrayReduce('max', [1, 2, 3])
```



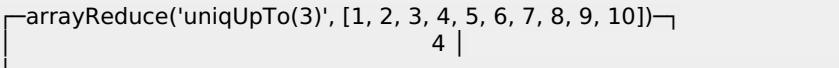
集計関数が複数の引数を取る場合、この関数は同じサイズの複数の配列に適用する必要があります。

```
SELECT arrayReduce('maxIf', [3, 5], [1, 0])
```



### パラメトリック集計関数の例:

```
SELECT arrayReduce('uniqUpTo(3)', [1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```



## arrayReduceInRanges

指定された範囲の配列要素に集計関数を適用し、各範囲に対応する結果を含む配列を返します。関数は `multiple` と同じ結果を返します `arrayReduce(agg_func, arraySlice(arr1, index, length), ...)`.

### 構文

```
arrayReduceInRanges(agg_func, ranges, arr1, arr2, ..., arrN)
```

## パラメータ

- `agg_func` — The name of an aggregate function which should be a constant 文字列.
- `ranges` — The ranges to aggregate which should be an 配列 の タプル これには、各範囲のインデックスと長さが含まれます。
- `arr` — Any number of 配列 集計関数のパラメーターとして列を入力します。

## 戻り値

### 例

```
SELECT arrayReduceInRanges(  
    'sum',  
    [(1, 5), (2, 3), (3, 4), (4, 4)],  
    [1000000, 200000, 30000, 4000, 500, 60, 7]  
) AS res
```

```
res  
[1234500,234000,34560,4567] |
```

## アレイリバース(arr)

逆の順序で要素を含む元の配列と同じサイズの配列を返します。

例:

```
SELECT arrayReverse([1, 2, 3])
```

```
arrayReverse([1, 2, 3])  
[3,2,1] |
```

## リバース(arr)

の同義語 “arrayReverse”

## アレイフラッテン

配列の配列をフラット配列に変換します。

関数:

- ネストされた配列の任意の深さに適用されます。
- 既にフラットな配列は変更しません。

の平坦化された配列を含むすべての要素をすべてソース配列.

## 構文

```
flatten(array_of_arrays)
```

別名: `flatten`.

パラメータ

- `array_of_arrays` — 配列 配列の。例えば, `[[1,2,3], [4,5]]`.

例

```
SELECT flatten([[1], [2], [3]])
```

```
  flatten(array(array([1]), array([2], [3])))  
  [1,2,3]
```

## アレイコンパクト

配列から連続した重複要素を削除します。結果の値の順序は、ソース配列の順序によって決まります。

構文

```
arrayCompact(arr)
```

パラメータ

`arr` — The 配列 検査する。

戻り値

重複のない配列。

タイプ: `Array`.

例

クエリ:

```
SELECT arrayCompact([1, 1, nan, nan, 2, 3, 3, 3])
```

結果:

```
  arrayCompact([1, 1, nan, nan, 2, 3, 3, 3])  
  [1,nan,nan,2,3]
```

## arrayZip

複数の配列を单一の配列に結合します。結果の配列には、引数の順序で組にグループ化されたソース配列の対応する要素が含まれます。

構文

```
arrayZip(arr1, arr2, ..., arrN)
```

パラメータ

- `arrN` — 配列.

関数は、異なる型の任意の数の配列を取ることができます。すべての入力配列は同じサイズでなければなりません。

戻り値

- ソース配列の要素をグループ化した配列 **タプル**。タプル内のデータ型は、入力配列の型と同じで、配列が渡される順序と同じです。

タイプ: **配列**.

#### 例

クエリ:

```
SELECT arrayZip(['a', 'b', 'c'], [5, 2, 1])
```

結果:

```
arrayZip(['a', 'b', 'c'], [5, 2, 1])—  
[('a',5),('b',2),('c',1)] |
```

## アレイオ一ク

計算AUC（機械学習の概念である曲線の下の面積、詳細を参照してください

い: [https://en.wikipedia.org/wiki/Receiver\\_operating\\_characteristic#Area\\_under\\_the\\_curve](https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Area_under_the_curve)）。

#### 構文

```
arrayAUC(arr_scores, arr_labels)
```

パラメータ

- arr\_scores — scores prediction model gives.

- arr\_labels — labels of samples, usually 1 for positive sample and 0 for negative sample.

#### 戻り値

FLOAT64型のAUC値を返します。

#### 例

クエリ:

```
select arrayAUC([0.1, 0.4, 0.35, 0.8], [0, 0, 1, 1])
```

結果:

```
arrayAUC([0.1, 0.4, 0.35, 0.8], [0, 0, 1, 1])—  
0.75 |
```

## 文字列と配列を分割および結合するための関数

### splitByChar(セパレーター、s)

文字列を、指定した文字で区切られた部分文字列に分割します。定数文字列を使用します **separator** これは正確に一つの文字で構成されます。

選択した部分文字列の配列を返します。文字列の先頭または末尾に区切り文字がある場合、または複数の連続した区切り文字がある場合は、空の部分文字列を選択できます。

#### 構文

```
splitByChar(<separator>, <s>)
```

#### パラメータ

- **separator** — The separator which should contain exactly one character. 文字列.
- **s** — The string to split. 文字列.

#### 戻り値)

選択した部分文字列の配列を返します。 空の部分文字列は、次の場合に選択できます:

- 区切り文字は、文字列の先頭または末尾にあります;
- 複数の連続した区切り文字があります;
- 元の文字列 **s** 空です。

タイプ: 配列 の 文字列.

#### 例

```
SELECT splitByChar(',', '1,2,3,abcde')
```

```
splitByChar(',', '1,2,3,abcde') └  
['1','2','3','abcde'] ┌
```

## splitByString(セパレーター、s)

文字列を文字列で区切られた部分文字列に分割します。 定数文字列を使用します **separator** 区切り文字として複数の文字を指定します。 文字列の場合 **separator** 空の場合は、文字列を分割します **s** 単一の文字の配列に。

#### 構文

```
splitByString(<separator>, <s>)
```

#### パラメータ

- **separator** — The separator. 文字列.
- **s** — The string to split. 文字列.

#### 戻り値)

選択した部分文字列の配列を返します。 空の部分文字列は、次の場合に選択できます:

タイプ: 配列 の 文字列.

- 空でない区切り文字は、文字列の先頭または末尾にあります;
- 連続した空でない区切り文字が複数あります;
- 元の文字列 **s** 区切り文字が空でない間は空です。

#### 例

```
SELECT splitByString(' ', '1, 2, 3, 4,5, abcde')
```

```
splitByString(',', '1, 2 3, 4,5, abcde')  
['1','2 3','4,5','abcde']
```

```
SELECT splitByString("", 'abcde')
```

```
splitByString("", 'abcde')  
['a','b','c','d','e']
```

## arrayStringConcat(arr[, 区切り記号])

配列にリストされている文字列を区切り文字で連結します。'separator'はオプションのパラメータです:定数文字列で、デフォルトでは空の文字列に設定されます。

文字列を返します。

## アルファトケンス(曖昧さ回避)

A-ZおよびA-Zの範囲から連続したバイトの部分文字列を選択します。

例

```
SELECT alphaTokens('abca1abc')
```

```
alphaTokens('abca1abc')  
['abca','abc']
```

## ビット関数

ビット関数は、UInt8、UInt16、UInt32、UInt64、Int8、Int16、Int32、Int64、Float32、Float64のいずれかの型のペアに対しても機能します。

結果の型は、引数の最大ビットに等しいビットを持つ整数です。引数のうち少なくともいずれかが署名されている場合、結果は符号付き番号になります。引数が浮動小数点数の場合は、Int64にキャストされます。

### ビタン(a,b)

### ビタ一(a,b)

### bitXor(a,b)

### bitNot(a)

### ビットシフトレフト(a,b)

### ビットシフトライト(a,b)

### ビットロタテレフト(a,b)

### ビットロータライト(a,b)

# bitTest

間の任意の整数に換算しています。 **バイナリ形式**, 指定された位置にあるビットの値を返します。 カウントダウンは右から左に0から始まります。

## 構文

```
SELECT bitTest(number, index)
```

### パラメータ

- **number** – integer number.
- **index** – position of bit.

### 戻り値

指定された位置にあるbitの値を返します。

タイプ: **UInt8**.

### 例

たとえば、基数43-2(二進数)の数値システムでは101011です。

クエリ:

```
SELECT bitTest(43, 1)
```

結果:

```
bitTest(43, 1)  
1 |
```

別の例:

クエリ:

```
SELECT bitTest(43, 2)
```

結果:

```
bitTest(43, 2)  
0 |
```

## ビットスター

の結果を返します **論理結合** 指定された位置にあるすべてのビットの(and演算子)。 カウントダウンは右から左に0から始まります。

ビットごとの演算のための結合:

0 AND 0 = 0

0 AND 1 = 0

**1 AND 0 = 0**

**1 AND 1 = 1**

## 構文

```
SELECT bitTestAll(number, index1, index2, index3, index4, ...)
```

## パラメータ

- **number** – integer number.
- **index1, index2, index3, index4** – positions of bit. For example, for set of positions (index1, index2, index3, index4)は、すべての位置が真である場合にのみtrueです ( $\text{index1} \wedge \text{index2}, \wedge \text{index3} \wedge \text{index4}$ ).

## 戻り値

論理結合の結果を返します。

タイプ: **UInt8**.

## 例

たとえば、基数43-2(二進数)の数値システムでは101011です。

クエリ:

```
SELECT bitTestAll(43, 0, 1, 3, 5)
```

結果:

```
bitTestAll(43, 0, 1, 3, 5)─  
 1 | ─
```

別の例:

クエリ:

```
SELECT bitTestAll(43, 0, 1, 3, 5, 2)
```

結果:

```
bitTestAll(43, 0, 1, 3, 5, 2)─  
 0 | ─
```

## ビットマスク

の結果を返します **論理和** 指定された位置にあるすべてのビットの（または演算子）。カウントダウンは右から左に0から始まります。

ビットごとの演算の分離:

**0 OR 0 = 0**

**0 OR 1 = 1**

**1 OR 0 = 1**

**1 OR 1 = 1**

構文

```
SELECT bitTestAny(number, index1, index2, index3, index4, ...)
```

パラメータ

- `number` – integer number.
- `index1, index2, index3, index4` – positions of bit.

戻り値

論理解釈の結果を返します。

タイプ: UInt8.

例

たとえば、基数43-2(二進数)の数値システムでは101011です。

クエリ:

```
SELECT bitTestAny(43, 0, 2)
```

結果:

```
bitTestAny(43, 0, 2)  
1 |
```

別の例:

クエリ:

```
SELECT bitTestAny(43, 4, 2)
```

結果:

```
bitTestAny(43, 4, 2)  
0 |
```

## ビット数

数値のバイナリ表現で一つに設定されたビット数を計算します。

構文

```
bitCount(x)
```

パラメータ

- **x** – 整数 または 浮動小数点数 番号 この関数は、メモリ内の値表現を使用します。浮動小数点数をサポートできます。

## 戻り値

- 入力番号の一つに設定されたビット数。

この関数は、入力値をより大きな型に変換しません (符号拡張). 例えば, `bitCount(toUInt8(-1)) = 8.`

タイプ: `UInt8.`

## 例

たとえば、数333を取る。そのバイナリ表現: 00000000101001101。

クエリ:

```
SELECT bitCount(333)
```

結果:

```
bitCount(333) └─  
      5 |
```

## ビットマップ関数

`And`、`or`、`xor`、`not`などの式の計算を使用しながら、新しいビットマップまたは基底を返すことです。

ビットマップオブジェクトの構築方法には2種類あります。一つは-`State`を持つ集計関数`groupBitmap`によって構築され、もう一つは`Array Object`によって構築されます。でも変換するビットマップオブジェクト配列のオブジェクトです。

`RoaringBitmap`は、ビットマップオブジェクトの実際の格納中にデータ構造にラップされます。基底が32以下の場合は、`Set object`が使用されます。基底が32より大きい場合は、`RoaringBitmap`オブジェクトを使用します。そのため、低基底セットの保存が高速になります。

`RoaringBitmap`の詳細については、以下を参照してください: [クロアリング](#).

## bitmapBuild

符号なし整数配列からビットマップを作成します。

```
bitmapBuild(array)
```

## パラメータ

- **array** – unsigned integer array.

## 例

```
SELECT bitmapBuild([1, 2, 3, 4, 5]) AS res, toTypeName(res)
```

```
res ── toTypeName(bitmapBuild([1, 2, 3, 4, 5])) ──  
      | AggregateFunction(groupBitmap, UInt8) |
```

## bitmapToArray

ビットマップを整数配列に変換します。

```
bitmapToArray(bitmap)
```

パラメータ

- bitmap – bitmap object.

例

```
SELECT bitmapToArray(bitmapBuild([1, 2, 3, 4, 5])) AS res
```

```
res  
[1,2,3,4,5] |
```

## bitmapSubsetInRange

指定された範囲のサブセットを返します(range\_endは含まれません)。

```
bitmapSubsetInRange(bitmap, range_start, range_end)
```

パラメータ

- bitmap – ビットマップ.
- range\_start – range start point. Type: **UInt32**.
- range\_end – range end point(excluded). Type: **UInt32**.

例

```
SELECT  
bitmapToArray(bitmapSubsetInRange(bitmapBuild([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,  
25,26,27,28,29,30,31,32,33,100,200,500]), toUInt32(30), toUInt32(200))) AS res
```

```
res  
[30,31,32,33,100] |
```

## bitmapSubsetLimit

ビットマップのサブセットを作成します。 range\_start と cardinality\_limit.

構文

```
bitmapSubsetLimit(bitmap, range_start, cardinality_limit)
```

パラメータ

- bitmap – ビットマップ.
- range\_start – The subset starting point. Type: **UInt32**.

- **cardinality\_limit** – The subset cardinality upper limit. Type: **UInt32**.

#### 戻り値

サブセット。

タイプ: **Bitmap object**.

#### 例

クエリ:

```
SELECT
bitmapToArray(bitmapSubsetLimit(bitmapBuild([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,
26,27,28,29,30,31,32,33,100,200,500]), toUInt32(30), toUInt32(200))) AS res
```

結果:

```
res
[30,31,32,33,100,200,500] |
```

## bitmapContains

かどうかをチェックしますビットマップを含む要素になります。

```
bitmapContains(haystack, needle)
```

#### パラメータ

- **haystack** – ビットマップ、関数が検索する場所。
- **needle** – Value that the function searches. Type: **UInt32**.

#### 戻り値

- 0 — If **haystack** 含まない **needle**.

- 1 — If **haystack** 含む **needle**.

タイプ: **UInt8**.

#### 例

```
SELECT bitmapContains(bitmapBuild([1,5,7,9]), toUInt32(9)) AS res
```

```
res
1 |
```

## bitmapHasAny

るかどうかを判二つのビットマップしていることで交差点にある。

```
bitmapHasAny(bitmap1, bitmap2)
```

あなたがそれを確信している場合 `bitmap2` 厳密に一つの要素が含まれています。`bitmapContains` 機能。これは、より効率的に動作します。

#### パラメータ

- `bitmap*` – bitmap object.

#### 戻り値

- `1`,if `bitmap1` と `bitmap2` 少なくとも一つの同様の要素があります。

- `0` そうでなければ

#### 例

```
SELECT bitmapHasAny(bitmapBuild([1,2,3]),bitmapBuild([3,4,5])) AS res
```

res
1

## bitmapHasAll

類似する `hasAll(array, array)` 戻り値が`1`の場合は最初のビットマップを含むすべての要素は、`0`です。

二番目の引数が空のビットマップの場合は、`1`を返します。

```
bitmapHasAll(bitmap,bitmap)
```

#### パラメータ

- `bitmap` – bitmap object.

#### 例

```
SELECT bitmapHasAll(bitmapBuild([1,2,3]),bitmapBuild([3,4,5])) AS res
```

res
0

## bitmapCardinality

UInt64型のビットマップ基数を再実行します。

```
bitmapCardinality(bitmap)
```

#### パラメータ

- `bitmap` – bitmap object.

#### 例

```
SELECT bitmapCardinality(bitmapBuild([1, 2, 3, 4, 5])) AS res
```

```
res  
5 |
```

## bitmapMin

セット内のUInt64型の最小値を再実行し、セットが空の場合はUINT32\_MAX。

```
bitmapMin(bitmap)
```

パラメータ

- `bitmap` – bitmap object.

例

```
SELECT bitmapMin(bitmapBuild([1, 2, 3, 4, 5])) AS res
```

```
res  
1 |
```

## bitmapMax

セット内のUInt64型の最大値を再試行し、セットが空の場合は0を再試行します。

```
bitmapMax(bitmap)
```

パラメータ

- `bitmap` – bitmap object.

例

```
SELECT bitmapMax(bitmapBuild([1, 2, 3, 4, 5])) AS res
```

```
res  
5 |
```

## bitmapTransform

ビットマップ内の値の配列を別の値の配列に変換すると、結果は新しいビットマップになります。

```
bitmapTransform(bitmap, from_array, to_array)
```

パラメータ

- `bitmap` – bitmap object.
- `from_array` – UInt32 array. For idx in range [0, `from_array.size()`), if `bitmap` contains `from_array[idx]`, then replace it with `to_array[idx]`. Note that the result depends on array ordering if there are common elements between `from_array` and `to_array`.

- `to_array` – UInt32 array, its size shall be the same to `from_array`.

例

```
SELECT bitmapToArray(bitmapTransform(bitmapBuild([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]), cast([5,999,2] as Array(UInt32)),  
cast([2,888,20] as Array(UInt32)))) AS res
```

```
res  
[1,3,4,6,7,8,9,10,20] |
```

## bitmapAnd

二つのビットマップと計算、結果は新しいビットマップです。

```
bitmapAnd(bitmap,bitmap)
```

パラメータ

- `bitmap` – bitmap object.

例

```
SELECT bitmapToArray(bitmapAnd(bitmapBuild([1,2,3]),bitmapBuild([3,4,5]))) AS res
```

```
res  
[3] |
```

## bitmapOr

結果は新しいビットマップです。

```
bitmapOr(bitmap,bitmap)
```

パラメータ

- `bitmap` – bitmap object.

例

```
SELECT bitmapToArray(bitmapOr(bitmapBuild([1,2,3]),bitmapBuild([3,4,5]))) AS res
```

```
res  
[1,2,3,4,5] |
```

## bitmapXor

二つのビットマップ xor の計算は、結果は新しいビットマップです。

```
bitmapXor(bitmap,bitmap)
```

## パラメータ

- `bitmap` – bitmap object.

## 例

```
SELECT bitmapToArray(bitmapXor(bitmapBuild([1,2,3]),bitmapBuild([3,4,5]))) AS res
```

```
res  
[1,2,4,5] |
```

## bitmapAndnot

二つのビットマップ `andnot` 計算、結果は新しいビットマップです。

```
bitmapAndnot(bitmap,bitmap)
```

## パラメータ

- `bitmap` – bitmap object.

## 例

```
SELECT bitmapToArray(bitmapAndnot(bitmapBuild([1,2,3]),bitmapBuild([3,4,5]))) AS res
```

```
res  
[1,2] |
```

## bitmapAndCardinality

Uint64型の基数を返します。

```
bitmapAndCardinality(bitmap,bitmap)
```

## パラメータ

- `bitmap` – bitmap object.

## 例

```
SELECT bitmapAndCardinality(bitmapBuild([1,2,3]),bitmapBuild([3,4,5])) AS res;
```

```
res  
1 |
```

## bitmapOrCardinality

Uint64型の基数を返します。

```
bitmapOrCardinality(bitmap,bitmap)
```

パラメータ

- `bitmap` – bitmap object.

例

```
SELECT bitmapOrCardinality(bitmapBuild([1,2,3]),bitmapBuild([3,4,5])) AS res;
```

```
res  
5 |
```

## bitmapXorCardinality

UInt64型の基数を返します。

```
bitmapXorCardinality(bitmap,bitmap)
```

パラメータ

- `bitmap` – bitmap object.

例

```
SELECT bitmapXorCardinality(bitmapBuild([1,2,3]),bitmapBuild([3,4,5])) AS res;
```

```
res  
4 |
```

## bitmapAndnotCardinality

UInt64型の基数を返します。

```
bitmapAndnotCardinality(bitmap,bitmap)
```

パラメータ

- `bitmap` – bitmap object.

例

```
SELECT bitmapAndnotCardinality(bitmapBuild([1,2,3]),bitmapBuild([3,4,5])) AS res;
```

```
res  
2 |
```

## ハッシュ関数

ハッシュ関数は、要素の決定論的pseudo似乱数シャッフルに使用することができます。

## halfMD5

**解釈する** すべての入力パラメータを文字列として計算し、**MD5** それぞれのハッシュ値。次に、ハッシュを結合し、結果の文字列のハッシュの最初の8バイトを取り、それらを次のように解釈します **UInt64** ビッグエンディアンのバイト順。

```
halfMD5(par1, ...)
```

この機能は比較的遅い（プロセッサコアあたり毎秒5万個の短い文字列）。  
の使用を検討します。**sipHash64** 代わりに関数。

パラメータ

この関数は、可変数の入力パラメータを受け取ります。パラメータは、以下のいずれかです **対応するデータ型**。

戻り値

A **UInt64** データ型のハッシュ値。

例

```
SELECT halfMD5(array('e','x','a'), 'mple', 10, toDateTime('2019-06-15 23:00:00')) AS halfMD5hash,  
toTypeName(halfMD5hash) AS type
```

halfMD5hash	type
186182704141653334	UInt64

## MD4

文字列からMD4を計算し、結果のバイトセットを**FixedString(16)**として返します。

## MD5

文字列からMD5を計算し、結果のバイトセットを**FixedString(16)**として返します。

特にMD5を必要としないが、まともな暗号化128ビットハッシュが必要な場合は、「**sipHash128**」代わりに関数。**Md5sum**ユーティリティによる出力と同じ結果を得たい場合は、**lower(hex(MD5(s)))**を使用します。

## sipHash64

64ビットを生成する **サイファッシュ** ハッシュ値。

```
sipHash64(par1,...)
```

これは暗号化ハッシュ関数です。それはより少なくとも三倍速く働きます **MD5** 機能。

関数 **解釈する** すべての入力パラメータを文字列として計算し、それぞれのハッシュ値を計算します。次のアルゴリズムでハッシュを結合します：

1. すべての入力パラメータをハッシュした後、関数はハッシュの配列を取得します。
2. 関数は、最初と第二の要素を取り、それらの配列のハッシュを計算します。
3. 次に、関数は、前のステップで計算されたハッシュ値、および最初のハッシュ配列の第三の要素を取り、それらの配列のハッシュを計算します。
4. 前のステップは、初期ハッシュ配列の残りのすべての要素に対して繰り返されます。

## パラメータ

この関数は、可変数の入力パラメータを受け取ります。パラメータは、以下のいずれかです **対応するデータ型**。

## 戻り値

A **UInt64** データ型のハッシュ値。

## 例

```
SELECT sipHash64(array('e','x','a'), 'mple', 10, toDateTime('2019-06-15 23:00:00')) AS SipHash, toTypeName(SipHash)
AS type
```

SipHash	type
13726873534472839665	UInt64

## sipHash128

文字列からSipHashを計算します。

文字列型の引数を受け取ります。 **FixedString(16)**を返します。

SipHash64とは異なり、最終的なxor折り畳み状態は128ビットまでしか行われない。

## シティハッシュ64

64ビットを生成する **シティハッシュ** ハッシュ値。

```
cityHash64(par1,...)
```

これは高速な非暗号ハッシュ関数です。 文字列パラメータには**CityHash**アルゴリズムを使用し、他のデータ型のパラメータには実装固有の高速非暗号化ハッシュ関数を使用します。 この関数は、最終的な結果を得るために**CityHash combinator**を使用します。

## パラメータ

この関数は、可変数の入力パラメータを受け取ります。パラメータは、以下のいずれかです **対応するデータ型**。

## 戻り値

A **UInt64** データ型のハッシュ値。

## 例

呼び出し例:

```
SELECT cityHash64(array('e','x','a'), 'mple', 10, toDateTime('2019-06-15 23:00:00')) AS CityHash,
toTypeName(CityHash) AS type
```

CityHash	type
12072650598913549138	UInt64

次の例では、行の順序までの精度でテーブル全体のチェックサムを計算する方法を示します:

```
SELECT groupBitXor(cityHash64(*)) FROM table
```

## intHash32

任意のタイプの整数から32ビットハッシュコードを計算します。  
これは、数値の平均品質の比較的高速な非暗号ハッシュ関数です。

## intHash64

任意のタイプの整数から64ビットハッシュコードを計算します。  
それはintHash32よりも速く動作します。 平均品質。

## SHA1

## SHA224

## SHA256

## SHA384

## SHA512

文字列からSHA-1、SHA-224、SHA-256、SHA-384、またはSHA-512を計算し、結果のバイトセットを  
FixedString(20)、FixedString(28)、FixedString(32)、FixedString(48)またはFixedString(64)として返します。  
この機能はかなりゆっくりと動作します（SHA-1はプロセッサコア毎秒約5万の短い文字列を処理しますが、SHA-224  
とSHA-256は約2.2万の短い文字列を処理  
この関数は、特定のハッシュ関数が必要で選択できない場合にのみ使用することをお勧めします。  
このような場合でも、SELECTに適用するのではなく、関数をオフラインで適用し、テーブルに挿入するときに値を事前に計算することをお勧めします。

## URLHash(url[,N])

何らかのタイプの正規化を使用してURLから取得された文字列に対する、高速でまともな品質の非暗号化ハッシュ関数。  
URLHash(s) – Calculates a hash from a string without one of the trailing symbols /,? または # 最後に、存在する場合。

URLHash(s, N) – Calculates a hash from a string up to the N level in the URL hierarchy, without one of the trailing symbols /,? または # 最後に、存在する場合。

レベルはURLHierarchyと同じです。 この機能はYandexに固有のものです。メトリカ

## farnhash64

64ビットを生成する フームハッシュ ハッシュ値。

```
farmHash64(par1, ...)
```

この関数は Hash64 すべてからの方法 利用可能な方法.

パラメータ

この関数は、可変数の入力パラメータを受け取ります。 パラメータは、以下のいずれかです 対応するデータ型.

戻り値

A UInt64 データ型のハッシュ値。

例

```
SELECT farmHash64(array('e','x','a'), 'mple', 10, toDateTime('2019-06-15 23:00:00')) AS FarmHash,  
toTypeName(FarmHash) AS type
```

FarmHash	type
17790458267262532859	UInt64

## javaHash

計算 **JavaHash** 文字列から。このハッシュ関数は高速でも良質でもありません。これを使用する唯一の理由は、このアルゴリズムが既に別のシステムで使用されており、まったく同じ結果を計算する必要がある場合です。

構文

```
SELECT javaHash();
```

戻り値

A Int32 データ型のハッシュ値。

例

クエリ:

```
SELECT javaHash('Hello, world!');
```

結果:

```
javaHash('Hello, world!')  
-1880044555
```

## javaHashUTF16LE

計算 **JavaHash** 文字列から、UTF-16LEエンコーディングで文字列を表すバイトが含まれていると仮定します。

構文

```
javaHashUTF16LE(stringUtf16le)
```

パラメータ

- **stringUtf16le** — a string in UTF-16LE encoding.

戻り値

A Int32 データ型のハッシュ値。

例

UTF-16LEエンコード文字列でクエリを修正します。

クエリ:

```
SELECT javaHashUTF16LE(convertCharset('test', 'utf-8', 'utf-16le'))
```

結果:

```
javaHashUTF16LE(convertCharset('test', 'utf-8', 'utf-16le'))→  
3556498 |
```

## hiveHash

計算 **HiveHash** 文字列から。

```
SELECT hiveHash("");
```

これはちょうどです **JavaHash** ゼロアウト符号ビットを持つ。この関数は **Apacheハイフ** 3.0より前のバージョンの場合。このハッシュ関数は高速でも良質でもありません。これを使用する唯一の理由は、このアルゴリズムが既に別のシステムで使用されており、まったく同じ結果を計算する必要がある場合です。

戻り値

A **Int32** データ型のハッシュ値。

タイプ: **hiveHash**.

例

クエリ:

```
SELECT hiveHash('Hello, world!');
```

結果:

```
hiveHash('Hello, world!')→  
267439093 |
```

## metroHash64

64ビットを生成する **メトロハッシュ** ハッシュ値。

```
metroHash64(par1, ...)
```

パラメータ

この関数は、可変数の入力パラメータを受け取ります。パラメータは、以下のいずれかです **対応するデータ型**.

戻り値

A **UInt64** データ型のハッシュ値。

例

```
SELECT metroHash64(array('e','x','a'), 'mple', 10, toDate('2019-06-15 23:00:00')) AS MetroHash,  
toTypeName(MetroHash) AS type
```

MetroHash	type
14235658766382344533	UInt64

## jumpConsistentHash

JumpConsistentHashを計算すると、UInt64を形成します。

UInt64型のキーとバケットの数です。 Int32を返します。

詳細は、リンクを参照してください: [JumpConsistentHash](#)

## murmurHash2\_32,murmurHash2\_64

を生成する。 [つぶやき2](#) ハッシュ値。

```
murmurHash2_32(par1, ...)
murmurHash2_64(par1, ...)
```

パラメータ

両方の関数は、可変数の入力パラメータを取ります。 パラメータは、以下のいずれかです [対応するデータ型](#).

戻り値

- その `murmurHash2_32` 関数はハッシュ値を返します。 [UInt32](#) データ型。
- その `murmurHash2_64` 関数はハッシュ値を返します。 [UInt64](#) データ型。

例

```
SELECT murmurHash2_64(array('e','x','a'), 'mple', 10, toDateTime('2019-06-15 23:00:00')) AS MurmurHash2,
toTypeName(MurmurHash2) AS type
```

MurmurHash2	type
11832096901709403633	UInt64

## gccMurmurHash

64ビットの計算 [つぶやき2](#) 同じハッシュシードを使用するハッシュ値 `gcc`. これは、CLangとGCCビルドの間で移植可能です。

構文

```
gccMurmurHash(par1, ...);
```

パラメータ

- `par1, ...` — A variable number of parameters that can be any of the [対応するデータ型](#).

戻り値

- 計算されたハッシュ値。

タイプ: [UInt64](#).

例

クエリ:

```
SELECT
    gccMurmurHash(1, 2, 3) AS res1,
    gccMurmurHash('a', [1, 2, 3], 4, (4, ['foo', 'bar'], 1, (1, 2))) AS res2
```

結果:

12384823029245979431	1188926775431157506
----------------------	---------------------

## murmurHash3\_32,murmurHash3\_64

を生成する。マムルハッシュ3世ハッシュ値。

```
murmurHash3_32(par1, ...)
murmurHash3_64(par1, ...)
```

パラメータ

両方の関数は、可変数の入力パラメータを取ります。パラメータは、以下のいずれかです [対応するデータ型](#)。

戻り値

- その `murmurHash3_32` 関数はaを返します `UInt32` データ型のハッシュ値。
- その `murmurHash3_64` 関数はaを返します `UInt64` データ型のハッシュ値。

例

```
SELECT murmurHash3_32(array('e','x','a'), 'mple', 10, toDateTime('2019-06-15 23:00:00')) AS MurmurHash3,
toTypeName(MurmurHash3) AS type
```

MurmurHash3	type
2152717	UInt32

## つぶやき3\_128

128ビットを生成するマムルハッシュ3世ハッシュ値。

```
murmurHash3_128( expr )
```

パラメータ

- `expr` — 式 aを返す文字列-タイプ値。

戻り値

A `FixedString(16)` データ型のハッシュ値。

例

```
SELECT hex(murmurHash3_128('example_string')) AS MurmurHash3, toTypeName(MurmurHash3) AS type;
```

```
MurmurHash3----- type-----  
368A1A311CB7342253354B548E7E7E71 | String |
```

## xxHash32,xxHash64

計算 `xxHash` 文字列から。これは、二つの味、32と64ビットで提案されています。

```
SELECT xxHash32('');
```

OR

```
SELECT xxHash64('');
```

### 戻り値

A `UInt32` または `UInt64` データ型のハッシュ値。

タイプ: `xxHash`.

### 例

クエリ:

```
SELECT xxHash32('Hello, world!');
```

結果:

```
xxHash32('Hello, world!')-----  
834093149 |
```

も参照。

■ [xxHash](#).

## 擬似乱数を生成するための関数

擬似乱数の非暗号生成器が使用される。

全ての機能を受け入れゼロの引数または一つの引数。

引数が渡された場合は、任意の型にすることができ、その値は何にも使用されません。

この引数の唯一の目的は、同じ関数の二つの異なるインスタンスが異なる乱数を持つ異なる列を返すように、共通の部分式の除去を防ぐことです。

### ランド

すべての `UInt32` 型の数値に均等に分布する擬似ランダムな `UInt32` 数値を返します。

線形合同ジェネレータを使用します。

### rand64

すべての `UInt64` 型の数値に均等に分布する擬似ランダムな `UInt64` 数値を返します。

線形合同ジェネレータを使用します。

### ランドコンスタント

ランダムな値を持つ定数列を生成します。

## 構文

```
randConstant([x])
```

### パラメータ

- `x` — 式の何れかに終って **対応するデータ型**。結果の値は破棄されますが、式自体がバイパスに使用されている場合は破棄されます **共通の部分式の除去** 関数が一つのクエリで複数回呼び出された場合。任意パラメータ。

### 戻り値

- 擬似乱数。

タイプ: **UInt32**.

### 例

クエリ:

```
SELECT rand(), rand(1), rand(number), randConstant(), randConstant(1), randConstant(number)  
FROM numbers(3)
```

結果:

rand()	rand(1)	rand(number)	randConstant()	randConstant(1)	randConstant(number)
3047369878	4132449925	4044508545	2740811946	4229401477	1924032898
2938880146	1267722397	4154983056	2740811946	4229401477	1924032898
956619638	4238287282	1104342490	2740811946	4229401477	1924032898

## エンコード機能

### char

渡された引数の数として長さの文字列を返し、各バイトは対応する引数の値を持ちます。数値型の複数の引数を受け取ります。引数の値が **UInt8** データ型の範囲外である場合、丸めとオーバーフローの可能性がある **UInt8** に変換されます。

### 構文

```
char(number_1, [number_2, ..., number_n]);
```

### パラメータ

- `number_1, number_2, ..., number_n` — Numerical arguments interpreted as integers. Types: **Int**, **Float**.

### 戻り値

- 指定されたバイトの文字列。

タイプ: **String**.

### 例

クエリ:

```
SELECT char(104.1, 101, 108.9, 108.9, 111) AS hello
```

結果:

```
hello  
hello |
```

を構築できます文字列の任意のエンコードに対応するバイトまでとなります。UTF-8の例を次に示します:

クエリ:

```
SELECT char(0xD0, 0xBF, 0xD1, 0x80, 0xD0, 0xB8, 0xD0, 0xB2, 0xD0, 0xB5, 0xD1, 0x82) AS hello;
```

結果:

```
hello  
привет |
```

クエリ:

```
SELECT char(0xE4, 0xBD, 0xA0, 0xE5, 0xA5, 0xBD) AS hello;
```

結果:

```
hello  
你好 |
```

## hex

引数の十六進表現を含む文字列を返します。

構文

```
hex(arg)
```

関数は大文字を使用しています A-F そして、接頭辞を使用しない (のような 0x) または接尾辞 (のような h).

整数引数の場合は、六桁を出力します ("nibbles") 最上位から最下位へ(ビッグエンディアンまたは "human readable" 順序)。これは、最上位の非ゼロバイト (先頭のゼロバイトは省略されます) で始まりますが、先頭の数字がゼロであっても、常にすべてのバイトの両方の桁を

例:

例

クエリ:

```
SELECT hex(1);
```

結果:

型の値 `Date` と `DateTime` 対応する整数(`Date`の場合はエポックからの日数、`DateTime`の場合はUnix Timestampの値)としてフォーマットされます。

のために `String` と `FixedString`、すべてのバイトは単に二進数として符号化される。ゼロバイトは省略されません。

浮動小数点型と `Decimal` 型の値は、メモリ内の表現としてエンコードされます。支援においても少しエンディアン、建築、その符号化されたのでちょっとエンディアンです。※ 先頭/末尾のゼロバイトは省略されません。

#### パラメータ

- `arg` — A value to convert to hexadecimal. Types: 文字列, `UInt`, フロート, 小数点, 日付 または `DateTime`.

#### 戻り値

- 引数の十六進表現を持つ文字列。

タイプ: `String`.

#### 例

クエリ:

```
SELECT hex(toFloat32(number)) as hex_presentation FROM numbers(15, 2);
```

結果:

hex_presentation
00007041
00008041

クエリ:

```
SELECT hex(toFloat64(number)) as hex_presentation FROM numbers(15, 2);
```

結果:

hex_presentation
0000000000002E40
0000000000003040

## unhex(str)

任意の数の十六進数を含む文字列を受け取り、対応するバイトを含む文字列を返します。大文字と小文字の両方をサポートしていますa-F進の桁数は偶数である必要はありません。奇数の場合、最後の桁は00-0Fバイトの最下位半分として解釈されます。引数文字列に十六進以外の桁数が含まれている場合、実装定義の結果が返されます(例外はスローされません)。

結果を数値に変換する場合は、「reverse」と「reinterpretAsType」機能。

## UUIDStringToNum(str)

形式で36文字を含む文字列を受け入れます `123e4567-e89b-12d3-a456-426655440000` これを `FixedString(16)` 内のバイトセットとして返す。

## UUIDNumToString(str)

FixedSize(16)値を受け取ります。36文字を含む文字列をテキスト形式で返します。

## ビットマスクリスト(num)

整数を受け取ります。ソース番号を合計する二つの累乗のリストを含む文字列を返します。これらは、テキスト形式のスペースなしで、昇順でコンマ区切りです。

## ビットマスクトアレイ(num)

整数を受け取ります。合計されたときにソース番号を合計する二つのべき乗のリストを含むUInt64数値の配列を返します。配列内の数値は昇順です。

# UUIDを操作するための関数

UUIDを操作するための関数を以下に示します。

## generateUUIDv4

を生成する。UUIDのバージョン4。

```
generateUUIDv4()
```

戻り値

UUID型の値。

使用例

この例では、UUID型の列を持つテーブルを作成し、テーブルに値を挿入する方法を示します。

```
CREATE TABLE t_uuid (x UUID) ENGINE=TinyLog  
INSERT INTO t_uuid SELECT generateUUIDv4()  
SELECT * FROM t_uuid
```

f4bf890f-f9dc-4332-ad5c-0c18e73f28e9 |<sup>x</sup>

## トワイド(x)

文字列型の値をUUID型に変換します。

```
toUUID(String)
```

戻り値

UUID型の値。

使用例

```
SELECT toUUID('61f0c404-5cb3-11e7-907b-a6006ad3dba0') AS uuid
```

```
61f0c404-5cb3-11e7-907b-a6006ad3dba0 |  
          ^uuid
```

## UUIDStringToNum

形式で36文字を含む文字列を受け入れます `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx` これを、aのバイトセットとして返します **FixedString(16)**.

```
UUIDStringToNum(String)
```

戻り値

**FixedString(16)**

使用例

```
SELECT  
'612f3c40-5d3b-217e-707b-6a546a3d7b29' AS uuid,  
UUIDStringToNum(uuid) AS bytes
```

```
uuid———bytes———  
612f3c40-5d3b-217e-707b-6a546a3d7b29 | a/<@];!~p{jTj={} |
```

## UUIDNumToString

受け入れるa **FixedString(16)** テキスト形式で36文字を含む文字列を返します。

```
UUIDNumToString(FixedString(16))
```

戻り値

文字列

使用例

```
SELECT  
'a/<@];!~p{jTj={} AS bytes,  
UUIDNumToString(toFixedString(bytes, 16)) AS uuid
```

```
bytes———uuid———  
a/<@];!~p{jTj={} | 612f3c40-5d3b-217e-707b-6a546a3d7b29 |
```

も参照。

- [dictGetUUID](#)

## Urlを操作するための関数

これらの関数はすべてRFCに従っていません。 それらは改善された性能のために最大限に簡単にされる。

### URLの一部を抽出する関数

関連する部分がURLにない場合は、空の文字列が返されます。

## プロトコル

URLからプロトコルを抽出します。

Examples of typical returned values: http, https, ftp, mailto, tel, magnet...

## ドメイン

URLからホスト名を抽出します。

domain(url)

パラメータ

- url — URL. Type: 文字列.

URLは、スキームの有無にかかわらず指定できます。例:

```
svn+ssh://some.svn-hosting.com:80/repo/trunk  
some.svn-hosting.com:80/repo/trunk  
https://yandex.com/time/
```

これらの例では、`domain` 関数は、次の結果を返します:

```
some.svn-hosting.com  
some.svn-hosting.com  
yandex.com
```

## 戻り値

- ホスト名。ClickHouseが入力文字列をURLとして解析できる場合。
- 空の文字列。ClickHouseが入力文字列をURLとして解析できない場合。

タイプ: String.

例

```
SELECT domain('svn+ssh://some.svn-hosting.com:80/repo/trunk')
```

```
domain('svn+ssh://some.svn-hosting.com:80/repo/trunk')—  
some.svn-hosting.com |
```

## domainWithoutWWW

ドメインを返し、複数のドメインを削除しません ‘www.’ その始めから、存在する場合。

## トップレベルドメイン

URLからトップレベルドメインを抽出します。

topLevelDomain(url)

## パラメータ

- `url` — URL. Type: 文字列.

URLは、スキームの有無にかかわらず指定できます。例:

```
svn+ssh://some.svn-hosting.com:80/repo/trunk  
some.svn-hosting.com:80/repo/trunk  
https://yandex.com/time/
```

## 戻り値

- ドメイン名。ClickHouseが入力文字列をURLとして解析できる場合。
- 空の文字列。ClickHouseが入力文字列をURLとして解析できない場合。

タイプ: `String`.

## 例

```
SELECT topLevelDomain('svn+ssh://www.some.svn-hosting.com:80/repo/trunk')
```

```
topLevelDomain('svn+ssh://www.some.svn-hosting.com:80/repo/trunk')—  
com |
```

## firstSignificantSubdomain

を返す。“first significant subdomain”. これはYandexに固有の非標準的な概念です。メトリカ 最初の重要なサブドメインは、次の場合にセカンドレベルドメインです ‘com’, ‘net’, ‘org’, または ‘co’. それ以外の場合は、サードレベルドメインです。例えば, `firstSignificantSubdomain ('https://news.yandex.ru/')` = ‘yandex’, `firstSignificantSubdomain ('https://news.yandex.com.tr/')` = ‘yandex’. のリスト “insignificant” セカンドレベルドメイ

## cutToFirstSignificantSubdomain

トップレベルのサブドメインを含むドメインの一部を返します。“first significant subdomain”（上記の説明を参照）。

例えば, `cutToFirstSignificantSubdomain('https://news.yandex.com.tr/')` = ‘yandex.com.tr’.

## パス

パスを返します。例: `/top/news.html` パスにはクエリ文字列は含まれません。

## パスフル

上記と同じですが、クエリ文字列とフラグメントを含みます。例: `/top/news.html?ページ=2#コメント`

## クエリ文字列

クエリ文字列を返します。例: `ページ=1&lr=213`. query-stringには、最初の疑問符だけでなく、#と#の後のすべても含まれていません。

## 断片

フラグメント識別子を返します。fragmentには初期ハッシュ記号は含まれません。

## queryStringAndFragment

クエリ文字列とフラグメント識別子を返します。例:ページ=1#29390.

## extractURLParameter(URL,名前)

の値を返します。'name' URLにパラメータがある場合。それ以外の場合は、空の文字列です。この名前のパラメーターが多い場合は、最初に出現するパラメーターを返します。この関数は、パラメータ名が渡された引数とまったく同じ方法でURLにエンコードされるという前提の下で機能します。

## extractURLParameters(URL)

URLパラメーターに対応するname=value文字列の配列を返します。値はどのような方法でもデコードされません。

## extractURLParameterNames(URL)

URLパラメーターの名前に対応する名前文字列の配列を返します。値はどのような方法でもデコードされません。

## URLHierarchy(URL)

URLを含む配列を返します。パスとクエリ文字列で。連続セパレータ文字として数えます。カットは、すべての連続した区切り文字の後の位置に行われます。

## URLPathHierarchy(URL)

上記と同じですが、結果にプロトコルとホストはありません。要素(ルート)は含まれません。例:関数は、ツリーを実装するために使用されるYandexの中のURLを報告します。メートル法

```
URLPathHierarchy('https://example.com/browse/CONV-6788') =  
[  
    '/browse/',  
    '/browse/CONV-6788'  
]
```

## decodeURLComponent(URL)

デコードしたURLを返します。

例:

```
SELECT decodeURLComponent('http://127.0.0.1:8123/?query=SELECT%201%3B') AS DecodedURL;
```

```
DecodedURL  
http://127.0.0.1:8123/?query=SELECT 1; |
```

## URLの一部を削除する関数

URLに類似のものがない場合、URLは変更されません。

## cutWWW

複数を削除しません 'www.' URLのドメインの先頭から存在する場合。

## cutQueryString

クエリ文字列を削除します。疑問符も削除されます。

## カットフラグメント

フラグメント識別子を削除します。番号記号も削除されます。

## cutQueryStringAndFragment

クエリ文字列とフラグメント識別子を削除します。疑問符と番号記号も削除されます。

**cutURLParameter(URL,名前)**

を削除します。'name' URLパラメータが存在する場合。この関数は、パラメータ名が渡された引数とまったく同じ方法でURLにエンコードされるという前提の下で機能します。

## IPアドレスを操作するための関数

## IPv4NumToString(num)

UInt32番号を取ります。ビッグエンディアンのIPv4アドレスとして解釈します。対応するIPv4アドレスを含む文字列を、a.B.C.d(小数点以下のドット区切りの数値)の形式で返します。

## IPv4StringToNum(s)

`IPv4NumToString`の逆関数。 IPv4アドレスの形式が無効な場合は、0を返します。

**IPv4NumToStringClassC(num)**

`IPv4NumToString`に似ていますが、最後のオクテットの代わりに`xxx`を使用します。

例：

```
SELECT
    IPv4NumToStringClassC(ClientIP) AS k,
    count() AS c
FROM test.hits
GROUP BY k
ORDER BY c DESC
LIMIT 10
```

k		c
83.149.9.xxx	26238	
217.118.81.xxx	26074	
213.87.129.xxx	25481	
83.149.8.xxx	24984	
217.118.83.xxx	22797	
78.25.120.xxx	22354	
213.87.131.xxx	21285	
78.25.121.xxx	20887	
188.162.65.xxx	19694	
83.149.48.xxx	17406	

を使用して以来 'XXX' 非常に珍しいですが、これは将来的に変更される可能性があります。このフラグメントの正確な形式に依存しないことをお勧めします。

## IPv6NumToString(x)

バイナリ形式のIPv6アドレスを含む**FixedString(16)**値を受け入れます。このアドレスを含む文字列をテキスト形式で返します。

IPv6マップされたIPv4アドレスは、::ffff:111.222.33.44の形式で出力されます。例：

```
addr  
2a02:6b8::11 |
```

```
SELECT  
    IPv6NumToString(ClientIP6 AS k),  
    count() AS c  
FROM hits_all  
WHERE EventDate = today() AND substring(ClientIP6, 1, 12) != unhex('00000000000000000000FFFF')  
GROUP BY k  
ORDER BY c DESC  
LIMIT 10
```

```
IPv6NumToString(ClientIP6)-----c-----  
2a02:2168:aaa:bbbb::2 | 24695 |  
2a02:2698:abcd:abcd:abcd:8888:5555 | 22408 |  
2a02:6b8:0:fff::ff | 16389 |  
2a01:4f8:111:6666::2 | 16016 |  
2a02:2168:888:222::1 | 15896 |  
2a01:7e00::ffff:ffff:ffff:222 | 14774 |  
2a02:8109:eee:ee:eeee:eeee:eeee:eee | 14443 |  
2a02:810b:8888:888:8888:8888:8888:8888 | 14345 |  
2a02:6b8:0:444:4444:4444:4444:4444 | 14279 |  
2a01:7e00::ffff:ffff:ffff:ffff | 13880 |
```

```
SELECT  
    IPv6NumToString(ClientIP6 AS k),  
    count() AS c  
FROM hits_all  
WHERE EventDate = today()  
GROUP BY k  
ORDER BY c DESC  
LIMIT 10
```

```
IPv6NumToString(ClientIP6)-----c-----  
::ffff:94.26.111.111 | 747440 |  
::ffff:37.143.222.4 | 529483 |  
::ffff:5.166.111.99 | 317707 |  
::ffff:46.38.11.77 | 263086 |  
::ffff:79.105.111.111 | 186611 |  
::ffff:93.92.111.88 | 176773 |  
::ffff:84.53.111.33 | 158709 |  
::ffff:217.118.11.22 | 154004 |  
::ffff:217.118.11.33 | 148449 |  
::ffff:217.118.11.44 | 148243 |
```

## IPv6StringToNum(s)

IPv6NumToStringの逆関数。IPv6アドレスの形式が無効な場合は、nullバイトの文字列を返します。  
HEXは大文字または小文字です。

## IPv4ToIntPv6(x)

を取る UInt32 番号 これをIPv4アドレスとして解釈します。 ビッグエンディアン Aを返します FixedString(16) IPv6ア  
ドレスをバイナリ形式で含む値。例:

```
SELECT IPv6NumToString(IPv4ToIntPv6(IPv4StringToNum('192.168.0.1'))) AS addr
```

```
addr  
::ffff:192.168.0.1 |
```

## cutIPv6(x,bytesToCutForIPv6,bytesToCutForIPv4)

バイナリ形式のIPv6アドレスを含むFixedString(16)値を受け入れます。テキスト形式で削除された指定されたバイト数のアドレスを含む文字列を返します。例えば:

```
WITH  
    IPv6StringToNum('2001:0DB8:AC10:FE01:FEED:BABE:CAFE:F00D') AS ipv6,  
    IPv4ToIntPv6(IPv4StringToNum('192.168.0.1')) AS ipv4  
SELECT  
    cutIPv6(ipv6, 2, 0),  
    cutIPv6(ipv4, 0, 2)
```

```
cutIPv6(ipv6, 2, 0)-----cutIPv6(ipv4, 0, 2)---  
2001:db8:ac10:fe01:feed:babe:0 | ::ffff:192.168.0.0 |
```

## IPv4CIDRToRange(ipv4,Cidr),

IPv4およびuint8の値を受け入れます。 CIDR. サブネットの低い範囲と高い範囲を含むIPv4のタプルを返します。

```
SELECT IPv4CIDRToRange(toIPv4('192.168.5.2'), 16)
```

```
IPv4CIDRToRange(toIPv4('192.168.5.2'), 16)---  
('192.168.0.0','192.168.255.255')
```

## IPv6CIDRToRange(ipv6,Cidr),

IPv6およびCidrを含むUInt8値を受け入れます。サブネットの下位範囲と上位の範囲を含むIPv6のタプルを返します。

```
SELECT IPv6CIDRToRange(toIPv6('2001:0db8:0000:85a3:0000:0000:ac1f:8001'), 32);
```

```
IPv6CIDRToRange(toIPv6('2001:0db8:0000:85a3:0000:0000:ac1f:8001'), 32)---  
('2001:db8::','2001:db8:ffff:ffff:ffff:ffff:ffff')
```

## toIPv4(文字列)

エイリアス IPv4StringToNum() これは、IPv4アドレスの文字列形式をとり、の値を返します IPv4 返される値と等しいバイナリ型 IPv4StringToNum().

```
WITH  
    '171.225.130.45' as IPv4_string  
SELECT  
    toTypeName(IPv4StringToNum(IPv4_string)),  
    toTypeName(toIPv4(IPv4_string))
```

```
toTypeName(IPv4StringToNum(IPv4_string))---toTypeName(toIPv4(IPv4_string))---  
UInt32 | IPv4 |
```

```
WITH
  '171.225.130.45' as IPv4_string
SELECT
  hex(IPv4StringToNum(IPv4_string)),
  hex(toIPv4(IPv4_string))
```

```
hex(IPv4StringToNum(IPv4_string)) —> hex(toIPv4(IPv4_string)) —>
ABE1822D           ABE1822D
```

## toIPv6(文字列)

エイリアス `IPv6StringToNum()` これは、IPv6アドレスの文字列形式を取り、の値を返します **IPv6** 返される値と等しいバイナリ型 `IPv6StringToNum()`.

```
WITH
  '2001:438:ffff::407d:1bc1' as IPv6_string
SELECT
  toTypeName(IPv6StringToNum(IPv6_string)),
  toTypeName(toIPv6(IPv6_string))
```

```
toTypeName(IPv6StringToNum(IPv6_string)) —> toTypeName(toIPv6(IPv6_string)) —>
FixedString(16)           | IPv6
```

```
WITH
  '2001:438:ffff::407d:1bc1' as IPv6_string
SELECT
  hex(IPv6StringToNum(IPv6_string)),
  hex(toIPv6(IPv6_string))
```

```
hex(IPv6StringToNum(IPv6_string)) —> hex(toIPv6(IPv6_string)) —>
20010438FFFF000000000000407D1BC1 | 20010438FFFF000000000000407D1BC1 |
```

## JSONを操作するための関数

YandexでのMetrica、JSONで送信したユーザーとしてセッションパラメータ。このJSONを操作するための特別な関数がいくつかあります。（ほとんどの場合、Jsonはさらに前処理され、結果の値は処理された形式で別々の列に入れられます。）これらの関数はすべて、JSONが何であるかについての強い前提に基づいていますが、できるだけ少なくして仕事を終わらせようとします。

以下の仮定が行われます：

1. フィールド名(関数の引数)は定数でなければなりません。
2. フィールド名は何らかの形でjsonでエンコードされます。例えば: `visitParamHas('{"abc":"def"}', 'abc') = 1` でも `visitParamHas('"\u0061\u0062\u0063":"def"', 'abc') = 0`
3. フィールドは、任意の入れ子レベルで無差別に検索されます。複数の一致するフィールドがある場合は、最初の出現が使用されます。
4. JSONには文字列リテラルの外側に空白文字はありません。

## visitParamHas(パラムス、名前)

があるかどうかをチェックします。'name' 名前

## ビジットパラメクストラクチュイント(params,name)

名前のフィールドの値から UInt64 を解析します 'name'. これが文字列フィールドの場合、文字列の先頭から数値を解析しようとします。フィールドが存在しない場合、または存在するが数値が含まれていない場合は、0を返します。

## ビジットパラメクストラクチント(params,name)

Int64と同じです。

パラメーター)

Float64と同じです。

## ビジットパラメクストラクトブル(params,name)

真/偽の値を解析します。結果は UInt8 です。

## ビジットパラメクストラクトロー(params,name)

区切り文字を含むフィールドの値を返します。

例:

```
visitParamExtractRaw('{"abc":"\\n\\u0000"}', 'abc') = '\"\\n\\u0000\"'  
visitParamExtractRaw('{"abc":{"def":[1,2,3]}}', 'abc') = '{"def":[1,2,3]}'
```

パラメーターを指定します)

二重引用符で文字列を解析します。値はエスケープされていません。エスケープ解除に失敗した場合は、空の文字列を返します。

例:

```
visitParamExtractString('{"abc":"\\n\\u0000"}', 'abc') = '\\n\\0'  
visitParamExtractString('{"abc":"\\u263a"}', 'abc') = '@'  
visitParamExtractString('{"abc":"\\u263"}', 'abc') = ''  
visitParamExtractString('{"abc":"hello"}', 'abc') = ''
```

現在、この形式のコードポイントはサポートされていません \uXXXX\uYYYY それは基本的な多言語面からではありません (UTF-8 の代わりに CESU-8 に変換されます)。

以下の機能は、次のとおりです **simdjson** より複雑な JSON 解析要件用に設計されています。上記の仮定 2 はまだ適用されます。

## isValidJSON(json)

渡された文字列が有効な json であることを確認します。

例:

```
SELECT isValidJSON('{"a": "hello", "b": [-100, 200.0, 300]}') = 1  
SELECT isValidJSON('not a json') = 0
```

## JSONHas(json[, indices\_or\_keys]...)

値が JSON ドキュメントに存在する場合、1 返されます。

値が存在しない場合、0 返されます。

例:

```
SELECT JSONHas('{"a": "hello", "b": [-100, 200.0, 300]}', 'b') = 1
SELECT JSONHas('{"a": "hello", "b": [-100, 200.0, 300]}', 'b', 4) = 0
```

`indices_or_keys` ゼロ以上の引数のリストは、それぞれ文字列または整数のいずれかになります。

- `String`=キー
- 正の整数=最初からn番目のメンバー/キーにアクセスします。
- 負の整数=最後からn番目のメンバー/キーにアクセスします。

要素の最小インデックスは1です。したがって、要素0は存在しません。

整数を使用してJSON配列とJSONオブジェクトの両方にアクセスできます。

例ええば:

```
SELECT JSONExtractKey('{"a": "hello", "b": [-100, 200.0, 300]}', 1) = 'a'
SELECT JSONExtractKey('{"a": "hello", "b": [-100, 200.0, 300]}', 2) = 'b'
SELECT JSONExtractKey('{"a": "hello", "b": [-100, 200.0, 300]}', -1) = 'b'
SELECT JSONExtractKey('{"a": "hello", "b": [-100, 200.0, 300]}', -2) = 'a'
SELECT JSONExtractString('{"a": "hello", "b": [-100, 200.0, 300]}', 1) = 'hello'
```

## JSONLength(json[, indices\_or\_keys]...)

JSON配列またはJSONオブジェクトの長さを返します。

値が存在しない場合、または型が間違っている場合、`0`返されます。

例:

```
SELECT JSONLength('{"a": "hello", "b": [-100, 200.0, 300]}', 'b') = 3
SELECT JSONLength('{"a": "hello", "b": [-100, 200.0, 300]}') = 2
```

## JSONType(json[, indices\_or\_keys]...)

JSON値の型を返します。

値が存在しない場合、`Null`返されます。

例:

```
SELECT JSONType('{"a": "hello", "b": [-100, 200.0, 300]}') = 'Object'
SELECT JSONType('{"a": "hello", "b": [-100, 200.0, 300]}', 'a') = 'String'
SELECT JSONType('{"a": "hello", "b": [-100, 200.0, 300]}', 'b') = 'Array'
```

## JSONExtractUInt(json[, indices\_or\_keys]...)

## JSONExtractInt(json[, indices\_or\_keys]...)

## JSONExtractFloat(json[, indices\_or\_keys]...)

## JSONExtractBool(json[, indices\_or\_keys]...)

JSONを解析して値を抽出します。これらの機能と類似 `visitParam` 機能。

値が存在しない場合、または型が間違っている場合、0 返されます。

例:

```
SELECT JSONExtractInt('{"a": "hello", "b": [-100, 200.0, 300]}', 'b', 1) = -100
SELECT JSONExtractFloat('{"a": "hello", "b": [-100, 200.0, 300]}', 'b', 2) = 200.0
SELECT JSONExtractUInt('{"a": "hello", "b": [-100, 200.0, 300]}', 'b', -1) = 300
```

## JSONExtractString(json[, indices\_or\_keys]...)

JSONを解析して文字列を抽出します。この関数は次のようにになります `visitParamExtractString` 機能。

値が存在しないか、型が間違っている場合は、空の文字列が返されます。

値はエスケープされていません。エスケープ解除に失敗した場合は、空の文字列を返します。

例:

```
SELECT JSONExtractString('{"a": "hello", "b": [-100, 200.0, 300]}', 'a') = 'hello'
SELECT JSONExtractString('{"abc": "\n\u0000"}', 'abc') = '\n\0'
SELECT JSONExtractString('{"abc": "\u263a"}', 'abc') = '@'
SELECT JSONExtractString('{"abc": "\u263"}', 'abc') =
SELECT JSONExtractString('{"abc": "hello"}', 'abc') = ''
```

## JSONExtract(json[, indices\_or\_keys...], Return\_type)

JSONを解析し、指定されたClickHouseデータ型の値を抽出します。

これは以前のもの的一般化です `JSONExtract<type>` 機能。

つまり

`JSONExtract(..., 'String')` とまったく同じを返します `JSONExtractString()`,  
`JSONExtract(..., 'Float64')` とまったく同じを返します `JSONExtractFloat()`.

例:

```
SELECT JSONExtract('{"a": "hello", "b": [-100, 200.0, 300]}', 'Tuple(String, Array(Float64))') = ('hello',[-100,200,300])
SELECT JSONExtract('{"a": "hello", "b": [-100, 200.0, 300]}', 'Tuple(b Array(Float64), a String)') =
([-100,200,300],'hello')
SELECT JSONExtract('{"a": "hello", "b": [-100, 200.0, 300]}', 'b', 'Array(Nullable(Int8))') = [-100, NULL, NULL]
SELECT JSONExtract('{"a": "hello", "b": [-100, 200.0, 300]}', 'b', 4, 'Nullable(Int64)') = NULL
SELECT JSONExtract('{"passed": true}', 'passed', 'UInt8') = 1
SELECT JSONExtract('{"day": "Thursday"}', 'day', 'Enum8(\\"Sunday\\") = 0, \\"Monday\\") = 1, \\"Tuesday\\") = 2,
\\"Wednesday\\") = 3, \\"Thursday\\") = 4, \\"Friday\\") = 5, \\"Saturday\\") = 6)') = 'Thursday'
SELECT JSONExtract('{"day": 5}', 'day', 'Enum8(\\"Sunday\\") = 0, \\"Monday\\") = 1, \\"Tuesday\\") = 2, \\"Wednesday\\") = 3,
\\"Thursday\\") = 4, \\"Friday\\") = 5, \\"Saturday\\") = 6)') = 'Friday'
```

## JSONExtractKeysAndValues(json[, indices\_or\_keys...], Value\_type)

値が指定されたClickHouseデータ型であるJSONからキーと値のペアを解析します。

例:

```
SELECT JSONExtractKeysAndValues('{"x": {"a": 5, "b": 7, "c": 11}}', 'x', 'Int8') = [('a',5),('b',7),('c',11)]
```

## JSONExtractRaw(json[, indices\_or\_keys]...)

JSONの一部を解析されていない文字列として返します。

パートが存在しないか、型が間違っている場合は、空の文字列が返されます。

例:

```
SELECT JSONExtractRaw('{"a": "hello", "b": [-100, 200.0, 300]}', 'b') = '[-100, 200.0, 300]'
```

## JSONExtractArrayRaw(json[, indices\_or\_keys...])

JSON配列の要素を持つ配列を返します。

パートが存在しないか配列でない場合は、空の配列が返されます。

例:

```
SELECT JSONExtractArrayRaw('{"a": "hello", "b": [-100, 200.0, "hello"]}', 'b') = ['-100', '200.0', "hello"]'
```

## JSONExtractKeysAndValuesRaw

JSONオブジェクトから生データを抽出します。

構文

```
JSONExtractKeysAndValuesRaw(json[, p, a, t, h])
```

パラメータ

- **json** — 文字列 有効なJSONで。
- **p, a, t, h** — Comma-separated indices or keys that specify the path to the inner field in a nested JSON object. Each argument can be either a 文字列 キーまたはキーでフィールドを取得するには 整数 N番目のフィールドを取得するには（1からインデックス付けされ、負の整数は最後から数えます）。設定されていない場合、JSON全体がトップレベルのオブジェクトとして解析されます。任意パラメータ。

戻り値

- との配列 ('key', 'value') タプル 両方のタプルメンバーは文字列です。
- 要求されたオブジェクトが存在しない場合、または入力JSONが無効な場合は空の配列。

タイプ: 配列(タプル(文字列, 文字列)).

例

クエリ:

```
SELECT JSONExtractKeysAndValuesRaw('{"a": [-100, 200.0], "b": {"c": {"d": "hello", "f": "world"}}}')
```

結果:

```
└─JSONExtractKeysAndValuesRaw('{"a": [-100, 200.0], "b": {"c": {"d": "hello", "f": "world"}}}') ─┘  
[('a', '[-100,200]'), ('b', {'c': {'d': 'hello', 'f': 'world'}})]
```

クエリ:

```
SELECT JSONExtractKeysAndValuesRaw('{"a": [-100, 200.0], "b": {"c": {"d": "hello", "f": "world"}}}', 'b')
```

結果:

```
JSONExtractKeysAndValuesRaw('{"a": [-100, 200.0], "b": {"c": {"d": "hello", "f": "world"} }}', 'b')  
[("c", {"d": "hello", "f": "world"})]
```

クエリ:

```
SELECT JSONExtractKeysAndValuesRaw('{"a": [-100, 200.0], "b": {"c": {"d": "hello", "f": "world"} }}', -1, 'c')
```

結果:

```
JSONExtractKeysAndValuesRaw('{"a": [-100, 200.0], "b": {"c": {"d": "hello", "f": "world"} }}', -1, 'c')  
[("d", "hello"), ("f", "world")]
```

## 高次関数

### -> 演算子,lambda(params,expr) 関数

Allows describing a lambda function for passing to a higher-order function. The left side of the arrow has a formal parameter, which is any ID, or multiple formal parameters – any IDs in a tuple. The right side of the arrow has an expression that can use these formal parameters, as well as any table columns.

例: `x -> 2 * x, str -> str != Referer.`

高階関数は、関数の引数としてラムダ関数のみを受け入れることができます。

複数の引数を受け入れるlambda関数は、高次関数に渡すことができます。この場合、高次関数には、これらの引数が対応する同じ長さのいくつかの配列が渡されます。

以下のような一部の機能については [arrayCount](#) または [アレイサム](#)、最初の引数（ラムダ関数）は省略することができます。この場合、同一の写像が仮定される。

Lambda関数は、次の関数では省略できません:

- [arrayMap](#)
- [arrayFilter](#)
- [arrayFill](#)
- [arrayReverseFill](#)
- [arraySplit](#)
- [arrayReverseSplit](#)
- [arrayFirst](#)
- [arrayFirstIndex](#)

### arrayMap(func, arr1, ...)

の元のアプリケーションから取得した配列を返します。func の各要素に対する関数 arr 配列

例:

```
SELECT arrayMap(x -> (x + 2), [1, 2, 3]) AS res;
```

```
res  
[3,4,5] |
```

異なる配列から要素のタプルを作成する方法を次の例に示します:

```
SELECT arrayMap((x, y) -> (x, y), [1, 2, 3], [4, 5, 6]) AS res
```

```
res  
[(1,4),(2,5),(3,6)] |
```

最初の引数（ラムダ関数）は省略できないことに注意してください。`arrayMap` 機能。

## arrayFilter(func, arr1, ...)

要素のみを含む配列を返します。`arr1` そのために `func` 0以外のものを返します。

例:

```
SELECT arrayFilter(x -> x LIKE '%World%', ['Hello', 'abc World']) AS res
```

```
res  
['abc World'] |
```

```
SELECT  
    arrayFilter(  
        (i, x) -> x LIKE '%World%',  
        arrayEnumerate(arr),  
        ['Hello', 'abc World'] AS arr)  
    AS res
```

```
res  
[2] |
```

最初の引数（ラムダ関数）は省略できないことに注意してください。`arrayFilter` 機能。

## arrayFill(func, arr1, ...)

スキヤンスルー `arr1` 最初の要素から最後の要素へと置き換えます `arr1[i]` によって `arr1[i - 1]` もし `func` 0を返します。の最初の要素 `arr1` 交換されません。

例:

```
SELECT arrayFill(x -> notisNull(x), [1, null, 3, 11, 12, null, null, 5, 6, 14, null, null]) AS res
```

```
res  
[1,1,3,11,12,12,12,5,6,14,14,14] |
```

最初の引数（ラムダ関数）は省略できないことに注意してください。arrayFill 機能。

## arrayReverseFill(func, arr1, ...)

スキヤンスルー arr1 最後の要素から最初の要素へと置き換えます arr1[i] によって arr1[i + 1] もし func 0を返します。の最後の要素 arr1 交換されません。

例:

```
SELECT arrayReverseFill(x -> notisNull(x), [1, null, 3, 11, 12, null, null, 5, 6, 14, null, null]) AS res
```

```
res  
[1,3,3,11,12,5,5,6,14,NULL,NULL] |
```

最初の引数（ラムダ関数）は省略できないことに注意してください。arrayReverseFill 機能。

## arraySplit(func, arr1, ...)

分割 arr1 複数の配列に。とき func 0以外のものを返すと、配列は要素の左側で分割されます。配列は最初の要素の前に分割されません。

例:

```
SELECT arraySplit((x, y) -> y, [1, 2, 3, 4, 5], [1, 0, 0, 1, 0]) AS res
```

```
res  
[[1,2,3],[4,5]] |
```

最初の引数（ラムダ関数）は省略できないことに注意してください。arraySplit 機能。

## arrayReverseSplit(func, arr1, ...)

分割 arr1 複数の配列に。とき func 0以外のものを返すと、配列は要素の右側に分割されます。配列は最後の要素の後に分割されません。

例:

```
SELECT arrayReverseSplit((x, y) -> y, [1, 2, 3, 4, 5], [1, 0, 0, 1, 0]) AS res
```

```
res  
[[1],[2,3,4],[5]] |
```

最初の引数（ラムダ関数）は省略できないことに注意してください。arraySplit 機能。

## arrayCount([func,] arr1, ...)

Funcが0以外のものを返すarr配列内の要素の数を返します。もし ‘func’ 指定されていない場合は、配列内のゼロ以外の要素の数を返します。

## arrayExists([func,] arr1, ...)

少なくとも一つの要素がある場合は1を返します ‘arr’ そのために ‘func’ 0以外のものを返します。それ以外の場合は0を返します。

## arrayAll([func,] arr1, ...)

場合は1を返します ‘func’ 内のすべての要素に対して0以外のものを返します ‘arr’. それ以外の場合は0を返します。

## arraySum([func,] arr1, ...)

の合計を返します。‘func’ 値。関数を省略すると、配列要素の合計が返されます。

## arrayFirst(func, arr1, ...)

の最初の要素を返します。‘arr1’ これに対する配列 ‘func’ 0以外のものを返します。

最初の引数 (ラムダ関数) は省略できないことに注意してください。arrayFirst 機能。

## arrayFirstIndex(func, arr1, ...)

の最初の要素のインデックスを返します。‘arr1’ これに対する配列 ‘func’ 0以外のものを返します。

最初の引数 (ラムダ関数) は省略できないことに注意してください。arrayFirstIndex 機能。

## arrayCumSum([func,] arr1, ...)

ソース配列内の要素の部分和(実行中の合計)の配列を返します。もし func 関数が指定されると、配列要素の値は合計する前にこの関数によって変換されます。

例:

```
SELECT arrayCumSum([1, 1, 1, 1]) AS res
```

```
res  
[1, 2, 3, 4] |
```

## arrayCumSumNonNegative(arr)

同じ arrayCumSum, ソース配列内の要素の部分和の配列を返します(実行中の合計)。異なる arrayCumSum 返された値にゼロ未満の値が含まれている場合、値はゼロに置き換えられ、その後の計算はゼロパラメータで実行されます。例えば:

```
SELECT arrayCumSumNonNegative([1, 1, -4, 1]) AS res
```

```
res  
[1,2,0,1] |
```

## arraySort([func,] arr1, ...)

の要素を並べ替えた結果として配列を返します arr1 昇順で。もし func 関数が指定され、ソート順序は関数の結果によって決定されます func 配列 (配列) の要素に適用されます)

その シュワルツ変換 分類の効率を改善するのに使用されています。

例:

```
SELECT arraySort((x, y) -> y, ['hello', 'world'], [2, 1]);
```

```
res
['world', 'hello'] |
```

詳細については、`arraySort` 方法は、参照してください [配列を操作するための関数](#) セクション

## arrayReverseSort([func,] arr1, ...)

の要素を並べ替えた結果として配列を返します `arr1` 降順で。もし `func` 関数が指定され、ソート順序は関数の結果によって決定されます `func` 配列（配列）の要素に適用されます。

例:

```
SELECT arrayReverseSort((x, y) -> y, ['hello', 'world'], [2, 1]) as res;
```

```
res
['hello','world'] |
```

詳細については、`arrayReverseSort` 方法は、参照してください [配列を操作するための関数](#) セクション

## 外部辞書を操作するための関数

情報の接続や設定の外部辞書参照 [外部辞書](#).

### dictGet

外部ディクショナリから値を取得します。

```
dictGet('dict_name', 'attr_name', id_expr)
dictGetOrDefault('dict_name', 'attr_name', id_expr, default_value_expr)
```

パラメータ

- `dict_name` — Name of the dictionary. 文字列リテラル.
- `attr_name` — Name of the column of the dictionary. 文字列リテラル.
- `id_expr` — Key value. 式 `a` を返す `UInt64` または タプル-辞書構成に応じて値を入力します。
- `default_value_expr` — Value returned if the dictionary doesn't contain a row with the `id_expr` キー 式 データ型の値を返します。 `attr_name` 属性。

戻り値

- ClickHouseが属性を正常に解析した場合、属性のデータ型, 関数は、に対応する辞書属性の値を返します `id_expr`.
- キーがない場合、対応する `id_expr`、辞書では、:

```
- `dictGet` returns the content of the `<null_value>` element specified for the attribute in the dictionary configuration.
- `dictGetOrDefault` returns the value passed as the `default_value_expr` parameter.
```

ClickHouseは、属性の値を解析できない場合、または値が属性データ型と一致しない場合に例外をスローします。

## 例

テキストファイルの作成 `ext-dict-text.csv` 以下を含む:

```
1,1  
2,2
```

最初の列は次のとおりです `id` 二つ目の列は `c1`.

外部辞書の構成:

```
<clickhouse>  
  <dictionary>  
    <name>ext-dict-test</name>  
    <source>  
      <file>  
        <path>/path-to/ext-dict-test.csv</path>  
        <format>CSV</format>  
      </file>  
    </source>  
    <layout>  
      <flat />  
    </layout>  
    <structure>  
      <id>  
        <name>id</name>  
      </id>  
      <attribute>  
        <name>c1</name>  
        <type>UInt32</type>  
        <null_value></null_value>  
      </attribute>  
    </structure>  
    <lifetime>0</lifetime>  
  </dictionary>  
</clickhouse>
```

クエリの実行:

```
SELECT  
  dictGetOrDefault('ext-dict-test', 'c1', number + 1, toUInt32(number * 10)) AS val,  
  toTypeName(val) AS type  
FROM system.numbers  
LIMIT 3
```

val	type
1	UInt32
2	UInt32
20	UInt32

も参照。

- [外部辞書](#)

## ディクタス

キーが辞書に存在するかどうかを確認します。

```
dictHas('dict_name', id_expr)
```

## パラメータ

- `dict_name` — Name of the dictionary. 文字列リテラル.
- `id_expr` — Key value. 式 `a`を返す UInt64-タイプ値。

## 戻り値

- キーがない場合は0。
- 1、キーがある場合。

タイプ: UInt8.

## dictGetHierarchy

キーのすべての親を含む配列を作成します。 階層辞書.

## 構文

```
dictGetHierarchy('dict_name', key)
```

## パラメータ

- `dict_name` — Name of the dictionary. 文字列リテラル.
- `key` — Key value. 式 `a`を返す UInt64-タイプ値。

## 戻り値

- 鍵の親。

タイプ: 配列(UInt64).

## ジクチシン

辞書の階層チェーン全体を通して、キーの祖先をチェックします。

```
dictIsIn('dict_name', child_id_expr, ancestor_id_expr)
```

## パラメータ

- `dict_name` — Name of the dictionary. 文字列リテラル.
- `child_id_expr` — Key to be checked. 式 `a`を返す UInt64-タイプ値。
- `ancestor_id_expr` — Alleged ancestor of the `child_id_expr` キー 式 `a`を返す UInt64-タイプ値。

## 戻り値

- 0の場合 `child_id_expr` の子ではありません `ancestor_id_expr`.
- 1の場合 `child_id_expr` の子である `ancestor_id_expr` または `child_id_expr` は `ancestor_id_expr`.

タイプ: UInt8.

## その他の機能

ClickHouseは、辞書構成に関係なく、辞書属性の値を特定のデータ型に変換する特殊な関数をサポートしています。

関数:

- `dictGetInt8`, `dictGetInt16`, `dictGetInt32`, `dictGetInt64`
- `dictGetUInt8`, `dictGetUInt16`, `dictGetUInt32`, `dictGetUInt64`
- `dictGetFloat32`, `dictGetFloat64`
- `dictGetDate`
- `dictGetDateTime`
- `dictGetUUID`
- `dictGetString`

これらの関数はすべて、`OrDefault` 修正 例えは、`dictGetDateOrDefault`.

構文:

```
dictGet[Type](#sql-reference-functions--dict_name---attr_name---id_expr)
dictGet[Type]OrDefault('dict_name', 'attr_name', id_expr, default_value_expr)
```

パラメータ

- `dict_name` — Name of the dictionary. 文字列リテラル.
- `attr_name` — Name of the column of the dictionary. 文字列リテラル.
- `id_expr` — Key value. 式 `a`を返す `UInt64`-タイプ値。
- `default_value_expr` — Value which is returned if the dictionary doesn't contain a row with the `id_expr` キー 式 データ型に設定された値を返す `attr_name` 属性。

戻り値

- ClickHouseが属性を正常に解析した場合、属性のデータ型, 関数は、に対応する辞書属性の値を返します `id_expr`.
- 要求がない場合 `id_expr` 辞書では:

```
- `dictGet[Type]` returns the content of the `<null_value>` element specified for the attribute in the dictionary configuration.
- `dictGet[Type]OrDefault` returns the value passed as the `default_value_expr` parameter.
```

ClickHouseは、属性の値を解析できない場合、または値が属性データ型と一致しない場合に例外をスローします。

## Yandexで作業するための機能。メトリカ辞書

以下の機能を機能させるには、サーバー設定ですべてのYandexを取得するためのパスとアドレスを指定する必要があります。メトリカ辞書。辞書は、これらの関数の最初の呼び出し時に読み込まれます。参照リストをロードできない場合は、例外がスローされます。

そのための情報を参照リストの項をご参照ください “Dictionaries”.

## 複数のジオベース

ClickHouseは、特定の地域が属する国のさまざまな視点をサポートするために、複数の代替ジオベース（地域階層）を同時に使用することをサポートしています。

その ‘clickhouse-server’ configは、地域階層を持つファイルを指定します:  
`::<path_to_regions_hierarchy_file>/opt/geo/regions_hierarchy.txt</path_to_regions_hierarchy_file>`

このファイル以外にも、名前に\_記号と接尾辞が付加された近くのファイルも検索します（ファイル拡張子の前に）。たとえば、次のファイルも検索します `/opt/geo/regions_hierarchy_ua.txt`、存在する場合。

`ua` 辞書キーと呼ばれます。接尾辞のない辞書の場合、キーは空の文字列です。

すべての辞書は実行時に再ロードされます(`builtin_dictionaries_reload_interval`設定パラメータで定義されているように、一定の秒数ごとに、またはデフォルトでは時間に一度)。ただし、使用可能な辞書のリストは、サーバーの起動時に一度に定義されます。

All functions for working with regions have an optional argument at the end – the dictionary key. It is referred to as the geobase.

例:

```
regionToCountry(RegionID) - Uses the default dictionary: /opt/geo/regions_hierarchy.txt  
regionToCountry(RegionID, "") - Uses the default dictionary: /opt/geo/regions_hierarchy.txt  
regionToCountry(RegionID, 'ua') - Uses the dictionary for the 'ua' key: /opt/geo/regions_hierarchy_ua.txt
```

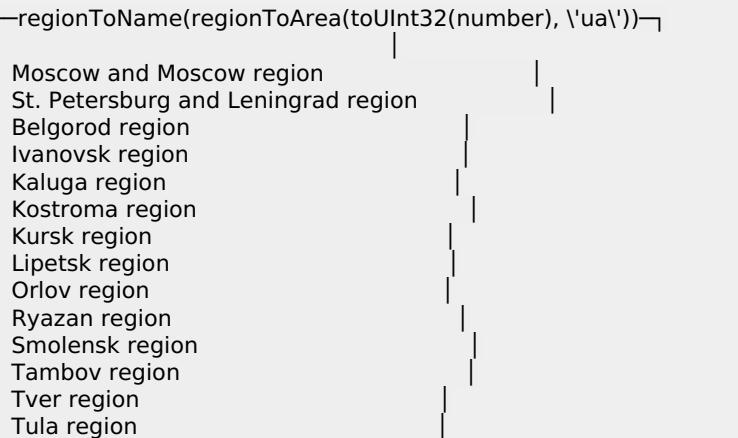
## リージョントシティ (id[,geobase])

Accepts a UInt32 number – the region ID from the Yandex geobase. If this region is a city or part of a city, it returns the region ID for the appropriate city. Otherwise, returns 0.

## レギオントアレア (id[,geobase])

領域を領域に変換します(ジオベースのタイプ5)。他のすべての方法では、この関数は次のように同じです 'regionToCity'.

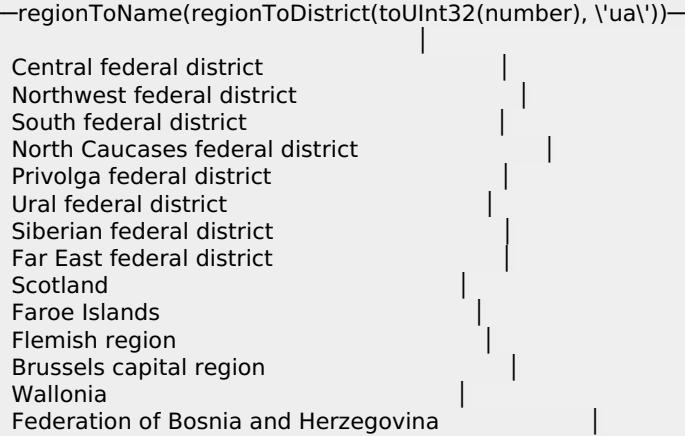
```
SELECT DISTINCT regionToName(regionToArea(toUInt32(number), 'ua'))  
FROM system.numbers  
LIMIT 15
```



## レジオントディストリクト (id[,geobase])

地域を連邦区（ジオベースのタイプ4）に変換します。他のすべての方法では、この関数は次のように同じです 'regionToCity'.

```
SELECT DISTINCT regionToName(regionToDistrict(toUInt32(number), 'ua'))  
FROM system.numbers  
LIMIT 15
```



## regionToCountry(id[,geobase])

地域を国に変換します。他のすべての方法では、この関数は次のように同じです ‘regionToCity’.

例: `regionToCountry(toUInt32(213)) = 225` モスクワ (213) をロシア (225) に変換します。

## レジオントコンテンツ(id[,geobase])

地域を大陸に変換します。他のすべての方法では、この関数は次のように同じです ‘regionToCity’.

例: `regionToContinent(toUInt32(213)) = 10001` モスクワ(213)をユーラシア(10001)に変換します。

## regionToTopContinent(#regiontotopcontinent)

リージョンの階層内で最も高い大陸を検索します。

### 構文

```
regionToTopContinent(id[, geobase]);
```

### パラメータ

- `id` — Region ID from the Yandex geobase. **UInt32**.
- `geobase` — Dictionary key. See [複数のジオベース](#), [文字列](#). 任意。

### 戻り値

- トップレベルの大洲の識別子（後者は地域の階層を登るとき）。
- ない場合は0。

タイプ: **UInt32**.

## リージョントポピュレーション(id[,geobase])

地域の人口を取得します。

母集団は、ジオベースを持つファイルに記録することができます。セクションを参照 “External dictionaries”.

人口が地域に記録されていない場合は、0を返します。

Yandexジオベースでは、子リージョンに対して母集団が記録されますが、親リージョンに対しては記録されません。

## レギオニン(lhs,rhs[,geobase])

Aかどうかチェック ‘lhs’ 地域はaに属します ‘rhs’ 地域。 UInt8が属している場合は1、属していない場合は0を返します。

The relationship is reflexive – any region also belongs to itself.

## リージョンハイアーキテクチャ(id[,geobase])

Accepts a UInt32 number - the region ID from the Yandex geobase. Returns an array of region IDs consisting of the passed region and all parents along the chain.

例: `regionHierarchy(toUInt32(213)) = [213,1,3,225,10001,10000]`.

## レジオントナム(id[,lang])

Accepts a UInt32 number - the region ID from the Yandex geobase. A string with the name of the language can be passed as a second argument. Supported languages are: ru, en, ua, uk, by, kz, tr. If the second argument is omitted, the language 'ru' is used. If the language is not supported, an exception is thrown. Returns a string - the name of the region in the corresponding language. If the region with the specified ID doesn't exist, an empty string is returned.

`ua` と `uk` もうクです。

## IN演算子を実装するための関数

で、ノチン、グロバリン、グロバルノチン

セクションを参照 [演算子で](#).

## tuple(x, y, ...), operator (x, y, ...)

複数の列をグループ化できる関数。

For columns with the types T1, T2, ..., it returns a Tuple(T1, T2, ...) type tuple containing these columns. There is no cost to execute the function.

タプルは、通常、in演算子の引数の中間値として、またはラムダ関数の仮パラメータのリストを作成するために使用されます。 タプルをテーブルに書き込むことはできません。

## tupleElement(タプル,n), 演算子x.N

タプルから列を取得できる関数。

'N' 1から始まる列インデックスです。 Nは定数でなければなりません。 'N' 定数でなければなりません。 'N' 組のサイズより大きくない厳密なpositive整数でなければなりません。

関数を実行するコストはありません。

## arrayJoin関数

これは非常に珍しい機能です。

通常の関数は行のセットを変更するのではなく、各行 (map) の値を変更するだけです。

集計関数は、行のセットを圧縮します (foldまたはreduce)。

その 'arrayJoin' 関数は、各行を取り、行のセットを生成します (展開)。

この関数は、配列を引数として受け取り、配列内の要素数に対してソース行を複数の行に伝播します。 この関数が適用される列の値を除いて、列内のすべての値は単純にコピーされます。

クエリでは、複数の `arrayJoin` 機能。 この場合、変換は複数回実行されます。

SELECTクエリの配列結合構文に注意してください。

例:

```
SELECT arrayJoin([1, 2, 3] AS src) AS dst, 'Hello', src
```

dst	'Hello'	src
1	Hello	[1,2,3]
2	Hello	[1,2,3]
3	Hello	[1,2,3]

## Functions for Working with Geographical Coordinates

### greatCircleDistance

Calculates the distance between two points on the Earth's surface using [the great-circle formula](#).

```
greatCircleDistance(lon1Deg, lat1Deg, lon2Deg, lat2Deg)
```

#### Input parameters

- `lon1Deg` — Longitude of the first point in degrees. Range: [-180°, 180°].
- `lat1Deg` — Latitude of the first point in degrees. Range: [-90°, 90°].
- `lon2Deg` — Longitude of the second point in degrees. Range: [-180°, 180°].
- `lat2Deg` — Latitude of the second point in degrees. Range: [-90°, 90°].

Positive values correspond to North latitude and East longitude, and negative values correspond to South latitude and West longitude.

#### Returned value

The distance between two points on the Earth's surface, in meters.

Generates an exception when the input parameter values fall outside of the range.

#### Example

```
SELECT greatCircleDistance(55.755831, 37.617673, -55.755831, -37.617673)
```

```
greatCircleDistance(55.755831, 37.617673, -55.755831, -37.617673) —
14132374.194975413 |
```

### geoDistance

Similar to `greatCircleDistance` but calculates the distance on WGS-84 ellipsoid instead of sphere. This is more precise approximation of the Earth Geoid.

The performance is the same as for `greatCircleDistance` (no performance drawback). It is recommended to use `geoDistance` to calculate the distances on Earth.

Technical note: for close enough points we calculate the distance using planar approximation with the metric on the tangent plane at the midpoint of the coordinates.

### greatCircleAngle

Calculates the central angle between two points on the Earth's surface using [the great-circle formula](#).

```
greatCircleAngle(lon1Deg, lat1Deg, lon2Deg, lat2Deg)
```

## Input parameters

- `lon1Deg` — Longitude of the first point in degrees.
- `lat1Deg` — Latitude of the first point in degrees.
- `lon2Deg` — Longitude of the second point in degrees.
- `lat2Deg` — Latitude of the second point in degrees.

## Returned value

The central angle between two points in degrees.

## Example

```
SELECT greatCircleAngle(0, 0, 45, 0) AS arc
```

```
arc  
45 |
```

## pointInEllipses

Checks whether the point belongs to at least one of the ellipses.

Coordinates are geometric in the Cartesian coordinate system.

```
pointInEllipses(x, y, x0, y0, a0, b0, ..., xn, yn, an, bn)
```

## Input parameters

- `x, y` — Coordinates of a point on the plane.
- `xi, yi` — Coordinates of the center of the *i*-th ellipsis.
- `ai, bi` — Axes of the *i*-th ellipsis in units of x, y coordinates.

The input parameters must be `2+4·n`, where `n` is the number of ellipses.

## Returned values

`1` if the point is inside at least one of the ellipses; `0` if it is not.

## Example

```
SELECT pointInEllipses(10., 10., 10., 9.1, 1., 0.9999)
```

```
pointInEllipses(10., 10., 10., 9.1, 1., 0.9999)  
1 |
```

## pointInPolygon

Checks whether the point belongs to the polygon on the plane.

```
pointInPolygon((x, y), [(a, b), (c, d) ...], ...)
```

## Input values

- `(x, y)` — Coordinates of a point on the plane. Data type — **Tuple** — A tuple of two numbers.
- `[(a, b), (c, d) ...]` — Polygon vertices. Data type — **Array**. Each vertex is represented by a pair of coordinates `(a, b)`. Vertices should be specified in a clockwise or counterclockwise order. The minimum number of vertices is 3. The polygon must be constant.
- The function also supports polygons with holes (cut out sections). In this case, add polygons that define the cut out sections using additional arguments of the function. The function does not support non-simply-connected polygons.

## Returned values

`1` if the point is inside the polygon, `0` if it is not.

If the point is on the polygon boundary, the function may return either `0` or `1`.

## Example

```
SELECT pointInPolygon((3., 3.), [(6, 0), (8, 4), (5, 8), (0, 2)]) AS res
```

```
res  
1 |
```

# Functions for Working with Geohash

**Geohash** is the geocode system, which subdivides Earth's surface into buckets of grid shape and encodes each cell into a short string of letters and digits. It is a hierarchical data structure, so the longer is the geohash string, the more precise is the geographic location.

If you need to manually convert geographic coordinates to geohash strings, you can use [geohash.org](http://geohash.org).

## geohashEncode

Encodes latitude and longitude as a **geohash**-string.

```
geohashEncode(longitude, latitude, [precision])
```

## Input values

- `longitude` - longitude part of the coordinate you want to encode. Floating in range `[-180°, 180°]`
- `latitude` - latitude part of the coordinate you want to encode. Floating in range `[-90°, 90°]`
- `precision` - Optional, length of the resulting encoded string, defaults to `12`. Integer in range `[1, 12]`. Any value less than `1` or greater than `12` is silently converted to `12`.

## Returned values

- alphanumeric **String** of encoded coordinate (modified version of the base32-encoding alphabet is used).

## Example

```
SELECT geohashEncode(-5.60302734375, 42.593994140625, 0) AS res;
```

```
res  
ezs42d000000 |
```

## geohashDecode

Decodes any **geohash**-encoded string into longitude and latitude.

### Input values

- encoded string - geohash-encoded string.

### Returned values

- (longitude, latitude) - 2-tuple of **Float64** values of longitude and latitude.

### Example

```
SELECT geohashDecode('ezs42') AS res;
```

```
res  
(-5.60302734375,42.60498046875) |
```

## geohashesInBox

Returns an array of **geohash**-encoded strings of given precision that fall inside and intersect boundaries of given box, basically a 2D grid flattened into array.

### Syntax

```
geohashesInBox(longitude_min, latitude_min, longitude_max, latitude_max, precision)
```

### Arguments

- `longitude_min` — Minimum longitude. Range: [-180°, 180°]. Type: **Float**.
- `latitude_min` — Minimum latitude. Range: [-90°, 90°]. Type: **Float**.
- `longitude_max` — Maximum longitude. Range: [-180°, 180°]. Type: **Float**.
- `latitude_max` — Maximum latitude. Range: [-90°, 90°]. Type: **Float**.
- `precision` — Geohash precision. Range: [1, 12]. Type: **UInt8**.

### Note

All coordinate parameters must be of the same type: either **Float32** or **Float64**.

### Returned values

- Array of precision-long strings of geohash-boxes covering provided area, you should not rely on order of items.

- [] - Empty array if minimum latitude and longitude values aren't less than corresponding maximum values.

Type: [Array\(String\)](#).

## Note

Function throws an exception if resulting array is over 10'000'000 items long.

## Example

Query:

```
SELECT geohashesInBox(24.48, 40.56, 24.785, 40.81, 4) AS thasos;
```

Result:

```
thasos  
['sx1q','sx1r','sx32','sx1w','sx1x','sx38'] |
```

# Functions for Working with H3 Indexes

[H3](#) is a geographical indexing system where Earth's surface divided into a grid of even hexagonal cells. This system is hierarchical, i. e. each hexagon on the top level ("parent") can be splitted into seven even but smaller ones ("children"), and so on.

The level of the hierarchy is called resolution and can receive a value from 0 till 15, where 0 is the base level with the largest and coarsest cells.

A latitude and longitude pair can be transformed to a 64-bit H3 index, identifying a grid cell.

The H3 index is used primarily for bucketing locations and other geospatial manipulations.

The full description of the H3 system is available at [the Uber Engeneering site](#).

## h3IsValid

Verifies whether the number is a valid [H3](#) index.

### Syntax

```
h3IsValid(h3index)
```

### Parameter

- h3index — Hexagon index number. Type: [UInt64](#).

### Returned values

- 1 — The number is a valid H3 index.
- 0 — The number is not a valid H3 index.

Type: [UInt8](#).

## Example

Query:

```
SELECT h3IsValid(630814730351855103) AS h3IsValid;
```

Result:

```
h3IsValid  
1 |
```

## h3GetResolution

Defines the resolution of the given **H3** index.

### Syntax

```
h3GetResolution(h3index)
```

### Parameter

- `h3index` — Hexagon index number. Type: **UInt64**.

### Returned values

- Index resolution. Range: [0, 15].
- If the index is not valid, the function returns a random value. Use **h3IsValid** to verify the index.

Type: **UInt8**.

### Example

Query:

```
SELECT h3GetResolution(639821929606596015) AS resolution;
```

Result:

```
resolution  
14 |
```

## h3EdgeAngle

Calculates the average length of the **H3** hexagon edge in grades.

### Syntax

```
h3EdgeAngle(resolution)
```

### Parameter

- `resolution` — Index resolution. Type: **UInt8**. Range: [0, 15].

### Returned values

- The average length of the H3 hexagon edge in grades. Type: **Float64**.

## Example

Query:

```
SELECT h3EdgeAngle(10) AS edgeAngle;
```

Result:

```
h3EdgeAngle(10)  
0.0005927224846720883 |
```

## h3EdgeLengthM

Calculates the average length of the H3 hexagon edge in meters.

## Syntax

```
h3EdgeLengthM(resolution)
```

## Parameter

- **resolution** — Index resolution. Type: **UInt8**. Range: [0, 15].

## Returned values

- The average length of the H3 hexagon edge in meters. Type: **Float64**.

## Example

Query:

```
SELECT h3EdgeLengthM(15) AS edgeLengthM;
```

Result:

```
edgeLengthM  
0.509713273 |
```

## geoToH3

Returns H3 point index (lon, lat) with specified resolution.

## Syntax

```
geoToH3(lon, lat, resolution)
```

## Arguments

- **lon** — Longitude. Type: **Float64**.
- **lat** — Latitude. Type: **Float64**.
- **resolution** — Index resolution. Range: [0, 15]. Type: **UInt8**.

## Returned values

- Hexagon index number.
- 0 in case of error.

Type: [UInt64](#).

## Example

Query:

```
SELECT geoToH3(37.79506683, 55.71290588, 15) AS h3Index;
```

Result:

```
h3Index  
644325524701193974
```

## h3ToGeo

Returns the geographical coordinates of longitude and latitude corresponding to the provided [H3](#) index.

## Syntax

```
h3ToGeo(h3Index)
```

## Arguments

- `h3Index` — H3 Index. [UInt64](#).

## Returned values

- A tuple consisting of two values: `tuple(lon,lat)`. `lon` — Longitude. [Float64](#). `lat` — Latitude. [Float64](#).

## Example

Query:

```
SELECT h3ToGeo(644325524701193974) AS coordinates;
```

Result:

```
coordinates  
(37.79506616830252,55.71290243145668)
```

## h3ToGeoBoundary

Returns array of pairs `(lon, lat)`, which corresponds to the boundary of the provided H3 index.

## Syntax

```
h3ToGeoBoundary(h3Index)
```

## Arguments

- `h3Index` — H3 Index. Type: `UInt64`.

## Returned values

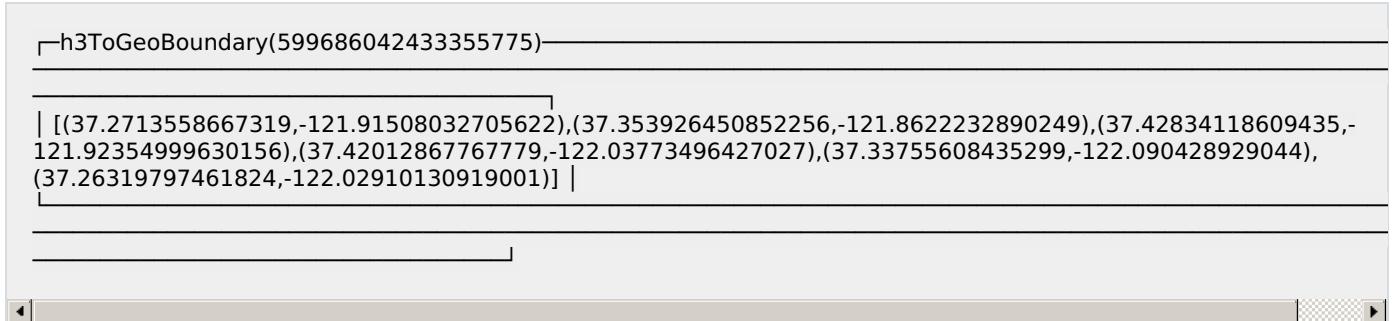
- Array of pairs '(lon, lat)'.  
Type: `Array(Float64, Float64)`.

## Example

Query:

```
SELECT h3ToGeoBoundary(644325524701193974) AS coordinates;
```

Result:



```
h3ToGeoBoundary(599686042433355775)
[ (37.2713558667319, -121.91508032705622), (37.353926450852256, -121.8622232890249), (37.42834118609435, -121.92354999630156), (37.42012867767779, -122.03773496427027), (37.33755608435299, -122.090428929044), (37.26319797461824, -122.02910130919001) ]
```

## h3kRing

Lists all the `H3` hexagons in the radius of `k` from the given hexagon in random order.

## Syntax

```
h3kRing(h3index, k)
```

## Arguments

- `h3index` — Hexagon index number. Type: `UInt64`.
- `k` — Raduis. Type: `integer`

## Returned values

- Array of H3 indexes.

Type: `Array(UInt64)`.

## Example

Query:

```
SELECT arrayJoin(h3kRing(644325529233966508, 1)) AS h3index;
```

Result:

```
-----h3index-----
644325529233966508
644325529233966497
644325529233966510
644325529233966504
644325529233966509
644325529233966355
644325529233966354
```

## h3GetBaseCell

Returns the base cell number of the **H3** index.

### Syntax

```
h3GetBaseCell(index)
```

### Parameter

- `index` — Hexagon index number. Type: **UInt64**.

### Returned value

- Hexagon base cell number.

Type: **UInt8**.

### Example

Query:

```
SELECT h3GetBaseCell(612916788725809151) AS basecell;
```

Result:

```
-----basecell-----
12 |
```

## h3HexAreaM2

Returns average hexagon area in square meters at the given resolution.

### Syntax

```
h3HexAreaM2(resolution)
```

### Parameter

- `resolution` — Index resolution. Range: [0, 15]. Type: **UInt8**.

### Returned value

- Area in square meters.

Type: **Float64**.

### Example

Query:

```
SELECT h3HexAreaM2(13) AS area;
```

Result:

```
area  
43.9 |
```

## h3IndexesAreNeighbors

Returns whether or not the provided **H3** indexes are neighbors.

### Syntax

```
h3IndexesAreNeighbors(index1, index2)
```

### Arguments

- `index1` — Hexagon index number. Type: **UInt64**.
- `index2` — Hexagon index number. Type: **UInt64**.

### Returned value

- `1` — Indexes are neighbours.
- `0` — Indexes are not neighbours.

Type: **UInt8**.

### Example

Query:

```
SELECT h3IndexesAreNeighbors(617420388351344639, 617420388352655359) AS n;
```

Result:

```
n  
1 |
```

## h3ToChildren

Returns an array of child indexes for the given **H3** index.

### Syntax

```
h3ToChildren(index, resolution)
```

### Arguments

- `index` — Hexagon index number. Type: **UInt64**.
- `resolution` — Index resolution. Range: [0, 15]. Type: **UInt8**.

## Returned values

- Array of the child H3-indexes.

Type: `Array(UInt64)`.

## Example

Query:

```
SELECT h3ToChildren(599405990164561919, 6) AS children;
```

Result:

```
{children: [603909588852408319, 603909588986626047, 603909589120843775, 603909589255061503, 603909589389279231, 603909589523496959, 603909589657714687]}
```

## h3ToParent

Returns the parent (coarser) index containing the given **H3** index.

## Syntax

```
h3ToParent(index, resolution)
```

## Arguments

- `index` — Hexagon index number. Type: `UInt64`.
- `resolution` — Index resolution. Range: [0, 15]. Type: `UInt8`.

## Returned value

- Parent H3 index.

Type: `UInt64`.

## Example

Query:

```
SELECT h3ToParent(599405990164561919, 3) AS parent;
```

Result:

```
{parent: 590398848891879423}
```

## h3ToString

Converts the `H3Index` representation of the index to the string representation.

```
h3ToString(index)
```

## Parameter

- index — Hexagon index number. Type: [UInt64](#).

## Returned value

- String representation of the H3 index.

Type: [String](#).

## Example

Query:

```
SELECT h3ToString(617420388352917503) AS h3_string;
```

Result:

```
h3_string  
89184926cdbffff |
```

## stringToH3

Converts the string representation to the [H3Index](#) (UInt64) representation.

## Syntax

```
stringToH3(index_str)
```

## Parameter

- index\_str — String representation of the H3 index. Type: [String](#).

## Returned value

- Hexagon index number. Returns 0 on error. Type: [UInt64](#).

## Example

Query:

```
SELECT stringToH3('89184926cc3ffff') AS index;
```

Result:

```
index  
617420388351344639 |
```

## h3GetResolution

Returns the resolution of the [H3](#) index.

## Syntax

```
h3GetResolution(index)
```

## Parameter

- `index` — Hexagon index number. Type: [UInt64](#).

## Returned value

- Index resolution. Range: [0, 15]. Type: [UInt8](#).

## Example

Query:

```
SELECT h3GetResolution(617420388352917503) AS res;
```

Result:

res
9

## h3IsResClassIII

Returns whether [H3](#) index has a resolution with Class III orientation.

## Syntax

```
h3IsResClassIII(index)
```

## Parameter

- `index` — Hexagon index number. Type: [UInt64](#).

## Returned value

- `1` — Index has a resolution with Class III orientation.
- `0` — Index doesn't have a resolution with Class III orientation.

Type: [UInt8](#).

## Example

Query:

```
SELECT h3IsResClassIII(617420388352917503) AS res;
```

Result:

res
1

## h3IsPentagon

Returns whether this [H3](#) index represents a pentagonal cell.

## Syntax

```
h3IsPentagon(index)
```

## Parameter

- `index` — Hexagon index number. Type: [UInt64](#).

## Returned value

- `1` — Index represents a pentagonal cell.
- `0` — Index doesn't represent a pentagonal cell.

Type: [UInt8](#).

## Example

Query:

```
SELECT h3IsPentagon(644721767722457330) AS pentagon;
```

Result:

```
pentagon
0 |
```

## h3GetFaces

Returns icosahedron faces intersected by a given [H3](#) index.

## Syntax

```
h3GetFaces(index)
```

## Parameter

- `index` — Hexagon index number. Type: [UInt64](#).

## Returned values

- Array containing icosahedron faces intersected by a given H3 index.

Type: [Array\(UInt64\)](#).

## Example

Query:

```
SELECT h3GetFaces(599686042433355775) AS faces;
```

Result:

```
faces
[7] |
```

# 地理座標を操作するための関数

## グレートサークル距離

を使用して、地球の表面上の二つの点の間の距離を計算します **大円の公式**。

```
greatCircleDistance(lon1Deg, lat1Deg, lon2Deg, lat2Deg)
```

### 入力パラメータ

- `lon1Deg` — Longitude of the first point in degrees. Range: [-180°, 180°].
- `lat1Deg` — Latitude of the first point in degrees. Range: [-90°, 90°].
- `lon2Deg` — Longitude of the second point in degrees. Range: [-180°, 180°].
- `lat2Deg` — Latitude of the second point in degrees. Range: [-90°, 90°].

正の値は北緯と東経に対応し、負の値は南緯と西経に対応します。

### 戻り値

メートル単位で、地球の表面上の二つの点の間の距離。

入力パラメーター値が範囲外にある場合に例外を生成します。

### 例

```
SELECT greatCircleDistance(55.755831, 37.617673, -55.755831, -37.617673)
```

```
greatCircleDistance(55.755831, 37.617673, -55.755831, -37.617673) —  
14132374.194975413 |
```

## ポイントネリップス

点が少なくとも一方の楕円に属しているかどうかをチェックします。

座標はデカルト座標系では幾何学的です。

```
pointInEllipses(x, y, x₀, y₀, a₀, b₀,...,xₙ, yₙ, aₙ, bₙ)
```

### 入力パラメータ

- `x, y` — Coordinates of a point on the plane.
- `xi, yi` — Coordinates of the center of the *i*-番目の省略記号。
- `ai, bi` — Axes of the *i*-x、y座標の単位で番目の省略記号。

入力パラメータ `2+4·n`、ここで `n` 楕円の数です。

### 戻り値

1 点が楕円の少なくとも一方の内側にある場合; 0 そうでない場合。

### 例

```
SELECT pointInEllipses(10., 10., 10., 9.1, 1., 0.9999)
```

```
pointInEllipses(10., 10., 10., 9.1, 1., 0.9999)  
1 |
```

## ポイントポリゴン

ポイントが平面上のポリゴンに属するかどうかを確認します。

```
pointInPolygon((x, y), [(a, b), (c, d) ...], ...)
```

### 入力値

- $(x, y)$  — Coordinates of a point on the plane. Data type — タプル — A tuple of two numbers.
- $[(a, b), (c, d) \dots]$  — Polygon vertices. Data type — 配列. 各頂点は、座標のペアで表されます  $(a, b)$ . 頂点は時計回りまたは反時計回りの順序で指定する必要があります。頂点の最小数は3です。多角形は一定でなければなりません。
- この機能にも対応し多角形穴あき(切り抜く部門). この場合、関数の追加引数を使用して切り取られたセクションを定義するポリゴンを追加します。この機能はサポートしない非単に接続ポリゴン.

### 戻り値

1 ポイントがポリゴンの内側にある場合, 0 そうでない場合。

ポイントがポリゴン境界上にある場合、関数は0または1のいずれかを返します。

### 例

```
SELECT pointInPolygon((3., 3.), [(6, 0), (8, 4), (5, 8), (0, 2)]) AS res
```

```
res  
1 |
```

## geohashEncode

緯度と経度をジオハッシュ文字列としてエンコードします。<http://geohash.org/>, <https://en.wikipedia.org/wiki/Geohash>）。

```
geohashEncode(longitude, latitude, [precision])
```

### 入力値

- 経度-エンコードする座標の経度部分。範囲の浮遊  $[-180^\circ, 180^\circ]$
- 緯度-エンコードする座標の緯度の部分。範囲の浮遊  $[-90^\circ, 90^\circ]$
- 精度-オプションで、結果のエンコードされた文字列の長さ。12. 範囲内の整数  $[1, 12]$ . より小さい値 1 またはより大きい 12 に変換されます。12.

### 戻り値

- 英数字 String 符号化座標 (base32エンコーディングアルファベットの修正バージョンが使用されます)。

例

```
SELECT geohashEncode(-5.60302734375, 42.593994140625, 0) AS res
```

```
res  
ezs42d000000 |
```

## geohashDecode

ジオハッシュでエンコードされた文字列を経度と緯度にデコードします。

入力値

- エンコード文字列-ジオハッシュエンコード文字列。

戻り値

- (経度,緯度)-2-タプルの **Float64** 経度と緯度の値。

例

```
SELECT geohashDecode('ezs42') AS res
```

```
res  
(-5.60302734375,42.60498046875) |
```

## geoToH3

ツヅツ。 **H3** ポイント指標 (**lon**, **lat**) 指定決断を使って。

**H3** 地球の表面が六角形のタイルに分割された地理的索引システムです。このシステムは階層的であり、すなわち最上位の各六角形は七つに分割することができます、より小さなものなどです。

このインデックスは、主にパケットの場所やその他の地理空間操作に使用されます。

構文

```
geoToH3(lon, lat, resolution)
```

パラメータ

- **lon** — Longitude. Type: **Float64**.
- **lat** — Latitude. Type: **Float64**.
- **resolution** — Index resolution. Range: [0, 15]. タイプ: **UInt8**.

戻り値

- 六角形のインデックス番号。

- エラーの場合は0。

タイプ: **UInt64**.

例

クエリ:

```
SELECT geoToH3(37.79506683, 55.71290588, 15) as h3Index
```

結果:

```
h3Index  
644325524701193974
```

## geohashesInBox

指定されたボックスの内側にあり、指定されたボックスの境界と交差する、指定された精度のジオハッシュエンコードされた文字列の配列を返します

入力値

- `longitude_min`-最小の経度、範囲内の浮動小数点値 [-180°, 180°]
- `latitude_min`-最小緯度、範囲内の浮動小数点値 [-90°, 90°]
- `longitude_max`-最大経度、範囲内の浮動小数点値 [-180°, 180°]
- `latitude_max`-最大緯度、範囲内の浮動小数点値 [-90°, 90°]
- 精度-ジオハッシュ精度, `UInt8` 範囲内 [1, 12]

すべての座標パラメータは同じタイプでなければなりません。 `Float32` または `Float64`.

戻り値

- 精度の配列-提供された領域をカバーするジオハッシュボックスの長い文字列、あなたは項目の順序に頼るべきではありません。
- []-空の配列の場合 分の値 緯度と 経度 対応するよりも小さくない 最大 値。

結果の配列が10'000'000項目以上の場合、関数は例外をスローすることに注意してください。

例

```
SELECT geohashesInBox(24.48, 40.56, 24.785, 40.81, 4) AS thasos
```

```
thasos  
['sx1q','sx1r','sx32','sx1w','sx1x','sx38'] |
```

## h3GetBaseCell

インデックスの基本セル番号を返します。

構文

```
h3GetBaseCell(index)
```

パラメータ

- `index` — Hexagon index number. Type: `UInt64`.

## 戻り値

- 六角形のベースのセル番号。 タイプ: **UInt8**.

## 例

クエリ:

```
SELECT h3GetBaseCell(612916788725809151) as basecell
```

結果:

```
basecell
12 |
```

## h3HexAreaM2

与えられた解像度での平方メートルの平均六角形面積。

## 構文

```
h3HexAreaM2(resolution)
```

## パラメータ

- `resolution` — Index resolution. Range: [0, 15]. タイプ: **UInt8**.

## 戻り値

- Area in m<sup>2</sup>. Type: **Float64**.

## 例

クエリ:

```
SELECT h3HexAreaM2(13) as area
```

結果:

```
area
43.9 |
```

## h3IndexesAreNeighbors

指定されたH3indexがneighborであるかどうかを返します。

## 構文

```
h3IndexesAreNeighbors(index1, index2)
```

## パラメータ

- `index1` — Hexagon index number. Type: **UInt64**.
- `index2` — Hexagon index number. Type: **UInt64**.

## 戻り値

- ツヅツ。 **1** インデックスが近傍の場合, **0** そうでなければ タイプ: **UInt8**.

## 例

クエリ:

```
SELECT h3IndexesAreNeighbors(617420388351344639, 617420388352655359) AS n
```

結果:

```
[n  
1]
```

## h3ToChildren

指定したインデックスの子インデックスを持つ配列を返します。

## 構文

```
h3ToChildren(index, resolution)
```

## パラメータ

- **index** — Hexagon index number. Type: **UInt64**.
- **resolution** — Index resolution. Range: [0, 15]. タイプ: **UInt8**.

## 戻り値

- 子h3インデックスを持つ配列。型の配列: **UInt64**.

## 例

クエリ:

```
SELECT h3ToChildren(599405990164561919, 6) AS children
```

結果:

```
children  
[603909588852408319, 603909588986626047, 603909589120843775, 603909589255061503, 603909589389279231,  
603909589523496959, 603909589657714687] |
```

## h3ToParent

指定された索引を含む親(より粗い)索引を返します。

## 構文

## `h3ToParent(index, resolution)`

パラメータ

- `index` — Hexagon index number. Type: `UInt64`.
- `resolution` — Index resolution. Range: [0, 15]. タイプ: `UInt8`.

戻り値

- 親H3インデックス。 タイプ: `UInt64`.

例

クエリ:

```
SELECT h3ToParent(599405990164561919, 3) as parent
```

結果:

```
parent  
590398848891879423
```

## `h3ToString(index)`

パラメータ

- `index` — Hexagon index number. Type: `UInt64`.

戻り値

- H3インデックスの文字列表現。 タイプ: `文字列`.

例

クエリ:

```
SELECT h3ToString(617420388352917503) as h3_string
```

結果:

```
h3_string  
89184926cdbfffff |
```

## `stringToH3`

文字列表現をH3Index(UInt64)表現に変換します。

## `stringToH3(index_str)`

## パラメータ

- `index_str` — String representation of the H3 index. Type: 文字列.

## 戻り値

- 六角形のインデックス番号。エラーの場合は0を返します。タイプ: UInt64.

## 例

クエリ:

```
SELECT stringToH3('89184926cc3ffff') as index
```

結果:

```
index  
617420388351344639
```

## h3GetResolution

インデックスの解像度を返します。

## 構文

```
h3GetResolution(index)
```

## パラメータ

- `index` — Hexagon index number. Type: UInt64.

## 戻り値

- インデックスの解決。範囲: [0, 15]. タイプ: UInt8.

## 例

クエリ:

```
SELECT h3GetResolution(617420388352917503) as res
```

結果:

```
res  
9 |
```

## Null許容集計を操作するための関数

### isNull

引数が NULL.

```
isNull(x)
```

## パラメータ

- `x` — A value with a non-compound data type.

## 戻り値

- `1` もし `x` は `NULL`.

- `0` もし `x` ではない `NULL`.

## 例

入力テーブル

x	y
1	NULL
2	3

クエリ

```
SELECT x FROM t_null WHEREisNull(y)
```

x
1

## isNotNull

引数が `NULL`.

```
isNotNull(x)
```

## パラメータ:

- `x` — A value with a non-compound data type.

## 戻り値

- `0` もし `x` は `NULL`.

- `1` もし `x` ではない `NULL`.

## 例

入力テーブル

x	y
1	NULL
2	3

クエリ

```
SELECT x FROM t_null WHERE isNotNull(y)
```

X  
2 |

## 合体

左から右にチェックするかどうか **NULL** 引数が渡され、最初の **non** を返します - **NULL** 引数。

`coalesce(x,...)`

パラメータ:

- 非複合型の任意の数のパラメーター。すべてのパラメータに対応していることが必要となるデータ型になります。

戻り値

- 最初の非-**NULL** 引数。
- NULL** すべての引数が **NULL**.

例

顧客に連絡する複数の方法を指定できる連絡先のリストを検討してください。

name	mail	phone	icq
client 1	NULL	123-45-67	123
client 2	NULL	NULL	NULL

その **mail** と **phone** フィールドは **String** 型ですが、 **icq** フィールドは **UInt32** したがって、変換する必要があります **String**.

連絡先リストから顧客に対して最初に使用可能な連絡先方法を取得する:

```
SELECT coalesce(mail, phone, CAST(icq,'Nullable(String)')) FROM aBook
```

name	coalesce(mail, phone, CAST(icq, 'Nullable(String)'))
client 1	123-45-67
client 2	NULL

## ifNull

**Main** 引数が次の場合、代替値を返します **NULL**.

`ifNull(x,alt)`

パラメータ:

- x** — The value to check for **NULL**.
- alt** — The value that the function returns if **x** は **NULL**.

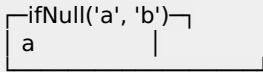
戻り値

- 値 **x**, if **x** ではない **NULL**.

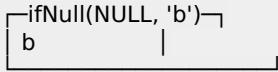
- 値 alt, if `x` は `NULL`.

例

```
SELECT ifNull('a', 'b')
```



```
SELECT ifNull(NULL, 'b')
```



## ヌリフ

ツヅケ。 `NULL` 引数が等しい場合。

```
nullIf(x, y)
```

パラメータ:

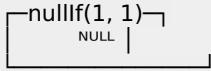
`x, y` — Values for comparison. They must be compatible types, or ClickHouse will generate an exception.

戻り値

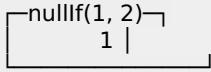
- `NULL`、引数が等しい場合。
- その `x` 引数が等しくない場合の値。

例

```
SELECT nullIf(1, 1)
```



```
SELECT nullIf(1, 2)
```



## assumeNotNull

結果は type の値になります **Null 可能** 非のため - `Nullable` 値が `NULL`.

```
assumeNotNull(x)
```

パラメータ:

- `x` — The original value.

戻り値

- 非からの元の値-`Nullable` そうでない場合は、タイプ `NULL`.

- 非のデフォルト値-`Nullable` 元の値が `NULL`.

例

を考える `t_null` テーブル。

```
SHOW CREATE TABLE t_null
```

```
statement
CREATE TABLE default.t_null ( x Int8, y Nullable(Int8) ) ENGINE = TinyLog |
```

x	y
1	NULL
2	3

を適用する `assumeNotNull` に対する関数 `y` 列。

```
SELECT assumeNotNull(y) FROM t_null
```

assumeNotNull(y)
0
3

```
SELECT toTypeName(assumeNotNull(y)) FROM t_null
```

toTypeName(assumeNotNull(y))
Int8
Int8

## toNullable

引数の型を次のように変換します `Nullable`.

```
toNullable(x)
```

パラメータ:

- `x` — The value of any non-compound type.

戻り値

- Aを持つ入力値 `Nullable` タイプ。

例

```
SELECT toTypeName(10)
```

```
└─ toTypeName(10) ──  
   └─ UInt8 ──
```

```
SELECT toTypeName(toNullable(10))
```

```
└─ toTypeName(toNullable(10)) ──  
   └─ Nullable(UInt8) ──
```

## 機械学習機能

### evalMLMethod(予測)

近似回帰モデルを使用した予測では `evalMLMethod` 機能。 リンクを参照 [linearRegression](#).

#### 確率的線形回帰

その `stochasticLinearRegression` 集計関数は線形モデルとMSE損失関数を用いた確率勾配降下法を実装する。用途 `evalMLMethod` 新しいデータを予測する。

#### 確率ロジスティック回帰

その `stochasticLogisticRegression` 集計関数は、バイナリ分類問題のための確率勾配降下法を実装しています。用途 `evalMLMethod` 新しいデータを予測する。

## 内観関数

この章で説明する関数を使用して、インストロスペクトできます `ELF` と `DWARF` クエリプロファイル用。

### 警告

これらの機能は、が必要となる場合があり安全に配慮し

内観機能の適切な操作のため:

- インストール `clickhouse-common-static-dbg` パッケージ。
- セット `allow_introspection_functions` 1に設定します。

For security reasons introspection functions are disabled by default.

ClickHouseはプロファイラレポートを `trace_log` システムテーブル。 のテーブルプロファイラで設定されます。

## アドレスストリン

ClickHouse serverプロセス内の仮想メモリアドレスを、ClickHouseソースコード内のファイル名と行番号に変換します。

公式のClickHouseパッケージを使用する場合は、`clickhouse-common-static-dbg` パッケージ。

## 構文

```
addressToLine(address_of_binary_instruction)
```

### パラメータ

- `address_of_binary_instruction` (`UInt64`) — Address of instruction in a running process.

### 戻り値

- ソースコードのファイル名とこのファイル内の行番号をコロンで区切ります。

For example, `/build/obj-x86\_64-linux-gnu/../src/Common/ThreadPool.cpp:199`, where `199` is a line number.

- 関数がデバッグ情報を見つけられなかった場合のバイナリの名前。

- アドレスが無効な場合は、空の文字列。

タイプ: 文字列.

### 例

イントロスペクション機能の有効化:

```
SET allow_introspection_functions=1
```

から最初の文字列を選択する `trace_log` システム表:

```
SELECT * FROM system.trace_log LIMIT 1 \G
```

Row 1:

```
event_date: 2019-11-19
event_time: 2019-11-19 18:57:23
revision: 54429
timer_type: Real
thread_number: 48
query_id: 421b6855-1858-45a5-8f37-f383409d6d72
trace:
[140658411141617,94784174532828,94784076370703,94784076372094,94784076361020,94784175007680,14065841116251,140658403895439]
```

その `trace` 分野のスタックトレースを瞬時にサンプリングします。

単一のアドレスのソースコードのファイル名と行番号の取得:

```
SELECT addressToLine(94784076370703) \G
```

Row 1:

```
addressToLine(94784076370703): /build/obj-x86_64-linux-gnu/../src/Common/ThreadPool.cpp:199
```

スタックトレース全体への関数の適用:

```
SELECT
    arrayStringConcat(arrayMap(x -> addressToLine(x), trace), '\n') AS trace_source_code_lines
FROM system.trace_log
LIMIT 1
\G
```

その `arrayMap` 機能はの各々の個々の要素を処理することを割り当てます `trace` による配列 `addressToLine` 機能。この処理の結果は `trace_source_code_lines` 出力の列。

Row 1:

```
trace_source_code_lines: /lib/x86_64-linux-gnu/libpthread-2.27.so
/usr/lib/debug/usr/bin/clickhouse
/build/obj-x86_64-linux-gnu../src/Common/ThreadPool.cpp:199
/build/obj-x86_64-linux-gnu../src/Common/ThreadPool.h:155
/usr/include/c++/9/bits/atomic_base.h:551
/usr/lib/debug/usr/bin/clickhouse
/lib/x86_64-linux-gnu/libpthread-2.27.so
/build/glibc-OTsEL5/glibc-2.27/misc../sysdeps/unix/sysv/linux/x86_64/clone.S:97
```

## addressToSymbol

に変換する仮想メモリアドレス内ClickHouseサーバプロセスのシンボルからClickHouseオブジェクトファイルです。

構文

```
addressToSymbol(address_of_binary_instruction)
```

パラメータ

- `address_of_binary_instruction` (`UInt64`) — Address of instruction in a running process.

戻り値

- ClickHouseオブジェクトファイルからの記号。
- アドレスが無効な場合は、空の文字列。

タイプ: [文字列](#).

例

イントロスペクション機能の有効化:

```
SET allow_introspection_functions=1
```

から最初の文字列を選択する `trace_log` システム表:

```
SELECT * FROM system.trace_log LIMIT 1 \G
```

Row 1:

```
event_date: 2019-11-20
event_time: 2019-11-20 16:57:59
revision: 54429
timer_type: Real
thread_number: 48
query_id: 724028bf-f550-45aa-910d-2af6212b94ac
trace:
[94138803686098, 94138815010911, 94138815096522, 94138815101224, 94138815102091, 94138814222988, 94138806823642, 94138814457211, 94138806823642, 94138814457211, 94138806823642, 94138806795179, 94138806796, 94138753770094, 94138753771646, 94138753760572, 94138852407232, 140399185266395, 140399178045583]
```

その `trace` 分野のスタックトレースを瞬時にサンプリングします。

単一のアドレスのシンボルの取得:

```
SELECT addressToSymbol(94138803686098) \G
```

Row 1:

```
addressToSymbol(94138803686098):
_ZNK2DB24IAggregateFunctionHelperINS_20AggregateFunctionSumImmNS_24AggregateFunctionSumDataImEEEEEE1
9addBatchSinglePlaceEmPcPPKNS_7IColumnEPNS_5ArenaE
```

スタックトレース全体への関数の適用:

```
SELECT
  arrayStringConcat(arrayMap(x -> addressToSymbol(x), trace), '\n') AS trace_symbols
FROM system.trace_log
LIMIT 1
\G
```

その `arrayMap` 機能はの各々の個々の要素を処理することを割り当てます `trace` による配列 `addressToSymbols` 機能。  
この処理の結果は `trace_symbols` 出力の列。

Row 1:

```
trace_symbols:
_ZNK2DB24IAggregateFunctionHelperINS_20AggregateFunctionSumImmNS_24AggregateFunctionSumDataImEEEEEE1
9addBatchSinglePlaceEmPcPPKNS_7IColumnEPNS_5ArenaE
_ZNK2DB10Aggregator21executeWithoutKeyImplERPcmPNS0_28AggregateFunctionInstructionEPNS_5ArenaE
_ZN2DB10Aggregator14executeOnBlockESt6vectorIN3COWINS_7IColumnEE13immutable_ptrIS3_EESaIS6_EEmRNS_2
2AggregatedDataVariantsERS1_IPKS3_SaISC_EERS1_ISE_SaISE_EERb
_ZN2DB10Aggregator14executeOnBlockERKNS_5BlockERNs_22AggregatedDataVariantsERSt6vectorIPKNS_7IColumn
ESaIS9_EERS6_ISB_SaISB_EERb
_ZN2DB10Aggregator7executeERKSt10shared_ptrINS_17IBlockInputStreamEERNS_22AggregatedDataVariantsE
_ZN2DB27AggregatingBlockInputStream8readImplEv
_ZN2DB17IBlockInputStream4readEv
_ZN2DB26ExpressionBlockInputStream8readImplEv
_ZN2DB17IBlockInputStream4readEv
_ZN2DB26ExpressionBlockInputStream8readImplEv
_ZN2DB17IBlockInputStream4readEv
_ZN2DB28AsynchronousBlockInputStream9calculateEv
_ZNSt17_Function_handlerIFvvEZN2DB28AsynchronousBlockInputStream4nextEvEUIvE_E9_M_invokeERKSt9_Any_dat
a
_ZN14ThreadPoolImplI20ThreadFromGlobalPoolE6workerESt14_List_iteratorIS0_E
_ZZN20ThreadFromGlobalPoolC4IZN14ThreadPoolImplIS_E12scheduleImplIvEET_St8functionIFvvEEiSt8optionalImEEU
vE1_JEEEOS4_DpOT0_ENKUIvE_cIEv
_ZN14ThreadPoolImplISt6threadE6workerESt14_List_iteratorIS0_E
execute_native_thread_routine
start_thread
clone
```

## デマングル

を使用して取得できるシンボルを変換します **addressToSymbol** C++関数名への関数。

### 構文

```
demangle(symbol)
```

### パラメータ

- **symbol** (文字列) — Symbol from an object file.

### 戻り値

- C++関数の名前。
- シンボルが無効な場合は空の文字列。

タイプ: 文字列.

### 例

introspection機能の有効化:

```
SET allow_introspection_functions=1
```

から最初の文字列を選択する **trace\_log** システム表:

```
SELECT * FROM system.trace_log LIMIT 1 \G
```

Row 1:

```
event_date: 2019-11-20
event_time: 2019-11-20 16:57:59
revision: 54429
timer_type: Real
thread_number: 48
query_id: 724028bf-f550-45aa-910d-2af6212b94ac
trace:
[94138803686098,94138815010911,94138815096522,94138815101224,94138815102091,94138814222988,94138806795179,94138806796144,94138753770094,94138753771646,94138753760572,94138852407232,140399185266395,140399178045583]
```

その **trace** 分野のスタックトレースを瞬時にサンプリングします。

単一のアドレスの関数名の取得:

```
SELECT demangle(addressToSymbol(94138803686098)) \G
```

Row 1:

```
demangle(addressToSymbol(94138803686098)):
DB::IAggregateFunctionHelper<DB::AggregateFunctionSum<unsigned long, unsigned long,
DB::AggregateFunctionSumData<unsigned long> >>::addBatchSinglePlace(unsigned long, char*, DB::IColumn
const**, DB::Arena*) const
```

スタックトレース全体への関数の適用:

```
SELECT
    arrayStringConcat(arrayMap(x -> demangle(addressToSymbol(x)), trace), '\n') AS trace_functions
FROM system.trace_log
LIMIT 1
\G
```

その `arrayMap` 機能はの各々の個々の要素を処理することを割り当てます `trace` による配列 `demangle` 機能。この処理の結果は `trace_functions` 出力の列。

Row 1:

```
trace_functions: DB::IAggregateFunctionHelper<DB::AggregateFunctionSum<unsigned long, unsigned long,
DB::AggregateFunctionSumData<unsigned long> >::addBatchSinglePlace(unsigned long, char*, DB::IColumn
const**, DB::Arena*) const
DB::Aggregator::executeWithoutKeyImpl(char*&, unsigned long, DB::Aggregator::AggregateFunctionInstruction*,
DB::Arena*) const
DB::Aggregator::executeOnBlock(std::vector<COW<DB::IColumn>::immutable_ptr<DB::IColumn>,
std::allocator<COW<DB::IColumn>::immutable_ptr<DB::IColumn> >, unsigned long,
DB::AggregatedDataVariants&, std::vector<DB::IColumn const*, std::allocator<DB::IColumn const*> >&,
std::vector<std::vector<DB::IColumn const*, std::allocator<DB::IColumn const*> >,
std::allocator<std::vector<DB::IColumn const*, std::allocator<DB::IColumn const*> >>&, bool&)
DB::Aggregator::executeOnBlock(DB::Block const&, DB::AggregatedDataVariants&, std::vector<DB::IColumn const*,
std::allocator<DB::IColumn const*> >&, std::vector<std::vector<DB::IColumn const*, std::allocator<DB::IColumn
const*> >, std::allocator<std::vector<DB::IColumn const*, std::allocator<DB::IColumn const*> >>&, bool&)
DB::Aggregator::execute(std::shared_ptr<DB::IBlockInputStream> const&, DB::AggregatedDataVariants&)
DB::AggregatingBlockInputStream::readImpl()
DB::IBlockInputStream::read()
DB::ExpressionBlockInputStream::readImpl()
DB::IBlockInputStream::read()
DB::ExpressionBlockInputStream::readImpl()
DB::IBlockInputStream::read()
DB::AsynchronousBlockInputStream::calculate()
std::function<void ()> DB::AsynchronousBlockInputStream::next()::
{lambda()#1}>::_M_invoke(std::any_data const&)
ThreadPoolImpl<ThreadFromGlobalPool>::worker(std::list<ThreadFromGlobalPool>)
ThreadFromGlobalPool::ThreadFromGlobalPool<ThreadPoolImpl<ThreadFromGlobalPool>::scheduleImpl<void>
(std::function<void ()>, int, std::optional<unsigned long>){lambda()#3}
(ThreadPoolImpl<ThreadFromGlobalPool>::scheduleImpl<void>(std::function<void ()>, int, std::optional<unsigned
long>){lambda()#3} && ){lambda()#1}:operator()() const
ThreadPoolImpl<std::thread>::worker(std::list<std::thread>)
execute_native_thread_routine
start_thread
clone
```

## Functions for Working with Tuples **tuple**

A function that allows grouping multiple columns.

For columns with the types T1, T2, ..., it returns a `Tuple(T1, T2, ...)` type tuple containing these columns. There is no cost to execute the function.

Tuples are normally used as intermediate values for an argument of IN operators, or for creating a list of formal parameters of lambda functions. Tuples can't be written to a table.

The function implements the operator `(x, y, ...)`.

### Syntax

```
tuple(x, y, ...)
```

## tupleElement

A function that allows getting a column from a tuple.

'N' is the column index, starting from 1. N must be a constant. 'N' must be a constant. 'N' must be a strict positive integer no greater than the size of the tuple.

There is no cost to execute the function.

The function implements the operator `x.N`.

## Syntax

```
tupleElement(tuple, n)
```

# untuple

Performs syntactic substitution of `tuple` elements in the call location.

## Syntax

```
untuple(x)
```

You can use the `EXCEPT` expression to skip columns as a result of the query.

## Arguments

- `x` — A `tuple` function, column, or tuple of elements. [Tuple](#).

## Returned value

- None.

## Examples

Input table:

key	v1	v2	v3	v4	v5	v6
1	10	20	40	30	15	(33,'ab')
2	25	65	70	40	6	(44,'cd')
3	57	30	20	10	5	(55,'ef')
4	55	12	7	80	90	(66,'gh')
5	30	50	70	25	55	(77,'kl')

Example of using a `Tuple`-type column as the `untuple` function parameter:

Query:

```
SELECT untuple(v6) FROM kv;
```

Result:

ut_1	ut_2
33	ab
44	cd
55	ef
66	gh
77	kl

Note: the names are implementation specific and are subject to change. You should not assume specific names of the columns after application of the `untuple`.

Example of using an EXCEPT expression:

Query:

```
SELECT untuple(* EXCEPT (v2, v3),) FROM kv;
```

Result:

key	v1	v4	v5	v6
1	10	30	15	(33,'ab')
2	25	40	6	(44,'cd')
3	57	10	5	(55,'ef')
4	55	80	90	(66,'gh')
5	30	25	55	(77,'kl')

## See Also

- [Tuple](#)

# tupleHammingDistance

Returns the [Hamming Distance](#) between two tuples of the same size.

## Syntax

```
tupleHammingDistance(tuple1, tuple2)
```

## Arguments

- `tuple1` — First tuple. [Tuple](#).
- `tuple2` — Second tuple. [Tuple](#).

Tuples should have the same type of the elements.

## Returned value

- The Hamming distance.

Type: [UInt8](#).

## Examples

Query:

```
SELECT tupleHammingDistance((1, 2, 3), (3, 2, 1)) AS HammingDistance;
```

Result:

HammingDistance
2

Can be used with [MinHash](#) functions for detection of semi-duplicate strings:

```
SELECT tupleHammingDistance(wordShingleMinHash(string), wordShingleMinHashCaseInsensitive(string)) as HammingDistance FROM (SELECT 'ClickHouse is a column-oriented database management system for online analytical processing of queries.' AS string);
```

Result:

```
HammingDistance
2 |
```

## tupleToNameValuePairs

Turns a named tuple into an array of (name, value) pairs. For a Tuple(a T, b T, ..., c T) returns Array(Tuple(String, T), ...)

in which the `String`s represents the named fields of the tuple and `T` are the values associated with those names. All values in the tuple should be of the same type.

### Syntax

```
tupleToNameValuePairs(tuple)
```

**\*\*Arguments\*\***

- `tuple` — Named tuple. [Tuple](#sql-reference-data-types-tuple-md) with any types of values.

**\*\*Returned value\*\***

- An array with (name, value) pairs.

Type: [Array](#sql-reference-data-types-array-md)([Tuple](#sql-reference-data-types-tuple-md)([String](#sql-reference-data-types-string-md), ...)).

**\*\*Example\*\***

Query:

```
``` sql
CREATE TABLE tupletest (`col` Tuple(user_ID UInt64, session_ID UInt64) ENGINE = Memory;
INSERT INTO tupletest VALUES (tuple( 100, 2502)), (tuple(1,100));
SELECT tupleToNameValuePairs(col) FROM tupletest;
```

Result:

```
tupleToNameValuePairs(col)
[('user_ID',100),('session_ID',2502)] |
[('user_ID',1),('session_ID',100)] |
```

It is possible to transform columns to rows using this function:

```
CREATE TABLE tupletest (`col` Tuple(CPU Float64, Memory Float64, Disk Float64)) ENGINE = Memory;
INSERT INTO tupletest VALUES(tuple(3.3, 5.5, 6.6));
SELECT arrayJoin(tupleToNameValuePairs(col))FROM tupletest;
```

Result:

```
arrayJoin(tupleToNameValuePairs(col))  
('CPU',3.3) |  
('Memory',5.5) |  
('Disk',6.6) ]
```

If you pass a simple tuple to the function, ClickHouse uses the indexes of the values as their names:

```
SELECT tupleToNameValuePairs(tuple(3, 2, 1));
```

Result:

```
tupleToNameValuePairs(tuple(3, 2, 1))  
[('1',3),('2',2),('3',1)] ]
```

```
### tuplePlus {#tupleplus}
```

Calculates the sum of corresponding values of two tuples of the same size.

**Syntax**

```
```sql  
tuplePlus(tuple1, tuple2)
```

Alias: `vectorSum`.

## Arguments

- `tuple1` — First tuple. [Tuple](#).
- `tuple2` — Second tuple. [Tuple](#).

## Returned value

- Tuple with the sum.

Type: [Tuple](#).

## Example

Query:

```
SELECT tuplePlus((1, 2), (2, 3));
```

Result:

```
tuplePlus((1, 2), (2, 3))  
(3,5) ]
```

## tupleMinus

Calculates the subtraction of corresponding values of two tuples of the same size.

## Syntax

```
tupleMinus(tuple1, tuple2)
```

Alias: `vectorDifference`.

## Arguments

- `tuple1` — First tuple. [Tuple](#).
- `tuple2` — Second tuple. [Tuple](#).

## Returned value

- Tuple with the result of subtraction.

Type: [Tuple](#).

## Example

Query:

```
SELECT tupleMinus((1, 2), (2, 3));
```

Result:

```
tupleMinus((1, 2), (2, 3))—  
(-1,-1) |
```

# tupleMultiply

Calculates the multiplication of corresponding values of two tuples of the same size.

## Syntax

```
tupleMultiply(tuple1, tuple2)
```

## Arguments

- `tuple1` — First tuple. [Tuple](#).
- `tuple2` — Second tuple. [Tuple](#).

## Returned value

- Tuple with the multiplication.

Type: [Tuple](#).

## Example

Query:

```
SELECT tupleMultiply((1, 2), (2, 3));
```

Result:

```
tupleMultiply((1, 2), (2, 3))—  
(2,6) |
```

# tupleDivide

Calculates the division of corresponding values of two tuples of the same size. Note that division by zero will return inf.

## Syntax

```
tupleDivide(tuple1, tuple2)
```

## Arguments

- `tuple1` — First tuple. [Tuple](#).
- `tuple2` — Second tuple. [Tuple](#).

## Returned value

- Tuple with the result of division.

Type: [Tuple](#).

## Example

Query:

```
SELECT tupleDivide((1, 2), (2, 3));
```

Result:

```
tupleDivide((1, 2), (2, 3))—  
(0.5,0.6666666666666666) |
```

# tupleNegate

Calculates the negation of the tuple values.

## Syntax

```
tupleNegate(tuple)
```

## Arguments

- `tuple` — [Tuple](#).

## Returned value

- Tuple with the result of negation.

Type: [Tuple](#).

## Example

Query:

```
SELECT tupleNegate((1, 2));
```

Result:

```
tupleNegate((1, 2))  
(-1,-2)
```

## tupleMultiplyByNumber

Returns a tuple with all values multiplied by a number.

### Syntax

```
tupleMultiplyByNumber(tuple, number)
```

### Arguments

- `tuple` — [Tuple](#).
- `number` — Multiplier. [Int/UInt](#), [Float](#) or [Decimal](#).

### Returned value

- Tuple with multiplied values.

Type: [Tuple](#).

### Example

Query:

```
SELECT tupleMultiplyByNumber((1, 2), -2.1);
```

Result:

```
tupleMultiplyByNumber((1, 2), -2.1)  
(-2.1,-4.2)
```

## tupleDivideByNumber

Returns a tuple with all values divided by a number. Note that division by zero will return `inf`.

### Syntax

```
tupleDivideByNumber(tuple, number)
```

### Arguments

- `tuple` — [Tuple](#).
- `number` — Divider. [Int/UInt](#), [Float](#) or [Decimal](#).

### Returned value

- Tuple with divided values.

Type: [Tuple](#).

## Example

Query:

```
SELECT tupleDivideByNumber((1, 2), 0.5);
```

Result:

```
tupleDivideByNumber((1, 2), 0.5)─  
|  
(2,4)
```

## dotProduct

Calculates the scalar product of two tuples of the same size.

### Syntax

```
dotProduct(tuple1, tuple2)
```

Alias: `scalarProduct`.

### Arguments

- `tuple1` — First tuple. [Tuple](#).
- `tuple2` — Second tuple. [Tuple](#).

### Returned value

- Scalar product.

Type: [Int/UInt](#), [Float](#) or [Decimal](#).

## Example

Query:

```
SELECT dotProduct((1, 2), (2, 3));
```

Result:

```
dotProduct((1, 2), (2, 3))─  
|  
8
```

## L1Norm

Calculates the sum of absolute values of a tuple.

### Syntax

```
L1Norm(tuple)
```

Alias: `normL1`.

### Arguments

- `tuple` — [Tuple](#).

### Returned value

- L1-norm or [taxicab geometry](#) distance.

Type: [UInt](#), [Float](#) or [Decimal](#).

### Example

Query:

```
SELECT L1Norm((1, 2));
```

Result:

```
└─L1Norm((1, 2))─┐  
      3 |
```

## L2Norm

Calculates the square root of the sum of the squares of the tuple values.

### Syntax

```
L2Norm(tuple)
```

Alias: `normL2`.

### Arguments

- `tuple` — [Tuple](#).

### Returned value

- L2-norm or [Euclidean distance](#).

Type: [Float](#).

### Example

Query:

```
SELECT L2Norm((1, 2));
```

Result:

```
└─L2Norm((1, 2))─┐  
      2.23606797749979 |
```

## LinfNorm

Calculates the maximum of absolute values of a tuple.

### Syntax

```
LinfNorm(tuple)
```

Alias: `normLinf`.

## Arguments

- `tuple` — `Tuple`.

## Returned value

- Linf-norm or the maximum absolute value.

Type: `Float`.

## Example

Query:

```
SELECT LinfNorm((1, -2));
```

Result:

```
└─LinfNorm((1, -2)) ─  
    2 |
```

# LpNorm

Calculates the root of p-th power of the sum of the absolute values of a tuple in the power of p.

## Syntax

```
LpNorm(tuple, p)
```

Alias: `normLp`.

## Arguments

- `tuple` — `Tuple`.
- `p` — The power. Possible values: real number in [1; inf]. `UInt` or `Float`.

## Returned value

- Lp-norm

Type: `Float`.

## Example

Query:

```
SELECT LpNorm((1, -2), 2);
```

Result:

```
LpNorm((1, -2), 2)─  
2.23606797749979 |
```

## L1Distance

Calculates the distance between two points (the values of the tuples are the coordinates) in L1 space (1-norm ([taxicab geometry](#) distance)).

### Syntax

```
L1Distance(tuple1, tuple2)
```

Alias: `distanceL1`.

### Arguments

- `tuple1` — First tuple. [Tuple](#).
- `tuple1` — Second tuple. [Tuple](#).

### Returned value

- 1-norm distance.

Type: [Float](#).

### Example

Query:

```
SELECT L1Distance((1, 2), (2, 3));
```

Result:

```
L1Distance((1, 2), (2, 3))─  
2 |
```

## L2Distance

Calculates the distance between two points (the values of the tuples are the coordinates) in Euclidean space ([Euclidean distance](#)).

### Syntax

```
L2Distance(tuple1, tuple2)
```

Alias: `distanceL2`.

### Arguments

- `tuple1` — First tuple. [Tuple](#).
- `tuple1` — Second tuple. [Tuple](#).

### Returned value

- 2-norm distance.

Type: **Float**.

### Example

Query:

```
SELECT L2Distance((1, 2), (2, 3));
```

Result:

```
L2Distance((1, 2), (2, 3))  
1.4142135623730951 |
```

## LinfDistance

Calculates the distance between two points (the values of the tuples are the coordinates) in  $L_{\infty}$  space (**maximum norm**).

### Syntax

```
LinfDistance(tuple1, tuple2)
```

Alias: `distanceLinf`.

### Arguments

- `tuple1` — First tuple. **Tuple**.
- `tuple2` — Second tuple. **Tuple**.

### Returned value

- Infinity-norm distance.

Type: **Float**.

### Example

Query:

```
SELECT LinfDistance((1, 2), (2, 3));
```

Result:

```
LinfDistance((1, 2), (2, 3))  
1 |
```

## LpDistance

Calculates the distance between two points (the values of the tuples are the coordinates) in  $L_p$  space (**p-norm distance**).

### Syntax

```
LpDistance(tuple1, tuple2, p)
```

Alias: `distanceLp`.

## Arguments

- `tuple1` — First tuple. [Tuple](#).
- `tuple2` — Second tuple. [Tuple](#).
- `p` — The power. Possible values: real number from  $[1; \infty)$ . [UInt](#) or [Float](#).

## Returned value

- p-norm distance.

Type: [Float](#).

## Example

Query:

```
SELECT LpDistance((1, 2), (2, 3), 3);
```

Result:

```
└── LpDistance((1, 2), (2, 3), 3) ─  
    1.2599210498948732 |
```

# L1Normalize

Calculates the unit vector of a given vector (the values of the tuple are the coordinates) in [L1](#) space ([taxicab geometry](#)).

## Syntax

```
L1Normalize(tuple)
```

Alias: `normalizeL1`.

## Arguments

- `tuple` — [Tuple](#).

## Returned value

- Unit vector.

Type: [Tuple of Float](#).

## Example

Query:

```
SELECT L1Normalize((1, 2));
```

Result:

```
L1Normalize((1, 2))  
[0.33333333333333, 0.66666666666666] |
```

## L2Normalize

Calculates the unit vector of a given vector (the values of the tuple are the coordinates) in Euclidean space (using [Euclidean distance](#)).

### Syntax

```
L2Normalize(tuple)
```

Alias: `normalizeL1`.

### Arguments

- `tuple` — [Tuple](#).

### Returned value

- Unit vector.

Type: [Tuple of Float](#).

### Example

Query:

```
SELECT L2Normalize((3, 4));
```

Result:

```
L2Normalize((3, 4))  
[0.6, 0.8] |
```

## LinfNormalize

Calculates the unit vector of a given vector (the values of the tuple are the coordinates) in  $L_{\{\infty\}}$  space (using [maximum norm](#)).

### Syntax

```
LinfNormalize(tuple)
```

Alias: `normalizeLinf`.

### Arguments

- `tuple` — [Tuple](#).

### Returned value

- Unit vector.

Type: [Tuple of Float](#).

## Example

Query:

```
SELECT LinfNormalize((3, 4));
```

Result:

```
LinfNormalize((3, 4))  
|  
(0.75,1)
```

## LpNormalize

Calculates the unit vector of a given vector (the values of the tuple are the coordinates) in Lp space (using p-norm).

### Syntax

```
LpNormalize(tuple, p)
```

Alias: `normalizeLp`.

### Arguments

- `tuple` — [Tuple](#).
- `p` — The power. Possible values: any number from [1;inf). [UInt](#) or [Float](#).

### Returned value

- Unit vector.

Type: [Tuple](#) of [Float](#).

## Example

Query:

```
SELECT LpNormalize((3, 4),5);
```

Result:

```
LpNormalize((3, 4), 5)  
|  
(0.7187302630182624,0.9583070173576831)
```

## cosineDistance

Calculates the cosine distance between two vectors (the values of the tuples are the coordinates). The less the returned value is, the more similar are the vectors.

### Syntax

```
cosineDistance(tuple1, tuple2)
```

## Arguments

- `tuple1` — First tuple. **Tuple**.
- `tuple2` — Second tuple. **Tuple**.

## Returned value

- Cosine of the angle between two vectors subtracted from one.

Type: **Float**.

## Example

Query:

```
SELECT cosineDistance((1, 2), (2, 3));
```

Result:

```
cosineDistance((1, 2), (2, 3))  
0.007722123286332261 |
```

## その他の機能

### ホスト名()

この関数が実行されたホストの名前を持つ文字列を返します。分散処理の場合、関数がリモートサーバーで実行される場合、これはリモートサーバーホストの名前です。

### getMacro

から名前付きの値を取得します **マクロ** サーバー構成のセクション。

#### 構文

```
getMacro(name);
```

パラメータ

- `name` — Name to retrieve from the `macros` セクション **文字列**.

#### 戻り値

- 指定されたマクロの値。

タイプ: **文字列**.

#### 例

例 `macros` サーバー設定ファイルのセクション:

```
<macros>  
  <test>Value</test>  
</macros>
```

クエリ:

```
SELECT getMacro('test');
```

結果:

getMacro('test')
Value

同じ値を取得する別の方法:

```
SELECT * FROM system.macros  
WHERE macro = 'test';
```

macro	substitution
test	Value

## FQDN

完全修飾ドメイン名を返します。

構文

```
fqdn();
```

この関数は、大文字と小文字を区別しません。

戻り値

- 完全修飾ドメイン名を持つ文字列。

タイプ: `String`.

例

クエリ:

```
SELECT FQDN();
```

結果:

FQDN()
clickhouse.ru-central1.internal

## ベース名

最後のスラッシュまたはバックスラッシュの後の文字列の末尾の部分を抽出します。 この関数は、パスからファイル名を抽出するためによく使用されます。

```
basename( expr )
```

## パラメータ

- **expr** — Expression resulting in a 文字列 タイプ値。結果の値では、すべての円記号をエスケープする必要があります。

## 戻り値

以下を含む文字列:

- 最後のスラッシュまたはバックスラッシュの後の文字列の末尾の部分。

If the input string contains a path ending with slash or backslash, for example, `/` or `c:\`, the function returns an empty string.

- スラッシュまたは円記号がない場合の元の文字列。

## 例

```
SELECT 'some/long/path/to/file' AS a, basename(a)
```

a——— basename('some\\long\\path\\to\\file')  
some\\long\\path\\to\\file | file |

```
SELECT 'some\\long\\path\\to\\file' AS a, basename(a)
```

a——— basename('some\\long\\path\\to\\file')  
some\\long\\path\\to\\file | file |

```
SELECT 'some-file-name' AS a, basename(a)
```

a——— basename('some-file-name')  
some-file-name | some-file-name |

## visibleWidth(x)

テキスト形式（タブ区切り）でコンソールに値を出力するときのおおよその幅を計算します。

この関数は、システムがPretty形式を実装するために使用されます。

NULL に対応する文字列として表される。NULL で Pretty フォーマット。

```
SELECT visibleWidth(NULL)
```

visibleWidth(NULL)  
4 |

## トータイプ名(x)

渡された引数の型名を含む文字列を返します。

もし `NULL` 関数に入力として渡されると、`Nullable(Nothing)` 内部に対応する型 `NULL ClickHouse`での表現。

## ブロックサイズ()

ブロックのサイズを取得します。

`ClickHouse`では、クエリは常にブロック(列部分のセット)で実行されます。この関数では、呼び出したブロックのサイズを取得できます。

## マテリアライズ(x)

定数を一つの値だけを含む完全な列に変換します。

`ClickHouse`では、完全な列と定数はメモリ内で異なって表されます。関数は、定数引数と通常の引数（異なるコードが実行される）では異なる動作をしますが、結果はほとんど常に同じです。この機能はデバッグするための

## ignore(...)

受け入れる引数を含む `NULL`。常に0を返します。

しかし、引数はまだ評価されています。これはベンチマークに使用できます。

## スリープ(秒)

眠る ‘seconds’ 各データブロックの秒数。整数または浮動小数点数を指定できます。

## sleepEachRow(秒)

眠る ‘seconds’ 各行の秒数。整数または浮動小数点数を指定できます。

## currentDatabase()

現在のデータベースの名前を返します。

この関数は、データベースを指定する必要がある`CREATE TABLE`クエリのテーブルエンジンパラメーターで使用できます。

## currentUser()

現在のユーザーのログインを返します。ユーザーのログイン、その開始されたクエリは、`distributed`クエリの場合に返されます。

```
SELECT currentUser();
```

別名: `user()`, `USER()`.

### 戻り値

- 現在のユーザーのログイン。
- 分割されたクエリの場合にクエリを開始したユーザーのログイン。

タイプ: `String`.

### 例

クエリ:

```
SELECT currentUser();
```

結果:

```
currentUser()─  
default |
```

## isConstant

引数が定数式かどうかを確認します。

A constant expression means an expression whose resulting value is known at the query analysis (i.e. before execution). For example, expressions over リテラル は定数式です。

この機能は開発のアプリケーションのデバッグおよびデモンストレーション

### 構文

```
isConstant(x)
```

#### パラメータ

- $x$  — Expression to check.

#### 戻り値

- 1 —  $x$  定数です。
- 0 —  $x$  は定数ではありません。

タイプ: UInt8.

#### 例

クエリ:

```
SELECT isConstant(x + 1) FROM (SELECT 43 AS x)
```

結果:

```
isConstant(plus(x, 1))─  
1 |
```

クエリ:

```
WITH 3.14 AS pi SELECT isConstant(cos(pi))
```

結果:

```
isConstant(cos(pi))─  
1 |
```

クエリ:

```
SELECT isConstant(number) FROM numbers(1)
```

結果:

```
isConstant(number)
  0 |
```

## isFinite(x)

Float32とFloat64を受け入れ、引数が無限でなくNaNでない場合はUInt8を1、それ以外の場合は0を返します。

### イシンフィナイト(x)

Float32とFloat64を受け入れ、引数が無限の場合はUInt8を1、それ以外の場合は0を返します。 NaNの場合は0が返されることに注意してください。

## ifNotFinite

浮動小数点値が有限かどうかをチェックします。

### 構文

```
ifNotFinite(x,y)
```

### パラメータ

- `x` — Value to be checked for infinity. Type: フロート\*.
- `y` — Fallback value. Type: フロート\*.

### 戻り値

- `x` もし `x` は有限である。
- `y` もし `x` 有限ではない。

### 例

クエリ:

```
SELECT 1/0 as infimum, ifNotFinite(infimum,42)
```

結果:

```
infimum---ifNotFinite(divide(1, 0), 42)-
  inf |           42 |
```

同様の結果を得るには、次のようにします 三項演算子: `isFinite(x) ? x : y`.

### イスナン(x)

Float32とFloat64を受け入れ、引数がNaNの場合はUInt8が1、それ以外の場合は0を返します。

`hasColumnInTable(['hostname'][, 'username'][, 'password'][], [, 'database', 'table', 'column'])`

データベース名、テーブル名、および列名の定数文字列を受け入れます。列がある場合はUInt8定数式を1、それ以外の場合は0を返します。Hostnameパラメータが設定されている場合、テストはリモートサーバーで実行されます。テーブルが存在しない場合、この関数は例外をスローします。

入れ子になったデータ構造の要素の場合、関数は列の存在をチェックします。入れ子になったデータ構造自体の場合、関数は0を返します。

ノバ一

Unicodeアート図を作成できます。

`bar(x, min, max, width)` 幅に比例したバンドを描画します ( $x - \text{min}$ ) と等しい `width` ときの文字  $x = \text{max}$ .

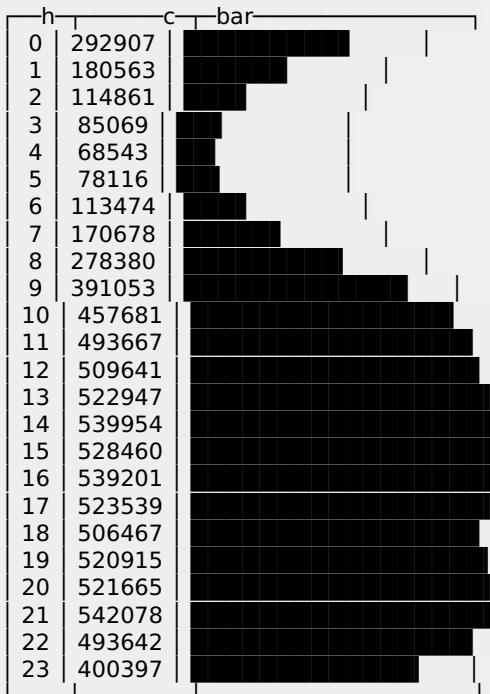
## パラメータ:

- `x` — Size to display.
  - `min, max` — Integer constants. The value must fit in `Int64`.
  - `width` — Constant, positive integer, can be fractional.

バンドは、シンボルの第八に正確に描かれています。

例：

```
SELECT
    toHour(EventTime) AS h,
    count() AS c,
    bar(c, 0, 600000, 20) AS bar
FROM test.hits
GROUP BY h
ORDER BY h ASC
```



# 变换

いくつかの要素と他の要素との明示的に定義されたマッピングに従って値を変換します。  
あ一ならではのバリエーション機能:

`transform(x, array from, array to, デフォルト)`

`x` – What to transform.

`array_from` – Constant array of values for converting.

`array_to` – Constant array of values to convert the values in ‘from’ に。

`default` – Which value to use if ‘`x`’ の値のいずれとも等しくない。‘from’.

`array_from` と `array_to` – Arrays of the same size.

タイプ:

`transform(T, Array(T), Array(U), U) -> U`

`T` と `U` 数値、文字列、または日付または`DateTime`型を指定できます。

同じ文字が示されている場合 (`T`または`U`) 、数値型の場合、これらは一致する型ではなく、共通の型を持つ型である可能性があります。

たとえば、最初の引数は`Int64`型で、二番目の引数は配列(`UInt16`)型です。

もし ‘`x`’ 値は、要素のいずれかに等しいです。‘`array_from`’ これは、配列から既存の要素（つまり、同じ番号が付けられている）を返します。‘`array_to`’ 配列 それ以外の場合は、‘`default`’. 複数の一致する要素がある場合 ‘`array_from`’, 一致するもののいずれかを返します。

例:

```
SELECT
    transform(SearchEngineID, [2, 3], ['Yandex', 'Google'], 'Other') AS title,
    count() AS c
FROM test.hits
WHERE SearchEngineID != 0
GROUP BY title
ORDER BY c DESC
```

title	c
Yandex	498635
Google	229872
Other	104472

## 変換(`x, array_from, array_to`)

最初のパリエーションとは異なり、‘`default`’ 引数は省略される。

もし ‘`x`’ 値は、要素のいずれかに等しいです。‘`array_from`’ これは、配列から一致する要素（つまり、同じ番号が付けられている）を返します。‘`array_to`’ 配列 それ以外の場合は、‘`x`’.

タイプ:

`transform(T, Array(T), Array(T)) -> T`

例:

```
SELECT
    transform(domain(Referer), ['yandex.ru', 'google.ru', 'vk.com'], ['www.yandex', 'example.com']) AS s,
    count() AS c
FROM test.hits
GROUP BY domain(Referer)
ORDER BY count() DESC
LIMIT 10
```

S		C
www.yandex	2906259	
[REDACTED].ru	867767	
mail.yandex.ru	313599	
[REDACTED].ru	107147	
[REDACTED].ru	100355	
news.yandex.ru	65040	
[REDACTED].net	64515	
example.com	59141	
	57316	

## formatReadableSize(x)

サイズ(バイト数)を受け入れます。 サフィックス(kib、MiBなど)を持つ丸められたサイズを返します。) 文字列として。

例:

```
SELECT
    arrayJoin([1, 1024, 1024*1024, 192851925]) AS filesize_bytes,
    formatReadableSize(filesize_bytes) AS filesize
```

filesize_bytes	filesize
1	1.00 B
1024	1.00 KiB
1048576	1.00 MiB
192851925	183.92 MiB

## 最小(a,b)

Aとbの最小値を返します。

## グレイテスト(a,b)

Aとbの最大値を返します。

## 稼働時間()

サーバーの稼働時間を秒単位で返します。

## バージョン()

サーバーのバージョンを文字列として返します。

## タイムゾーン()

サーバーのタイムゾーンを返します。

## ブロックナンバー

行があるデータブロックのシーケンス番号を返します。

## rowNumberInBlock

データブロック内の行の序数を返します。 異なるデータブロックは常に再計算されます。

## ローナンバリンブロック()

データブロック内の行の序数を返します。この機能のみを考慮した影響のデータブロックとなります。

## 隣人

指定された列の現在の行の前または後に来る指定されたオフセットの行へのアクセスを提供するウィンドウ関数。

### 構文

```
neighbor(column, offset[, default_value])
```

関数の結果は、影響を受けるデータブロックとブロック内のデータの順序によって異なります。

ORDER BYを使用してサブクエリを作成し、サブクエリの外部から関数を呼び出すと、期待される結果が得られます。

### パラメータ

- `column` — A column name or scalar expression.
- `offset` — The number of rows forwards or backwards from the current row of `column`. [Int64](#).
- `default_value` — Optional. The value to be returned if offset goes beyond the scope of the block. Type of data blocks affected.

### 戻り値

- の値 `column` で `offset` 現在の行からの距離if `offset` 値がブロック境界外ではありません。
- のデフォルト値 `column` もし `offset` 値はブロック境界外です。もし `default_value` が与えられると、それが使用されます。

タイプ: タイプのデータブロックの影響を受けまたはデフォルト値タイプです。

### 例

クエリ:

```
SELECT number, neighbor(number, 2) FROM system.numbers LIMIT 10;
```

結果:

number	neighbor(number, 2)
0	2
1	3
2	4
3	5
4	6
5	7
6	8
7	9
8	0
9	0

クエリ:

```
SELECT number, neighbor(number, 2, 999) FROM system.numbers LIMIT 10;
```

結果:

number	neighbor(number, 2, 999)
0	2
1	3
2	4
3	5
4	6
5	7
6	8
7	9
8	999
9	999

この関数を使用して、前年比メトリック値を計算できます:

クエリ:

```
WITH toDate('2018-01-01') AS start_date
SELECT
    toStartOfMonth(start_date + (number * 32)) AS month,
    toInt32(month) % 100 AS money,
    neighbor(money, -12) AS prev_year,
    round(prev_year / money, 2) AS year_over_year
FROM numbers(16)
```

結果:

month	money	prev_year	year_over_year
2018-01-01	32	0	0
2018-02-01	63	0	0
2018-03-01	91	0	0
2018-04-01	22	0	0
2018-05-01	52	0	0
2018-06-01	83	0	0
2018-07-01	13	0	0
2018-08-01	44	0	0
2018-09-01	75	0	0
2018-10-01	5	0	0
2018-11-01	36	0	0
2018-12-01	66	0	0
2019-01-01	97	32	0.33
2019-02-01	28	63	2.25
2019-03-01	56	91	1.62
2019-04-01	87	22	0.25

## runningDifference(x)

Calculates the difference between successive row values in the data block.

最初の行には0を返し、後続の行ごとに前の行との差を返します。

関数の結果は、影響を受けるデータブロックとブロック内のデータの順序によって異なります。

ORDER BYを使用してサブクエリを作成し、サブクエリの外部から関数を呼び出すと、期待される結果が得られます。

例:

```

SELECT
    EventID,
    EventTime,
    runningDifference(EventTime) AS delta
FROM
(
    SELECT
        EventID,
        EventTime
    FROM events
    WHERE EventDate = '2016-11-24'
    ORDER BY EventTime ASC
    LIMIT 5
)

```

EventID	EventTime	delta
1106	2016-11-24 00:00:04	0
1107	2016-11-24 00:00:05	1
1108	2016-11-24 00:00:05	0
1109	2016-11-24 00:00:09	4
1110	2016-11-24 00:00:10	1

注意-ブロックサイズは結果に影響します。 それぞれの新しいブロックでは、runningDifference 状態がリセットされます。

```

SELECT
    number,
    runningDifference(number + 1) AS diff
FROM numbers(100000)
WHERE diff != 1

```

number	diff
0	0
65536	0

```
set max_block_size=100000 -- default value is 65536!
```

```

SELECT
    number,
    runningDifference(number + 1) AS diff
FROM numbers(100000)
WHERE diff != 1

```

number	diff
0	0

## runningDifferenceStartingWithFirstvalue

同じように runningDifference、差は、最初の行の値であり、最初の行の値を返し、後続の各行は、前の行からの差を返します。

### マクナムトストリング(num)

UInt64番号を受け入れます。 ピッグエンディアンのMACアドレスとして解釈します。 対応するMACアドレスを含む文字列をAA:BB:CC:DD:EE:FF(十六進形式のコロン区切りの数値)の形式で返します。

## マクストリングトナム(s)

MACNumToStringの逆関数。 MACアドレスの形式が無効な場合は、0を返します。

## マクストリングトゥーイ(s)

AA:BB:CC:DD:EE:FF(十六進形式のコロン区切りの数字)の形式でMACアドレスを受け入れます。 UInt64番号として最初の三つのオクテットを返します。 MACアドレスの形式が無効な場合は、0を返します。

### getSizeOfEnumType

フィールドの数を返します。 [Enum](#).

```
getSizeOfEnumType(value)
```

パラメータ:

- `value` — Value of type [Enum](#).

戻り値

- フィールドの数 [Enum](#) 入力値。
- 型が指定されていない場合、例外がスローされます [Enum](#).

例

```
SELECT getSizeOfEnumType( CAST('a' AS Enum8('a' = 1, 'b' = 2) ) ) AS x
```

## blockSerializedSize

圧縮を考慮せずにディスク上のサイズを返します。

```
blockSerializedSize(value[, value[, ...]])
```

パラメータ:

- `value` — Any value.

戻り値

- 値のブロック(圧縮なし)のためにディスクに書き込まれるバイト数。

例

```
SELECT blockSerializedSize(maxState(1)) as x
```

## toColumnName

RAM内の列のデータ型を表すクラスの名前を返します。

```
toColumnName(value)
```

パラメータ:

- `value` — Any type of value.

戻り値

- 表すために使用されるクラスの名前を持つ文字列 `value` RAMのデータ型。

の違いの例 `toTypeName` ' and ' `toColumnName`

```
SELECT toTypeName(CAST('2018-01-01 01:02:03' AS DateTime))
```

```
toTypeName(CAST('2018-01-01 01:02:03', 'DateTime'))  
|  
DateTime
```

```
SELECT toColumnName(CAST('2018-01-01 01:02:03' AS DateTime))
```

```
toColumnName(CAST('2018-01-01 01:02:03', 'DateTime'))  
|  
Const(UInt32)
```

この例では、`DateTime` が `Const(UInt32)` です。

## dumpColumnStructure

RAM内のデータ構造の詳細な説明を出力します

```
dumpColumnStructure(value)
```

パラメータ:

- `value` — Any type of value.

戻り値

- 表すために使用される構造体を記述する文字列 `value` RAMのデータ型。

例

```
SELECT dumpColumnStructure(CAST('2018-01-01 01:02:03', 'DateTime'))
```

```
dumpColumnStructure(CAST('2018-01-01 01:02:03', 'DateTime'))  
|  
DateTime, Const(size = 1, UInt32(size = 1))
```

## defaultValueOfArgumentType

データ型の既定値を出力します。

ユーザーが設定したカスタム列の既定値は含まれません。

```
defaultValueOfArgumentType(expression)
```

パラメータ:

- `expression` — Arbitrary type of value or an expression that results in a value of an arbitrary type.

戻り値

- `0` 数字のために。
- 文字列の場合は空の文字列です。
- `NULL` のために Null 可能。

例

```
SELECT defaultValueOfArgumentType( CAST(1 AS Int8) )
```

```
defaultValueOfArgumentType(CAST(1, 'Int8'))—  
0 |
```

```
SELECT defaultValueOfArgumentType( CAST(1 AS Nullable(Int8) ) )
```

```
defaultValueOfArgumentType(CAST(1, 'Nullable(Int8)'))—  
NULL |
```

## 複製

単一の値を持つ配列を作成します。

内部実装のために使用される [アレイジョイン](#)。

```
SELECT replicate(x, arr);
```

パラメータ:

- `arr` — Original array. ClickHouse creates a new array of the same length as the original and fills it with the value `x`.
- `x` — The value that the resulting array will be filled with.

戻り値

値で満たされた配列 `x`。

タイプ: `Array`.

例

クエリ:

```
SELECT replicate(1, ['a', 'b', 'c'])
```

結果:

```
replicate(1, ['a', 'b', 'c'])  
[1,1,1]
```

## filesystemAvailable

返金額の残存スペースのファイルシステムのファイルのデータベースはあります。これは、常に合計空き領域よりも小さいです ([filesystemFree](#)) 一部のスペースはOS用に予約されているため。

構文

```
filesystemAvailable()
```

戻り値

- バイト単位で使用可能な残りの領域の量。

タイプ: [UInt64](#).

例

クエリ:

```
SELECT formatReadableSize(filesystemAvailable()) AS "Available space", toTypeName(filesystemAvailable()) AS  
"Type";
```

結果:

```
Available space——— Type  
30.75 GiB | UInt64 |
```

## filesystemFree

データベースのファイルがあるファイルシステム上の空き領域の合計量を返します。も参照。[filesystemAvailable](#)

構文

```
filesystemFree()
```

戻り値

- バイト単位の空き領域の量。

タイプ: [UInt64](#).

例

クエリ:

```
SELECT formatReadableSize(filesystemFree()) AS "Free space", toTypeName(filesystemFree()) AS "Type";
```

結果:

Free space	Type
32.39 GiB	UInt64

## filesystemCapacity

ファイルシステムの容量をバイト単位で返します。評価のために、[パス](#) データディレ

構文

```
filesystemCapacity()
```

戻り値

- バイト単位のファイルシステムの容量情報。

タイプ: [UInt64](#).

例

クエリ:

```
SELECT formatReadableSize(filesystemCapacity()) AS "Capacity", toTypeName(filesystemCapacity()) AS "Type"
```

結果:

Capacity	Type
39.32 GiB	UInt64

## finalizeAggregation

集計関数の状態をとります。集計結果(最終状態)を返します。

## runningAccumulate

集計関数の状態を取り、値を持つ列を返し、最初から現在の行に、ブロック行のセットのためにこれらの状態の蓄積の結果です。

たとえば、集計関数の状態をとり（例えばrunningAccumulate(uniqState(UserID)))）、ブロックの各行に対して、すべての前の行と現在の行の状態のマージ時に集計関数の結果を返します。

したがって、関数の結果は、データのブロックへの分割およびブロック内のデータの順序に依存する。

## joinGet

この関数を使用すると、aと同じ方法でテーブルからデータを抽出できます[辞書](#)。

データの取得 [参加](#) 指定された結合キーを使用するテーブル。

サポートするだけでなくテーブルで作成された `ENGINE = Join(ANY, LEFT, <join_keys>)` 声明。

構文

```
joinGet(join_storage_table_name, `value_column`, join_keys)
```

## パラメータ

- `join_storage_table_name` — an 識別子 検索が実行される場所を示します。 識別子は既定のデータベースで検索されます(パラメーターを参照 `default_database` 設定ファイル内)。 デフォルトのデータベースを上書きするには `USE db_name` またはを指定しデータベースのテーブルのセパレータ `db_name.db_table`、例を参照。
- `value_column` — name of the column of the table that contains required data.
- `join_keys` — list of keys.

## 戻り値

キーのリストに対応する値のリストを返します。

ソーステーブルに特定のものが存在しない場合、`0` または `null` に基づいて返されます `join_use_nulls` 設定。

詳細について `join_use_nulls` で 結合操作.

## 例

入力テーブル:

```
CREATE DATABASE db_test
CREATE TABLE db_test.id_val(`id` UInt32, `val` UInt32) ENGINE = Join(ANY, LEFT, id) SETTINGS join_use_nulls = 1
INSERT INTO db_test.id_val VALUES (1,11)(2,12)(4,13)
```

id	val
4	13
2	12
1	11

クエリ:

```
SELECT joinGet(db_test.id_val,'val',toUInt32(number)) from numbers(4) SETTINGS join_use_nulls = 1
```

結果:

joinGet(db_test.id_val, 'val', toUInt32(number))
0
11
12
0

## modelEvaluate(model\_name, ...)

外部モデルの評価

モデル名とモデル引数を受け取ります。 `Float64`を返します。

## throwIf(x[,custom\_message])

引数がゼロ以外の場合は例外をスローします。

`custom_message`-オプションのパラメータです。

```
SELECT throwIf(number = 3, 'Too many') FROM numbers(10);
```

```
↙ Progress: 0.00 rows, 0.00 B (0.00 rows/s., 0.00 B/s.) Received exception from server (version 19.14.1):  
Code: 395. DB::Exception: Received from localhost:9000. DB::Exception: Too many.
```

## id

引数として使用されたのと同じ値を返します。インデックスを使用してキャンセルし、フルスキヤンのクエリパフォーマンスを取得することができます。クエリを分析してインデックスを使用する可能性がある場合、アナライザは内部を見ません `identity` 機能。

### 構文

```
identity(x)
```

### 例

クエリ:

```
SELECT identity(42)
```

結果:

```
identity(42)─  
 42 |
```

## randomPrintableASCII

のランダムなセットを持つ文字列を生成します **ASCII** 印刷可能な文字。

### 構文

```
randomPrintableASCII(length)
```

### パラメータ

- `length` — Resulting string length. Positive integer.

If you pass `length < 0`, behavior of the function is undefined.

### 戻り値

- のランダムなセットを持つ文字列 **ASCII** 印刷可能な文字。

タイプ: 文字列

### 例

```
SELECT number, randomPrintableASCII(30) as str, length(str) FROM system.numbers LIMIT 3
```

```
number─str────────────────────────────────────────────────length(randomPrintableASCII(30))─  
0 | SuiCOSTvC0csfABSw=UcSzp2.`rv8x | 30 |  
1 | 1Ag NIJ &RCN:>HVPG;PE-nO"SUFDF | 30 |  
2 | /"+<"wUTh:=Ljj Vm!c&hl*m#XTfzz | 30 |
```

# Encryption functions

These functions implement encryption and decryption of data with AES (Advanced Encryption Standard) algorithm.

Key length depends on encryption mode. It is 16, 24, and 32 bytes long for `-128-`, `-196-`, and `-256-` modes respectively.

Initialization vector length is always 16 bytes (bytes in excess of 16 are ignored).

Note that these functions work slowly until ClickHouse 21.1.

## encrypt

This function encrypts data using these modes:

- `aes-128-ecb`, `aes-192-ecb`, `aes-256-ecb`
- `aes-128-cbc`, `aes-192-cbc`, `aes-256-cbc`
- `aes-128-cfb1`, `aes-192-cfb1`, `aes-256-cfb1`
- `aes-128-cfb8`, `aes-192-cfb8`, `aes-256-cfb8`
- `aes-128-cfb128`, `aes-192-cfb128`, `aes-256-cfb128`
- `aes-128-ofb`, `aes-192-ofb`, `aes-256-ofb`
- `aes-128-gcm`, `aes-192-gcm`, `aes-256-gcm`

### Syntax

```
encrypt('mode', 'plaintext', 'key' [, iv, aad])
```

### Arguments

- `mode` — Encryption mode. [String](#).
- `plaintext` — Text that needs to be encrypted. [String](#).
- `key` — Encryption key. [String](#).
- `iv` — Initialization vector. Required for `-gcm` modes, optional for others. [String](#).
- `aad` — Additional authenticated data. It isn't encrypted, but it affects decryption. Works only in `-gcm` modes, for others would throw an exception. [String](#).

### Returned value

- Ciphertext binary string. [String](#).

### Examples

Create this table:

Query:

```

CREATE TABLE encryption_test
(
    `comment` String,
    `secret` String
)
ENGINE = Memory;

```

Insert some data (please avoid storing the keys/ivs in the database as this undermines the whole concept of encryption), also storing 'hints' is unsafe too and used only for illustrative purposes:

Query:

```

INSERT INTO encryption_test VALUES('aes-256-cfb128 no IV', encrypt('aes-256-cfb128', 'Secret',
'12345678910121314151617181920212')), \
('aes-256-cfb128 no IV, different key', encrypt('aes-256-cfb128', 'Secret', 'keykeykeykeykeykeykeykeykeyke')), \
('aes-256-cfb128 with IV', encrypt('aes-256-cfb128', 'Secret', '12345678910121314151617181920212',
'iviviviviviviv')), \
('aes-256-cbc no IV', encrypt('aes-256-cbc', 'Secret', '12345678910121314151617181920212'));

```

Query:

```
SELECT comment, hex(secret) FROM encryption_test;
```

Result:

comment	hex(secret)
aes-256-cfb128 no IV	B4972BDC4459
aes-256-cfb128 no IV, different key	2FF57C092DC9
aes-256-cfb128 with IV	5E6CB398F653
aes-256-cbc no IV	1BC0629A92450D9E73A00E7D02CF4142

Example with `-gcm`:

Query:

```

INSERT INTO encryption_test VALUES('aes-256-gcm', encrypt('aes-256-gcm', 'Secret',
'12345678910121314151617181920212', 'iviviviviviviv')), \
('aes-256-gcm with AAD', encrypt('aes-256-gcm', 'Secret', '12345678910121314151617181920212', 'iviviviviviviv',
'aad'));

SELECT comment, hex(secret) FROM encryption_test WHERE comment LIKE '%gcm%';

```

Result:

comment	hex(secret)
aes-256-gcm	A8A3CCBC6426CFEEB60E4EAE03D3E94204C1B09E0254
aes-256-gcm with AAD	A8A3CCBC6426D9A1017A0A932322F1852260A4AD6837

## aes\_encrypt\_mysql

Compatible with mysql encryption and resulting ciphertext can be decrypted with [AES\\_DECRYPT](#) function.

Will produce the same ciphertext as `encrypt` on equal inputs. But when `key` or `iv` are longer than they should normally be, `aes_encrypt_mysql` will stick to what MySQL's `aes_encrypt` does: 'fold' `key` and ignore excess bits of `iv`.

Supported encryption modes:

- aes-128-ecb, aes-192-ecb, aes-256-ecb
- aes-128-cbc, aes-192-cbc, aes-256-cbc
- aes-128-cfb1, aes-192-cfb1, aes-256-cfb1
- aes-128-cfb8, aes-192-cfb8, aes-256-cfb8
- aes-128-cfb128, aes-192-cfb128, aes-256-cfb128
- aes-128-ofb, aes-192-ofb, aes-256-ofb

## Syntax

```
aes_encrypt_mysql('mode', 'plaintext', 'key' [, iv])
```

## Arguments

- mode — Encryption mode. **String**.
- plaintext — Text that needs to be encrypted. **String**.
- key — Encryption key. If key is longer than required by mode, MySQL-specific key folding is performed. **String**.
- iv — Initialization vector. Optional, only first 16 bytes are taken into account **String**.

## Returned value

- Ciphertext binary string. **String**.

## Examples

Given equal input `encrypt` and `aes_encrypt_mysql` produce the same ciphertext:

Query:

```
SELECT encrypt('aes-256-cfb128', 'Secret', '12345678910121314151617181920212', 'iviviviviviviv') =
aes_encrypt_mysql('aes-256-cfb128', 'Secret', '12345678910121314151617181920212', 'iviviviviviviv') AS
ciphertexts_equal;
```

Result:

```
└─ciphertexts_equal─
  1 └─────────────────┘
```

But `encrypt` fails when `key` or `iv` is longer than expected:

Query:

```
SELECT encrypt('aes-256-cfb128', 'Secret', '123456789101213141516171819202122', 'iviviviviviviv123');
```

Result:

```
Received exception from server (version 21.1.2):
Code: 36. DB::Exception: Received from localhost:9000. DB::Exception: Invalid key size: 33 expected 32: While
processing encrypt('aes-256-cfb128', 'Secret', '123456789101213141516171819202122', 'iviviviviviviv123').
```

While `aes_encrypt_mysql` produces MySQL-compatitalbe output:

Query:

```
SELECT hex(aes_encrypt_mysql('aes-256-cfb128', 'Secret', '123456789101213141516171819202122',  
'iviviviviviviv123')) AS ciphertext;
```

Result:

```
ciphertext  
24E9E4966469 |
```

Notice how supplying even longer IV produces the same result

Query:

```
SELECT hex(aes_encrypt_mysql('aes-256-cfb128', 'Secret', '123456789101213141516171819202122',  
'iviviviviviviv123456')) AS ciphertext
```

Result:

```
ciphertext  
24E9E4966469 |
```

Which is binary equal to what MySQL produces on same inputs:

```
mysql> SET block_encryption_mode='aes-256-cfb128';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> SELECT aes_encrypt('Secret', '123456789101213141516171819202122', 'iviviviviviviv123456') as  
ciphertext;  
+-----+  
| ciphertext |  
+-----+  
| 0x24E9E4966469 |  
+-----+  
1 row in set (0.00 sec)
```

## decrypt

This function decrypts ciphertext into a plaintext using these modes:

- aes-128-ecb, aes-192-ecb, aes-256-ecb
- aes-128-cbc, aes-192-cbc, aes-256-cbc
- aes-128-cfb1, aes-192-cfb1, aes-256-cfb1
- aes-128-cfb8, aes-192-cfb8, aes-256-cfb8
- aes-128-cfb128, aes-192-cfb128, aes-256-cfb128
- aes-128-ofb, aes-192-ofb, aes-256-ofb
- aes-128-gcm, aes-192-gcm, aes-256-gcm

## Syntax

```
decrypt('mode', 'ciphertext', 'key' [, iv, aad])
```

## Arguments

- mode — Decryption mode. [String](#).
- ciphertext — Encrypted text that needs to be decrypted. [String](#).
- key — Decryption key. [String](#).
- iv — Initialization vector. Required for `-gcm` modes, optional for others. [String](#).
- aad — Additional authenticated data. Won't decrypt if this value is incorrect. Works only in `-gcm` modes, for others would throw an exception. [String](#).

## Returned value

- Decrypted String. [String](#).

## Examples

Re-using table from [encrypt](#).

Query:

```
SELECT comment, hex(secret) FROM encryption_test;
```

Result:

comment	hex(secret)
aes-256-gcm	A8A3CCBC6426CFEEB60E4EAE03D3E94204C1B09E0254
aes-256-gcm with AAD	A8A3CCBC6426D9A1017A0A932322F1852260A4AD6837
comment	hex(secret)
aes-256-cfb128 no IV	B4972BDC4459
aes-256-cfb128 no IV, different key	2FF57C092DC9
aes-256-cfb128 with IV	5E6CB398F653
aes-256-cbc no IV	1BC0629A92450D9E73A00E7D02CF4142

Now let's try to decrypt all that data.

Query:

```
SELECT comment, decrypt('aes-256-cfb128', secret, '12345678910121314151617181920212') as plaintext FROM encryption_test
```

Result:

comment	Secret	plaintext
aes-256-cfb128 no IV	Secret	
aes-256-cfb128 no IV, different key	♦4♦	
aes-256-cfb128 with IV	♦♦♦6♦~	
aes-256-cbc no IV	♦2*4♦h3c♦4w♦♦@	

Notice how only a portion of the data was properly decrypted, and the rest is gibberish since either mode, key, or iv were different upon encryption.

# aes\_decrypt\_mysql

Compatible with mysql encryption and decrypts data encrypted with [AES\\_ENCRYPT](#) function.

Will produce same plaintext as `decrypt` on equal inputs. But when `key` or `iv` are longer than they should normally be, `aes_decrypt_mysql` will stick to what MySQL's `aes_decrypt` does: 'fold' `key` and ignore excess bits of `IV`.

Supported decryption modes:

- aes-128-ecb, aes-192-ecb, aes-256-ecb
- aes-128-cbc, aes-192-cbc, aes-256-cbc
- aes-128-cfb1, aes-192-cfb1, aes-256-cfb1
- aes-128-cfb8, aes-192-cfb8, aes-256-cfb8
- aes-128-cfb128, aes-192-cfb128, aes-256-cfb128
- aes-128-ofb, aes-192-ofb, aes-256-ofb

## Syntax

```
aes_decrypt_mysql('mode', 'ciphertext', 'key' [, iv])
```

## Arguments

- `mode` — Decryption mode. [String](#).
- `ciphertext` — Encrypted text that needs to be decrypted. [String](#).
- `key` — Decryption key. [String](#).
- `iv` — Initialization vector. Optional. [String](#).

## Returned value

- Decrypted String. [String](#).

## Examples

Let's decrypt data we've previously encrypted with MySQL:

```
mysql> SET block_encryption_mode='aes-256-cfb128';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT aes_encrypt('Secret', '123456789101213141516171819202122', 'iviviviviviviv123456') as
ciphertext;
+-----+
| ciphertext      |
+-----+
| 0x24E9E4966469 |
+-----+
1 row in set (0.00 sec)
```

Query:

```
SELECT aes_decrypt_mysql('aes-256-cfb128', unhex('24E9E4966469'), '123456789101213141516171819202122',
'iviviviviviviv123456') AS plaintext
```

Result:

```
plaintext  
Secret |
```

# [experimental] Natural Language Processing functions

## Warning

This is an experimental feature that is currently in development and is not ready for general use. It will change in unpredictable backwards-incompatible ways in future releases. Set `allow_experimental_nlp_functions = 1` to enable it.

## stem

Performs stemming on a given word.

### Syntax

```
stem('language', word)
```

### Arguments

- `language` — Language which rules will be applied. Must be in lowercase. [String](#).
- `word` — word that needs to be stemmed. Must be in lowercase. [String](#).

### Examples

Query:

```
SELECT arrayMap(x -> stem('en', x), ['I', 'think', 'it', 'is', 'a', 'blessing', 'in', 'disguise']) as res;
```

Result:

```
res  
['I', 'think', 'it', 'is', 'a', 'bless', 'in', 'disguis'] |
```

## lemmatize

Performs lemmatization on a given word. Needs dictionaries to operate, which can be obtained [here](#).

### Syntax

```
lemmatize('language', word)
```

### Arguments

- `language` — Language which rules will be applied. [String](#).
- `word` — Word that needs to be lemmatized. Must be lowercase. [String](#).

## Examples

Query:

```
SELECT lemmatize('en', 'wolves');
```

Result:

```
└── lemmatize("wolves") └─  
      "wolf" |
```

Configuration:

```
<lemmatizers>  
  <lemmatizer>  
    <lang>en</lang>  
    <path>en.bin</path>  
  </lemmatizer>  
</lemmatizers>
```

## synonyms

Finds synonyms to a given word. There are two types of synonym extensions: plain and wordnet.

With the `plain` extension type we need to provide a path to a simple text file, where each line corresponds to a certain synonym set. Words in this line must be separated with space or tab characters.

With the `wordnet` extension type we need to provide a path to a directory with WordNet thesaurus in it. Thesaurus must contain a WordNet sense index.

## Syntax

```
synonyms('extension_name', word)
```

## Arguments

- `extension_name` — Name of the extension in which search will be performed. [String](#).
- `word` — Word that will be searched in extension. [String](#).

## Examples

Query:

```
SELECT synonyms('list', 'important');
```

Result:

```
└── synonyms('list', 'important') └─  
      ['important', 'big', 'critical', 'crucial'] |
```

Configuration:

```
<synonyms_extensions>
  <extension>
    <name>en</name>
    <type>plain</type>
    <path>en.txt</path>
  </extension>
  <extension>
    <name>en</name>
    <type>wordnet</type>
    <path>en/</path>
  </extension>
</synonyms_extensions>
```

## 集計関数

集計関数は、**標準** データベースの専門家が予想通りの方法。

ClickHouseはまた支えます:

- **パラメトリック集計関数** 列に加えて他のパラメータを受け入れます。
- **コンビネータ**、集計関数の動作を変更します。

## ヌル処理

集計中、すべて **NULL** はスキップされます。

例:

次の表を考えます:

x	y
1	2
2	NULL
3	2
3	3
3	NULL

の値を合計する必要があるとしましょう **y** 列:

```
SELECT sum(y) FROM t_null_big
```

sum(y)
7

その **sum** 関数は解釈します **NULL** として **0**。特に、これは、関数がすべての値が次のような選択範囲の入力を受け取った場合 **NULL** 結果は次のようにになります **0** ない **NULL**。

今すぐ使用することができます **groupArray** から配列を作成する関数 **y** 列:

```
SELECT groupArray(y) FROM t_null_big
```

groupArray(y)
[2,2,3]

`groupArray` 含まない `NULL` 結果の配列で。

# COUNT

Counts the number of rows or not-`NULL` values.

ClickHouse supports the following syntaxes for `count`:

- `count(expr)` or `COUNT(DISTINCT expr)`.
- `count()` or `COUNT(*)`. The `count()` syntax is ClickHouse-specific.

## Arguments

The function can take:

- Zero parameters.
- One **expression**.

## Returned value

- If the function is called without parameters it counts the number of rows.
- If the **expression** is passed, then the function counts how many times this expression returned not null. If the expression returns a **Nullable**-type value, then the result of `count` stays not **Nullable**. The function returns 0 if the expression returned `NULL` for all the rows.

In both cases the type of the returned value is **UInt64**.

## Details

ClickHouse supports the `COUNT(DISTINCT ...)` syntax. The behavior of this construction depends on the `count_distinctImplementation` setting. It defines which of the `uniq*` functions is used to perform the operation. The default is the `uniqExact` function.

The `SELECT count() FROM table` query is optimized by default using metadata from MergeTree. If you need to use row-level security, disable optimization using the `optimize_trivial_count_query` setting.

However `SELECT count(nullable_column) FROM table` query can be optimized by enabling the `optimize_functions_to_subcolumns` setting. With `optimize_functions_to_subcolumns = 1` the function reads only `null` subcolumn instead of reading and processing the whole column data. The query `SELECT count(n) FROM table` transforms to `SELECT sum(NOT n.null) FROM table`.

## Examples

Example 1:

```
SELECT count() FROM t
```

```
count()  
5 |
```

Example 2:

```
SELECT name, value FROM system.settings WHERE name = 'count_distinctImplementation'
```

name	value
count_distinctImplementation	uniqExact

```
SELECT count(DISTINCT num) FROM t
```

```
uniqExact(num)  
3 |
```

This example shows that `count(DISTINCT num)` is performed by the `uniqExact` function according to the `count_distinctImplementation` setting value.

## min

Aggregate function that calculates the minimum across a group of values.

Example:

```
SELECT min(salary) FROM employees;
```

```
SELECT department, min(salary) FROM employees GROUP BY department;
```

If you need non-aggregate function to choose a minimum of two values, see `least`:

```
SELECT least(a, b) FROM table;
```

## max

Aggregate function that calculates the maximum across a group of values.

Example:

```
SELECT max(salary) FROM employees;
```

```
SELECT department, max(salary) FROM employees GROUP BY department;
```

If you need non-aggregate function to choose a maximum of two values, see `greatest`:

```
SELECT greatest(a, b) FROM table;
```

## sum

Calculates the sum. Only works for numbers.

## avg

Calculates the arithmetic mean.

## Syntax

```
avg(x)
```

## Arguments

- `x` — input values, must be [Integer](#), [Float](#), or [Decimal](#).

## Returned value

- The arithmetic mean, always as [Float64](#).
- `Nan` if the input parameter `x` is empty.

## Example

Query:

```
SELECT avg(x) FROM values('x Int8', 0, 1, 2, 3, 4, 5);
```

Result:

```
avg(x)  
2.5 |
```

## Example

Create a temp table:

Query:

```
CREATE table test (t UInt8) ENGINE = Memory;
```

Get the arithmetic mean:

Query:

```
SELECT avg(t) FROM test;
```

Result:

```
avg(x)  
nan |
```

## any

Selects the first encountered value.

The query can be executed in any order and even in a different order each time, so the result of this function is indeterminate.

To get a determinate result, you can use the ‘min’ or ‘max’ function instead of ‘any’.

In some cases, you can rely on the order of execution. This applies to cases when SELECT comes from a subquery that uses ORDER BY.

When a SELECT query has the GROUP BY clause or at least one aggregate function, ClickHouse (in contrast to MySQL) requires that all expressions in the SELECT, HAVING, and ORDER BY clauses be calculated from keys or from aggregate functions. In other words, each column selected from the table must be used either in keys or inside aggregate functions. To get behavior like in MySQL, you can put the other columns in the `any` aggregate function.

## stddevPop

The result is equal to the square root of `varPop`.

### Note

This function uses a numerically unstable algorithm. If you need **numerical stability** in calculations, use the `stddevPopStable` function. It works slower but provides a lower computational error.

## stddevSamp

The result is equal to the square root of `varSamp`.

### Note

This function uses a numerically unstable algorithm. If you need **numerical stability** in calculations, use the `stddevSampStable` function. It works slower but provides a lower computational error.

## varPop(x)

Calculates the amount  $\sum((x - \bar{x})^2) / n$ , where `n` is the sample size and  $\bar{x}$  is the average value of `x`.

In other words, dispersion for a set of values. Returns `Float64`.

### Note

This function uses a numerically unstable algorithm. If you need **numerical stability** in calculations, use the `varPopStable` function. It works slower but provides a lower computational error.

## varSamp

Calculates the amount  $\sum((x - \bar{x})^2) / (n - 1)$ , where `n` is the sample size and  $\bar{x}$  is the average value of `x`.

It represents an unbiased estimate of the variance of a random variable if passed values form its sample.

Returns `Float64`. When `n <= 1`, returns  $+\infty$ .

### Note

This function uses a numerically unstable algorithm. If you need **numerical stability** in calculations, use the `varSampStable` function. It works slower but provides a lower computational error.

## covarPop

Syntax: `covarPop(x, y)`

Calculates the value of  $\Sigma((x - \bar{x})(y - \bar{y})) / n$ .

### Note

This function uses a numerically unstable algorithm. If you need **numerical stability** in calculations, use the `covarPopStable` function. It works slower but provides a lower computational error.

## List of Aggregate Functions

Standard aggregate functions:

- [count](#)
- [min](#)
- [max](#)
- [sum](#)
- [avg](#)
- [any](#)
- [stddevPop](#)
- [stddevSamp](#)
- [varPop](#)
- [varSamp](#)
- [covarPop](#)
- [covarSamp](#)

ClickHouse-specific aggregate functions:

- [anyHeavy](#)
- [anyLast](#)
- [argMin](#)
- [argMax](#)
- [avgWeighted](#)
- [topK](#)
- [topKWeighted](#)

- groupArray
- groupUniqArray
- groupArrayInsertAt
- groupArrayMovingAvg
- groupArrayMovingSum
- groupBitAnd
- groupBitOr
- groupBitXor
- groupBitmap
- groupBitmapAnd
- groupBitmapOr
- groupBitmapXor
- sumWithOverflow
- sumMap
- minMap
- maxMap
- skewSamp
- skewPop
- kurtSamp
- kurtPop
- uniq
- uniqExact
- uniqCombined
- uniqCombined64
- uniqHLL12
- quantile
- quantiles
- quantileExact
- quantileExactLow
- quantileExactHigh
- quantileExactWeighted
- quantileTiming
- quantileTimingWeighted

- `quantileDeterministic`
  - `quantileTDigest`
  - `quantileTDigestWeighted`
  - `quantileBFloat16`
  - `quantileBFloat16Weighted`
  - `simpleLinearRegression`
  - `stochasticLinearRegression`
  - `stochasticLogisticRegression`
  - `categoricalInformationValue`
- 

## covarSamp

Calculates the value of  $\Sigma((x - \bar{x})(y - \bar{y})) / (n - 1)$

Returns `Float64`. When `n <= 1`, returns  $+\infty$ .

### Note

This function uses a numerically unstable algorithm. If you need **numerical stability** in calculations, use the `covarSampStable` function. It works slower but provides a lower computational error.

## anyHeavy

Selects a frequently occurring value using the **heavy hitters** algorithm. If there is a value that occurs more than in half the cases in each of the query's execution threads, this value is returned. Normally, the result is nondeterministic.

```
anyHeavy(column)
```

### Arguments

- `column` – The column name.

### Example

Take the `OnTime` data set and select any frequently occurring value in the `AirlineID` column.

```
SELECT anyHeavy(AirlineID) AS res  
FROM ontime
```

```
res-  
19690 |
```

## anyLast

Selects the last value encountered.

The result is just as indeterminate as for the [any](#) function.

## argMin

Calculates the `arg` value for a minimum `val` value. If there are several different values of `arg` for minimum values of `val`, returns the first of these values encountered.

### Syntax

```
argMin(arg, val)
```

### Arguments

- `arg` — Argument.
- `val` — Value.

### Returned value

- `arg` value that corresponds to minimum `val` value.

Type: matches `arg` type.

### Example

Input table:

user	salary
director	5000
manager	3000
worker	1000

Query:

```
SELECT argMin(user, salary) FROM salary
```

Result:

argMin(user, salary)
worker

## argMax

Calculates the `arg` value for a maximum `val` value. If there are several different values of `arg` for maximum values of `val`, returns the first of these values encountered.

### Syntax

```
argMax(arg, val)
```

### Arguments

- `arg` — Argument.

- `val` — Value.

## Returned value

- `arg` value that corresponds to maximum `val` value.

Type: matches `arg` type.

## Example

Input table:

user	salary
director	5000
manager	3000
worker	1000

Query:

```
SELECT argMax(user, salary) FROM salary;
```

Result:

```
argMax(user, salary)
director
```

# avgWeighted

Calculates the [weighted arithmetic mean](#).

## Syntax

```
avgWeighted(x, weight)
```

## Arguments

- `x` — Values.
- `weight` — Weights of the values.

`x` and `weight` must both be

[Integer](#),  
[floating-point](#), or  
[Decimal](#),

but may have different types.

## Returned value

- `Nan` if all the weights are equal to 0 or the supplied `weights` parameter is empty.
- Weighted mean otherwise.

**Return type** is always [Float64](#).

## Example

Query:

```
SELECT avgWeighted(x, w)
FROM values('x Int8, w Int8', (4, 1), (1, 0), (10, 2))
```

Result:

```
avgWeighted(x, weight)─
  8 |
```

## Example

Query:

```
SELECT avgWeighted(x, w)
FROM values('x Int8, w Float64', (4, 1), (1, 0), (10, 2))
```

Result:

```
avgWeighted(x, weight)─
  8 |
```

## Example

Query:

```
SELECT avgWeighted(x, w)
FROM values('x Int8, w Int8', (0, 0), (1, 0), (10, 0))
```

Result:

```
avgWeighted(x, weight)─
  nan |
```

## Example

Query:

```
CREATE table test (t UInt8) ENGINE = Memory;
SELECT avgWeighted(t) FROM test
```

Result:

```
avgWeighted(x, weight)─
  nan |
```

corr

Syntax: `corr(x, y)`

Calculates the Pearson correlation coefficient:  $\Sigma((x - \bar{x})(y - \bar{y})) / \sqrt{\Sigma((x - \bar{x})^2) * \Sigma((y - \bar{y})^2)}$

## Note

This function uses a numerically unstable algorithm. If you need **numerical stability** in calculations, use the `corrStable` function. It works slower but provides a lower computational error.

# topK

Returns an array of the approximately most frequent values in the specified column. The resulting array is sorted in descending order of approximate frequency of values (not by the values themselves).

Implements the **Filtered Space-Saving** algorithm for analyzing TopK, based on the reduce-and-combine algorithm from **Parallel Space Saving**.

```
topK(N)(column)
```

This function does not provide a guaranteed result. In certain situations, errors might occur and it might return frequent values that aren't the most frequent values.

We recommend using the `N < 10` value; performance is reduced with large `N` values. Maximum value of `N = 65536`.

### Arguments

- `N` – The number of elements to return.

If the parameter is omitted, default value 10 is used.

### Arguments

- `x` – The value to calculate frequency.

### Example

Take the **OnTime** data set and select the three most frequently occurring values in the `AirlineID` column.

```
SELECT topK(3)(AirlineID) AS res  
FROM ontime
```

```
res  
[19393,19790,19805] |
```

# topKWeighted

Returns an array of the approximately most frequent values in the specified column. The resulting array is sorted in descending order of approximate frequency of values (not by the values themselves). Additionally, the weight of the value is taken into account.

### Syntax

```
topKWeighted(N)(x, weight)
```

## Arguments

- `N` — The number of elements to return.
- `x` — The value.
- `weight` — The weight. Every value is accounted `weight` times for frequency calculation. [UInt64](#).

## Returned value

Returns an array of the values with maximum approximate sum of weights.

## Example

Query:

```
SELECT topKWeighted(10)(number, number) FROM numbers(1000)
```

Result:

```
topKWeighted(10)(number, number)
[999,998,997,996,995,994,993,992,991,990] |
```

## See Also

- [topK](#)

# groupArray

Syntax: `groupArray(x)` or `groupArray(max_size)(x)`

Creates an array of argument values.

Values can be added to the array in any (indeterminate) order.

The second version (with the `max_size` parameter) limits the size of the resulting array to `max_size` elements. For example, `groupArray(1)(x)` is equivalent to `[any(x)]`.

In some cases, you can still rely on the order of execution. This applies to cases when `SELECT` comes from a subquery that uses `ORDER BY`.

# groupUniqArray

Syntax: `groupUniqArray(x)` or `groupUniqArray(max_size)(x)`

Creates an array from different argument values. Memory consumption is the same as for the [uniqExact](#) function.

The second version (with the `max_size` parameter) limits the size of the resulting array to `max_size` elements. For example, `groupUniqArray(1)(x)` is equivalent to `[any(x)]`.

# groupArrayInsertAt

Inserts a value into the array at the specified position.

## Syntax

```
groupArrayInsertAt(default_x, size)(x, pos)
```

If in one query several values are inserted into the same position, the function behaves in the following ways:

- If a query is executed in a single thread, the first one of the inserted values is used.
- If a query is executed in multiple threads, the resulting value is an undetermined one of the inserted values.

## Arguments

- `x` — Value to be inserted. [Expression](#) resulting in one of the [supported data types](#).
- `pos` — Position at which the specified element `x` is to be inserted. Index numbering in the array starts from zero. [UInt32](#).
- `default_x` — Default value for substituting in empty positions. Optional parameter. [Expression](#) resulting in the data type configured for the `x` parameter. If `default_x` is not defined, the [default values](#) are used.
- `size` — Length of the resulting array. Optional parameter. When using this parameter, the default value `default_x` must be specified. [UInt32](#).

## Returned value

- Array with inserted values.

Type: [Array](#).

## Example

Query:

```
SELECT groupArrayInsertAt(toString(number), number * 2) FROM numbers(5);
```

Result:

```
groupArrayInsertAt(toString(number), multiply(number, 2))—  
['0','1','2','3','4'] |
```

Query:

```
SELECT groupArrayInsertAt('-')(toString(number), number * 2) FROM numbers(5);
```

Result:

```
groupArrayInsertAt('')(toString(number), multiply(number, 2))—  
['0','-','1','-','2','-','3','-','4'] |
```

Query:

```
SELECT groupArrayInsertAt('-', 5)(toString(number), number * 2) FROM numbers(5);
```

Result:

```
groupArrayInsertAt('-', 5)(toString(number), multiply(number, 2))  
['0','-' '1','-' '2']
```

Multi-threaded insertion of elements into one position.

Query:

```
SELECT groupArrayInsertAt(number, 0) FROM numbers_mt(10) SETTINGS max_block_size = 1;
```

As a result of this query you get random integer in the [0,9] range. For example:

```
groupArrayInsertAt(number, 0)  
[7]
```

## groupArrayMovingSum

Calculates the moving sum of input values.

```
groupArrayMovingSum(numbers_for_summing)  
groupArrayMovingSum(window_size)(numbers_for_summing)
```

The function can take the window size as a parameter. If left unspecified, the function takes the window size equal to the number of rows in the column.

### Arguments

- `numbers_for_summing` — [Expression](#) resulting in a numeric data type value.
- `window_size` — Size of the calculation window.

### Returned values

- Array of the same size and type as the input data.

### Example

The sample table:

```
CREATE TABLE t  
(  
    `int` UInt8,  
    `float` Float32,  
    `dec` Decimal32(2)  
)  
ENGINE = TinyLog
```

int	float	dec
1	1.1	1.10
2	2.2	2.20
4	4.4	4.40
7	7.77	7.77

The queries:

```
SELECT
    groupArrayMovingSum(int) AS I,
    groupArrayMovingSum(float) AS F,
    groupArrayMovingSum(dec) AS D
FROM t
```

I	F	D
[1,3,7,14]	[1.1,3.3000002,7.7000003,15.47]	[1.10,3.30,7.70,15.47]

```
SELECT
    groupArrayMovingSum(2)(int) AS I,
    groupArrayMovingSum(2)(float) AS F,
    groupArrayMovingSum(2)(dec) AS D
FROM t
```

I	F	D
[1,3,6,11]	[1.1,3.3000002,6.6000004,12.17]	[1.10,3.30,6.60,12.17]

## groupArrayMovingAvg

Calculates the moving average of input values.

```
groupArrayMovingAvg(numbers_for_summing)
groupArrayMovingAvg(window_size)(numbers_for_summing)
```

The function can take the window size as a parameter. If left unspecified, the function takes the window size equal to the number of rows in the column.

### Arguments

- `numbers_for_summing` — **Expression** resulting in a numeric data type value.
- `window_size` — Size of the calculation window.

### Returned values

- Array of the same size and type as the input data.

The function uses **rounding towards zero**. It truncates the decimal places insignificant for the resulting data type.

### Example

The sample table b:

```
CREATE TABLE t
(
    `int` UInt8,
    `float` Float32,
    `dec` Decimal32(2)
)
ENGINE = TinyLog
```

int	float	dec
1	1.1	1.10
2	2.2	2.20
4	4.4	4.40
7	7.77	7.77

The queries:

```
SELECT
    groupArrayMovingAvg(int) AS I,
    groupArrayMovingAvg(float) AS F,
    groupArrayMovingAvg(dec) AS D
FROM t
```

I	F	D
[0,0,1,3]	[0.275,0.82500005,1.9250001,3.8675]	[0.27,0.82,1.92,3.86]

```
SELECT
    groupArrayMovingAvg(2)(int) AS I,
    groupArrayMovingAvg(2)(float) AS F,
    groupArrayMovingAvg(2)(dec) AS D
FROM t
```

I	F	D
[0,1,3,5]	[0.55,1.6500001,3.3000002,6.085]	[0.55,1.65,3.30,6.08]

## groupArraySample

Creates an array of sample argument values. The size of the resulting array is limited to `max_size` elements. Argument values are selected and added to the array randomly.

### Syntax

```
groupArraySample(max_size[, seed])(x)
```

### Arguments

- `max_size` — Maximum size of the resulting array. **UInt64**.
- `seed` — Seed for the random number generator. Optional. **UInt64**. Default value: **123456**.
- `x` — Argument (column name or expression).

### Returned values

- Array of randomly selected `x` arguments.

Type: **Array**.

## Examples

Consider table colors:

id	color
1	red
2	blue
3	green
4	white
5	orange

Query with column name as argument:

```
SELECT groupArraySample(3)(color) as newcolors FROM colors;
```

Result:

```
newcolors  
['white','blue','green'] |
```

Query with column name and different seed:

```
SELECT groupArraySample(3, 987654321)(color) as newcolors FROM colors;
```

Result:

```
newcolors  
['red','orange','green'] |
```

Query with expression as argument:

```
SELECT groupArraySample(3)(concat('light-', color)) as newcolors FROM colors;
```

Result:

```
newcolors  
['light-blue','light-orange','light-green'] |
```

## groupBitAnd

Applies bitwise AND for series of numbers.

```
groupBitAnd(expr)
```

## Arguments

`expr` – An expression that results in `UInt*` type.

## Return value

Value of the UInt\* type.

## Example

Test data:

```
binary  decimal
00101100 = 44
00011100 = 28
00001101 = 13
01010101 = 85
```

Query:

```
SELECT groupBitAnd(num) FROM t
```

Where num is the column with the test data.

Result:

```
binary  decimal
00000100 = 4
```

# groupBitOr

Applies bitwise OR for series of numbers.

```
groupBitOr(expr)
```

## Arguments

expr – An expression that results in UInt\* type.

## Returned value

Value of the UInt\* type.

## Example

Test data:

```
binary  decimal
00101100 = 44
00011100 = 28
00001101 = 13
01010101 = 85
```

Query:

```
SELECT groupBitOr(num) FROM t
```

Where num is the column with the test data.

Result:

```
binary  decimal  
01111101 = 125
```

## groupBitXor

Applies bitwise XOR for series of numbers.

```
groupBitXor(expr)
```

### Arguments

expr – An expression that results in UInt\* type.

### Return value

Value of the UInt\* type.

### Example

Test data:

```
binary  decimal  
00101100 = 44  
00011100 = 28  
00001101 = 13  
01010101 = 85
```

Query:

```
SELECT groupBitXor(num) FROM t
```

Where num is the column with the test data.

Result:

```
binary  decimal  
01101000 = 104
```

## groupBitmap

Bitmap or Aggregate calculations from a unsigned integer column, return cardinality of type UInt64, if add suffix -State, then return [bitmap object](#).

```
groupBitmap(expr)
```

### Arguments

expr – An expression that results in UInt\* type.

### Return value

Value of the UInt64 type.

### Example

Test data:

```
UserID
1
1
2
3
```

Query:

```
SELECT groupBitmap(UserID) as num FROM t
```

Result:

```
num
3
```

## groupBitmapAnd

Calculations the AND of a bitmap column, return cardinality of type UInt64, if add suffix -State, then return [bitmap object](#).

```
groupBitmapAnd(expr)
```

### Arguments

expr – An expression that results in AggregateFunction(groupBitmap, UInt\*) type.

### Return value

Value of the UInt64 type.

### Example

```
DROP TABLE IF EXISTS bitmap_column_expr_test2;
CREATE TABLE bitmap_column_expr_test2
(
    tag_id String,
    z AggregateFunction(groupBitmap, UInt32)
)
ENGINE = MergeTree
ORDER BY tag_id;

INSERT INTO bitmap_column_expr_test2 VALUES ('tag1', bitmapBuild(cast([1,2,3,4,5,6,7,8,9,10] as Array(UInt32))));
INSERT INTO bitmap_column_expr_test2 VALUES ('tag2', bitmapBuild(cast([6,7,8,9,10,11,12,13,14,15] as Array(UInt32))));
INSERT INTO bitmap_column_expr_test2 VALUES ('tag3', bitmapBuild(cast([2,4,6,8,10,12] as Array(UInt32))));

SELECT groupBitmapAnd(z) FROM bitmap_column_expr_test2 WHERE like(tag_id, 'tag%');
└groupBitmapAnd(z)┘
    3 |
```

```
SELECT arraySort(bitmapToArray(groupBitmapAndState(z))) FROM bitmap_column_expr_test2 WHERE like(tag_id, 'tag%');
└arraySort(bitmapToArray(groupBitmapAndState(z)))┘
    [6,8,10] |
```

# groupBitmapOr

Calculations the OR of a bitmap column, return cardinality of type UInt64, if add suffix -State, then return **bitmap object**. This is equivalent to `groupBitmapMerge`.

```
groupBitmapOr(expr)
```

## Arguments

`expr` – An expression that results in `AggregateFunction(groupBitmap, UInt*)` type.

## Returned value

Value of the `UInt64` type.

## Example

```
DROP TABLE IF EXISTS bitmap_column_expr_test2;
CREATE TABLE bitmap_column_expr_test2
(
    tag_id String,
    z AggregateFunction(groupBitmap, UInt32)
)
ENGINE = MergeTree
ORDER BY tag_id;

INSERT INTO bitmap_column_expr_test2 VALUES ('tag1', bitmapBuild(cast([1,2,3,4,5,6,7,8,9,10] as Array(UInt32))));
INSERT INTO bitmap_column_expr_test2 VALUES ('tag2', bitmapBuild(cast([6,7,8,9,10,11,12,13,14,15] as Array(UInt32))));
INSERT INTO bitmap_column_expr_test2 VALUES ('tag3', bitmapBuild(cast([2,4,6,8,10,12] as Array(UInt32))));

SELECT groupBitmapOr(z) FROM bitmap_column_expr_test2 WHERE like(tag_id, 'tag%');
└groupBitmapOr(z)─
    15 ┌─────────────────┐
      └────────────────┘

SELECT arraySort(bitmapToArray(groupBitmapOrState(z))) FROM bitmap_column_expr_test2 WHERE like(tag_id, 'tag%');
└arraySort(bitmapToArray(groupBitmapOrState(z)))─
    [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15] ┌─────────────────┐
                                              └────────────────┘
```

# groupBitmapXor

Calculations the XOR of a bitmap column, return cardinality of type UInt64, if add suffix -State, then return **bitmap object**.

```
groupBitmapOr(expr)
```

## Arguments

`expr` – An expression that results in `AggregateFunction(groupBitmap, UInt*)` type.

## Returned value

Value of the `UInt64` type.

## Example

```

DROP TABLE IF EXISTS bitmap_column_expr_test2;
CREATE TABLE bitmap_column_expr_test2
(
    tag_id String,
    z AggregateFunction(groupBitmap, UInt32)
)
ENGINE = MergeTree
ORDER BY tag_id;

INSERT INTO bitmap_column_expr_test2 VALUES ('tag1', bitmapBuild(cast([1,2,3,4,5,6,7,8,9,10] as Array(UInt32))));
INSERT INTO bitmap_column_expr_test2 VALUES ('tag2', bitmapBuild(cast([6,7,8,9,10,11,12,13,14,15] as
Array(UInt32))));
INSERT INTO bitmap_column_expr_test2 VALUES ('tag3', bitmapBuild(cast([2,4,6,8,10,12] as Array(UInt32))));

SELECT groupBitmapXor(z) FROM bitmap_column_expr_test2 WHERE like(tag_id, 'tag%');
└groupBitmapXor(z)─
  10 |

SELECT arraySort(bitmapToArray(groupBitmapXorState(z))) FROM bitmap_column_expr_test2 WHERE like(tag_id,
'tag%');
└arraySort(bitmapToArray(groupBitmapXorState(z)))─
[1,3,5,6,8,10,11,13,14,15] |

```

## sumWithOverflow

Computes the sum of the numbers, using the same data type for the result as for the input parameters. If the sum exceeds the maximum value for this data type, it is calculated with overflow.

Only works for numbers.

## deltaSum

Sums the arithmetic difference between consecutive rows. If the difference is negative, it is ignored.

### Note

The underlying data must be sorted for this function to work properly. If you would like to use this function in a **materialized view**, you most likely want to use the **deltaSumTimestamp** method instead.

### Syntax

```
deltaSum(value)
```

### Arguments

- **value** — Input values, must be **Integer** or **Float** type.

### Returned value

- A gained arithmetic difference of the **Integer** or **Float** type.

### Examples

Query:

```
SELECT deltaSum(arrayJoin([1, 2, 3]));
```

Result:

```
deltaSum(arrayJoin([1, 2, 3]))  
2 |
```

Query:

```
SELECT deltaSum(arrayJoin([1, 2, 3, 0, 3, 4, 2, 3]));
```

Result:

```
deltaSum(arrayJoin([1, 2, 3, 0, 3, 4, 2, 3]))  
7 |
```

Query:

```
SELECT deltaSum(arrayJoin([2.25, 3, 4.5]));
```

Result:

```
deltaSum(arrayJoin([2.25, 3, 4.5]))  
2.25 |
```

## See Also

- [runningDifference](#)

## deltaSumTimestamp

Adds the difference between consecutive rows. If the difference is negative, it is ignored.

This function is primarily for [materialized views](#) that are ordered by some time bucket-aligned timestamp, for example, a `toStartOfMinute` bucket. Because the rows in such a materialized view will all have the same timestamp, it is impossible for them to be merged in the "right" order. This function keeps track of the timestamp of the values it's seen, so it's possible to order the states correctly during merging.

To calculate the delta sum across an ordered collection you can simply use the [deltaSum](#) function.

### Syntax

```
deltaSumTimestamp(value, timestamp)
```

### Arguments

- `value` — Input values, must be some [Integer](#) type or [Float](#) type or a [Date](#) or [DateTime](#).
- `timestamp` — The parameter for order values, must be some [Integer](#) type or [Float](#) type or a [Date](#) or [DateTime](#).

## Returned value

- Accumulated differences between consecutive values, ordered by the `timestamp` parameter.

Type: `Integer` or `Float` or `Date` or `DateTime`.

## Example

Query:

```
SELECT deltaSumTimestamp(value, timestamp)
FROM (SELECT number AS timestamp, [0, 4, 8, 3, 0, 0, 0, 1, 3, 5][number] AS value FROM numbers(1, 10));
```

Result:

```
deltaSumTimestamp(value, timestamp)---  
13 |-----
```

## sumMap

Syntax: `sumMap(key, value)` or `sumMap(Tuple(key, value))`

Totals the `value` array according to the `key` specified in the `key` array.

Passing tuple of keys and values arrays is a synonym to passing two arrays of keys and values.

The number of elements in `key` and `value` must be the same for each row that is totaled.

Returns a tuple of two arrays: keys in sorted order, and values summed for the corresponding keys.

Example:

```
CREATE TABLE sum_map(
    date Date,
    timeslot DateTime,
    statusMap Nested(
        status UInt16,
        requests UInt64
    ),
    statusMapTuple Tuple(Array(Int32), Array(Int32))
) ENGINE = Log;
INSERT INTO sum_map VALUES
('2000-01-01', '2000-01-01 00:00:00', [1, 2, 3], [10, 10, 10], ([1, 2, 3], [10, 10, 10])),
('2000-01-01', '2000-01-01 00:00:00', [3, 4, 5], [10, 10, 10], ([3, 4, 5], [10, 10, 10])),
('2000-01-01', '2000-01-01 00:01:00', [4, 5, 6], [10, 10, 10], ([4, 5, 6], [10, 10, 10])),
('2000-01-01', '2000-01-01 00:01:00', [6, 7, 8], [10, 10, 10], ([6, 7, 8], [10, 10, 10]));
```

```
SELECT
    timeslot,
    sumMap(statusMap.status, statusMap.requests),
    sumMap(statusMapTuple)
FROM sum_map
GROUP BY timeslot
```

timeslot	sumMap(statusMap.status, statusMap.requests)	sumMap(statusMapTuple)
2000-01-01 00:00:00	[1,2,3,4,5],[10,10,20,10,10]	[1,2,3,4,5],[10,10,20,10,10]
2000-01-01 00:01:00	[4,5,6,7,8],[10,10,20,10,10]	[4,5,6,7,8],[10,10,20,10,10]

## minMap

Syntax: `minMap(key, value)` or `minMap(Tuple(key, value))`

Calculates the minimum from `value` array according to the keys specified in the `key` array.

Passing a tuple of keys and value arrays is identical to passing two arrays of keys and values.

The number of elements in `key` and `value` must be the same for each row that is totaled.

Returns a tuple of two arrays: keys in sorted order, and values calculated for the corresponding keys.

Example:

```
SELECT minMap(a, b)
FROM values('a Array(Int32), b Array(Int64)', ([1, 2], [2, 2]), ([2, 3], [1, 1]))
```

```
minMap(a, b)
([1,2,3],[2,1,1]) |
```

## maxMap

Syntax: `maxMap(key, value)` or `maxMap(Tuple(key, value))`

Calculates the maximum from `value` array according to the keys specified in the `key` array.

Passing a tuple of keys and value arrays is identical to passing two arrays of keys and values.

The number of elements in `key` and `value` must be the same for each row that is totaled.

Returns a tuple of two arrays: keys and values calculated for the corresponding keys.

Example:

```
SELECT maxMap(a, b)
FROM values('a Array(Int32), b Array(Int64)', ([1, 2], [2, 2]), ([2, 3], [1, 1]))
```

```
maxMap(a, b)
([1,2,3],[2,2,1]) |
```

## sumCount

Calculates the sum of the numbers and counts the number of rows at the same time. The function is used by ClickHouse query optimizer: if there are multiple sum, count or avg functions in a query, they can be replaced to single `sumCount` function to reuse the calculations. The function is rarely needed to use explicitly.

### Syntax

```
sumCount(x)
```

### Arguments

- `x` — Input value, must be [Integer](#), [Float](#), or [Decimal](#).

## Returned value

- Tuple (`sum`, `count`), where `sum` is the sum of numbers and `count` is the number of rows with not-NULL values.

Type: [Tuple](#).

## Example

Query:

```
CREATE TABLE s_table (x Int8) Engine = Log;
INSERT INTO s_table SELECT number FROM numbers(0, 20);
INSERT INTO s_table VALUES (NULL);
SELECT sumCount(x) from s_table;
```

Result:

```
sumCount(x)
(190,20)
```

## See also

- [optimize\\_syntax\\_fuse\\_functions](#) setting.

# rankCorr

Computes a rank correlation coefficient.

## Syntax

```
rankCorr(x, y)
```

## Arguments

- `x` — Arbitrary value. [Float32](#) or [Float64](#).
- `y` — Arbitrary value. [Float32](#) or [Float64](#).

## Returned value(s)

- Returns a rank correlation coefficient of the ranks of `x` and `y`. The value of the correlation coefficient ranges from -1 to +1. If less than two arguments are passed, the function will return an exception. The value close to +1 denotes a high linear relationship, and with an increase of one random variable, the second random variable also increases. The value close to -1 denotes a high linear relationship, and with an increase of one random variable, the second random variable decreases. The value close or equal to 0 denotes no relationship between the two random variables.

Type: [Float64](#).

## Example

Query:

```
SELECT rankCorr(number, number) FROM numbers(100);
```

Result:

```
rankCorr(number, number)─  
 1 |
```

Query:

```
SELECT roundBankers(rankCorr(exp(number), sin(number)), 3) FROM numbers(100);
```

Result:

```
roundBankers(rankCorr(exp(number), sin(number)), 3)─  
 -0.037 |
```

## See Also

- [Spearman's rank correlation coefficient](#)

## sumKahan

Calculates the sum of the numbers with [Kahan compensated summation algorithm](#)

Slower than [sum](#) function.

The compensation works only for [Float](#) types.

### Syntax

```
sumKahan(x)
```

### Arguments

- $x$  — Input value, must be [Integer](#), [Float](#), or [Decimal](#).

### Returned value

- the sum of numbers, with type [Integer](#), [Float](#), or [Decimal](#) depends on type of input arguments

### Example

Query:

```
SELECT sum(0.1), sumKahan(0.1) FROM numbers(10);
```

Result:

```
sum(0.1) ─ sumKahan(0.1) ─  
 0.9999999999999999 | 1 |
```

## intervalLengthSum

Calculates the total length of union of all ranges (segments on numeric axis).

## Syntax

```
intervalLengthSum(start, end)
```

## Arguments

- **start** — The starting value of the interval. [Int32](#), [Int64](#), [UInt32](#), [UInt64](#), [Float32](#), [Float64](#), [DateTime](#) or [Date](#).
- **end** — The ending value of the interval. [Int32](#), [Int64](#), [UInt32](#), [UInt64](#), [Float32](#), [Float64](#), [DateTime](#) or [Date](#).

## Note

Arguments must be of the same data type. Otherwise, an exception will be thrown.

## Returned value

- Total length of union of all ranges (segments on numeric axis). Depending on the type of the argument, the return value may be [UInt64](#) or [Float64](#) type.

## Examples

1. Input table:

id	start	end
a	1.1	2.9
a	2.5	3.2
a	4	5

In this example, the arguments of the [Float32](#) type are used. The function returns a value of the [Float64](#) type.

Result is the sum of lengths of intervals [1.1, 3.2] (union of [1.1, 2.9] and [2.5, 3.2]) and [4, 5]

Query:

```
SELECT id, intervalLengthSum(start, end), toTypeName(intervalLengthSum(start, end)) FROM fl_interval GROUP BY id;  
ORDER BY id;
```

Result:

id	intervalLengthSum(start, end)	toTypeName(intervalLengthSum(start, end))
a	3.1	Float64

2. Input table:

id	start	end
a	2020-01-01 01:12:30	2020-01-01 02:10:10
a	2020-01-01 02:05:30	2020-01-01 02:50:31
a	2020-01-01 03:11:22	2020-01-01 03:23:31

In this example, the arguments of the DateTime type are used. The function returns a value in seconds.

Query:

```
SELECT id, intervalLengthSum(start, end), toTypeName(intervalLengthSum(start, end)) FROM dt_interval GROUP BY id  
ORDER BY id;
```

Result:

id	intervalLengthSum(start, end)	toTypeName(intervalLengthSum(start, end))
a	6610   UInt64	

3. Input table:

id	start	end
a	2020-01-01	2020-01-04
a	2020-01-12	2020-01-18

In this example, the arguments of the Date type are used. The function returns a value in days.

Query:

```
SELECT id, intervalLengthSum(start, end), toTypeName(intervalLengthSum(start, end)) FROM date_interval GROUP BY id ORDER BY id;
```

Result:

id	intervalLengthSum(start, end)	toTypeName(intervalLengthSum(start, end))
a	9   UInt64	

## skewPop

Computes the **skewness** of a sequence.

```
skewPop(expr)
```

### Arguments

**expr** — **Expression** returning a number.

### Returned value

The skewness of the given distribution. Type — **Float64**

### Example

```
SELECT skewPop(value) FROM series_with_value_column;
```

## skewSamp

Computes the **sample skewness** of a sequence.

It represents an unbiased estimate of the skewness of a random variable if passed values form its sample.

```
skewSamp(expr)
```

## Arguments

`expr` — **Expression** returning a number.

## Returned value

The skewness of the given distribution. Type — **Float64**. If `n <= 1` (`n` is the size of the sample), then the function returns `nan`.

## Example

```
SELECT skewSamp(value) FROM series_with_value_column;
```

# kurtPop

Computes the **kurtosis** of a sequence.

```
kurtPop(expr)
```

## Arguments

`expr` — **Expression** returning a number.

## Returned value

The kurtosis of the given distribution. Type — **Float64**

## Example

```
SELECT kurtPop(value) FROM series_with_value_column;
```

# kurtSamp

Computes the **sample kurtosis** of a sequence.

It represents an unbiased estimate of the kurtosis of a random variable if passed values form its sample.

```
kurtSamp(expr)
```

## Arguments

`expr` — **Expression** returning a number.

## Returned value

The kurtosis of the given distribution. Type — **Float64**. If `n <= 1` (`n` is a size of the sample), then the function returns `nan`.

## Example

```
SELECT kurtSamp(value) FROM series_with_value_column;
```

# uniq

Calculates the approximate number of different values of the argument.

```
uniq(x[, ...])
```

## Arguments

The function takes a variable number of parameters. Parameters can be `Tuple`, `Array`, `Date`, `DateTime`, `String`, or numeric types.

## Returned value

- A `UInt64`-type number.

## Implementation details

Function:

- Calculates a hash for all parameters in the aggregate, then uses it in calculations.
- Uses an adaptive sampling algorithm. For the calculation state, the function uses a sample of element hash values up to 65536.

This algorithm is very accurate and very efficient on the CPU. When the query contains several of these functions, using ``uniq`` is almost as fast as using other aggregate functions.

- Provides the result deterministically (it does not depend on the query processing order).

We recommend using this function in almost all scenarios.

## See Also

- [uniqCombined](#)
- [uniqCombined64](#)
- [uniqHLL12](#)
- [uniqExact](#)
- [uniqTheta](#)

# uniqExact

Calculates the exact number of different argument values.

```
uniqExact(x[, ...])
```

Use the `uniqExact` function if you absolutely need an exact result. Otherwise use the `uniq` function.

The `uniqExact` function uses more memory than `uniq`, because the size of the state has unbounded growth as the number of different values increases.

## Arguments

The function takes a variable number of parameters. Parameters can be `Tuple`, `Array`, `Date`, `DateTime`, `String`, or numeric types.

## See Also

- [uniq](#)
- [uniqCombined](#)
- [uniqHLL12](#)
- [uniqTheta](#)

# uniqCombined

Calculates the approximate number of different argument values.

```
uniqCombined(HLL_precision)(x[, ...])
```

The `uniqCombined` function is a good choice for calculating the number of different values.

## Arguments

The function takes a variable number of parameters. Parameters can be `Tuple`, `Array`, `Date`, `DateTime`, `String`, or numeric types.

`HLL_precision` is the base-2 logarithm of the number of cells in [HyperLogLog](#). Optional, you can use the function as `uniqCombined(x[, ...])`. The default value for `HLL_precision` is 17, which is effectively 96 KiB of space ( $2^{17}$  cells, 6 bits each).

## Returned value

- A number `UInt64`-type number.

## Implementation details

Function:

- Calculates a hash (64-bit hash for `String` and 32-bit otherwise) for all parameters in the aggregate, then uses it in calculations.
- Uses a combination of three algorithms: array, hash table, and [HyperLogLog](#) with an error correction table.

For a small number of distinct elements, an array is used. When the set size is larger, a hash table is used. For a larger number of elements, [HyperLogLog](#) is used, which will occupy a fixed amount of memory.

- Provides the result deterministically (it does not depend on the query processing order).

## Note

Since it uses 32-bit hash for non-String type, the result will have very high error for cardinalities significantly larger than `UINT_MAX` (error will raise quickly after a few tens of billions of distinct values), hence in this case you should use **uniqCombined64**

Compared to the [uniq](#) function, the [uniqCombined](#):

- Consumes several times less memory.
- Calculates with several times higher accuracy.
- Usually has slightly lower performance. In some scenarios, [uniqCombined](#) can perform better than [uniq](#), for example, with distributed queries that transmit a large number of aggregation states over the network.

## See Also

- [uniq](#)
- [uniqCombined64](#)
- [uniqHLL12](#)
- [uniqExact](#)
- [uniqTheta](#)

## uniqCombined64

Same as [uniqCombined](#), but uses 64-bit hash for all data types.

## uniqHLL12

Calculates the approximate number of different argument values, using the [HyperLogLog](#) algorithm.

```
uniqHLL12(x[, ...])
```

### Arguments

The function takes a variable number of parameters. Parameters can be `Tuple`, `Array`, `Date`, `DateTime`, `String`, or numeric types.

### Returned value

- A `UInt64`-type number.

### Implementation details

Function:

- Calculates a hash for all parameters in the aggregate, then uses it in calculations.
- Uses the HyperLogLog algorithm to approximate the number of different argument values.

$2^{12}$  5-bit cells are used. The size of the state is slightly more than 2.5 KB. The result is not very accurate (up to ~10% error) for small data sets (<10K elements). However, the result is fairly accurate for high-cardinality data sets (10K-100M), with a maximum error of ~1.6%. Starting from 100M, the estimation error increases, and the function will return very inaccurate results for data sets with extremely high cardinality (1B+ elements).

- Provides the determinate result (it does not depend on the query processing order).

We do not recommend using this function. In most cases, use the [uniq](#) or [uniqCombined](#) function.

## See Also

- [uniq](#)
- [uniqCombined](#)
- [uniqExact](#)
- [uniqTheta](#)

# uniqTheta

Calculates the approximate number of different argument values, using the [Theta Sketch Framework](#).

```
uniqTheta(x[, ...])
```

## Arguments

The function takes a variable number of parameters. Parameters can be [Tuple](#), [Array](#), [Date](#), [DateTime](#), [String](#), or numeric types.

## Returned value

- A [UInt64](#)-type number.

## Implementation details

Function:

- Calculates a hash for all parameters in the aggregate, then uses it in calculations.
- Uses the [KMV](#) algorithm to approximate the number of different argument values.

4096( $2^{12}$ ) 64-bit sketch are used. The size of the state is about 41 KB.

- The relative error is 3.125% (95% confidence), see the [relative error table](#) for detail.

## See Also

- [uniq](#)
- [uniqCombined](#)
- [uniqCombined64](#)
- [uniqHLL12](#)
- [uniqExact](#)

# quantile

Computes an approximate [quantile](#) of a numeric data sequence.

This function applies [reservoir sampling](#) with a reservoir size up to 8192 and a random number generator for sampling. The result is non-deterministic. To get an exact quantile, use the [quantileExact](#) function.

When using multiple `quantile*` functions with different levels in a query, the internal states are not combined (that is, the query works less efficiently than it could). In this case, use the [quantiles](#) function.

## Syntax

```
quantile(level)(expr)
```

Alias: `median`.

## Arguments

- `level` — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a `level` value in the range of [0.01, 0.99]. Default value: 0.5. At `level=0.5` the function calculates [median](#).
- `expr` — Expression over the column values resulting in numeric [data types](#), [Date](#) or [DateTime](#).

## Returned value

- Approximate quantile of the specified level.

Type:

- [Float64](#) for numeric data type input.
- [Date](#) if input values have the `Date` type.
- [DateTime](#) if input values have the `DateTime` type.

## Example

Input table:

val
1
1
2
3

Query:

```
SELECT quantile(val) FROM t
```

Result:

quantile(val)
1.5

## See Also

- [median](#)
- [quantiles](#)

# quantiles Functions

## quantiles

Syntax: `quantiles(level1, level2, ...)(x)`

All the quantile functions also have corresponding quantiles functions: `quantiles`, `quantilesDeterministic`, `quantilesTiming`, `quantilesTimingWeighted`, `quantilesExact`, `quantilesExactWeighted`, `quantilesTDigest`, `quantilesBFloat16`. These functions calculate all the quantiles of the listed levels in one pass, and return an array of the resulting values.

## quantilesExactExclusive

Exactly computes the `quantiles` of a numeric data sequence.

To get exact value, all the passed values are combined into an array, which is then partially sorted. Therefore, the function consumes  $O(n)$  memory, where  $n$  is a number of values that were passed. However, for a small number of values, the function is very effective.

This function is equivalent to `PERCENTILE.EXC` Excel function, ([type R6](#)).

Works more efficiently with sets of levels than `quantileExactExclusive`.

### Syntax

```
quantilesExactExclusive(level1, level2, ...)(expr)
```

### Arguments

- `expr` — Expression over the column values resulting in numeric [data types](#), [Date](#) or [DateTime](#).

### Parameters

- `level` — Levels of quantiles. Possible values: (0, 1) — bounds not included. [Float](#).

### Returned value

- [Array](#) of quantiles of the specified levels.

Type of array values:

- [Float64](#) for numeric data type input.
- [Date](#) if input values have the [Date](#) type.
- [DateTime](#) if input values have the [DateTime](#) type.

### Example

Query:

```
CREATE TABLE num AS numbers(1000);
SELECT quantilesExactExclusive(0.25, 0.5, 0.75, 0.9, 0.95, 0.99, 0.999)(x) FROM (SELECT number AS x FROM num);
```

Result:

```
quantilesExactExclusive(0.25, 0.5, 0.75, 0.9, 0.95, 0.99, 0.999)(x) └
```

```
[249.25,499.5,749.75,899.9,949.949999999999,989.99,998.999]
```

```
|
```

```
└
```

## quantilesExactInclusive

Exactly computes the [quantiles](#) of a numeric data sequence.

To get exact value, all the passed values are combined into an array, which is then partially sorted.

Therefore, the function consumes  $O(n)$  memory, where  $n$  is a number of values that were passed. However, for a small number of values, the function is very effective.

This function is equivalent to [PERCENTILE.INC](#) Excel function, ([type R7](#)).

Works more efficiently with sets of levels than [quantileExactInclusive](#).

### Syntax

```
quantilesExactInclusive(level1, level2, ...)(expr)
```

### Arguments

- `expr` — Expression over the column values resulting in numeric [data types](#), [Date](#) or [DateTime](#).

### Parameters

- `level` — Levels of quantiles. Possible values: [0, 1] — bounds included. [Float](#).

### Returned value

- [Array](#) of quantiles of the specified levels.

Type of array values:

- [Float64](#) for numeric data type input.
- [Date](#) if input values have the [Date](#) type.
- [DateTime](#) if input values have the [DateTime](#) type.

### Example

Query:

```
CREATE TABLE num AS numbers(1000);
```

```
SELECT quantilesExactInclusive(0.25, 0.5, 0.75, 0.9, 0.95, 0.99, 0.999)(x) FROM (SELECT number AS x FROM num);
```

Result:

```
quantilesExactInclusive(0.25, 0.5, 0.75, 0.9, 0.95, 0.99, 0.999)(x) └
```

```
[249.75,499.5,749.25,899.1,949.05,989.01,998.001]
```

```
|
```

```
└
```

## quantileExact Functions

### quantileExact

Exactly computes the [quantile](#) of a numeric data sequence.

To get exact value, all the passed values are combined into an array, which is then partially sorted.

Therefore, the function consumes  $O(n)$  memory, where  $n$  is a number of values that were passed. However, for a small number of values, the function is very effective.

When using multiple `quantile*` functions with different levels in a query, the internal states are not combined (that is, the query works less efficiently than it could). In this case, use the [quantiles](#) function.

## Syntax

```
quantileExact(level)(expr)
```

Alias: `medianExact`.

## Arguments

- `level` — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a `level` value in the range of `[0.01, 0.99]`. Default value: 0.5. At `level=0.5` the function calculates [median](#).
- `expr` — Expression over the column values resulting in numeric [data types](#), [Date](#) or [DateTime](#).

## Returned value

- Quantile of the specified level.

Type:

- [Float64](#) for numeric data type input.
- [Date](#) if input values have the `Date` type.
- [DateTime](#) if input values have the `DateTime` type.

## Example

Query:

```
SELECT quantileExact(number) FROM numbers(10)
```

Result:

```
quantileExact(number)─  
5 |
```

## quantileExactLow

Similar to `quantileExact`, this computes the exact [quantile](#) of a numeric data sequence.

To get the exact value, all the passed values are combined into an array, which is then fully sorted. The sorting [algorithm's](#) complexity is  $O(N \cdot \log(N))$ , where  $N = \text{std}::\text{distance}(\text{first}, \text{last})$  comparisons.

The return value depends on the quantile level and the number of elements in the selection, i.e. if the level is 0.5, then the function returns the lower median value for an even number of elements and the middle median value for an odd number of elements. Median is calculated similarly to the [median\\_low](#) implementation which is used in python.

For all other levels, the element at the index corresponding to the value of `level * size_of_array` is returned.  
For example:

```
SELECT quantileExactLow(0.1)(number) FROM numbers(10)
```

```
└─quantileExactLow(0.1)(number)─
```

```
  1 |
```

When using multiple `quantile*` functions with different levels in a query, the internal states are not combined (that is, the query works less efficiently than it could). In this case, use the `quantiles` function.

## Syntax

```
quantileExactLow(level)(expr)
```

Alias: `medianExactLow`.

## Arguments

- `level` — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a `level` value in the range of [0.01, 0.99]. Default value: 0.5. At `level=0.5` the function calculates `median`.
- `expr` — Expression over the column values resulting in numeric `data types`, `Date` or `DateTime`.

## Returned value

- Quantile of the specified level.

Type:

- `Float64` for numeric data type input.
- `Date` if input values have the `Date` type.
- `DateTime` if input values have the `DateTime` type.

## Example

Query:

```
SELECT quantileExactLow(number) FROM numbers(10)
```

Result:

```
└─quantileExactLow(number)─
```

```
  4 |
```

## quantileExactHigh

Similar to `quantileExact`, this computes the exact `quantile` of a numeric data sequence.

All the passed values are combined into an array, which is then fully sorted, to get the exact value. The sorting `algorithm's` complexity is  $O(N \cdot \log(N))$ , where  $N = \text{std}::\text{distance}(\text{first}, \text{last})$  comparisons.

The return value depends on the quantile level and the number of elements in the selection, i.e. if the level is 0.5, then the function returns the higher median value for an even number of elements and the middle median value for an odd number of elements. Median is calculated similarly to the [median\\_high](#) implementation which is used in python. For all other levels, the element at the index corresponding to the value of `level * size_of_array` is returned.

This implementation behaves exactly similar to the current `quantileExact` implementation.

When using multiple `quantile*` functions with different levels in a query, the internal states are not combined (that is, the query works less efficiently than it could). In this case, use the [quantiles](#) function.

## Syntax

```
quantileExactHigh(level)(expr)
```

Alias: `medianExactHigh`.

## Arguments

- `level` — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a `level` value in the range of `[0.01, 0.99]`. Default value: 0.5. At `level=0.5` the function calculates [median](#).
- `expr` — Expression over the column values resulting in numeric [data types](#), [Date](#) or [DateTime](#).

## Returned value

- Quantile of the specified level.

Type:

- [Float64](#) for numeric data type input.
- [Date](#) if input values have the `Date` type.
- [DateTime](#) if input values have the `DateTime` type.

## Example

Query:

```
SELECT quantileExactHigh(number) FROM numbers(10)
```

Result:

```
quantileExactHigh(number)─  
      5 |
```

# quantileExactExclusive

Exactly computes the [quantile](#) of a numeric data sequence.

To get exact value, all the passed values are combined into an array, which is then partially sorted. Therefore, the function consumes  $O(n)$  memory, where  $n$  is a number of values that were passed. However, for a small number of values, the function is very effective.

This function is equivalent to [PERCENTILE.EXC](#) Excel function, ([type R6](#)).

When using multiple `quantileExactExclusive` functions with different levels in a query, the internal states are not combined (that is, the query works less efficiently than it could). In this case, use the `quantilesExactExclusive` function.

## Syntax

```
quantileExactExclusive(level)(expr)
```

## Arguments

- `expr` — Expression over the column values resulting in numeric [data types](#), [Date](#) or [DateTime](#).

## Parameters

- `level` — Level of quantile. Optional. Possible values: (0, 1) — bounds not included. Default value: 0.5. At `level=0.5` the function calculates [median](#). [Float](#).

## Returned value

- Quantile of the specified level.

Type:

- [Float64](#) for numeric data type input.
- [Date](#) if input values have the `Date` type.
- [DateTime](#) if input values have the `DateTime` type.

## Example

Query:

```
CREATE TABLE num AS numbers(1000);
SELECT quantileExactExclusive(0.6)(x) FROM (SELECT number AS x FROM num);
```

Result:

```
quantileExactExclusive(0.6)(x)─
  599.6 |
```

## quantileExactInclusive

Exactly computes the [quantile](#) of a numeric data sequence.

To get exact value, all the passed values are combined into an array, which is then partially sorted. Therefore, the function consumes  $O(n)$  memory, where  $n$  is a number of values that were passed. However, for a small number of values, the function is very effective.

This function is equivalent to [PERCENTILE.INC](#) Excel function, ([type R7](#)).

When using multiple `quantileExactInclusive` functions with different levels in a query, the internal states are not combined (that is, the query works less efficiently than it could). In this case, use the `quantilesExactInclusive` function.

## Syntax

```
quantileExactInclusive(level)(expr)
```

## Arguments

- `expr` — Expression over the column values resulting in numeric [data types](#), [Date](#) or [DateTime](#).

## Parameters

- `level` — Level of quantile. Optional. Possible values: [0, 1] — bounds included. Default value: 0.5. At `level=0.5` the function calculates [median](#). [Float](#).

## Returned value

- Quantile of the specified level.

Type:

- [Float64](#) for numeric data type input.
- [Date](#) if input values have the [Date](#) type.
- [DateTime](#) if input values have the [DateTime](#) type.

## Example

Query:

```
CREATE TABLE num AS numbers(1000);
SELECT quantileExactInclusive(0.6)(x) FROM (SELECT number AS x FROM num);
```

Result:

```
quantileExactInclusive(0.6)(x)
599.4 |
```

## See Also

- [median](#)
- [quantiles](#)

# quantileExactWeighted

Exactly computes the [quantile](#) of a numeric data sequence, taking into account the weight of each element.

To get exact value, all the passed values are combined into an array, which is then partially sorted. Each value is counted with its weight, as if it is present `weight` times. A hash table is used in the algorithm. Because of this, if the passed values are frequently repeated, the function consumes less RAM than [quantileExact](#). You can use this function instead of [quantileExact](#) and specify the weight 1.

When using multiple `quantile*` functions with different levels in a query, the internal states are not combined (that is, the query works less efficiently than it could). In this case, use the [quantiles](#) function.

## Syntax

```
quantileExactWeighted(level)(expr, weight)
```

Alias: `medianExactWeighted`.

## Arguments

- `level` — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a `level` value in the range of [0.01, 0.99]. Default value: 0.5. At `level=0.5` the function calculates `median`.
- `expr` — Expression over the column values resulting in numeric [data types](#), [Date](#) or [DateTime](#).
- `weight` — Column with weights of sequence members. Weight is a number of value occurrences.

## Returned value

- Quantile of the specified level.

Type:

- [Float64](#) for numeric data type input.
- [Date](#) if input values have the `Date` type.
- [DateTime](#) if input values have the `DateTime` type.

## Example

Input table:

n	val
0	3
1	2
2	1
5	4

Query:

```
SELECT quantileExactWeighted(n, val) FROM t
```

Result:

```
quantileExactWeighted(n, val)
  1 |
```

## See Also

- [median](#)
- [quantiles](#)

# quantileTiming

With the determined precision computes the [quantile](#) of a numeric data sequence.

The result is deterministic (it does not depend on the query processing order). The function is optimized for working with sequences which describe distributions like loading web pages times or backend response times.

When using multiple `quantile*` functions with different levels in a query, the internal states are not combined (that is, the query works less efficiently than it could). In this case, use the `quantiles` function.

## Syntax

```
quantileTiming(level)(expr)
```

Alias: `medianTiming`.

## Arguments

- `level` — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a `level` value in the range of `[0.01, 0.99]`. Default value: 0.5. At `level=0.5` the function calculates `median`.
- `expr` — **Expression** over a column values returning a `Float*`-type number.
  - If negative values are passed to the function, the behavior is undefined.
  - If the value is greater than 30,000 (a page loading time of more than 30 seconds), it is assumed to be 30,000.

## Accuracy

The calculation is accurate if:

- Total number of values does not exceed 5670.
- Total number of values exceeds 5670, but the page loading time is less than 1024ms.

Otherwise, the result of the calculation is rounded to the nearest multiple of 16 ms.

## Note

For calculating page loading time quantiles, this function is more effective and accurate than `quantile`.

## Returned value

- Quantile of the specified level.

Type: `Float32`.

## Note

If no values are passed to the function (when using `quantileTimingIf`), `NaN` is returned. The purpose of this is to differentiate these cases from cases that result in zero. See **ORDER BY clause** for notes on sorting `NaN` values.

## Example

Input table:

```
└── response_time ──  
    72 |  
    112 |  
    126 |  
    145 |  
    104 |  
    242 |  
    313 |  
    168 |  
    108 |
```

Query:

```
SELECT quantileTiming(response_time) FROM t
```

Result:

```
└── quantileTiming(response_time) ──  
    126 |
```

## See Also

- [median](#)
- [quantiles](#)

## quantileTimingWeighted

With the determined precision computes the [quantile](#) of a numeric data sequence according to the weight of each sequence member.

The result is deterministic (it does not depend on the query processing order). The function is optimized for working with sequences which describe distributions like loading web pages times or backend response times.

When using multiple `quantile*` functions with different levels in a query, the internal states are not combined (that is, the query works less efficiently than it could). In this case, use the [quantiles](#) function.

## Syntax

```
quantileTimingWeighted(level)(expr, weight)
```

Alias: `medianTimingWeighted`.

## Arguments

- **level** — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a `level` value in the range of [0.01, 0.99]. Default value: 0.5. At `level=0.5` the function calculates [median](#).
- **expr** — [Expression](#) over a column values returning a [Float\\*](#)-type number.

- If negative values are passed to the function, the behavior is undefined.
- If the value is greater than 30,000 (a page loading time of more than 30 seconds), it is assumed to be 30,000.

- `weight` — Column with weights of sequence elements. Weight is a number of value occurrences.

## Accuracy

The calculation is accurate if:

- Total number of values does not exceed 5670.
- Total number of values exceeds 5670, but the page loading time is less than 1024ms.

Otherwise, the result of the calculation is rounded to the nearest multiple of 16 ms.

## Note

For calculating page loading time quantiles, this function is more effective and accurate than [quantile](#).

## Returned value

- Quantile of the specified level.

Type: `Float32`.

## Note

If no values are passed to the function (when using `quantileTimingIf`), **NaN** is returned. The purpose of this is to differentiate these cases from cases that result in zero. See [ORDER BY clause](#) for notes on sorting **NaN** values.

## Example

Input table:

response_time	weight
68	1
104	2
112	3
126	2
138	1
162	1

Query:

```
SELECT quantileTimingWeighted(response_time, weight) FROM t
```

Result:

```
quantileTimingWeighted(response_time, weight)
112
```

# quantilesTimingWeighted

Same as `quantileTimingWeighted`, but accept multiple parameters with quantile levels and return an Array filled with many values of that quantiles.

## Example

Input table:

response_time	weight
68	1
104	2
112	3
126	2
138	1
162	1

Query:

```
SELECT quantilesTimingWeighted(0.5, 0.99)(response_time, weight) FROM t
```

Result:

```
quantilesTimingWeighted(0.5, 0.99)(response_time, weight)
[112,162]
```

## See Also

- [median](#)
- [quantiles](#)

# quantileDeterministic

Computes an approximate [quantile](#) of a numeric data sequence.

This function applies [reservoir sampling](#) with a reservoir size up to 8192 and deterministic algorithm of sampling. The result is deterministic. To get an exact quantile, use the [quantileExact](#) function.

When using multiple `quantile*` functions with different levels in a query, the internal states are not combined (that is, the query works less efficiently than it could). In this case, use the [quantiles](#) function.

## Syntax

```
quantileDeterministic(level)(expr, determinator)
```

Alias: `medianDeterministic`.

## Arguments

- `level` — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a `level` value in the range of `[0.01, 0.99]`. Default value: 0.5. At `level=0.5` the function calculates [median](#).
- `expr` — Expression over the column values resulting in numeric [data types](#), [Date](#) or [DateTime](#).

- **determinator** — Number whose hash is used instead of a random number generator in the reservoir sampling algorithm to make the result of sampling deterministic. As a determinator you can use any deterministic positive number, for example, a user id or an event id. If the same determinator value occurs too often, the function works incorrectly.

## Returned value

- Approximate quantile of the specified level.

Type:

- **Float64** for numeric data type input.
- **Date** if input values have the **Date** type.
- **DateTime** if input values have the **DateTime** type.

## Example

Input table:

val
1
1
2
3

Query:

```
SELECT quantileDeterministic(val, 1) FROM t
```

Result:

```
quantileDeterministic(val, 1)─  
1.5 |
```

## See Also

- [median](#)
- [quantiles](#)

# quantileTDigest

Computes an approximate **quantile** of a numeric data sequence using the **t-digest** algorithm.

Memory consumption is  $\log(n)$ , where  $n$  is a number of values. The result depends on the order of running the query, and is nondeterministic.

The performance of the function is lower than performance of **quantile** or **quantileTiming**. In terms of the ratio of State size to precision, this function is much better than **quantile**.

When using multiple **quantile\*** functions with different levels in a query, the internal states are not combined (that is, the query works less efficiently than it could). In this case, use the **quantiles** function.

## Syntax

```
quantileTDigest(level)(expr)
```

Alias: `medianTDigest`.

## Arguments

- `level` — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a `level` value in the range of [0.01, 0.99]. Default value: 0.5. At `level=0.5` the function calculates `median`.
- `expr` — Expression over the column values resulting in numeric [data types](#), [Date](#) or [DateTime](#).

## Returned value

- Approximate quantile of the specified level.

Type:

- [Float64](#) for numeric data type input.
- [Date](#) if input values have the `Date` type.
- [DateTime](#) if input values have the `DateTime` type.

## Example

Query:

```
SELECT quantileTDigest(number) FROM numbers(10)
```

Result:

```
quantileTDigest(number)─  
4.5 |
```

## See Also

- [median](#)
- [quantiles](#)

# quantileTDigestWeighted

Computes an approximate [quantile](#) of a numeric data sequence using the [t-digest](#) algorithm. The function takes into account the weight of each sequence member. The maximum error is 1%. Memory consumption is  $\log(n)$ , where `n` is a number of values.

The performance of the function is lower than performance of [quantile](#) or [quantileTiming](#). In terms of the ratio of State size to precision, this function is much better than [quantile](#).

The result depends on the order of running the query, and is nondeterministic.

When using multiple `quantile*` functions with different levels in a query, the internal states are not combined (that is, the query works less efficiently than it could). In this case, use the [quantiles](#) function.

## Note

Using `quantileTDigestWeighted` is not recommended for tiny data sets and can lead to significant error. In this case, consider possibility of using `quantileTDigest` instead.

## Syntax

```
quantileTDigestWeighted(level)(expr, weight)
```

Alias: `medianTDigestWeighted`.

## Arguments

- `level` — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a `level` value in the range of [0.01, 0.99]. Default value: 0.5. At `level=0.5` the function calculates `median`.
- `expr` — Expression over the column values resulting in numeric [data types](#), [Date](#) or [DateTime](#).
- `weight` — Column with weights of sequence elements. Weight is a number of value occurrences.

## Returned value

- Approximate quantile of the specified level.

Type:

- [Float64](#) for numeric data type input.
- [Date](#) if input values have the `Date` type.
- [DateTime](#) if input values have the `DateTime` type.

## Example

Query:

```
SELECT quantileTDigestWeighted(number, 1) FROM numbers(10)
```

Result:

```
quantileTDigestWeighted(number, 1)─  
4.5 |
```

## See Also

- [median](#)
- [quantiles](#)

## quantileBFloat16

Computes an approximate `quantile` of a sample consisting of `bfloat16` numbers. `bfloat16` is a floating-point data type with 1 sign bit, 8 exponent bits and 7 fraction bits.

The function converts input values to 32-bit floats and takes the most significant 16 bits. Then it calculates `bfloat16` quantile value and converts the result to a 64-bit float by appending zero bits.

The function is a fast quantile estimator with a relative error no more than 0.390625%.

## Syntax

```
quantileBFloat16[(level)](#sql-reference-aggregate-functions-reference-expr)
```

Alias: medianBFloat16

## Arguments

- `expr` — Column with numeric data. [Integer](#), [Float](#).

## Parameters

- `level` — Level of quantile. Optional. Possible values are in the range from 0 to 1. Default value: 0.5. [Float](#).

## Returned value

- Approximate quantile of the specified level.

Type: [Float64](#).

## Example

Input table has an integer and a float columns:

a	b
1	1.001
2	1.002
3	1.003
4	1.004

Query to calculate 0.75-quantile (third quartile):

```
SELECT quantileBFloat16(0.75)(a), quantileBFloat16(0.75)(b) FROM example_table;
```

Result:

quantileBFloat16(0.75)(a)	quantileBFloat16(0.75)(b)
3	1

Note that all floating point values in the example are truncated to 1.0 when converting to `bfloat16`.

## quantileBFloat16Weighted

Like `quantileBFloat16` but takes into account the weight of each sequence member.

## See Also

- [median](#)
- [quantiles](#)

## median

The `median*` functions are the aliases for the corresponding `quantile*` functions. They calculate median of a numeric data sample.

Functions:

- `median` — Alias for `quantile`.
- `medianDeterministic` — Alias for `quantileDeterministic`.
- `medianExact` — Alias for `quantileExact`.
- `medianExactWeighted` — Alias for `quantileExactWeighted`.
- `medianTiming` — Alias for `quantileTiming`.
- `medianTimingWeighted` — Alias for `quantileTimingWeighted`.
- `medianTDigest` — Alias for `quantileTDigest`.
- `medianTDigestWeighted` — Alias for `quantileTDigestWeighted`.
- `medianBFloat16` — Alias for `quantileBFloat16`.

## Example

Input table:

val
1
1
2
3

Query:

```
SELECT medianDeterministic(val, 1) FROM t;
```

Result:

```
medianDeterministic(val, 1)  
1.5 |
```

## simpleLinearRegression

Performs simple (unidimensional) linear regression.

```
simpleLinearRegression(x, y)
```

Parameters:

- `x` — Column with dependent variable values.
- `y` — Column with explanatory variable values.

Returned values:

Constants (`a`, `b`) of the resulting line  $y = a*x + b$ .

## Examples

```
SELECT arrayReduce('simpleLinearRegression', [0, 1, 2, 3], [0, 1, 2, 3])
```

```
arrayReduce('simpleLinearRegression', [0, 1, 2, 3], [0, 1, 2, 3])—  
(1,0)
```

```
SELECT arrayReduce('simpleLinearRegression', [0, 1, 2, 3], [3, 4, 5, 6])
```

```
arrayReduce('simpleLinearRegression', [0, 1, 2, 3], [3, 4, 5, 6])—  
(1,3)
```

## stochasticLinearRegression

This function implements stochastic linear regression. It supports custom parameters for learning rate, L2 regularization coefficient, mini-batch size and has few methods for updating weights ([Adam](#) (used by default), [simple SGD](#), [Momentum](#), [Nesterov](#)).

### Parameters

There are 4 customizable parameters. They are passed to the function sequentially, but there is no need to pass all four - default values will be used, however good model required some parameter tuning.

```
stochasticLinearRegression(1.0, 1.0, 10, 'SGD')
```

1. learning rate is the coefficient on step length, when gradient descent step is performed. Too big learning rate may cause infinite weights of the model. Default is 0.00001.
2. L2 regularization coefficient which may help to prevent overfitting. Default is 0.1.
3. mini-batch size sets the number of elements, which gradients will be computed and summed to perform one step of gradient descent. Pure stochastic descent uses one element, however having small batches (about 10 elements) make gradient steps more stable. Default is 15.
4. method for updating weights, they are: Adam (by default), SGD, Momentum, Nesterov. Momentum and Nesterov require little bit more computations and memory, however they happen to be useful in terms of speed of convergance and stability of stochastic gradient methods.

### Usage

`stochasticLinearRegression` is used in two steps: fitting the model and predicting on new data. In order to fit the model and save its state for later usage we use `-State` combinator, which basically saves the state (model weights, etc).

To predict we use function [evalMLMethod](#), which takes a state as an argument as well as features to predict on.

#### 1. Fitting

Such query may be used.

```

CREATE TABLE IF NOT EXISTS train_data
(
    param1 Float64,
    param2 Float64,
    target Float64
) ENGINE = Memory;

CREATE TABLE your_model ENGINE = Memory AS SELECT
stochasticLinearRegressionState(0.1, 0.0, 5, 'SGD')(target, param1, param2)
AS state FROM train_data;

```

Here we also need to insert data into `train_data` table. The number of parameters is not fixed, it depends only on number of arguments, passed into `linearRegressionState`. They all must be numeric values. Note that the column with target value(which we would like to learn to predict) is inserted as the first argument.

## 2. Predicting

After saving a state into the table, we may use it multiple times for prediction, or even merge with other states and create new even better models.

```

WITH (SELECT state FROM your_model) AS model SELECT
evalMLMethod(model, param1, param2) FROM test_data

```

The query will return a column of predicted values. Note that first argument of `evalMLMethod` is `AggregateFunctionState` object, next are columns of features.

`test_data` is a table like `train_data` but may not contain target value.

## Notes

1. To merge two models user may create such query:

```
sql SELECT state1 + state2 FROM your_models
```

where `your_models` table contains both models. This query will return new `AggregateFunctionState` object.

2. User may fetch weights of the created model for its own purposes without saving the model if no-State combinator is used.

```
sql SELECT stochasticLinearRegression(0.01)(target, param1, param2) FROM train_data
```

Such query will fit the model and return its weights - first are weights, which correspond to the parameters of the model, the last one is bias. So in the example above the query will return a column with 3 values.

## See Also

- [stochasticLogisticRegression](#)
- [Difference between linear and logistic regressions](#)

## stochasticLogisticRegression

This function implements stochastic logistic regression. It can be used for binary classification problem, supports the same custom parameters as `stochasticLinearRegression` and works the same way.

## Parameters

Parameters are exactly the same as in `stochasticLinearRegression`: learning rate, l2 regularization coefficient, mini-batch size, method for updating weights. For more information see [parameters](#).

```
stochasticLogisticRegression(1.0, 1.0, 10, 'SGD')
```

## 1. Fitting

See the `Fitting` section in the [stochasticLinearRegression](#stochasticlinearregression-usage-fitting) description.

Predicted labels have to be in  $[-1, 1]$ .

## 2. Predicting

Using saved state we can predict probability of object having label `1`.

```
```sql
WITH (SELECT state FROM your_model) AS model SELECT
evalMLMethod(model, param1, param2) FROM test_data
```
```

The query will return a column of probabilities. Note that first argument of `evalMLMethod` is `AggregateFunctionState` object, next are columns of features.

We can also set a bound of probability, which assigns elements to different labels.

```
```sql
SELECT ans < 1.1 AND ans > 0.5 FROM
(WITH (SELECT state FROM your_model) AS model SELECT
evalMLMethod(model, param1, param2) AS ans FROM test_data)
```
```

Then the result will be labels.

`test\_data` is a table like `train\_data` but may not contain target value.

## See Also

- [stochasticLinearRegression](#)
- [Difference between linear and logistic regressions.](#)

# categoricalInformationValue

Calculates the value of  $(P(\text{tag} = 1) - P(\text{tag} = 0))(\log(P(\text{tag} = 1)) - \log(P(\text{tag} = 0)))$  for each category.

```
categoricalInformationValue(category1, category2, ..., tag)
```

The result indicates how a discrete (categorical) feature [category1, category2, ...] contribute to a learning model which predicting the value of tag.

# studentTTest

Applies Student's t-test to samples from two populations.

## Syntax

```
studentTTest(sample_data, sample_index)
```

Values of both samples are in the `sample_data` column. If `sample_index` equals to 0 then the value in that row belongs to the sample from the first population. Otherwise it belongs to the sample from the second population.

The null hypothesis is that means of populations are equal. Normal distribution with equal variances is assumed.

## Arguments

- `sample_data` — Sample data. [Integer](#), [Float](#) or [Decimal](#).
- `sample_index` — Sample index. [Integer](#).

## Returned values

[Tuple](#) with two elements:

- calculated t-statistic. [Float64](#).
- calculated p-value. [Float64](#).

## Example

Input table:

| sample_data | sample_index |
|-------------|--------------|
| 20.3        | 0            |
| 21.1        | 0            |
| 21.9        | 1            |
| 21.7        | 0            |
| 19.9        | 1            |
| 21.8        | 1            |

Query:

```
SELECT studentTTest(sample_data, sample_index) FROM student_ttest;
```

Result:

```
studentTTest(sample_data, sample_index)
(-0.21739130434783777,0.8385421208415731) |
```

## See Also

- [Student's t-test](#)
- [welchTTest function](#)

# welchTTest

Applies Welch's t-test to samples from two populations.

## Syntax

```
welchTTest(sample_data, sample_index)
```

Values of both samples are in the `sample_data` column. If `sample_index` equals to 0 then the value in that row belongs to the sample from the first population. Otherwise it belongs to the sample from the second population.

The null hypothesis is that means of populations are equal. Normal distribution is assumed. Populations may have unequal variance.

## Arguments

- `sample_data` — Sample data. [Integer](#), [Float](#) or [Decimal](#).
- `sample_index` — Sample index. [Integer](#).

## Returned values

[Tuple](#) with two elements:

- calculated t-statistic. [Float64](#).
- calculated p-value. [Float64](#).

## Example

Input table:

| sample_data | sample_index |
|-------------|--------------|
| 20.3        | 0            |
| 22.1        | 0            |
| 21.9        | 0            |
| 18.9        | 1            |
| 20.3        | 1            |
| 19          | 1            |

Query:

```
SELECT welchTTest(sample_data, sample_index) FROM welch_ttest;
```

Result:

```
welchTTest(sample_data, sample_index)
(2.7988719532211235,0.051807360348581945) |
```

## See Also

- [Welch's t-test](#)
- [studentTTest function](#)

# mannWhitneyUTest

Applies the Mann-Whitney rank test to samples from two populations.

## Syntax

```
mannWhitneyUTest[(alternative[, continuity_correction])](sample_data, sample_index)
```

Values of both samples are in the `sample_data` column. If `sample_index` equals to 0 then the value in that row belongs to the sample from the first population. Otherwise it belongs to the sample from the second population.

The null hypothesis is that two populations are stochastically equal. Also one-sided hypotheses can be tested. This test does not assume that data have normal distribution.

## Arguments

- `sample_data` — sample data. [Integer](#), [Float](#) or [Decimal](#).
- `sample_index` — sample index. [Integer](#).

## Parameters

- `alternative` — alternative hypothesis. (Optional, default: 'two-sided'.) [String](#).
  - 'two-sided';
  - 'greater';
  - 'less'.
- `continuity_correction` — if not 0 then continuity correction in the normal approximation for the p-value is applied. (Optional, default: 1.) [UInt64](#).

## Returned values

[Tuple](#) with two elements:

- calculated U-statistic. [Float64](#).
- calculated p-value. [Float64](#).

## Example

Input table:

| sample_data | sample_index |
|-------------|--------------|
| 10          | 0            |
| 11          | 0            |
| 12          | 0            |
| 1           | 1            |
| 2           | 1            |
| 3           | 1            |

Query:

```
SELECT mannWhitneyUTest('greater')(sample_data, sample_index) FROM mww_ttest;
```

Result:

```
mannWhitneyUTest('greater')(sample_data, sample_index)→
(9,0.04042779918503192)
```

## See Also

- [Mann-Whitney U test](#)
- [Stochastic ordering](#)

# 集計関数の参照

## カウント

行数またはnot-NULL値をカウントします。

ClickHouseは以下の構文をサポートしています `count`:

- `count(expr)` または `COUNT(DISTINCT expr)`.
- `count()` または `COUNT(*)`. その `count()` 構文はClickHouse固有です。

### パラメータ

機能は取ることができます:

- ゼロパラメータ。
- ワン式.

### 戻り値

- 関数がパラメータなしで呼び出された場合、行の数がカウントされます。
- もし式渡されると、関数はこの式がnot nullを返した回数をカウントします。式がaを返す場合 Null可能型の値、その後の結果 `count` 滞在しない Nullable. 式が返された場合、関数は0を返します NULLすべての行について。

どちらの場合も、戻り値の型は次のとおりです `UInt64`.

### 詳細

クリックハウスは `COUNT(DISTINCT ...)` 構文。この構造の動作は、`count_distinctImplementation` 設定。それはのどれを定義します `uniq*` 関数は、操作を実行するために使用されます。デフォルトは `uniqExact` 機能。

その `SELECT count() FROM table` テーブル内のエントリの数が別々に格納されないため、クエリは最適化されません。テーブルから小さな列を選択し、その中の値の数をカウントします。

### 例

例1:

```
SELECT count() FROM t
```

|         |   |
|---------|---|
| count() | 5 |
|---------|---|

例2:

```
SELECT name, value FROM system.settings WHERE name = 'count_distinctImplementation'
```

|                              |           |
|------------------------------|-----------|
| name                         | value     |
| count_distinctImplementation | uniqExact |

```
SELECT count(DISTINCT num) FROM t
```

```
uniqExact(num)
  3 |
```

この例では、`count(DISTINCT num)` によって実行されます。`uniqExact` に従う機能 `count_distinctImplementation` 設定値。

## 任意(x)

最初に検出された値を選択します。

クエリは毎回異なる順序でも任意の順序で実行できるため、この関数の結果は不確定です。

決定的な結果を得るには、「`min`」または「`max`」関数の代わりに「`any`」。

場合によっては、実行の順序に依存することができます。これは、`SELECT`が`ORDER BY`を使用するサブクエリから来ている場合に適用されます。

とき a `SELECT` クエリには `GROUP BY` クラスタ内のすべての式は、クラスタ内のすべての式を必要とします。`SELECT`, `HAVING`, and `ORDER BY` 句は、キーまたは集計関数から計算されます。つまり、テーブルから選択された各列は、キーまたは集計関数内で使用する必要があります。MySQLのような動作を得るには、他の列を `any` 集計関数。

## anyHeavy(x)

を使用して頻繁に発生する値を選択します。ヘビーヒッター アルゴリズム 各クエリの実行スレッドでケースの半分を超える値がある場合、この値が返されます。通常、結果は非決定的です。

```
anyHeavy(column)
```

引数

- `column` – The column name.

例

を取る オンタイム データ-セットはの頻繁に発生する値選び、`AirlineID` 列。

```
SELECT anyHeavy(AirlineID) AS res
FROM ontime
```

```
res
19690 |
```

## anyLast(x)

最後に検出された値を選択します。

結果は同じように不確定です。`any` 機能。

グループビット

ビット単位で適用 `AND` 数のシリーズのために。

```
groupBitAnd(expr)
```

パラメータ

`expr` – An expression that results in `UInt*` タイプ。

## 戻り値

の値 `UInt*` タイプ。

### 例

テストデータ:

```
binary  decimal
00101100 = 44
00011100 = 28
00001101 = 13
01010101 = 85
```

クエリ:

```
SELECT groupBitAnd(num) FROM t
```

どこに `num` テストデータを含む列です。

結果:

```
binary  decimal
00000100 = 4
```

## groupBitOr

ビット単位で適用 `OR` 数のシリーズのために。

```
groupBitOr(expr)
```

パラメータ

`expr` – An expression that results in `UInt*` タイプ。

## 戻り値

の値 `UInt*` タイプ。

### 例

テストデータ:

```
binary  decimal
00101100 = 44
00011100 = 28
00001101 = 13
01010101 = 85
```

クエリ:

```
SELECT groupBitOr(num) FROM t
```

どこに `num` テストデータを含む列です。

結果:

```
binary  decimal
01111101 = 125
```

## groupBitXor

ビット単位で適用 XOR 数のシリーズのために。

```
groupBitXor(expr)
```

パラメータ

expr – An expression that results in UInt\* タイプ。

戻り値

の値 UInt\* タイプ。

例

テストデータ:

```
binary  decimal
00101100 = 44
00011100 = 28
00001101 = 13
01010101 = 85
```

クエリ:

```
SELECT groupBitXor(num) FROM t
```

どこに num テストデータを含む列です。

結果:

```
binary  decimal
01101000 = 104
```

## groupBitmap

符号なし整数列からのビットマップまたは集計計算、uint64型の基数を返します。ビットマップ。

```
groupBitmap(expr)
```

パラメータ

expr – An expression that results in UInt\* タイプ。

戻り値

の値 UInt64 タイプ。

例

テストデータ:

```
UserID
1
1
2
3
```

クエリ：

```
SELECT groupBitmap(UserID) as num FROM t
```

結果：

```
num
3
```

## min(x)

最小値を計算します。

## max(x)

最大値を計算します。

## アルグミン(arg,val)

計算します ‘arg’ 最小値の値 ‘val’ 値。 のいくつかの異なる値がある場合 ‘arg’ の最小値に対して ‘val’、検出された最初のこれらの値が output されます。

例：

| user     | salary |
|----------|--------|
| director | 5000   |
| manager  | 3000   |
| worker   | 1000   |

```
SELECT argMin(user, salary) FROM salary
```

| argMin(user, salary) |
|----------------------|
| worker               |

## アルグマックス(arg,val)

計算します ‘arg’ 最大値 ‘val’ 値。 のいくつかの異なる値がある場合 ‘arg’ の最大値の場合 ‘val’、検出された最初のこれらの値が output されます。

## sum(x)

合計を計算します。

数字のみで動作します。

## sumWithOverflow(x)

結果には、入力パラメーターと同じデータ型を使用して、数値の合計を計算します。合計がこのデータ型の最大値を超えると、関数はエラーを返します。

数字のみで動作します。

## sumMap(キー, 値), sumMap(タプル(キー, 値))

合計 'value' に指定されたキーに従って配列 'key' 配列

キーと値の配列のタプルを渡すことは、キーと値の二つの配列を渡すことと同義です。

要素の数 'key' と 'value' 合計される行ごとに同じである必要があります。

Returns a tuple of two arrays: keys in sorted order, and values summed for the corresponding keys.

例:

```
CREATE TABLE sum_map(
    date Date,
    timeslot DateTime,
    statusMap Nested(
        status UInt16,
        requests UInt64
    ),
    statusMapTuple Tuple(Array(Int32), Array(Int32))
) ENGINE = Log;
INSERT INTO sum_map VALUES
('2000-01-01', '2000-01-01 00:00:00', [1, 2, 3], [10, 10, 10], ([1, 2, 3], [10, 10, 10])),
('2000-01-01', '2000-01-01 00:00:00', [3, 4, 5], [10, 10, 10], ([3, 4, 5], [10, 10, 10])),
('2000-01-01', '2000-01-01 00:01:00', [4, 5, 6], [10, 10, 10], ([4, 5, 6], [10, 10, 10])),
('2000-01-01', '2000-01-01 00:01:00', [6, 7, 8], [10, 10, 10], ([6, 7, 8], [10, 10, 10]));

SELECT
    timeslot,
    sumMap(statusMap.status, statusMap.requests),
    sumMap(statusMapTuple)
FROM sum_map
GROUP BY timeslot
```

| timeslot            | sumMap(statusMap.status, statusMap.requests) | sumMap(statusMapTuple)         |
|---------------------|--|--------------------------------|
| 2000-01-01 00:00:00 | ([1,2,3,4,5],[10,10,20,10,10])               | ([1,2,3,4,5],[10,10,20,10,10]) |
| 2000-01-01 00:01:00 | ([4,5,6,7,8],[10,10,20,10,10])               | ([4,5,6,7,8],[10,10,20,10,10]) |

## skewPop

を計算する 歪み シーケンスの。

```
skewPop(expr)
```

パラメータ

expr — 式 番号を返す。

戻り値

The skewness of the given distribution. Type — **Float64**

例

```
SELECT skewPop(value) FROM series_with_value_column
```

# skewSamp

を計算する サンプル歪度 シーケンスの。

渡された値がサンプルを形成する場合、確率変数の歪度の不偏推定値を表します。

```
skewSamp(expr)
```

パラメータ

expr — 式 番号を返す。

戻り値

The skewness of the given distribution. Type — **Float64**. もし  $n \leq 1$  ( $n$  はサンプルのサイズです) 、その後、関数が返します `nan`.

例

```
SELECT skewSamp(value) FROM series_with_value_column
```

## クルトポップ

を計算する 尖度 シーケンスの。

```
kurtPop(expr)
```

パラメータ

expr — 式 番号を返す。

戻り値

The kurtosis of the given distribution. Type — **Float64**

例

```
SELECT kurtPop(value) FROM series_with_value_column
```

## クルツアンプ

を計算する 尖度のサンプル シーケンスの。

渡された値がその標本を形成する場合、確率変数の尖度の不偏推定値を表します。

```
kurtSamp(expr)
```

パラメータ

expr — 式 番号を返す。

戻り値

The kurtosis of the given distribution. Type — **Float64**. もし  $n \leq 1$  ( $n$  はサンプルのサイズです) 、関数は `nan`.

例

```
SELECT kurtSamp(value) FROM series_with_value_column
```

## avg(x)

平均を計算します。  
数字のみで動作します。  
結果は常にFloat64です。

## avgWeighted

計算します 加重算術平均.

### 構文

```
avgWeighted(x, weight)
```

### パラメータ

- `x` — Values. 整数 または 浮動小数点数.
- `weight` — Weights of the values. 整数 または 浮動小数点数.

タイプの `x` と `weight` 同じである必要があります。

### 戻り値

- 加重平均。
- `NAN`. すべての重みが0に等しい場合。

タイプ: `Float64`.

### 例

クエリ:

```
SELECT avgWeighted(x, w)
FROM values('x Int8, w Int8', (4, 1), (1, 0), (10, 2))
```

結果:

```
avgWeighted(x, weight)─
  8 | ─
```

## uniq

引数の異なる値のおおよその数を計算します。

```
uniq(x[, ...])
```

### パラメータ

この関数は、可変数のパラメータを取ります。変数はあります `Tuple`, `Array`, `Date`, `DateTime`, `String` または数値型。

### 戻り値

- A UInt64-タイプ番号。

## 実装の詳細

関数:

- 集計内のすべてのパラメーターのハッシュを計算し、それを計算に使用します。
- を使用して適応サンプリングアルゴリズムです。 計算状態では、関数は65536までの要素ハッシュ値のサンプルを使用します。

This algorithm is very accurate and very efficient on the CPU. When the query contains several of these functions, using `uniq` is almost as fast as using other aggregate functions.

- 結果を確定的に提供します(クエリ処理順序に依存しません)。

使用をお勧めしますこの機能はほとんど全てのシナリオ。

も参照。

- [uniqCombined](#)
- [uniqCombined64](#)
- [uniqHLL12](#)
- [uniqExact](#)

## uniqCombined

異なる引数値のおおよその数を計算します。

```
uniqCombined(HLL_precision)(x[, ...])
```

その `uniqCombined` 関数は、異なる値の数を計算するための良い選択です。

パラメータ

この関数は、可変数のパラメータを取ります。 変数はあります `Tuple`, `Array`, `Date`, `DateTime`, `String` または数値型。

`HLL_precision` は、セルの数の底2対数です。 [HyperLogLog](#). オプションで、次のように関数を使用できます `uniqCombined(x[, ...])`. のデフォルト値 `HLL_precision` は17であり、これは実質的に96KiBの空間 ( $2^{17}$ セル、各6ビット) である。

## 戻り値

- 番号 UInt64-タイプ番号。

## 実装の詳細

関数:

- ハッシュ(64ビットのハッシュ `String` それ以外の場合は32ビット) 集計内のすべてのパラメータについて、それを計算に使用します。
- 配列、ハッシュテーブル、HyperLogLogと誤り訂正表の組み合わせを使用します。

For a small number of distinct elements, an array is used. When the set size is larger, a hash table is used. For a larger number of elements, HyperLogLog is used, which will occupy a fixed amount of memory.

- 結果を確定的に提供します(クエリ処理順序に依存しません)。

## 注

非32ビットハッシュを使用するので-String タイプは、結果よりも有意に大きい基数のための非常に高い誤差を有する `UINT_MAX` (エラーは数十億の異なる値の後にすぐに発生します)、この場合は以下を使用する必要があります **uniqCombined64**

と比較される **uniq** 関数は、**uniqCombined**:

- 数倍少ないメモリを消費します。
- 数倍高い精度で計算します。
- 通常はやや性能が低い。シナリオによっては、**uniqCombined** ではどのように絡んでいるのかを調べ **uniq** たとえば、ネットワーク経由で多数の集約状態を送信する分散クエリです。

も参照。

- **uniq**
- **uniqCombined64**
- **uniqHLL12**
- **uniqExact**

## uniqCombined64

同じ **uniqCombined** しかし、すべてのデータ型に64ビットハッシュを使用します。

## uniqHLL12

異なる引数値のおおよその数を計算します。 **HyperLogLog** アルゴリズム

```
uniqHLL12(x[, ...])
```

### パラメータ

この関数は、可変数のパラメータを取ります。変数はあります `Tuple`, `Array`, `Date`, `DateTime`, `String` または数値型。

### 戻り値

- A `UInt64`-タイプ番号。

### 実装の詳細

関数:

- 集計内のすべてのパラメーターのハッシュを計算し、それを計算に使用します。
- HyperLogLogアルゴリズムを使用して、異なる引数値の数を近似します。

2<sup>12</sup> 5-bit cells are used. The size of the state is slightly more than 2.5 KB. The result is not very accurate (up to ~10% error) for small data sets (<10K elements). However, the result is fairly accurate for high-cardinality data sets (10K-100M), with a maximum error of ~1.6%. Starting from 100M, the estimation error increases, and the function will return very inaccurate results for data sets with extremely high cardinality (1B+ elements).

- 決定的な結果を提供します(クエリ処理順序に依存しません)。

この機能の使用はお勧めしません。ほとんどの場合、[uniq](#) または [uniqCombined](#) 機能。

も参照。

- [uniq](#)
- [uniqCombined](#)
- [uniqExact](#)

## uniqExact

異なる引数値の正確な数を計算します。

```
uniqExact(x[, ...])
```

使用する `uniqExact` あなたが絶対に正確な結果が必要な場合は、関数。それ以外の場合は、[uniq](#) 機能。

その `uniqExact` 関数はより多くのメモリを使用 `uniq`、状態の大きさは、異なる値の数が増加するにつれて無限の成長を有するからである。

パラメータ

この関数は、可変数のパラメータを取ります。変数はあります `Tuple`, `Array`, `Date`, `DateTime`, `String` または数値型。

も参照。

- [uniq](#)
- [uniqCombined](#)
- [uniqHLL12](#)

## groupArray(x), groupArray(max\_size)(x)

引数値の配列を作成します。

値は、任意の（不確定な）順序で配列に追加できます。

第二のバージョン（と `max_size` パラメータ）結果の配列のサイズを以下に制限します `max_size` 要素。

例えば, `groupArray(1)(x)` に等しい。`[any(x)]`.

場合によっては、実行順序に依存することもあります。これは、次の場合に適用されます `SELECT` を使用するサブクエリから来ています `ORDER BY`.

## グループパラインセルタット

指定された位置に配列に値を挿入します。

構文

```
groupArrayInsertAt(default_x, size)(x, pos);
```

あるクエリで複数の値が同じ位置に挿入された場合、関数は次のように動作します:

- クエリが单一のスレッドで実行される場合、挿入された最初の値が使用されます。
- クエリが複数のスレッドで実行される場合、結果の値は挿入された値のいずれかになります。

パラメータ

- `x` — Value to be inserted. 式 のいずれかになります 対応するデータ型.
- `pos` — Position at which the specified element `x` 挿入されるべきです。 配列のインデックス番号はゼロから始まります。 `UInt32`.
- `default_x`— Default value for substituting in empty positions. Optional parameter. 式 その結果、データ型は `x` パラメータ。 もし `default_x` は定義されていない。 デフォルト値 使用されます。
- `size`— Length of the resulting array. Optional parameter. When using this parameter, the default value `default_x` 指定する必要があります。 `UInt32`.

## 戻り値

- 値が挿入された配列。

タイプ: 配列.

## 例

クエリ:

```
SELECT groupArrayInsertAt(toString(number), number * 2) FROM numbers(5);
```

結果:

```
groupArrayInsertAt(toString(number), multiply(number, 2))—
['0','1','2','3','4']
```

クエリ:

```
SELECT groupArrayInsertAt('')(toString(number), number * 2) FROM numbers(5);
```

結果:

```
groupArrayInsertAt('')(toString(number), multiply(number, 2))—
['0',' ','1',' ','2',' ','3',' ','4']
```

クエリ:

```
SELECT groupArrayInsertAt('', 5)(toString(number), number * 2) FROM numbers(5);
```

結果:

```
groupArrayInsertAt('', 5)(toString(number), multiply(number, 2))—
['0',' ','1',' ','2']
```

一つの位置に要素のマルチスレッド挿入。

クエリ:

```
SELECT groupArrayInsertAt(number, 0) FROM numbers_mt(10) SETTINGS max_block_size = 1;
```

このクエリの結果として、あなたはランダムな整数を取得します [0,9] 範囲 例えば:

```
groupArrayInsertAt(number, 0)
[7]
```

## グループパレイモビングスム

入力値の移動合計を計算します。

```
groupArrayMovingSum(numbers_for_summing)
groupArrayMovingSum(window_size)(numbers_for_summing)
```

この機能できるウインドウサイズとしてのパラメータとします。指定されていない場合、関数は列の行数に等しいウインドウサイズを取ります。

### パラメータ

- `numbers_for_summing` — 式 数値データ型の値になります。
- `window_size` — Size of the calculation window.

### 戻り値

- 入力データと同じサイズと型の配列。

### 例

サンプルテーブル:

```
CREATE TABLE t
(
    `int` UInt8,
    `float` Float32,
    `dec` Decimal32(2)
)
ENGINE = TinyLog
```

| int | float | dec  |
|-----|-------|------|
| 1   | 1.1   | 1.10 |
| 2   | 2.2   | 2.20 |
| 4   | 4.4   | 4.40 |
| 7   | 7.77  | 7.77 |

クエリ:

```
SELECT
    groupArrayMovingSum(int) AS I,
    groupArrayMovingSum(float) AS F,
    groupArrayMovingSum(dec) AS D
FROM t
```

| I          | F                               | D                      |
|------------|---------------------------------|------------------------|
| [1,3,7,14] | [1.1,3.3000002,7.7000003,15.47] | [1.10,3.30,7.70,15.47] |

```
SELECT
    groupArrayMovingSum(2)(int) AS I,
    groupArrayMovingSum(2)(float) AS F,
    groupArrayMovingSum(2)(dec) AS D
FROM t
```

| I          | F                               | D                      |
|------------|---------------------------------|------------------------|
| [1,3,6,11] | [1.1,3.3000002,6.6000004,12.17] | [1.10,3.30,6.60,12.17] |

## groupArrayMovingAvg

入力値の移動平均を計算します。

```
groupArrayMovingAvg(numbers_for_summing)
groupArrayMovingAvg(window_size)(numbers_for_summing)
```

この機能できるウィンドウサイズとしてのパラメータとします。指定されていない場合、関数は列の行数に等しいウィンドウサイズを取ります。

パラメータ

- `numbers_for_summing` — 式 数値データ型の値になります。
- `window_size` — Size of the calculation window.

戻り値

- 入力データと同じサイズと型の配列。

この関数は ゼロへの丸め。結果のデータ型の小数点以下の桁数を切り捨てます。

例

サンプルテーブル `b`:

```
CREATE TABLE t
(
    `int` UInt8,
    `float` Float32,
    `dec` Decimal32(2)
)
ENGINE = TinyLog
```

| int | float | dec  |
|-----|-------|------|
| 1   | 1.1   | 1.10 |
| 2   | 2.2   | 2.20 |
| 4   | 4.4   | 4.40 |
| 7   | 7.77  | 7.77 |

クエリ:

```
SELECT
    groupArrayMovingAvg(int) AS I,
    groupArrayMovingAvg(float) AS F,
    groupArrayMovingAvg(dec) AS D
FROM t
```

```
[0,0,1,3] | [0.275,0.82500005,1.9250001,3.8675] | [0.27,0.82,1.92,3.86]
```

```
SELECT
    groupArrayMovingAvg(2)(int) AS I,
    groupArrayMovingAvg(2)(float) AS F,
    groupArrayMovingAvg(2)(dec) AS D
FROM t
```

```
[0,1,3,5] | [0.55,1.6500001,3.3000002,6.085] | [0.55,1.65,3.30,6.08]
```

## groupUniqArray(x),groupUniqArray(max\_size)(x)

異なる引数値から配列を作成します。メモリ消費量は `uniqExact` 機能。

第二のバージョン（と `max_size` パラメータ）結果の配列のサイズを以下に制限します `max_size` 要素。  
例えば, `groupUniqArray(1)(x)` に等しい。`[any(x)]`.

## 分位数

近似を計算します **分位数** 数値データシーケンスの。

この関数が適用されます **貯蔵所の見本抽出** 8192までの貯蔵所のサイズおよび見本抽出のための乱数発生器を使って。  
結果は非決定的です。正確な分位値を取得するには、次の式を使用します `quantileExact` 機能。

複数を使用する場合 `quantile*` クエリ内の異なるレベルを持つ関数では、内部状態は結合されません（つまり、クエリの動作ができるほど効率的ではありません）。この場合、**分位数** 機能。

### 構文

```
quantile(level)(expr)
```

別名: `median`.

パラメータ

- `level` — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a `level` の範囲の値 `[0.01, 0.99]`. デフォルト値は 0.5です。で `level=0.5` この関数は **中央値**.
- `expr` — Expression over the column values resulting in numeric **データ型**, **日付** または **DateTime**.

### 戻り値

- 指定されたレベルの近似分位値。

タイプ:

- **Float64** 数値データ型入力の場合。
- **日付** 入力値が `Date` タイプ。
- **DateTime** 入力値が `DateTime` タイプ。

### 例

入力テーブル:

| val |
|-----|
| 1   |
| 1   |
| 2   |
| 3   |

クエリ:

```
SELECT quantile(val) FROM t
```

結果:

| quantile(val) |
|---------------|
| 1.5           |

も参照。

- 中央値
- 分位数

## quantileDeterministic

近似を計算します **分位数** 数値データシーケンスの。

この関数が適用されます **貯蔵所の見本抽出** 貯蔵所のサイズ8192までおよび見本抽出の決定論的なアルゴリズムを使って。結果は決定的です。正確な分位値を取得するには、次の式を使用します **quantileExact** 機能。

複数を使用する場合 **quantile\*** クエリ内の異なるレベルを持つ関数では、内部状態は結合されません(つまり、クエリの動作ができるほど効率的ではありません)。この場合、**分位数** 機能。

### 構文

```
quantileDeterministic(level)(expr, determinator)
```

別名: **medianDeterministic**.

### パラメータ

- **level** — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a **level** の範囲の値 [0.01, 0.99]. デフォルト値は0.5です。で **level=0.5** この関数は **中央値**.
- **expr** — Expression over the column values resulting in numeric **データ型**, **日付** または **DateTime**.
- **determinator** — Number whose hash is used instead of a random number generator in the reservoir sampling algorithm to make the result of sampling deterministic. As a determinator you can use any deterministic positive number, for example, a user id or an event id. If the same determinator value occurs too often, the function works incorrectly.

### 戻り値

- 指定されたレベルの近似分位値。

タイプ:

- **Float64** 数値データ型入力の場合。

- **日付** 入力値が **Date** タイプ。
- **DateTime** 入力値が **DateTime** タイプ。

#### 例

入力テーブル:

| val |
|-----|
| 1   |
| 1   |
| 2   |
| 3   |

クエリ:

```
SELECT quantileDeterministic(val, 1) FROM t
```

結果:

| quantileDeterministic(val, 1) |
|-------------------------------|
| 1.5                           |

も参照。

- **中央値**
- **分位数**

## quantileExact

正確に計算する **分位数** 数値データシーケンスの。

To get exact value, all the passed values are combined into an array, which is then partially sorted. Therefore, the function consumes  $O(n)$  メモリ、どこ  $n$  渡された値の数です。しかし、少數の値の場合、この関数は非常に効果的です。

複数を使用する場合 `quantile*` クエリ内の異なるレベルを持つ関数では、内部状態は結合されません(つまり、クエリの動作ができるほど効率的ではありません)。この場合、**分位数** 機能。

#### 構文

```
quantileExact(level)(expr)
```

別名: `medianExact`.

#### パラメータ

- **level** — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a `level` の範囲の値 [0.01, 0.99]。デフォルト値は0.5です。で `level=0.5` この関数は **中央値**。
- **expr** — Expression over the column values resulting in numeric **データ型**, **日付** または **DateTime**.

#### 戻り値

- 指定されたレベルの分位値。

タイプ:

- **Float64** 数値データ型入力の場合。
- **日付** 入力値が **Date** タイプ。
- **DateTime** 入力値が **DateTime** タイプ。

例

クエリ:

```
SELECT quantileExact(number) FROM numbers(10)
```

結果:

```
quantileExact(number)  
5 |
```

も参照。

- **中央値**
- **分位数**

## quantileExactWeighted

正確に計算する **分位数** 各要素の重みを考慮して、数値データシーケンスの重みを指定します。

To get exact value, all the passed values are combined into an array, which is then partially sorted. Each value is counted with its weight, as if it is present **weight** times. A hash table is used in the algorithm. Because of this, if the passed values are frequently repeated, the function consumes less RAM than **quantileExact**. この関数は、次の代わりに使用できます **quantileExact** そして、重み1を指定します。

複数を使用する場合 **quantile\*** クエリ内の異なるレベルを持つ関数では、内部状態は結合されません(つまり、クエリの動作ができるほど効率的ではありません)。この場合、**分位数** 機能。

構文

```
quantileExactWeighted(level)(expr, weight)
```

別名: **medianExactWeighted**.

パラメータ

- **level** — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a **level** の範囲の値 [0.01, 0.99]. デフォルト値は0.5です。で **level=0.5** この関数は **中央値**。
- **expr** — Expression over the column values resulting in numeric **データ型**, **日付** または **DateTime**.
- **weight** — Column with weights of sequence members. Weight is a number of value occurrences.

戻り値

- 指定されたレベルの分位値。

タイプ:

- **Float64** 数値データ型入力の場合。

- **日付** 入力値が **Date** タイプ。
- **DateTime** 入力値が **DateTime** タイプ。

例

入力テーブル:

| n | val |
|---|-----|
| 0 | 3   |
| 1 | 2   |
| 2 | 1   |
| 5 | 4   |

クエリ:

```
SELECT quantileExactWeighted(n, val) FROM t
```

結果:

|                               |
|-------------------------------|
| quantileExactWeighted(n, val) |
| 1                             |

も参照。

- **中央値**
- **分位数**

## クオントタイミング

決定された精度では、**分位数** 数値データシーケンスの。

結果は決定的です（クエリ処理順序に依存しません）。この機能を最適化と配列における分布のような積載ウェブページではバックエンド対応。

複数を使用する場合 **quantile\*** クエリ内の異なるレベルを持つ関数では、内部状態は結合されません(つまり、クエリの動作ができるほど効率的ではありません)。この場合、**分位数** 機能。

### 構文

```
quantileTiming(level)(expr)
```

別名: **medianTiming**.

### パラメータ

- **level** — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a **level** の範囲の値 [0.01, 0.99]. デフォルト値は0.5です。で **level=0.5** この関数は **中央値**。
- **expr** — 式 aを返す列の値に対して **フロート\*-タイプ** 番号。

- If negative values are passed to the function, the behavior is undefined.  
- If the value is greater than 30,000 (a page loading time of more than 30 seconds), it is assumed to be 30,000.

### 精度

計算は次の場合に正確です:

- 値の総数は5670を超えません。
  - 値の総数は5670を超えるが、ページの読み込み時間は1024ms未満です。
- それ以外の場合、計算の結果は16msの最も近い倍数に丸められます。

## 注

ページの読み込み時間の分位数を計算するために、この機能はより有効、正確です **分位数**.

## 戻り値

- 指定されたレベルの分位値。

タイプ: `Float32`.

## 注

関数に値が渡されない場合（使用する場合 `quantileTimingIf`）、**NaN** 返されます。この目的は、これらのケースをゼロになるケースと区別することです。見る **ORDER BY句** ソートに関する注意事項 **NaN** 値。

## 例

入力テーブル:

| response_time |
|---------------|
| 72            |
| 112           |
| 126           |
| 145           |
| 104           |
| 242           |
| 313           |
| 168           |
| 108           |

クエリ:

```
SELECT quantileTiming(response_time) FROM t
```

結果:

| quantileTiming(response_time) |
|-------------------------------|
| 126                           |

も参照。

- **中央値**
- **分位数**

## quantitetimingweighted

決定された精度では、**分位数** 各シーケンスメンバの重みに応じた数値データシーケンスの。

結果は決定的です（クエリ処理順序に依存しません）。この機能を最適化と配列における分布のような積載ウェブページではバックエンド対応。

複数を使用する場合 `quantile*` クエリ内の異なるレベルを持つ関数では、内部状態は結合されません(つまり、クエリの動作ができるほど効率的ではありません)。この場合、**分位数** 機能。

## 構文

```
quantileTimingWeighted(level)(expr, weight)
```

別名: `medianTimingWeighted.`

### パラメータ

- `level` — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a `level` の範囲の値 [0.01, 0.99]. デフォルト値は0.5です。で `level=0.5` この関数は **中央値**。
- `expr` — 式 aを返す列の値に対して フロート\*-タイプ番号。

- If negative values are passed to the function, the behavior is undefined.  
- If the value is greater than 30,000 (a page loading time of more than 30 seconds), it is assumed to be 30,000.

- `weight` — Column with weights of sequence elements. Weight is a number of value occurrences.

## 精度

計算は次の場合に正確です:

- 値の総数は5670を超えません。
- 値の総数は5670を超えますが、ページの読み込み時間は1024ms未満です。

それ以外の場合、計算の結果は16msの最も近い倍数に丸められます。

## 注

ページの読み込み時間の分位数を計算するために、この機能はより有効、正確です **分位数**.

## 戻り値

- 指定されたレベルの分位値。

タイプ: `Float32`.

## 注

関数に値が渡されない場合（使用する場合 `quantileTimingIf`）、**NaN** 返されます。この目的は、これらのケースをゼロになるケースと区別することです。見る **ORDER BY句** ソートに関する注意事項 **NaN** 値。

## 例

入力テーブル:

| response_time | weight |
|---------------|--------|
| 68            | 1      |
| 104           | 2      |
| 112           | 3      |
| 126           | 2      |
| 138           | 1      |
| 162           | 1      |

クエリ:

```
SELECT quantileTimingWeighted(response_time, weight) FROM t
```

結果:

|   |
|---|
| quantileTimingWeighted(response_time, weight) |
| 112   |

も参照。

- 中央値
- 分位数

## quantileTDigest

近似を計算します **分位数** を用いた数値データシーケンスの **t-ダイジェスト** アルゴリズム

最大誤差は1%です。メモリ消費は  $\log(n)$ , ここで  $n$  値の数です。結果は、クエリの実行順序に依存し、非決定的です。

機能の性能はの性能より低いです **分位数** または **クオンタイミング**. 状態サイズと精度の比に関しては、この関数は **quantile**.

複数を使用する場合 **quantile\*** クエリ内の異なるレベルを持つ関数では、内部状態は結合されません(つまり、クエリの動作ができるほど効率的ではありません)。この場合、**分位数** 機能。

### 構文

```
quantileTDigest(level)(expr)
```

別名: **medianTDigest**.

### パラメータ

- **level** — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a **level** の範囲の値 [0.01, 0.99]. デフォルト値は0.5です。で **level=0.5** この関数は **中央値**.
- **expr** — Expression over the column values resulting in numeric **データ型**, **日付** または **DateTime**.

### 戻り値

- 指定されたレベルの近似分位数。

タイプ:

- **Float64** 数値データ型入力の場合。
- **日付** 入力値が **Date** タイプ。

- **DateTime** 入力値が **DateTime** タイプ。

#### 例

クエリ:

```
SELECT quantileTDigest(number) FROM numbers(10)
```

結果:

```
quantileTDigest(number)
4.5 |
```

も参照。

- 中央値
- 分位数

## quantileTDigestWeighted

近似を計算します **分位数** を用いた数値データシーケンスの **t-ダイジェスト** アルゴリズム。この関数は、各シーケンスメンバーの重みを考慮に入れます。最大誤差は1%です。メモリ消費は  $\log(n)$ , ここで  $n$  値の数です。

機能の性能はの性能より低いです **分位数** または **クオンタイミング**。状態サイズと精度の比に関しては、この関数は **quantile**。

結果は、クエリの実行順序に依存し、非決定的です。

複数を使用する場合 **quantile\*** クエリ内の異なるレベルを持つ関数では、内部状態は結合されません(つまり、クエリの動作ができるほど効率的ではありません)。この場合、**分位数** 機能。

#### 構文

```
quantileTDigest(level)(expr)
```

別名: **medianTDigest**.

#### パラメータ

- **level** — Level of quantile. Optional parameter. Constant floating-point number from 0 to 1. We recommend using a **level** の範囲の値 [0.01, 0.99]. デフォルト値は0.5です。で **level=0.5** この関数は **中央値**。
- **expr** — Expression over the column values resulting in numeric **データ型**, **日付** または **DateTime**.
- **weight** — Column with weights of sequence elements. Weight is a number of value occurrences.

#### 戻り値

- 指定されたレベルの近似分位数。

タイプ:

- **Float64** 数値データ型入力の場合。
- **日付** 入力値が **Date** タイプ。
- **DateTime** 入力値が **DateTime** タイプ。

例

クエリ:

```
SELECT quantileTDigestWeighted(number, 1) FROM numbers(10)
```

結果:

```
quantileTDigestWeighted(number, 1)─  
4.5 |
```

も参照。

- 中央値
- 分位数

## 中央値

その `median*` 関数は、対応する関数のエイリアスです `quantile*` 機能。 数値データサンプルの中央値を計算します。

関数:

- `median` — Alias for 分位数.
- `medianDeterministic` — Alias for `quantileDeterministic`.
- `medianExact` — Alias for `quantileExact`.
- `medianExactWeighted` — Alias for `quantileExactWeighted`.
- `medianTiming` — Alias for クォンタイミング.
- `medianTimingWeighted` — Alias for `quantitetimingweighted`.
- `medianTDigest` — Alias for `quantileTDigest`.
- `medianTDigestWeighted` — Alias for `quantileTDigestWeighted`.

例

入力テーブル:

```
val─  
1  
1  
2  
3
```

クエリ:

```
SELECT medianDeterministic(val, 1) FROM t
```

結果:

```
medianDeterministic(val, 1)─  
 1.5 | ─
```

## quantiles(level1, level2, ...)(x)

すべての分位関数にも対応する分位関数があります: `quantiles`, `quantilesDeterministic`, `quantilesTiming`, `quantilesTimingWeighted`, `quantilesExact`, `quantilesExactWeighted`, `quantilesTDigest`. これらの関数は、リストされたレベルのすべての分位数を一つのパスで計算し、結果の値の配列を返します。

## varSamp(x)

金額を計算します  $\sum((x - \bar{x})^2) / (n - 1)$ , ここで  $n$  はサンプルサイズであり、 $\bar{x}$  の平均値です  $x$ .

渡された値がその標本を形成する場合、確率変数の分散の不偏推定値を表します。

ツヅケツ。 `Float64`. とき  $n \leq 1$ , 戻り値  $+\infty$ .

### 注

この機能の利用を数値的に不安定なアルゴリズムです。必要とすれば **数値安定性** 計算では、`varSampStable` 機能。それは遅く動作しますが、計算誤差は低くなります。

## varPop(x)

金額を計算します  $\sum((x - \bar{x})^2) / n$ , ここで  $n$  はサンプルサイズであり、 $\bar{x}$  の平均値です  $x$ .

言い換えれば、値の集合に対する分散。ツヅケツ。 `Float64`.

### 注

この機能の利用を数値的に不安定なアルゴリズムです。必要とすれば **数値安定性** 計算では、`varPopStable` 機能。それは遅く動作しますが、計算誤差は低くなります。

## stddevSamp(x)

結果はの平方根に等しくなります `varSamp(x)`.

### 注

この機能の利用を数値的に不安定なアルゴリズムです。必要とすれば **数値安定性** 計算では、`stddevSampStable` 機能。それは遅く動作しますが、計算誤差は低くなります。

## stddevPop(x)

結果はの平方根に等しくなります `varPop(x)`.

### 注

この機能の利用を数値的に不安定なアルゴリズムです。必要とすれば **数値安定性** 計算では、`stddevPopStable` 機能。それは遅く動作しますが、計算誤差は低くなります。

## topK(N)(x)

指定された列の中で最も頻度の高い値の配列を返します。結果の配列は、値の近似頻度の降順でソートされます（値自体ではありません）。

を実装する。ろ過されたスペース節約 TopKを解析するためのアルゴリズム。並列スペース節約。

topK(N)(column)

この関数は保証された結果を提供しません。特定の状況では、エラーが発生し、最も頻繁な値ではない頻繁な値が返されることがあります。

私達は使用を推薦します `N < 10` 価値;性能は大きいと減ります `N` 値。の最大値 `N = 65536`.

パラメータ

- 'N' 返す要素の数です。

このパラメーターを省略すると、既定値10が使用されます。

引数

- 'x' – The value to calculate frequency.

例

を取る オンタイム データセットで最も頻繁に発生する三つの値を選択します。AirlineID 列。

```
SELECT topK(3)(AirlineID) AS res  
FROM ontime
```

```
res  
[19393,19790,19805] |
```

## トップウェイト

に類似した `topK` しかし、整数型の追加引数を取ります - `weight`. すべての値が考慮されます `weight` 周波数計算のための時間。

構文

topKWeighted(N)(x, weight)

パラメータ

- `N` — The number of elements to return.

引数

- `x` – The value.
- `weight` — The weight. `UInt8`.

戻り値

おおよその重みの合計が最大の値の配列を返します。

例

クエリ:

```
SELECT topKWeighted(10)(number, number) FROM numbers(1000)
```

結果:

```
topKWeighted(10)(number, number)
[999,998,997,996,995,994,993,992,991,990] |
```

## コバルサンプ(x,y)

の値を計算します  $\Sigma((x - \bar{x})(y - \bar{y})) / (n - 1)$

Float64を返します。とき  $n \leq 1$ , returns  $+\infty$ .

### 注

この機能の利用を数値的に不安定なアルゴリズムです。必要とすれば 数値安定性 計算では、covarSampStable 機能。それは遅く動作しますが、計算誤差は低くなります。

## コバルポップ(x,y)

の値を計算します  $\Sigma((x - \bar{x})(y - \bar{y})) / n$ .

### 注

この機能の利用を数値的に不安定なアルゴリズムです。必要とすれば 数値安定性 計算では、covarPopStable 機能。それは遅く動作しますが、計算誤差は低くなります。

## corr(x,y)

ピアソン相関係数を計算します:  $\Sigma((x - \bar{x})(y - \bar{y})) / \sqrt{\Sigma((x - \bar{x})^2) * \Sigma((y - \bar{y})^2)}$

### 注

この機能の利用を数値的に不安定なアルゴリズムです。必要とすれば 数値安定性 計算では、corrStable 機能。それは遅く動作しますが、計算誤差は低くなります。

## categoricalInformationValue

の値を計算します  $(P(tag = 1) - P(tag = 0))(\log(P(tag = 1)) - \log(P(tag = 0)))$  カテゴリごとに。

```
categoricalInformationValue(category1, category2, ..., tag)
```

結果は、離散(カテゴリカル)フィーチャの方法を示します [category1, category2, ...] の値を予測する学習モデルに貢献する tag.

# simpleLinearRegression

単純(一次元)線形回帰を実行します。

```
simpleLinearRegression(x, y)
```

パラメータ:

- **x** — Column with dependent variable values.
- **y** — Column with explanatory variable values.

戻り値:

定数 (a, b) 結果の行の  $y = a*x + b$ .

例

```
SELECT arrayReduce('simpleLinearRegression', [0, 1, 2, 3], [0, 1, 2, 3])
```

```
arrayReduce('simpleLinearRegression', [0, 1, 2, 3], [0, 1, 2, 3])—  
(1,0) |
```

```
SELECT arrayReduce('simpleLinearRegression', [0, 1, 2, 3], [3, 4, 5, 6])
```

```
arrayReduce('simpleLinearRegression', [0, 1, 2, 3], [3, 4, 5, 6])—  
(1,3) |
```

# stochasticLinearRegression

この関数は、確率的線形回帰を実装します。学習率、L2正則化係数、ミニバッチサイズのカスタムパラメータをサポートし、重みを更新する方法はほとんどありません (**アダム** (デフォルトで使用), **シンプルSGD**, **勢い**, **ネステロフ**)。

パラメータ

が4カスタマイズ可能パラメータ。デフォルト値が使用されますが、良いモデルではパラメータの調整が必要です。

```
stochasticLinearRegression(1.0, 1.0, 10, 'SGD')
```

1. **learning rate** 勾配降下ステップを実行したときのステップ長の係数です。大きすぎる学習率の原因となり無限の量のモデルです。デフォルトは **0.00001**.
2. **L2 regularization coefficient** 過適合を防ぐのに役立つかもしれません。デフォルトは **0.1**.
3. **mini-batch size** グラデーションが計算され、グラデーション降下のステップを実行するために合計される要素の数を設定します。純粋な確率降下は一つの要素を使用しますが、小さなバッチ (約10要素) を持つと勾配ステップがより安定します。デフォルトは **15**.
4. **method for updating weights**、彼らは: **Adam** (デフォルトでは), **SGD**, **Momentum**, **Nesterov**. **Momentum** と **Nesterov** もう少し多くの計算とメモリを必要とするが、収束の速度と確率勾配法の安定性の点で有用であることが起こる。

## 使用法

`stochasticLinearRegression` モデルの近似と新しいデータの予測です。モデルを適合させ、後で使用するためにその状態を保存するために、`-State` 基本的に状態（モデルの重みなど）を保存するcombinator。

予測するには関数を使います `evalMLMethod` は、状態を引数として取り、予測する機能も備えています。

## 1. フィット

このようなクエリが使用され得る。

```
CREATE TABLE IF NOT EXISTS train_data
(
    param1 Float64,
    param2 Float64,
    target Float64
) ENGINE = Memory;

CREATE TABLE your_model ENGINE = Memory AS SELECT
stochasticLinearRegressionState(0.1, 0.0, 5, 'SGD')(target, param1, param2)
AS state FROM train_data;
```

ここでは、データを挿入する必要もあります `train_data` テーブル。パラメータの数は固定されておらず、渡された引数の数にのみ依存します `linearRegressionState`。すべて数値でなければなりません。

ターゲット値を持つ列（予測することを学びたい）が最初の引数として挿入されることに注意してください。

## 2. 予測

テーブルに状態を保存した後、予測に複数回使用したり、他の状態とマージして新しいより良いモデルを作成したりすることもできます。

```
WITH (SELECT state FROM your_model) AS model SELECT
evalMLMethod(model, param1, param2) FROM test_data
```

クエリは予測値の列を返します。その最初の引数に注意してください `evalMLMethod` は `AggregateFunctionState` オブジェクト、次はフィーチャの列です。

`test_data` のようなテーブルです `train_data` が含まれないことがあります。

### ノート

- 統合モデルにはユーザーの作成などのクエリ：

```
sql SELECT state1 + state2 FROM your_models
```

どこに `your_models` テーブルの両方のモデルです。このクエリは `new` を返します `AggregateFunctionState` オブジェクト

- ユーザーは、作成されたモデルのウェイトを独自の目的で取得することができます。`-State combinator` が使用されます。

```
sql SELECT stochasticLinearRegression(0.01)(target, param1, param2) FROM train_data
```

このようなクエリはモデルに適合し、その重みを返します-最初はモデルのパラメータに対応する重みであり、最後はバイアスです。したがって、上記の例では、クエリは3つの値を持つ列を返します。

も参照。

- [stochasticLogisticRegression](#)
- [線形回帰とロジスティック回帰の違い](#)

## stochasticLogisticRegression

この関数は、確率的ロジスティック回帰を実装します。これは、バイナリ分類問題に使用することができます、`stochasticLinearRegression` と同じカスタムパラメータをサポートし、同じように動作します。

## パラメータ

パラメーターは stochasticLinearRegression とまったく同じです:

learning rate, l2 regularization coefficient, mini-batch size, method for updating weights.

詳細については、[パラメータ](#)。

```
stochasticLogisticRegression(1.0, 1.0, 10, 'SGD')
```

### 1. フィット

See the `Fitting` section in the [stochasticLinearRegression](#stochasticlinearregression-usage-fitting) description.

Predicted labels have to be in  $[-1, 1]$ .

### 1. 予測

Using saved state we can predict probability of object having label `1`.

```
```sql
WITH (SELECT state FROM your_model) AS model SELECT
evalMLMethod(model, param1, param2) FROM test_data
```
```

The query will return a column of probabilities. Note that first argument of `evalMLMethod` is `AggregateFunctionState` object, next are columns of features.

We can also set a bound of probability, which assigns elements to different labels.

```
```sql
SELECT ans < 1.1 AND ans > 0.5 FROM
(WITH (SELECT state FROM your_model) AS model SELECT
evalMLMethod(model, param1, param2) AS ans FROM test_data)
```
```

Then the result will be labels.

`test\_data` is a table like `train\_data` but may not contain target value.

も参照。

- [stochasticLinearRegression](#)
- [線形回帰とロジスティック回帰の違い。](#)

## groupBitmapAnd

ビットマップ列のANDを計算し、uint64型の基数を返します。 [ビットマップ](#)。

```
groupBitmapAnd(expr)
```

### パラメータ

expr – An expression that results in AggregateFunction(groupBitmap, UInt\*) タイプ。

### 戻り値

の値 UInt64 タイプ。

### 例

```

DROP TABLE IF EXISTS bitmap_column_expr_test2;
CREATE TABLE bitmap_column_expr_test2
(
    tag_id String,
    z AggregateFunction(groupBitmap, UInt32)
)
ENGINE = MergeTree
ORDER BY tag_id;

INSERT INTO bitmap_column_expr_test2 VALUES ('tag1', bitmapBuild(cast([1,2,3,4,5,6,7,8,9,10] as Array(UInt32))));
INSERT INTO bitmap_column_expr_test2 VALUES ('tag2', bitmapBuild(cast([6,7,8,9,10,11,12,13,14,15] as
Array(UInt32))));
INSERT INTO bitmap_column_expr_test2 VALUES ('tag3', bitmapBuild(cast([2,4,6,8,10,12] as Array(UInt32))));

SELECT groupBitmapAnd(z) FROM bitmap_column_expr_test2 WHERE like(tag_id, 'tag%');
└groupBitmapAnd(z)─
  3 |
```

```

SELECT arraySort(bitmapToArray(groupBitmapAndState(z))) FROM bitmap_column_expr_test2 WHERE like(tag_id,
'tag%');
└arraySort(bitmapToArray(groupBitmapAndState(z)))─
  [6,8,10] |
```

## groupBitmapOr

ビットマップ列のORを計算し、uint64型の基数を返します。 [ビットマップ](#)。これは `groupBitmapMerge`。

`groupBitmapOr(expr)`

パラメータ

`expr` – An expression that results in `AggregateFunction(groupBitmap, UInt*)` タイプ。

戻り値

の値 `UInt64` タイプ。

例

```

DROP TABLE IF EXISTS bitmap_column_expr_test2;
CREATE TABLE bitmap_column_expr_test2
(
    tag_id String,
    z AggregateFunction(groupBitmap, UInt32)
)
ENGINE = MergeTree
ORDER BY tag_id;

INSERT INTO bitmap_column_expr_test2 VALUES ('tag1', bitmapBuild(cast([1,2,3,4,5,6,7,8,9,10] as Array(UInt32))));
INSERT INTO bitmap_column_expr_test2 VALUES ('tag2', bitmapBuild(cast([6,7,8,9,10,11,12,13,14,15] as
Array(UInt32))));
INSERT INTO bitmap_column_expr_test2 VALUES ('tag3', bitmapBuild(cast([2,4,6,8,10,12] as Array(UInt32))));

SELECT groupBitmapOr(z) FROM bitmap_column_expr_test2 WHERE like(tag_id, 'tag%');
└groupBitmapOr(z)─
  15 |
```

```

SELECT arraySort(bitmapToArray(groupBitmapOrState(z))) FROM bitmap_column_expr_test2 WHERE like(tag_id,
'tag%');
└arraySort(bitmapToArray(groupBitmapOrState(z)))─
  [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15] |
```

## groupBitmapXor

ビットマップ列のXORを計算し、uint64型の基数を返します。 [ビットマップ](#)。

```
groupBitmapOr(expr)
```

パラメータ

expr – An expression that results in AggregateFunction(groupBitmap, UInt\*) type。

戻り値

の値 UInt64 タイプ。

例

```
DROP TABLE IF EXISTS bitmap_column_expr_test2;
CREATE TABLE bitmap_column_expr_test2
(
    tag_id String,
    z AggregateFunction(groupBitmap, UInt32)
)
ENGINE = MergeTree
ORDER BY tag_id;

INSERT INTO bitmap_column_expr_test2 VALUES ('tag1', bitmapBuild(cast([1,2,3,4,5,6,7,8,9,10] as Array(UInt32))));
INSERT INTO bitmap_column_expr_test2 VALUES ('tag2', bitmapBuild(cast([6,7,8,9,10,11,12,13,14,15] as
Array(UInt32))));
INSERT INTO bitmap_column_expr_test2 VALUES ('tag3', bitmapBuild(cast([2,4,6,8,10,12] as Array(UInt32))));

SELECT groupBitmapXor(z) FROM bitmap_column_expr_test2 WHERE like(tag_id, 'tag%');
└groupBitmapXor(z)┘
  10 |
```

```
SELECT arraySort(bitmapToArray(groupBitmapXorState(z))) FROM bitmap_column_expr_test2 WHERE like(tag_id,
'tag%');
└arraySort(bitmapToArray(groupBitmapXorState(z)))┘
[1,3,5,6,8,10,11,13,14,15] |
```

## 集計関数コンビネータ

集計関数の名前には、接尾辞を付加することができます。これにより、集計関数の動作方法が変更されます。

-もし

The suffix -If can be appended to the name of any aggregate function. In this case, the aggregate function accepts an extra argument – a condition (UInt8 type). The aggregate function processes only the rows that trigger the condition. If the condition was not triggered even once, it returns a default value (usually zeros or empty strings).

例: sumIf(column, cond), countIf(cond), avgIf(x, cond), quantilesTimingIf(level1, level2)(x, cond), argMinIf(arg, val, cond)など。

条件集計関数を使用すると、サブクエリとサブクエリを使用せずに、複数の条件の集計を一度に計算できます。 [JOIN](#) 例えば、Yandexの中。Metricaは、条件付き集計関数を使用してセグメント比較機能を実装します。

-配列

-Array サフィックスは、任意の集計関数に追加できます。この場合、集計関数は ‘Array(T)’ 型(配列)の代わりに ‘T’ 引数を入力します。集計関数が複数の引数を受け入れる場合、これは同じ長さの配列でなければなりません。配列を処理する場合、集計関数はすべての配列要素にわたって元の集計関数と同様に機能します。

例1: `sumArray(arr)` -すべてのすべての要素を合計します ‘arr’ 配列。この例では、より簡単に書くことができます:  
`sum(arraySum(arr))`.

例2: `uniqArray(arr)` – Counts the number of unique elements in all ‘arr’ 配列。これは簡単な方法で行うことができます: `uniq(arrayJoin(arr))` しかし、常に追加することはできません ‘arrayJoin’ クエリに。

-Ifと-Arrayを組み合わせることができます。しかし, ‘Array’ 最後に来なければならない ‘If’. 例: `uniqArrayIf(arr, cond), quantilesTimingArrayIf(level1, level2)(arr, cond)`. この順序が原因で、‘cond’ 引数は配列ではありません。

## -状態

このコンビネータを適用すると、集計関数は結果の値（例えば、コンビネータの一意の値の数など）を返しません。`uniq` の中間状態である。`uniq`、これは一意の値の数を計算するためのハッシュテーブルです）。これは `AggregateFunction(...)` 利用できるため、さらなる処理や保存のテーブルに仕上げを集計します。

これらの固は、利用:

- `AggregatingMergeTree` テーブルエンジン。
- `finalizeAggregation` 機能。
- `runningAccumulate` 機能。
- -マージ コンビネーター
- -MergeState コンビネーター

## -マージ

このコンビネータを適用すると、`aggregate` 関数は中間集計状態を引数として受け取り、状態を結合して集計を終了し、結果の値を返します。

## -MergeState

-Merge コンビネータと同じ方法で中間集計状態をマージします。ただし、結果の値を返すのではなく、-State コンビネータに似た中間集計状態を返します。

## -ForEach

テーブルの集計関数を、対応する配列項目を集計し、結果の配列を返す配列の集計関数に変換します。例えば、`sumForEach` 配列の場合 [1, 2], [3, 4, 5] と [6, 7] 結果を返します [10, 13, 5] 対応する配列項目と一緒に追加した後。

## -オルデフオルト

集計関数の動作を変更します。

集計関数に入力値がない場合、このコンビネータを使用すると、戻り値のデータ型のデフォルト値が返されます。空の入力データを取ることができる集計関数に適用されます。

-OrDefault 他の組合せ器と使用することができます。

## 構文

```
<aggFunction>OrDefault(x)
```

### パラメータ

- x — Aggregate function parameters.

### 戻り値

集計するものがない場合は、集計関数の戻り値の型の既定値を返します。

型は、使用される集計関数に依存します。

## 例

クエリ：

```
SELECT avg(number), avgOrDefault(number) FROM numbers(0)
```

結果：

|             |                      |
|-------------|----------------------|
| avg(number) | avgOrDefault(number) |
| nan         | 0                    |

また `-OrDefault` 他のコンビネータと併用できます。 集計関数が空の入力を受け入れない場合に便利です。

クエリ：

```
SELECT avgOrDefaultIf(x, x > 10)
FROM
(
    SELECT toDecimal32(1.23, 2) AS x
)
```

結果：

|                                   |
|-----------------------------------|
| avgOrDefaultIf(x, greater(x, 10)) |
| 0.00                              |

## -オルヌル

集計関数の動作を変更します。

このコンビネータは、集計関数の結果を **Null可能** データ型。 集計関数に計算する値がない場合は、次の値が返されます **NULL**。

`-OrNull` 他の組合せ器と使用することができる。

### 構文

```
<aggFunction>OrNull(x)
```

### パラメータ

- `x` — Aggregate function parameters.

### 戻り値

- 集計関数の結果は、次のように変換されます。 **Nullable** データ型。
- **NULL**、集約するものがいない場合。

タイプ: `Nullable(aggregate function return type)`.

## 例

追加 `-orNull` 集計関数の最後に。

クエリ:

```
SELECT sumOrNull(number), toTypeName(sumOrNull(number)) FROM numbers(10) WHERE number > 10
```

結果:

```
sumOrNull(number)---toTypeName(sumOrNull(number))---  
    NULL | Nullable(UInt64) |
```

また `-OrNull` 他のコンビネータと併用できます。集計関数が空の入力を受け入れない場合に便利です。

クエリ:

```
SELECT avgOrNullIf(x, x > 10)  
FROM  
(  
    SELECT toDecimal32(1.23, 2) AS x  
)
```

結果:

```
avgOrNullIf(x, greater(x, 10))---  
    NULL |
```

## -リサンプリング

データをグループに分割し、それらのグループ内のデータを個別に集計できます。グループは、ある列の値を間隔に分割することによって作成されます。

```
<aggFunction>Resample(start, end, step)(<aggFunction_params>, resampling_key)
```

パラメータ

- `start` — Starting value of the whole required interval for `resampling_key` 値。
- `stop` — Ending value of the whole required interval for `resampling_key` 値。全体の区間は含まれていません `stop` 値 `[start, stop)`。
- `step` — Step for separating the whole interval into subintervals. The `aggFunction` これらの部分区間のそれぞれに対して独立に実行されます。
- `resampling_key` — Column whose values are used for separating data into intervals.
- `aggFunction_params` — `aggFunction` 変数。

戻り値

- の配列 `aggFunction` 各サブインターバルの結果。

例

を考える `people` 次のデータを含むテーブル:

| name   | age | wage |
|--------|-----|------|
| John   | 16  | 10   |
| Alice  | 30  | 15   |
| Mary   | 35  | 8    |
| Evelyn | 48  | 11.5 |
| David  | 62  | 9.9  |
| Brian  | 60  | 16   |

のは、その年齢の間隔にある人の名前を取得してみましょう `[30,60)` と `[60,75)`. 年齢に整数表現を使用するので、年齢を取得します `[30, 59]` と `[60,74]` 間隔。

配列内の名前を集計するには、**グルーバレイ** 集計関数。それは一つの引数を取ります。私たちの場合、それは `name` 列。その `groupArrayResample` 関数は、`age` 年齢別に名前を集計する列。必要な間隔を定義するために、`30, 75, 30` の引数 `groupArrayResample` 機能。

```
SELECT groupArrayResample(30, 75, 30)(name, age) FROM people
```

|  |
|--|
| groupArrayResample(30, 75, 30)(name, age)——                |
| <code>[['Alice','Mary','Evelyn'],['David','Brian']]</code> |

結果を考えてみましょう。

`John` 彼は若すぎるので、サンプルの外にあります。他の人は、指定された年齢間隔に従って分配されます。

今度は、指定された年齢間隔での人々の総数とその平均賃金を数えましょう。

```
SELECT
  countResample(30, 75, 30)(name, age) AS amount,
  avgResample(30, 75, 30)(wage, age) AS avg_wage
FROM people
```

|   |
|---|
| amount——  |
| <code>[3,2]</code>   <code>[11.5,12.949999809265137]</code> |

## パラメトリック集計関数

Some aggregate functions can accept not only argument columns (used for compression), but a set of parameters – constants for initialization. The syntax is two pairs of brackets instead of one. The first is for parameters, and the second is for arguments.

### ヒス

適応ヒストグラムを計算します。正確な結果を保証するものではありません。

```
histogram(number_of_bins)(values)
```

関数は以下を使用します **ストリーミン**。ヒストグラム bin の境界は、新しいデータが関数に入ると調整されます。一般的なケースでは、bin の幅は等しくありません。

パラメータ

`number_of_bins` — Upper limit for the number of bins in the histogram. The function automatically calculates the number of bins. It tries to reach the specified number of bins, but if it fails, it uses fewer bins.

`values` — 式 結果として入力値が生成されます。

## 戻り値

- 配列 の タプル 次の形式の:

```
```
[(lower_1, upper_1, height_1), ... (lower_N, upper_N, height_N)]
```

- `lower` — Lower bound of the bin.
- `upper` — Upper bound of the bin.
- `height` — Calculated height of the bin.

## 例

```
SELECT histogram(5)(number + 1)
FROM (
  SELECT *
  FROM system.numbers
  LIMIT 20
)
```

```
histogram(5)(plus(number, 1))——
[(1,4.5,4),(4.5,8.5,4),(8.5,12.75,4.125),(12.75,17,4.625),(17,20,3.25)] |
```

ヒストグラムを視覚化するには `バー` 関数、例えば:

```
WITH histogram(5)(rand() % 100) AS hist
SELECT
  arrayJoin(hist).3 AS height,
  bar(height, 0, 6, 5) AS bar
FROM
(
  SELECT *
  FROM system.numbers
  LIMIT 20
)
```



この場合、ヒストグラム bin の境界線がわからないことを覚えておく必要があります。

## sequenceMatch(pattern)(timestamp, cond1, cond2, ...)

かどうかをチェックします配列を含むイベントのチェーンに一致するパターンです。

```
sequenceMatch(pattern)(timestamp, cond1, cond2, ...)
```

警告

同じ秒で発生するイベントは、結果に影響を与える未定義の順序でシーケンス内に存在する可能性があります。

## パラメータ

- **pattern** — Pattern string. See パターン構文.
- **timestamp** — Column considered to contain time data. Typical data types are **Date** と **DateTime**. も利用できますの対応 **UInt** データ型。
- **cond1, cond2** — Conditions that describe the chain of events. Data type: **UInt8**. 最大32個の条件引数を渡すことができます。この関数は、これらの条件で説明されているイベントのみを考慮します。シーケンスに条件で記述されていないデータが含まれている場合、関数はそれらをスキップします。

## 戻り値

- パターンが一致する場合は、1。
- パターンが一致しない場合は0。

タイプ: **UInt8**.

## パターン構文

- **(?N)** — Matches the condition argument at position N. 条件は、[1, 32] 範囲 例えば、(?1) に渡された引数と一致します。cond1 パラメータ。
- **.\*** — Matches any number of events. You don't need conditional arguments to match this element of the pattern.
- **(?t operator value)** — Sets the time in seconds that should separate two events. For example, pattern (?1)(?t>1800)(?2) お互いから1800秒以上発生するイベントと一致します。これらのイベントの間に任意の数のイベントを配置できます。を使用することができます **>=, >, <, <=, ==** 演算子。

## 例

のデータを考慮する t テーブル:

time	number
1	1
2	3
3	2

クエリの実行:

```
SELECT sequenceMatch('(?1)(?2)')(time, number = 1, number = 2) FROM t
```

```
sequenceMatch('(?1)(?2)')(time, equals(number, 1), equals(number, 2))
```

関数は、番号2が番号1に続くイベントチェーンを見つけました。番号はイベントとして記述されていないため、それらの間の番号3をスキップしました。例で与えられたイベントチェーンを検索するときにこの番号を考慮に入れたい場合は、それに対する条件を作成する必要があります。

```
SELECT sequenceMatch('(?1)(?2)')(time, number = 1, number = 2, number = 3) FROM t
```

```
sequenceMatch('(?1)(?2)')(time, equals(number, 1), equals(number, 2), equals(number, 3))→  
0 |
```

この場合、番号3のイベントが1と2の間で発生したため、関数はパターンに一致するイベントチェーンを見つけることができませんでした。同じケースで4番の条件をチェックした場合、シーケンスはパターンと一致します。

```
SELECT sequenceMatch('(?1)(?2)')(time, number = 1, number = 2, number = 4) FROM t
```

```
sequenceMatch('(?1)(?2)')(time, equals(number, 1), equals(number, 2), equals(number, 4))→  
1 |
```

も参照。

- [シーケンスカウント](#)

## sequenceCount(pattern)(time, cond1, cond2, ...)

パターンに一致したイベントチェーンの数を数えます。この関数は、重複しないイベントチェーンを検索します。現在のチェーンが一致した後、次のチェーンの検索を開始します。

### 警告

同じ秒で発生するイベントは、結果に影響を与える未定義の順序でシーケンス内に存在する可能性があります。

```
sequenceCount(pattern)(timestamp, cond1, cond2, ...)
```

パラメータ

- **pattern** — Pattern string. See [パターン構文](#).
- **timestamp** — Column considered to contain time data. Typical data types are **Date** と **DateTime**. も利用できますの対応 **UInt** データ型。
- **cond1, cond2** — Conditions that describe the chain of events. Data type: **UInt8**. 最大32個の条件引数を渡すことができます。この関数は、これらの条件で説明されているイベントのみを考慮します。シーケンスに条件で記述されていないデータが含まれている場合、関数はそれらをスキップします。

戻り値

- 一致する重複しないイベントチェーンの数。

タイプ: **UInt64**.

例

のデータを考慮する **t** テーブル:

time	number
1	1
2	3
3	2
4	1
5	3
6	2

数2が数1の後に何回発生するかを数えます。:

```
SELECT sequenceCount('(?1).*(?2)')(time, number = 1, number = 2) FROM t
```

```
sequenceCount('(?1).*(?2)')(time, equals(number, 1), equals(number, 2))—
2 |
```

も参照。

- シーケンスマッチ

## ウインドウファンネル

スライドタイムウインドウ内のイベントチェーンを検索し、チェーンから発生したイベントの最大数を計算します。

機能の動作に応じてアルゴリズム:

- この関数は、チェーン内の最初の条件をトリガーし、イベントカウンターを1に設定するデータを検索します。これはスライディングウインドウが始まる瞬間です。
- だから、チェーンが順次内のウインドウのカウントを増加されます。シーケンスのイベントに障害が発生、カウンターが増加されます。
- さまざまな完了ポイントでデータに複数のイベントチェーンがある場合、関数は最も長いチェーンのサイズのみを出力します。

### 構文

```
windowFunnel(window, [mode])(timestamp, cond1, cond2, ..., condN)
```

#### パラメータ

- **window** — Length of the sliding window in seconds.
- **mode** - これはオプションの引数です。
  - 'strict' - とき 'strict' 設定されている場合、windowFunnel()は一意の値に対してのみ条件を適用します。
- **timestamp** — Name of the column containing the timestamp. Data types supported: 日付, DateTime その他の符号なし整数型 (timestampがサポートしているにもかかわらず UInt64 値はInt64最大値を超えることはできません ( $2^{63}-1$ ) )。
- **cond** — Conditions or data describing the chain of events. UInt8.

#### 戻り値

スライディングタイムウインドウ内のチェーンからの連続したトリガ条件の最大数。  
選択内のすべてのチェーンが分析されます。

タイプ: Integer.

## 例

ユーザーが電話を選択してオンラインストアで二度購入するのに十分な時間があるかどうかを判断します。

次の一連のイベントを設定します:

1. ユーザーがストアのアカウントにログインした (eventID = 1003).
2. ユーザーが電話を検索する (eventID = 1007, product = 'phone').
3. ユーザーが注文した (eventID = 1009).
4. ユーザーが再び注文しました (eventID = 1010).

入力テーブル:

event_date	user_id	timestamp	eventID	product
2019-01-28	1	2019-01-29 10:00:00	1003	phone
2019-01-31	1	2019-01-31 09:00:00	1007	phone
2019-01-30	1	2019-01-30 08:00:00	1009	phone
2019-02-01	1	2019-02-01 08:00:00	1010	phone

ユーザーの距離を調べる user\_id 2019年にチェーンを抜けることができた。

クエリ:

```
SELECT
    level,
    count() AS c
FROM
(
    SELECT
        user_id,
        windowFunnel(6048000000000000)(timestamp, eventID = 1003, eventID = 1009, eventID = 1007, eventID = 1010) AS level
    FROM trend
    WHERE (event_date >= '2019-01-01') AND (event_date <= '2019-02-02')
    GROUP BY user_id
)
GROUP BY level
ORDER BY level ASC
```

結果:

level	c
4	1

## 保持

この関数は、1から32までの型の引数の条件のセットを引数として受け取ります UInt8 るかどうかを示す一定の条件を満たためのイベントです。

任意の条件を引数として指定することができます WHERE).

最初の条件を除く条件は、ペアで適用されます：最初と第二が真であれば第二の結果は真になり、最初と third が真であれば第三の結果は真になります。

## 構文

```
retention(cond1, cond2, ..., cond32);
```

#### パラメータ

- `cond` — an expression that returns a `UInt8` 結果(1または0)。

#### 戻り値

1または0の配列。

- 1 — condition was met for the event.
- 0 — condition wasn't met for the event.

タイプ: `UInt8`.

#### 例

のは、計算の例を考えてみましょう `retention` サイトトラフィックを決定する機能。

#### 1. Create a table to illustrate an example.

```
CREATE TABLE retention_test(date Date, uid Int32) ENGINE = Memory;  
  
INSERT INTO retention_test SELECT '2020-01-01', number FROM numbers(5);  
INSERT INTO retention_test SELECT '2020-01-02', number FROM numbers(10);  
INSERT INTO retention_test SELECT '2020-01-03', number FROM numbers(15);
```

入力テーブル:

クエリ:

```
SELECT * FROM retention_test
```

結果:

date	uid
2020-01-01	0
2020-01-01	1
2020-01-01	2
2020-01-01	3
2020-01-01	4

date	uid
2020-01-02	0
2020-01-02	1
2020-01-02	2
2020-01-02	3
2020-01-02	4
2020-01-02	5
2020-01-02	6
2020-01-02	7
2020-01-02	8
2020-01-02	9

date	uid
2020-01-03	0
2020-01-03	1
2020-01-03	2
2020-01-03	3
2020-01-03	4
2020-01-03	5
2020-01-03	6
2020-01-03	7
2020-01-03	8
2020-01-03	9
2020-01-03	10
2020-01-03	11
2020-01-03	12
2020-01-03	13
2020-01-03	14

## 2. ユーザを一意のIDでグループ化 uid を使用して retention 機能。

クエリ:

```
SELECT
    uid,
    retention(date = '2020-01-01', date = '2020-01-02', date = '2020-01-03') AS r
FROM retention_test
WHERE date IN ('2020-01-01', '2020-01-02', '2020-01-03')
GROUP BY uid
ORDER BY uid ASC
```

結果:

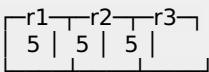
uid	r
0	[1,1,1]
1	[1,1,1]
2	[1,1,1]
3	[1,1,1]
4	[1,1,1]
5	[0,0,0]
6	[0,0,0]
7	[0,0,0]
8	[0,0,0]
9	[0,0,0]
10	[0,0,0]
11	[0,0,0]
12	[0,0,0]
13	[0,0,0]
14	[0,0,0]

3. 一日あたりのサイト訪問の合計数を計算します。

クエリ:

```
SELECT
    sum(r[1]) AS r1,
    sum(r[2]) AS r2,
    sum(r[3]) AS r3
FROM
(
    SELECT
        uid,
        retention(date = '2020-01-01', date = '2020-01-02', date = '2020-01-03') AS r
    FROM retention_test
    WHERE date IN ('2020-01-01', '2020-01-02', '2020-01-03')
    GROUP BY uid
)
```

結果:



どこに:

- r1-2020-01-01中にサイトを訪問したユニークな訪問者の数 (cond1 条件)。
- r2-2020-01-01から2020-01-02の間の特定の期間にサイトを訪問したユニーク訪問者の数 (cond1 と cond2 条件)。
- r3-2020-01-01から2020-01-03の間の特定の期間にサイトを訪問したユニーク訪問者の数 (cond1 と cond3 条件)。

## uniqUpTo(N)(x)

Calculates the number of different argument values if it is less than or equal to N. If the number of different argument values is greater than N, it returns N + 1.

小さいNsの使用のために推薦される、10まで。Nの最大値は100です。

集計関数の状態については、1+N\*バイトの値のサイズに等しいメモリ量を使用します。

文字列の場合、8バイトの非暗号化ハッシュを格納します。つまり、計算は文字列に対して近似されます。

この関数は、いくつかの引数でも機能します。

大きなN値が使用され、一意の値の数がNよりわずかに小さい場合を除いて、できるだけ高速に動作します。

使用例:

```
Problem: Generate a report that shows only keywords that produced at least 5 unique users.  
Solution: Write in the GROUP BY query SearchPhrase HAVING uniqUpTo(4)(UserID) >= 5
```

## sumMapFiltered(keys\_to\_keep)(キー, 値)

と同じ動作 サマップ ただし、キーの配列はパラメータとして渡されます。これは、キーの基数が高い場合に特に便利です。

## テーブル関数

表関数は、表を構築するためのメソッドです。

次の表関数を使用できます:

- **FROM** の節 **SELECT** クエリ。

The method for creating a temporary table that is available only in the current query. The table is deleted when the query finishes.

- テーブルを\<table\_function()>として作成 クエリ。

It's one of the methods of creating a table.

## 警告

テーブル関数を使用することはできません。 **allow\_ddl** 設定は無効です。

関数	説明
ファイル	を作成します。 <b>ファイル</b> -エンジンテーブル。
マージ	を作成します。 <b>マージ</b> -エンジンテーブル。
数字	整数で満たされた单一の列を持つテーブルを作成します。
リモート	へ自由にアクセスできるリモートサーバーを作成することなく <b>分散</b> -エンジンテーブル。
url	を作成します。 <b>Url</b> -エンジンテーブル。
mysql	を作成します。 <b>MySQL</b> -エンジンテーブル。
jdbc	を作成します。 <b>JDBC</b> -エンジンテーブル。
odbc	を作成します。 <b>ODBC</b> -エンジンテーブル。
hdfs	を作成します。 <b>HDFS</b> -エンジンテーブル。

## ファイル

ファイルからテーブルを作成します。 この表関数は次のようにになります **url** と **hdfs** ワンズ

`file(path, format, structure)`

### 入力パラメータ

- **path** — The relative path to the file from **user\_files\_path**. パスファイルをサポート **glob** に読み取り専用モード: \*, ?, {abc,def} と {N..M} どこに N, M — numbers, `'abc', 'def' — strings.
- **format** — The **形式** ファイルの。
- **structure** — Structure of the table. Format 'column1\_name column1\_type, column2\_name column2\_type, ...'.

## 戻り値

テーブルの指定された構造を読み取りまたは書き込みデータを、指定されたファイルです。

### 例

設定 `user_files_path` そして、ファイルの内容 `test.csv`:

```
$ grep user_files_path /etc/clickhouse-server/config.xml
<user_files_path>/var/lib/clickhouse/user_files/</user_files_path>

$ cat /var/lib/clickhouse/user_files/test.csv
1,2,3
3,2,1
78,43,45
```

テーブルから `test.csv` そして、それから最初の二つの行の選択:

```
SELECT *
FROM file('test.csv', 'CSV', 'column1 UInt32, column2 UInt32, column3 UInt32')
LIMIT 2
```

column1	column2	column3
1	2	3
3	2	1

```
-- getting the first 10 lines of a table that contains 3 columns of UInt32 type from a CSV file
SELECT * FROM file('test.csv', 'CSV', 'column1 UInt32, column2 UInt32, column3 UInt32') LIMIT 10
```

### パス内のグループ

複数のパスコンポーネン のための処理中のファイルが存在するマッチのパスのパターンのみならず接尾辞または接頭)。

- `*` — Substitutes any number of any characters except/ 空の文字列を含む。
- `?` — Substitutes any single character.
- `{some_string,another_string,yet_another_one}` — Substitutes any of strings 'some\_string', 'another\_string', 'yet\_another\_one'.
- `{N..M}` — Substitutes any number in range from N to M including both borders.

構造との `{}` に類似しています [遠隔テーブル機能](#)).

### 例

1. 次の相対パスを持つ複数のファイルがあるとします:

- 'some\_dir/some\_file\_1'
- 'some\_dir/some\_file\_2'
- 'some\_dir/some\_file\_3'
- 'another\_dir/some\_file\_1'
- 'another\_dir/some\_file\_2'
- 'another\_dir/some\_file\_3'

1. これらのファイル内の行の量を照会します:

```
SELECT count(*)  
FROM file('{some,another}_dir/some_file_{1..3}', 'TSV', 'name String, value UInt32')
```

1. クエリの量の行のすべてのファイルのディレクトリ:

```
SELECT count(*)  
FROM file('{some,another}_dir/*', 'TSV', 'name String, value UInt32')
```

## 警告

ファイル ?.

## 例

クエリからのデータファイル名 file000, file001, ..., file999:

```
SELECT count(*)  
FROM file('big_dir/file{0..9}{0..9}{0..9}', 'CSV', 'name String, value UInt32')
```

## 仮想列

■ `_path` — Path to the file.

■ `_file` — Name of the file.

も参照。

■ [仮想列](#)

## マージ

`merge(db_name, 'tables_regexp')` – Creates a temporary Merge table. For more information, see the section “Table engines, Merge”.

テーブル構造は、正規表現に一致する最初に検出されたテーブルから取得されます。

## 数字

`numbers(N)` – Returns a table with the single ‘number’ 0からN-1までの整数を含む列(UInt64)。

`numbers(N, M)` - 単一のテーブルを返します ‘number’ nから(N+M-1)までの整数を含む列(UInt64)。

に類似した `system.numbers` テーブルに使用でき試験および発生連続値, `numbers(N, M)` より有効 `system.numbers`.

次のクエリは同等です:

```
SELECT * FROM numbers(10);  
SELECT * FROM numbers(0, 10);  
SELECT * FROM system.numbers LIMIT 10;
```

例:

```
-- Generate a sequence of dates from 2010-01-01 to 2010-12-31
select toDate('2010-01-01') + number as d FROM numbers(365);
```

## リモート、remoteSecure

へ自由にアクセスできるリモートサーバーを作成することなく **Distributed** テーブル。

署名:

```
remote('addresses_expr', db, table[, 'user'][, 'password']])
remote('addresses_expr', db.table[, 'user'][, 'password'])
remoteSecure('addresses_expr', db, table[, 'user'][, 'password'])
remoteSecure('addresses_expr', db.table[, 'user'][, 'password'])
```

**addresses\_expr** – An expression that generates addresses of remote servers. This may be just one server address. The server address is `host:port`、または単に `host`. ホストは、サーバー名またはIPv4またはIPv6アドレスとして指定できます。IPv6アドレスは角かっこで指定します。ポートは、リモートサーバー上のTCPポートです。ポートが省略されると、次のようにになります `tcp_port` サーバーの設定ファイルから(デフォルトでは9000)。

### 重要

ポートはIPv6アドレスに必要です。

例:

```
example01-01-1
example01-01-1:9000
localhost
127.0.0.1
[::]:9000
[2a02:6b8:0:1111::11]:9000
```

複数のアドレスはコンマ区切りできます。この場合、ClickHouseは分散処理を使用するため、指定されたすべてのアドレス（異なるデータを持つシャードなど）にクエリを送信します。

例:

```
example01-01-1,example01-02-1
```

式の一部は中括弧で指定できます。前の例は次のように記述できます:

```
example01-0{1,2}-1
```

中括弧には、二つのドット（非負の整数）で区切られた数値の範囲を含めることができます。この場合、範囲はシャードアドレスを生成する値のセットに拡張されます。最初の数値がゼロで始まる場合、値は同じゼロ配置で形成されます。前の例は次のように記述できます:

```
example01-{01..02}-1
```

中括弧のペアが複数ある場合、対応する集合の直接積を生成します。

中括弧の中の住所と住所の一部は、パイプ記号(PIPE)で区切ることができます。この場合、対応するアドレスのセットはレプリカとして解釈され、クエリは最初の正常なレプリカに送信されます。ただし、レプリカは、現在設定されている順序で反復処理されます。**load\_balancing** 設定。

例:

```
example01-{01..02}-{1|2}
```

この例では、レプリカが二つあるシャードを指定します。

生成されるアドレスの数は定数によって制限されます。今これは1000アドレスです。

を使用して **remote** テーブル関数は、**Distributed** この場合、サーバー接続は要求ごとに再確立されるからです。さらに、ホスト名が設定されている場合、名前は解決され、さまざまなレプリカで作業するときにエラーはカウントされません。多数のクエリを処理する場合は、常に **Distributed** を使用しないでください。**remote** テーブル関数。

その **remote** 表関数は、次の場合に役立ちます:

- アクセスの特定のサーバーのためのデータとの比較、デバッグ、テスト実施をしておりました。
- 研究目的のための様々なClickHouseクラスター間のクエリ。
- 手動で行われる頻度の低い分散要求。
- サーバーのセットが毎回定義される分散要求。

ユーザーが指定されていない場合、**default** が使用される。

パスワードを指定しない場合は、空のパスワードが使用されます。

**remoteSecure** -同じように **remote but with secured connection**. Default port — **tcp\_port\_secure** 設定または9440から。

## url

**url(URL, format, structure)** -から作成されたテーブルを返します **URL** 与えられたと **format** と **structure**.

URL-HTTPまたはHTTPSサーバーアドレス。**GET** および/または **POST** リクエスト

形式 - **形式** データの。

**structure**-テーブルの構造 **'UserID UInt64, Name String'** 形式。列名と型を決定します。

例

```
-- getting the first 3 lines of a table that contains columns of String and UInt32 type from HTTP-server which answers in CSV format.  
SELECT * FROM url('http://127.0.0.1:12345/', CSV, 'column1 String, column2 UInt32') LIMIT 3
```

## mysql

許可 **SELECT** リモートMySQLサーバーに格納されているデータに対して実行されるクエリ。

```
mysql('host:port', 'database', 'table', 'user', 'password'[, replace_query, 'on_duplicate_clause']);
```

パラメータ

- `host:port` — MySQL server address.
- `database` — Remote database name.
- `table` — Remote table name.
- `user` — MySQL user.
- `password` — User password.
- `replace_query` — Flag that converts `INSERT INTO`へのクエリ `REPLACE INTO`. もし `replace_query=1`、クエリが置き換えられます。
- `on_duplicate_clause` — The ON DUPLICATE KEY `on_duplicate_clause`に追加される式 `INSERT` クエリ。

Example: `INSERT INTO t (c1,c2) VALUES ('a', 2) ON DUPLICATE KEY UPDATE c2 = c2 + 1` , where `on\_duplicate\_clause` is `UPDATE c2 = c2 + 1`. See the MySQL documentation to find which `on\_duplicate\_clause` you can use with the `ON DUPLICATE KEY` clause.

To specify `on\_duplicate\_clause` you need to pass `0` to the `replace\_query` parameter. If you simultaneously pass `replace\_query = 1` and `on\_duplicate\_clause` , ClickHouse generates an exception.

シンプル `WHERE` 次のような句 `=, !=, >, >=, <, <=` 現在、MySQLサーバー上で実行されています。

残りの条件と `LIMIT` サンプリング制約は、MySQLへのクエリが終了した後にのみClickHouseで実行されます。

## 戻り値

元のMySQLテーブルと同じ列を持つテーブルオブジェクト。

## 使用例

MySQLのテーブル:

```
mysql> CREATE TABLE `test`.`test` (
->   `int_id` INT NOT NULL AUTO_INCREMENT,
->   `int_nullable` INT NULL DEFAULT NULL,
->   `float` FLOAT NOT NULL,
->   `float_nullable` FLOAT NULL DEFAULT NULL,
->   PRIMARY KEY (`int_id`));
Query OK, 0 rows affected (0,09 sec)

mysql> insert into test (`int_id`, `float`) VALUES (1,2);
Query OK, 1 row affected (0,00 sec)

mysql> select * from test;
+-----+-----+-----+
| int_id | int_nullable | float | float_nullable |
+-----+-----+-----+
|    1 |      NULL |    2 |        NULL |
+-----+-----+-----+
1 row in set (0,00 sec)
```

ClickHouseからのデータの選択:

```
SELECT * FROM mysql('localhost:3306', 'test', 'test', 'bayonet', '123')
```

int_id	int_nullable	float	float_nullable
1	NULL	2	NULL

も参照。

- その ‘MySQL’ 表エンジン
- 外部辞書のソースとしてMySQLを使用する

## postgresql

Allows `SELECT` and `INSERT` queries to be performed on data that is stored on a remote PostgreSQL server.

### Syntax

```
postgresql('host:port', 'database', 'table', 'user', 'password'[, `schema`])
```

### Arguments

- `host:port` — PostgreSQL server address.
- `database` — Remote database name.
- `table` — Remote table name.
- `user` — PostgreSQL user.
- `password` — User password.
- `schema` — Non-default table schema. Optional.

### Returned Value

A table object with the same columns as the original PostgreSQL table.

### Note

In the `INSERT` query to distinguish table function `postgresql(...)` from table name with column names list you must use keywords `FUNCTION` or `TABLE FUNCTION`. See examples below.

## Implementation Details

`SELECT` queries on PostgreSQL side run as `COPY (SELECT ...)` TO `STDOUT` inside read-only PostgreSQL transaction with commit after each `SELECT` query.

Simple `WHERE` clauses such as `=`, `!=`, `>`, `>=`, `<`, `<=`, and `IN` are executed on the PostgreSQL server.

All joins, aggregations, sorting, `IN [ array ]` conditions and the `LIMIT` sampling constraint are executed in ClickHouse only after the query to PostgreSQL finishes.

`INSERT` queries on PostgreSQL side run as `COPY "table_name" (field1, field2, ... fieldN) FROM STDIN` inside PostgreSQL transaction with auto-commit after each `INSERT` statement.

PostgreSQL Array types converts into ClickHouse arrays.

### Note

Be careful, in PostgreSQL an array data type column like `Integer[]` may contain arrays of different dimensions in different rows, but in ClickHouse it is only allowed to have multidimensional arrays of the same dimension in all rows.

Supports multiple replicas that must be listed by |. For example:

```
SELECT name FROM postgresql(`postgres{1|2|3}:5432`, 'postgres_database', 'postgres_table', 'user', 'password');
```

or

```
SELECT name FROM postgresql(`postgres1:5431|postgres2:5432`, 'postgres_database', 'postgres_table', 'user', 'password');
```

Supports replicas priority for PostgreSQL dictionary source. The bigger the number in map, the less the priority. The highest priority is 0.

## Examples

Table in PostgreSQL:

```
postgres=# CREATE TABLE "public"."test" (
  "int_id" SERIAL,
  "int_nullable" INT NULL DEFAULT NULL,
  "float" FLOAT NOT NULL,
  "str" VARCHAR(100) NOT NULL DEFAULT '',
  "float_nullable" FLOAT NULL DEFAULT NULL,
  PRIMARY KEY (int_id));

CREATE TABLE

postgres=# INSERT INTO test (int_id, str, "float") VALUES (1,'test',2);
INSERT 0 1

postgresql> SELECT * FROM test;
 int_id | int_nullable | float | str  | float_nullable
-----+-----+-----+-----+
  1    |      NULL    |     2 | test |      NULL
(1 row)
```

Selecting data from ClickHouse:

```
SELECT * FROM postgresql('localhost:5432', 'test', 'test', 'postgresql_user', 'password') WHERE str IN ('test');
```

int_id	int_nullable	float	str	float_nullable
1	NULL	2	test	NULL

Inserting:

```
INSERT INTO TABLE FUNCTION postgresql('localhost:5432', 'test', 'test', 'postgresql_user', 'password') (int_id, float)
VALUES (2, 3);
SELECT * FROM postgresql('localhost:5432', 'test', 'test', 'postgresql_user', 'password');
```

int_id	int_nullable	float	str	float_nullable
1	NULL	2	test	NULL
2	NULL	3		NULL

Using Non-default Schema:

```
postgres=# CREATE SCHEMA "nice.schema";
postgres=# CREATE TABLE "nice.schema"."nice.table" (a integer);
postgres=# INSERT INTO "nice.schema"."nice.table" SELECT i FROM generate_series(0, 99) as t(i)
```

```
CREATE TABLE pg_table_schema_with_dots (a UInt32)
    ENGINE PostgreSQL('localhost:5432', 'clickhouse', 'nice.table', 'postgrsql_user', 'password', 'nice.schema');
```

## See Also

- [The PostgreSQL table engine](#)
- [Using PostgreSQL as a source of external dictionary](#)

## jdbc

`jdbc(datasource, schema, table)` - JDBC ドライバ経由で接続されたテーブルを返します。

このテーブル関数には、別々の [clickhouse-jdbc-bridge](#) 実行するプログラム。

Null許容型をサポートします(照会されるリモートテーブルのDDLに基づきます)。

例

```
SELECT * FROM jdbc('jdbc:mysql://localhost:3306/?user=root&password=root', 'schema', 'table')
```

```
SELECT * FROM jdbc('mysql://localhost:3306/?user=root&password=root', 'select * from schema.table')
```

```
SELECT * FROM jdbc('mysql-dev?p1=233', 'num Int32', 'selecttoInt32OrZero("") as num')
```

```
SELECT *
FROM jdbc('mysql-dev?p1=233', 'num Int32', 'selecttoInt32OrZero("") as num')
```

```
SELECT a.datasource AS server1, b.datasource AS server2, b.name AS db
FROM jdbc('mysql-dev?datasource_column', 'show databases') a
INNER JOIN jdbc('self?datasource_column', 'show databases') b ON a.Database = b.name
```

## odbc

接続されているテーブルを返します [ODBC](#).

```
odbc(connection_settings, external_database, external_table)
```

パラメータ:

- `connection_settings` — Name of the section with connection settings in the `odbc.ini` ファイル
- `external_database` — Name of a database in an external DBMS.
- `external_table` — Name of a table in the `external_database`.

ODBC接続を安全に実装するために、ClickHouseは別のプログラムを使用します `clickhouse-odbc-bridge`. ODBCドライバーが直接ロードされる場合 `clickhouse-server` ドライバの問題でクラッシュのClickHouseサーバーです。ClickHouseは自動的に起動します `clickhouse-odbc-bridge` それが必要なとき。ODBC bridgeプログラムは、`clickhouse-server`.

を持つフィールド `NULL` 外部テーブルの値は、基本データ型の既定値に変換されます。たとえば、リモートMySQLテーブルフィールドに `INT NULL 0` に変換される型(ClickHouseのデフォルト値 `Int32` データ型)。

## 使用例

**Pps**はインタラクティブの**MySQL**のインストール目盛

この例では、Ubuntu Linux18.04およびMySQL server5.7がチェックされています。

UnixODBCとMySQL Connectorがインストールされていることを確認します。

デフォルトでインストールされた場合、パッケージから),ClickHouse開始してユーザー `clickhouse`. したがって、MySQLサーバでこのユーザを作成して構成する必要があります。

```
$ sudo mysql
```

```
mysql> CREATE USER 'clickhouse'@'localhost' IDENTIFIED BY 'clickhouse';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'clickhouse'@'clickhouse' WITH GRANT OPTION;
```

次に、接続を設定します `/etc/odbc.ini`.

```
$ cat /etc/odbc.ini
[mysqlconn]
DRIVER = /usr/local/lib/libmyodbc5w.so
SERVER = 127.0.0.1
PORT = 3306
DATABASE = test
USERNAME = clickhouse
PASSWORD = clickhouse
```

接続を確認するには `isql` unixODBCの取付けからの実用性。

```
$ isql -v mysqlconn
+-----+
| Connected!
|
...
```

MySQLのテーブル:

```
mysql> CREATE TABLE `test`.`test` (
->   `int_id` INT NOT NULL AUTO_INCREMENT,
->   `int_nullable` INT NULL DEFAULT NULL,
->   `float` FLOAT NOT NULL,
->   `float_nullable` FLOAT NULL DEFAULT NULL,
->   PRIMARY KEY (`int_id`));
Query OK, 0 rows affected (0,09 sec)
```

```
mysql> insert into test (`int_id`, `float`) VALUES (1,2);
Query OK, 1 row affected (0,00 sec)
```

```
mysql> select * from test;
+-----+-----+-----+
| int_id | int_nullable | float | float_nullable |
+-----+-----+-----+
|    1 |      NULL |     2 |        NULL |
+-----+-----+-----+
1 row in set (0,00 sec)
```

ClickHouseのMySQLテーブルからのデータの取得:

```
SELECT * FROM odbc('DSN=mysqlconn', 'test', 'test')
```

int_id	int_nullable	float	float_nullable
1	0	2	0

も参照。

- [ODBC外部辞書](#)
- [ODBCテーブルエンジン](#).

## hdfs

HDFS内のファイルからテーブルを作成します。この表関数は次のようにになります **url** と **ファイル** ワンズ

```
hdfs(URL, format, structure)
```

入力パラメータ

- **URI** — The relative URI to the file in HDFS. Path to file support following globs in readonly mode: `*`, `?`, `{abc,def}` と `{N..M}` どこに `N, M` — numbers, `'abc', 'def'` — strings.
- **format** — The [形式](#) ファイルの。
- **structure** — Structure of the table. Format 'column1\_name column1\_type, column2\_name column2\_type, ...'.

戻り値

テーブルの指定された構造を読み取りまたは書き込みデータを、指定されたファイルです。

例

テーブルから `hdfs://hdfs1:9000/test` そして、それから最初の二つの行の選択:

```
SELECT *
FROM hdfs('hdfs://hdfs1:9000/test', 'TSV', 'column1 UInt32, column2 UInt32, column3 UInt32')
LIMIT 2
```

column1	column2	column3
1	2	3
3	2	1

## パス内のグロブ

複数のパスコンポーネン のための処理中のファイルが存在するマッチのパスのパターンのみならず接尾辞または接頭)。

- \* — Substitutes any number of any characters except/ 空の文字列を含む。
- ? — Substitutes any single character.
- {some\_string,another\_string,yet\_another\_one} — Substitutes any of strings 'some\_string', 'another\_string', 'yet\_another\_one'.
- {N..M} — Substitutes any number in range from N to M including both borders.

構造との {} に類似しています [遠隔テーブル機能](#)).

## 例

1. HDFSに次のUriを持つ複数のファイルがあるとします:

- 'hdfs://hdfs1:9000/some\_dir/some\_file\_1'
- 'hdfs://hdfs1:9000/some\_dir/some\_file\_2'
- 'hdfs://hdfs1:9000/some\_dir/some\_file\_3'
- 'hdfs://hdfs1:9000/another\_dir/some\_file\_1'
- 'hdfs://hdfs1:9000/another\_dir/some\_file\_2'
- 'hdfs://hdfs1:9000/another\_dir/some\_file\_3'

1. これらのファイル内の行の量を照会します:

```
SELECT count(*)
FROM hdfs('hdfs://hdfs1:9000/{some,another}_dir/some_file_{1..3}', 'TSV', 'name String, value UInt32')
```

1. クエリの量の行のすべてのファイルのディレクトリ:

```
SELECT count(*)
FROM hdfs('hdfs://hdfs1:9000/{some,another}_dir/*', 'TSV', 'name String, value UInt32')
```

## 警告

ファイル ?.

## 例

クエリからのデータファイル名 file000, file001, ..., file999:

```
SELECT count(*)
FROM hdfs('hdfs://hdfs1:9000/big_dir/file{0..9}{0..9}{0..9}', 'CSV', 'name String, value UInt32')
```

## 仮想列

■ `_path` — Path to the file.

■ `_file` — Name of the file.

も参照。

■ [仮想列](#)

## s3 Table Function

Provides table-like interface to select/insert files in [Amazon S3](#). This table function is similar to [hdfs](#), but provides S3-specific features.

### Syntax

```
s3(path, [aws_access_key_id, aws_secret_access_key,] format, structure, [compression])
```

### Arguments

- `path` — Bucket url with path to file. Supports following wildcards in readonly mode: `*`, `?`, `{abc,def}` and `{N..M}` where `N, M` — numbers, `'abc'`, `'def'` — strings. For more information see [here](#).
- `format` — The [format](#) of the file.
- `structure` — Structure of the table. Format '`column1_name column1_type, column2_name column2_type, ...!`'.
- `compression` — Parameter is optional. Supported values: `none`, `gzip/gz`, `brotli/br`, `xz/LZMA`, `zstd/zst`. By default, it will autodetect compression by file extension.

### Returned value

A table with the specified structure for reading or writing data in the specified file.

### Examples

Selecting the first two rows from the table from S3 file <https://storage.yandexcloud.net/my-test-bucket-768/data.csv>:

```
SELECT *
FROM s3('https://storage.yandexcloud.net/my-test-bucket-768/data.csv', 'CSV', 'column1 UInt32, column2 UInt32,
column3 UInt32')
LIMIT 2;
```

column1	column2	column3
1	2	3
3	2	1

The similar but from file with `gzip` compression:

```
SELECT *
FROM s3('https://storage.yandexcloud.net/my-test-bucket-768/data.csv.gz', 'CSV', 'column1 UInt32, column2 UInt32,
column3 UInt32', 'gzip')
LIMIT 2;
```

column1	column2	column3
1	2	3
3	2	1

## Usage

Suppose that we have several files with following URLs on S3:

- '[https://storage.yandexcloud.net/my-test-bucket-768/some\\_prefix/some\\_file\\_1.csv](https://storage.yandexcloud.net/my-test-bucket-768/some_prefix/some_file_1.csv)'
- '[https://storage.yandexcloud.net/my-test-bucket-768/some\\_prefix/some\\_file\\_2.csv](https://storage.yandexcloud.net/my-test-bucket-768/some_prefix/some_file_2.csv)'
- '[https://storage.yandexcloud.net/my-test-bucket-768/some\\_prefix/some\\_file\\_3.csv](https://storage.yandexcloud.net/my-test-bucket-768/some_prefix/some_file_3.csv)'
- '[https://storage.yandexcloud.net/my-test-bucket-768/some\\_prefix/some\\_file\\_4.csv](https://storage.yandexcloud.net/my-test-bucket-768/some_prefix/some_file_4.csv)'
- '[https://storage.yandexcloud.net/my-test-bucket-768/another\\_prefix/some\\_file\\_1.csv](https://storage.yandexcloud.net/my-test-bucket-768/another_prefix/some_file_1.csv)'
- '[https://storage.yandexcloud.net/my-test-bucket-768/another\\_prefix/some\\_file\\_2.csv](https://storage.yandexcloud.net/my-test-bucket-768/another_prefix/some_file_2.csv)'
- '[https://storage.yandexcloud.net/my-test-bucket-768/another\\_prefix/some\\_file\\_3.csv](https://storage.yandexcloud.net/my-test-bucket-768/another_prefix/some_file_3.csv)'
- '[https://storage.yandexcloud.net/my-test-bucket-768/another\\_prefix/some\\_file\\_4.csv](https://storage.yandexcloud.net/my-test-bucket-768/another_prefix/some_file_4.csv)'

Count the amount of rows in files ending with numbers from 1 to 3:

```
SELECT count(*)
FROM s3('https://storage.yandexcloud.net/my-test-bucket-768/{some,another}_prefix/some_file_{1..3}.csv', 'CSV',
'name String, value UInt32')
```

count()
18

Count the total amount of rows in all files in these two directories:

```
SELECT count(*)
FROM s3('https://storage.yandexcloud.net/my-test-bucket-768/{some,another}_prefix/*', 'CSV', 'name String, value
UInt32')
```

count()
24

## Warning

If your listing of files contains number ranges with leading zeros, use the construction with braces for each digit separately or use ?.

Count the total amount of rows in files named file-000.csv, file-001.csv, ... , file-999.csv:

```
SELECT count(*)
FROM s3('https://storage.yandexcloud.net/my-test-bucket-768/big_prefix/file-{000..999}.csv', 'CSV', 'name String,
value UInt32');
```

```
count()
12 |
```

Insert data into file `test-data.csv.gz`:

```
INSERT INTO FUNCTION s3('https://storage.yandexcloud.net/my-test-bucket-768/test-data.csv.gz', 'CSV', 'name String,
value UInt32', 'gzip')
VALUES ('test-data', 1), ('test-data-2', 2);
```

Insert data into file `test-data.csv.gz` from existing table:

```
INSERT INTO FUNCTION s3('https://storage.yandexcloud.net/my-test-bucket-768/test-data.csv.gz', 'CSV', 'name String,
value UInt32', 'gzip')
SELECT name, value FROM existing_table;
```

## Partitioned Write

If you specify `PARTITION BY` expression when inserting data into `s3` table, a separate file is created for each partition value. Splitting the data into separate files helps to improve reading operations efficiency.

### Examples

1. Using partition ID in a key creates separate files:

```
INSERT INTO TABLE FUNCTION
s3('http://bucket.amazonaws.com/my_bucket/file_{_partition_id}.csv', 'CSV', 'a String, b UInt32, c UInt32')
PARTITION BY a VALUES ('x', 2, 3), ('x', 4, 5), ('y', 11, 12), ('y', 13, 14), ('z', 21, 22), ('z', 23, 24);
```

As a result, the data is written into three files: `file_x.csv`, `file_y.csv`, and `file_z.csv`.

2. Using partition ID in a bucket name creates files in different buckets:

```
INSERT INTO TABLE FUNCTION
s3('http://bucket.amazonaws.com/my_bucket_{_partition_id}/file.csv', 'CSV', 'a UInt32, b UInt32, c UInt32')
PARTITION BY a VALUES (1, 2, 3), (1, 4, 5), (10, 11, 12), (10, 13, 14), (20, 21, 22), (20, 23, 24);
```

As a result, the data is written into three files in different buckets: `my_bucket_1/file.csv`, `my_bucket_10/file.csv`, and `my_bucket_20/file.csv`.

### See Also

- [S3 engine](#)

## 入力

`input(structure)` - に送られるデータを効果的に変え、挿入することを可能にする表機能  
別の構造を持つテーブルに指定された構造を持つサーバー。

`structure` - 以下の形式でサーバーに送信されるデータの構造 '`column1_name column1_type, column2_name column2_type, ...`'.

例えば, `'id UInt32, name String'`.

この関数は、次の場合にのみ使用できます `INSERT SELECT` それ以外の場合は通常の表関数のように動作します  
(例えば、サブクエリなどで使用できます。).

データは通常のような方法で送信することができます `INSERT` クエリと任意の利用可能に渡されます **形式** クエリの最後に指定する必要があります(通常とは異なります `INSERT SELECT`).

この機能の主な特徴は、サーバがクライアントからデータを受信すると同時に変換することです の式のリストに従って `SELECT` 節とターゲットテーブルへの挿入。一時テーブル 転送されたすべてのデータは作成されません。

## 例

- は、`test` 表の構造は次のとおりです (`a String, b String`) そしてデータ `data.csv` 異なる構造を持っています (`col1 String, col2 Date, col3 Int32`)、挿入のクエリからのデータ `data.csv` に `test` 同時変換のテーブルは次のようにになります:

```
$ cat data.csv | clickhouse-client --query="INSERT INTO test SELECT lower(col1), col3 * col3 FROM input('col1 String, col2 Date, col3 Int32') FORMAT CSV";
```

- もし `data.csv` 同じ構造のデータを含む `test_structure` 表として `test` そしてこれら二つのクエリが等しい:

```
$ cat data.csv | clickhouse-client --query="INSERT INTO test FORMAT CSV"  
$ cat data.csv | clickhouse-client --query="INSERT INTO test SELECT * FROM input('test_structure') FORMAT CSV"
```

## generateRandom

をランダムなデータを指定されたschema.

テストテーブルにデータを設定できます。

テーブルに格納できるすべてのデータ型をサポートします `LowCardinality` と `AggregateFunction`.

```
generateRandom('name TypeName[, name TypeName]...', [, 'random_seed'][, 'max_string_length'][, 'max_array_length']]));
```

### パラメータ

- `name` — Name of corresponding column.
- `TypeName` — Type of corresponding column.
- `max_array_length` — Maximum array length for all generated arrays. Defaults to 10.
- `max_string_length` — Maximum string length for all generated strings. Defaults to 10.
- `random_seed` — Specify random seed manually to produce stable results. If NULL — seed is randomly generated.

### 戻り値

テーブルオブジェクトご希望のスキーマ.

## 使用例

```
SELECT * FROM generateRandom('a Array(Int8), d Decimal32(4), c Tuple(DateTime64(3), UUID)', 1, 10, 2) LIMIT 3;
```

a	d	c
[77]	-124167.6723	('2061-04-17 21:59:44.573','3f72f405-ec3e-13c8-44ca-66ef335f7835')
[32,110]	-141397.7312	('1979-02-09 03:43:48.526','982486d1-5a5d-a308-e525-7bd8b80ffa73')
[68]	-67417.0770	('2080-03-12 14:17:31.269','110425e5-413f-10a6-05ba-fa6b3e929f15')

## cluster, clusterAllReplicas

Allows to access all shards in an existing cluster which configured in `remote_servers` section without creating a **Distributed** table. One replica of each shard is queried.

`clusterAllReplicas` function — same as `cluster`, but all replicas are queried. Each replica in a cluster is used as a separate shard/connection.

### Note

All available clusters are listed in the **system.clusters** table.

### Syntax

```
cluster('cluster_name', db.table[, sharding_key])
cluster('cluster_name', db, table[, sharding_key])
clusterAllReplicas('cluster_name', db.table[, sharding_key])
clusterAllReplicas('cluster_name', db, table[, sharding_key])
```

### Arguments

- `cluster_name` – Name of a cluster that is used to build a set of addresses and connection parameters to remote and local servers.
- `db.table` or `db, table` - Name of a database and a table.
- `sharding_key` - A sharding key. Optional. Needs to be specified if the cluster has more than one shard.

### Returned value

The dataset from clusters.

### Using Macros

`cluster_name` can contain macros — substitution in curly brackets. The substituted value is taken from the **macros** section of the server configuration file.

Example:

```
SELECT * FROM cluster('{cluster}', default.example_table);
```

### Usage and Recommendations

Using the `cluster` and `clusterAllReplicas` table functions are less efficient than creating a **Distributed** table because in this case, the server connection is re-established for every request. When processing a large number of queries, please always create the **Distributed** table ahead of time, and do not use the `cluster` and `clusterAllReplicas` table functions.

The `cluster` and `clusterAllReplicas` table functions can be useful in the following cases:

- Accessing a specific cluster for data comparison, debugging, and testing.
- Queries to various ClickHouse clusters and replicas for research purposes.
- Infrequent distributed requests that are made manually.

Connection settings like `host`, `port`, `user`, `password`, `compression`, `secure` are taken from `<remote_servers>` config section. See details in [Distributed engine](#).

## See Also

- [skip\\_unavailable\\_shards](#)
- [load\\_balancing](#)

# view

Turns a subquery into a table. The function implements views (see [CREATE VIEW](#)). The resulting table does not store data, but only stores the specified `SELECT` query. When reading from the table, ClickHouse executes the query and deletes all unnecessary columns from the result.

## Syntax

```
view(subquery)
```

## Arguments

- `subquery` — `SELECT` query.

## Returned value

- A table.

## Example

Input table:

id	name	days
1	January	31
2	February	29
3	March	31
4	April	30

Query:

```
SELECT * FROM view(SELECT name FROM months);
```

Result:

name
January
February
March
April

You can use the `view` function as a parameter of the `remote` and `cluster` table functions:

```
SELECT * FROM remote(`127.0.0.1`, view(SELECT a, b, c FROM table_name));
```

```
SELECT * FROM cluster(`cluster_name`, view(SELECT a, b, c FROM table_name));
```

## See Also

- [View Table Engine](#)

# null

Creates a temporary table of the specified structure with the [Null](#) table engine. According to the [Null](#)-engine properties, the table data is ignored and the table itself is immediately dropped right after the query execution. The function is used for the convenience of test writing and demonstrations.

## Syntax

```
null('structure')
```

## Parameter

- `structure` — A list of columns and column types. [String](#).

## Returned value

A temporary [Null](#)-engine table with the specified structure.

## Example

Query with the `null` function:

```
INSERT INTO function null('x UInt64') SELECT * FROM numbers_mt(1000000000);
```

can replace three queries:

```
CREATE TABLE t (x UInt64) ENGINE = Null;
INSERT INTO t SELECT * FROM numbers_mt(1000000000);
DROP TABLE IF EXISTS t;
```

See also:

- [Null table engine](#)

# dictionary

Displays the [dictionary](#) data as a ClickHouse table. Works the same way as [Dictionary](#) engine.

## Syntax

```
dictionary('dict')
```

## Arguments

- `dict` — A dictionary name. [String](#).

## Returned value

A ClickHouse table.

## Example

Input table `dictionary_source_table`:

id	value
0	0
1	1

Create a dictionary:

```
CREATE DICTIONARY new_dictionary(id UInt64, value UInt64 DEFAULT 0) PRIMARY KEY id
SOURCE(CLICKHOUSE(HOST 'localhost' PORT tcpPort() USER 'default' TABLE 'dictionary_source_table'))
LAYOUT(DIRECT());
```

Query:

```
SELECT * FROM dictionary('new_dictionary');
```

Result:

id	value
0	0
1	1

## See Also

- [Dictionary engine](#)

# s3Cluster Table Function

Allows processing files from [Amazon S3](#) in parallel from many nodes in a specified cluster. On initiator it creates a connection to all nodes in the cluster, discloses asterisks in S3 file path, and dispatches each file dynamically. On the worker node it asks the initiator about the next task to process and processes it. This is repeated until all tasks are finished.

## Syntax

```
s3Cluster(cluster_name, source, [access_key_id, secret_access_key,] format, structure)
```

## Arguments

- `cluster_name` — Name of a cluster that is used to build a set of addresses and connection parameters to remote and local servers.
- `source` — URL to a file or a bunch of files. Supports following wildcards in readonly mode: `*`, `?`, `{'abc','def'}` and `{N..M}` where `N, M` — numbers, `abc, def` — strings. For more information see [Wildcards In Path](#).
- `access_key_id` and `secret_access_key` — Keys that specify credentials to use with given endpoint. Optional.
- `format` — The [format](#) of the file.

- **structure** — Structure of the table. Format 'column1\_name column1\_type, column2\_name column2\_type, ...'.

## Returned value

A table with the specified structure for reading or writing data in the specified file.

## Examples

Select the data from all files in the cluster `cluster_simple`:

```
SELECT * FROM s3Cluster('cluster_simple', 'http://minio1:9001/root/data/{clickhouse,database}/*', 'minio', 'minio123', 'CSV', 'name String, value UInt32, polygon Array(Array(Tuple(Float64, Float64))))' ORDER BY (name, value, polygon);
```

Count the total amount of rows in all files in the cluster `cluster_simple`:

```
SELECT count(*) FROM s3Cluster('cluster_simple', 'http://minio1:9001/root/data/{clickhouse,database}/*', 'minio', 'minio123', 'CSV', 'name String, value UInt32, polygon Array(Array(Tuple(Float64, Float64))))';
```

## Warning

If your listing of files contains number ranges with leading zeros, use the construction with braces for each digit separately or use `?`.

## See Also

- [S3 engine](#)
- [s3 table function](#)

## sqlite

Allows to perform queries on a data stored in an [SQLite](#) database.

## Syntax

```
sqlite('db_path', 'table_name')
```

## Arguments

- `db_path` — Path to a file with an SQLite database. [String](#).
- `table_name` — Name of a table in the SQLite database. [String](#).

## Returned value

- A table object with the same columns as in the original [SQLite](#) table.

## Example

Query:

```
SELECT * FROM sqlite('sqlite.db', 'table1') ORDER BY col2;
```

Result:

col1	col2
line1	1
line2	2
line3	3

## See Also

- [SQLite table engine](#)

## 辞書

辞書はマッピングです (`key -> attributes`) それはさまざまなタイプの参照リストのために便利です。

ClickHouseは、クエリで使用できる辞書を操作するための特別な関数をサポートしています。 関数で辞書を使用する方が簡単で効率的です。[JOIN](#) 参照のテーブルを使って。

**NULL** 値を辞書に格納することはできません。

ClickHouseサポート:

- [組み込み辞書](#) 特定の 関数のセット .
- [プラグイン\(外部\)辞書](#) と 関数のセット .

## 外部辞書

さまざまなデータソースから独自の辞書を追加できます。 デイクショナリのデータソースには、ローカルテキストまたは実行可能ファイル、HTTPリソース、または別のDBMSを使用できます。 詳細については、“[外部辞書のソース](#)”.

クリックハウス:

- 完全または部分的にRAMに辞書を格納します。
- 辞書を定期的に更新し、欠損値を動的に読み込みます。 つまり、辞書は動的に読み込むことができます。
- XMLファイルで外部辞書を作成することができます。 [DDLクエリ](#) .

外部辞書の構成は、一つ以上のxmlファイルに配置できます。 設定へのパスは `dictionaries_config` パラメータ。

辞書は、サーバーの起動時または最初の使用時に読み込むことができます。 `dictionaries_lazy_load` 設定。

その [辞書](#) システムテーブルについての情報が含まれて辞書に設定されます。 各辞書については、そこにあります:

- 辞書の状態。
- 設定パラメータ。
- メトリクスのような量のメモリ割り当てのための辞書は多数のクエリからの辞書に成功したとみなされます。

辞書構成ファイルの形式は次のとおりです:

```

<clickhouse>
  <comment>An optional element with any content. Ignored by the ClickHouse server.</comment>
  <!--Optional element. File name with substitutions-->
  <include_from>/etc/metrika.xml</include_from>

  <dictionary>
    <!-- Dictionary configuration. -->
    <!-- There can be any number of <dictionary> sections in the configuration file. -->
  </dictionary>

</clickhouse>

```

あなたはできる [設定](#) 同じファイル内の任意の数の辞書。

[辞書のDDLクエリ](#) サーバー構成に追加のレコードは必要ありません。この仕事を辞書として第一級の体のように、テーブルやビュー。

## 注意

小さな辞書の値を変換するには、次のように記述します [SELECT](#) クエリ（参照 [変換](#) 機能）。この機能は外部辞書とは関係ありません。

も参照。

- [外部ディクショナリの構成](#)
- [メモリへの辞書の格納](#)
- [辞書の更新](#)
- [外部辞書のソース](#)
- [辞書キーとフィールド](#)
- [外部辞書を操作するための関数](#)

## 外部ディクショナリの構成

辞書がxmlファイルを使用して構成されている場合、辞書の構成は次の構造になります:

```

<dictionary>
  <name>dict_name</name>

  <structure>
    <!-- Complex key configuration -->
  </structure>

  <source>
    <!-- Source configuration -->
  </source>

  <layout>
    <!-- Memory layout configuration -->
  </layout>

  <lifetime>
    <!-- Lifetime of dictionary in memory -->
  </lifetime>
</dictionary>

```

対応する DDL-クエリ 次の構造を持ちます:

```
CREATE DICTIONARY dict_name
(
    ... -- attributes
)
PRIMARY KEY ... -- complex or single key configuration
SOURCE(...) -- Source configuration
LAYOUT(...) -- Memory layout configuration
LIFETIME(...) -- Lifetime of dictionary in memory
```

- **name** – The identifier that can be used to access the dictionary. Use the characters [a-zA-Z0-9\_\-].
- ソース — Source of the dictionary.
- レイアウト — Dictionary layout in memory.
- 構造 — Structure of the dictionary . A key and attributes that can be retrieved by this key.
- 生涯 — Frequency of dictionary updates.

## メモリへの辞書の格納

辞書をメモリに保存するには、さまざまな方法があります。

私達は推薦します 平ら, ハッシュ と **complex\_key\_hashed**. 最適の処理速度を提供するかどれが。

キャッシュ推奨されていないものになる可能性のある性能や困難の選定に最適なパラメータ。セクションの続きを読む “**キャッシュ**”。

する方法は幾つかあるが、今回は改善辞書性能:

- 後で辞書を操作するための関数を呼び出します **GROUP BY**.
- 抽出する属性を**injective**としてマークします。異なる属性値が異なるキーに対応する場合、属性は**injective**と呼ばれます。だからとき **GROUP BY** キーによって属性値を取得する関数を使用すると、この関数は自動的に **GROUP BY**.

ClickHouseは、辞書のエラーに対して例外を生成します。エラーの例:

- アクセス中の辞書を読み込めませんでした。
- クエリエラー **cached** 辞書。

外部辞書のリストとそのステータスは、**system.dictionaries** テーブル。

設定は次のようにになります:

```
<clickhouse>
<dictionary>
...
<layout>
    <layout_type>
        <!-- layout settings -->
    </layout_type>
</layout>
...
</dictionary>
</clickhouse>
```

対応する DDL-クエリ:

```
CREATE DICTIONARY (...)  
...  
LAYOUT(LAYOUT_TYPE(param value)) -- layout settings  
...
```

## 辞書をメモリに保存する方法

- 平ら
- ハッシュ
- sparse\_hashed
- キャッシュ
- 直接
- range\_hashed
- complex\_key\_hashed
- complex\_key\_cache
- ip\_trie

### 平ら

辞書は完全にフラット配列の形でメモリに格納されます。 辞書はどのくらいのメモリを使用しますか？ 量は、最大のキーのサイズに比例します（使用されるスペース）。

辞書キーには UInt64 タイプと値は 500,000 に制限されています。 辞書の作成時に大きなキーが検出された場合、ClickHouse は例外をスローし、辞書を作成しません。

すべての種類の源対応しています。 更新時には、ファイルまたはテーブルからのデータが全体として読み込まれます。

この方法は最高性能の中で利用可能なすべての方法を格納する辞書です。

設定例:

```
<layout>  
 <flat />  
</layout>
```

または

```
LAYOUT(FLAT())
```

### ハッシュ

辞書は、ハッシュテーブルの形でメモリに完全に格納されます。 辞書には、実際には任意の識別子を持つ任意の数の要素を含めることができます。 キーの数は数千万の項目に達することができます。

すべての種類の源対応しています。 更新時には、ファイルまたはテーブルからのデータが全体として読み込まれます。

設定例:

```
<layout>  
 <hashed />  
</layout>
```

または

```
LAYOUT(HASHED())
```

## sparse\_hashed

に類似した `hashed` が、使用メモリ賛以上のCPUます。

設定例:

```
<layout>
  <sparse_hashed />
</layout>
```

```
LAYOUT(SPARSE_HASHED())
```

## complex\_key\_hashed

このタイプの貯蔵は合成物との使用のためです キー. に類似した `hashed`.

設定例:

```
<layout>
  <complex_key_hashed />
</layout>
```

```
LAYOUT(COMPLEX_KEY_HASHED())
```

## range\_hashed

辞書は、範囲とそれに対応する値の順序付き配列を持つハッシュテーブルの形式でメモリに格納されます。

このストレージメソッドはハッシュ処理と同じように動作し、キーに加えて日付/時刻(任意の数値型)範囲を使用できます。

例:この表には、各広告主の割引が次の形式で含まれています:

advertiser id	discount start date	discount end date	amount
123	2015-01-01	2015-01-15	0.15
123	2015-01-16	2015-01-31	0.25
456	2015-01-01	2015-01-15	0.05

日付範囲のサンプルを使用するには、`range_min` と `range_max` の要素 構造. これらの要素の要素が含まれている必要があ `name` と `type` (もし `type` 指定されていない場合、デフォルトの型はuse-Dateになります)。 `type` 任意の数値型 (Date/DateTime/UInt64/Int32/others)を指定できます。

例:

```
<structure>
  <id>
    <name>Id</name>
  </id>
  <range_min>
    <name>first</name>
    <type>Date</type>
  </range_min>
  <range_max>
    <name>last</name>
    <type>Date</type>
  </range_max>
...
...
```

または

```
CREATE DICTIONARY somedict (
  id UInt64,
  first Date,
  last Date
)
PRIMARY KEY id
LAYOUT(RANGE_HASHED())
RANGE(MIN first MAX last)
```

これらの辞書を操作するには、追加の引数を渡す必要があります。 dictGetT 範囲が選択される関数:

```
dictGetT('dict_name', 'attr_name', id, date)
```

この関数は、指定された値を返します **ids** および渡された日付を含む日付範囲。

アルゴリズムの詳細:

- もし **id** が見つからないか、範囲が見つからない。 **id**, ディクショナリのデフォルト値を返します。
- 重複する範囲がある場合は、anyを使用できます。
- 範囲区切り文字が **NULL** または無効な日付(1900-01-01または2039-01-01など)、範囲は開いたままになります。 範囲は両側で開くことができる。

設定例:

```

<clickhouse>
  <dictionary>

  ...
  <layout>
    <range_hashed />
  </layout>

  <structure>
    <id>
      <name>Abcdef</name>
    </id>
    <range_min>
      <name>StartTimeStamp</name>
      <type>UInt64</type>
    </range_min>
    <range_max>
      <name>EndTimeStamp</name>
      <type>UInt64</type>
    </range_max>
    <attribute>
      <name>XXXType</name>
      <type>String</type>
      <null_value />
    </attribute>
  </structure>

  </dictionary>
</clickhouse>

```

または

```

CREATE DICTIONARY somedict(
  Abcdef UInt64,
  StartTimeStamp UInt64,
  EndTimeStamp UInt64,
  XXXType String DEFAULT ""
)
PRIMARY KEY Abcdef
RANGE(MIN StartTimeStamp MAX EndTimeStamp)

```

## キャッシュ

辞書は、固定数のセルを持つキャッシュに格納されます。これらの細胞を含む使用頻度の高いいます。

辞書を検索するときは、まずキャッシュが検索されます。データの各ブロックについて、キャッシュ内に見つからなければ、または古いすべてのキーがソースから要求されます。 `SELECT attrs... FROM db.table WHERE id IN (k1, k2, ...)` 受信したデータは、キャッシュに書き込まれます。

キャッシュディクショナリの有効期限 **生涯** キャッシュ内のデータの設定が可能です。より多くの時間が **lifetime** セルにデータをロードしてから経過した場合、セルの値は使用されず、次に使用する必要があるときに再要求されます。

これは、辞書を保存するすべての方法の中で最も効果的ではありません。キャッシュの速度は、正しい設定と使用シナリオに強く依存します。キャッシュタイプディクショナリは、ヒット率が十分に高い(推奨99%以上)場合にのみ適切に機能します。の平均ヒット率を表示することができます `system.dictionaries` テーブル。

キャッシュのパフォーマンス LIMIT、および外部辞書を使用して関数を呼び出します。

サポート ソース : MySQL、ClickHouse、実行可能ファイル、HTTP。

設定例:

```
<layout>
  <cache>
    <!-- The size of the cache, in number of cells. Rounded up to a power of two. -->
    <size_in_cells>1000000000</size_in_cells>
  </cache>
</layout>
```

または

```
LAYOUT(CACHE(SIZE_IN_CELLS 1000000000))
```

設定するのに十分な大きさのキャッシュサイズです。あなたは細胞の数を選択するために実験する必要があります:

1. 値を設定します。
2. 走行クエリーまでのキャッシュを完全に。
3. を使用してメモリ消費量を評価する `system.dictionaries` テーブル。
4. 必要なメモリ消費に達するまで、セル数を増減します。

## 警告

ClickHouseをソースとして使用しないでください。

## complex\_key\_cache

このタイプの貯蔵は合成物との使用のためです キー. に類似した `cache`.

### 直接

辞書はメモリに格納されず、要求の処理中にソースに直接移動します。

辞書キーには `UInt64` タイプ。

すべてのタイプの ソース ローカルファイル

設定例:

```
<layout>
  <direct />
</layout>
```

または

```
LAYOUT(DIRECT())
```

## ip\_trie

このタイプの貯蔵するマッピングするネットワーク接頭辞(IPアドレスへのメタデータなどのASN.

例:テーブルを含むネットワークの接頭辞およびその対応としての数および国コード:

prefix	asn	cca2
202.79.32.0/20	17501	NP
2620:0:870::/48	3856	US
2a02:6b8:1::/48	13238	RU
2001:db8::/32	65536	ZZ

このタイプのレイアウトを使用する場合、構造に複合キーが必要です。

例:

```
<structure>
  <key>
    <attribute>
      <name>prefix</name>
      <type>String</type>
    </attribute>
  </key>
  <attribute>
    <name>asn</name>
    <type>UInt32</type>
    <null_value />
  </attribute>
  <attribute>
    <name>cca2</name>
    <type>String</type>
    <null_value>??</null_value>
  </attribute>
  ...
</structure>
<layout>
  <ip_trie>
    <access_to_key_from_attributes>true</access_to_key_from_attributes>
  </ip_trie>
</layout>
```

または

```
CREATE DICTIONARY somedict (
  prefix String,
  asn UInt32,
  cca2 String DEFAULT '??'
)
PRIMARY KEY prefix
```

キーには、許可されたIPプレフィックスを含む文字列型属性のみが必要です。 その他のタイプはサポートされていませんか。

クエリでは、同じ関数を使用する必要があります (dictGetT タプル付き) 複合キーを持つ辞書については:

```
dictGetT('dict_name', 'attr_name', tuple(ip))
```

この関数は、 UInt32 IPv4の場合、または FixedString(16) IPv6の場合:

```
dictGetString('prefix', 'asn', tuple(IPv6StringToNum('2001:db8::1')))
```

その他のタイプはサポートされていませんか。 この関数は、このIPアドレスに対応するプレフィックスの属性を返します。 重複する接頭辞がある場合は、最も特定の接頭辞が返されます。

データは `trie`。それはRAMに完全に収まる必要があります。

## 辞書の更新

ClickHouseは定期的に辞書を更新します。完全にダウンロードされた辞書の更新間隔と、キャッシュされた辞書の無効化間隔は、`<lifetime>` 秒単位のタグ。

辞書の更新(最初の使用のための読み込み以外)は、クエリをブロックしません。更新時には、古いバージョンの辞書が使用されます。更新中にエラーが発生すると、エラーはサーバーログに書き込まれ、クエリは古いバージョンの辞書を使用し続けます。

設定例:

```
<dictionary>
...
<lifetime>300</lifetime>
...
</dictionary>
```

```
CREATE DICTIONARY (...)
```

```
...
LIFETIME(300)
...
```

設定 `<lifetime>0</lifetime>` (`LIFETIME(0)`) 辞書の更新を防ぎます。

しかしながら、セッションの時間間隔のアップグレード、ClickHouseを選定しランダム均一に時間以内であることが判明した。これは、多数のサーバーでアップグレードするときに辞書ソースの負荷を分散するために必要です。

設定例:

```
<dictionary>
...
<lifetime>
  <min>300</min>
  <max>360</max>
</lifetime>
...
</dictionary>
```

または

```
LIFETIME(MIN 300 MAX 360)
```

もし `<min>0</min>` と `<max>0</max>`, ClickHouseはタイムアウトによって辞書をリロードしません。

この場合、clickhouseは辞書設定ファイルが変更された場合、または `SYSTEM RELOAD DICTIONARY` コマンドが実行された。

アップする場合には辞書にClickHouseサーバーに適用の異なるロジックの種類によって ソース:

アップする場合には辞書にClickHouseサーバーに適用の異なるロジックの種類によって ソース:

- テキストファイルの場合、変更の時刻をチェックします。時刻が以前に記録された時刻と異なる場合、辞書が更新されます。
- MyISAMテーブルの場合、変更時刻は `SHOW TABLE STATUS` クエリ。
- 他のソースからの辞書は、デフォルトで毎回updatedされます。

MySQL(InnoDB)、ODBC、およびClickHouseソースでは、毎回ではなく、実際に変更された場合にのみ辞書を更新するクエリを設定できます。これを行うには、次の手順に従います:

- 辞書テーブルには、ソースデータの更新時に常に変更されるフィールドが必要です。
- ソースの設定では、変更フィールドを取得するクエリを指定する必要があります。ClickHouseサーバーは、クエリ結果を行として解釈し、この行が以前の状態に対して相対的に変更された場合、辞書が更新されます。のクエリを指定します。`<invalidate_query>` の設定のフィールド `ソース`.

設定例:

```
<dictionary>
...
<odbc>
...
  <invalidate_query>SELECT update_time FROM dictionary_source where id = 1</invalidate_query>
</odbc>
...
</dictionary>
```

または

```
...
SOURCE(ODBC(... invalidate_query 'SELECT update_time FROM dictionary_source where id = 1'))
...
```

## 外部辞書のソース

外部辞書は、さまざまなソースから接続できます。

辞書がxml-fileを使用して構成されている場合、構成は次のようにになります:

```
<clickhouse>
<dictionary>
...
<source>
<source_type>
  <!-- Source configuration -->
</source_type>
</source>
...
</dictionary>
...
</clickhouse>
```

の場合 `DDL-クエリ`、等しい構成は次のようにになります:

```
CREATE DICTIONARY dict_name (...)

...
SOURCE(SOURCE_TYPE(param1 val1 ... paramN valN)) -- Source configuration
...
```

ソースは、`source`セクション

ソースタイプの場合 `ローカル`, `実行可能ファイル`, `HTTP(s)`, `クリックハウス`任意設定は利用できる:

```
<source>
<file>
  <path>/opt/dictionaries/os.tsv</path>
  <format>TabSeparated</format>
</file>
<settings>
  <format_csv_allow_single_quotes>0</format_csv_allow_single_quotes>
</settings>
</source>
```

または

```
SOURCE(FILE(path './user_files/os.tsv' format 'TabSeparated'))
SETTINGS(format_csv_allow_single_quotes = 0)
```

ソースの種類 (source\_type):

- ロ一カル
- 実行可能ファイル
- HTTP(s)
- DBMS
  - ODBC
  - MySQL
  - クリックハウス
  - MongoDB
  - Redis

## ロ一カル

設定例:

```
<source>
<file>
  <path>/opt/dictionaries/os.tsv</path>
  <format>TabSeparated</format>
</file>
</source>
```

または

```
SOURCE(FILE(path './user_files/os.tsv' format 'TabSeparated'))
```

フィールドの設定:

- path – The absolute path to the file.
- format – The file format. All the formats described in “[形式](#)” サポートされます。

## 実行可能ファイル

実行可能ファイルを操作するには [辞書をメモリに格納する方法](#). 辞書が以下を使用して格納されている場合 cache と complex\_key\_cache, ClickHouseは、実行可能ファイルのSTDINに要求を送信することによって、必要なキーを要求します。 その他、ClickHouse始まり実行可能ファイルを扱い、その出力としての辞書のデータです。

設定例:

```
<source>
  <executable>
    <command>cat /opt/dictionaries/os.tsv</command>
    <format>TabSeparated</format>
  </executable>
</source>
```

または

```
SOURCE(EXECUTABLE(command 'cat /opt/dictionaries/os.tsv' format 'TabSeparated'))
```

フィールドの設定:

- **command** – The absolute path to the executable file, or the file name (if the program directory is written to PATH).
- **format** – The file format. All the formats described in “[形式](#)” サポートされます。

## Http(s)

HTTPサーバーでの作業は次のように依存します [辞書をメモリに格納する方法](#). 辞書が以下を使用して格納されている場合 **cache** と **complex\_key\_cache** クトを送信することにより、必要なキーを要求します。 POST 方法。

設定例:

```
<source>
  <http>
    <url>http://[::1]/os.tsv</url>
    <format>TabSeparated</format>
    <credentials>
      <user>user</user>
      <password>password</password>
    </credentials>
    <headers>
      <header>
        <name>API-KEY</name>
        <value>key</value>
      </header>
    </headers>
  </http>
</source>
```

または

```
SOURCE(HTTP(
  url 'http://[::1]/os.tsv'
  format 'TabSeparated'
  credentials(user 'user' password 'password')
  headers(header(name 'API-KEY' value 'key'))
))
```

ClickHouseがHTTPSリソースにアクセスするには、次の操作が必要です [openSSLの設定](#) サーバー構成で。

フィールドの設定:

- **url** – The source URL.
- **format** – The file format. All the formats described in “[形式](#)” サポートされます。

- `credentials` – Basic HTTP authentication. Optional parameter.
  - `user` – Username required for the authentication.
  - `password` – Password required for the authentication.
- `headers` – All custom HTTP headers entries used for the HTTP request. Optional parameter.
  - `header` – Single HTTP header entry.
  - `name` – Identifiant name used for the header send on the request.
  - `value` – Value set for a specific identifiant name.

## ODBC

このメソッドを使用して、ODBCドライバーを持つデータベースに接続できます。

設定例:

```
<source>
<odbc>
  <db>DatabaseName</db>
  <table>ShemaName.TableName</table>
  <connection_string>DSN=some_parameters</connection_string>
  <invalidate_query>SQL_QUERY</invalidate_query>
</odbc>
</source>
```

または

```
SOURCE(ODBC(
  db 'DatabaseName'
  table 'SchemaName.TableName'
  connection_string 'DSN=some_parameters'
  invalidate_query 'SQL_QUERY'
))
```

フィールドの設定:

- `db` – Name of the database. Omit it if the database name is set in the `<connection_string>` 変数。
- `table` – Name of the table and schema if exists.
- `connection_string` – Connection string.
- `invalidate_query` – Query for checking the dictionary status. Optional parameter. Read more in the section [辞書の更新](#).

ClickHouseはODBC-driverから引用シンボルを受け取り、クエリ内のすべての設定をdriverに引用するため、データベース内のテーブル名の大文字と小文字に応じてテーブル名を

Oracleの使用時にエンコーディングに問題がある場合は、対応するものを参照してください [FAQ](#) 記事だ

## ODBCディクショナリ機能の既知の脆弱性

### 注意

ODBCドライバー接続パラメーターでデータベースに接続する場合 `Servername` 置換可能である。この場合の値は `USERNAME` と `PASSWORD` から `odbc.ini` リモートサーバーに送信され、侵害される可能性があります。

安全でない使用の例

PostgreSQL用のunixODBCを設定しましょう。の内容 /etc/odbc.ini:

```
[gregtest]
Driver = /usr/lib/pgsqlodbc.so
Servername = localhost
PORT = 5432
DATABASE = test_db
##OPTION = 3
USERNAME = test
PASSWORD = test
```

次に、次のようなクエリを作成する場合

```
SELECT * FROM odbc('DSN=gregtest;Servername=some-server.com', 'test_db');
```

ODBC ドライバは、次の値を送信します USERNAME と PASSWORD から odbc.ini に some-server.com.

## Postgresqlの接続例

Ubuntu OS。

UnixodbcとPOSTGRESQL用ODBC ドライバのインストール:

```
$ sudo apt-get install -y unixodbc odbcinst odbc-postgresql
```

設定 /etc/odbc.ini ( または ~/.odbc.ini):

```
[DEFAULT]
Driver = myconnection

[myconnection]
Description      = PostgreSQL connection to my_db
Driver          = PostgreSQL Unicode
Database        = my_db
Servername      = 127.0.0.1
UserName        = username
Password        = password
Port            = 5432
Protocol        = 9.3
ReadOnly         = No
RowVersioning   = No
ShowSystemTables = No
ConnSettings    =
```

ClickHouseの辞書構成:

```

<clickhouse>
  <dictionary>
    <name>table_name</name>
    <source>
      <odbc>
        <!-- You can specify the following parameters in connection_string: -->
        <!-- DSN=myconnection;UID=username;PWD=password;HOST=127.0.0.1;PORT=5432;DATABASE=my_db -->
    >
      <connection_string>DSN=myconnection</connection_string>
      <table>postgresql_table</table>
    </odbc>
  </source>
  <lifetime>
    <min>300</min>
    <max>360</max>
  </lifetime>
  <layout>
    <hashed/>
  </layout>
  <structure>
    <id>
      <name>id</name>
    </id>
    <attribute>
      <name>some_column</name>
      <type>UInt64</type>
      <null_value>0</null_value>
    </attribute>
  </structure>
  </dictionary>
</clickhouse>

```

または

```

CREATE DICTIONARY table_name (
  id UInt64,
  some_column UInt64 DEFAULT 0
)
PRIMARY KEY id
SOURCE(ODBC(connection_string 'DSN=myconnection' table 'postgresql_table'))
LAYOUT(HASHED())
LIFETIME(MIN 300 MAX 360)

```

編集が必要な場合があります `odbc.ini` ドライバを使用してライブラリへの完全パスを指定するには  
`DRIVER=/usr/local/lib/psqlodbcw.so`.

## MS SQL Serverの接続例

Ubuntu OS。

ドライバの取り付け:::

```
$ sudo apt-get install tdsodbc freetds-bin sqsh
```

ドライバの設定:

```

$ cat /etc/freetds/freetds.conf
...
[MSQL]
host = 192.168.56.101
port = 1433
tds version = 7.0
client charset = UTF-8

$ cat /etc/odbcinst.ini
...
[FreeTDS]
Description = FreeTDS
Driver = /usr/lib/x86_64-linux-gnu/odbc/libtdsodbc.so
Setup = /usr/lib/x86_64-linux-gnu/odbc/libtdsS.so
FileUsage = 1
UsageCount = 5

$ cat ~/.odbc.ini
...
[MSQL]
Description = FreeTDS
Driver = FreeTDS
Servername = MSQL
Database = test
UID = test
PWD = test
Port = 1433

```

ClickHouseでの辞書の構成:

```

<clickhouse>
  <dictionary>
    <name>test</name>
    <source>
      <odbc>
        <table>dict</table>
        <connection_string>DSN=MSSQL;UID=test;PWD=test</connection_string>
      </odbc>
    </source>

    <lifetime>
      <min>300</min>
      <max>360</max>
    </lifetime>

    <layout>
      <flat />
    </layout>

    <structure>
      <id>
        <name>k</name>
      </id>
      <attribute>
        <name>s</name>
        <type>String</type>
        <null_value></null_value>
      </attribute>
    </structure>
  </dictionary>
</clickhouse>

```

または

```

CREATE DICTIONARY test (
    k UInt64,
    s String DEFAULT ""
)
PRIMARY KEY k
SOURCE(ODBC(table 'dict' connection_string 'DSN=MSSQL;UID=test;PWD=test'))
LAYOUT(FLAT())
LIFETIME(MIN 300 MAX 360)

```

## DBMS

### Mysql

設定例:

```

<source>
  <mysql>
    <port>3306</port>
    <user>clickhouse</user>
    <password>qwerty</password>
    <replica>
      <host>example01-1</host>
      <priority>1</priority>
    </replica>
    <replica>
      <host>example01-2</host>
      <priority>1</priority>
    </replica>
    <db>db_name</db>
    <table>table_name</table>
    <where>id=10</where>
    <invalidate_query>SQL_QUERY</invalidate_query>
  </mysql>
</source>

```

または

```

SOURCE(MYSQL(
  port 3306
  user 'clickhouse'
  password 'qwerty'
  replica(host 'example01-1' priority 1)
  replica(host 'example01-2' priority 1)
  db 'db_name'
  table 'table_name'
  where 'id=10'
  invalidate_query 'SQL_QUERY'
))

```

フィールドの設定:

- **port** – The port on the MySQL server. You can specify it for all replicas, or for each one individually (inside `<replica>`).
- **user** – Name of the MySQL user. You can specify it for all replicas, or for each one individually (inside `<replica>`).
- **password** – Password of the MySQL user. You can specify it for all replicas, or for each one individually (inside `<replica>`).

- **replica** – Section of replica configurations. There can be multiple sections.

```
- `replica/host` – The MySQL host.  
- `replica/priority` – The replica priority. When attempting to connect, ClickHouse traverses the replicas in order of priority. The lower the number, the higher the priority.
```

- **db** – Name of the database.
- **table** – Name of the table.
- **where** – The selection criteria. The syntax for conditions is the same as for **WHERE** MySQLの句、例えば, **id > 10 AND id < 20**. 任意パラメータ。
- **invalidate\_query** – Query for checking the dictionary status. Optional parameter. Read more in the section [辞書の更新](#).

MySQLはソケットを介してローカルホストに接続できます。これを行うには、**host** と **socket**.

設定例:

```
<source>
<mysql>
<host>localhost</host>
<socket>/path/to/socket/file.sock</socket>
<user>clickhouse</user>
<password>qwerty</password>
<db>db_name</db>
<table>table_name</table>
<where>id=10</where>
<invalidate_query>SQL_QUERY</invalidate_query>
</mysql>
</source>
```

または

```
SOURCE(MYSQL(
  host 'localhost'
  socket '/path/to/socket/file.sock'
  user 'clickhouse'
  password 'qwerty'
  db 'db_name'
  table 'table_name'
  where 'id=10'
  invalidate_query 'SQL_QUERY'
))
```

## クリックハウス

設定例:

```
<source>
<clickhouse>
<host>example01-01-1</host>
<port>9000</port>
<user>default</user>
<password></password>
<db>default</db>
<table>ids</table>
<where>id=10</where>
</clickhouse>
</source>
```

または

```
SOURCE(CLICKHOUSE(
    host 'example01-01-1'
    port 9000
    user 'default'
    password ""
    db 'default'
    table 'ids'
    where 'id=10'
))
```

フィールドの設定:

- **host** – The ClickHouse host. If it is a local host, the query is processed without any network activity. To improve fault tolerance, you can create a **分散** テーブルと後続の構成でそれを入力します。
- **port** – The port on the ClickHouse server.
- **user** – Name of the ClickHouse user.
- **password** – Password of the ClickHouse user.
- **db** – Name of the database.
- **table** – Name of the table.
- **where** – The selection criteria. May be omitted.
- **invalidate\_query** – Query for checking the dictionary status. Optional parameter. Read more in the section [辞書の更新](#).

## Mongodb

設定例:

```
<source>
  <mongodb>
    <host>localhost</host>
    <port>27017</port>
    <user></user>
    <password></password>
    <db>test</db>
    <collection>dictionary_source</collection>
  </mongodb>
</source>
```

または

```
SOURCE(MONGO(
    host 'localhost'
    port 27017
    user ""
    password ""
    db 'test'
    collection 'dictionary_source'
))
```

フィールドの設定:

- **host** – The MongoDB host.
- **port** – The port on the MongoDB server.
- **user** – Name of the MongoDB user.

- `password` – Password of the MongoDB user.
- `db` – Name of the database.
- `collection` – Name of the collection.

## Redis

設定例:

```
<source>
  <redis>
    <host>localhost</host>
    <port>6379</port>
    <storage_type>simple</storage_type>
    <db_index>0</db_index>
  </redis>
</source>
```

または

```
SOURCE(REDIS(
  host 'localhost'
  port 6379
  storage_type 'simple'
  db_index 0
))
```

フィールドの設定:

- `host` – The Redis host.
- `port` – The port on the Redis server.
- `storage_type` – The structure of internal Redis storage using for work with keys. `simple` は簡単な源のためのハッシュされたシングルキー源, `hash_map` 二つのキーを持つハッシュソース用です。 距源およびキャッシュ源の複雑な鍵サポートされていません。 省略可能であり、デフォルト値は `simple`.
- `db_index` – The specific numeric index of Redis logical database. May be omitted, default value is 0.

## 辞書キーとフィールド

その `<structure>` 勾エリで使用できる辞書キーとフィールドを説明します。

XMLの説明:

```
<dictionary>
  <structure>
    <id>
      <name>Id</name>
    </id>

    <attribute>
      <!-- Attribute parameters -->
    </attribute>

    ...
  </structure>
</dictionary>
```

属性は要素に記述されています:

- `<id>` — キー列。
- `<attribute>` — データ列。複数の属性が存在する可能性があります。

DDLクエリ:

```
CREATE DICTIONARY dict_name (
    Id UInt64,
    -- attributes
)
PRIMARY KEY Id
...
```

属性は、クエリの本文に記述されています:

- `PRIMARY KEY` — キー列
- `AttrName AttrType` — データ列。複数の属性が存在する可能性があります。

## キー

ClickHouseは次の種類のキーをサポートしています:

- 数値キー。 `UInt64` で定義される。`<id>` タグまたは使用 `PRIMARY KEY` キーワード。
- 複合キー。異なる型の値のセット。タグで定義 `<key>` または `PRIMARY KEY` キーワード。

XMLの構造を含むことができま `<id>` または `<key>`。DDL-クエリには单一を含む必要があります `PRIMARY KEY`.

### 警告

キーを属性として記述することはできません。

## 数値キー

タイプ: `UInt64`.

設定例:

```
<id>
    <name>Id</name>
</id>
```

設定フィールド:

- `name` – The name of the column with keys.

DDLクエリの場合:

```
CREATE DICTIONARY (
    Id UInt64,
    ...
)
PRIMARY KEY Id
...
```

- `PRIMARY KEY` – The name of the column with keys.

## 複合キー

キーはaである場合もあります `tuple` フィールドの任意のタイプから。その **レイアウト** この場合、`complex_key_hashed` または `complex_key_cache`.

## ヒント

複合キーは、単一の要素で構成できます。これにより、たとえば文字列をキーとして使用することができます。

キー構造は要素に設定されます `<key>`. キーフィールドは、辞書と同じ形式で指定されます **属性**. 例:

```
<structure>
  <key>
    <attribute>
      <name>field1</name>
      <type>String</type>
    </attribute>
    <attribute>
      <name>field2</name>
      <type>UInt32</type>
    </attribute>
    ...
  </key>
  ...

```

または

```
CREATE DICTIONARY (
  field1 String,
  field2 String
  ...
)
PRIMARY KEY field1, field2
...
```

クエリに対して `dictGet*` 関数は、タプルがキーとして渡されます。例: `dictGetString('dict_name', 'attr_name', tuple('string for field1', num_for_field2))`.

## 属性

設定例:

```
<structure>
  ...
  <attribute>
    <name>Name</name>
    <type>ClickHouseDataType</type>
    <null_value></null_value>
    <expression>rand64()</expression>
    <hierarchical>true</hierarchical>
    <injective>true</injective>
    <is_object_id>true</is_object_id>
  </attribute>
</structure>
```

または

```
CREATE DICTIONARY somename (
  Name ClickHouseDataType DEFAULT '' EXPRESSION rand64() HIERARCHICAL INJECTIVE IS_OBJECT_ID
)
```

設定フィールド:

タグ	説明	必須
<code>name</code>	列名。	はい。
<code>type</code>	ClickHouseデータ型。 ClickHouseは、 <code>dictionary</code> から指定されたデータ型に値をキャストしようとします。例えば、MySQL、フィールドが <code>TEXT</code> , <code>VARCHAR</code> , または <code>BLOB</code> MySQLソーステーブルでは、次のようにアップロードできます <code>String</code> クリックハウスで <b>Null可能</b> サポートされていない。	はい。
<code>null_value</code>	既存の要素以外の既定値。 この例では、空の文字列です。使用できません <code>NULL</code> この分野で。	はい。
<code>expression</code>	式 そのClickHouseは値に対して実行されます。 式には、リモートSQLデータベースの列名を指定できます。したがって、リモート列の別名を作成するために使用できます。  デフォルト値:式なし。	いいえ。
<code>hierarchical</code>	もし <code>true</code> この属性には、現在のキーの親キーの値が含まれます。見る <b>階層辞書</b> 。  デフォルト値: <code>false</code> .	いいえ。
<code>injective</code>	このフラグは <code>id -&gt; attribute</code> 画像は <b>injective</b> . もし <code>true</code> 、ClickHouseはの後に自動的に置くことができます <code>GROUP BY</code> 句インジェクションを使用した辞書への要求。通常、そのような要求の量を大幅に削減します。  デフォルト値: <code>false</code> .	いいえ。
<code>is_object_id</code>	クエリがMongoDBドキュメントに対して実行されるかどうかを示すフラグ <code>ObjectID</code> .  デフォルト値: <code>false</code> .	いいえ。

も参照。

- 外部辞書を操作するための関数.

## 階層辞書

ClickHouseは、階層辞書をサポートしています **数値キー**.

次の階層構造を見てください:



この階層は、次の辞書テーブルとして表すことができます。

region_id	parent_region	region_name
1	0	ロシア
2	1	モスクワ
3	2	中央
4	0	イギリス
5	4	ロンドン

このテーブル列 `parent_region` これには、要素の最も近い親のキーが含まれます。

クリックハウスは **階層** プロパティ **外部辞書** 属性。このプロパティを使用すると、上記のような階層辞書を構成できます。

その `dictGetHierarchy` 関数を使用すると、要素の親チェーンを取得することができます。

この例では、`dictionary`の構造は次のようにになります:

```
<dictionary>
  <structure>
    <id>
      <name>region_id</name>
    </id>

    <attribute>
      <name>parent_region</name>
      <type>UInt64</type>
      <null_value>0</null_value>
      <hierarchical>true</hierarchical>
    </attribute>

    <attribute>
      <name>region_name</name>
      <type>String</type>
      <null_value></null_value>
    </attribute>
  </structure>
</dictionary>
```

## Polygon dictionaries

Polygon dictionaries allow you to efficiently search for the polygon containing specified points.  
For example: defining a city area by geographical coordinates.

Example configuration:

```

<dictionary>
  <structure>
    <key>
      <name>key</name>
      <type>Array(Array(Array(Float64)))</type>
    </key>

    <attribute>
      <name>name</name>
      <type>String</type>
      <null_value></null_value>
    </attribute>

    <attribute>
      <name>value</name>
      <type>UInt64</type>
      <null_value>0</null_value>
    </attribute>
  </structure>

  <layout>
    <polygon />
  </layout>
</dictionary>

```

The corresponding [DDL-query](#):

```

CREATE DICTIONARY polygon_dict_name (
  key Array(Array(Array(Float64))),
  name String,
  value UInt64
)
PRIMARY KEY key
LAYOUT(POLYGON())
...

```

When configuring the polygon dictionary, the key must have one of two types:

- A simple polygon. It is an array of points.
- MultiPolygon. It is an array of polygons. Each polygon is a two-dimensional array of points. The first element of this array is the outer boundary of the polygon, and subsequent elements specify areas to be excluded from it.

Points can be specified as an array or a tuple of their coordinates. In the current implementation, only two-dimensional points are supported.

The user can [upload their own data](#) in all formats supported by ClickHouse.

There are 3 types of [in-memory storage](#) available:

- POLYGON\_SIMPLE. This is a naive implementation, where a linear pass through all polygons is made for each query, and membership is checked for each one without using additional indexes.
- POLYGON\_INDEX\_EACH. A separate index is built for each polygon, which allows you to quickly check whether it belongs in most cases (optimized for geographical regions).  
Also, a grid is superimposed on the area under consideration, which significantly narrows the number of polygons under consideration.

The grid is created by recursively dividing the cell into 16 equal parts and is configured with two parameters.

The division stops when the recursion depth reaches MAX\_DEPTH or when the cell crosses no more than MIN\_INTERSECTIONS polygons.

To respond to the query, there is a corresponding cell, and the index for the polygons stored in it is accessed alternately.

- POLYGON\_INDEX\_CELL. This placement also creates the grid described above. The same options are available. For each sheet cell, an index is built on all pieces of polygons that fall into it, which allows you to quickly respond to a request.
- POLYGON. Synonym to POLYGON\_INDEX\_CELL.

Dictionary queries are carried out using standard **functions** for working with external dictionaries. An important difference is that here the keys will be the points for which you want to find the polygon containing them.

Example of working with the dictionary defined above:

```
CREATE TABLE points (
    x Float64,
    y Float64
)
...
SELECT tuple(x, y) AS key, dictGet(dict_name, 'name', key), dictGet(dict_name, 'value', key) FROM points ORDER BY x, y;
```

As a result of executing the last command for each point in the 'points' table, a minimum area polygon containing this point will be found, and the requested attributes will be output.

## 内部辞書

ClickHouseには、ジオベースを操作するための組み込み機能が含まれています。

これにより、:

- 地域のIDを使用して、目的の言語でその名前を取得します。
- 地域のIDを使用して、都市、地域、連邦区、国、または大陸のIDを取得します。
- ある地域が別の地域の一部であるかどうかを確認します。
- 親領域のチェーンを取得します。

すべての機能サポート “translocality,” 地域の所有権に関する異なる視点を同時に使用する能力。 詳細については “Functions for working with Yandex.Metrica dictionaries”.

既定のパッケージでは、内部辞書は無効になっています。

よって、`strncasecmp`のパラメータ `path_to_regions_hierarchy_file` と `path_to_regions_names_files` サーバー設定ファイル内。

Geobaseはテキストファイルから読み込まれます。

を置く `regions_hierarchy*.txt` にファイル `path_to_regions_hierarchy_file` ディレクトリ。 この構成パラメータには、`regions_hierarchy.txt` ファイル(既定の地域階層)、およびその他のファイル (`regions_hierarchy_ua.txt`)同じディレクトリにある必要があります。

を置く `regions_names_*.txt` のファイル `path_to_regions_names_files` ディレクトリ。

を作ることもできますこれらのファイル。 ファイル形式は次のとおりです:

`regions_hierarchy*.txt:TabSeparated(ヘッダーなし),列:`

- 地域ID (UInt32)
- 親リージョンID (UInt32)
- 地域タイプ (UInt8):1大陸、3国、4連邦区、5地域、6都市。

- 人口 (UInt32) — optional column

regions\_names\_\*.txt:TabSeparated(ヘッダーなし),列:

- 地域ID (UInt32)
- 地域名 (String) — Can't contain tabs or line feeds, even escaped ones.

フラットアレイは、RAMに格納するために使用されます。このため、Idは百万を超えるべきではありません。

辞書は、サーバーを再起動せずに更新できます。ただし、使用可能な辞書のセットは更新されません。

更新の場合、ファイルの変更時間がチェックされます。ファイルが変更された場合は、辞書が更新されます。

変更をチェックする間隔は、`builtin_dictionaries_reload_interval` パラメータ。

辞書の更新（最初の使用時の読み込み以外）は、クエリをロックしません。更新中、クエリは古いバージョンの辞書を使用します。更新中にエラーが発生すると、エラーはサーバーログに書き込まれ、クエリは古いバージョンの辞書を使用し続けます。

Geobaseで辞書を定期的に更新することをお勧めします。更新中に、新しいファイルを生成し、別の場所に書き込みます。すべての準備ができたら、サーバーが使用するファイルに名前を変更します。

OS識別子とYandexを操作するための機能もあります。Metricaの調査エンジン、しかしそれらは使用されるべきではない。

## データ型

ClickHouseは、表のセルにさまざまな種類のデータを格納できます。

この章ではサポートされているデータの種類と特別な配慮のための利用および/または実施しています。

## UInt8、UInt16、UInt32、UInt64、Int8、Int16、Int32、Int64

符号の有無にかかわらず、固定長の整数。

### Intの範囲

- Int8-[-128:127]
- Int16-[-32768:32767]
- Int32-[-2147483648:2147483647]
- Int64-[-9223372036854775808:9223372036854775807]

### UIntの範囲

- UInt8-[0:255]
- UInt16-[0:65535]
- UInt32-[0:4294967295]
- UInt64-[0:18446744073709551615]

## Float32、Float64

浮動小数点数。

型はCの型と同等です:

- `Float32` - `float`

- `Float64` - `double`

可能な限り整数形式でデータを格納することをお勧めします。たとえば、固定精度の数値を、金額やページの読み込み時間などの整数値にミリ秒単位で変換します。

## 浮動小数点数の使用

- 浮動小数点数を使用した計算では、丸め誤差が生じことがあります。

```
SELECT 1 - 0.9
```

```
minus(1, 0.9) |  
0.0999999999999998 |
```

- 計算の結果は、計算方法（コンピュータシステムのプロセッサタイプおよびアーキテクチャ）に依存する。
- 浮動小数点計算では、無限大などの数値が生成されることがあります (`Inf`) と “not-a-number” (`NaN`)。これは、計算結果を処理するときに考慮する必要があります。
- テキストから浮動小数点数を解析する場合、結果が最も近いマシン表現可能な数値ではない可能性があります。

## `NaN` および `Inf`

標準のSQLとは対照的に、ClickHouseは浮動小数点数の次のカテゴリをサポートしています:

- `Inf` – `Infinity`.

```
SELECT 0.5 / 0
```

```
divide(0.5, 0) |  
inf |
```

- `-Inf` – `Negative infinity`.

```
SELECT -0.5 / 0
```

```
divide(-0.5, 0) |  
-inf |
```

- `NAN` – `Not a number`.

```
SELECT 0 / 0
```

```
divide(0, 0) |  
nan |
```

See the rules for `NaN` sorting in the section [ORDER BY clause](#sql-reference-sql\_reference-statements-select-order-by-md).

## Decimal(P,S), Decimal32(S), Decimal64(S), Decimal128(S)

加算、減算、および乗算の演算中に精度を維持する符号付き固定小数点数。除算の場合、最下位桁数は破棄されます（丸められません）。

### パラメータ

- P-精度。有効な範囲:[1:38]。小数の桁数を決定します(分数を含む)。
- Sスケール 有効な範囲:[0:P]。小数の桁数を指定します。

Pに依存するパラメータ値Decimal(P,S)は、:

-P from[1:9]-For Decimal32(S)  
-P from[10:18]-For Decimal64(S)  
-P from[19:38]-For Decimal128(S)

### 小数点以下の値の範囲

- Decimal32(S) - (-1 \* 10^(9-S), 1 \* 10^(9-S))
- Decimal64(S) - (-1 \* 10^(18-S), 1 \* 10^(18-S))
- Decimal128(S) - (-1 \* 10^(38-S), 1 \* 10^(38-S))

たとえば、Decimal32(4)には、-99999.9999から99999.9999までの0.0001ステップの数値を含めることができます。

### 内部表現

社内データとして表される通常の署名の整数をそれぞれのビット幅になります。メモリに格納できる実際の値の範囲は、上記の指定よりも少し大きく、文字列からの変換時にのみチェックされます。

現代のCPUは128ビット整数をネイティブにサポートしていないため、Decimal128の演算はエミュレートされます。このため、Decimal128はDecimal32/Decimal64よりも大幅に遅く動作します。

### 操作と結果の種類

Decimalのバイナリ演算は、より広い結果の型になります（引数の順序は任意です）。

- Decimal64(S1) <op> Decimal32(S2) -> Decimal64(S)
- Decimal128(S1) <op> Decimal32(S2) -> Decimal128(S)
- Decimal128(S1) <op> Decimal64(S2) -> Decimal128(S)

スケールのルール:

- 加算、減算: S=max (S1、S2)。
- multiply:S=S1+S2.
- 除算:S=S1.

Decimalと整数の間の同様の演算の場合、結果は引数と同じサイズのDecimalになります。

DecimalとFloat32/Float64の間の演算は定義されていません。必要な場合は、toDecimal32、toDecimal64、toDecimal128、またはtoFloat32、toFloat64ビルトインを使用して引数のいずれかを明示的にキャストできます。結果は精度が失われ、型変換は計算コストがかかる操作であることに注意してください。

Decimalの一部の関数はFloat64(varやstddevなど)として結果を返します。これにより、Float64入力とDecimal入力が同じ値で異なる結果になる可能性があります。

## オーバーフロー一チェック

中計算は小数、整数であふれかが起こる。小数部の過剰な数字は破棄されます（丸められません）。整数部分の数字が過剰になると、例外が発生します。

```
SELECT toDecimal32(2, 4) AS x, x / 3
```

x	divide(toDecimal32(2, 4), 3)
2.0000	0.6666

```
SELECT toDecimal32(4.2, 8) AS x, x * x
```

```
DB::Exception: Scale is out of bounds.
```

```
SELECT toDecimal32(4.2, 8) AS x, 6 * x
```

```
DB::Exception: Decimal math overflow.
```

オーバーフロー一チェックが業務に減速した。が知られていることができませんことを無効にするチェック decimal\_check\_overflow 設定。時チェックを無効とオーバーフローが起こり、結果は正しくあり:

```
SET decimal_check_overflow = 0;
SELECT toDecimal32(4.2, 8) AS x, 6 * x
```

x	multiply(6, toDecimal32(4.2, 8))
4.20000000	-17.74967296

オーバーフロー一チェックが起こるだけでなく演算業務はもとより、価値との比較:

```
SELECT toDecimal32(1, 8) < 100
```

```
DB::Exception: Can't compare.
```

## ブール値

ブール値には別の型はありません。UInt8型を使用し、値0または1に制限します。

## 文字列

任意の長さの文字列。長さは限定されない。値には、nullバイトを含む任意のバイトセットを含めることができます。文字列型は、他のDbmsのVARCHAR型、BLOB型、CLOB型などを置き換えます。

## エンコード

ClickHouseにはエンコーディングの概念はありません。文字列には、任意のバイトセットを含めることができます。が必要な場合は店舗テキストの使用をお勧めしまUTF-8エンコーディングです。少なくとも、端末がUTF-8（推奨）を使用している場合は、変換を行わずに値を読み書きできます。

同様に、文字列を操作するための特定の関数には、UTF-8でエンコードされたテキストを表すバイトセットが文字列に含まれているという前提の下で

例えば、「length」関数は、文字列の長さをバイト単位で計算します。「lengthUTF8」関数は、値がUTF-8でエンコードされていると仮定して、Unicodeコードポイントで文字列の長さを計算します。

## Fixedstring

固定長の文字列 **N** バイト(文字もコードポイントもない)。

の列を宣言するには **FixedString** 次の構文を使用します:

```
<column_name> FixedString(N)
```

どこに **N** は自然数である。

その **FixedString** データが正確に長さを持つ場合、型は効率的です **N** バイト数。他のすべてのケースでは、効率を低下させる可能性があります。

効率的に格納できる値の例 **FixedString**-型指定された列:

- IPアドレスのバイナリ表現 (**FixedString(16)** IPv6の場合)。
- Language codes (ru\_RU, en\_US ...).
- Currency codes (USD, RUB ...).
- ハッシュのバイナリ表現 (**FixedString(16)** MD5の場合, **FixedString(32)** SHA256) のため。

UUID値を格納するには、**UUID** データ型。

データを挿入するとき、ClickHouse:

- 文字列が含まれている数がnullバイトの文字列を補完します **N** バイト数。
- スロー **Too large value for FixedString(N)** 文字列に以下のものが含まれている場合は例外 **N** バイト数。

データを選択すると、ClickHouseは文字列の末尾にあるnullバイトを削除しません。を使用する場合 **WHERE** この節では、nullバイトを手動で追加する必要があります。**FixedString** 値。次の例では、次の操作を実行する方法を示します **WHERE** との句 **FixedString**。

次の表を单一のもので考えてみましょう **FixedString(2)** 列:

name
b

クエリ **SELECT \* FROM FixedStringTable WHERE a = 'b'** 結果としてデータを返しません。このフィルターパターンはnullバイトまでとなります。

```
SELECT * FROM FixedStringTable  
WHERE a = 'b\0'
```

a
b

この動作はMySQLとは異なります。CHAR type(文字列はスペースで埋められ、スペースは出力のために削除されます)。

の長さに注意してください。FixedString(N) 値は一定です。その 長さ 関数の戻り値 N たとえ FixedString(N) 値はnullバイトのみで埋められますが、空 関数の戻り値 1 この場合。

## UUID

普遍一意識別子(UUID)は、レコードを識別するために使用される16バイトの番号です。UUIDの詳細については、以下を参照してください [Wikipedia](#).

UUID型の値の例を以下に示します:

```
61f0c404-5cb3-11e7-907b-a6006ad3dba0
```

新しいレコードを挿入するときにUUID列の値を指定しない場合、UUID値はゼロで埋められます:

```
00000000-0000-0000-0000-000000000000
```

## 生成する方法

Uuid値を生成するために、ClickHouseは [generateuidv4](#) 機能。

## 使用例

### 例1

この例では、UUID型の列を持つテーブルを作成し、テーブルに値を挿入する方法を示します。

```
CREATE TABLE t_uuid (x UUID, y String) ENGINE=TinyLog
```

```
INSERT INTO t_uuid SELECT generateUUIDv4(), 'Example 1'
```

```
SELECT * FROM t_uuid
```

x	y
417ddc5d-e556-4d27-95dd-a34d84e46a50	Example 1

### 例2

この例では、新しいレコードを挿入するときにUUID列の値は指定されません。

```
INSERT INTO t_uuid (y) VALUES ('Example 2')
```

```
SELECT * FROM t_uuid
```

x	y
417ddc5d-e556-4d27-95dd-a34d84e46a50	Example 1
00000000-0000-0000-0000-000000000000	Example 2

## 制限

UUIDデータ型は、以下の関数のみをサポートします 文字列 データ型もサポートしています(例、分、最大, and カウント).

Uuidデータ型は、算術演算ではサポートされません(たとえば, abs) または、以下のような集計関数 和 と avg.

## 日付

日付型です。1970-01-01からの日数が2バイトの符号なし整数として格納されます。UNIX時間の開始直後から、変換段階で定数として定義される上限しきい値までの値を格納できます（現在は2106年までですが、一年分を完全にサポートしているのは2105年までです）。

日付値は、タイムゾーンなしで格納されます。

## Date32

A date. Supports the date range same with [Datetime64](#). Stored in four bytes as the number of days since 1925-01-01. Allows storing values till 2283-11-11.

### Examples

Creating a table with a Date32-type column and inserting data into it:

```
CREATE TABLE new
(
    `timestamp` Date32,
    `event_id` UInt8
)
ENGINE = TinyLog;
```

```
INSERT INTO new VALUES (4102444800, 1), ('2100-01-01', 2);
SELECT * FROM new;
```

timestamp	event_id
2100-01-01	1
2100-01-01	2

### See Also

- [toDate32](#)
- [toDate32OrZero](#)
- [toDate32OrNull](#)

# Datetime

カレンダーの日付と一日の時間として表現することができ、時間内にインスタンスを格納することができます。

構文:

```
DateTime([timezone])
```

サポートされる値の範囲: [1970-01-01 00:00:00, 2105-12-31 23:59:59].

解像度:1秒.

## 使用上の注意

時間のポイントはaとして救われます **Unix タイムスタン** タイムゾーンまたは夏時間に関係なく。さらに、**DateTime** 型は、列全体で同じタイムゾーンを格納することができます。**DateTime** 型の値はテキスト形式で表示され、文字列として指定された値がどのように解析されるか ('2020-01-01 05:00:01'). タイムゾーンは、テーブルの行(またはresultset)には格納されませんが、列メタデータに格納されます。

サポートされているタイムゾーンのリストは、[IANA タイムゾーンデータベース](#).

その **tzdata** パッケージ, 含む [IANA タイムゾーンデータベース](#)、システムに取付けられているべきです。 使用する **timedatectl list-timezones** ローカルシステムが既知のタイムゾーンを一覧表示するコマンド。

タイムゾーンを明示的に設定することができます **DateTime**-テーブルを作成するときに列を入力します。 タイムゾーンが設定されていない場合、ClickHouseは **タイムゾーン** ClickHouseサーバー起動時のサーバー設定またはオペレーティングシステム設定のパラメータ。

その [clickhouse-クライアント](#) データ型の初期化時にタイムゾーンが明示的に設定されていない場合は、既定でサーバータイムゾーンを適用します。 クライアントのタイムゾーンを使用するには **clickhouse-client** と **--use\_client\_time\_zone** パラメータ。

ClickHouseは、次の値を出力します YYYY-MM-DD hh:mm:ss デフォルトのテキスト形式。 出力を変更するには **formatDateTime** 機能。

ClickHouseにデータを挿入するときは、データの値に応じて、日付と時刻の文字列の異なる形式を使用できます。  
**date\_time\_input\_format** 設定。

## 例

1. テーブルの作成 **DateTime**-列を入力し、そこにデータを挿入する:

```
CREATE TABLE dt
(
    `timestamp` DateTime('Europe/Moscow'),
    `event_id` UInt8
)
ENGINE = TinyLog;
```

```
INSERT INTO dt Values (1546300800, 1), ('2019-01-01 00:00:00', 2);
```

```
SELECT * FROM dt;
```

timestamp	event_id
2019-01-01 03:00:00	1
2019-01-01 00:00:00	2

- Datetimeを整数として挿入する場合は、Unix Timestamp(UTC)として扱われます。1546300800を表す'2019-01-01 00:00:00' UTCしかし、timestamp列はEurope/Moscow(UTC+3)タイムゾーンが指定されている場合、文字列として出力すると、値は次のように表示されます'2019-01-01 03:00:00'
- 文字列値をdatetimeとして挿入すると、列タイムゾーンにあるものとして扱われます。'2019-01-01 00:00:00'であるとして扱われますEurope/Moscowタイムゾーンとして保存1546290000.

## 2. フィルタリング DateTime 値

```
SELECT * FROM dt WHERE timestamp = toDateTime('2019-01-01 00:00:00', 'Europe/Moscow')
```

timestamp	event_id
2019-01-01 00:00:00	2

DateTime列の値は、文字列の値を使用してフィルター処理できます。WHERE述語。に変換されますDateTime自動的に:

```
SELECT * FROM dt WHERE timestamp = '2019-01-01 00:00:00'
```

timestamp	event_id
2019-01-01 03:00:00	1

## 3. Aのタイムゾーンの取得 DateTime-タイプ列:

```
SELECT toDateTime(now(), 'Europe/Moscow') AS column, toTypeName(column) AS x
```

column	x
2019-10-16 04:12:04	DateTime('Europe/Moscow')

## 4. タイムゾーン変換

```
SELECT
toDateTime(timestamp, 'Europe/London') as lon_time,
toDateTime(timestamp, 'Europe/Moscow') as mos_time
FROM dt
```

lon_time	mos_time
2019-01-01 00:00:00	2019-01-01 03:00:00
2018-12-31 21:00:00	2019-01-01 00:00:00

も参照。

- 型変換関数
- 日付と時刻を操作するための関数

- 配列を操作するための関数
- その `date_time_input_format` 設定
- その `timezone` サーバ構成パラメータ
- 日付と時刻を操作する演算子
- その `Date` データ型

## Datetime64

定義された秒以下の精度で、カレンダーの日付と時刻として表すことができるインスタンスを時間内に格納することができます

目盛りのサイズ（精密）:  $10^{-\text{精密}}$  秒

構文:

```
DateTime64(precision, [timezone])
```

内部的には、データを ‘ticks’ エポック開始（1970-01-01 00:00:00UTC）以来、Int64として。目盛りの解像度は、精度パラメータによって決定されます。さらに、`DateTime64` 型は、列全体で同じタイムゾーンを格納することができます。`DateTime64` 型の値はテキスト形式で表示され、文字列として指定された値がどのように解析されるか ('2020-01-01 05:00:01.000'). タイムゾーンは、テーブルの行(またはresultset)には格納されませんが、列メタデータに格納されます。詳細はを参照。[DateTime](#).

### 例

1. テーブルの作成 `DateTime64`-列を入力し、そこにデータを挿入する:

```
CREATE TABLE dt
(
    `timestamp` DateTime64(3, 'Europe/Moscow'),
    `event_id` UInt8
)
ENGINE = TinyLog
```

```
INSERT INTO dt Values (1546300800000, 1), ('2019-01-01 00:00:00', 2)
```

```
SELECT * FROM dt
```

timestamp	event_id
2019-01-01 03:00:00.000	1
2019-01-01 00:00:00.000	2

- Datetimeを整数として挿入する場合、適切にスケーリングされたUnixタイムスタンプ(UTC)として扱われます。  
1546300800000（精度3で）を表します '2019-01-01 00:00:00' UTC しかし、`timestamp` 列は Europe/Moscow (UTC+3) タイムゾーンが指定されている場合、文字列として出力すると、値は次のように表示されます '2019-01-01 03:00:00'
- 文字列値をdatetimeとして挿入すると、列タイムゾーンにあるものとして扱われます。'2019-01-01 00:00:00' であるとして扱われます Europe/Moscow タイムゾーンとして保存 1546290000000.

## 2. フィルタリング DateTime64 値

```
SELECT * FROM dt WHERE timestamp = toDateTime64('2019-01-01 00:00:00', 3, 'Europe/Moscow')
```

timestamp	event_id
2019-01-01 00:00:00.000	2

異なり DateTime, DateTime64 値は変換されません String 自動的に

## 3. Aのタイムゾーンの取得 DateTime64-タイプ値:

```
SELECT toDateTime64(now(), 3, 'Europe/Moscow') AS column, toTypeName(column) AS x
```

column	x
2019-10-16 04:12:04.000	DateTime64(3, 'Europe/Moscow')

## 4. タイムゾーン変換

```
SELECT
toDateTime64(timestamp, 3, 'Europe/London') as lon_time,
toDateTime64(timestamp, 3, 'Europe/Moscow') as mos_time
FROM dt
```

lon_time	mos_time
2019-01-01 00:00:00.000	2019-01-01 03:00:00.000
2018-12-31 21:00:00.000	2019-01-01 00:00:00.000

も参照。

- 型変換関数
- 日付と時刻を操作するための関数
- 配列を操作するための関数
- その date\_time\_input\_format 設定
- その timezone サーバ構成パラメータ
- 日付と時刻を操作する演算子
- Date データ型
- DateTime データ型

## Enum

名前付き値で構成される列挙型。

名前の値として宣言された 'string' = integer ペア。 ClickHouseは数値のみを格納しますが、名前による値の操作をサポートします。

ClickHouseサポート：

■ 8ビット `Enum`. それはで列挙される256までの値を含むことができます [-128, 127] 範囲

■ 16ビット `Enum`. それはで列挙される65536までの値を含むことができます [-32768, 32767] 範囲

ClickHouseは自動的に次のタイプを選択します `Enum` データが挿入されるとき。また、`Enum8` または `Enum16` ストレージのサイズを確認するタイプ。

## 使用例

ここでは、`Enum8('hello' = 1, 'world' = 2)` タイプ列:

```
CREATE TABLE t_enum
(
    x Enum('hello' = 1, 'world' = 2)
)
ENGINE = TinyLog
```

列 `x` 型定義にリストされている値のみを格納できます: 'hello' または 'world'. 他の値を保存しようとすると、ClickHouse は例外を発生させます。このための8ビットサイズ `Enum` 自動的に選択されます。

```
INSERT INTO t_enum VALUES ('hello'), ('world'), ('hello')
```

Ok.

```
INSERT INTO t_enum values('a')
```

Exception on client:

Code: 49. DB::Exception: Unknown element 'a' for type Enum('hello' = 1, 'world' = 2)

テーブルからデータを照会すると、ClickHouseは文字列値を `Enum`.

```
SELECT * FROM t_enum
```

x
hello
world
hello

行の等価な数値を見る必要がある場合は、次のようにキャストする必要があります `Enum` 整数型の値。

```
SELECT CAST(x, 'Int8') FROM t_enum
```

CAST(x, 'Int8')
1
2
1

クエリで `Enum` 値を作成するには、次のものも使用する必要があります `CAST`.

```
SELECT toTypeName(CAST('a', 'Enum('a\'' = 1, \'b\'' = 2)''))
```

```
toTypeName(CAST('a', 'Enum(\`a\` = 1, \`b\` = 2)'))  
Enum8('a' = 1, 'b' = 2)
```

## 一般的なルールと使用法

各値には、範囲内の数値が割り当てられます `-128 ... 127` のために `Enum8` または範囲 `-32768 ... 32767` のために `Enum16`。すべての文字列と数字は異なる必要があります。空の文字列を使用できます。この型が(テーブル定義で)指定されている場合、数値は任意の順序にすることができます。しかし、順序は重要ではありません。

の文字列も数値もない。`Enum` ことができます `NULL`。

アン `Enum` に含めることができます `Null可能` タイプ。だから、クエリを使用してテーブルを作成する場合

```
CREATE TABLE t_enum_nullable  
(  
    x Nullable( Enum8('hello' = 1, 'world' = 2) )  
)  
ENGINE = TinyLog
```

できるアプリ `'hello'` と `'world'` でも `NULL` 同様に。

```
INSERT INTO t_enum_nullable Values('hello'), ('world'), (NULL)
```

RAMでは、`Enum` 列は次のように格納されます `Int8` または `Int16` 対応する数値の。

テキスト形式で読み込む場合、ClickHouseは値を文字列として解析し、列挙型の値のセットから対応する文字列を検索します。見つからない場合は、例外がスローされます。テキスト形式で読み込むと、文字列が読み込まれ、対応する数値が検索されます。見つからない場合は例外がスローされます。

テキスト形式で書き込む場合、その値を対応する文字列として書き込みます。列データにガベージ(有効なセット以外の数値)が含まれている場合は、例外がスローされます。バイナリ形式で読み書きする場合、`Int8`および`Int16`データ型と同じように動作します。

暗黙的なデフォルト値は、数値が最も小さい値です。

中 `ORDER BY`, `GROUP BY`, `IN`, `DISTINCT` というように、列挙型は対応する数値と同じように動作します。たとえば、`ORDER BY`は数値で並べ替えます。等価演算子と比較演算子は、列挙型では、基になる数値と同じように動作します。

列挙型の値を数値と比較することはできません。列挙型は定数文字列と比較できます。比較対象の文字列が列挙型の有効な値でない場合は、例外がスローされます。`IN`演算子は、左側の列挙型と右側の文字列のセットでサポートされています。文字列は、対応する列挙型の値です。

Most numeric and string operations are not defined for `Enum` values, e.g. adding a number to an `Enum` or concatenating a string to an `Enum`.

しかし、`Enum`には自然なものがあります `toString` 文字列値を返す関数。

列挙型の値は、次の式を使用して数値型にも変換できます `toT` ここで、Tは数値型です。Tが列挙型の基になる数値型に対応する場合、この変換はゼロコストになります。

列挙型は、値のセットのみが変更されている場合、`ALTER`を使用してコストなしで変更できます。`ALTER`を使用して列挙型のメンバーを追加および削除することは可能です（削除された値がテーブルで使用されていない場合にのみ削除が安全です）。セーフガードとして、以前に定義された列挙型メンバーの数値を変更すると、例外がスローされます。

`ALTER`を使用すると、`Int8`を`Int16`に変更するのと同じように、`Enum8`を`Enum16`に変更することも、その逆も可能です。

## 配列(t)

の配列 T-タイプ項目。T 配列を含む任意のデータ型を指定できます。

## 配列の作成

関数を使用して配列を作成できます:

```
array(T)
```

角括弧を使用することもできます。

```
[]
```

配列の作成例:

```
SELECT array(1, 2) AS x, toTypeName(x)
```

```
x-----toTypeName(array(1, 2))-----  
[1,2] | Array(UInt8) |
```

```
SELECT [1, 2] AS x, toTypeName(x)
```

```
x-----toTypeName([1, 2])-----  
[1,2] | Array(UInt8) |
```

## データ型の操作

オンザフライで配列を作成するとき、ClickHouseは、リストされたすべての引数を格納できる最も狭いデータ型として引数型を自動的に定義します。 いずれかがある場合 Null可能 またはリテラル NULL 値を指定すると、配列要素の型も次のようにになります Null可能.

ClickHouseがデータ型を判別できなかった場合、例外が生成されます。 たとえば、これは文字列と数値を同時に配列を作成しようとするときに発生します (SELECT array(1, 'a')).

自動データ型検出の例:

```
SELECT array(1, 2, NULL) AS x, toTypeName(x)
```

```
x-----toTypeName(array(1, 2, NULL))-----  
[1,2,NULL] | Array(Nullable(UInt8)) |
```

互換性のないデータ型の配列を作成しようとすると、ClickHouseは例外をスローします:

```
SELECT array(1, 'a')
```

Received exception from server (version 1.1.54388):

Code: 386. DB::Exception: Received from localhost:9000, 127.0.0.1. DB::Exception: There is no supertype for types UInt8, String because some of them are String/FixedString and some of them are not.

# AggregateFunction(name, types\_of\_arguments...)

Aggregate functions can have an implementation-defined intermediate state that can be serialized to an `AggregateFunction(...)` data type and stored in a table, usually, by means of マテリアライズドビュー。集計関数の状態を生成する一般的な方法は、集計関数を呼び出すことです。-State接尾辞。将来集計の最終結果を取得するには、同じ集計関数を使用する必要があります。-Merge接尾辞。

`AggregateFunction` — parametric data type.

パラメータ

- 集計関数の名前。

If the function is parametric, specify its parameters too.

- 集計関数の引数の型。

例

```
CREATE TABLE t
(
    column1 AggregateFunction(uniq, UInt64),
    column2 AggregateFunction(anyIf, String, UInt8),
    column3 AggregateFunction(quantiles(0.5, 0.9), UInt64)
) ENGINE = ...
```

`uniq`,`anyIf` (任意+もし)と 分位数 ClickHouseでサポートされている集計関数です。

## 使用法

### データ挿入

データを挿入するには、`INSERT SELECT` 総計を使って -State- 機能。

関数の例

```
uniqState(UserID)
quantilesState(0.5, 0.9)(SendTiming)
```

対応する機能とは対照的に `uniq` と `quantiles`, -State- 関数は、最終的な値の代わりに状態を返します。言い換えれば、彼らの値を返します `AggregateFunction` タイプ。

の結果 `SELECT` クエリ、の値 `AggregateFunction` typeは、すべてのClickHouse出力形式に対して実装固有のバイナリ表現を持ちます。たとえば、データをダンプする場合、`TabSeparated` フォーマット `SELECT` このダンプは、以下を使用してロードバックできます `INSERT` クエリ。

### データ選択

データを選択するとき `AggregatingMergeTree` テーブル、使用 `GROUP BY` データを挿入するときと同じ集計関数ですが、-Merge接尾辞。

を持つ集合関数 -Merge suffixは、状態のセットを取得し、それらを結合し、完全なデータ集計の結果を返します。

たとえば、次の二つのクエリは同じ結果を返します:

```
SELECT uniq(UserID) FROM table
```

```
SELECT uniqMerge(state) FROM (SELECT uniqState(UserID) AS state FROM table GROUP BY RegionID)
```

## 使用例

見る [AggregatingMergeTree エンジンの説明。](#)

## Tuple(t1, T2, ...)

要素のタプル、各個人を持つ [タイプ](#)。

組は、一時的な列のグループ化に使用されます。列は、`in`式がクエリで使用され、`lambda`関数の特定の仮パラメータを指定するときにグループ化できます。詳細については [演算子](#) と [高次関数](#)。

タプルは、クエリの結果になります。この場合、JSON以外のテキスト形式の場合、値は角かっこでカンマ区切られます。JSON形式では、タプルは配列として出力されます（角括弧で囲みます）。

### タプルの作成

関数を使用してタプルを作成できます：

```
tuple(T1, T2, ...)
```

タプルの作成例：

```
SELECT tuple(1,'a') AS x, toTypeName(x)
```

```
x-----toTypeName(tuple(1, 'a'))-----  
(1,'a') | Tuple(UInt8, String) |
```

### データ型の操作

オンザフライでタプルを作成するとき、ClickHouseは自動的に各引数の型を引数値を格納できる型の最小値として検出します。引数が `NULL` タプル要素の型は [Null可能](#)。

自動データ型検出の例：

```
SELECT tuple(1, NULL) AS x, toTypeName(x)
```

```
x-----toTypeName(tuple(1, NULL))-----  
(1,NULL) | Tuple(UInt8, Nullable(Nothing)) |
```

## Nullable(型名)

できる特別マークー (`NULL`) を表す。“missing value”と共に正常値を許可する `TypeName`。たとえば、`Nullable(Int8)` 型列が格納できます `Int8` 値を入力し、値を持たない行には格納されます `NULL`。

のために `TypeName` 複合データ型は使用できません [配列](#) と [タプル](#)。複合データ型には `Nullable` 次のような型の値 `Array(Nullable(Int8))`。

A `Nullable type` フィールドできない含まれてテーブルスを作成します。

`NULL` のデフォルト値です `Nullable ClickHouse` サーバー構成で特に指定がない限り、入力します。

## ストレージ機能

保存するには `Nullable` テーブルの列に値を入力すると、ClickHouseは別のファイルを使用します `NULL` 値を持つ通常のファイルに加えて、マスク。マスクファイル内のエントリはClickHouseが `NULL` テーブル行ごとに対応するデータ型のデフォルト値。追加のファイルのために、`Nullable column`は、同様の通常のものと比較して追加の記憶領域を消費します。

### 注

を使用して `Nullable` ほとんどの場合、パフォーマンスに悪影響を及ぼします。

## 使用例

```
CREATE TABLE t_null(x Int8, y Nullable(Int8)) ENGINE TinyLog
```

```
INSERT INTO t_null VALUES (1, NULL), (2, 3)
```

```
SELECT x + y FROM t_null
```

```
plus(x, y)
  └── NULL
    └── 5
```

## 入れ子データ構造

### Nested(name1 Type1, Name2 Type2, ...)

A nested data structure is like a table inside a cell. The parameters of a nested data structure – the column names and types – are specified the same way as in a `CREATE TABLE` クエリ。各テーブルの行は、入れ子になったデータ構造内の任意の数の行に対応できます。

例:

```

CREATE TABLE test.visits
(
    CounterID UInt32,
    StartDate Date,
    Sign Int8,
    IsNew UInt8,
    VisitID UInt64,
    UserID UInt64,
    ...
    Goals Nested
    (
        ID UInt32,
        Serial UInt32,
        EventTime DateTime,
        Price Int64,
        OrderID String,
        CurrencyID UInt32
    ),
    ...
) ENGINE = CollapsingMergeTree(StartDate, intHash32(UserID), (CounterID, StartDate, intHash32(UserID), VisitID),
8192, Sign)

```

この例では、Goals コンバージョンに関するデータを含む入れ子になったデータ構造。各行 ‘visits’ tableは、ゼロまたは任意の数の変換に対応できます。

単一の入れ子レベルのみがサポートされます。配列を含む入れ子になった構造体の列は多次元配列と同等なので、サポートが限られています（MergeTreeエンジンでこれらの列をテーブルに格納するサポートは

ほとんどの場合、入れ子になったデータ構造を操作する場合、その列はドットで区切られた列名で指定されます。これらの列は、一致する型の配列を構成します。単一の入れ子になったデータ構造のすべての列配列の長さは同じです。

例：

```

SELECT
    Goals.ID,
    Goals.EventTime
FROM test.visits
WHERE CounterID = 101500 AND length(Goals.ID) < 5
LIMIT 10

```



ネストされたデータ構造は、同じ長さの複数の列配列のセットと考えるのが最も簡単です。

SELECTクエリで個々の列ではなく入れ子になったデータ構造全体の名前を指定できるのは、ARRAY JOIN句だけです。詳細については、“ARRAY JOIN clause”. 例：

```
SELECT
    Goal.ID,
    Goal.EventTime
FROM test.visits
ARRAY JOIN Goals AS Goal
WHERE CounterID = 101500 AND length(Goals.ID) < 5
LIMIT 10
```

Goal.ID	Goal.EventTime
1073752	2014-03-17 16:38:10
591325	2014-03-17 16:38:48
591325	2014-03-17 16:42:27
1073752	2014-03-17 00:28:25
1073752	2014-03-17 10:46:20
1073752	2014-03-17 13:59:20
591325	2014-03-17 22:17:55
591325	2014-03-17 22:18:07
591325	2014-03-17 22:18:51
1073752	2014-03-17 11:37:06

入れ子になったデータ構造全体に対してSELECTは実行できません。明示的にリストできるのは、その一部である個々の列のみです。

INSERTクエリでは、入れ子になったデータ構造のすべてのコンポーネント列配列を個別に渡す必要があります(個々の列配列と同様に)。挿入時に、同じ長さであることがチェックされます。

DESCRIBEクエリの場合、入れ子になったデータ構造の列は、同じ方法で別々に表示されます。

入れ子になったデータ構造内の要素のALTERクエリには制限があります。

## 特殊データ型

特別なデータ型の値は、テーブルへの保存やクエリ結果の出力のためにシリアル化することはできませんが、クエリ実行時の中間結果として使用でき

### 式

式は、高次関数のラムダを表すために使用されます。

### セット

の右半分に使用されます IN 式。

### 何もない

このデータ型の唯一の目的は、値が期待されないケースを表すことです。だから、作成することはできません Nothing タイプ値。

たとえば、リテラル NULL タイプをの有する Nullable(Nothing). 詳細はこちら Null可能.

その Nothing 型は空の配列を表すためにも使用できます:

```
SELECT toTypeName(array())
```

```
toTypeName(array())  
Array(Nothing)
```

## 間隔

時間と日付間隔を表すデータ型のファミリ。の結果の型 **INTERVAL** オペレーター

### 警告

Interval データ型の値はテーブルに格納できません。

構造:

- 符号なし整数值としての時間間隔。
- 間隔のタイプ。

サポートさ:

- SECOND
- MINUTE
- HOUR
- DAY
- WEEK
- MONTH
- QUARTER
- YEAR

間隔タイプごとに、個別のデータタイプがあります。例えば、DAY 区間は IntervalDay データ型:

```
SELECT toTypeName(INTERVAL 4 DAY)
```

```
toTypeName(toIntervalDay(4))  
IntervalDay
```

## 使用上の注意

以下を使用できます Interval-算術演算の値を **日付** と **DateTime**-値を入力します。たとえば、現在の時刻に4日を追加できます:

```
SELECT now() as current_date_time, current_date_time + INTERVAL 4 DAY
```

```
current_date_time plus(now(), toIntervalDay(4))  
2019-10-23 10:58:45 | 2019-10-27 10:58:45 |
```

異なるタイプの間隔は結合できません。次のような間隔は使用できません **4 DAY 1 HOUR**。間隔は、間隔など、間隔の最小単位より小さい単位または等しい単位で指定します **1 day and an hour** 間隔は次のように表現できます **25 HOUR** または **90000 SECOND**。

次の操作で算術演算を実行することはできません **Interval**-値を入力しますが、結果として異なるタイプの間隔を値に追加することができます。 **Date** または **DateTime** データ型。例えば:

```
SELECT now() AS current_date_time, current_date_time + INTERVAL 4 DAY + INTERVAL 3 HOUR
```

```
current_date_time plus(plus(now(), toIntervalDay(4)), toIntervalHour(3))  
2019-10-23 11:16:28 | 2019-10-27 14:16:28 |
```

次のクエリでは、例外が発生します:

```
select now() AS current_date_time, current_date_time + (INTERVAL 4 DAY + INTERVAL 3 HOUR)
```

Received exception from server (version 19.14.1):

Code: 43. DB::Exception: Received from localhost:9000. DB::Exception: Wrong argument types for function plus: if one argument is Interval, then another must be Date or DateTime..

も参照。

- **INTERVAL** 演算子
- **toInterval** 型変換関数

## ドメイン

ドメインは、既存の基本型の上にいくつかの余分な機能を追加する特殊な目的の型ですが、基になるデータ型のオンウェイやおよびオンディスク形式は現時点では、ClickHouseはユーザー定義ドメインをサポートしていません。

たとえば、対応する基本タイプを使用できる任意の場所でドメインを使用できます:

- ドメイン型の列を作成する
- ドメイン列からへの読み取り/書き込み値
- 基本型をインデックスとして使用できる場合は、インデックスとして使用します
- ドメイン列の値を持つ関数の呼び出し

### ドメインの追加機能

- 明示的な列タイプ名 **SHOW CREATE TABLE** または **DESCRIBE TABLE**
- 人間に優しいフォーマットからの入力 **INSERT INTO domain\_table(domain\_column) VALUES(...)**
- 人間に優しいフォーマットへの出力 **SELECT domain\_column FROM domain\_table**
- 人間に優しい形式で外部ソースからデータを読み込む: **INSERT INTO domain\_table FORMAT CSV ...**

### 制限

- 基本型のインデックス列をドメイン型に変換できません **ALTER TABLE**.
- 別の列または表からデータを挿入するときに、文字列値を暗黙的にドメイン値に変換できません。

- ドメインは、格納された値に制約を追加しません。

## IPv4

IPv4 に基づくドメインです UInt32 IPv4 値を格納するための型指定された置換として機能します。それは点検で人間に適する入出力フォーマットおよびコラムのタイプ情報を密集した貯蔵に与える。

### 基本的な使用法

```
CREATE TABLE hits (url String, from IPv4) ENGINE = MergeTree() ORDER BY url;
DESCRIBE TABLE hits;
```

name	type	default_type	default_expression	comment	codec_expression
url	String				
from	IPv4				

または、IPv4 ドメインをキーとして使用できます:

```
CREATE TABLE hits (url String, from IPv4) ENGINE = MergeTree() ORDER BY from;
```

IPv4 ドメインはIPv4文字列としてカスタム入力形式をサポート:

```
INSERT INTO hits (url, from) VALUES ('https://wikipedia.org', '116.253.40.133')('https://clickhouse.com', '183.247.232.58')('https://clickhouse.com/docs/en/', '116.106.34.242');
SELECT * FROM hits;
```

url	from
https://clickhouse.com/docs/en/	116.106.34.242
https://wikipedia.org	116.253.40.133
https://clickhouse.com	183.247.232.58

値が格納されコンパクトにバイナリ形式:

```
SELECT toTypeName(from), hex(from) FROM hits LIMIT 1;
```

toTypeName(from)	hex(from)
IPv4	B7F7E83A

ドメイン値は、暗黙的に型以外に変換できません UInt32.

変換したい場合 IPv4 文字列への値は、明示的にそれを行う必要があります `IPv4NumToString()` 関数:

```
SELECT toTypeName(s), IPv4NumToString(from) as s FROM hits LIMIT 1;
```

toTypeName(IPv4NumToString(from))	s
String	183.247.232.58

または a にキャスト UInt32 値:

```
SELECT toTypeName(i), CAST(from as UInt32) as i FROM hits LIMIT 1;
```

toTypeName(	CAST(from, 'UInt32')	)	i
UInt32	3086477370		

## IPv6

IPv6 に基づくドメインです `FixedString(16)` IPv6 値を格納するための型指定された置換として機能します。それは点検で人間に適する入出力フォーマットおよびコラムのタイプ情報を密集した貯蔵に与える。

### 基本的な使用法

```
CREATE TABLE hits (url String, from IPv6) ENGINE = MergeTree() ORDER BY url;
```

```
DESCRIBE TABLE hits;
```

name	type	default_type	default_expression	comment	codec_expression
url	String				
from	IPv6				

または、IPv6 キーとしてのドメイン：

```
CREATE TABLE hits (url String, from IPv6) ENGINE = MergeTree() ORDER BY from;
```

IPv6 ドメイン：

```
INSERT INTO hits (url, from) VALUES ('https://wikipedia.org', '2a02:aa08:e000:3100::2')('https://clickhouse.com', '2001:44c8:129:2632:33:0:252:2')('https://clickhouse.com/docs/en/', '2a02:e980:1e::1');
```

```
SELECT * FROM hits;
```

url	from
https://clickhouse.com	2001:44c8:129:2632:33:0:252:2
https://clickhouse.com/docs/en/	2a02:e980:1e::1
https://wikipedia.org	2a02:aa08:e000:3100::2

値が格納されコンパクトにバイナリ形式：

```
SELECT toTypeName(from), hex(from) FROM hits LIMIT 1;
```

toTypeName(from)	hex(from)
IPv6	200144C8012926320033000002520002

ドメイン値は、暗黙的に型以外に変換できません `FixedString(16)`。

変換したい場合 IPv6 文字列への値は、明示的にそれを行う必要があります `IPv6NumToString()` 関数：

```
SELECT toTypeName(s), IPv6NumToString(from) as s FROM hits LIMIT 1;
```

```
toTypeName(IPv6NumToString(from))---s---  
String | 2001:44c8:129:2632:33:0:252:2 |
```

またはaにキャスト FixedString(16) 値:

```
SELECT toTypeName(i), CAST(from as FixedString(16)) as i FROM hits LIMIT 1;
```

```
toTypeName(CAST(from, 'FixedString(16)'))---i---  
FixedString(16) | ♦♦♦ |
```

## Multiword Types

When creating tables, you can use data types with a name consisting of several words. This is implemented for better SQL compatibility.

## Multiword Types Support

Multiword types	Simple types
DOUBLE PRECISION	Float64
CHAR LARGE OBJECT	String
CHAR VARYING	String
CHARACTER LARGE OBJECT	String
CHARACTER VARYING	String
NCHAR LARGE OBJECT	String
NCHAR VARYING	String
NATIONAL CHARACTER LARGE OBJECT	String
NATIONAL CHARACTER VARYING	String
NATIONAL CHAR VARYING	String
NATIONAL CHARACTER	String
NATIONAL CHAR	String
BINARY LARGE OBJECT	String
BINARY VARYING	String

## Geo Data Types

ClickHouse supports data types for representing geographical objects — locations, lands, etc.

## Warning

Currently geo data types are an experimental feature. To work with them you must set `allow_experimental_geo_types = 1`.

### See Also

- [Representing simple geographical features](#).
- [allow\\_experimental\\_geo\\_types setting](#).

## Point

Point is represented by its X and Y coordinates, stored as a [Tuple\(Float64, Float64\)](#).

### Example

Query:

```
SET allow_experimental_geo_types = 1;
CREATE TABLE geo_point (p Point) ENGINE = Memory();
INSERT INTO geo_point VALUES((10, 10));
SELECT p, toTypeName(p) FROM geo_point;
```

Result:

p	toTypeName(p)
(10,10)	Point

## Ring

Ring is a simple polygon without holes stored as an array of points: [Array\(Point\)](#).

### Example

Query:

```
SET allow_experimental_geo_types = 1;
CREATE TABLE geo_ring (r Ring) ENGINE = Memory();
INSERT INTO geo_ring VALUES([(0, 0), (10, 0), (10, 10), (0, 10)]);
SELECT r, toTypeName(r) FROM geo_ring;
```

Result:

r	toTypeName(r)
[(0,0),(10,0),(10,10),(0,10)]	Ring

## Polygon

Polygon is a polygon with holes stored as an array of rings: [Array\(Ring\)](#). First element of outer array is the outer shape of polygon and all the following elements are holes.

### Example

This is a polygon with one hole:

```
SET allow_experimental_geo_types = 1;
CREATE TABLE geo_polygon (pg Polygon) ENGINE = Memory();
INSERT INTO geo_polygon VALUES([(20, 20), (50, 20), (50, 50), (20, 50)], [(30, 30), (50, 50), (50, 30)]);
SELECT pg, toTypeName(pg) FROM geo_polygon;
```

Result:

pg	[(20,20),(50,20),(50,50),(20,50)],[(30,30),(50,50),(50,30)]	Polygon	toTypeName(pg)
----	-------------------------------------------------------------	---------	----------------

## MultiPolygon

`MultiPolygon` consists of multiple polygons and is stored as an array of polygons: [Array\(Polygon\)](#).

### Example

This multipolygon consists of two separate polygons — the first one without holes, and the second with one hole:

```
SET allow_experimental_geo_types = 1;
CREATE TABLE geo_multipolygon (mpg MultiPolygon) ENGINE = Memory();
INSERT INTO geo_multipolygon VALUES([[(0, 0), (10, 0), (10, 10), (0, 10)], [(20, 20), (50, 20), (50, 50), (20, 50)], [(30, 30), (50, 50), (50, 30)]]);
SELECT mpg, toTypeName(mpg) FROM geo_multipolygon;
```

Result:

mpg	TypeName(mpg)	[[[(0,0),(10,0),(10,10),(0,10)],[(20,20),(50,20),(50,50),(20,50)],[(30,30),(50,50),(50,30)]]]	MultiPolygon
-----	---------------	-----------------------------------------------------------------------------------------------	--------------

## Map(key, value)

`Map(key, value)` data type stores key:value pairs.

### Parameters

- `key` — The key part of the pair. [String](#), [Integer](#), [LowCardinality](#), or [FixedString](#).
- `value` — The value part of the pair. [String](#), [Integer](#), [Array](#), [LowCardinality](#), or [FixedString](#).

To get the value from an `a Map('key', 'value')` column, use `a['key']` syntax. This lookup works now with a linear complexity.

### Examples

Consider the table:

```
CREATE TABLE table_map (a Map(String, UInt64)) ENGINE=Memory;
INSERT INTO table_map VALUES ('key1':1, 'key2':10), ('key1':2,'key2':20}, ('key1':3,'key2':30});
```

Select all `key2` values:

```
SELECT a['key2'] FROM table_map;
```

Result:

```
arrayElement(a, 'key2')—  
 10 |  
 20 |  
 30 |
```

If there's no such key in the Map() column, the query returns zeros for numerical values, empty strings or empty arrays.

```
INSERT INTO table_map VALUES ( {'key3':100}), ({});  
SELECT a['key3'] FROM table_map;
```

Result:

```
arrayElement(a, 'key3')—  
 100 |  
 0 |  
  
arrayElement(a, 'key3')—  
 0 |  
 0 |  
 0 |
```

## Convert Tuple to Map Type

You can cast Tuple() as Map() using **CAST** function:

```
SELECT CAST(([1, 2, 3], ['Ready', 'Steady', 'Go']), 'Map(UInt8, String)') AS map;
```

```
map—  
 {1:'Ready',2:'Steady',3:'Go'} |
```

## Map.keys and Map.values Subcolumns

To optimize Map column processing, in some cases you can use the keys and values subcolumns instead of reading the whole column.

### Example

Query:

```
CREATE TABLE t_map (`a` Map(String, UInt64)) ENGINE = Memory;  
INSERT INTO t_map VALUES (map('key1', 1, 'key2', 2, 'key3', 3));  
SELECT a.keys FROM t_map;  
SELECT a.values FROM t_map;
```

Result:

```
a.keys  
['key1','key2','key3'] |
```

```
a.values  
[1,2,3] |
```

## See Also

- [map\(\)](#) function
- [CAST\(\)](#) function

## SimpleAggregateFunction

`SimpleAggregateFunction(name, types_of_arguments...)` データ型は、集計関数の現在の値を格納し、その完全な状態を次のように格納しません `AggregateFunction` そうだ この最適化は、次のプロパティが保持される関数に適用できます。`f` 行セットに `S1 UNION ALL S2` 取得できるよ `f` 行の一部に別々に設定し、再び適用します `f` 結果に: `f(S1 UNION ALL S2) = f(f(S1) UNION ALL f(S2))`. このプロパティは、部分集計の結果が結合された結果を計算するのに十分であることを保証するため、余分なデータを格納して処理する必要はありません

次の集計関数がサポートされます:

- `any`
- `anyLast`
- `min`
- `max`
- `sum`
- `groupBitAnd`
- `groupBitOr`
- `groupBitXor`
- `groupArrayArray`
- `groupUniqArrayArray`

の値 `SimpleAggregateFunction(func, Type)` 見て、同じように格納 `Type` したがって、次の関数を適用する必要はありません `-Merge/-State` 接尾辞。 `SimpleAggregateFunction` は以上のパフォーマンス `AggregateFunction` 同じ集計機能を使って。

### パラメータ

- 集計関数の名前。
- 集計関数の引数の型。

### 例

```
CREATE TABLE t
(
    column1 SimpleAggregateFunction(sum, UInt64),
    column2 SimpleAggregateFunction(any, String)
) ENGINE = ...
```

## 演算子

ClickHouseは、優先順位、優先順位、および連想に従って、クエリ解析段階で演算子を対応する関数に変換します。

### アクセス演算子

`a[N]` – Access to an element of an array. The `arrayElement(a, N)` 機能。

`a.N` – Access to a tuple element. The `tupleElement(a, N)` 機能。

### 数値否定演算子

`-a` – The `negate(a)` 機能。

### 乗算演算子と除算演算子

`a * b` – The `multiply(a, b)` 機能。

`a / b` – The `divide(a, b)` 機能。

`a % b` – The `modulo(a, b)` 機能。

### 加算演算子と減算演算子

`a + b` – The `plus(a, b)` 機能。

`a - b` – The `minus(a, b)` 機能。

### 比較演算子

`a = b` – The `equals(a, b)` 機能。

`a == b` – The `equals(a, b)` 機能。

`a != b` – The `notEquals(a, b)` 機能。

`a <> b` – The `notEquals(a, b)` 機能。

`a <= b` – The `lessOrEquals(a, b)` 機能。

`a >= b` – The `greaterOrEquals(a, b)` 機能。

`a < b` – The `less(a, b)` 機能。

`a > b` – The `greater(a, b)` 機能。

`a LIKE s` – The `like(a, b)` 機能。

`a NOT LIKE s` – The `notLike(a, b)` 機能。

`a BETWEEN b AND c` – The same as `a >= b AND a <= c`.

`a NOT BETWEEN b AND c` – The same as `a < b OR a > c`.

### データセットを操作する演算子

見る 演算子で。

a IN ... – The `in(a, b)` 機能。

a NOT IN ... – The `notIn(a, b)` 機能。

a GLOBAL IN ... – The `globalIn(a, b)` 機能。

a GLOBAL NOT IN ... – The `globalNotIn(a, b)` 機能。

## 日付と時刻を操作する演算子

### EXTRACT

```
EXTRACT(part FROM date);
```

特定の日付から部品を抽出します。たとえば、特定の日付から月、または時間から秒を取得できます。

その `part` パラメータ取得する日付のどの部分を指定します。使用可能な値は次のとおりです:

- `DAY` — The day of the month. Possible values: 1-31.
- `MONTH` — The number of a month. Possible values: 1-12.
- `YEAR` — The year.
- `SECOND` — The second. Possible values: 0-59.
- `MINUTE` — The minute. Possible values: 0-59.
- `HOUR` — The hour. Possible values: 0-23.

その `part` パラメータは大文字と小文字を区別する。

その `date` パラメータ処理する日付または時刻を指定します。どちらか **日付** または **DateTime** タイプに対応しています。

例:

```
SELECT EXTRACT(DAY FROM toDate('2017-06-15'));
SELECT EXTRACT(MONTH FROM toDate('2017-06-15'));
SELECT EXTRACT(YEAR FROM toDate('2017-06-15'));
```

次の例では、テーブルを作成し、そこに値を挿入します `DateTime` タイプ。

```
CREATE TABLE test.Orders
(
    OrderId UInt64,
    OrderName String,
    OrderDate DateTime
)
ENGINE = Log;
```

```
INSERT INTO test.Orders VALUES (1, 'Jarlsberg Cheese', toDateTime('2008-10-11 13:23:44'));
```

```

SELECT
    toYear(OrderDate) AS OrderYear,
    toMonth(OrderDate) AS OrderMonth,
    toDayOfMonth(OrderDate) AS OrderDay,
    toHour(OrderDate) AS OrderHour,
    toMinute(OrderDate) AS OrderMinute,
    toSecond(OrderDate) AS OrderSecond
FROM test.Orders;

```

OrderYear	OrderMonth	OrderDay	OrderHour	OrderMinute	OrderSecond
2008	10	11	13	23	44

以下の例を見ることができます [テスト](#)。

## INTERVAL

を作成します。間隔-算術演算で使用されるべき型の値 [日付](#) と [DateTime](#)-値を入力します。

間隔のタイプ:

- SECOND
- MINUTE
- HOUR
- DAY
- WEEK
- MONTH
- QUARTER
- YEAR

### 警告

異なるタイプの間隔は結合できません。次のような式は使用できません `INTERVAL 4 DAY 1 HOUR`。間隔は、間隔の最小単位より小さい単位または等しい単位で指定します。, `INTERVAL 25 HOUR`。次の例のように、連続した操作を使用できます。

例:

```
SELECT now() AS current_date_time, current_date_time + INTERVAL 4 DAY + INTERVAL 3 HOUR
```

current_date_time	plus(plus(now(), toIntervalDay(4)), toIntervalHour(3))
2019-10-23 11:16:28	2019-10-27 14:16:28

も参照。

- [間隔](#) データ型
- [toInterval](#) 型変換関数

## 論理否定演算子

`NOT a` – The `not(a)` 機能。

## 論理And演算子

`a AND b` – The `and(a, b)` function.

## 論理OR演算子

`a OR b` – The `or(a, b)` function.

## 条件演算子

`a ? b : c` – The `if(a, b, c)` function.

注:

条件演算子は、`b`と`c`の値を計算し、条件`a`が満たされているかどうかをチェックし、対応する値を返します。もし `b` または `C` は [アレイジョイン\(\)](#) 関数は、各行は関係なく、レプリケートされます “`a`” 条件だ

## 条件式

```
CASE [x]
  WHEN a THEN b
  [WHEN ... THEN ...]
  [ELSE c]
END
```

もし `x` が指定される。 `transform(x, [a, ...], [b, ...], c)` function is used. Otherwise – `multilf(a, b, ..., c)`.

がない場合 `ELSE c` 句式のデフォルト値は次のとおりです `NULL`.

その `transform` 関数は動作しません `NULL`.

## 連結演算子

`s1 || s2` – The `concat(s1, s2)` function.

## ラムダ作成演算子

`x -> expr` – The `lambda(x, expr)` function.

次の演算子は角かっこであるため、優先順位はありません:

## 配列作成演算子

`[x1, ...]` – The `array(x1, ...)` function.

## タプル作成演算子

`(x1, x2, ...)` – The `tuple(x2, x2, ...)` function.

## 連想性

すべての二項演算子は連想性を残しています。例えば, `1 + 2 + 3` に変換されます `plus(plus(1, 2), 3)`.

時にはこれはあなたが期待するように動作しません。例えば, `SELECT 4 > 2 > 3` 結果は0になります。

効率のために、`and` と `or` 関数は任意の数の引数を受け入れます。の対応する鎖 `AND` と `OR` 演算子は、これらの関数の单一の呼び出しに変換されます。

## チェック `NULL`

クリックハウスは `IS NULL` と `IS NOT NULL` 演算子。

## `IS NULL`

- のために Null 可能 型の値は、IS NULL 演算子の戻り値:
  - 1 値が NULL.
  - 0 そうでなければ
- その他の値については、IS NULL 演算子は常に返します 0.

```
SELECT x+100 FROM t_null WHERE y IS NULL
```

plus(x, 100)	101
--------------	-----

## IS NOT NULL

- のために Null 可能 型の値は、IS NOT NULL 演算子の戻り値:
  - 0 値が NULL.
  - 1 そうでなければ
- その他の値については、IS NOT NULL 演算子は常に返します 1.

```
SELECT * FROM t_null WHERE y IS NOT NULL
```

x	y
2	3

## IN Operators

The IN, NOT IN, GLOBAL IN, and GLOBAL NOT IN operators are covered separately, since their functionality is quite rich.

The left side of the operator is either a single column or a tuple.

Examples:

```
SELECT UserID IN (123, 456) FROM ...
SELECT (CounterID, UserID) IN ((34, 123), (101500, 456)) FROM ...
```

If the left side is a single column that is in the index, and the right side is a set of constants, the system uses the index for processing the query.

Don't list too many values explicitly (i.e. millions). If a data set is large, put it in a temporary table (for example, see the section [External data for query processing](#)), then use a subquery.

The right side of the operator can be a set of constant expressions, a set of tuples with constant expressions (shown in the examples above), or the name of a database table or SELECT subquery in brackets.

ClickHouse allows types to differ in the left and the right parts of IN subquery. In this case it converts the left side value to the type of the right side, as if the `accurateCastOrNull` function is applied. That means, that the data type becomes `Nullable`, and if the conversion cannot be performed, it returns `NULL`.

### Example

Query:

```
SELECT '1' IN (SELECT 1);
```

Result:

in('1', _subquery49)
1

If the right side of the operator is the name of a table (for example, `UserID IN users`), this is equivalent to the subquery `UserID IN (SELECT * FROM users)`. Use this when working with external data that is sent along with the query. For example, the query can be sent together with a set of user IDs loaded to the ‘users’ temporary table, which should be filtered.

If the right side of the operator is a table name that has the Set engine (a prepared data set that is always in RAM), the data set will not be created over again for each query.

The subquery may specify more than one column for filtering tuples.

Example:

```
SELECT (CounterID, UserID) IN (SELECT CounterID, UserID FROM ...)
```

The columns to the left and right of the IN operator should have the same type.

The IN operator and subquery may occur in any part of the query, including in aggregate functions and lambda functions.

Example:

```
SELECT
    EventDate,
    avg(UserID IN
    (
        SELECT UserID
        FROM test.hits
        WHERE EventDate = toDate('2014-03-17')
    )) AS ratio
FROM test.hits
GROUP BY EventDate
ORDER BY EventDate ASC
```

EventDate	ratio
2014-03-17	1
2014-03-18	0.807696
2014-03-19	0.755406
2014-03-20	0.723218
2014-03-21	0.697021
2014-03-22	0.647851
2014-03-23	0.648416

For each day after March 17th, count the percentage of pageviews made by users who visited the site on March 17th.

A subquery in the IN clause is always run just one time on a single server. There are no dependent subqueries.

## NULL Processing

During request processing, the `IN` operator assumes that the result of an operation with `NULL` always equals `0`, regardless of whether `NULL` is on the right or left side of the operator. `NULL` values are not included in any dataset, do not correspond to each other and cannot be compared if `transform_null_in = 0`.

Here is an example with the `t_null` table:

x	y
1	NULL
2	3

Running the query `SELECT x FROM t_null WHERE y IN (NULL,3)` gives you the following result:

x
2

You can see that the row in which `y = NULL` is thrown out of the query results. This is because ClickHouse can't decide whether `NULL` is included in the `(NULL,3)` set, returns `0` as the result of the operation, and `SELECT` excludes this row from the final output.

```
SELECT y IN (NULL, 3)
FROM t_null
```

```
in(y, tuple(NULL, 3))—
  0 |
  1
```

## Distributed Subqueries

There are two options for `IN`-s with subqueries (similar to `JOINS`): normal `IN` / `JOIN` and `GLOBAL IN` / `GLOBAL JOIN`. They differ in how they are run for distributed query processing.

### Attention

Remember that the algorithms described below may work differently depending on the `settings distributed_product_mode` setting.

When using the regular `IN`, the query is sent to remote servers, and each of them runs the subqueries in the `IN` or `JOIN` clause.

When using `GLOBAL IN` / `GLOBAL JOINS`, first all the subqueries are run for `GLOBAL IN` / `GLOBAL JOINS`, and the results are collected in temporary tables. Then the temporary tables are sent to each remote server, where the queries are run using this temporary data.

For a non-distributed query, use the regular `IN` / `JOIN`.

Be careful when using subqueries in the `IN` / `JOIN` clauses for distributed query processing.

Let's look at some examples. Assume that each server in the cluster has a normal `local_table`. Each server also has a `distributed_table` table with the `Distributed` type, which looks at all the servers in the cluster.

For a query to the **distributed\_table**, the query will be sent to all the remote servers and run on them using the **local\_table**.

For example, the query

```
SELECT uniq(UserID) FROM distributed_table
```

will be sent to all remote servers as

```
SELECT uniq(UserID) FROM local_table
```

and run on each of them in parallel, until it reaches the stage where intermediate results can be combined. Then the intermediate results will be returned to the requestor server and merged on it, and the final result will be sent to the client.

Now let's examine a query with IN:

```
SELECT uniq(UserID) FROM distributed_table WHERE CounterID = 101500 AND UserID IN (SELECT UserID FROM local_table WHERE CounterID = 34)
```

- Calculation of the intersection of audiences of two sites.

This query will be sent to all remote servers as

```
SELECT uniq(UserID) FROM local_table WHERE CounterID = 101500 AND UserID IN (SELECT UserID FROM local_table WHERE CounterID = 34)
```

In other words, the data set in the IN clause will be collected on each server independently, only across the data that is stored locally on each of the servers.

This will work correctly and optimally if you are prepared for this case and have spread data across the cluster servers such that the data for a single UserID resides entirely on a single server. In this case, all the necessary data will be available locally on each server. Otherwise, the result will be inaccurate. We refer to this variation of the query as “local IN”.

To correct how the query works when data is spread randomly across the cluster servers, you could specify **distributed\_table** inside a subquery. The query would look like this:

```
SELECT uniq(UserID) FROM distributed_table WHERE CounterID = 101500 AND UserID IN (SELECT UserID FROM distributed_table WHERE CounterID = 34)
```

This query will be sent to all remote servers as

```
SELECT uniq(UserID) FROM local_table WHERE CounterID = 101500 AND UserID IN (SELECT UserID FROM distributed_table WHERE CounterID = 34)
```

The subquery will begin running on each remote server. Since the subquery uses a distributed table, the subquery that is on each remote server will be resent to every remote server as

```
SELECT UserID FROM local_table WHERE CounterID = 34
```

For example, if you have a cluster of 100 servers, executing the entire query will require 10,000 elementary requests, which is generally considered unacceptable.

In such cases, you should always use GLOBAL IN instead of IN. Let's look at how it works for the query

```
SELECT uniq(UserID) FROM distributed_table WHERE CounterID = 101500 AND UserID GLOBAL IN (SELECT UserID  
FROM distributed_table WHERE CounterID = 34)
```

The requestor server will run the subquery

```
SELECT UserID FROM distributed_table WHERE CounterID = 34
```

and the result will be put in a temporary table in RAM. Then the request will be sent to each remote server as

```
SELECT uniq(UserID) FROM local_table WHERE CounterID = 101500 AND UserID GLOBAL IN _data1
```

and the temporary table `_data1` will be sent to every remote server with the query (the name of the temporary table is implementation-defined).

This is more optimal than using the normal IN. However, keep the following points in mind:

1. When creating a temporary table, data is not made unique. To reduce the volume of data transmitted over the network, specify DISTINCT in the subquery. (You do not need to do this for a normal IN.)
2. The temporary table will be sent to all the remote servers. Transmission does not account for network topology. For example, if 10 remote servers reside in a datacenter that is very remote in relation to the requestor server, the data will be sent 10 times over the channel to the remote datacenter. Try to avoid large data sets when using GLOBAL IN.
3. When transmitting data to remote servers, restrictions on network bandwidth are not configurable. You might overload the network.
4. Try to distribute data across servers so that you do not need to use GLOBAL IN on a regular basis.
5. If you need to use GLOBAL IN often, plan the location of the ClickHouse cluster so that a single group of replicas resides in no more than one data center with a fast network between them, so that a query can be processed entirely within a single data center.

It also makes sense to specify a local table in the GLOBAL IN clause, in case this local table is only available on the requestor server and you want to use data from it on remote servers.

## Distributed Subqueries and max\_parallel\_replicas

When `max_parallel_replicas` is greater than 1, distributed queries are further transformed. For example, the following:

```
SELECT CounterID, count() FROM distributed_table_1 WHERE UserID IN (SELECT UserID FROM local_table_2 WHERE  
CounterID < 100)  
SETTINGS max_parallel_replicas=3
```

is transformed on each server into

```
SELECT CounterID, count() FROM local_table_1 WHERE UserID IN (SELECT UserID FROM local_table_2 WHERE  
CounterID < 100)  
SETTINGS parallel_replicas_count=3, parallel_replicas_offset=M
```

where M is between 1 and 3 depending on which replica the local query is executing on. These settings affect every MergeTree-family table in the query and have the same effect as applying `SAMPLE 1/3 OFFSET (M-1)/3` on each table.

Therefore adding the `max_parallel_replicas` setting will only produce correct results if both tables have the same replication scheme and are sampled by UserID or a subkey of it. In particular, if `local_table_2` does not have a sampling key, incorrect results will be produced. The same rule applies to JOIN.

One workaround if `local_table_2` does not meet the requirements, is to use `GLOBAL IN` or `GLOBAL JOIN`.

## ClickHouse SQLの方言のANSI SQLの互換性

### 注

この記事では、表38に依存しています，“Feature taxonomy and definition for mandatory features”，Annex F of ISO/IEC CD 9075-2:2013.

### 行動の違い

次の表は、ClickHouseでクエリ機能が機能しますが、ANSI SQLで指定されているように動作しないケースを示しています。

Feature ID	機能名	違い
E011	数値データ型	数値リテラルと期間として解釈される近似 ( <code>Float64</code> ) の代わりに正確な ( <code>Decimal</code> )
E051-05	選択した項目の名前を変更できます	アイテムの名前変更は、選択結果だけよりも可視性の範囲が広くなります
E141-01	NULLでない制約	<code>NOT NULL</code> は默示のためのテーブル列によるデフォルト
E011-04	算術演算子	ClickHouse溢れの代わりに次の算術演算の結果のデータ型に基づくカスタムルール

### 機能の状態

Feature ID	機能名	状態	コメント
E011	数値データ型	部分的	
E011-01	整数およびSMALLINTデータ型	はい。	
E011-02	実数、倍精度および浮動小数点データ型データ型	部分的	<code>FLOAT(&lt;binary_precision&gt;)</code> , <code>REAL</code> と <code>DOUBLE PRECISION</code> 対応していません
E011-03	DECIMALおよびNUMERICデータ型	部分的	のみ <code>DECIMAL(p,s)</code> サポートされています。 <code>NUMERIC</code>
E011-04	算術演算子	はい。	
E011-05	数値比較	はい。	

Feature ID	機能名	状態	コメント
E011-06	数値データ型間の暗黙的なキャスト	いいえ。	ANSI SQLできる任意の暗黙的な数値型の間のキャストがClickHouseに依存しての機能を有する複数の過負荷の代わりに暗黙的なキャスト
<b>E021</b>	文字列タイプ	部分的	
E021-01	文字データ型	いいえ。 .text-danger}	
E021-02	文字変化型データ型	いいえ。	<code>String</code> 動作同様に、長さの制限内
E021-03	文字リテラル	部分的	連続したリテラルと文字セットの自動連結はサポートされません
E021-04	CHARACTER_LENGTH関数	部分的	いいえ。 <code>USING</code> 句
E021-05	OCTET_LENGTH関数	いいえ。	<code>LENGTH</code> 同様に動作します
E021-06	SUBSTRING	部分的	サポートなし <code>SIMILAR</code> と <code>ESCAPE</code> 句、ない <code>SUBSTRING_REGEX</code> パリアント
E021-07	文字の連結	部分的	いいえ。 <code>COLLATE</code> 句
E021-08	上部および下の機能	はい。	
E021-09	トリム機能	はい。	
E021-10	固定長および可変長文字ストリング型間の暗黙的なキャスト	いいえ。	ANSI SQLできる任意の暗黙の間のキャスト文字列の種類がClickHouseに依存しての機能を有する複数の過負荷の代わりに暗黙的なキャスト
E021-11	位置関数	部分的	サポートなし <code>IN</code> と <code>USING</code> 句、ない <code>POSITION_REGEX</code> パリアント
E021-12	文字の比較	はい。	
<b>E031</b>	識別子	部分的	
E031-01	区切り識別子	部分的	Unicodeリテラルの支援は限られ
E031-02	小文字の識別子	はい。	
E031-03	末尾のアンダースコア	はい。	
<b>E051</b>	基本的なクエリ仕様	部分的	
E051-01	SELECT DISTINCT	はい。	
E051-02	GROUP BY句	はい。	

Feature ID	機能名	状態	コメント
E051-04	グループによる列を含むことが できない <select list>	はい。	
E051-05	選択した項目の名前を変更でき ます	はい。	
E051-06	句を持つ	はい。	
E051-07	選択リストの修飾*	はい。	
E051-08	FROM句の相関名	はい。	
E051-09	FROM句の列の名前を変更しま す	いいえ。	
<b>E061</b>	基本的な述語と検索条件	部分的	
E061-01	比較述語	はい。	
E061-02	述語の間	部分的	いいえ。 SYMMETRIC と ASYMMETRIC 句
E061-03	値のリストを持つ述語で	はい。	
E061-04	述語のように	はい。	
E061-05	LIKE述語:エスケープ句	いいえ。	
E061-06	Null述語	はい。	
E061-07	定量化された比較述語	いいえ。	
E061-08	存在する述語	いいえ。	
E061-09	比較述語のサブクエリ	はい。	
E061-11	In述語のサブクエリ	はい。	
E061-12	定量化された比較述語のサブク エリ	いいえ。	
E061-13	相関サブクエリ	いいえ。	
E061-14	検索条件	はい。	
<b>E071</b>	基本的なクエリ式	部分的	
E071-01	UNION DISTINCTテーブル演算 子	いいえ。	
E071-02	UNION ALLテーブル演算子	はい。	

Feature ID	機能名	状態	コメント
E071-03	DISTINCTテーブル演算子を除く	いいえ。	
E071-05	列の結合経由でテーブル事業者の必要のない全く同じデータ型	はい。	
E071-06	サブクエリ内のテーブル演算子	はい。	
<b>E081</b>	基本権限	部分的	進行中の作業
<b>E091</b>	関数の設定	はい。	
E091-01	AVG	はい。	
E091-02	COUNT	はい。	
E091-03	MAX	はい。	
E091-04	MIN	はい。	
E091-05	SUM	はい。	
E091-06	すべての量指定子	いいえ。	
E091-07	異なる量指定子	部分的	な集計機能に対応
<b>E101</b>	基本的なデータ操作	部分的	
E101-01	INSERT文	はい。	注：ClickHouseの主キーは、 <b>UNIQUE</b> 制約
E101-03	検索されたUPDATE文	いいえ。	そこには <b>ALTER UPDATE</b> パッチデータ変更のための命令
E101-04	検索されたDELETE文	いいえ。	そこには <b>ALTER DELETE</b> パッチデータ削除のための命令
<b>E111</b>	単一行 <b>SELECT</b> ステートメント	いいえ。	
<b>E121</b>	基本的にカーソルを支援	いいえ。	
E121-01	DECLARE CURSOR	いいえ。	
E121-02	ORDER BY列を選択リストに含める必要はありません	いいえ。	
E121-03	ORDER BY句の値式	いいえ。	
E121-04	開いた声明	いいえ。	
E121-06	位置付きUPDATE文	いいえ。	

Feature ID	機能名	状態	コメント
E121-07	位置づけDELETEステートメント	いいえ。	
E121-08	閉じる文	いいえ。	
E121-10	FETCHステートメント:暗黙的なNEXT	いいえ。	
E121-17	ホールドカーソル付き	いいえ。	
<b>E131</b>	<b>Null値のサポート(値の代わりにnull)</b>	部分的	一部の制限が適用されます
<b>E141</b>	基本的な整合性制約	部分的	
E141-01	NULLでない制約	はい。	注: NOT NULL は默示のためのテーブル列によるデフォルト
E141-02	NULLでない列の一意制約	いいえ。	
E141-03	主キー制約	いいえ。	
E141-04	参照削除アクションと参照updateアクションの両方に対するNO ACTIONのデフォルトを持つ基本外部キー制約	いいえ。	
E141-06	制約のチェック	はい。	
E141-07	列の既定値	はい。	
E141-08	主キーで推論されるNULLではありません	はい。	
E141-10	外部キーの名前は任意の順序で指定できます	いいえ。	
<b>E151</b>	取引サポート	いいえ。	
E151-01	COMMIT文	いいえ。	
E151-02	ROLLBACKステートメント	いいえ。	
<b>E152</b>	基本セット取引明細書	いいえ。	
E152-01	SET TRANSACTION文:分離レベルSERIALIZABLE句	いいえ。	
E152-02	SET TRANSACTION文:READ ONLY句とREAD WRITE句	いいえ。	

Feature ID	機能名	状態	コメント
E153	サブクエリを使用した更新可能なクエリ	いいえ。	
E161	先頭の <b>double minus</b> を使用したSQLコメント	はい。	
E171	<b>SQLSTATE</b> サポート	いいえ。	
E182	ホスト言語バインド	いいえ。	
F031	基本的なスキーマ操作	部分的	
F031-01	永続ベーステーブルを作成するCREATE TABLE文	部分的	いいえ。 SYSTEM VERSIONING, ON COMMIT, GLOBAL, LOCAL, PRESERVE, DELETE, REF IS, WITH OPTIONS, UNDER, LIKE, PERIOD FOR 句およびユーザー解決データ型のサポートなし
F031-02	CREATE VIEW文	部分的	いいえ。 RECURSIVE, CHECK, UNDER, WITH OPTIONS 句およびユーザー解決データ型のサポートなし
F031-03	グラント声明	はい。	
F031-04	ALTER TABLE文:ADD COLUMN句	部分的	サポートなし GENERATED 節およびシステム期間
F031-13	DROP TABLE文:RESTRICT句	いいえ。	
F031-16	DROP VIEW文:RESTRICT句	いいえ。	
F031-19	REVOKEステートメント:RESTRICT句	いいえ。	
F041	基本的な結合テーブル	部分的	
F041-01	Inner join必ずというわけではないが、内側のキーワード)	はい。	
F041-02	内部キーワード	はい。	
F041-03	LEFT OUTER JOIN	はい。	
F041-04	RIGHT OUTER JOIN	はい。	
F041-05	外部結合は入れ子にできます	はい。	
F041-07	左外部結合または右外部結合の内部テーブルは、内部結合でも使用できます	はい。	

Feature ID	機能名	状態	コメント
F041-08	すべての比較演算子がサポート	いいえ。	
<b>F051</b>	基本日時	部分的	
F051-01	日付データ型(日付リテラルのサポートを含む)	部分的	リテラルなし
F051-02	秒の小数部の精度が0以上の時刻データ型(時刻リテラルのサポートを含む)	いいえ。	
F051-03	タイムスタンプのデータ型を含む支援のタイムスタンプ文字と小数点以下の秒の精度で少なくとも0-6	いいえ。	DateTime64 timeは同様の機能を提供します
F051-04	日付、時刻、およびタイムスタンプのデータ型の比較述語	部分的	使用可能なデータ型は一つだけです
F051-05	Datetime型と文字列型の間の明示的なキャスト	はい。	
F051-06	CURRENT_DATE	いいえ。	today() 似ています
F051-07	LOCALTIME	いいえ。	now() 似ています
F051-08	LOCALTIMESTAMP	いいえ。	
<b>F081</b>	ビュー内の組合および除外	部分的	
<b>F131</b>	グループ化操作	部分的	
F131-01	ここで、グループにより、条項対応してクエリを処理するクラウドの場合グ眺望	はい。	
F131-02	グループ化されたビュー	はい。	
F131-03	セット機能に対応してクエリを処理するクラウドの場合グ眺望	はい。	
F131-04	GROUP BY句とHAVING句およびグループ化ビューを持つサブクエリ	はい。	
F131-05	GROUP BY句およびHAVING句およびグループ化ビューを使用した單一行選択	いいえ。	
<b>F181</b>	複数モジュール対応	いいえ。	

Feature ID	機能名	状態	コメント
F201	キャスト機能	はい。	
F221	明示的な既定値	いいえ。	
F261	大文字と小文字の式	はい。	
F261-01	簡単な場合	はい。	
F261-02	検索ケース	はい。	
F261-03	NULLIF	はい。	
F261-04	COALESCE	はい。	
F311	スキーマ定義文	部分的	
F311-01	CREATE SCHEMA	いいえ。	
F311-02	永続ベーステーブルのテーブルの作成	はい。	
F311-03	CREATE VIEW	はい。	
F311-04	CREATE VIEW: WITH CHECK OPTION	いいえ。	
F311-05	グラント声明	はい。	
F471	スカラーサブクエリ値	はい。	
F481	展開されたNULL述語	はい。	
F812	基本的なフラグ設定	いいえ。	
T321	基本的なSQL呼び出しルーチン	いいえ。	
T321-01	オーバーロードのないユーザー定義関数	いいえ。	
T321-02	過負荷のないユーザー定義ストアドプロシージャ	いいえ。	
T321-03	関数呼び出し	いいえ。	
T321-04	CALL文	いいえ。	
T321-05	RETURN文	いいえ。	
T631	一つのリスト要素を持つ述語で	はい。	

# [experimental] Window Functions

ClickHouse supports the standard grammar for defining windows and window functions. The following features are currently supported:

Feature	Support or workaround
ad hoc window specification (count(*) over (partition by id order by time desc))	supported
expressions involving window functions, e.g. (count(*) over () / 2)	not supported, wrap in a subquery ( <a href="#">feature request</a> )
WINDOW clause (select ... from table window w as (partiton by id))	supported
ROWS frame	supported
RANGE frame	supported, the default
INTERVAL syntax for DateTime RANGE OFFSET frame	not supported, specify the number of seconds instead
GROUPS frame	not supported
Calculating aggregate functions over a frame (sum(value) over (order by time))	all aggregate functions are supported
rank(), dense_rank(), row_number()	supported
lag/lead(value, offset)	Not supported. Workarounds:  1) replace with any(value) over (.... rows between <offset> preceding and <offset> preceding), or following for lead  2) use lagInFrame/leadInFrame, which are analogous, but respect the window frame. To get behavior identical to lag/lead, use rows between unbounded preceding and unbounded following

## References

### GitHub Issues

The roadmap for the initial support of window functions is [in this issue](#).

All GitHub issues related to window funtions have the [comp-window-functions](#) tag.

### Tests

These tests contain the examples of the currently supported grammar:

[https://github.com/ClickHouse/ClickHouse/blob/master/tests/performance/window\\_functions.xml](https://github.com/ClickHouse/ClickHouse/blob/master/tests/performance/window_functions.xml)

[https://github.com/ClickHouse/ClickHouse/blob/master/tests/queries/0\\_stateless/01591\\_window\\_functions.sql](https://github.com/ClickHouse/ClickHouse/blob/master/tests/queries/0_stateless/01591_window_functions.sql)

## Postgres Docs

<https://www.postgresql.org/docs/current/sql-select.html#SQL-WINDOW>

<https://www.postgresql.org/docs/devel/sql-expressions.html#SYNTAX-WINDOW-FUNCTIONS>

<https://www.postgresql.org/docs/devel/functions-window.html>

<https://www.postgresql.org/docs/devel/tutorial-window.html>

## MySQL Docs

<https://dev.mysql.com/doc/refman/8.0/en/window-function-descriptions.html>

<https://dev.mysql.com/doc/refman/8.0/en/window-functions-usage.html>

<https://dev.mysql.com/doc/refman/8.0/en/window-functions-frames.html>

# ClickHouse ガイド

ClickHouseを使用してさまざまなタスクを解決するのに役立つ詳細なステップバイステップの手順の一覧:

- 簡単なクラスター設定のチュートリアル
- ClickHouseでのCatBoostモデルの適用

## ClickHouseでのCatboostモデルの適用

CatBoostで開発された無料でオープンソースの勾配昇圧ライブラリです Yandex 機械学習のために。

この手順では、SQLからモデル推論を実行して、ClickHouseで事前に訓練されたモデルを適用する方法を学習します。

ClickHouseでCatBoostモデルを適用するには:

1. テーブルの作成.
2. テーブルにデータを挿入します.
3. ClickHouseにCatBoostを統合する (任意ステップ)。
4. SQLからモデル推論を実行する.

CatBoostモデルのトレーニングの詳細については、[モデルの学習と適用](#).

## 前提条件

あなたが持っていない場合 [ドッカー](#) まだ、それを取付けなさい。

### 注

ドッカー CatBoostとClickHouseのインストールをシステムの残りの部分から分離するコンテナを作成できるソフトウェアプラットフォームです。

CatBoostモデルを適用する前に:

## 1. プル Dockerイメージ レジストリから:

```
$ docker pull yandex/tutorial-catboost-clickhouse
```

このDockerイメージには、CatBoostとClickHouseを実行するために必要なコード、ランタイム、ライブラリ、環境変数、設定ファイルがすべて含まれています。

## 2. Dockerイメージが正常にプルされたことを確認します:

```
$ docker image ls
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
yandex/tutorial-catboost-clickhouse  latest   622e4d17945b  22 hours ago  1.37GB
```

## 3. 起Dockerコンテナに基づくこのイメージ:

```
$ docker run -it -p 8888:8888 yandex/tutorial-catboost-clickhouse
```

## 1. テーブルの作成

トレーニングサンプルのClickHouseテーブルを作成するには:

### 1. 対話モードでClickHouse consoleクライアントを起動する:

```
$ clickhouse client
```

#### 注

ClickHouseサーバーはDockerコンテナ内で既に実行されています。

### 2. コマンドを使用して表を作成します:

```
:) CREATE TABLE amazon_train
(
    date Date MATERIALIZED today(),
    ACTION UInt8,
    RESOURCE UInt32,
    MGR_ID UInt32,
    ROLE_ROLLUP_1 UInt32,
    ROLE_ROLLUP_2 UInt32,
    ROLE_DEPTNAME UInt32,
    ROLE_TITLE UInt32,
    ROLE_FAMILY_DESC UInt32,
    ROLE_FAMILY UInt32,
    ROLE_CODE UInt32
)
ENGINE = MergeTree ORDER BY date
```

### 3. ClickHouse consoleクライアントからの終了:

```
:) exit
```

## 2. テーブルにデータを挿入します

データを挿入するには:

## 1. 次のコマンドを実行します:

```
$ clickhouse client --host 127.0.0.1 --query 'INSERT INTO amazon_train FORMAT CSVWithNames' < ~/amazon/train.csv
```

## 2. 対話モードでClickHouse console クライアントを起動する:

```
$ clickhouse client
```

## 3. データがアップロードされたことを確認:

```
:) SELECT count() FROM amazon_train  
  
SELECT count()  
FROM amazon_train  
  
+-count()-+  
| 65538 |  
+-----+
```

## 3. ClickHouseにCatBoostを統合する

### 注

任意ステップ。 Dockerイメージには、CatBoostとClickHouseを実行するために必要なすべてが含まれています。

ClickhouseにCatBoostを統合するには:

### 1. 評価ライブラリを構築します。

CatBoostモデルを評価する最速の方法はコンパイルです `libcatboostmodel.<so|dll|dylib>` 図書館に関する詳細については、図書館を参照 [CatBoostドキュメント](#)。

### 2. 新しいディレクトリを任意の場所に作成し、任意の名前で作成します。, `data` 作成したライブラリをその中に入れます。のDocker画像がすでに含まれている図書館 `data/libcatboostmodel.so`.

### 3. Config modelの新しいディレクトリを任意の場所に、任意の名前で作成します。, `models`.

### 4. 任意の名前のモデル構成ファイルを作成します。, `models/amazon_model.xml`.

### 5. モデル構成の説明:

```
<models>  
  <model>  
    <!-- Model type. Now catboost only. -->  
    <type>catboost</type>  
    <!-- Model name. -->  
    <name>amazon</name>  
    <!-- Path to trained model. -->  
    <path>/home/catboost/tutorial/catboost_model.bin</path>  
    <!-- Update interval. -->  
    <lifetime>0</lifetime>  
  </model>  
</models>
```

## 6. CatBoostへのパスとモデル設定をClickHouse設定に追加します:

```
<!-- File etc/clickhouse-server/config.d/models_config.xml. -->
<catboost_dynamic_library_path>/home/catboost/data/libcatboostmodel.so</catboost_dynamic_library_path>
<models_config>/home/catboost/models/*_model.xml</models_config>
```

## 4. SQLからモデル推論を実行する

試験モデルのClickHouseへ \$ clickhouse client

モデルが動作していることを確認しましょう:

```
:) SELECT
    modelEvaluate('amazon',
        RESOURCE,
        MGR_ID,
        ROLE_ROLLUP_1,
        ROLE_ROLLUP_2,
        ROLE_DEPTNAME,
        ROLE_TITLE,
        ROLE_FAMILY_DESC,
        ROLE_FAMILY,
        ROLE_CODE) > 0 AS prediction,
    ACTION AS target
FROM amazon_train
LIMIT 10
```

### 注

関数 **モデル評価** マルチクラスモデルのクラスごとの生の予測を持つタプルを返します。

のは、確率を予測してみましょう:

```
:) SELECT
    modelEvaluate('amazon',
        RESOURCE,
        MGR_ID,
        ROLE_ROLLUP_1,
        ROLE_ROLLUP_2,
        ROLE_DEPTNAME,
        ROLE_TITLE,
        ROLE_FAMILY_DESC,
        ROLE_FAMILY,
        ROLE_CODE) AS prediction,
    1. / (1 + exp(-prediction)) AS probability,
    ACTION AS target
FROM amazon_train
LIMIT 10
```

### 注

詳細について **exp()** 機能。

サンプルのLogLossを計算しましょう:

```
:) SELECT -avg(tg * log(prob) + (1 - tg) * log(1 - prob)) AS logloss
FROM
(
  SELECT
    modelEvaluate('amazon',
      RESOURCE,
      MGR_ID,
      ROLE_ROLLUP_1,
      ROLE_ROLLUP_2,
      ROLE_DEPTNAME,
      ROLE_TITLE,
      ROLE_FAMILY_DESC,
      ROLE_FAMILY,
      ROLE_CODE) AS prediction,
      1. / (1. + exp(-prediction)) AS prob,
      ACTION AS tg
    FROM amazon_train
)
```

## 注

詳細について **avg()** と **ログ()** 機能。

## 操作

ClickHouse操作マニュアルの以下の主要部:

- 要件
- 監視
- トラブル
- 使用法の推奨事項
- 更新手順
- アクセス権
- データバックア
- 設定ファイル
- クオータ
- システム表
- サーバ構成パラメータ
- ClickHouseでハードウェアをテストする方法
- 設定
- ユーティリ

## 要件

### CPU

ビルド済みのdebパッケージからインストールするには、X86\_64アーキテクチャのCPUを使用し、SSE4.2命令をサポートします。走ClickHouseプロセッサをサポートしていないSSE4.2でAArch64はPowerPC64LE建築、協力して進めることが必要でありClickHouseから。

ClickHouseを実装した並列データの処理-利用のすべてのハードウェア資料を備えています。プロセッサを選択するときは、ClickHouseが多数のコアを持つ構成でより効率的に動作するが、コアが少なくクロックレートが高い構成ではatよりも低い。たとえば、16 2600MHzのコアは、8 3600MHzのコアよりも好みです。

使用することをお勧めします ターボブースト と ハイパースレッド テクノロジー 大幅なパフォーマンス向上を図る、典型的な負荷も大きくなっています。

## RAM

些細でないクエリを実行するには、最小4GBのRAMを使用することをお勧めします。ClickHouseサーバーは、はるかに少ない量のRAMで実行できますが、クエリを処理するためにメモリが必要です。

必要なRAM容量は次のとおりです:

- クエリの複雑さ。
- クエリで処理されるデータの量。

RAMの必要量を計算するには、次のような一時データのサイズを推定する必要があります **GROUP BY, DISTINCT, JOIN** そしてあなたが使用する他の操作。

ClickHouseは一時データに外部メモリを使用できます。見る [外部メモリのGROUP BY 詳細](#)については。

## Swapファイル

運用環境のスワップファイルを無効にします。

## Storageサブシステム

ClickHouseをインストールするには、2GBの空きディスク領域が必要です。

データに必要なストレージ容量は、別々に計算する必要があります。評価には:

- データ量の推定。

データのサンプルを取得し、そこから行の平均サイズを取得できます。次に、値に格納する予定の行数を掛けます。

- データ圧縮係数。

データ圧縮係数を推定するには、データのサンプルをClickHouseに読み込み、データの実際のサイズと格納されているテーブルのサイズを比較します。たとえば、clickstreamデータは通常6-10回圧縮されます。

格納するデータの最終ボリュームを計算するには、推定データボリュームに圧縮係数を適用します。複数のレプリカにデータを格納する場合は、推定ボリュームにレプリカの数を掛けます。

## ネット

可能であれば、10G以上のネットワークを使用してください。

大量の中間データを含む分散クエリを処理するには、ネットワーク帯域幅が重要です。また、ネットワーク速度に影響する複製プロセス。

## ソフト

ClickHouseの開発を中心に、Linuxの家族システムです。推奨されるLinuxディストリビューションのtzdataパッケ

ClickHouse働きかけることができ、その他業務システム。の詳細を参照してください [はじめに](#) ドキュメントのセクション。

---

## 監視

監視できます:

- ハードウェアリソースの利用。
- ClickHouseサーバー指標。

## リソース使用率

ClickHouseは、ハードウェアリソースの状態を単独で監視しません。

監視をセットアップすることを強く推奨します:

- プロセッサの負荷と温度。

以下を使用できます [dmesg](#), [ターボスタッフ](#) または他の楽器。

- ストレージシステム、RAM、ネットワークの利用。

## ClickHouseサーバー指標

ClickHouse serverには、自己状態の監視のための計測器が組み込まれています。

追跡サーバのイベントサーバーを利用します。を参照。[ロガー](#) 設定ファイルのセクション。

クリックハウス収集:

- 異なるメトリクスのサーバがどのように利用計算資源です。
- クエリ処理に関する一般的な統計。

メトリックは、次のとおりです。 [システムメトリック](#), [システムイベント](#), and [システムasynchronous\\_metrics](#) テーブル

を設定することができClickHouse輸出の指標に [黒鉛](#)。を参照。[グラフライ特部](#) ClickHouseサーバー設定ファイル内。指標のエクスポートを設定する前に、公式に従ってGraphiteを設定する必要があります [ガイド](#)。

を設定することができClickHouse輸出の指標に [プロメテウス](#)。を参照。[プロメテウス節](#) ClickHouseサーバー設定ファイル内。メトリックのエクスポートを設定する前に、公式に従ってPrometheusを設定してください [ガイド](#)。

さらに、HTTP APIを使用してサーバーの可用性を監視できます。送信 [HTTP GET](#) リクエスト先 [/ping](#)。サーバーが利用可能な場合は、次のように応答します [200 OK](#).

監視サーバーにクラスター構成設定してください [max\\_replica\\_delay\\_for\\_distributed\\_queries](#) パラメータとHTTPリソースの使用 [/replicas\\_status](#)。への要求 [/replicas\\_status](#) ツヅツ。 [200 OK](#) レプリカが使用可能で、他のレプリカより遅れていない場合。レプリカが遅延すると、次のようになります [503 HTTP\\_SERVICE\\_UNAVAILABLE](#) ギャップについての情報と。

---

## トラブル

- [設置](#)
- [サーバーへの接続](#)
- [クエリ処理](#)

- クエリ処理の効率

## 設置

### Apt-getではClickHouseリポジトリからDebパッケージを取得できません

- ファイアウォールの設定
- できない場合はアクセスリポジトリのために、何らかの理由でダウンロードパッケージに記載のとおり **はじめに** 記事とを使用して手動でインストール `sudo dpkg -i <packages>` コマンド また、必要になります `tzdata` パッケージ。

## サーバーへの接続

考えられる問題:

- サーバーが実行されていません。
- 想定外または誤った設定パラメータ。

## サーバの実行中に

サーバーが**running**かどうかを確認する

コマンド:

```
$ sudo service clickhouse-server status
```

サーバーが実行されていない場合は、次のコマンドで起動します:

```
$ sudo service clickhouse-server start
```

## ログの確認

のメインログ `clickhouse-server` である `/var/log/clickhouse-server/clickhouse-server.log` デフォルトでは。

サーバーが正常に起動した場合は、文字列が表示されます:

- `<Information> Application: starting up.` — Server started.
- `<Information> Application: Ready for connections.` — Server is running and ready for connections.

もし `clickhouse-server` 設定エラーで起動に失敗しました。 `<Error>` エラーの説明を含む文字列。 例えば:

```
2019.01.11 15:23:25.549505 [ 45 ] {} <Error> ExternalDictionaries: Failed reloading 'event2id' external dictionary: Poco::Exception. Code: 1000, e.code() = 111, e.displayText() = Connection refused, e.what() = Connection refused
```

ファイルの最後にエラーが表示されない場合は、文字列から始まるファイル全体を調べます:

```
<Information> Application: starting up.
```

あなたが第二のインスタンスを起動しようとすると `clickhouse-server` サーバーには、次のログが表示されます:

```
2019.01.11 15:25:11.151730 [ 1 ] {} <Information> : Starting ClickHouse 19.1.0 with revision 54413
2019.01.11 15:25:11.154578 [ 1 ] {} <Information> Application: starting up
2019.01.11 15:25:11.156361 [ 1 ] {} <Information> StatusFile: Status file ./status already exists - unclean restart.
Contents:
PID: 8510
Started at: 2019-01-11 15:24:23
Revision: 54413

2019.01.11 15:25:11.156673 [ 1 ] {} <Error> Application: DB::Exception: Cannot lock file ./status. Another server
instance in same directory is already running.
2019.01.11 15:25:11.156682 [ 1 ] {} <Information> Application: shutting down
2019.01.11 15:25:11.156686 [ 1 ] {} <Debug> Application: Uninitializing subsystem: Logging Subsystem
2019.01.11 15:25:11.156716 [ 2 ] {} <Information> BaseDaemon: Stop SignalListener thread
```

## システムを参照。dログ

で有用な情報が見つからない場合 `clickhouse-server` ログまたはログがない場合は、表示できます `system.d` コマンドを使用したログ:

```
$ sudo journalctl -u clickhouse-server
```

## Clickhouse-serverを対話モードで起動する

```
$ sudo -u clickhouse /usr/bin/clickhouse-server --config-file /etc/clickhouse-server/config.xml
```

このコマ このモードでは `clickhouse-server` 版画のすべてのイベントメッセージです。

## 構成パラメータ

チェック:

- ドッカーの設定。

DockerでClickhouseをIPv6ネットワークで実行する場合は、次のことを確認してください `network=host` 設定されています。

- エンドポイント設定。

チェック `listen_host` と `tcp_port` 設定。

ClickHouse serverはデフォルトでのみlocalhost接続を受け入れます。

- HTTPプロトコル設定。

HTTP APIのプロトコル設定を確認します。

- 安全な接続設定。

チェック:

- その `tcp_port_secure` 設定。

- の設定 `SSLセリティクス`.

適切なパラメータを接続たとえば、`port_secure` 変数との `clickhouse_client`.

- ユーザー設定。

きの違うユーザーネームやパスワードになります。

## クエリ処理

ClickHouseは、クエリを処理できない場合は、クライアントにエラーの説明を送信します。で `clickhouse-client` コンソールにエラーの説明が表示されます。HTTPインターフェイスを使用している場合、ClickHouseは応答本文にエラーの説明を送信します。例えば：

```
$ curl 'http://localhost:8123/' --data-binary "SELECT a"
Code: 47, e.displayText() = DB::Exception: Unknown identifier: a. Note that there are no tables (FROM clause) in your query, context: required_names: 'a' source_tables: table_aliases: private_aliases: column_aliases: public_columns: 'a' masked_columns: array_join_columns: source_columns: , e.what() = DB::Exception
```

あなたが開始した場合 `clickhouse-client` と `stack-trace` パラメータ ClickHouseは、エラーの説明を含むサーバースタックトレースを返します。

あるいは、メッセージが壊れて接続します。この場合、クエリを繰り返すことができます。クエリを実行するたびに接続が切断された場合は、サーバーログでエラーを確認します。

## クエリ処理の効率

ClickHouseの動作が遅すぎる場合は、クエリのサーバーリソースとネットワークの負荷をプロファイルする必要があります。

`Clickhouse-benchmark`ユーティリティ 毎秒処理されたクエリの数、毎秒処理された行数、およびクエリ処理時間の百分位数が表示されます。

## クリックハウスの更新

ClickHouseがdebパッケージからインストールさ：

```
$ sudo apt-get update
$ sudo apt-get install clickhouse-client clickhouse-server
$ sudo service clickhouse-server restart
```

推奨されるdebパッケージ以外のものを使用してClickHouseをインストールした場合は、適切な更新方法を使用します。

ClickHouseは分散updateをサポートしていません。操作は、個別のサーバーごとに連続して実行する必要があります。クラスター上のすべてのサーバーを同時に更新しないでください。

## External User Authenticators and Directories

ClickHouse supports authenticating and managing users using external services.

The following external authenticators and directories are supported:

- [LDAP Authenticator and Directory](#)
- [Kerberos Authenticator](#)

## Kerberos

Existing and properly configured ClickHouse users can be authenticated via Kerberos authentication protocol.

Currently, Kerberos can only be used as an external authenticator for existing users, which are defined in `users.xml` or in local access control paths. Those users may only use HTTP requests and must be able to authenticate using GSS-SPNEGO mechanism.

For this approach, Kerberos must be configured in the system and must be enabled in ClickHouse config.

## Enabling Kerberos in ClickHouse

To enable Kerberos, one should include `kerberos` section in `config.xml`. This section may contain additional parameters.

Parameters:

- `principal` - canonical service principal name that will be acquired and used when accepting security contexts.
- This parameter is optional, if omitted, the default principal will be used.
- `realm` - a realm, that will be used to restrict authentication to only those requests whose initiator's realm matches it.
- This parameter is optional, if omitted, no additional filtering by realm will be applied.

Example (goes into `config.xml`):

```
<clickhouse>
  <!-- ... -->
  <kerberos />
</clickhouse>
```

With principal specification:

```
<clickhouse>
  <!-- ... -->
  <kerberos>
    <principal>HTTP/clickhouse.example.com@EXAMPLE.COM</principal>
  </kerberos>
</clickhouse>
```

With filtering by realm:

```
<clickhouse>
  <!-- ... -->
  <kerberos>
    <realm>EXAMPLE.COM</realm>
  </kerberos>
</clickhouse>
```

### Note

You can define only one `kerberos` section. The presence of multiple `kerberos` sections will force ClickHouse to disable Kerberos authentication.

### Note

`principal` and `realm` sections cannot be specified at the same time. The presence of both `principal` and `realm` sections will force ClickHouse to disable Kerberos authentication.

## Kerberos as an external authenticator for existing users

Kerberos can be used as a method for verifying the identity of locally defined users (users defined in `users.xml` or in local access control paths). Currently, **only** requests over the HTTP interface can be *kerberized* (via GSS-SPNEGO mechanism).

Kerberos principal name format usually follows this pattern:

- `primary/instance@REALM`

The `/instance` part may occur zero or more times. **The primary part of the canonical principal name of the initiator is expected to match the kerberized user name for authentication to succeed.**

## Enabling Kerberos in `users.xml`

In order to enable Kerberos authentication for the user, specify `kerberos` section instead of `password` or similar sections in the user definition.

Parameters:

- `realm` - a realm that will be used to restrict authentication to only those requests whose initiator's realm matches it.
- This parameter is optional, if omitted, no additional filtering by realm will be applied.

Example (goes into `users.xml`):

```
<clickhouse>
  <!-- ... -->
  <users>
    <!-- ... -->
    <my_user>
      <!-- ... -->
      <kerberos>
        <realm>EXAMPLE.COM</realm>
      </kerberos>
    </my_user>
  </users>
</clickhouse>
```

## Warning

Note that Kerberos authentication cannot be used alongside with any other authentication mechanism. The presence of any other sections like `password` alongside `kerberos` will force ClickHouse to shutdown.

## Reminder

Note, that now, once user `my_user` uses `kerberos`, Kerberos must be enabled in the main `config.xml` file as described previously.

## Enabling Kerberos using SQL

When [SQL-driven Access Control and Account Management](#) is enabled in ClickHouse, users identified by Kerberos can also be created using SQL statements.

```
CREATE USER my_user IDENTIFIED WITH kerberos REALM 'EXAMPLE.COM'
```

...or, without filtering by realm:

```
CREATE USER my_user IDENTIFIED WITH kerberos
```

## LDAP

LDAP server can be used to authenticate ClickHouse users. There are two different approaches for doing this:

- Use LDAP as an external authenticator for existing users, which are defined in `users.xml` or in local access control paths.
- Use LDAP as an external user directory and allow locally undefined users to be authenticated if they exist on the LDAP server.

For both of these approaches, an internally named LDAP server must be defined in the ClickHouse config so that other parts of the config can refer to it.

## LDAP Server Definition

To define LDAP server you must add `ldap_servers` section to the `config.xml`.

### Example

```
<clickhouse>
  <!-- ... -->
  <ldap_servers>
    <!-- Typical LDAP server. -->
    <my_ldap_server>
      <host>localhost</host>
      <port>636</port>
      <bind_dn>uid={user_name},ou=users,dc=example,dc=com</bind_dn>
      <verification_cooldown>300</verification_cooldown>
      <enable_tls>yes</enable_tls>
      <tls_minimum_protocol_version>tls1.2</tls_minimum_protocol_version>
      <tls_require_cert>demand</tls_require_cert>
      <tls_cert_file>/path/to/tls_cert_file</tls_cert_file>
      <tls_key_file>/path/to/tls_key_file</tls_key_file>
      <tls_ca_cert_file>/path/to/tls_ca_cert_file</tls_ca_cert_file>
      <tls_ca_cert_dir>/path/to/tls_ca_cert_dir</tls_ca_cert_dir>
      <tls_cipher_suite>ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:AES256-GCM-
SHA384</tls_cipher_suite>
    </my_ldap_server>

    <!-- Typical Active Directory with configured user DN detection for further role mapping. -->
    <my_ad_server>
      <host>localhost</host>
      <port>389</port>
      <bind_dn>EXAMPLE\{user_name}</bind_dn>
      <user_dn_detection>
        <base_dn>CN=Users,DC=example,DC=com</base_dn>
        <search_filter>(&(objectClass=user)(sAMAccountName={user_name}))</search_filter>
      </user_dn_detection>
      <enable_tls>no</enable_tls>
    </my_ad_server>
  </ldap_servers>
</clickhouse>
```

Note, that you can define multiple LDAP servers inside the `ldap_servers` section using distinct names.

### Parameters

- `host` — LDAP server hostname or IP, this parameter is mandatory and cannot be empty.

- `port` — LDAP server port, default is `636` if `enable_tls` is set to `true`, `389` otherwise.
- `bind_dn` — Template used to construct the DN to bind to.
  - The resulting DN will be constructed by replacing all `{user_name}` substrings of the template with the actual user name during each authentication attempt.
- `user_dn_detection` — Section with LDAP search parameters for detecting the actual user DN of the bound user.
  - This is mainly used in search filters for further role mapping when the server is Active Directory. The resulting user DN will be used when replacing `{user_dn}` substrings wherever they are allowed. By default, user DN is set equal to bind DN, but once search is performed, it will be updated with to the actual detected user DN value.
  - `base_dn` — Template used to construct the base DN for the LDAP search.
    - The resulting DN will be constructed by replacing all `{user_name}` and `{bind_dn}` substrings of the template with the actual user name and bind DN during the LDAP search.
  - `scope` — Scope of the LDAP search.
    - Accepted values are: `base`, `one_level`, `children`, `subtree` (the default).
  - `search_filter` — Template used to construct the search filter for the LDAP search.
    - The resulting filter will be constructed by replacing all `{user_name}`, `{bind_dn}`, and `{base_dn}` substrings of the template with the actual user name, bind DN, and base DN during the LDAP search.
    - Note, that the special characters must be escaped properly in XML.
- `verification_cooldown` — A period of time, in seconds, after a successful bind attempt, during which the user will be assumed to be successfully authenticated for all consecutive requests without contacting the LDAP server.
  - Specify `0` (the default) to disable caching and force contacting the LDAP server for each authentication request.
- `enable_tls` — A flag to trigger the use of the secure connection to the LDAP server.
  - Specify `no` for plain text `ldap://` protocol (not recommended).
  - Specify `yes` for LDAP over SSL/TLS `ldaps://` protocol (recommended, the default).
  - Specify `starttls` for legacy StartTLS protocol (plain text `ldap://` protocol, upgraded to TLS).
- `tls_minimum_protocol_version` — The minimum protocol version of SSL/TLS.
  - Accepted values are: `ssl2`, `ssl3`, `tls1.0`, `tls1.1`, `tls1.2` (the default).
- `tls_require_cert` — SSL/TLS peer certificate verification behavior.
  - Accepted values are: `never`, `allow`, `try`, `demand` (the default).
- `tls_cert_file` — Path to certificate file.
- `tls_key_file` — Path to certificate key file.
- `tls_ca_cert_file` — Path to CA certificate file.
- `tls_ca_cert_dir` — Path to the directory containing CA certificates.
- `tls_cipher_suite` — Allowed cipher suite (in OpenSSL notation).

## LDAP External Authenticator

A remote LDAP server can be used as a method for verifying passwords for locally defined users (users defined in `users.xml` or in local access control paths). To achieve this, specify previously defined LDAP server name instead of `password` or similar sections in the user definition.

At each login attempt, ClickHouse tries to "bind" to the specified DN defined by the `bind_dn` parameter in the [LDAP server definition](#) using the provided credentials, and if successful, the user is considered authenticated. This is often called a "simple bind" method.

## Example

```
<clickhouse>
  <!- ... -->
  <users>
    <!- ... -->
    <my_user>
      <!- ... -->
      <ldap>
        <server>my_ldap_server</server>
      </ldap>
    </my_user>
  </users>
</clickhouse>
```

Note, that user `my_user` refers to `my_ldap_server`. This LDAP server must be configured in the main `config.xml` file as described previously.

When SQL-driven [Access Control and Account Management](#) is enabled, users that are authenticated by LDAP servers can also be created using the [CREATE USER](#) statement.

Query:

```
CREATE USER my_user IDENTIFIED WITH ldap SERVER 'my_ldap_server';
```

## LDAP External User Directory

In addition to the locally defined users, a remote LDAP server can be used as a source of user definitions. To achieve this, specify previously defined LDAP server name (see [LDAP Server Definition](#)) in the `ldap` section inside the `users_directories` section of the `config.xml` file.

At each login attempt, ClickHouse tries to find the user definition locally and authenticate it as usual. If the user is not defined, ClickHouse will assume the definition exists in the external LDAP directory and will try to "bind" to the specified DN at the LDAP server using the provided credentials. If successful, the user will be considered existing and authenticated. The user will be assigned roles from the list specified in the `roles` section. Additionally, LDAP "search" can be performed and results can be transformed and treated as role names and then be assigned to the user if the `role_mapping` section is also configured. All this implies that the SQL-driven [Access Control and Account Management](#) is enabled and roles are created using the [CREATE ROLE](#) statement.

## Example

Goes into `config.xml`.

```

<clickhouse>
  <!-- ... -->
  <user_directories>
    <!-- Typical LDAP server. -->
    <ldap>
      <server>my_ldap_server</server>
      <roles>
        <my_local_role1 />
        <my_local_role2 />
      </roles>
      <role_mapping>
        <base_dn>ou=groups,dc=example,dc=com</base_dn>
        <scope>subtree</scope>
        <search_filter>(&(objectClass=groupOfNames)(member={bind_dn}))</search_filter>
        <attribute>cn</attribute>
        <prefix>clickhouse_</prefix>
      </role_mapping>
    </ldap>

    <!-- Typical Active Directory with role mapping that relies on the detected user DN. -->
    <ldap>
      <server>my_ad_server</server>
      <role_mapping>
        <base_dn>CN=Users,DC=example,DC=com</base_dn>
        <attribute>CN</attribute>
        <scope>subtree</scope>
        <search_filter>(&(objectClass=group)(member={user_dn}))</search_filter>
        <prefix>clickhouse_</prefix>
      </role_mapping>
    </ldap>
  </user_directories>
</clickhouse>

```

Note that `my_ldap_server` referred in the `ldap` section inside the `user_directories` section must be a previously defined LDAP server that is configured in the [config.xml](#) (see [LDAP Server Definition](#)).

## Parameters

- `server` — One of LDAP server names defined in the `ldap_servers` config section above. This parameter is mandatory and cannot be empty.
- `roles` — Section with a list of locally defined roles that will be assigned to each user retrieved from the LDAP server.
  - If no roles are specified here or assigned during role mapping (below), user will not be able to perform any actions after authentication.

- `role_mapping` — Section with LDAP search parameters and mapping rules.
  - When a user authenticates, while still bound to LDAP, an LDAP search is performed using `search_filter` and the name of the logged-in user. For each entry found during that search, the value of the specified attribute is extracted. For each attribute value that has the specified prefix, the prefix is removed, and the rest of the value becomes the name of a local role defined in ClickHouse, which is expected to be created beforehand by the `CREATE ROLE` statement.
- There can be multiple `role_mapping` sections defined inside the same `ldap` section. All of them will be applied.
  - `base_dn` — Template used to construct the base DN for the LDAP search.
    - The resulting DN will be constructed by replacing all `{user_name}`, `{bind_dn}`, and `{user_dn}` substrings of the template with the actual user name, bind DN, and user DN during each LDAP search.
  - `scope` — Scope of the LDAP search.
    - Accepted values are: `base`, `one_level`, `children`, `subtree` (the default).
  - `search_filter` — Template used to construct the search filter for the LDAP search.
    - The resulting filter will be constructed by replacing all `{user_name}`, `{bind_dn}`, `{user_dn}`, and `{base_dn}` substrings of the template with the actual user name, bind DN, user DN, and base DN during each LDAP search.
  - Note, that the special characters must be escaped properly in XML.
  - `attribute` — Attribute name whose values will be returned by the LDAP search. `cn`, by default.
  - `prefix` — Prefix, that will be expected to be in front of each string in the original list of strings returned by the LDAP search. The prefix will be removed from the original strings and the resulting strings will be treated as local role names. Empty by default.

## アクセス制御とアカウント管理

ClickHouseはに基づくアクセス制御管理を支えます `RBAC` アプローチ

ClickHouseアクセス事業体:

- ユーザー
- 役割
- 行ポリシー
- 設定プロファイル
- クオータ

を設定することができアクセスを用いた:

- SQL駆動型ワークフロー。  
する必要があります **有効にする** この機能。
- サーバ `設定ファイル` `users.xml` と `config.xml`.

SQL駆動型ワークフローの使用をお勧めします。両方の構成方法が同時に機能するため、アカウントとアクセス権を管理するためにサーバー構成ファイルを使用する場合は、sql駆動型ワークフローに簡単

### 警告

両方の構成方法で同じ `access` エンティティを同時に管理することはできません。

# 使用法

既定では、ClickHouseサーバーはユーザー アカウントを提供します `default` SQL 駆動型のアクセス制御とアカウント管理を使用することはできませんが、すべての権限と権限を持っていています。その `default` ユーザー アカウントは、クライアントからのログイン時や分散クエリなど、ユーザー名が定義されていない場合に使用されます。分散クエリ処理のデフォルトのユーザー アカウントをお使いの場合、設定のサーバまたはクラスターを指定していないの **ユーザとパスワード** プロパティ。

ClickHouseの使用を開始したばかりの場合は、次のシナリオを使用できます：

1. **有効にする** のためのSQL駆動型アクセス制御およびアカウント管理 `default` ユーザー。
2. の下でログイン `default` ユーザー アカウントを作成し、必要なすべてのユーザー 管理者アカウントの作成を忘れないでください (`GRANT ALL ON *.* WITH GRANT OPTION TO admin_user_account`)。
3. **権限の制限** のために `default` SQL 駆動型のアクセス制御とアカウント管理をユーザーと無効にします。

## 現在のソリューションの特性

- データベースとテーブルが存在しない場合でも、権限を付与できます。
- テーブルが削除された場合、このテーブルに対応するすべての権限は取り消されません。したがって、後で同じ名前で新しいテーブルが作成されると、すべての特権が再び実際になります。削除されたテーブルに対応する権限を取り消すには、次のように実行する必要があります。 `REVOKE ALL PRIVILEGES ON db.table FROM ALL クエリ`。
- 特権の有効期間の設定はありません。

## ユーザー

ユーザー アカウントは、ClickHouseで誰かを承認できるアクセスエンティティです。ユーザー アカウント:

- 識別情報。
- **特権** これは、ユーザーが実行できるクエリの範囲を定義します。
- ClickHouseサーバーへの接続が許可されているホスト。
- 付与された役割と既定の役割。
- ユーザーのログイン時にデフォルトで適用される制約を含む設定。
- 割り当ての設定を行います。

ユーザー アカウントに対する権限は、**GRANT** クエリまたは割り当て **役割**。ユーザーから特権を取り消すために、ClickHouseは **REVOKE** クエリ。ユーザーの権限を一覧表示するには、- **SHOW GRANTS** 声明。

管理 クエリ:

- **CREATE USER**
- **ALTER USER**
- **DROP USER**
- **SHOW CREATE USER**

## 設定の適用

設定は、さまざまな方法で設定できます。ユーザー ログイン時に、異なるアクセスエンティティで設定が設定されている場合、この設定の値および制約は、以下の優先順位によって適用されます():

1. ユーザー アカウント 設定。

2. ユーザー アカウントの既定のロールの設定。一部のロールで設定が設定されている場合、設定の適用順序は未定義です。
3. ユーザーまたはその既定のロールに割り当てられた設定プロファイルの設定。いくつかのプロファイルで設定が設定されている場合、設定適用の順序は未定義です。
4. すべてのサーバーにデフォルトまたは [標準プロファイル](#)。

## 役割

Roleは、ユーザー アカウントに付与できるaccessエンティティのコンテナです。

ロール:

- [特権](#)
- 設定と制約
- 付与されたロールのリスト

管理クエリ:

- [CREATE ROLE](#)
- [ALTER ROLE](#)
- [DROP ROLE](#)
- [SET ROLE](#)
- [SET DEFAULT ROLE](#)
- [SHOW CREATE ROLE](#)

ロールに対する権限は、[GRANT](#) クエリ。ロールClickHouseから特権を取り消すには[REVOKE](#) クエリ。

## 行ポリシー

行ポリシーは、ユーザーまたはロールで使用できる行または行を定義するフィルターです。行政政策を含むイのための特定のテーブルリストの役割および/またはユーザーはこの行政政策です。

管理クエリ:

- [CREATE ROW POLICY](#)
- [ALTER ROW POLICY](#)
- [DROP ROW POLICY](#)
- [SHOW CREATE ROW POLICY](#)

## 設定プロファイル

設定プロファイルは[設定](#)。設定プロファイルには、設定と制約、およびこのクオータが適用されるロールやユーザーのリストが含まれます。

管理クエリ:

- [CREATE SETTINGS PROFILE](#)
- [ALTER SETTINGS PROFILE](#)
- [DROP SETTINGS PROFILE](#)

## ■ SHOW CREATE SETTINGS PROFILE

### クオータ

クオータ制限資源利用に見る クオータ。

定員の制限のために一部の時間、リストの役割および/またはユーザーはこの数量に達した場合。

管理クエリ:

- CREATE QUOTA
- ALTER QUOTA
- DROP QUOTA
- SHOW CREATE QUOTA

## SQL駆動型アクセス制御とアカウント管理の有効化

- 設定ディレクトリ構成を保管します。

ClickHouseは、アクセスエンティティ設定を `access_control_path` サーバー構成パラメータ。

- SQL駆動型のアクセス制御とアカウント管理を有効にします。

デフォルトのSQL型のアクセス制御及び特別口座の口座管理オのすべてのユーザー ユーザーを設定する必要があります。`users.xml` に1を割り当てます。 `access_management` 設定。

## データバックアップ

ながら [複製](#) provides protection from hardware failures, it does not protect against human errors: accidental deletion of data, deletion of the wrong table or a table on the wrong cluster, and software bugs that result in incorrect data processing or data corruption. In many cases mistakes like these will affect all replicas. ClickHouse has built-in safeguards to prevent some types of mistakes — for example, by default [50Gbを超えるデータを含むMergeTreeのようなエンジンでは、テーブルを削除することはできません](#). しかし、これらの保障措置がカバーしないすべてのケースで回避。

ヒューマンエラーを効果的に軽減するには、データのバックアップと復元のための戦略を慎重に準備する必要があります事前に。

それぞれの会社には、利用可能なリソースとビジネス要件が異なるため、あらゆる状況に合ったClickHouseバックアップと復元のための普遍的なソリューション 何がデータのギガバイトのために働く可能性が高い数十ペタバイトのために動作しません。自分の長所と短所を持つ様々な可能なアプローチがありますが、これは以下で説明します。で使うようにするといいでしょくつかの方法でそれを補うためにさまの様々な問題があった。

### 注

あなたが何かをバックアップし、それを復元しようとしたことがない場合、チャンスは、あなたが実際にそれを必要とするときに復元が正常に動作したがって、どのようなバックアップアプローチを選択しても、復元プロセスも自動化し、予備のClickHouseクラスターで定期的に練習してください。

## ソースデータを他の場所に複製する

多くの場合、ClickHouseに取り込まれたデータは、次のような何らかの永続キューを介して配信されます **アパッチ-カフカ**。この場合、ClickHouseに書き込まれている間に同じデータストリームを読み取り、どこかのコールドストレージに保存する追加のサブスクライバーセットを構成することほとんど企業はすでにいくつかのデフォルト推奨コールドストレージを持っています。 **HDFS**。

## ファイルシステム

一部地域のファイルシステムをスナップショット機能(例えば、**ZFS**しかし、ライブクエリを提供するための最良の選択ではないかもしれません。可能な解決策は、この種のファイルシステムで追加のレプリカを作成し、それらを **分散** 以下の目的で使用されるテーブル **SELECT** クエリ。スナップショットなどを複製するものでなければならないのクエリがデータを変更する。ボーナスパートとして、これらのレプリカが特別なハードウェア構成によりディスクに付属のサーバ、コスト効果的です。

## クリックハウス-複写機

**クリックハウス-複写機** ペタバイトサイズのテーブルを再シャードするために最初に作成された多目的な用具はある。ClickHouseテーブルとクラスター間でデータを確実にコピーするため、バックアップと復元の目的でも使用できます。

データの小さなボリュームのために、単純な **INSERT INTO ... SELECT ...** リモートテーブルが作業しています。

## 部品による操作

ClickHouseは、**ALTER TABLE ... FREEZE PARTITION ...** クエリをコピーのテーブル割。これはハードリンクを使って実装される。**/var/lib/clickhouse/shadow/** それは通常、古いデータのための余分なディスク領域を消費しません。作成されたファイルのコピーはClickHouse serverによって処理されないので、そこに残すことができます：追加の外部システムを必要としない簡単なバックアップ このため、リモートで別の場所にコピーしてからローカルコピーを削除する方が良いでしょう。分散ファイルシステムとオブジェクトストアはまだこのための良いオプションですが、十分な容量を持つ通常の添付ファイルサーバも同様に動作 **rsync**)。

パーティション操作に関するクエリの詳細については、[文書の変更](#)。

第三者ツールを自動化するこのアプローチ: [clickhouse-バックアップ](#)。

## 設定ファイル

ClickHouseは複数のファイル構成管理をサポートします。主サーバ設定ファイルで指定することができます **/etc/clickhouse-server/config.xml**。その他のファイルは **/etc/clickhouse-server/config.d** ディレクトリ。

### 注

すべての構成ファイルはXML形式である必要があります。また、通常は同じルート要素を持つ必要があります  
**<clickhouse>**。

メイン構成ファイルで指定された一部の設定は、他の構成ファイルで上書きできます。その **replace** または **remove** これらの構成ファイルの要素に属性を指定できます。

どちらも指定されていない場合は、要素の内容を再帰的に結合し、重複する子の値を置き換えます。

もし **replace** 指定された要素全体を指定された要素に置き換えます。

もし **remove** 指定されると、要素を削除します。

この設定はまた、“substitutions”. 要素が `incl` 属性は、ファイルからの対応する置換が値として使用されます。デフォルトでは、ファイルへのパスとの置換を行う `/etc/metrika.xml`. これはで変えることができます `include_from` サーパー設定の要素。置換値は、次のように指定されます `/clickhouse/substitution_name` このファイル内の要素。で指定された

置換の場合 `incl` 存在しない場合は、ログに記録されます。ClickHouseが不足している置換をログに記録しないようにするには、`optional="true"` 属性(たとえば、マクロ).

置換はZooKeeperからも実行できます。これを行うには、属性を指定します `from_zk = "/path/to/node"`. 要素の値は、ノードの内容に置き換えられます。`/path/to/node` 飼育係で。また、ZooKeeperノードにXMLサブツリー全体を配置することもできます。

その `config.xml` ファイルを指定することで別の `config` ユーザー設定、プロファイルに割り振ります。この設定への相対パスは `users_config` 要素。既定では、次のようにになります `users.xml`. もし `users_config` ユーザー設定、プロファイル、およびクオータは、`config.xml`.

ユーザー設定は、次のような別々のファイルに分割できます `config.xml` と `config.d/`.

ディレク `users_config` 設定なし `.xml` と連結された後置 `.d`.

Directory `users.d` デフォルトでは `users_config` デフォルトは `users.xml`.

たとえば、次のようにユーザーごとに別々の設定ファイルを作成できます:

```
$ cat /etc/clickhouse-server/users.d/alice.xml
```

```
<clickhouse>
  <users>
    <alice>
      <profile>analytics</profile>
      <networks>
        <ip>::/0</ip>
      </networks>
      <password_sha256_hex>...</password_sha256_hex>
      <quota>analytics</quota>
    </alice>
  </users>
</clickhouse>
```

各設定ファイルでは、サーバともある `file-preprocessed.xml` 起動時のファイル。これらのファイルには、完了したすべての置換と上書きが含まれており、情報提供を目的としています。設定ファイルでZooKeeperの置換が使用されていても、サーバーの起動時にZooKeeperが使用できない場合、サーバーは前処理されたファイルから設定をロードします。

サーバーは、設定ファイルの変更、置換および上書きの実行時に使用されたファイルおよびZooKeeperノードを追跡し、ユーザーおよびクラスターの設定をその場で再ロードつまり、サーバーを再起動せずにクラスター、ユーザー、およびその設定を変更できます。

## クオータ

クオータを使用すると、一定期間のリソース使用量を制限したり、リソースの使用を追跡したりできます。

クオータはユーザー設定で設定されます。'users.xml'.

このシステムには、単一のクエリの複雑さを制限する機能もあります。セクションを参照 "Restrictions on query complexity").

クエリの複雑さの制限とは対照的に、クオータ:

- 単一のクエリを制限するのではなく、一定期間にわたって実行できるクエリのセットに制限を設定します。
- 口座のために費やすべてのリモートサーバーのための分散クエリ処となります。

のセクションを見てみましょう 'users.xml' クオータを定義するファイル。

```

<!-- Quotas -->
<quotas>
  <!-- Quota name. -->
  <default>
    <!-- Restrictions for a time period. You can set many intervals with different restrictions. -->
    <interval>
      <!-- Length of the interval. -->
      <duration>3600</duration>

      <!-- Unlimited. Just collect data for the specified time interval. -->
      <queries>0</queries>
      <errors>0</errors>
      <result_rows>0</result_rows>
      <read_rows>0</read_rows>
      <execution_time>0</execution_time>
    </interval>
  </default>

```

既定では、クオータは、使用量を制限することなく、各時間のリソース消費量を追跡します。  
各間隔ごとに計算されたリソース消費量は、各要求の後にサーバーログに出力されます。

```

<statbox>
  <!-- Restrictions for a time period. You can set many intervals with different restrictions. -->
  <interval>
    <!-- Length of the interval. -->
    <duration>3600</duration>

    <queries>1000</queries>
    <errors>100</errors>
    <result_rows>1000000000</result_rows>
    <read_rows>1000000000000</read_rows>
    <execution_time>900</execution_time>
  </interval>

  <interval>
    <duration>86400</duration>

    <queries>10000</queries>
    <errors>1000</errors>
    <result_rows>5000000000</result_rows>
    <read_rows>5000000000000</read_rows>
    <execution_time>7200</execution_time>
  </interval>
</statbox>

```

のために ‘statbox’ クオータ、制限は、時間ごとおよび24時間ごと(86,400秒)に設定されます。時間間隔は、実装定義の固定モーメントから開始してカウントされます。つまり、24時間の間隔は必ずしも深夜に開始されるわけではありません。

間隔が終了すると、収集された値はすべてクリアされます。次の時間は、クオータの計算がやり直されます。

制限できる金額は次のとおりです:

**queries** – The total number of requests.

**errors** – The number of queries that threw an exception.

**result\_rows** – The total number of rows given as a result.

**read\_rows** – The total number of source rows read from tables for running the query on all remote servers.

**execution\_time** – The total query execution time, in seconds (wall time).

制限を超えた場合は、どの制限を超えたか、どの間隔を超えたか、および新しい間隔が開始されたとき(クエリを再度送信できるとき)に関するテキス

クオータは、“**quota key**”複数のキーのリソースを個別に報告する機能。この例を次に示します:

```

<!-- For the global reports designer. -->
<web_global>
    <!-- keyed - The quota_key "key" is passed in the query parameter,
        and the quota is tracked separately for each key value.
        For example, you can pass a Yandex.Metrica username as the key,
        so the quota will be counted separately for each username.
        Using keys makes sense only if quota_key is transmitted by the program, not by a user.

        You can also write <keyed_by_ip />, so the IP address is used as the quota key.
        (But keep in mind that users can change the IPv6 address fairly easily.)
    -->
    <keyed />

```

クオータは ‘users’ 設定のセクション。セクションを参照 “Access rights”.

分散クエリ処理の場合、累積金額は要求元サーバーに格納されます。ここでは、ユーザーが別のサーバーの定員がありま “start over”.

サーバーを再起動すると、クオータがリセットされます。

## System Tables

### Introduction

System tables provide information about:

- Server states, processes, and environment.
- Server’s internal processes.

System tables:

- Located in the `system` database.
- Available only for reading data.
- Can’t be dropped or altered, but can be detached.

Most of system tables store their data in RAM. A ClickHouse server creates such system tables at the start.

Unlike other system tables, the system log tables `metric_log`, `query_log`, `query_thread_log`, `trace_log`, `part_log`, `crash_log` and `text_log` are served by `MergeTree` table engine and store their data in a filesystem by default. If you remove a table from a filesystem, the ClickHouse server creates the empty one again at the time of the next data writing. If system table schema changed in a new release, then ClickHouse renames the current table and creates a new one.

System log tables can be customized by creating a config file with the same name as the table under `/etc/clickhouse-server/config.d/`, or setting corresponding elements in `/etc/clickhouse-server/config.xml`. Elements can be customized are:

- `database`: database the system log table belongs to. This option is deprecated now. All system log tables are under database `system`.
- `table`: table to insert data.
- `partition_by`: specify `PARTITION BY` expression.
- `ttl`: specify table `TTL` expression.
- `flush_interval_milliseconds`: interval of flushing data to disk.

- `engine`: provide full engine expression (starting with `ENGINE =` ) with parameters. This option is contradict with `partition_by` and `ttl`. If set together, the server would raise an exception and exit.

An example:

```
<clickhouse>
  <query_log>
    <database>system</database>
    <table>query_log</table>
    <partition_by>toYYYYMM(event_date)</partition_by>
    <ttl>event_date + INTERVAL 30 DAY DELETE</ttl>
    <!--
      <engine>ENGINE = MergeTree PARTITION BY toYYYYMM(event_date) ORDER BY (event_date, event_time)
      SETTINGS index_granularity = 1024</engine>
    -->
    <flush_interval_milliseconds>7500</flush_interval_milliseconds>
  </query_log>
</clickhouse>
```

By default, table growth is unlimited. To control a size of a table, you can use `TTL` settings for removing outdated log records. Also you can use the partitioning feature of `MergeTree`-engine tables.

## Sources of System Metrics

For collecting system metrics ClickHouse server uses:

- `CAP_NET_ADMIN` capability.
- `procfs` (only in Linux).

### **procfs**

If ClickHouse server does not have `CAP_NET_ADMIN` capability, it tries to fall back to `ProcfsMetricsProvider`. `ProcfsMetricsProvider` allows collecting per-query system metrics (for CPU and I/O).

If `procfs` is supported and enabled on the system, ClickHouse server collects these metrics:

- `OSCPUVirtualTimeMicroseconds`
- `OSCPUWaitMicroseconds`
- `OSIOWaitMicroseconds`
- `OSReadChars`
- `OSWriteChars`
- `OSReadBytes`
- `OSWriteBytes`

---

## system.asynchronous\_metric\_log

Contains the historical values for `system.asynchronous_metrics`, which are saved once per minute. Enabled by default.

Columns:

- `event_date` (`Date`) — Event date.
- `event_time` (`DateTime`) — Event time.
- `event_time_microseconds` (`DateTime64`) — Event time with microseconds resolution.

- `name` ([String](#)) — Metric name.
- `value` ([Float64](#)) — Metric value.

## Example

```
SELECT * FROM system.asynchronous_metric_log LIMIT 10
```

event_date	event_time	event_time_microseconds	name	value
2020-09-05	2020-09-05 15:56:30	2020-09-05 15:56:30.025227	CPUFrequencyMHz_0	2120.9
2020-09-05	2020-09-05 15:56:30	2020-09-05 15:56:30.025227	jemalloc.arenas.all.pmuzzy	743
2020-09-05	2020-09-05 15:56:30	2020-09-05 15:56:30.025227	jemalloc.arenas.all.pdirty	26288
2020-09-05	2020-09-05 15:56:30	2020-09-05 15:56:30.025227	jemalloc.background_thread.run_intervals	0
2020-09-05	2020-09-05 15:56:30	2020-09-05 15:56:30.025227	jemalloc.background_thread.num_runs	0
2020-09-05	2020-09-05 15:56:30	2020-09-05 15:56:30.025227	jemalloc.retained	60694528
2020-09-05	2020-09-05 15:56:30	2020-09-05 15:56:30.025227	jemalloc.mapped	303161344
2020-09-05	2020-09-05 15:56:30	2020-09-05 15:56:30.025227	jemalloc.resident	260931584
2020-09-05	2020-09-05 15:56:30	2020-09-05 15:56:30.025227	jemalloc.metadata	12079488
2020-09-05	2020-09-05 15:56:30	2020-09-05 15:56:30.025227	jemalloc.allocated	133756128

## See Also

- [system.asynchronous\\_metrics](#) — Contains metrics, calculated periodically in the background.
- [system.metric\\_log](#) — Contains history of metrics values from tables `system.metrics` and `system.events`, periodically flushed to disk.

## system.asynchronous\_metrics

Contains metrics that are calculated periodically in the background. For example, the amount of RAM in use.

Columns:

- `metric` ([String](#)) — Metric name.
- `value` ([Float64](#)) — Metric value.

## Example

```
SELECT * FROM system.asynchronous_metrics LIMIT 10
```

metric	value
jemalloc.background_thread.run_interval	0
jemalloc.background_thread.num_runs	0
jemalloc.background_thread.num_threads	0
jemalloc.retained	422551552
jemalloc.mapped	1682989056
jemalloc.resident	1656446976
jemalloc.metadata_thp	0
jemalloc.metadata	10226856
UncompressedCacheCells	0
MarkCacheFiles	0

## See Also

- [Monitoring](#) — Base concepts of ClickHouse monitoring.
- [system.metrics](#) — Contains instantly calculated metrics.
- [system.events](#) — Contains a number of events that have occurred.
- [system.metric\\_log](#) — Contains a history of metrics values from tables `system.metrics` and `system.events`.

# system.clusters

Contains information about clusters available in the config file and the servers in them.

Columns:

- `cluster` ([String](#)) — The cluster name.
- `shard_num` ([UInt32](#)) — The shard number in the cluster, starting from 1.
- `shard_weight` ([UInt32](#)) — The relative weight of the shard when writing data.
- `replica_num` ([UInt32](#)) — The replica number in the shard, starting from 1.
- `host_name` ([String](#)) — The host name, as specified in the config.
- `host_address` ([String](#)) — The host IP address obtained from DNS.
- `port` ([UInt16](#)) — The port to use for connecting to the server.
- `is_local` ([UInt8](#)) — Flag that indicates whether the host is local.
- `user` ([String](#)) — The name of the user for connecting to the server.
- `default_database` ([String](#)) — The default database name.
- `errors_count` ([UInt32](#)) — The number of times this host failed to reach replica.
- `slowdowns_count` ([UInt32](#)) — The number of slowdowns that led to changing replica when establishing a connection with hedged requests.
- `estimated_recovery_time` ([UInt32](#)) — Seconds remaining until the replica error count is zeroed and it is considered to be back to normal.

## Example

Query:

```
SELECT * FROM system.clusters LIMIT 2 FORMAT Vertical;
```

Result:

Row 1:

```
cluster:      test_cluster_two_shards
shard_num:    1
shard_weight: 1
replica_num:  1
host_name:   127.0.0.1
host_address: 127.0.0.1
port:        9000
is_local:    1
user:        default
default_database:
errors_count: 0
slowdowns_count: 0
estimated_recovery_time: 0
```

Row 2:

```
cluster:      test_cluster_two_shards
shard_num:    2
shard_weight: 1
replica_num:  1
host_name:   127.0.0.2
host_address: 127.0.0.2
port:        9000
is_local:    0
user:        default
default_database:
errors_count: 0
slowdowns_count: 0
estimated_recovery_time: 0
```

## See Also

- [Table engine Distributed](#)
- [distributed\\_replica\\_error\\_cap setting](#)
- [distributed\\_replica\\_error\\_half\\_life setting](#)

## system.columns

Contains information about columns in all the tables.

You can use this table to get information similar to the [DESCRIBE TABLE](#) query, but for multiple tables at once.

Columns from [temporary tables](#) are visible in the system.columns only in those session where they have been created. They are shown with the empty `database` field.

Columns:

- `database` ([String](#)) — Database name.
- `table` ([String](#)) — Table name.
- `name` ([String](#)) — Column name.
- `type` ([String](#)) — Column type.
- `position` ([UInt64](#)) — Ordinal position of a column in a table starting with 1.
- `default_kind` ([String](#)) — Expression type (DEFAULT, MATERIALIZED, ALIAS) for the default value, or an empty string if it is not defined.
- `default_expression` ([String](#)) — Expression for the default value, or an empty string if it is not defined.

- `data_compressed_bytes` (`UInt64`) — The size of compressed data, in bytes.
- `data_uncompressed_bytes` (`UInt64`) — The size of decompressed data, in bytes.
- `marks_bytes` (`UInt64`) — The size of marks, in bytes.
- `comment` (`String`) — Comment on the column, or an empty string if it is not defined.
- `is_in_partition_key` (`UInt8`) — Flag that indicates whether the column is in the partition expression.
- `is_in_sorting_key` (`UInt8`) — Flag that indicates whether the column is in the sorting key expression.
- `is_in_primary_key` (`UInt8`) — Flag that indicates whether the column is in the primary key expression.
- `is_in_sampling_key` (`UInt8`) — Flag that indicates whether the column is in the sampling key expression.
- `compression_codec` (`String`) — Compression codec name.

## Example

```
SELECT * FROM system.columns LIMIT 2 FORMAT Vertical;
```

Row 1:

---

```
database:      system
table:        aggregate_function_combinators
name:         name
type:          String
default_kind:
default_expression:
data_compressed_bytes: 0
data_uncompressed_bytes: 0
marks_bytes:    0
comment:
is_in_partition_key: 0
is_in_sorting_key:   0
is_in_primary_key:   0
is_in_sampling_key:  0
compression_codec:
```

Row 2:

---

```
database:      system
table:        aggregate_function_combinators
name:         is_internal
type:          UInt8
default_kind:
default_expression:
data_compressed_bytes: 0
data_uncompressed_bytes: 0
marks_bytes:    0
comment:
is_in_partition_key: 0
is_in_sorting_key:   0
is_in_primary_key:   0
is_in_sampling_key:  0
compression_codec:
```

The `system.columns` table contains the following columns (the column type is shown in brackets):

- `database` (`String`) — Database name.
- `table` (`String`) — Table name.
- `name` (`String`) — Column name.
- `type` (`String`) — Column type.

- `default_kind` (String) — Expression type (DEFAULT, MATERIALIZED, ALIAS) for the default value, or an empty string if it is not defined.
- `default_expression` (String) — Expression for the default value, or an empty string if it is not defined.
- `data_compressed_bytes` (UInt64) — The size of compressed data, in bytes.
- `data_uncompressed_bytes` (UInt64) — The size of decompressed data, in bytes.
- `marks_bytes` (UInt64) — The size of marks, in bytes.
- `comment` (String) — Comment on the column, or an empty string if it is not defined.
- `is_in_partition_key` (UInt8) — Flag that indicates whether the column is in the partition expression.
- `is_in_sorting_key` (UInt8) — Flag that indicates whether the column is in the sorting key expression.
- `is_in_primary_key` (UInt8) — Flag that indicates whether the column is in the primary key expression.
- `is_in_sampling_key` (UInt8) — Flag that indicates whether the column is in the sampling key expression.

## system.contributors

Contains information about contributors. The order is random at query execution time.

Columns:

- `name` (String) — Contributor (author) name from git log.

### Example

```
SELECT * FROM system.contributors LIMIT 10
```

name
Olga Khvostikova
Max Vetrov
LiuYangkuan
svladykin
zamulla
Šimon Podlipský
BayoNet
Ilya Khomutov
Amy Krishnevsky
Loud_Scream

To find out yourself in the table, use a query:

```
SELECT * FROM system.contributors WHERE name = 'Olga Khvostikova'
```

name
Olga Khvostikova

## system.crash\_log

Contains information about stack traces for fatal errors. The table does not exist in the database by default, it is created only when fatal errors occur.

Columns:

- `event_date` ([Datetime](#)) — Date of the event.
- `event_time` ([Datetime](#)) — Time of the event.
- `timestamp_ns` ([UInt64](#)) — Timestamp of the event with nanoseconds.
- `signal` ([Int32](#)) — Signal number.
- `thread_id` ([UInt64](#)) — Thread ID.
- `query_id` ([String](#)) — Query ID.
- `trace` ([Array\(UInt64\)](#)) — Stack trace at the moment of crash. Each element is a virtual memory address inside ClickHouse server process.
- `trace_full` ([Array\(String\)](#)) — Stack trace at the moment of crash. Each element contains a called method inside ClickHouse server process.
- `version` ([String](#)) — ClickHouse server version.
- `revision` ([UInt32](#)) — ClickHouse server revision.
- `build_id` ([String](#)) — BuildID that is generated by compiler.

## Example

Query:

```
SELECT * FROM system.crash_log ORDER BY event_time DESC LIMIT 1;
```

Result (not full):

Row 1:

```
event_date: 2020-10-14
event_time: 2020-10-14 15:47:40
timestamp_ns: 1602679660271312710
signal: 11
thread_id: 23624
query_id: 428aab7c-8f5c-44e9-9607-d16b44467e69
trace: [188531193,...]
trace_full: ['3. DB::(anonymous
namespace)::FunctionFormatReadableTimeDelta::executImpl(std::__1::vector<DB::ColumnWithTypeName,
std::__1::allocator<DB::ColumnWithTypeName> >&, std::__1::vector<unsigned long, std::__1::allocator<unsigned
long> > const&, unsigned long, unsigned long) const @ 0xb3cc1f9 in
/home/username/work/ClickHouse/build/programs/clickhouse',...]
version: ClickHouse 20.11.1.1
revision: 54442
build_id:
```

## See also

- [trace\\_log](#) system table

[Original article](#)

# system.current\_roles

Contains active roles of a current user. [SET ROLE](#) changes the contents of this table.

Columns:

- `role_name` (`String`) — Role name.
- `with_admin_option` (`UInt8`) — Flag that shows whether `current_role` is a role with `ADMIN OPTION` privilege.
- `is_default` (`UInt8`) — Flag that shows whether `current_role` is a default role.

## system.data\_skipping\_indices

Contains information about existing data skipping indices in all the tables.

Columns:

- `database` (`String`) — Database name.
- `table` (`String`) — Table name.
- `name` (`String`) — Index name.
- `type` (`String`) — Index type.
- `expr` (`String`) — Expression for the index calculation.
- `granularity` (`UInt64`) — The number of granules in the block.
- `data_compressed_bytes` (`UInt64`) — The size of compressed data, in bytes.
- `data_uncompressed_bytes` (`UInt64`) — The size of decompressed data, in bytes.
- `marks_bytes` (`UInt64`) — The size of marks, in bytes.

### Example

```
SELECT * FROM system.data_skipping_indices LIMIT 2 FORMAT Vertical;
```

Row 1:

```
database: default
table: user_actions
name: clicks_idx
type: minmax
expr: clicks
granularity: 1
data_compressed_bytes: 58
data_uncompressed_bytes: 6
marks: 48
```

Row 2:

```
database: default
table: users
name: contacts_null_idx
type: minmax
expr: assumeNotNull(contacts_null)
granularity: 1
data_compressed_bytes: 58
data_uncompressed_bytes: 6
marks: 48
```

## system.data\_type\_families

Contains information about supported [data types](#).

Columns:

- `name` ([String](#)) — Data type name.
- `case_insensitive` ([UInt8](#)) — Property that shows whether you can use a data type name in a query in case insensitive manner or not. For example, `Date` and `date` are both valid.
- `alias_to` ([String](#)) — Data type name for which `name` is an alias.

### Example

```
SELECT * FROM system.data_type_families WHERE alias_to = 'String'
```

name	case_insensitive	alias_to
LONGBLOB	1	String
LONGTEXT	1	String
TINYTEXT	1	String
TEXT	1	String
VARCHAR	1	String
MEDIUMBLOB	1	String
BLOB	1	String
TINYBLOB	1	String
CHAR	1	String
MEDIUMTEXT	1	String

### See Also

- [Syntax](#) — Information about supported syntax.

## system.databases

Contains information about the databases that are available to the current user.

Columns:

- `name` ([String](#)) — Database name.
- `engine` ([String](#)) — [Database engine](#).
- `data_path` ([String](#)) — Data path.
- `metadata_path` ([String](#)) — Metadata path.
- `uuid` ([UUID](#)) — Database UUID.

The `name` column from this system table is used for implementing the `SHOW DATABASES` query.

### Example

Create a database.

```
CREATE DATABASE test
```

Check all of the available databases to the user.

```
SELECT * FROM system.databases
```

name	engine	data_path	metadata_path	uuid
_temporary_and_external_tables	Memory	/var/lib/clickhouse/		
00000000-0000-0000-0000-000000000000				
default	Atomic	/var/lib/clickhouse/store/	/var/lib/clickhouse/store/d31/d317b4bd-3595-4386-81ee-c2334694128a/	d317b4bd-3595-4386-81ee-c2334694128a
test	Atomic	/var/lib/clickhouse/store/	/var/lib/clickhouse/store/39b/39bf0cc5-4c06-4717-87fe-c75ff3bd8ebb/	39bf0cc5-4c06-4717-87fe-c75ff3bd8ebb
system	Atomic	/var/lib/clickhouse/store/	/var/lib/clickhouse/store/1d1/1d1c869d-e465-4b1b-a51f-be033436ebf9/	1d1c869d-e465-4b1b-a51f-be033436ebf9

## system.detached\_parts

Contains information about detached parts of [MergeTree](#) tables. The `reason` column specifies why the part was detached.

For user-detached parts, the reason is empty. Such parts can be attached with [ALTER TABLE ATTACH PARTITION|PART](#) command.

For the description of other columns, see [system.parts](#).

If part name is invalid, values of some columns may be NULL. Such parts can be deleted with [ALTER TABLE DROP DETACHED PART](#).

## system.dictionaries

Contains information about [external dictionaries](#).

Columns:

- `database` ([String](#)) — Name of the database containing the dictionary created by DDL query. Empty string for other dictionaries.
- `name` ([String](#)) — Dictionary name.
- `status` ([Enum8](#)) — Dictionary status. Possible values:
  - `NOT_LOADED` — Dictionary was not loaded because it was not used.
  - `LOADED` — Dictionary loaded successfully.
  - `FAILED` — Unable to load the dictionary as a result of an error.
  - `LOADING` — Dictionary is loading now.
  - `LOADED_AND_RELOADING` — Dictionary is loaded successfully, and is being reloaded right now (frequent reasons: [SYSTEM RELOAD DICTIONARY](#) query, timeout, dictionary config has changed).
  - `FAILED_AND_RELOADING` — Could not load the dictionary as a result of an error and is loading now.
- `origin` ([String](#)) — Path to the configuration file that describes the dictionary.
- `type` ([String](#)) — Type of a dictionary allocation. [Storing Dictionaries in Memory](#).
- `key` — **Key type:** Numeric Key ([UInt64](#)) or Composite key ([String](#)) — form “(type 1, type 2, ..., type n)”.
- `attribute.names` ([Array\(String\)](#)) — Array of `attribute names` provided by the dictionary.

- `attribute.types` (`Array(String)`) — Corresponding array of **attribute types** that are provided by the dictionary.
- `bytes_allocated` (`UInt64`) — Amount of RAM allocated for the dictionary.
- `query_count` (`UInt64`) — Number of queries since the dictionary was loaded or since the last successful reboot.
- `hit_rate` (`Float64`) — For cache dictionaries, the percentage of uses for which the value was in the cache.
- `found_rate` (`Float64`) — The percentage of uses for which the value was found.
- `element_count` (`UInt64`) — Number of items stored in the dictionary.
- `load_factor` (`Float64`) — Percentage filled in the dictionary (for a hashed dictionary, the percentage filled in the hash table).
- `source` (`String`) — Text describing the **data source** for the dictionary.
- `lifetime_min` (`UInt64`) — Minimum **lifetime** of the dictionary in memory, after which ClickHouse tries to reload the dictionary (if `invalidate_query` is set, then only if it has changed). Set in seconds.
- `lifetime_max` (`UInt64`) — Maximum **lifetime** of the dictionary in memory, after which ClickHouse tries to reload the dictionary (if `invalidate_query` is set, then only if it has changed). Set in seconds.
- `loading_start_time` (`DateTime`) — Start time for loading the dictionary.
- `last_successful_update_time` (`DateTime`) — End time for loading or updating the dictionary. Helps to monitor some troubles with external sources and investigate causes.
- `loading_duration` (`Float32`) — Duration of a dictionary loading.
- `last_exception` (`String`) — Text of the error that occurs when creating or reloading the dictionary if the dictionary couldn't be created.

## Example

Configure the dictionary.

```
CREATE DICTIONARY dictdb.dict
(
    `key` Int64 DEFAULT -1,
    `value_default` String DEFAULT 'world',
    `value_expression` String DEFAULT 'xxx' EXPRESSION 'toString(127 * 172)'
)
PRIMARY KEY key
SOURCE(CLICKHOUSE(HOST 'localhost' PORT 9000 USER 'default' TABLE 'dicttbl' DB 'dictdb'))
LIFETIME(MIN 0 MAX 1)
LAYOUT(FLAT())
```

Make sure that the dictionary is loaded.

```
SELECT * FROM system.dictionaries
```

database	name	status	origin	type	key	attribute.names	attribute.types	bytes_allocated	query_count	hit_rate	element_count	load_factor	source	loading_duration	last_exception
dictdb	dict	LOADED	dictdb.dict	Flat	UInt64	['value_default','value_expression']	['String','String']	74032	0	1	1	0.0004887585532746823	ClickHouse: dictdb.dicttbl	0	1
2020-03-04 04:17:34			2020-03-04 04:30:34			0.002									

## system.disks

Contains information about disks defined in the [server configuration](#).

Columns:

- `name` ([String](#)) — Name of a disk in the server configuration.
- `path` ([String](#)) — Path to the mount point in the file system.
- `free_space` ([UInt64](#)) — Free space on disk in bytes.
- `total_space` ([UInt64](#)) — Disk volume in bytes.
- `keep_free_space` ([UInt64](#)) — Amount of disk space that should stay free on disk in bytes. Defined in the `keep_free_space_bytes` parameter of disk configuration.

### Example

```
:) SELECT * FROM system.disks;
```

name	path	free_space	total_space	keep_free_space
default	/var/lib/clickhouse/	276392587264	490652508160	0

1 rows in set. Elapsed: 0.001 sec.

## system.distributed\_ddl\_queue

Contains information about [distributed ddl queries \(ON CLUSTER clause\)](#) that were executed on a cluster.

Columns:

- `entry` ([String](#)) — Query id.
- `host_name` ([String](#)) — Hostname.
- `host_address` ([String](#)) — IP address that the Hostname resolves to.
- `port` ([UInt16](#)) — Host Port.
- `status` ([Enum8](#)) — Status of the query.
- `cluster` ([String](#)) — Cluster name.
- `query` ([String](#)) — Query executed.

- `initiator` (`String`) — Node that executed the query.
- `query_start_time` (`DateTime`) — Query start time.
- `query_finish_time` (`DateTime`) — Query finish time.
- `query_duration_ms` (`UInt64`) — Duration of query execution (in milliseconds).
- `exception_code` (`Enum8`) — Exception code from [ZooKeeper](#).

## Example

```
SELECT *
FROM system.distributed_ddl_queue
WHERE cluster = 'test_cluster'
LIMIT 2
FORMAT Vertical
```

Query id: f544e72a-6641-43f1-836b-24baa1c9632a

Row 1:

---

```
entry:      query-000000000000
host_name:  clickhouse01
host_address: 172.23.0.11
port:      9000
status:    Finished
cluster:   test_cluster
query:     CREATE DATABASE test_db UUID '4a82697e-c85e-4e5b-a01e-a36f2a758456' ON CLUSTER test_cluster
initiator: clickhouse01:9000
query_start_time: 2020-12-30 13:07:51
query_finish_time: 2020-12-30 13:07:51
query_duration_ms: 6
exception_code: ZOK
```

Row 2:

---

```
entry:      query-000000000000
host_name:  clickhouse02
host_address: 172.23.0.12
port:      9000
status:    Finished
cluster:   test_cluster
query:     CREATE DATABASE test_db UUID '4a82697e-c85e-4e5b-a01e-a36f2a758456' ON CLUSTER test_cluster
initiator: clickhouse01:9000
query_start_time: 2020-12-30 13:07:51
query_finish_time: 2020-12-30 13:07:51
query_duration_ms: 6
exception_code: ZOK
```

2 rows in set. Elapsed: 0.025 sec.

## system.distribution\_queue

Contains information about local files that are in the queue to be sent to the shards. These local files contain new parts that are created by inserting new data into the Distributed table in asynchronous mode.

Columns:

- `database` (`String`) — Name of the database.
- `table` (`String`) — Name of the table.
- `data_path` (`String`) — Path to the folder with local files.
- `is_blocked` (`UInt8`) — Flag indicates whether sending local files to the server is blocked.

- `error_count` (`UInt64`) — Number of errors.
- `data_files` (`UInt64`) — Number of local files in a folder.
- `data_compressed_bytes` (`UInt64`) — Size of compressed data in local files, in bytes.
- `broken_data_files` (`UInt64`) — Number of files that has been marked as broken (due to an error).
- `broken_data_compressed_bytes` (`UInt64`) — Size of compressed data in broken files, in bytes.
- `last_exception` (`String`) — Text message about the last error that occurred (if any).

## Example

```
SELECT * FROM system.distribution_queue LIMIT 1 FORMAT Vertical;
```

Row 1:

```
database:      default
table:        dist
data_path:    ./store/268/268bc070-3aad-4b1a-9cf2-4987580161af/default@127%2E0%2E0%2E2:9000/
is_blocked:   1
error_count:  0
data_files:   1
data_compressed_bytes: 499
last_exception:
```

## See Also

- [Distributed table engine](#)

## system.enabled\_roles

Contains all active roles at the moment, including current role of the current user and granted roles for current role.

Columns:

- `role_name` (`String`) — Role name.
- `with_admin_option` (`UInt8`) — Flag that shows whether `enabled_role` is a role with `ADMIN OPTION` privilege.
- `is_current` (`UInt8`) — Flag that shows whether `enabled_role` is a current role of a current user.
- `is_default` (`UInt8`) — Flag that shows whether `enabled_role` is a default role.

## system.errors

Contains error codes with the number of times they have been triggered.

Columns:

- `name` (`String`) — name of the error (`errorCodeToName`).
- `code` (`Int32`) — code number of the error.
- `value` (`UInt64`) — the number of times this error has been happened.
- `last_error_time` (`DateTime`) — time when the last error happened.

- `last_error_message` (`String`) — message for the last error.
- `last_error_trace` (`Array(UInt64)`) — A `stack trace` which represents a list of physical addresses where the called methods are stored.
- `remote` (`UInt8`) — remote exception (i.e. received during one of the distributed query).

## Example

```
SELECT name, code, value
FROM system.errors
WHERE value > 0
ORDER BY code ASC
LIMIT 1
```

name	code	value
CANNOT_OPEN_FILE	76	1

```
WITH arrayMap(x -> demangle(addressToSymbol(x)), last_error_trace) AS all
SELECT name, arrayStringConcat(all, '\n') AS res
FROM system.errors
LIMIT 1
SETTINGS allow_introspection_functions=1\G
```

## system.events

Contains information about the number of events that have occurred in the system. For example, in the table, you can find how many `SELECT` queries were processed since the ClickHouse server started.

Columns:

- `event` (`String`) — Event name.
- `value` (`UInt64`) — Number of events occurred.
- `description` (`String`) — Event description.

## Example

```
SELECT * FROM system.events LIMIT 5
```

event	value	description
Query	12	Number of queries to be interpreted and potentially executed. Does not include queries that failed to parse or were rejected due to AST size limits, quota limits or limits on the number of simultaneously running queries. May include internal queries initiated by ClickHouse itself. Does not count subqueries.
SelectQuery	8	Same as Query, but only for SELECT queries.
FileOpen	73	Number of files opened.
ReadBufferFromFileDescriptorRead	155	Number of reads (read/pread) from a file descriptor. Does not include sockets.
ReadBufferFromFileDescriptorReadBytes	9931	Number of bytes read from file descriptors. If the file is compressed, this will show the compressed data size.

## See Also

- [system.asynchronous\\_metrics](#) — Contains periodically calculated metrics.
- [system.metrics](#) — Contains instantly calculated metrics.
- [system.metric\\_log](#) — Contains a history of metrics values from tables `system.metrics` и `system.events`.
- [Monitoring](#) — Base concepts of ClickHouse monitoring.

## system.functions

Contains information about normal and aggregate functions.

Columns:

- `name(String)` — The name of the function.
- `is_aggregate(UInt8)` — Whether the function is aggregate.

### Example

```
SELECT * FROM system.functions LIMIT 10;
```

name	is_aggregate	case_insensitive	alias_to
sumburConsistentHash	0	0	
yandexConsistentHash	0	0	
demangle	0	0	
addressToLine	0	0	
JSONExtractRaw	0	0	
JSONExtractKeysAndValues	0	0	
JSONExtract	0	0	
JSONExtractString	0	0	
JSONExtractFloat	0	0	
JSONExtractInt	0	0	

10 rows in set. Elapsed: 0.002 sec.

## system.grants

Privileges granted to ClickHouse user accounts.

Columns:

- `user_name (Nullable(String))` — User name.
- `role_name (Nullable(String))` — Role assigned to user account.
- `access_type (Enum8)` — Access parameters for ClickHouse user account.
- `database (Nullable(String))` — Name of a database.
- `table (Nullable(String))` — Name of a table.
- `column (Nullable(String))` — Name of a column to which access is granted.
- `is_partial_revoke (UInt8)` — Logical value. It shows whether some privileges have been revoked. Possible values:
  - 0 — The row describes a partial revoke.

- `1` — The row describes a grant.
  - `grant_option` (`UInt8`) — Permission is granted `WITH GRANT OPTION`, see [GRANT](#).
- 

## system.graphite\_retentions

Contains information about parameters `graphite_rollup` which are used in tables with [\\*GraphiteMergeTree](#) engines.

Columns:

- `config_name` (`String`) - `graphite_rollup` parameter name.
  - `regexp` (`String`) - A pattern for the metric name.
  - `function` (`String`) - The name of the aggregating function.
  - `age` (`UInt64`) - The minimum age of the data in seconds.
  - `precision` (`UInt64`) - How precisely to define the age of the data in seconds.
  - `priority` (`UInt16`) - Pattern priority.
  - `is_default` (`UInt8`) - Whether the pattern is the default.
  - `Tables.database` (`Array(String)`) - Array of names of database tables that use the `config_name` parameter.
  - `Tables.table` (`Array(String)`) - Array of table names that use the `config_name` parameter.
- 

## system.licenses

Contains licenses of third-party libraries that are located in the [contrib](#) directory of ClickHouse sources.

Columns:

- `library_name` (`String`) — Name of the library, which is license connected with.
- `license_type` (`String`) — License type — e.g. Apache, MIT.
- `license_path` (`String`) — Path to the file with the license text.
- `license_text` (`String`) — License text.

### Example

```
SELECT library_name, license_type, license_path FROM system.licenses LIMIT 15
```

library_name	license_type	license_path
FastMemcpy	MIT	/contrib/FastMemcpy/LICENSE
arrow	Apache	/contrib/arrow/LICENSE.txt
avro	Apache	/contrib/avro/LICENSE.txt
aws-c-common	Apache	/contrib/aws-c-common/LICENSE
aws-c-event-stream	Apache	/contrib/aws-c-event-stream/LICENSE
aws-checksums	Apache	/contrib/aws-checksums/LICENSE
aws	Apache	/contrib/aws/LICENSE.txt
base64	BSD 2-clause	/contrib/base64/LICENSE
boost	Boost	/contrib/boost/LICENSE_1_0.txt
brotli	MIT	/contrib/brotli/LICENSE
capnproto	MIT	/contrib/capnproto/LICENSE
cassandra	Apache	/contrib/cassandra/LICENSE.txt
cctz	Apache	/contrib/cctz/LICENSE.txt
cityhash102	MIT	/contrib/cityhash102/COPYING
cppkafka	BSD 2-clause	/contrib/cppkafka/LICENSE

## system.merge\_tree\_settings

Contains information about settings for MergeTree tables.

Columns:

- `name` (String) — Setting name.
- `value` (String) — Setting value.
- `description` (String) — Setting description.
- `type` (String) — Setting type (implementation specific string value).
- `changed` (UInt8) — Whether the setting was explicitly defined in the config or explicitly changed.

### Example

```
:) SELECT * FROM system.merge_tree_settings LIMIT 4 FORMAT Vertical;
```

Row 1:

```
name: index_granularity
value: 8192
changed: 0
description: How many rows correspond to one primary key value.
type: SettingUInt64
```

Row 2:

```
name: min_bytes_for_wide_part
value: 0
changed: 0
description: Minimal uncompressed size in bytes to create part in wide format instead of compact
type: SettingUInt64
```

Row 3:

```
name: min_rows_for_wide_part
value: 0
changed: 0
description: Minimal number of rows to create part in wide format instead of compact
type: SettingUInt64
```

Row 4:

```
name: merge_max_block_size
value: 8192
changed: 0
description: How many rows in blocks should be formed for merge operations.
type: SettingUInt64
```

4 rows in set. Elapsed: 0.001 sec.

## system.merges

Contains information about merges and part mutations currently in process for tables in the MergeTree family.

Columns:

- `database` (String) — The name of the database the table is in.
- `table` (String) — Table name.
- `elapsed` (Float64) — The time elapsed (in seconds) since the merge started.
- `progress` (Float64) — The percentage of completed work from 0 to 1.
- `num_parts` (UInt64) — The number of pieces to be merged.
- `result_part_name` (String) — The name of the part that will be formed as the result of merging.
- `is_mutation` (UInt8) — 1 if this process is a part mutation.
- `total_size_bytes_compressed` (UInt64) — The total size of the compressed data in the merged chunks.
- `total_size_marks` (UInt64) — The total number of marks in the merged parts.
- `bytes_read_uncompressed` (UInt64) — Number of bytes read, uncompressed.
- `rows_read` (UInt64) — Number of rows read.
- `bytes_written_uncompressed` (UInt64) — Number of bytes written, uncompressed.
- `rows_written` (UInt64) — Number of rows written.

- `memory_usage` (UInt64) — Memory consumption of the merge process.
- `thread_id` (UInt64) — Thread ID of the merge process.
- `merge_type` — The type of current merge. Empty if it's an mutation.
- `merge_algorithm` — The algorithm used in current merge. Empty if it's an mutation.

## system.metric\_log

Contains history of metrics values from tables `system.metrics` and `system.events`, periodically flushed to disk.

Columns:

- `event_date` (Date) — Event date.
- `event_time` (DateTime) — Event time.
- `event_time_microseconds` (DateTime64) — Event time with microseconds resolution.

### Example

```
SELECT * FROM system.metric_log LIMIT 1 FORMAT Vertical;
```

Row 1:

<code>event_date:</code>	2020-09-05
<code>event_time:</code>	2020-09-05 16:22:33
<code>event_time_microseconds:</code>	2020-09-05 16:22:33.196807
<code>milliseconds:</code>	196
<code>ProfileEvent_Query:</code>	0
<code>ProfileEvent_SelectQuery:</code>	0
<code>ProfileEvent_InsertQuery:</code>	0
<code>ProfileEvent_FailedQuery:</code>	0
<code>ProfileEvent_FailedSelectQuery:</code>	0
...	
...	
<code>CurrentMetric_Revision:</code>	54439
<code>CurrentMetric_VersionInteger:</code>	20009001
<code>CurrentMetric_RWLockWaitingReaders:</code>	0
<code>CurrentMetric_RWLockWaitingWriters:</code>	0
<code>CurrentMetric_RWLockActiveReaders:</code>	0
<code>CurrentMetric_RWLockActiveWriters:</code>	0
<code>CurrentMetric_GlobalThread:</code>	74
<code>CurrentMetric_GlobalThreadActive:</code>	26
<code>CurrentMetric_LocalThread:</code>	0
<code>CurrentMetric_LocalThreadActive:</code>	0
<code>CurrentMetric_DistributedFilesToInsert:</code>	0

### See also

- [metric\\_log setting](#) — Enabling and disabling the setting.
- [system.asynchronous\\_metrics](#) — Contains periodically calculated metrics.
- [system.events](#) — Contains a number of events that occurred.
- [system.metrics](#) — Contains instantly calculated metrics.
- [Monitoring](#) — Base concepts of ClickHouse monitoring.

## system.metrics

Contains metrics which can be calculated instantly, or have a current value. For example, the number of simultaneously processed queries or the current replica delay. This table is always up to date.

Columns:

- `metric` ([String](#)) — Metric name.
- `value` ([Int64](#)) — Metric value.
- `description` ([String](#)) — Metric description.

The list of supported metrics you can find in the [src/Common/CurrentMetrics.cpp](#) source file of ClickHouse.

## Example

```
SELECT * FROM system.metrics LIMIT 10
```

metric	value	description
Query	1	Number of executing queries
Merge	0	Number of executing background merges
PartMutation	0	Number of mutations (ALTER DELETE/UPDATE)
ReplicatedFetch	0	Number of data parts being fetched from replicas
ReplicatedSend	0	Number of data parts being sent to replicas
ReplicatedChecks	0	Number of data parts checking for consistency
BackgroundPoolTask	0	Number of active tasks in BackgroundProcessingPool (merges, mutations, fetches, or replication queue bookkeeping)
BackgroundSchedulePoolTask	0	Number of active tasks in BackgroundSchedulePool. This pool is used for periodic ReplicatedMergeTree tasks, like cleaning old data parts, altering data parts, replica re-initialization, etc.
DiskSpaceReservedForMerge	0	Disk space reserved for currently running background merges. It is slightly more than the total size of currently merging parts.
DistributedSend	0	Number of connections to remote servers sending data that was INSERTed into Distributed tables. Both synchronous and asynchronous mode.

## See Also

- [system.asynchronous\\_metrics](#) — Contains periodically calculated metrics.
- [system.events](#) — Contains a number of events that occurred.
- [system.metric\\_log](#) — Contains a history of metrics values from tables `system.metrics` и `system.events`.
- [Monitoring](#) — Base concepts of ClickHouse monitoring.

## system.mutations

The table contains information about [mutations](#) of [MergeTree](#) tables and their progress. Each mutation command is represented by a single row.

Columns:

- `database` ([String](#)) — The name of the database to which the mutation was applied.

- `table` (`String`) — The name of the table to which the mutation was applied.
- `mutation_id` (`String`) — The ID of the mutation. For replicated tables these IDs correspond to znode names in the `<table_path_in_zookeeper>/mutations/` directory in ZooKeeper. For non-replicated tables the IDs correspond to file names in the data directory of the table.
- `command` (`String`) — The mutation command string (the part of the query after `ALTER TABLE [db.]table`).
- `create_time` (`Datetime`) — Date and time when the mutation command was submitted for execution.
- `block_numbers.partition_id` (`Array(String)`) — For mutations of replicated tables, the array contains the partitions' IDs (one record for each partition). For mutations of non-replicated tables the array is empty.
- `block_numbers.number` (`Array(Int64)`) — For mutations of replicated tables, the array contains one record for each partition, with the block number that was acquired by the mutation. Only parts that contain blocks with numbers less than this number will be mutated in the partition.

In non-replicated tables, block numbers in all partitions form a single sequence. This means that for mutations of non-replicated tables, the column will contain one record with a single block number acquired by the mutation.

- `parts_to_do_names` (`Array(String)`) — An array of names of data parts that need to be mutated for the mutation to complete.
- `parts_to_do` (`Int64`) — The number of data parts that need to be mutated for the mutation to complete.
- `is_done` (`UInt8`) — The flag whether the mutation is done or not. Possible values:
  - `1` if the mutation is completed,
  - `0` if the mutation is still in process.

## Note

Even if `parts_to_do = 0` it is possible that a mutation of a replicated table is not completed yet because of a long-running `INSERT` query, that will create a new data part needed to be mutated.

If there were problems with mutating some data parts, the following columns contain additional information:

- `latest_failed_part` (`String`) — The name of the most recent part that could not be mutated.
- `latest_fail_time` (`Datetime`) — The date and time of the most recent part mutation failure.
- `latest_fail_reason` (`String`) — The exception message that caused the most recent part mutation failure.

## See Also

- [Mutations](#)
- [MergeTree table engine](#)
- [ReplicatedMergeTree family](#)

## system.numbers

This table contains a single `UInt64` column named `number` that contains almost all the natural numbers starting from zero.

You can use this table for tests, or if you need to do a brute force search.

Reads from this table are not parallelized.

### Example

```
:) SELECT * FROM system.numbers LIMIT 10;
```

number
0
1
2
3
4
5
6
7
8
9

10 rows in set. Elapsed: 0.001 sec.

## system.numbers\_mt

The same as [system.numbers](#) but reads are parallelized. The numbers can be returned in any order.

Used for tests.

### Example

```
:) SELECT * FROM system.numbers_mt LIMIT 10;
```

number
0
1
2
3
4
5
6
7
8
9

10 rows in set. Elapsed: 0.001 sec.

## system.one

This table contains a single row with a single dummy UInt8 column containing the value 0.

This table is used if a `SELECT` query does not specify the `FROM` clause.

This is similar to the `DUAL` table found in other DBMSs.

### Example

```
:) SELECT * FROM system.one LIMIT 10;
```

```
dummy
  0 |
```

1 rows in set. Elapsed: 0.001 sec.

## system.opentelemetry\_span\_log

Contains information about [trace spans](#) for executed queries.

Columns:

- `trace_id` ([UUID](#)) — ID of the trace for executed query.
- `span_id` ([UInt64](#)) — ID of the trace span.
- `parent_span_id` ([UInt64](#)) — ID of the parent trace span.
- `operation_name` ([String](#)) — The name of the operation.
- `start_time_us` ([UInt64](#)) — The start time of the `trace span` (in microseconds).
- `finish_time_us` ([UInt64](#)) — The finish time of the `trace span` (in microseconds).
- `finish_date` ([Date](#)) — The finish date of the trace span.
- `attribute.names` ([Array\(String\)](#)) — [Attribute names](#) depending on the `trace span`. They are filled in according to the recommendations in the [OpenTelemetry](#) standard.
- `attribute.values` ([Array\(String\)](#)) — Attribute values depending on the `trace span`. They are filled in according to the recommendations in the [OpenTelemetry](#) standard.

### Example

Query:

```
SELECT * FROM system.opentelemetry_span_log LIMIT 1 FORMAT Vertical;
```

Result:

Row 1:

```
trace_id:      cdab0847-0d62-61d5-4d38-dd65b19a1914
span_id:      701487461015578150
parent_span_id: 2991972114672045096
operation_name: DB::Block DB::InterpreterSelectQuery::getSampleBlockImpl()
start_time_us: 1612374594529090
finish_time_us: 1612374594529108
finish_date:   2021-02-03
attribute.names: []
attribute.values: []
```

### See Also

- [OpenTelemetry](#)

## system.part\_log

The `system.part_log` table is created only if the [part\\_log](#) server setting is specified.

This table contains information about events that occurred with [data parts](#) in the [MergeTree](#) family tables, such as adding or merging data.

The `system.part_log` table contains the following columns:

- `query_id` ([String](#)) — Identifier of the `INSERT` query that created this data part.
- `event_type` ([Enum8](#)) — Type of the event that occurred with the data part. Can have one of the following values:
  - `NEW_PART` — Inserting of a new data part.
  - `MERGE_PARTS` — Merging of data parts.
  - `DOWNLOAD_PART` — Downloading a data part.
  - `REMOVE_PART` — Removing or detaching a data part using [DETACH PARTITION](#).
  - `MUTATE_PART` — Mutating of a data part.
  - `MOVE_PART` — Moving the data part from the one disk to another one.
- `event_date` ([Date](#)) — Event date.
- `event_time` ([DateTime](#)) — Event time.
- `event_time_microseconds` ([DateTime64](#)) — Event time with microseconds precision.
- `duration_ms` ([UInt64](#)) — Duration.
- `database` ([String](#)) — Name of the database the data part is in.
- `table` ([String](#)) — Name of the table the data part is in.
- `part_name` ([String](#)) — Name of the data part.
- `partition_id` ([String](#)) — ID of the partition that the data part was inserted to. The column takes the `all` value if the partitioning is by `tuple()`.
- `path_on_disk` ([String](#)) — Absolute path to the folder with data part files.
- `rows` ([UInt64](#)) — The number of rows in the data part.
- `size_in_bytes` ([UInt64](#)) — Size of the data part in bytes.
- `merged_from` ([Array\(String\)](#)) — An array of names of the parts which the current part was made up from (after the merge).
- `bytes_uncompressed` ([UInt64](#)) — Size of uncompressed bytes.
- `read_rows` ([UInt64](#)) — The number of rows was read during the merge.
- `read_bytes` ([UInt64](#)) — The number of bytes was read during the merge.
- `peak_memory_usage` ([Int64](#)) — The maximum difference between the amount of allocated and freed memory in context of this thread.
- `error` ([UInt16](#)) — The code number of the occurred error.
- `exception` ([String](#)) — Text message of the occurred error.

The `system.part_log` table is created after the first inserting data to the [MergeTree](#) table.

## Example

```
SELECT * FROM system.part_log LIMIT 1 FORMAT Vertical;
```

Row 1:

```
query_id: 983ad9c7-28d5-4ae1-844e-603116b7de31
event_type: NewPart
event_date: 2021-02-02
event_time: 2021-02-02 11:14:28
event_time_microseconds: 2021-02-02 11:14:28.861919
duration_ms: 35
database: default
table: log_mt_2
part_name: all_1_1_0
partition_id: all
path_on_disk: db/data/default/log_mt_2/all_1_1_0/
rows: 115418
size_in_bytes: 1074311
merged_from: []
bytes_uncompressed: 0
read_rows: 0
read_bytes: 0
peak_memory_usage: 0
error: 0
exception:
```

## system.parts

Contains information about parts of [MergeTree](#) tables.

Each row describes one data part.

Columns:

- `partition` ([String](#)) – The partition name. To learn what a partition is, see the description of the [ALTER](#) query.

Formats:

- `YYYYMM` for automatic partitioning by month.
- `any_string` when partitioning manually.
- `name` ([String](#)) – Name of the data part.
- `part_type` ([String](#)) — The data part storing format.

Possible Values:

- `Wide` — Each column is stored in a separate file in a filesystem.
- `Compact` — All columns are stored in one file in a filesystem.

Data storing format is controlled by the `min_bytes_for_wide_part` and `min_rows_for_wide_part` settings of the [MergeTree](#) table.

- `active` ([UInt8](#)) – Flag that indicates whether the data part is active. If a data part is active, it's used in a table. Otherwise, it's deleted. Inactive data parts remain after merging.
- `marks` ([UInt64](#)) – The number of marks. To get the approximate number of rows in a data part, multiply marks by the index granularity (usually 8192) (this hint does not work for adaptive granularity).
- `rows` ([UInt64](#)) – The number of rows.
- `bytes_on_disk` ([UInt64](#)) – Total size of all the data part files in bytes.

- `data_compressed_bytes` (`UInt64`) – Total size of compressed data in the data part. All the auxiliary files (for example, files with marks) are not included.
- `data_uncompressed_bytes` (`UInt64`) – Total size of uncompressed data in the data part. All the auxiliary files (for example, files with marks) are not included.
- `marks_bytes` (`UInt64`) – The size of the file with marks.
- `secondary_indices_compressed_bytes` (`UInt64`) – Total size of compressed data for secondary indices in the data part. All the auxiliary files (for example, files with marks) are not included.
- `secondary_indices_uncompressed_bytes` (`UInt64`) – Total size of uncompressed data for secondary indices in the data part. All the auxiliary files (for example, files with marks) are not included.
- `secondary_indices_marks_bytes` (`UInt64`) – The size of the file with marks for secondary indices.
- `modification_time` (`DateTime`) – The time the directory with the data part was modified. This usually corresponds to the time of data part creation.
- `remove_time` (`DateTime`) – The time when the data part became inactive.
- `refcount` (`UInt32`) – The number of places where the data part is used. A value greater than 2 indicates that the data part is used in queries or merges.
- `min_date` (`Date`) – The minimum value of the date key in the data part.
- `max_date` (`Date`) – The maximum value of the date key in the data part.
- `min_time` (`DateTime`) – The minimum value of the date and time key in the data part.
- `max_time` (`DateTime`) – The maximum value of the date and time key in the data part.
- `partition_id` (`String`) – ID of the partition.
- `min_block_number` (`UInt64`) – The minimum number of data parts that make up the current part after merging.
- `max_block_number` (`UInt64`) – The maximum number of data parts that make up the current part after merging.
- `level` (`UInt32`) – Depth of the merge tree. Zero means that the current part was created by insert rather than by merging other parts.
- `data_version` (`UInt64`) – Number that is used to determine which mutations should be applied to the data part (mutations with a version higher than `data_version`).
- `primary_key_bytes_in_memory` (`UInt64`) – The amount of memory (in bytes) used by primary key values.
- `primary_key_bytes_in_memory_allocated` (`UInt64`) – The amount of memory (in bytes) reserved for primary key values.
- `is_frozen` (`UInt8`) – Flag that shows that a partition data backup exists. 1, the backup exists. 0, the backup does not exist. For more details, see [FREEZE PARTITION](#)
- `database` (`String`) – Name of the database.
- `table` (`String`) – Name of the table.
- `engine` (`String`) – Name of the table engine without parameters.
- `path` (`String`) – Absolute path to the folder with data part files.

- `disk` (`String`) – Name of a disk that stores the data part.
- `hash_of_all_files` (`String`) – `sipHash128` of compressed files.
- `hash_of_uncompressed_files` (`String`) – `sipHash128` of uncompressed files (files with marks, index file etc.).
- `uncompressed_hash_of_compressed_files` (`String`) – `sipHash128` of data in the compressed files as if they were uncompressed.
- `delete_ttl_info_min` (`DateTime`) — The minimum value of the date and time key for **TTL DELETE rule**.
- `delete_ttl_info_max` (`DateTime`) — The maximum value of the date and time key for **TTL DELETE rule**.
- `move_ttl_info.expression` (`Array(String)`) — Array of expressions. Each expression defines a **TTL MOVE rule**.

## Warning

The `move_ttl_info.expression` array is kept mostly for backward compatibility, now the simplest way to check **TTL MOVE rule** is to use the `move_ttl_info.min` and `move_ttl_info.max` fields.

- `move_ttl_info.min` (`Array(DateTime)`) — Array of date and time values. Each element describes the minimum key value for a **TTL MOVE rule**.
- `move_ttl_info.max` (`Array(DateTime)`) — Array of date and time values. Each element describes the maximum key value for a **TTL MOVE rule**.
- `bytes` (`UInt64`) – Alias for `bytes_on_disk`.
- `marks_size` (`UInt64`) – Alias for `marks_bytes`.

## Example

```
SELECT * FROM system.parts LIMIT 1 FORMAT Vertical;
```

Row 1:

```
partition:          tuple()
name:              all_1_4_1_6
part_type:         Wide
active:            1
marks:             2
rows:              6
bytes_on_disk:    310
data_compressed_bytes: 157
data_uncompressed_bytes: 91
secondary_indices_compressed_bytes: 58
secondary_indices_uncompressed_bytes: 6
secondary_indices_marks_bytes: 48
marks_bytes:       144
modification_time: 2020-06-18 13:01:49
remove_time:       1970-01-01 00:00:00
refcount:          1
min_date:          1970-01-01
max_date:          1970-01-01
min_time:          1970-01-01 00:00:00
max_time:          1970-01-01 00:00:00
partition_id:      all
min_block_number:  1
max_block_number:  4
level:             1
data_version:     6
primary_key_bytes_in_memory: 8
primary_key_bytes_in_memory_allocated: 64
is_frozen:         0
database:          default
table:             months
engine:            MergeTree
disk_name:         default
path:              /var/lib/clickhouse/data/default/months/all_1_4_1_6/
hash_of_all_files: 2d0657a16d9430824d35e327fcdbd87bf
hash_of_uncompressed_files: 84950cc30ba867c77a408ae21332ba29
uncompressed_hash_of_compressed_files: 1ad78f1c6843bbfb99a2c931abe7df7d
delete_ttl_info_min: 1970-01-01 00:00:00
delete_ttl_info_max: 1970-01-01 00:00:00
move_ttl_info.expression: []
move_ttl_info.min: []
move_ttl_info.max: []
```

## See Also

- [MergeTree family](#)
- [TTL for Columns and Tables](#)

## system.parts\_columns

Contains information about parts and columns of [MergeTree](#) tables.

Each row describes one data part.

Columns:

- **partition** ([String](#)) — The partition name. To learn what a partition is, see the description of the [ALTER](#) query.

Formats:

- `YYYYMM` for automatic partitioning by month.
- `any_string` when partitioning manually.
- **name** ([String](#)) — Name of the data part.

- `part_type` ([String](#)) — The data part storing format.

Possible values:

- `Wide` — Each column is stored in a separate file in a filesystem.
  - `Compact` — All columns are stored in one file in a filesystem.
- Data storing format is controlled by the `min_bytes_for_wide_part` and `min_rows_for_wide_part` settings of the [MergeTree](#) table.
- `active` ([UInt8](#)) — Flag that indicates whether the data part is active. If a data part is active, it's used in a table. Otherwise, it's deleted. Inactive data parts remain after merging.
  - `marks` ([UInt64](#)) — The number of marks. To get the approximate number of rows in a data part, multiply `marks` by the index granularity (usually 8192) (this hint does not work for adaptive granularity).
  - `rows` ([UInt64](#)) — The number of rows.
  - `bytes_on_disk` ([UInt64](#)) — Total size of all the data part files in bytes.
  - `data_compressed_bytes` ([UInt64](#)) — Total size of compressed data in the data part. All the auxiliary files (for example, files with marks) are not included.
  - `data_uncompressed_bytes` ([UInt64](#)) — Total size of uncompressed data in the data part. All the auxiliary files (for example, files with marks) are not included.
  - `marks_bytes` ([UInt64](#)) — The size of the file with marks.
  - `modification_time` ([DateTime](#)) — The time the directory with the data part was modified. This usually corresponds to the time of data part creation.
  - `remove_time` ([DateTime](#)) — The time when the data part became inactive.
  - `refcount` ([UInt32](#)) — The number of places where the data part is used. A value greater than 2 indicates that the data part is used in queries or merges.
  - `min_date` ([Date](#)) — The minimum value of the date key in the data part.
  - `max_date` ([Date](#)) — The maximum value of the date key in the data part.
  - `partition_id` ([String](#)) — ID of the partition.
  - `min_block_number` ([UInt64](#)) — The minimum number of data parts that make up the current part after merging.
  - `max_block_number` ([UInt64](#)) — The maximum number of data parts that make up the current part after merging.
  - `level` ([UInt32](#)) — Depth of the merge tree. Zero means that the current part was created by insert rather than by merging other parts.
  - `data_version` ([UInt64](#)) — Number that is used to determine which mutations should be applied to the data part (mutations with a version higher than `data_version`).
  - `primary_key_bytes_in_memory` ([UInt64](#)) — The amount of memory (in bytes) used by primary key values.
  - `primary_key_bytes_in_memory_allocated` ([UInt64](#)) — The amount of memory (in bytes) reserved for primary key values.
  - `database` ([String](#)) — Name of the database.
  - `table` ([String](#)) — Name of the table.

- `engine` ([String](#)) — Name of the table engine without parameters.
- `disk_name` ([String](#)) — Name of a disk that stores the data part.
- `path` ([String](#)) — Absolute path to the folder with data part files.
- `column` ([String](#)) — Name of the column.
- `type` ([String](#)) — Column type.
- `column_position` ([UInt64](#)) — Ordinal position of a column in a table starting with 1.
- `default_kind` ([String](#)) — Expression type (DEFAULT, MATERIALIZED, ALIAS) for the default value, or an empty string if it is not defined.
- `default_expression` ([String](#)) — Expression for the default value, or an empty string if it is not defined.
- `column_bytes_on_disk` ([UInt64](#)) — Total size of the column in bytes.
- `column_data_compressed_bytes` ([UInt64](#)) — Total size of compressed data in the column, in bytes.
- `column_data_uncompressed_bytes` ([UInt64](#)) — Total size of the decompressed data in the column, in bytes.
- `column_marks_bytes` ([UInt64](#)) — The size of the column with marks, in bytes.
- `bytes` ([UInt64](#)) — Alias for `bytes_on_disk`.
- `marks_size` ([UInt64](#)) — Alias for `marks_bytes`.

## Example

```
SELECT * FROM system.parts_columns LIMIT 1 FORMAT Vertical;
```

Row 1:

```
partition:          tuple()
name:              all_1_2_1
part_type:         Wide
active:            1
marks:             2
rows:              2
bytes_on_disk:    155
data_compressed_bytes: 56
data_uncompressed_bytes: 4
marks_bytes:       96
modification_time: 2020-09-23 10:13:36
remove_time:       2106-02-07 06:28:15
refcount:          1
min_date:          1970-01-01
max_date:          1970-01-01
partition_id:      all
min_block_number:  1
max_block_number:  2
level:             1
data_version:     1
primary_key_bytes_in_memory: 2
primary_key_bytes_in_memory_allocated: 64
database:          default
table:             53r93yleapyears
engine:            MergeTree
disk_name:         default
path:              /var/lib/clickhouse/data/default/53r93yleapyears/all_1_2_1/
column:            id
type:              Int8
column_position:  1
default_kind:      default
default_expression:
column_bytes_on_disk: 76
column_data_compressed_bytes: 28
column_data_uncompressed_bytes: 2
column_marks_bytes: 48
```

## See Also

- [MergeTree family](#)

# system.processes

This system table is used for implementing the SHOW PROCESSLIST query.

Columns:

- **user** (String) – The user who made the query. Keep in mind that for distributed processing, queries are sent to remote servers under the `default` user. The field contains the username for a specific query, not for a query that this query initiated.
- **address** (String) – The IP address the request was made from. The same for distributed processing. To track where a distributed query was originally made from, look at `system.processes` on the query requestor server.
- **elapsed** (Float64) – The time in seconds since request execution started.
- **rows\_read** (UInt64) – The number of rows read from the table. For distributed processing, on the requestor server, this is the total for all remote servers.
- **bytes\_read** (UInt64) – The number of uncompressed bytes read from the table. For distributed processing, on the requestor server, this is the total for all remote servers.

- `total_rows_approx` (UInt64) – The approximation of the total number of rows that should be read. For distributed processing, on the requestor server, this is the total for all remote servers. It can be updated during request processing, when new sources to process become known.
- `memory_usage` (UInt64) – Amount of RAM the request uses. It might not include some types of dedicated memory. See the [max\\_memory\\_usage](#) setting.
- `query` (String) – The query text. For `INSERT`, it does not include the data to insert.
- `query_id` (String) – Query ID, if defined.

```
:) SELECT * FROM system.processes LIMIT 10 FORMAT Vertical;
```

Row 1:

```
is_initial_query: 1
user: default
query_id: 35a360fa-3743-441d-8e1f-228c938268da
address: ::ffff:172.23.0.1
port: 47588
initial_user: default
initial_query_id: 35a360fa-3743-441d-8e1f-228c938268da
initial_address: ::ffff:172.23.0.1
initial_port: 47588
interface: 1
os_user: bharatnc
client_hostname: tower
client_name: ClickHouse
client_revision: 54437
client_version_major: 20
client_version_minor: 7
client_version_patch: 2
http_method: 0
http_user_agent:
quota_key:
elapsed: 0.000582537
is_cancelled: 0
read_rows: 0
read_bytes: 0
total_rows_approx: 0
written_rows: 0
written_bytes: 0
memory_usage: 0
peak_memory_usage: 0
query: SELECT * from system.processes LIMIT 10 FORMAT Vertical;
thread_ids: [67]
ProfileEvents:
{'Query':1,'SelectQuery':1,'ReadCompressedBytes':36,'CompressedReadBufferBlocks':1,'CompressedReadBufferBytes':10,'IOBufferAllocs':1,'IOBufferAllocBytes':89,'ContextLock':15,'RWLockAcquiredReadLocks':1}
Settings:
{'background_pool_size':'32','load_balancing':'random','allow_suspicious_low_cardinality_types':'1','distributed_aggregation_memory_efficient':'1','skip_unavailable_shards':'1','log_queries':'1','max_bytes_before_external_group_by':'20000000000','max_bytes_before_external_sort':'200000000000','allow_introspection_functions':'1'}
1 rows in set. Elapsed: 0.002 sec.
```

## system.query\_log

Contains information about executed queries, for example, start time, duration of processing, error messages.

### Note

This table does not contain the ingested data for `INSERT` queries.

You can change settings of queries logging in the `query_log` section of the server configuration.

You can disable queries logging by setting `log_queries = 0`. We do not recommend to turn off logging because information in this table is important for solving issues.

The flushing period of data is set in `flush_interval_milliseconds` parameter of the `query_log` server settings section. To force flushing, use the `SYSTEM FLUSH LOGS` query.

ClickHouse does not delete data from the table automatically. See [Introduction](#) for more details.

The `system.query_log` table registers two kinds of queries:

1. Initial queries that were run directly by the client.
2. Child queries that were initiated by other queries (for distributed query execution). For these types of queries, information about the parent queries is shown in the `initial_*` columns.

Each query creates one or two rows in the `query_log` table, depending on the status (see the `type` column) of the query:

1. If the query execution was successful, two rows with the `QueryStart` and `QueryFinish` types are created.
2. If an error occurred during query processing, two events with the `QueryStart` and `ExceptionWhileProcessing` types are created.
3. If an error occurred before launching the query, a single event with the `ExceptionBeforeStart` type is created.

You can use the `log_queries_probability` setting to reduce the number of queries, registered in the `query_log` table.

Columns:

- `type` (`Enum8`) — Type of an event that occurred when executing the query. Values:
  - `'QueryStart' = 1` — Successful start of query execution.
  - `'QueryFinish' = 2` — Successful end of query execution.
  - `'ExceptionBeforeStart' = 3` — Exception before the start of query execution.
  - `'ExceptionWhileProcessing' = 4` — Exception during the query execution.
- `event_date` (`Date`) — Query starting date.
- `event_time` (`DateTime`) — Query starting time.
- `event_time_microseconds` (`DateTime`) — Query starting time with microseconds precision.
- `query_start_time` (`DateTime`) — Start time of query execution.
- `query_start_time_microseconds` (`DateTime64`) — Start time of query execution with microsecond precision.
- `query_duration_ms` (`UInt64`) — Duration of query execution in milliseconds.
- `read_rows` (`UInt64`) — Total number of rows read from all tables and table functions participated in query. It includes usual subqueries, subqueries for `IN` and `JOIN`. For distributed queries `read_rows` includes the total number of rows read at all replicas. Each replica sends its `read_rows` value, and the server-initiator of the query summarizes all received and local values. The cache volumes do not affect this value.
- `read_bytes` (`UInt64`) — Total number of bytes read from all tables and table functions participated in query. It includes usual subqueries, subqueries for `IN` and `JOIN`. For distributed queries `read_bytes` includes the total number of rows read at all replicas. Each replica sends its `read_bytes` value, and the server-initiator of the query summarizes all received and local values. The cache volumes do not affect this value.

- `written_rows` (`UInt64`) — For `INSERT` queries, the number of written rows. For other queries, the column value is 0.
- `written_bytes` (`UInt64`) — For `INSERT` queries, the number of written bytes. For other queries, the column value is 0.
- `result_rows` (`UInt64`) — Number of rows in a result of the `SELECT` query, or a number of rows in the `INSERT` query.
- `result_bytes` (`UInt64`) — RAM volume in bytes used to store a query result.
- `memory_usage` (`UInt64`) — Memory consumption by the query.
- `current_database` (`String`) — Name of the current database.
- `query` (`String`) — Query string.
- `normalized_query_hash` (`UInt64`) — Identical hash value without the values of literals for similar queries.
- `query_kind` (`LowCardinality(String)`) — Type of the query.
- `databases` (`Array(LowCardinality(String))`) — Names of the databases present in the query.
- `tables` (`Array(LowCardinality(String))`) — Names of the tables present in the query.
- `views` (`Array(LowCardinality(String))`) — Names of the (materialized or live) views present in the query.
- `columns` (`Array(LowCardinality(String))`) — Names of the columns present in the query.
- `projections` (`String`) — Names of the projections used during the query execution.
- `exception_code` (`Int32`) — Code of an exception.
- `exception` (`String`) — Exception message.
- `stack_trace` (`String`) — **Stack trace**. An empty string, if the query was completed successfully.
- `is_initial_query` (`UInt8`) — Query type. Possible values:
  - 1 — Query was initiated by the client.
  - 0 — Query was initiated by another query as part of distributed query execution.
- `user` (`String`) — Name of the user who initiated the current query.
- `query_id` (`String`) — ID of the query.
- `address` (`IPv6`) — IP address that was used to make the query.
- `port` (`UInt16`) — The client port that was used to make the query.
- `initial_user` (`String`) — Name of the user who ran the initial query (for distributed query execution).
- `initial_query_id` (`String`) — ID of the initial query (for distributed query execution).
- `initial_address` (`IPv6`) — IP address that the parent query was launched from.
- `initial_port` (`UInt16`) — The client port that was used to make the parent query.
- `initial_query_start_time` (`DateTime`) — Initial query starting time (for distributed query execution).
- `initial_query_start_time_microseconds` (`DateTime64`) — Initial query starting time with microseconds precision (for distributed query execution).

- `interface` (`UInt8`) — Interface that the query was initiated from. Possible values:
  - 1 — TCP.
  - 2 — HTTP.
- `os_user` (`String`) — Operating system username who runs `clickhouse-client`.
- `client_hostname` (`String`) — Hostname of the client machine where the `clickhouse-client` or another TCP client is run.
- `client_name` (`String`) — The `clickhouse-client` or another TCP client name.
- `client_revision` (`UInt32`) — Revision of the `clickhouse-client` or another TCP client.
- `client_version_major` (`UInt32`) — Major version of the `clickhouse-client` or another TCP client.
- `client_version_minor` (`UInt32`) — Minor version of the `clickhouse-client` or another TCP client.
- `client_version_patch` (`UInt32`) — Patch component of the `clickhouse-client` or another TCP client version.
- `http_method` (`UInt8`) — HTTP method that initiated the query. Possible values:
  - 0 — The query was launched from the TCP interface.
  - 1 — `GET` method was used.
  - 2 — `POST` method was used.
- `http_user_agent` (`String`) — HTTP header `UserAgent` passed in the HTTP query.
- `http_referer` (`String`) — HTTP header `Referer` passed in the HTTP query (contains an absolute or partial address of the page making the query).
- `forwarded_for` (`String`) — HTTP header `X-Forwarded-For` passed in the HTTP query.
- `quota_key` (`String`) — The quota key specified in the `quotas` setting (see `keyed`).
- `revision` (`UInt32`) — ClickHouse revision.
- `ProfileEvents` (`Map(String, UInt64)`) — ProfileEvents that measure different metrics. The description of them could be found in the table `system.events`
- `Settings` (`Map(String, String)`) — Settings that were changed when the client ran the query. To enable logging changes to settings, set the `log_query_settings` parameter to 1.
- `log_comment` (`String`) — Log comment. It can be set to arbitrary string no longer than `max_query_size`. An empty string if it is not defined.
- `thread_ids` (`Array(UInt64)`) — Thread ids that are participating in query execution.
- `used_aggregate_functions` (`Array(String)`) — Canonical names of `aggregate functions`, which were used during query execution.
- `used_aggregate_function_combinators` (`Array(String)`) — Canonical names of `aggregate functions combinators`, which were used during query execution.
- `used_database_engines` (`Array(String)`) — Canonical names of `database engines`, which were used during query execution.
- `used_data_type_families` (`Array(String)`) — Canonical names of `data type families`, which were used during query execution.

- `used_dictionaries` (`Array(String)`) — Canonical names of `dictionaries`, which were used during query execution.
- `used_formats` (`Array(String)`) — Canonical names of `formats`, which were used during query execution.
- `used_functions` (`Array(String)`) — Canonical names of `functions`, which were used during query execution.
- `used_storages` (`Array(String)`) — Canonical names of `storages`, which were used during query execution.
- `used_table_functions` (`Array(String)`) — Canonical names of `table functions`, which were used during query execution.

## Example

```
SELECT * FROM system.query_log WHERE type = 'QueryFinish' ORDER BY query_start_time DESC LIMIT 1 FORMAT Vertical;
```

Row 1:

```

type:          QueryFinish
event_date:    2021-07-28
event_time:    2021-07-28 13:46:56
event_time_microseconds: 2021-07-28 13:46:56.719791
query_start_time: 2021-07-28 13:46:56
query_start_time_microseconds: 2021-07-28 13:46:56.704542
query_duration_ms: 14
read_rows:     8393
read_bytes:   374325
written_rows: 0
written_bytes: 0
result_rows:  4201
result_bytes: 153024
memory_usage: 4714038
current_database: default
query:          SELECT DISTINCT arrayJoin(extractAll(name, '[\\w_]{2,}') AS res FROM (SELECT name FROM system.functions UNION ALL SELECT name FROM system.table_engines UNION ALL SELECT name FROM system.formats UNION ALL SELECT name FROM system.table_functions UNION ALL SELECT name FROM system.data_type_families UNION ALL SELECT name FROM system.merge_tree_settings UNION ALL SELECT name FROM system.settings UNION ALL SELECT cluster FROM system.clusters UNION ALL SELECT macro FROM system.macros UNION ALL SELECT policy_name FROM system.storage_policies UNION ALL SELECT concat(func.name, comb.name) FROM system.functions AS func CROSS JOIN system.aggregate_function_combinators AS comb WHERE is_aggregate UNION ALL SELECT name FROM system.databases LIMIT 10000 UNION ALL SELECT DISTINCT name FROM system.tables LIMIT 10000 UNION ALL SELECT DISTINCT name FROM system.dictionaries LIMIT 10000 UNION ALL SELECT DISTINCT name FROM system.columns LIMIT 10000) WHERE notEmpty(res)
normalized_query_hash: 6666026786019643712
query_kind:      Select
databases:       ['system']
tables:          ['system.aggregate_function_combinators','system.clusters','system.columns','system.data_type_families','system.dataTables','system.dictionaries','system.formats','system.functions','system.macros','system.merge_tree_settings','system.settings','system.storage_policies','system.table_engines','system.table_functions','system.tables']
columns:         ['system.aggregate_function_combinators.name','system.clusters.cluster','system.columns.name','system.data_type_families.name','system.dataTables.name','system.dictionaries.name','system.formats.name','system.functions.is_aggregate','system.functions.name','system.macros.macro','system.merge_tree_settings.name','system.settings.name','system.storage_policies.policy_name','system.table_engines.name','system.table_functions.name','system.tables.name']
projections:     []
exception_code:  0
exception:
stack_trace:
is_initial_query: 1
user:            default
query_id:        a3361f6e-a1fd-4d54-9f6f-f93a08bab0bf
address:         ::ffff:127.0.0.1
port:            51006
initial_user:    default
initial_query_id: a3361f6e-a1fd-4d54-9f6f-f93a08bab0bf
initial_address: ::ffff:127.0.0.1
initial_port:    51006
initial_query_start_time: 2021-07-28 13:46:56

```

```

initial_query_start_time_microseconds: 2021-07-28 13:46:56.704542
interface: 1
os_user:
client_hostname:
client_name: ClickHouse client
client_revision: 54449
client_version_major: 21
client_version_minor: 8
client_version_patch: 0
http_method: 0
http_user_agent:
http_referer:
forwarded_for:
quota_key:
revision: 54453
log_comment:
thread_ids: [5058,22097,22110,22094]
ProfileEvents.Names:
['Query','SelectQuery','ArenaAllocChunks','ArenaAllocBytes','FunctionExecute','NetworkSendElapsedMicroseconds','SelectedRows','SelectedBytes','ContextLock','RWLockAcquiredReadLocks','RealTimeMicroseconds','UserTimeMicroseconds','SystemTimeMicroseconds','SoftPageFaults','OSCPUWaitMicroseconds','OSCPUVirtualTimeMicroseconds','OSWriteBytes','OSWriteChars']
ProfileEvents.Values:
[1,1,39,352256,64,360,8393,374325,412,440,34480,13108,4723,671,19,17828,8192,10240]
Settings.Names: ['load_balancing','max_memory_usage']
Settings.Values: ['random','10000000000']
used_aggregate_functions: []
used_aggregate_function_combinators: []
used_database_engines: []
used_data_type_families: ['UInt64','UInt8','Nullable','String','date']
used_dictionaries: []
used_formats: []
used_functions: ['concat','notEmpty','extractAll']
used_storages: []
used_table_functions: []

```

## See Also

- [system.query\\_thread\\_log](#) — This table contains information about each query execution thread.
- [system.query\\_views\\_log](#) — This table contains information about each view executed during a query.

## system.query\_thread\_log

Contains information about threads that execute queries, for example, thread name, thread start time, duration of query processing.

To start logging:

1. Configure parameters in the [query\\_thread\\_log](#) section.
2. Set [log\\_query\\_threads](#) to 1.

The flushing period of data is set in [flush\\_interval\\_milliseconds](#) parameter of the [query\\_thread\\_log](#) server settings section. To force flushing, use the [SYSTEM FLUSH LOGS](#) query.

ClickHouse does not delete data from the table automatically. See [Introduction](#) for more details.

You can use the [log\\_queries\\_probability](#) setting to reduce the number of queries, registered in the [query\\_thread\\_log](#) table.

Columns:

- [event\\_date](#) ([Date](#)) — The date when the thread has finished execution of the query.
- [event\\_time](#) ([DateTime](#)) — The date and time when the thread has finished execution of the query.

- `event_time_microseconds` (`DateTime`) — The date and time when the thread has finished execution of the query with microseconds precision.
- `query_start_time` (`DateTime`) — Start time of query execution.
- `query_start_time_microseconds` (`DateTime64`) — Start time of query execution with microsecond precision.
- `query_duration_ms` (`UInt64`) — Duration of query execution.
- `read_rows` (`UInt64`) — Number of read rows.
- `read_bytes` (`UInt64`) — Number of read bytes.
- `written_rows` (`UInt64`) — For `INSERT` queries, the number of written rows. For other queries, the column value is 0.
- `written_bytes` (`UInt64`) — For `INSERT` queries, the number of written bytes. For other queries, the column value is 0.
- `memory_usage` (`Int64`) — The difference between the amount of allocated and freed memory in context of this thread.
- `peak_memory_usage` (`Int64`) — The maximum difference between the amount of allocated and freed memory in context of this thread.
- `thread_name` (`String`) — Name of the thread.
- `thread_number` (`UInt32`) — Internal thread ID.
- `thread_id` (`Int32`) — thread ID.
- `master_thread_id` (`UInt64`) — OS initial ID of initial thread.
- `query` (`String`) — Query string.
- `is_initial_query` (`UInt8`) — Query type. Possible values:
  - 1 — Query was initiated by the client.
  - 0 — Query was initiated by another query for distributed query execution.
- `user` (`String`) — Name of the user who initiated the current query.
- `query_id` (`String`) — ID of the query.
- `address` (`IPv6`) — IP address that was used to make the query.
- `port` (`UInt16`) — The client port that was used to make the query.
- `initial_user` (`String`) — Name of the user who ran the initial query (for distributed query execution).
- `initial_query_id` (`String`) — ID of the initial query (for distributed query execution).
- `initial_address` (`IPv6`) — IP address that the parent query was launched from.
- `initial_port` (`UInt16`) — The client port that was used to make the parent query.
- `interface` (`UInt8`) — Interface that the query was initiated from. Possible values:
  - 1 — TCP.
  - 2 — HTTP.
- `os_user` (`String`) — OS's username who runs `clickhouse-client`.

- `client_hostname` (`String`) — Hostname of the client machine where the `clickhouse-client` or another TCP client is run.
- `client_name` (`String`) — The `clickhouse-client` or another TCP client name.
- `client_revision` (`UInt32`) — Revision of the `clickhouse-client` or another TCP client.
- `client_version_major` (`UInt32`) — Major version of the `clickhouse-client` or another TCP client.
- `client_version_minor` (`UInt32`) — Minor version of the `clickhouse-client` or another TCP client.
- `client_version_patch` (`UInt32`) — Patch component of the `clickhouse-client` or another TCP client version.
- `http_method` (`UInt8`) — HTTP method that initiated the query. Possible values:
  - 0 — The query was launched from the TCP interface.
  - 1 — `GET` method was used.
  - 2 — `POST` method was used.
- `http_user_agent` (`String`) — The `UserAgent` header passed in the HTTP request.
- `quota_key` (`String`) — The “quota key” specified in the `quotas` setting (see `keyed`).
- `revision` (`UInt32`) — ClickHouse revision.
- `ProfileEvents` (`Map(String, UInt64)`) — ProfileEvents that measure different metrics for this thread. The description of them could be found in the table `system.events`.

## Example

```
SELECT * FROM system.query_thread_log LIMIT 1 \G
```

Row 1:

```
event_date:          2020-09-11
event_time:          2020-09-11 10:08:17
event_time_microseconds: 2020-09-11 10:08:17.134042
query_start_time:    2020-09-11 10:08:17
query_start_time_microseconds: 2020-09-11 10:08:17.063150
query_duration_ms:   70
read_rows:           0
read_bytes:          0
written_rows:        1
written_bytes:       12
memory_usage:        4300844
peak_memory_usage:   4300844
thread_name:         TCPHandler
thread_id:           638133
master_thread_id:    638133
query:               INSERT INTO test1 VALUES
is_initial_query:   1
user:                default
query_id:            50a320fd-85a8-49b8-8761-98a86bcbacef
address:             ::ffff:127.0.0.1
port:                33452
initial_user:        default
initial_query_id:   50a320fd-85a8-49b8-8761-98a86bcbacef
initial_address:    ::ffff:127.0.0.1
initial_port:        33452
interface:          1
os_user:             bharatnc
client_hostname:    tower
client_name:         ClickHouse
client_revision:    54437
client_version_major: 20
client_version_minor: 7
client_version_patch: 2
http_method:         0
http_user_agent:
quota_key:
revision:            54440
ProfileEvents:
{'Query':1,'SelectQuery':1,'ReadCompressedBytes':36,'CompressedReadBufferBlocks':1,'CompressedReadBufferBytes':10,'IOBufferAllocs':1,'IOBufferAllocBytes':89,'ContextLock':15,'RWLockAcquiredReadLocks':1}
```

## See Also

- [system.query\\_log](#) — Description of the `query_log` system table which contains common information about queries execution.
- [system.query\\_views\\_log](#) — This table contains information about each view executed during a query.

## system.query\_views\_log

Contains information about the dependent views executed when running a query, for example, the view type or the execution time.

To start logging:

1. Configure parameters in the `query_views_log` section.
2. Set `log_query_views` to 1.

The flushing period of data is set in `flush_interval_milliseconds` parameter of the `query_views_log` server settings section. To force flushing, use the [SYSTEM FLUSH LOGS](#) query.

ClickHouse does not delete data from the table automatically. See [Introduction](#) for more details.

You can use the `log_queries_probability` setting to reduce the number of queries, registered in the `query_views_log` table.

Columns:

- `event_date` (`Date`) — The date when the last event of the view happened.
- `event_time` (`DateTime`) — The date and time when the view finished execution.
- `event_time_microseconds` (`DateTime`) — The date and time when the view finished execution with microseconds precision.
- `view_duration_ms` (`UInt64`) — Duration of view execution (sum of its stages) in milliseconds.
- `initial_query_id` (`String`) — ID of the initial query (for distributed query execution).
- `view_name` (`String`) — Name of the view.
- `view_uuid` (`UUID`) — UUID of the view.
- `view_type` (`Enum8`) — Type of the view. Values:
  - `'Default'` = 1 — **Default views**. Should not appear in this log.
  - `'Materialized'` = 2 — **Materialized views**.
  - `'Live'` = 3 — **Live views**.
- `view_query` (`String`) — The query executed by the view.
- `view_target` (`String`) — The name of the view target table.
- `read_rows` (`UInt64`) — Number of read rows.
- `read_bytes` (`UInt64`) — Number of read bytes.
- `written_rows` (`UInt64`) — Number of written rows.
- `written_bytes` (`UInt64`) — Number of written bytes.
- `peak_memory_usage` (`Int64`) — The maximum difference between the amount of allocated and freed memory in context of this view.
- `ProfileEvents` (`Map(String, UInt64)`) — ProfileEvents that measure different metrics. The description of them could be found in the table [system.events](#).
- `status` (`Enum8`) — Status of the view. Values:
  - `'QueryStart'` = 1 — Successful start the view execution. Should not appear.
  - `'QueryFinish'` = 2 — Successful end of the view execution.
  - `'ExceptionBeforeStart'` = 3 — Exception before the start of the view execution.
  - `'ExceptionWhileProcessing'` = 4 — Exception during the view execution.
- `exception_code` (`Int32`) — Code of an exception.
- `exception` (`String`) — Exception message.
- `stack_trace` (`String`) — **Stack trace**. An empty string, if the query was completed successfully.

## Example

Query:

```
SELECT * FROM system.query_views_log LIMIT 1 \G;
```

Result:

```
Row 1:  
event_date: 2021-06-22  
event_time: 2021-06-22 13:23:07  
event_time_microseconds: 2021-06-22 13:23:07.738221  
view_duration_ms: 0  
initial_query_id: c3a1ac02-9cad-479b-af54-9e9c0a7afd70  
view_name: default.matview_inner  
view_uuid: 00000000-0000-0000-0000-000000000000  
view_type: Materialized  
view_query: SELECT * FROM default.table_b  
view_target: default.`.inner.matview_inner`  
read_rows: 4  
read_bytes: 64  
written_rows: 2  
written_bytes: 32  
peak_memory_usage: 4196188  
ProfileEvents:  
{'FileOpen':2,'WriteBufferFromFileDescriptorWrite':2,'WriteBufferFromFileDescriptorWriteBytes':187,'IOBufferAllocs':3  
'IOBufferAllocBytes':3145773,'FunctionExecute':3,'DiskWriteElapsedMicroseconds':13,'InsertedRows':2,'InsertedBytes  
'SoftPageFaults':4,'OSReadChars':463}  
status: QueryFinish  
exception_code: 0  
exception:  
stack_trace:
```

## See Also

- [system.query\\_log](#) — Description of the `query_log` system table which contains common information about queries execution.
- [system.query\\_thread\\_log](#) — This table contains information about each query execution thread.

## system.quota\_limits

Contains information about maximums for all intervals of all quotas. Any number of rows or zero can correspond to one quota.

Columns:

- `quota_name` (`String`) — Quota name.
- `duration` (`UInt32`) — Length of the time interval for calculating resource consumption, in seconds.
- `is_randomized_interval` (`UInt8`) — Logical value. It shows whether the interval is randomized. Interval always starts at the same time if it is not randomized. For example, an interval of 1 minute always starts at an integer number of minutes (i.e. it can start at 11:20:00, but it never starts at 11:20:01), an interval of one day always starts at midnight UTC. If interval is randomized, the very first interval starts at random time, and subsequent intervals starts one by one. Values:
  - `0` — Interval is not randomized.
  - `1` — Interval is randomized.
- `max_queries` (`Nullable(UInt64)`) — Maximum number of queries.
- `max_query_selects` (`Nullable(UInt64)`) — Maximum number of select queries.
- `max_query_inserts` (`Nullable(UInt64)`) — Maximum number of insert queries.
- `max_errors` (`Nullable(UInt64)`) — Maximum number of errors.
- `max_result_rows` (`Nullable(UInt64)`) — Maximum number of result rows.
- `max_result_bytes` (`Nullable(UInt64)`) — Maximum number of RAM volume in bytes used to store a queries result.
- `max_read_rows` (`Nullable(UInt64)`) — Maximum number of rows read from all tables and table functions participated in queries.

- `max_read_bytes` (`Nullable(UInt64)`) — Maximum number of bytes read from all tables and table functions participated in queries.
  - `max_execution_time` (`Nullable(Float64)`) — Maximum of the query execution time, in seconds.
- 

## system.quota\_usage

Quota usage by the current user: how much is used and how much is left.

Columns:

- `quota_name` (`String`) — Quota name.
- `quota_key` (`String`) — Key value. For example, if keys = [ip address], `quota_key` may have a value '192.168.1.1'.
- `start_time` (`Nullable(DateTime)`) — Start time for calculating resource consumption.
- `end_time` (`Nullable(DateTime)`) — End time for calculating resource consumption.
- `duration` (`Nullable(UInt64)`) — Length of the time interval for calculating resource consumption, in seconds.
- `queries` (`Nullable(UInt64)`) — The total number of requests on this interval.
- `query_selects` (`Nullable(UInt64)`) — The total number of select requests on this interval.
- `query_inserts` (`Nullable(UInt64)`) — The total number of insert requests on this interval.
- `max_queries` (`Nullable(UInt64)`) — Maximum number of requests.
- `errors` (`Nullable(UInt64)`) — The number of queries that threw an exception.
- `max_errors` (`Nullable(UInt64)`) — Maximum number of errors.
- `result_rows` (`Nullable(UInt64)`) — The total number of rows given as a result.
- `max_result_rows` (`Nullable(UInt64)`) — Maximum number of result rows.
- `result_bytes` (`Nullable(UInt64)`) — RAM volume in bytes used to store a queries result.
- `max_result_bytes` (`Nullable(UInt64)`) — Maximum RAM volume used to store a queries result, in bytes.
- `read_rows` (`Nullable(UInt64)`) — The total number of source rows read from tables for running the query on all remote servers.
- `max_read_rows` (`Nullable(UInt64)`) — Maximum number of rows read from all tables and table functions participated in queries.
- `read_bytes` (`Nullable(UInt64)`) — The total number of bytes read from all tables and table functions participated in queries.
- `max_read_bytes` (`Nullable(UInt64)`) — Maximum of bytes read from all tables and table functions.
- `execution_time` (`Nullable(Float64)`) — The total query execution time, in seconds (wall time).
- `max_execution_time` (`Nullable(Float64)`) — Maximum of query execution time.

## See Also

- [SHOW QUOTA](#)
- 

## system.quotas

Contains information about [quotas](#).

Columns:

- `name` (**String**) — Quota name.
- `id` (**UUID**) — Quota ID.
- `storage`(**String**) — Storage of quotas. Possible value: “users.xml” if a quota configured in the users.xml file, “disk” if a quota configured by an SQL-query.
- `keys` (**Array(Enum8)**) — Key specifies how the quota should be shared. If two connections use the same quota and key, they share the same amounts of resources. Values:
  - `[]` — All users share the same quota.
  - `['user_name']` — Connections with the same user name share the same quota.
  - `['ip_address']` — Connections from the same IP share the same quota.
  - `['client_key']` — Connections with the same key share the same quota. A key must be explicitly provided by a client. When using `clickhouse-client`, pass a key value in the `--quota_key` parameter, or use the `quota_key` parameter in the client configuration file. When using HTTP interface, use the `X-ClickHouse-Quota` header.
  - `['user_name', 'client_key']` — Connections with the same `client_key` share the same quota. If a key isn’t provided by a client, the quota is tracked for `user_name`.
  - `['client_key', 'ip_address']` — Connections with the same `client_key` share the same quota. If a key isn’t provided by a client, the quota is tracked for `ip_address`.
- `durations` (**Array(UInt64)**) — Time interval lengths in seconds.
- `apply_to_all` (**UInt8**) — Logical value. It shows which users the quota is applied to. Values:
  - `0` — The quota applies to users specified in the `apply_to_list`.
  - `1` — The quota applies to all users except those listed in `apply_to_except`.
- `apply_to_list` (**Array(String)**) — List of user names/**roles** that the quota should be applied to.
- `apply_to_except` (**Array(String)**) — List of user names/**roles** that the quota should not apply to.

## See Also

- [SHOW QUOTAS](#)

## system.quotas\_usage

Quota usage by all users.

Columns:

- quota\_name ([String](#)) — Quota name.
- quota\_key ([String](#)) — Key value.
- is\_current ([UInt8](#)) — Quota usage for current user.
- start\_time ([Nullable\(DateTime\)](#)) — Start time for calculating resource consumption.
- end\_time ([Nullable\(DateTime\)](#)) — End time for calculating resource consumption.
- duration ([Nullable\(UInt32\)](#)) — Length of the time interval for calculating resource consumption, in seconds.
- queries ([Nullable\(UInt64\)](#)) — The total number of requests in this interval.
- max\_queries ([Nullable\(UInt64\)](#)) — Maximum number of requests.
- query\_selects ([Nullable\(UInt64\)](#)) — The total number of select requests in this interval.
- max\_query\_selects ([Nullable\(UInt64\)](#)) — Maximum number of select requests.
- query\_inserts ([Nullable\(UInt64\)](#)) — The total number of insert requests in this interval.
- max\_query\_inserts ([Nullable\(UInt64\)](#)) — Maximum number of insert requests.
- errors ([Nullable\(UInt64\)](#)) — The number of queries that threw an exception.
- max\_errors ([Nullable\(UInt64\)](#)) — Maximum number of errors.
- result\_rows ([Nullable\(UInt64\)](#)) — The total number of rows given as a result.
- max\_result\_rows ([Nullable\(UInt64\)](#)) — Maximum of source rows read from tables.
- result\_bytes ([Nullable\(UInt64\)](#)) — RAM volume in bytes used to store a queries result.
- max\_result\_bytes ([Nullable\(UInt64\)](#)) — Maximum RAM volume used to store a queries result, in bytes.
- read\_rows ([Nullable\(UInt64\)](#)) — The total number of source rows read from tables for running the query on all remote servers.
- max\_read\_rows ([Nullable\(UInt64\)](#)) — Maximum number of rows read from all tables and table functions participated in queries.
- read\_bytes ([Nullable\(UInt64\)](#)) — The total number of bytes read from all tables and table functions participated in queries.
- max\_read\_bytes ([Nullable\(UInt64\)](#)) — Maximum of bytes read from all tables and table functions.
- execution\_time ([Nullable\(Float64\)](#)) — The total query execution time, in seconds (wall time).
- max\_execution\_time ([Nullable\(Float64\)](#)) — Maximum of query execution time.

## See Also

- [SHOW QUOTA](#)

## system.replicas

Contains information and status for replicated tables residing on the local server.

This table can be used for monitoring. The table contains a row for every Replicated\* table.

Example:

```
SELECT *
FROM system.replicas
WHERE table = 'test_table'
FORMAT Vertical
```

Query id: dc6dcacb-dc28-4df9-ae27-4354f5b3b13e

Row 1:

```
database:          db
table:            test_table
engine:           ReplicatedMergeTree
is_leader:        1
can_become_leader: 1
is_readonly:      0
is_session_expired: 0
future_parts:    0
parts_to_check:  0
zookeeper_path: /test/test_table
replica_name:    r1
replica_path:   /test/test_table/replicas/r1
columns_version: -1
queue_size:      27
inserts_in_queue: 27
merges_in_queue:  0
part_mutations_in_queue: 0
queue_oldest_time: 2021-10-12 14:48:48
inserts_oldest_time: 2021-10-12 14:48:48
merges_oldest_time: 1970-01-01 03:00:00
part_mutations_oldest_time: 1970-01-01 03:00:00
oldest_part_to_get: 1_17_17_0
oldest_part_to_merge_to:
oldest_part_to_mutate_to:
log_max_index:   206
log_pointer:     207
last_queue_update: 2021-10-12 14:50:08
absolute_delay:  99
total_replicas:  5
active_replicas: 5
last_queue_update_exception:
zookeeper_exception:
replica_is_active: {'r1':1,'r2':1}
```

Columns:

- **database** (String) - Database name
- **table** (String) - Table name
- **engine** (String) - Table engine name
- **is\_leader** (UInt8) - Whether the replica is the leader.  
Multiple replicas can be leaders at the same time. A replica can be prevented from becoming a leader using the `merge_tree` setting `replicated_can_become_leader`. The leaders are responsible for scheduling background merges.  
Note that writes can be performed to any replica that is available and has a session in ZK, regardless of whether it is a leader.
- **can\_become\_leader** (UInt8) - Whether the replica can be a leader.
- **is\_READONLY** (UInt8) - Whether the replica is in read-only mode.  
This mode is turned on if the config does not have sections with ZooKeeper, if an unknown error occurred when reinitializing sessions in ZooKeeper, and during session reinitialization in ZooKeeper.
- **is\_SESSION\_EXPIRED** (UInt8) - the session with ZooKeeper has expired. Basically the same as `is_READONLY`.
- **future\_parts** (UInt32) - The number of data parts that will appear as the result of INSERTs or merges that haven't been done yet.
- **parts\_to\_check** (UInt32) - The number of data parts in the queue for verification. A part is put in the verification queue if there is suspicion that it might be damaged.

- `zookeeper_path` (String) - Path to table data in ZooKeeper.
- `replica_name` (String) - Replica name in ZooKeeper. Different replicas of the same table have different names.
- `replica_path` (String) - Path to replica data in ZooKeeper. The same as concatenating '`zookeeper_path/replicas/replica_path`'.
- `columns_version` (Int32) - Version number of the table structure. Indicates how many times ALTER was performed. If replicas have different versions, it means some replicas haven't made all of the ALTERs yet.
- `queue_size` (UInt32) - Size of the queue for operations waiting to be performed. Operations include inserting blocks of data, merges, and certain other actions. It usually coincides with `future_parts`.
- `inserts_in_queue` (UInt32) - Number of inserts of blocks of data that need to be made. Insertions are usually replicated fairly quickly. If this number is large, it means something is wrong.
- `merges_in_queue` (UInt32) - The number of merges waiting to be made. Sometimes merges are lengthy, so this value may be greater than zero for a long time.
- `part_mutations_in_queue` (UInt32) - The number of mutations waiting to be made.
- `queue_oldest_time` (DateTime) - If `queue_size` greater than 0, shows when the oldest operation was added to the queue.
- `inserts_oldest_time` (DateTime) - See `queue_oldest_time`
- `merges_oldest_time` (DateTime) - See `queue_oldest_time`
- `part_mutations_oldest_time` (DateTime) - See `queue_oldest_time`

The next 4 columns have a non-zero value only where there is an active session with ZK.

- `log_max_index` (UInt64) - Maximum entry number in the log of general activity.
- `log_pointer` (UInt64) - Maximum entry number in the log of general activity that the replica copied to its execution queue, plus one. If `log_pointer` is much smaller than `log_max_index`, something is wrong.
- `last_queue_update` (DateTime) - When the queue was updated last time.
- `absolute_delay` (UInt64) - How big lag in seconds the current replica has.
- `total_replicas` (UInt8) - The total number of known replicas of this table.
- `active_replicas` (UInt8) - The number of replicas of this table that have a session in ZooKeeper (i.e., the number of functioning replicas).
- `last_queue_update_exception` (String) - When the queue contains broken entries. Especially important when ClickHouse breaks backward compatibility between versions and log entries written by newer versions aren't parseable by old versions.
- `zookeeper_exception` (String) - The last exception message, got if the error happened when fetching the info from ZooKeeper.
- `replica_is_active` (Map(String, UInt8)) — Map between replica name and is replica active.

If you request all the columns, the table may work a bit slowly, since several reads from ZooKeeper are made for each row.

If you do not request the last 4 columns (`log_max_index`, `log_pointer`, `total_replicas`, `active_replicas`), the table works quickly.

For example, you can check that everything is working correctly like this:

```
SELECT
    database,
    table,
    is_leader,
    is_READONLY,
    is_SESSION_EXPIRED,
    future_parts,
    parts_to_check,
    columns_version,
    queue_size,
    inserts_in_queue,
    merges_in_queue,
    log_max_index,
    log_pointer,
    total_replicas,
    active_replicas
FROM system.replicas
WHERE
    is_READONLY
    OR is_SESSION_EXPIRED
    OR future_parts > 20
    OR parts_to_check > 10
    OR queue_size > 20
    OR inserts_in_queue > 10
    OR log_max_index - log_pointer > 10
    OR total_replicas < 2
    OR active_replicas < total_replicas
```

If this query does not return anything, it means that everything is fine.

## system.replicated\_fetches

Contains information about currently running background fetches.

Columns:

- `database` (`String`) — Name of the database.
- `table` (`String`) — Name of the table.
- `elapsed` (`Float64`) — The time elapsed (in seconds) since showing currently running background fetches started.
- `progress` (`Float64`) — The percentage of completed work from 0 to 1.
- `result_part_name` (`String`) — The name of the part that will be formed as the result of showing currently running background fetches.
- `result_part_path` (`String`) — Absolute path to the part that will be formed as the result of showing currently running background fetches.
- `partition_id` (`String`) — ID of the partition.
- `total_size_bytes_compressed` (`UInt64`) — The total size (in bytes) of the compressed data in the result part.
- `bytes_read_compressed` (`UInt64`) — The number of compressed bytes read from the result part.
- `source_replica_path` (`String`) — Absolute path to the source replica.
- `source_replica_hostname` (`String`) — Hostname of the source replica.
- `source_replica_port` (`UInt16`) — Port number of the source replica.

- `interserver_scheme` ([String](#)) — Name of the interserver scheme.
- `URI` ([String](#)) — Uniform resource identifier.
- `to_detached` ([UInt8](#)) — The flag indicates whether the currently running background fetch is being performed using the `TO DETACHED` expression.
- `thread_id` ([UInt64](#)) — Thread identifier.

## Example

```
SELECT * FROM system.replicated_fetches LIMIT 1 FORMAT Vertical;
```

Row 1:

```
database:          default
table:             t
elapsed:           7.243039876
progress:          0.41832135995612835
result_part_name: all_0_0_0
result_part_path: /var/lib/clickhouse/store/700/70080a04-b2de-4adf-9fa5-9ea210e81766/all_0_0_0/
partition_id:      all
total_size_bytes_compressed: 1052783726
bytes_read_compressed: 440401920
source_replica_path: /clickhouse/test/t/replicas/1
source_replica_hostname: node1
source_replica_port: 9009
interserver_scheme: http
URI:               http://node1:9009/?endpoint=DataPartsExchange%3A%2Fclickhouse%2Ftest%2Ft%2Freplicas%2F1&part=all_0_0_0&client_protocol_version=4&compress=false
to_detached:       0
thread_id:         54
```

## See Also

- [Managing ReplicatedMergeTree Tables](#)

# system.replication\_queue

Contains information about tasks from replication queues stored in ZooKeeper for tables in the `ReplicatedMergeTree` family.

Columns:

- `database` ([String](#)) — Name of the database.
- `table` ([String](#)) — Name of the table.
- `replica_name` ([String](#)) — Replica name in ZooKeeper. Different replicas of the same table have different names.
- `position` ([UInt32](#)) — Position of the task in the queue.
- `node_name` ([String](#)) — Node name in ZooKeeper.

- `type` (**String**) — Type of the task in the queue, one of:
  - `GET_PART` — Get the part from another replica.
  - `ATTACH_PART` — Attach the part, possibly from our own replica (if found in the `detached` folder). You may think of it as a `GET_PART` with some optimizations as they're nearly identical.
  - `MERGE_PARTS` — Merge the parts.
  - `DROP_RANGE` — Delete the parts in the specified partition in the specified number range.
  - `CLEAR_COLUMN` — NOTE: Deprecated. Drop specific column from specified partition.
  - `CLEAR_INDEX` — NOTE: Deprecated. Drop specific index from specified partition.
  - `REPLACE_RANGE` — Drop a certain range of parts and replace them with new ones.
  - `MUTATE_PART` — Apply one or several mutations to the part.
  - `ALTER_METADATA` — Apply alter modification according to global `/metadata` and `/columns` paths.
- `create_time` (**Datetime**) — Date and time when the task was submitted for execution.
- `required_quorum` (**UInt32**) — The number of replicas waiting for the task to complete with confirmation of completion. This column is only relevant for the `GET_PARTS` task.
- `source_replica` (**String**) — Name of the source replica.
- `new_part_name` (**String**) — Name of the new part.
- `parts_to_merge` (**Array (String)**) — Names of parts to merge or update.
- `is_detach` (**UInt8**) — The flag indicates whether the `DETACH_PARTS` task is in the queue.
- `is_currently_executing` (**UInt8**) — The flag indicates whether a specific task is being performed right now.
- `num_tries` (**UInt32**) — The number of failed attempts to complete the task.
- `last_exception` (**String**) — Text message about the last error that occurred (if any).
- `last_attempt_time` (**Datetime**) — Date and time when the task was last attempted.
- `num_postponed` (**UInt32**) — The number of postponed tasks.
- `postpone_reason` (**String**) — The reason why the task was postponed.
- `last_postpone_time` (**Datetime**) — Date and time when the task was last postponed.
- `merge_type` (**String**) — Type of the current merge. Empty if it's a mutation.

## Example

```
SELECT * FROM system.replication_queue LIMIT 1 FORMAT Vertical;
```

Row 1:

```
database:      merge
table:        visits_v2
replica_name:  mtgiga001-1t.metrika.yandex.net
position:     15
node_name:    queue-0009325559
type:         MERGE_PARTS
create_time:   2020-12-07 14:04:21
required_quorum: 0
source_replica: mtgiga001-1t.metrika.yandex.net
new_part_name:  20201130_121373_121384_2
parts_to_merge: ['20201130_121373_121378_1','20201130_121379_121379_0','20201130_121380_121380_0','20201130_121381_121381_0','20201130_121382_121382_0','20201130_121383_121383_0','20201130_121384_121384_0']
is_detach:    0
is_currently_executing: 0
num_tries:    36
last_exception: Code: 226, e.displayText() = DB::Exception: Marks file '/opt/clickhouse/data/merge/visits_v2/tmp_fetch_20201130_121373_121384_2/CounterID.mrk' does not exist (version 20.8.7.15 (official build))
last_attempt_time: 2020-12-08 17:35:54
num_postponed: 0
postpone_reason:
last_postpone_time: 1970-01-01 03:00:00
```

## See Also

- [Managing ReplicatedMergeTree Tables](#)

## system.role\_grants

Contains the role grants for users and roles. To add entries to this table, use `GRANT role TO user`.

Columns:

- `user_name` (`Nullable(String)`) — User name.
- `role_name` (`Nullable(String)`) — Role name.
- `granted_role_name` (`String`) — Name of role granted to the `role_name` role. To grant one role to another one use `GRANT role1 TO role2`.
- `granted_role_is_default` (`UInt8`) — Flag that shows whether `granted_role` is a default role. Possible values:
  - 1 — `granted_role` is a default role.
  - 0 — `granted_role` is not a default role.
- `with_admin_option` (`UInt8`) — Flag that shows whether `granted_role` is a role with `ADMIN OPTION` privilege. Possible values:
  - 1 — The role has `ADMIN OPTION` privilege.
  - 0 — The role without `ADMIN OPTION` privilege.

## system.roles

Contains information about configured [roles](#).

Columns:

- `name` (`String`) — Role name.

- `id` ([UUID](#)) — Role ID.
- `storage` ([String](#)) — Path to the storage of roles. Configured in the `access_control_path` parameter.

## See Also

- [SHOW ROLES](#)

## system.row\_policies

Contains filters for one particular table, as well as a list of roles and/or users which should use this row policy.

Columns:

- `name` ([String](#)) — Name of a row policy.
- `short_name` ([String](#)) — Short name of a row policy. Names of row policies are compound, for example: myfilter ON mydb.mytable. Here "myfilter ON mydb.mytable" is the name of the row policy, "myfilter" is its short name.
- `database` ([String](#)) — Database name.
- `table` ([String](#)) — Table name.
- `id` ([UUID](#)) — Row policy ID.
- `storage` ([String](#)) — Name of the directory where the row policy is stored.
- `select_filter` ([Nullable\(String\)](#)) — Condition which is used to filter rows.
- `is_restrictive` ([UInt8](#)) — Shows whether the row policy restricts access to rows, see [CREATE ROW POLICY](#). Value:
  - `0` — The row policy is defined with `AS PERMISSIVE` clause.
  - `1` — The row policy is defined with `AS RESTRICTIVE` clause.
- `apply_to_all` ([UInt8](#)) — Shows that the row policies set for all roles and/or users.
- `apply_to_list` ([Array\(String\)](#)) — List of the roles and/or users to which the row policies is applied.
- `apply_to_except` ([Array\(String\)](#)) — The row policies is applied to all roles and/or users excepting of the listed ones.

## See Also

- [SHOW POLICIES](#)

## system.settings

Contains information about session settings for current user.

Columns:

- `name` ([String](#)) — Setting name.
- `value` ([String](#)) — Setting value.
- `changed` ([UInt8](#)) — Shows whether a setting is changed from its default value.

- `description` (`String`) — Short setting description.
- `min` (`Nullable(String)`) — Minimum value of the setting, if any is set via `constraints`. If the setting has no minimum value, contains `NULL`.
- `max` (`Nullable(String)`) — Maximum value of the setting, if any is set via `constraints`. If the setting has no maximum value, contains `NULL`.
- `readonly` (`UInt8`) — Shows whether the current user can change the setting:
  - `0` — Current user can change the setting.
  - `1` — Current user can't change the setting.

## Example

The following example shows how to get information about settings which name contains `min_i`.

```
SELECT *
FROM system.settings
WHERE name LIKE '%min_i%'
```

name	value	changed	description
min	max	readonly	
<code>min_insert_block_size_rows</code> in rows, if blocks are not big enough.	<code>1048576</code>	<code>0</code>	Squash blocks passed to INSERT query to specified size <code>  NULL   NULL   0  </code>
<code>min_insert_block_size_bytes</code> in bytes, if blocks are not big enough.	<code>268435456</code>	<code>0</code>	Squash blocks passed to INSERT query to specified size <code>  NULL   NULL   0  </code>
<code>read_backoff_min_interval_between_events_ms</code>	<code>1000</code>	<code>0</code>	Settings to reduce the number of threads in case of slow reads. Do not pay attention to the event, if the previous one has passed less than a certain amount of time. <code>  NULL   NULL   0  </code>

Using of `WHERE changed` can be useful, for example, when you want to check:

- Whether settings in configuration files are loaded correctly and are in use.
- Settings that changed in the current session.

```
SELECT * FROM system.settings WHERE changed AND name='load_balancing'
```

## See also

- [Settings](#)
- [Permissions for Queries](#)
- [Constraints on Settings](#)
- [SHOW SETTINGS statement](#)

## system.settings\_profile\_elements

Describes the content of the settings profile:

- Constraints.
- Roles and users that the setting applies to.

- Parent settings profiles.

Columns:

- `profile_name` (`Nullable(String)`) — Setting profile name.
- `user_name` (`Nullable(String)`) — User name.
- `role_name` (`Nullable(String)`) — Role name.
- `index` (`UInt64`) — Sequential number of the settings profile element.
- `setting_name` (`Nullable(String)`) — Setting name.
- `value` (`Nullable(String)`) — Setting value.
- `min` (`Nullable(String)`) — The minimum value of the setting. `NULL` if not set.
- `max` (`Nullable(String)`) — The maximum value of the setting. `NULL` if not set.
- `readonly` (`Nullable(UInt8)`) — Profile that allows only read queries.
- `inherit_profile` (`Nullable(String)`) — A parent profile for this setting profile. `NULL` if not set. Setting profile will inherit all the settings' values and constraints (`min`, `max`, `readonly`) from its parent profiles.

## system.settings\_profiles

Contains properties of configured setting profiles.

Columns:

- `name` (`String`) — Setting profile name.
- `id` (`UUID`) — Setting profile ID.
- `storage` (`String`) — Path to the storage of setting profiles. Configured in the `access_control_path` parameter.
- `num_elements` (`UInt64`) — Number of elements for this profile in the `system.settings_profile_elements` table.
- `apply_to_all` (`UInt8`) — Shows that the settings profile set for all roles and/or users.
- `apply_to_list` (`Array(String)`) — List of the roles and/or users to which the setting profile is applied.
- `apply_to_except` (`Array(String)`) — The setting profile is applied to all roles and/or users excepting of the listed ones.

## See Also

- [SHOW PROFILES](#)

## system.stack\_trace

Contains stack traces of all server threads. Allows developers to introspect the server state.

To analyze stack frames, use the `addressToLine`, `addressToSymbol` and `demangle` [introspection functions](#).

Columns:

- `thread_name` (`String`) — Thread name.
- `thread_id` (`UInt64`) — Thread identifier.

- `query_id` (**String**) — Query identifier that can be used to get details about a query that was running from the `query_log` system table.
- `trace` (**Array(UInt64)**) — A **stack trace** which represents a list of physical addresses where the called methods are stored.

## Example

Enabling introspection functions:

```
SET allow_introspection_functions = 1;
```

Getting symbols from ClickHouse object files:

```
WITH arrayMap(x -> demangle(addressToSymbol(x)), trace) AS all SELECT thread_name, thread_id, query_id,
arrayStringConcat(all, '\n') AS res FROM system.stack_trace LIMIT 1 \G;
```

Row 1:

```
thread_name: clickhouse-serv

thread_id: 686
query_id: 1a11f70b-626d-47c1-b948-f9c7b206395d
res:    sigqueue
DB::StorageSystemStackTrace::fillData(std::__1::vector<COW<DB::IColumn>::mutable_ptr<DB::IColumn>,
std::__1::allocator<COW<DB::IColumn>::mutable_ptr<DB::IColumn> >>, DB::Context const&, DB::SelectQueryInfo const&) const
DB::IStorageSystemOneBlock<DB::StorageSystemStackTrace>::read(std::__1::vector<std::__1::basic_string<char>,
std::__1::char_traits<char>, std::__1::allocator<char> >, std::__1::allocator<std::__1::basic_string<char>,
std::__1::char_traits<char>, std::__1::allocator<char> > > const&, DB::SelectQueryInfo const&, DB::Context const&, DB::QueryProcessingStage::Enum, unsigned long, unsigned int)
DB::InterpreterSelectQuery::executeFetchColumns(DB::QueryProcessingStage::Enum, DB::QueryPipeline&,
std::__1::shared_ptr<DB::PrewhereInfo> const&, std::__1::vector<std::__1::basic_string<char>,
std::__1::char_traits<char>, std::__1::allocator<char> >, std::__1::allocator<std::__1::basic_string<char>,
std::__1::char_traits<char>, std::__1::allocator<char> > > const&)
DB::InterpreterSelectQuery::executelImpl(DB::QueryPipeline&, std::__1::shared_ptr<DB::IBlockInputStream> const&,
std::__1::optional<DB::Pipe>)
DB::InterpreterSelectQuery::execute()
DB::InterpreterSelectQueryWithUnionQuery::execute()
DB::executeQueryImpl(char const*, char const*, DB::Context&, bool, DB::QueryProcessingStage::Enum, bool,
DB::ReadBuffer*)
DB::executeQuery(std::__1::basic_string<char>, std::__1::char_traits<char>, std::__1::allocator<char> > const&,
DB::Context&, bool, DB::QueryProcessingStage::Enum, bool)
DB::TCPHandler::runImpl()
DB::TCPHandler::run()
Poco::Net::TCPServerConnection::start()
Poco::Net::TCPServerDispatcher::run()
Poco::PooledThread::run()
Poco::ThreadImpl::runnableEntry(void*)
start_thread
__clone
```

Getting filenames and line numbers in ClickHouse source code:

```
WITH arrayMap(x -> addressToLine(x), trace) AS all, arrayFilter(x -> x LIKE '%/dbms/%', all) AS dbms SELECT
thread_name, thread_id, query_id, arrayStringConcat(notEmpty(dbms) ? dbms : all, '\n') AS res FROM
system.stack_trace LIMIT 1 \G;
```

Row 1:

```
thread_name: clickhouse-serv

thread_id: 686
query_id: cad353e7-1c29-4b2e-949f-93e597ab7a54
res: /lib/x86_64-linux-gnu/libc-2.27.so
/build/obj-x86_64-linux-gnu/../src/Storages/System/StorageSystemStackTrace.cpp:182
/build/obj-x86_64-linux-gnu/../contrib/libcxx/include/vector:656
/build/obj-x86_64-linux-gnu/../src/Interpreters/InterpreterSelectQuery.cpp:1338
/build/obj-x86_64-linux-gnu/../src/Interpreters/InterpreterSelectQuery.cpp:751
/build/obj-x86_64-linux-gnu/../contrib/libcxx/include/optional:224
/build/obj-x86_64-linux-gnu/../src/Interpreters/InterpreterSelectWithUnionQuery.cpp:192
/build/obj-x86_64-linux-gnu/../src/Interpreters/executeQuery.cpp:384
/build/obj-x86_64-linux-gnu/../src/Interpreters/executeQuery.cpp:643
/build/obj-x86_64-linux-gnu/../src/Server/TCPHandler.cpp:251
/build/obj-x86_64-linux-gnu/../src/Server/TCPHandler.cpp:1197
/build/obj-x86_64-linux-gnu/../contrib/poco/Net/src/TCPServerConnection.cpp:57
/build/obj-x86_64-linux-gnu/../contrib/libcxx/include/atomic:856
/build/obj-x86_64-linux-gnu/../contrib/poco/Foundation/include/Poco/Mutex_POSIX.h:59
/build/obj-x86_64-linux-gnu/../contrib/poco/Foundation/include/Poco/AutoPtr.h:223
/lib/x86_64-linux-gnu/libpthread-2.27.so
/lib/x86_64-linux-gnu/libc-2.27.so
```

## See Also

- [Introspection Functions](#) — Which introspection functions are available and how to use them.
- [system.trace\\_log](#) — Contains stack traces collected by the sampling query profiler.
- [arrayMap](#) — Description and usage example of the `arrayMap` function.
- [arrayFilter](#) — Description and usage example of the `arrayFilter` function.

## system.storage\_policies

Contains information about storage policies and volumes defined in the [server configuration](#).

Columns:

- `policy_name` ([String](#)) — Name of the storage policy.
- `volume_name` ([String](#)) — Volume name defined in the storage policy.
- `volume_priority` ([UInt64](#)) — Volume order number in the configuration.
- `disks` ([Array\(String\)](#)) — Disk names, defined in the storage policy.
- `max_data_part_size` ([UInt64](#)) — Maximum size of a data part that can be stored on volume disks (0 — no limit).
- `move_factor` ([Float64](#)) — Ratio of free disk space. When the ratio exceeds the value of configuration parameter, ClickHouse start to move data to the next volume in order.
- `prefer_not_to_merge` ([UInt8](#)) — Value of the `prefer_not_to_merge` setting. When this setting is enabled, merging data on this volume is not allowed. This allows controlling how ClickHouse works with slow disks.

If the storage policy contains more than one volume, then information for each volume is stored in the individual row of the table.

## system.table\_engines

Contains description of table engines supported by server and their feature support information.

This table contains the following columns (the column type is shown in brackets):

- `name` (String) — The name of table engine.
- `supports_settings` (UInt8) — Flag that indicates if table engine supports `SETTINGS` clause.
- `supports_skipping_indices` (UInt8) — Flag that indicates if table engine supports **skipping indices**.
- `supports_ttl` (UInt8) — Flag that indicates if table engine supports **TTL**.
- `supports_sort_order` (UInt8) — Flag that indicates if table engine supports clauses `PARTITION_BY`, `PRIMARY_KEY`, `ORDER_BY` and `SAMPLE_BY`.
- `supports_replication` (UInt8) — Flag that indicates if table engine supports **data replication**.
- `supports_deduplication` (UInt8) — Flag that indicates if table engine supports data deduplication.
- `supports_parallel_insert` (UInt8) — Flag that indicates if table engine supports parallel insert (see `max_insert_threads` setting).

Example:

```
SELECT *
FROM system.table_engines
WHERE name in ('Kafka', 'MergeTree', 'ReplicatedCollapsingMergeTree')
```

name	supports_settings	supports_skipping_indices	supports_sort_order	supports_ttl	supports_replication	supports_deduplication	supports_parallel_insert
MergeTree	1	1	0	1	1	1	0
Kafka	0	1	0	0	0	0	0
ReplicatedCollapsingMergeTree	1	1	1	1	1	1	1

## See also

- MergeTree family **query clauses**
- Kafka **settings**
- Join **settings**

## system.tables

Contains metadata of each table that the server knows about.

**Detached tables** are not shown in `system.tables`.

**Temporary tables** are visible in the `system.tables` only in those session where they have been created. They are shown with the empty `database` field and with the `is_temporary` flag switched on.

Columns:

- `database` (String) — The name of the database the table is in.
- `name` (String) — Table name.

- `engine` (`String`) — Table engine name (without parameters).
- `is_temporary` (`UInt8`) - Flag that indicates whether the table is temporary.
- `data_path` (`String`) - Path to the table data in the file system.
- `metadata_path` (`String`) - Path to the table metadata in the file system.
- `metadata_modification_time` (`DateTime`) - Time of latest modification of the table metadata.
- `dependencies_database` (`Array(String)`) - Database dependencies.
- `dependencies_table` (`Array(String)`) - Table dependencies (`MaterializedView` tables based on the current table).
- `create_table_query` (`String`) - The query that was used to create the table.
- `engine_full` (`String`) - Parameters of the table engine.
- `partition_key` (`String`) - The partition key expression specified in the table.
- `sorting_key` (`String`) - The sorting key expression specified in the table.
- `primary_key` (`String`) - The primary key expression specified in the table.
- `sampling_key` (`String`) - The sampling key expression specified in the table.
- `storage_policy` (`String`) - The storage policy:
  - `MergeTree`
  - `Distributed`
- `total_rows` (`Nullable(UInt64)`) - Total number of rows, if it is possible to quickly determine exact number of rows in the table, otherwise `NULL` (including underlying `Buffer` table).
- `total_bytes` (`Nullable(UInt64)`) - Total number of bytes, if it is possible to quickly determine exact number of bytes for the table on storage, otherwise `NULL` (does not include any underlying storage).
  - If the table stores data on disk, returns used space on disk (i.e. compressed).
  - If the table stores data in memory, returns approximated number of used bytes in memory.
- `lifetime_rows` (`Nullable(UInt64)`) - Total number of rows INSERTed since server start (only for `Buffer` tables).
- `lifetime_bytes` (`Nullable(UInt64)`) - Total number of bytes INSERTed since server start (only for `Buffer` tables).
- `comment` (`String`) - The comment for the table.

The `system.tables` table is used in `SHOW TABLES` query implementation.

## Example

```
SELECT * FROM system.tables LIMIT 2 FORMAT Vertical;
```

#### Row 1:

```
database:      base
name:          t1
uuid:          81b1c20a-b7c6-4116-a2ce-7583fb6b6736
engine:        MergeTree
is_temporary:  0
data_paths:    ['/var/lib/clickhouse/store/81b1c20a-b7c6-4116-a2ce-7583fb6b6736/']
metadata_path: /var/lib/clickhouse/store/461/461cf698-fd0b-406d-8c01-5d8fd5748a91/t1.sql
metadata_modification_time: 2021-01-25 19:14:32
dependencies_database: []
dependencies_table: []
create_table_query: CREATE TABLE base.t1 (`n` UInt64) ENGINE = MergeTree ORDER BY n SETTINGS
index_granularity = 8192
engine_full:   MergeTree ORDER BY n SETTINGS index_granularity = 8192
partition_key: n
sorting_key:   n
primary_key:   n
sampling_key: 
storage_policy: default
total_rows:    1
total_bytes:   99
lifetime_rows: NULL
lifetime_bytes: NULL
comment:
```

#### Row 2:

```
database:      default
name:          53r93yleapyears
uuid:          00000000-0000-0000-0000-000000000000
engine:        MergeTree
is_temporary:  0
data_paths:    ['/var/lib/clickhouse/data/default/53r93yleapyears/']
metadata_path: /var/lib/clickhouse/metadata/default/53r93yleapyears.sql
metadata_modification_time: 2020-09-23 09:05:36
dependencies_database: []
dependencies_table: []
create_table_query: CREATE TABLE default.`53r93yleapyears`(`id` Int8, `febdays` Int8) ENGINE = MergeTree
ORDER BY id SETTINGS index_granularity = 8192
engine_full:   MergeTree ORDER BY id SETTINGS index_granularity = 8192
partition_key: id
sorting_key:   id
primary_key:   id
sampling_key: 
storage_policy: default
total_rows:    2
total_bytes:   155
lifetime_rows: NULL
lifetime_bytes: NULL
comment:
```

## system.text\_log

Contains logging entries. The logging level which goes to this table can be limited to the `text_log.level` server setting.

Columns:

- `event_date` (Date) — Date of the entry.
- `event_time` (DateTime) — Time of the entry.
- `event_time_microseconds` (DateTime) — Time of the entry with microseconds precision.
- `microseconds` (UInt32) — Microseconds of the entry.
- `thread_name` (String) — Name of the thread from which the logging was done.

- `thread_id` (UInt64) — OS thread ID.
- `level` (Enum8) — Entry level. Possible values:
  - 1 or 'Fatal'.
  - 2 or 'Critical'.
  - 3 or 'Error'.
  - 4 or 'Warning'.
  - 5 or 'Notice'.
  - 6 or 'Information'.
  - 7 or 'Debug'.
  - 8 or 'Trace'.
- `query_id` (String) — ID of the query.
- `logger_name` (LowCardinality(String)) — Name of the logger (i.e. `DDLWorker`).
- `message` (String) — The message itself.
- `revision` (UInt32) — ClickHouse revision.
- `source_file` (LowCardinality(String)) — Source file from which the logging was done.
- `source_line` (UInt64) — Source line from which the logging was done.

## Example

```
SELECT * FROM system.text_log LIMIT 1 \G
```

Row 1:

<code>event_date:</code>	2020-09-10
<code>event_time:</code>	2020-09-10 11:23:07
<code>event_time_microseconds:</code>	2020-09-10 11:23:07.871397
<code>microseconds:</code>	871397
<code>thread_name:</code>	clickhouse-serv
<code>thread_id:</code>	564917
<code>level:</code>	Information
<code>query_id:</code>	
<code>logger_name:</code>	DNSCacheUpdater
<code>message:</code>	Update period 15 seconds
<code>revision:</code>	54440
<code>source_file:</code>	/ClickHouse/src/Interpreters/DNSCacheUpdater.cpp; void DB::DNSCacheUpdater::start()
<code>source_line:</code>	45

## system.time\_zones

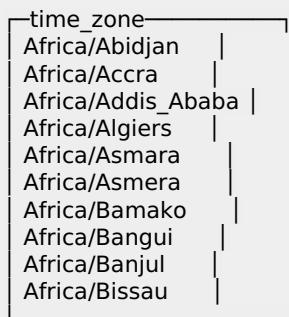
Contains a list of time zones that are supported by the ClickHouse server. This list of timezones might vary depending on the version of ClickHouse.

Columns:

- `time_zone` (String) — List of supported time zones.

## Example

```
SELECT * FROM system.time_zones LIMIT 10
```



## system.trace\_log

Contains stack traces collected by the sampling query profiler.

ClickHouse creates this table when the `trace_log` server configuration section is set. Also the `query_profiler_real_time_period_ns` and `query_profiler_cpu_time_period_ns` settings should be set.

To analyze logs, use the `addressToLine`, `addressToSymbol` and `demangle` introspection functions.

Columns:

- `event_date` (`Date`) — Date of sampling moment.
- `event_time` (`DateTime`) — Timestamp of the sampling moment.
- `event_time_microseconds` (`DateTime64`) — Timestamp of the sampling moment with microseconds precision.
- `timestamp_ns` (`UInt64`) — Timestamp of the sampling moment in nanoseconds.
- `revision` (`UInt32`) — ClickHouse server build revision.

When connecting to the server by `clickhouse-client`, you see the string similar to `Connected to ClickHouse server version 19.18.1 revision 54429..`. This field contains the `revision`, but not the version of a server.

- `trace_type` (`Enum8`) — Trace type:
  - `Real` represents collecting stack traces by wall-clock time.
  - `CPU` represents collecting stack traces by CPU time.
  - `Memory` represents collecting allocations and deallocations when memory allocation exceeds the subsequent watermark.
  - `MemorySample` represents collecting random allocations and deallocations.
- `thread_number` (`UInt32`) — Thread identifier.
- `query_id` (`String`) — Query identifier that can be used to get details about a query that was running from the `query_log` system table.
- `trace` (`Array(UInt64)`) — Stack trace at the moment of sampling. Each element is a virtual memory address inside ClickHouse server process.

### Example

```
SELECT * FROM system.trace_log LIMIT 1 \G
```

Row 1:

```
event_date: 2020-09-10
event_time: 2020-09-10 11:23:09
event_time_microseconds: 2020-09-10 11:23:09.872924
timestamp_ns: 1599762189872924510
revision: 54440
trace_type: Memory
thread_id: 564963
query_id:
trace: [371912858,371912789,371798468,371799717,371801313,371790250,624462773,566365041,566440261,566445834,566460071,566459914,566459842,566459580,566459469,566459389,566459341,566455774,371993941,371988245,372158848,372187428,372187309,372187093,372185478,140222123165193,140222122205443]
size: 5244400
```

## system.users

Contains a list of **user accounts** configured at the server.

Columns:

- `name` (**String**) — User name.
- `id` (**UUID**) — User ID.
- `storage` (**String**) — Path to the storage of users. Configured in the `access_control_path` parameter.
- `auth_type` (**Enum8**('no\_password' = 0,'plaintext\_password' = 1, 'sha256\_password' = 2, 'double\_sha1\_password' = 3)) — Shows the authentication type. There are multiple ways of user identification: with no password, with plain text password, with **SHA256**-encoded password or with **double SHA-1**-encoded password.
- `auth_params` (**String**) — Authentication parameters in the JSON format depending on the `auth_type`.
- `host_ip` (**Array(String)**) — IP addresses of hosts that are allowed to connect to the ClickHouse server.
- `host_names` (**Array(String)**) — Names of hosts that are allowed to connect to the ClickHouse server.
- `host_names_regex` (**Array(String)**) — Regular expression for host names that are allowed to connect to the ClickHouse server.
- `host_names_like` (**Array(String)**) — Names of hosts that are allowed to connect to the ClickHouse server, set using the LIKE predicate.
- `default_roles_all` (**UInt8**) — Shows that all granted roles set for user by default.
- `default_roles_list` (**Array(String)**) — List of granted roles provided by default.
- `default_roles_except` (**Array(String)**) — All the granted roles set as default excepting of the listed ones.

## See Also

- [SHOW USERS](#)

## system.zookeeper

The table does not exist if ZooKeeper is not configured. Allows reading data from the ZooKeeper cluster defined in the config.

The query must either have a 'path =' condition or a path IN condition set with the WHERE clause as shown below. This corresponds to the path of the children in ZooKeeper that you want to get data for.

The query `SELECT * FROM system.zookeeper WHERE path = '/clickhouse'` outputs data for all children on the `/clickhouse` node.

To output data for all root nodes, write path = '/'.

If the path specified in 'path' does not exist, an exception will be thrown.

The query `SELECT * FROM system.zookeeper WHERE path IN ('/', '/clickhouse')` outputs data for all children on the `/` and `/clickhouse` node.

If in the specified 'path' collection has does not exist path, an exception will be thrown.

It can be used to do a batch of ZooKeeper path queries.

Columns:

- `name` (String) — The name of the node.
- `path` (String) — The path to the node.
- `value` (String) — Node value.
- `dataLength` (Int32) — Size of the value.
- `numChildren` (Int32) — Number of descendants.
- `czxid` (Int64) — ID of the transaction that created the node.
- `mzxid` (Int64) — ID of the transaction that last changed the node.
- `pzxid` (Int64) — ID of the transaction that last deleted or added descendants.
- `ctime` (DateTime) — Time of node creation.
- `mtime` (DateTime) — Time of the last modification of the node.
- `version` (Int32) — Node version: the number of times the node was changed.
- `cversion` (Int32) — Number of added or removed descendants.
- `aversion` (Int32) — Number of changes to the ACL.
- `ephemeralOwner` (Int64) — For ephemeral nodes, the ID of the session that owns this node.

Example:

```
SELECT *
FROM system.zookeeper
WHERE path = '/clickhouse/tables/01-08/visits/replicas'
FORMAT Vertical
```

Row 1:

```
name: example01-08-1.yandex.ru
value:
czxid: 932998691229
mzxid: 932998691229
ctime: 2015-03-27 16:49:51
mtime: 2015-03-27 16:49:51
version: 0
cversion: 47
aversion: 0
ephemeralOwner: 0
dataLength: 0
numChildren: 7
pzxid: 987021031383
path: /clickhouse/tables/01-08/visits/replicas
```

Row 2:

```
name: example01-08-2.yandex.ru
value:
czxid: 933002738135
mzxid: 933002738135
ctime: 2015-03-27 16:57:01
mtime: 2015-03-27 16:57:01
version: 0
cversion: 37
aversion: 0
ephemeralOwner: 0
dataLength: 0
numChildren: 7
pzxid: 987021252247
path: /clickhouse/tables/01-08/visits/replicas
```

## system.zookeeper\_log

This table contains information about the parameters of the request to the ZooKeeper server and the response from it.

For requests, only columns with request parameters are filled in, and the remaining columns are filled with default values (0 or `NULL`). When the response arrives, the data from the response is added to the other columns.

Columns with request parameters:

- `type` (**Enum**) — Event type in the ZooKeeper client. Can have one of the following values:
  - `Request` — The request has been sent.
  - `Response` — The response was received.
  - `Finalize` — The connection is lost, no response was received.
- `event_date` (**Date**) — The date when the event happened.
- `event_time` (**DateTime64**) — The date and time when the event happened.
- `address` (**IPv6**) — IP address of ZooKeeper server that was used to make the request.
- `port` (**UInt16**) — The port of ZooKeeper server that was used to make the request.
- `session_id` (**Int64**) — The session ID that the ZooKeeper server sets for each connection.
- `xid` (**Int32**) — The ID of the request within the session. This is usually a sequential request number. It is the same for the request row and the paired `response/finalize` row.
- `has_watch` (**UInt8**) — The request whether the `watch` has been set.

- `op_num` (**Enum**) — The type of request or response.
- `path` (**String**) — The path to the ZooKeeper node specified in the request, or an empty string if the request does not require specifying a path.
- `data` (**String**) — The data written to the ZooKeeper node (for the `SET` and `CREATE` requests — what the request wanted to write, for the response to the `GET` request — what was read) or an empty string.
- `is_ephemeral` (**UInt8**) — Is the ZooKeeper node being created as an **ephemeral**.
- `is_sequential` (**UInt8**) — Is the ZooKeeper node being created as an **sequential**.
- `version` (**Nullable(Int32)**) — The version of the ZooKeeper node that the request expects when executing. This is supported for `CHECK`, `SET`, `REMOVE` requests (is relevant `-1` if the request does not check the version or `NULL` for other requests that do not support version checking).
- `requests_size` (**UInt32**) — The number of requests included in the multi request (this is a special request that consists of several consecutive ordinary requests and executes them atomically). All requests included in multi request will have the same `xid`.
- `request_idx` (**UInt32**) — The number of the request included in multi request (for multi request — `0`, then in order from `1`).

Columns with request response parameters:

- `xzid` (**Int64**) — ZooKeeper transaction ID. The serial number issued by the ZooKeeper server in response to a successfully executed request (`0` if the request was not executed/returned an error/the client does not know whether the request was executed).
- `error` (**Nullable(Enum)**) — Error code. Can have many values, here are just some of them:
  - `ZOK` — The request was executed successfully.
  - `ZCONNECTIONLOSS` — The connection was lost.
  - `ZOPERATIONTIMEOUT` — The request execution timeout has expired.
  - `ZSESSIONEXPIRED` — The session has expired.
  - `NULL` — The request is completed.
- `watch_type` (**Nullable(Enum)**) — The type of the `watch` event (for responses with `op_num = Watch`), for the remaining responses: `NULL`.
- `watch_state` (**Nullable(Enum)**) — The status of the `watch` event (for responses with `op_num = Watch`), for the remaining responses: `NULL`.
- `path_created` (**String**) — The path to the created ZooKeeper node (for responses to the `CREATE` request), may differ from the `path` if the node is created as a `sequential`.
- `stat_czxid` (**Int64**) — The `xzid` of the change that caused this ZooKeeper node to be created.
- `stat_mzxid` (**Int64**) — The `xzid` of the change that last modified this ZooKeeper node.
- `stat_pzxid` (**Int64**) — The transaction ID of the change that last modified children of this ZooKeeper node.
- `stat_version` (**Int32**) — The number of changes to the data of this ZooKeeper node.
- `stat_cversion` (**Int32**) — The number of changes to the children of this ZooKeeper node.
- `stat_dataLength` (**Int32**) — The length of the data field of this ZooKeeper node.
- `stat_numChildren` (**Int32**) — The number of children of this ZooKeeper node.

- `children` ([Array\(String\)](#)) — The list of child ZooKeeper nodes (for responses to LIST request).

## Example

Query:

```
SELECT * FROM system.zookeeper_log WHERE (session_id = '106662742089334927') AND (xid = '10858') FORMAT  
Vertical;
```

Result:

Row 1:

```
type: Request
event_date: 2021-08-09
event_time: 2021-08-09 21:38:30.291792
address: ::
port: 2181
session_id: 106662742089334927
xid: 10858
has_watch: 1
op_num: List
path: /clickhouse/task_queue/ddl
data:
is_ephemeral: 0
is_sequential: 0
version: NULL
requests_size: 0
request_idx: 0
zxid: 0
error: NULL
watch_type: NULL
watch_state: NULL
path_created:
stat_czid: 0
stat_mzid: 0
stat_pzid: 0
stat_version: 0
stat_cversion: 0
stat_dataLength: 0
stat_numChildren: 0
children: []
```

Row 2:

```
type: Response
event_date: 2021-08-09
event_time: 2021-08-09 21:38:30.292086
address: ::
port: 2181
session_id: 106662742089334927
xid: 10858
has_watch: 1
op_num: List
path: /clickhouse/task_queue/ddl
data:
is_ephemeral: 0
is_sequential: 0
version: NULL
requests_size: 0
request_idx: 0
zxid: 16926267
error: ZOK
watch_type: NULL
watch_state: NULL
path_created:
stat_czid: 16925469
stat_mzid: 16925469
stat_pzid: 16926179
stat_version: 0
stat_cversion: 7
stat_dataLength: 0
stat_numChildren: 7
children: ['query-0000000006','query-0000000005','query-0000000004','query-0000000003','query-0000000002','query-0000000001','query-0000000000']
```

## See Also

- [ZooKeeper](#)
- [ZooKeeper guide](#)

## システム表

システムテーブルは、システムの機能の一部を実装したり、システムの動作に関する情報へのアクセスを提供するために使用されます。

システムテーブルを削除することはできません（ただし、DETACHを実行できます）。

システムテーブルのないファイルデータのハードディスクまたはファイルとメタデータを指すものとします。サーバーは起動時にすべてのシステムテーブルを作成します。

システムテーブルは読み取り専用です。

彼らはに位置しています ‘system’ データベース。

## システム asynchronous\_metrics

バックグラウンドで定期的に計算される指標が含まれます。例えば、使用中のRAMの量。

列:

- metric (文字列) — Metric name.
- value (Float64) — Metric value.

例

```
SELECT * FROM system.asynchronous_metrics LIMIT 10
```

metric	value
jemalloc.background_thread.run_interval	0
jemalloc.background_thread.num_runs	0
jemalloc.background_thread.num_threads	0
jemalloc.retained	422551552
jemalloc.mapped	1682989056
jemalloc.resident	1656446976
jemalloc.metadata_thp	0
jemalloc.metadata	10226856
UncompressedCacheCells	0
MarkCacheFiles	0

も参照。

- [監視](#) — Base concepts of ClickHouse monitoring.
- [システムメトリック](#) — Contains instantly calculated metrics.
- [システムイベント](#) — Contains a number of events that have occurred.
- [システムmetric\\_log](#) — Contains a history of metrics values from tables `system.metrics` и `system.events`.

## システムクラスタ

についての情報が含まれてクラスターのコンフィグファイルをサーバーです。

列:

- cluster (String) — The cluster name.
- shard\_num (UInt32) — The shard number in the cluster, starting from 1.
- shard\_weight (UInt32) — The relative weight of the shard when writing data.
- replica\_num (UInt32) — The replica number in the shard, starting from 1.

- `host_name` (String) — The host name, as specified in the config.
- `host_address` (String) — The host IP address obtained from DNS.
- `port` (UInt16) — The port to use for connecting to the server.
- `user` (String) — The name of the user for connecting to the server.
- `errors_count` (UInt32)-このホストがレプリカに到達できなかった回数。
- `estimated_recovery_time` (UInt32)-レプリカエラーカウントがゼロになり、正常に戻るまでの秒数。

ご注意ください `errors_count` クラスタに対するクエリごとに一度updatedされますが `estimated_recovery_time` オンデマンドで再計算されます。だから、ゼロ以外の場合があるかもしれません `errors_count` とゼロ `estimated_recovery_time` 次のクエリはゼロになります `errors_count` エラーがないかのようにreplicaを使用してみてください。

も参照。

- 分散テーブルエンジン
- `distributed_replica_error_cap`設定
- `distributed_replica_error_half_life`設定

## システム列

すべてのテーブルの列に関する情報を格納します。

このテーブルを使用すると、次のような情報を取得できます **DESCRIBE TABLE** クエリが、一度に複数のテーブルのために。

その `system.columns` テーブルを含む以下のカラムのカラムタイプはプラケット):

- `database` (String) — Database name.
- `table` (String) — Table name.
- `name` (String) — Column name.
- `type` (String) — Column type.
- `default_kind` (String) — Expression type (DEFAULT, MATERIALIZED, ALIAS デフォルト値の場合は)、定義されていない場合は空の文字列。
- `default_expression` (String) — Expression for the default value, or an empty string if it is not defined.
- `data_compressed_bytes` (UInt64) — The size of compressed data, in bytes.
- `data_uncompressed_bytes` (UInt64) — The size of decompressed data, in bytes.
- `marks_bytes` (UInt64) — The size of marks, in bytes.
- `comment` (String) — Comment on the column, or an empty string if it is not defined.
- `is_in_partition_key` (UInt8) — Flag that indicates whether the column is in the partition expression.
- `is_in_sorting_key` (UInt8) — Flag that indicates whether the column is in the sorting key expression.
- `is_in_primary_key` (UInt8) — Flag that indicates whether the column is in the primary key expression.
- `is_in_sampling_key` (UInt8) — Flag that indicates whether the column is in the sampling key expression.

## システム貢献者

を含むに関する情報提供者が保持しています。ランダムな順序ですべてのcontributors。順序は、クエリ実行時にランダムです。

列:

- name (String) — Contributor (author) name from git log.

例

```
SELECT * FROM system.contributors LIMIT 10
```

name
Olga Khvostikova
Max Vetrov
LiuYangkuan
svladykin
zamulla
Šimon Podlipský
BayoNet
Ilya Khomutov
Amy Krishnevsky
Loud_Scream

テーブル内で自分自身を知るには、クエリを使用します:

```
SELECT * FROM system.contributors WHERE name='Olga Khvostikova'
```

name
Olga Khvostikova

## システムデータ

このテーブルを含む単一の文字列カラムと呼ばれ ‘name’ – the name of a database.

各データベースのサーバーについて知っていて対応するエントリの表に示す。

このシステムテーブルは、SHOW DATABASES クエリ。

## システムdetached\_parts

についての情報が含まれて外部 メルゲツリー テーブル その reason column 部品が切り離された理由を指定します。

ユーザーが取り外した部品の場合、その理由は空です。このような部品は、ALTER TABLE ATTACH

PARTITION|PART コマンド その他の列の説明については、システム部品。パート名が無効な場合、一部のカラムの値は次のようにになります NULL。このような部分は、以下で削除できます ALTER TABLE DROP DETACHED PART.

## システム辞書

についての情報が含まれて 外部辞書。

列:

- database (文字列) — Name of the database containing the dictionary created by DDL query. Empty string for other dictionaries.
- name (文字列) — 辞書名.

- `status` ([Enum8](#)) — Dictionary status. Possible values:
  - `NOT_LOADED` — Dictionary was not loaded because it was not used.
  - `LOADED` — Dictionary loaded successfully.
  - `FAILED` — Unable to load the dictionary as a result of an error.
  - `LOADING` — Dictionary is loading now.
  - `LOADED_AND_RELOADING` — Dictionary is loaded successfully, and is being reloaded right now (frequent reasons: [SYSTEM RELOAD DICTIONARY](#) クエリ、タイムアウト、辞書の設定が変更されました)。
  - `FAILED_AND_RELOADING` — Could not load the dictionary as a result of an error and is loading now.
- `origin` ([文字列](#)) — Path to the configuration file that describes the dictionary.
- `type` ([文字列](#)) — Type of a dictionary allocation. メモリへの辞書の格納.
- `key` — キータイプ: 数値キー ([UInt64](#)) or Composite key ([文字列](#)) — form “(type 1, type 2, ..., type n)”.
- `attribute.names` ([配列\(文字列\)](#)) — Array of 属性名 辞書によって提供されます。
- `attribute.types` ([配列\(文字列\)](#)) — Corresponding array of 属性タイプ それは辞書によって提供されます。
- `bytes_allocated` ([UInt64](#)) — Amount of RAM allocated for the dictionary.
- `query_count` ([UInt64](#)) — Number of queries since the dictionary was loaded or since the last successful reboot.
- `hit_rate` ([Float64](#)) — For cache dictionaries, the percentage of uses for which the value was in the cache.
- `element_count` ([UInt64](#)) — Number of items stored in the dictionary.
- `load_factor` ([Float64](#)) — Percentage filled in the dictionary (for a hashed dictionary, the percentage filled in the hash table).
- `source` ([文字列](#)) — Text describing the データソース 辞書のために。
- `lifetime_min` ([UInt64](#)) — Minimum 生涯 その後、ClickHouseは辞書をリロードしようとします（もし `invalidate_query` それが変更された場合にのみ、設定されています）。秒単位で設定します。
- `lifetime_max` ([UInt64](#)) — Maximum 生涯 その後、ClickHouseは辞書をリロードしようとします（もし `invalidate_query` それが変更された場合にのみ、設定されています）。秒単位で設定します。
- `loading_start_time` ([DateTime](#)) — Start time for loading the dictionary.
- `last_successful_update_time` ([DateTime](#)) — End time for loading or updating the dictionary. Helps to monitor some troubles with external sources and investigate causes.
- `loading_duration` ([Float32](#)) — Duration of a dictionary loading.
- `last_exception` ([文字列](#)) — Text of the error that occurs when creating or reloading the dictionary if the dictionary couldn't be created.

## 例

辞書を設定します。

```

CREATE DICTIONARY dictdb.dict
(
    `key` Int64 DEFAULT -1,
    `value_default` String DEFAULT 'world',
    `value_expression` String DEFAULT 'xxx' EXPRESSION 'toString(127 * 172)'
)
PRIMARY KEY key
SOURCE(CLICKHOUSE(HOST 'localhost' PORT 9000 USER 'default' TABLE 'dicttbl' DB 'dictdb'))
LIFETIME(MIN 0 MAX 1)
LAYOUT(FLAT())

```

辞書が読み込まれていることを確認します。

```
SELECT * FROM system.dictionaries
```

database	name	status	origin	type	key	attribute.names	attribute.types	bytes_allocated	query_count	hit_rate	element_count	load_factor	source
dictdb	dict	LOADED	dictdb.dict	Flat	UInt64	['value_default','value_expression']	['String','String']	74032	0	1	1	0.0004887585532746823	ClickHouse: dictdb.dicttbl
2020-03-04	04:17:34		2020-03-04	04:30:34								0.002	0   1

## システムイベント

システムで発生したイベントの数に関する情報が含まれます。例えば、テーブルでどのように多くの `SELECT` ClickHouseサーバーが起動してからクエリが処理されました。

列:

- `event` (文字列) — Event name.
- `value` (UInt64) — Number of events occurred.
- `description` (文字列) — Event description.

例

```
SELECT * FROM system.events LIMIT 5
```

event	value	description
Query	12   Number of queries to be interpreted and potentially executed. Does not include queries that failed to parse or were rejected due to AST size limits, quota limits or limits on the number of simultaneously running queries. May include internal queries initiated by ClickHouse itself. Does not count subqueries.	
SelectQuery	8   Same as Query, but only for SELECT queries.	
FileOpen	73   Number of files opened.	
ReadBufferFromFileDescriptorRead	155   Number of reads (read/pread) from a file descriptor. Does not include sockets.	
ReadBufferFromFileDescriptorReadBytes	9931   Number of bytes read from file descriptors. If the file is compressed, this will show the compressed data size.	

も参照。

- システム [asynchronous\\_metrics](#) — Contains periodically calculated metrics.
- システム [メトリック](#) — Contains instantly calculated metrics.
- システム [metric\\_log](#) — Contains a history of metrics values from tables `system.metrics` и `system.events`.
- [監視](#) — Base concepts of ClickHouse monitoring.

## システム関数

標準関数と集計関数に関する情報が含まれます。

列:

- `name(String)` – The name of the function.
- `is_aggregate(UInt8)` — Whether the function is aggregate.

## システム [graphite\\_retentions](#)

パラメ [graphite\\_rollup](#) テーブルで使用される [\\*GraphiteMergeTree](#) エンジンだ

列:

- `config_name` (文字列) - [graphite\\_rollup](#) パラメータ名。
- `regexp (String)`- メトリック名のパターン。
- `function (String)`- 集計関数の名前。
- `age (UInt64)`- データの最小年齢を秒単位で表します。
- `precision (UInt64)`- データの年齢を秒単位で正確に定義する方法。
- `priority (UInt16)`- パターンの優先度。
- `is_default (UInt8)`- パターンがデフォルトかどうか。
- `Tables.database (Array(String))`- データベーステーブルの名前の配列。 `config_name` パラメータ。
- `Tables.table (Array(String))`- テーブル名の配列 `config_name` パラメータ。

## システムマージ

MergeTreeファミリー内のテーブルで現在処理中のマージおよびパートの変異に関する情報が含まれます。

列:

- `database` (String) — The name of the database the table is in.
- `table` (String) — Table name.
- `elapsed` (Float64) — The time elapsed (in seconds) since the merge started.
- `progress` (Float64) — The percentage of completed work from 0 to 1.
- `num_parts` (UInt64) — The number of pieces to be merged.
- `result_part_name` (String) — The name of the part that will be formed as the result of merging.
- `is_mutation` (UInt8)-1このプロセスが部分突然変異である場合。
- `total_size_bytes_compressed` (UInt64) — The total size of the compressed data in the merged chunks.
- `total_size_marks` (UInt64) — The total number of marks in the merged parts.
- `bytes_read_uncompressed` (UInt64) — Number of bytes read, uncompressed.
- `rows_read` (UInt64) — Number of rows read.
- `bytes_written_uncompressed` (UInt64) — Number of bytes written, uncompressed.
- `rows_written` (UInt64) — Number of rows written.

## システムメトリック

即座に計算できるメトリック、または現在の値が含まれます。たとえば、同時に処理されたクエリの数や現在のレプリカ遅延などです。このテーブルは常に最新です。

列:

- `metric` (文字列) — Metric name.
- `value` (Int64) — Metric value.
- `description` (文字列) — Metric description.

サポートされている指標のリストは、次のとおりです [src/Common/CurrentMetrics.cpp](#) ClickHouseのソースファイル。

例

```
SELECT * FROM system.metrics LIMIT 10
```

metric	value	description
Query	1	Number of executing queries
Merge	0	Number of executing background merges
PartMutation	0	Number of mutations (ALTER DELETE/UPDATE)
ReplicatedFetch	0	Number of data parts being fetched from replicas
ReplicatedSend	0	Number of data parts being sent to replicas
ReplicatedChecks	0	Number of data parts checking for consistency
BackgroundPoolTask	0	Number of active tasks in BackgroundProcessingPool (merges, mutations, fetches, or replication queue bookkeeping)
BackgroundSchedulePoolTask	0	Number of active tasks in BackgroundSchedulePool. This pool is used for periodic ReplicatedMergeTree tasks, like cleaning old data parts, altering data parts, replica re-initialization, etc.
DiskSpaceReservedForMerge	0	Disk space reserved for currently running background merges. It is slightly more than the total size of currently merging parts.
DistributedSend	0	Number of connections to remote servers sending data that was INSERTed into Distributed tables. Both synchronous and asynchronous mode.

も参照。

- システム[asynchronous\\_metrics](#) — Contains periodically calculated metrics.
- システムイベント — Contains a number of events that occurred.
- システム[metric\\_log](#) — Contains a history of metrics values from tables `system.metrics` и `system.events`.
- [監視](#) — Base concepts of ClickHouse monitoring.

## システム[metric\\_log](#)

を含む履歴メトリクスの値からテーブル `system.metrics` と `system.events`、定期的にディスクにフラッシュ。

メトリック履歴の収集を有効にするには `system.metric_log`, 作成 `/etc/clickhouse-server/config.d/metric_log.xml` 次の内容を使って:

```
<clickhouse>
  <metric_log>
    <database>system</database>
    <table>metric_log</table>
    <flush_interval_milliseconds>7500</flush_interval_milliseconds>
    <collect_interval_milliseconds>1000</collect_interval_milliseconds>
  </metric_log>
</clickhouse>
```

例

```
SELECT * FROM system.metric_log LIMIT 1 FORMAT Vertical;
```

Row 1:

```
event_date:          2020-02-18
event_time:         2020-02-18 07:15:33
milliseconds:       554
ProfileEvent_Query:    0
ProfileEvent_SelectQuery: 0
ProfileEvent_InsertQuery: 0
ProfileEvent_FileOpen:   0
ProfileEvent_Seek:      0
ProfileEvent_ReadBufferFromFileDescriptorRead: 1
ProfileEvent_ReadBufferFromFileDescriptorReadFailed: 0
ProfileEvent_ReadBufferFromFileDescriptorReadBytes: 0
ProfileEvent_WriteBufferFromFileDescriptorWrite: 1
ProfileEvent_WriteBufferFromFileDescriptorWriteFailed: 0
ProfileEvent_WriteBufferFromFileDescriptorWriteBytes: 56
...
CurrentMetric_Query: 0
CurrentMetric_Merge: 0
CurrentMetric_PartMutation: 0
CurrentMetric_ReplicatedFetch: 0
CurrentMetric_ReplicatedSend: 0
CurrentMetric_ReplicatedChecks: 0
...
...
```

も参照。

- システム [asynchronous\\_metrics](#) — Contains periodically calculated metrics.
- システムイベント — Contains a number of events that occurred.
- システムメトリック — Contains instantly calculated metrics.
- [監視](#) — Base concepts of ClickHouse monitoring.

## システム数字

このテーブルを一 UInt64 カラム名 ‘number’ ゼロから始まるほぼすべての自然数が含まれています。  
このテーブルは、テストのため、またはブルートフォース検索を行う必要がある場合に使用できます。  
この表からの読み取りは並列化されません。

## システム numbers\_mt

と同じ ‘system.numbers’ しかし、読み取りは並列処理されます。番号は任意の順序で返すことができます。  
テストに使用されます。

## システムワン

このテーブルには、单一の行が含まれています。‘dummy’ UInt8 値を含む列 0。  
このテーブルは、SELECT クエリで FROM 句が指定されていない場合に使用されます。  
これは、他の Dbms にあるデュアルテーブルに似ています。

## システム部品

についての情報が含まれて部品の [メルゲツリー](#) テーブル

各行は、一つのデータ部分を記述します。

列:

- `partition` (`String`) – The partition name. To learn what a partition is, see the description of the [ALTER](#) クエリ。

形式:

- `YYYYMM` 月別の自動パーティション分割の場合。
- `any_string` 手動で分割する場合。
- `name` (`String`) – Name of the data part.
- `active` (`UInt8`) – Flag that indicates whether the data part is active. If a data part is active, it's used in a table. Otherwise, it's deleted. Inactive data parts remain after merging.
- `marks` (`UInt64`) – The number of marks. To get the approximate number of rows in a data part, multiply `marks` インデックスの粒度（通常は8192）で指定します（このヒントは適応的な粒度では機能しません）。
- `rows` (`UInt64`) – The number of rows.
- `bytes_on_disk` (`UInt64`) – Total size of all the data part files in bytes.
- `data_compressed_bytes` (`UInt64`) – Total size of compressed data in the data part. All the auxiliary files (for example, files with marks) are not included.
- `data_uncompressed_bytes` (`UInt64`) – Total size of uncompressed data in the data part. All the auxiliary files (for example, files with marks) are not included.
- `marks_bytes` (`UInt64`) – The size of the file with marks.
- `modification_time` (`DateTime`) – The time the directory with the data part was modified. This usually corresponds to the time of data part creation.]
- `remove_time` (`DateTime`) – The time when the data part became inactive.
- `refcount` (`UInt32`) – The number of places where the data part is used. A value greater than 2 indicates that the data part is used in queries or merges.
- `min_date` (`Date`) – The minimum value of the date key in the data part.
- `max_date` (`Date`) – The maximum value of the date key in the data part.
- `min_time` (`DateTime`) – The minimum value of the date and time key in the data part.
- `max_time` (`DateTime`) – The maximum value of the date and time key in the data part.
- `partition_id` (`String`) – ID of the partition.
- `min_block_number` (`UInt64`) – The minimum number of data parts that make up the current part after merging.
- `max_block_number` (`UInt64`) – The maximum number of data parts that make up the current part after merging.
- `level` (`UInt32`) – Depth of the merge tree. Zero means that the current part was created by insert rather than by merging other parts.
- `data_version` (`UInt64`) – Number that is used to determine which mutations should be applied to the data part (mutations with a version higher than `data_version`).
- `primary_key_bytes_in_memory` (`UInt64`) – The amount of memory (in bytes) used by primary key values.

- `primary_key_bytes_in_memory_allocated` (UInt64) – The amount of memory (in bytes) reserved for primary key values.
- `is_frozen` (UInt8) – Flag that shows that a partition data backup exists. 1, the backup exists. 0, the backup doesn't exist. For more details, see [FREEZE PARTITION](#)
- `database` (String) – Name of the database.
- `table` (String) – Name of the table.
- `engine` (String) – Name of the table engine without parameters.
- `path` (String) – Absolute path to the folder with data part files.
- `disk` (String) – Name of a disk that stores the data part.
- `hash_of_all_files` (String) – [sipHash128](#) 圧縮されたファイルの。
- `hash_of_uncompressed_files` (String) – [sipHash128](#) 非圧縮ファイル（マーク付きファイル、インデックスファイルなど。）。
- `uncompressed_hash_of_compressed_files` (String) – [sipHash128](#) それらが圧縮されていないかのように圧縮されたファイル内のデータの。
- `bytes` (UInt64) – Alias for `bytes_on_disk`.
- `marks_size` (UInt64) – Alias for `marks_bytes`.

## システム `part_log`

その `system.part_log` テーブルが作成されるのは、[part\\_log](#) サーバ設定を指定します。

このテーブルについての情報が含まれてイベントが発生した [データパート](#) で [メルゲツリー](#) データの追加やマージなどのファミリテーブル。

その `system.part_log` テーブルを含む以下のカラム:

- `event_type` (Enum) — Type of the event that occurred with the data part. Can have one of the following values:
  - `NEW_PART` — Inserting of a new data part.
  - `MERGE_PARTS` — Merging of data parts.
  - `DOWNLOAD_PART` — Downloading a data part.
  - `REMOVE_PART` — Removing or detaching a data part using [DETACH PARTITION](#).
  - `MUTATE_PART` — Mutating of a data part.
  - `MOVE_PART` — Moving the data part from the one disk to another one.
- `event_date` (Date) — Event date.
- `event_time` (DateTime) — Event time.
- `duration_ms` (UInt64) — Duration.
- `database` (String) — Name of the database the data part is in.
- `table` (String) — Name of the table the data part is in.
- `part_name` (String) — Name of the data part.

- `partition_id` (String) — ID of the partition that the data part was inserted to. The column takes the ‘all’ パーティション分割が `tuple()`.
- `rows` (UInt64) — The number of rows in the data part.
- `size_in_bytes` (UInt64) — Size of the data part in bytes.
- `merged_from` (Array(String)) — An array of names of the parts which the current part was made up from (after the merge).
- `bytes_uncompressed` (UInt64) — Size of uncompressed bytes.
- `read_rows` (UInt64) — The number of rows was read during the merge.
- `read_bytes` (UInt64) — The number of bytes was read during the merge.
- `error` (UInt16) — The code number of the occurred error.
- `exception` (String) — Text message of the occurred error.

その `system.part_log` テーブルは、最初にデータを挿入した後に作成されます。 `MergeTree` テーブル。

## システムプロセス

このシステムテーブルは、`SHOW PROCESSLIST` クエリ。

列:

- `user` (String) – The user who made the query. Keep in mind that for distributed processing, queries are sent to remote servers under the `default` ユーザー。の分野のユーザー名で特定のクエリは、クエリはこのクエリも開始しています。
- `address` (String) – The IP address the request was made from. The same for distributed processing. To track where a distributed query was originally made from, look at `system.processes` クエリ要求サーバー上。
- `elapsed` (Float64) – The time in seconds since request execution started.
- `rows_read` (UInt64) – The number of rows read from the table. For distributed processing, on the requestor server, this is the total for all remote servers.
- `bytes_read` (UInt64) – The number of uncompressed bytes read from the table. For distributed processing, on the requestor server, this is the total for all remote servers.
- `total_rows_approx` (UInt64) – The approximation of the total number of rows that should be read. For distributed processing, on the requestor server, this is the total for all remote servers. It can be updated during request processing, when new sources to process become known.
- `memory_usage` (UInt64) – Amount of RAM the request uses. It might not include some types of dedicated memory. See the `max_memory_usage` 設定。
- `query` (String) – The query text. For `INSERT`,挿入するデータは含まれません。
- `query_id` (String) – Query ID, if defined.

## システムtext\_log

を含むログイン作品の応募がありました。 ログレベルがこのテーブルで限定 `text_log.level` サーバー設定。

列:

- `event_date` (Date) - エントリの日付。

- `event_time` (`DateTime`) - エントリの時間。
- `microseconds` (`UInt32`) - エントリのマイクロ秒。
- `thread_name` (`String`) — Name of the thread from which the logging was done.
- `thread_id` (`UInt64`) — OS thread ID.
- `level` (`Enum8`) - エントリーレベル。
  - `'Fatal'` = 1
  - `'Critical'` = 2
  - `'Error'` = 3
  - `'Warning'` = 4
  - `'Notice'` = 5
  - `'Information'` = 6
  - `'Debug'` = 7
  - `'Trace'` = 8
- `query_id` (`String`) - クエリのID。
- `logger_name` (`LowCardinality(String)`) - Name of the logger (i.e. `DDLWorker`)
- `message` (`String`) - メッセージ自体。
- `revision` (`UInt32`) - ClickHouse リビジョン。
- `source_file` (`LowCardinality(String)`) - ロギングが行われたソースファイル。
- `source_line` (`UInt64`) - ロギングが行われたソース行。

## システム `query_log`

クエリの実行に関する情報が含まれます。 クエリごとに、処理開始時間、処理時間、エラーメッセージおよびその他の情報を確認できます。

### 注

テーブルには以下の入力データは含まれません `INSERT` クエリ。

ClickHouseはこのテーブルを作成します。`query_log` `server` パラメータを指定します。 このパラメーターは、クエリがログインするテーブルのログ間隔や名前などのログルールを設定します。

クエリロギングを有効にするには、`log_queries` パラメータは1。 詳細については、[設定 セクション](#)

その `system.query_log` テーブルレジスタの種類は問合せ:

1. クライアントによって直接実行された初期クエリ。
2. 他のクエリによって開始された子クエリ(分散クエリ実行用)。 これらのタイプのクエリについては、親クエリに関する情報が `initial_*` 列。

列:

- `type` (Enum8) — Type of event that occurred when executing the query. Values:
  - `'QueryStart' = 1` — Successful start of query execution.
  - `'QueryFinish' = 2` — Successful end of query execution.
  - `'ExceptionBeforeStart' = 3` — Exception before the start of query execution.
  - `'ExceptionWhileProcessing' = 4` — Exception during the query execution.
- `event_date` (Date) — Query starting date.
- `event_time` (DateTime) — Query starting time.
- `query_start_time` (DateTime) — Start time of query execution.
- `query_duration_ms` (UInt64) — Duration of query execution.
- `read_rows` (UInt64) — Number of read rows.
- `read_bytes` (UInt64) — Number of read bytes.
- `written_rows` (UInt64) — For `INSERT` クエリ、書き込まれた行の数。 その他のクエリの場合、列の値は0です。
- `written_bytes` (UInt64) — For `INSERT` クエリ、書き込まれたバイト数。 その他のクエリの場合、列の値は0です。
- `result_rows` (UInt64) — Number of rows in the result.
- `result_bytes` (UInt64) — Number of bytes in the result.
- `memory_usage` (UInt64) — Memory consumption by the query.
- `query` (String) — Query string.
- `exception` (String) — Exception message.
- `stack_trace` (String) — Stack trace (a list of methods called before the error occurred). An empty string, if the query is completed successfully.
- `is_initial_query` (UInt8) — Query type. Possible values:
  - 1 — Query was initiated by the client.
  - 0 — Query was initiated by another query for distributed query execution.
- `user` (String) — Name of the user who initiated the current query.
- `query_id` (String) — ID of the query.
- `address` (IPv6) — IP address that was used to make the query.
- `port` (UInt16) — The client port that was used to make the query.
- `initial_user` (String) — Name of the user who ran the initial query (for distributed query execution).
- `initial_query_id` (String) — ID of the initial query (for distributed query execution).
- `initial_address` (IPv6) — IP address that the parent query was launched from.
- `initial_port` (UInt16) — The client port that was used to make the parent query.
- `interface` (UInt8) — Interface that the query was initiated from. Possible values:
  - 1 — TCP.
  - 2 — HTTP.

- `os_user` (String) — OS's username who runs **clickhouse-クライアント**.
- `client_hostname` (String) — Hostname of the client machine where the **clickhouse-クライアント** または他のTCPクライアントが実行されます。
- `client_name` (String) — The **clickhouse-クライアント** または別のTCPクライアント名。
- `client_revision` (UInt32) — Revision of the **clickhouse-クライアント** または別のTCPクライアント。
- `client_version_major` (UInt32) — Major version of the **clickhouse-クライアント** または別のTCPクライアント。
- `client_version_minor` (UInt32) — Minor version of the **clickhouse-クライアント** または別のTCPクライアント。
- `client_version_patch` (UInt32) — Patch component of the **clickhouse-クライアント** 別のTCPクライアントバージョン。
- `http_method` (UInt8) — HTTP method that initiated the query. Possible values:
  - 0 — The query was launched from the TCP interface.
  - 1 — `GET` 方法を用いた。
  - 2 — `POST` 方法を用いた。
- `http_user_agent` (String) — The `UserAgent` HTTP要求で渡されるヘッダー。
- `quota_key` (String) — The “`quota key`” で指定される。 **クオータ** 設定(参照 `keyed`).
- `revision` (UInt32) — ClickHouse revision.
- `thread_numbers` (Array(UInt32)) — Number of threads that are participating in query execution.
- `ProfileEvents` (Map(String, UInt64)) — ProfileEvents that measure different metrics. The description of them could be found in the table **システムイベント**
- `Settings` (Map(String, String)) — Settings 列。

それぞれのクエリでは、一つまたは二つの行が `query_log` クエリのステータスに応じて、テーブル:

1. クエリの実行が成功すると、タイプ1とタイプ2のイベントが作成されます。`type` 列)。
2. クエリ処理中にエラーが発生した場合は、タイプ1と4のイベントが作成されます。
3. クエリを起動する前にエラーが発生した場合は、タイプ3の单一のイベントが作成されます。

既定では、ログは7.5秒間隔でテーブルに追加されます。この間隔は `query_log` サーバ設定(参照 `flush_interval_milliseconds` 変数)。ログをメモリバッファからテーブルに強制的にフラッシュするには、`SYSTEM FLUSH LOGS` クエリ。

テーブルを手動で削除すると、その場で自動的に作成されます。以前のログはすべて削除されます。

## 注

ログの保存期間は無制限です。ログはテーブルから自動的には削除されません。古いログの削除自分で整理する必要があります。

パーティショニングキーを指定できます。`system.query_log` のテーブル `query_log` サーバ設定(参照 `partition_by` 変数)。

## システム `query_thread_log`

のテーブルについての情報が含まれてそれぞれの検索キーワード実行スレッド。

ClickHouseはこのテーブルを作成します。`query_thread_log` serverパラメータを指定します。このパラメーターは、クエリがログインするテーブルのログ間隔や名前などのログルールを設定します。

クエリロギングを有効にするには、`log_query_threads` パラメータは1。 詳細については、[設定 セクション](#)列:

- `event_date` (Date) — the date when the thread has finished execution of the query.
- `event_time` (DateTime) — the date and time when the thread has finished execution of the query.
- `query_start_time` (DateTime) — Start time of query execution.
- `query_duration_ms` (UInt64) — Duration of query execution.
- `read_rows` (UInt64) — Number of read rows.
- `read_bytes` (UInt64) — Number of read bytes.
- `written_rows` (UInt64) — For `INSERT` クエリ、書き込まれた行の数。 その他のクエリの場合、列の値は0です。
- `written_bytes` (UInt64) — For `INSERT` クエリ、書き込まれたバイト数。 その他のクエリの場合、列の値は0です。
- `memory_usage` (Int64) — The difference between the amount of allocated and freed memory in context of this thread.
- `peak_memory_usage` (Int64) — The maximum difference between the amount of allocated and freed memory in context of this thread.
- `thread_name` (String) — Name of the thread.
- `thread_number` (UInt32) — Internal thread ID.
- `os_thread_id` (Int32) — OS thread ID.
- `master_thread_id` (UInt64) — OS initial ID of initial thread.
- `query` (String) — Query string.
- `is_initial_query` (UInt8) — Query type. Possible values:
  - 1 — Query was initiated by the client.
  - 0 — Query was initiated by another query for distributed query execution.
- `user` (String) — Name of the user who initiated the current query.
- `query_id` (String) — ID of the query.
- `address` (IPv6) — IP address that was used to make the query.
- `port` (UInt16) — The client port that was used to make the query.
- `initial_user` (String) — Name of the user who ran the initial query (for distributed query execution).
- `initial_query_id` (String) — ID of the initial query (for distributed query execution).
- `initial_address` (IPv6) — IP address that the parent query was launched from.
- `initial_port` (UInt16) — The client port that was used to make the parent query.
- `interface` (UInt8) — Interface that the query was initiated from. Possible values:
  - 1 — TCP.
  - 2 — HTTP.

- `os_user` (String) — OS's username who runs **clickhouse-クライアント**.
- `client_hostname` (String) — Hostname of the client machine where the **clickhouse-クライアント** または他の TCPクライアントが実行されます。
- `client_name` (String) — The **clickhouse-クライアント** または別のTCPクライアント名。
- `client_revision` (UInt32) — Revision of the **clickhouse-クライアント** または別のTCPクライアント。
- `client_version_major` (UInt32) — Major version of the **clickhouse-クライアント** または別のTCPクライアント。
- `client_version_minor` (UInt32) — Minor version of the **clickhouse-クライアント** または別のTCPクライアント。
- `client_version_patch` (UInt32) — Patch component of the **clickhouse-クライアント** 別のTCPクライアントバージョン。
- `http_method` (UInt8) — HTTP method that initiated the query. Possible values:
  - 0 — The query was launched from the TCP interface.
  - 1 — **GET** 方法を用いた。
  - 2 — **POST** 方法を用いた。
- `http_user_agent` (String) — The **UserAgent** HTTP要求で渡されるヘッダー。
- `quota_key` (String) — The “**quota key**” で指定される。 **クオータ** 設定(参照 `keyed`).
- `revision` (UInt32) — ClickHouse revision.
- `ProfileEvents` (Map(String, UInt64)) — ProfileEvents that measure different metrics for this thread. The description of them could be found in the table **システムイベント**

既定では、ログは7.5秒間隔でテーブルに追加されます。この間隔は **query\_thread\_log** サーバ設定(参照 `flush_interval_milliseconds` 変数)。ログをメモリバッファからテーブルに強制的にフラッシュするには、**SYSTEM FLUSH LOGS** クエリ。

テーブルを手動で削除すると、その場で自動的に作成されます。以前のログはすべて削除されます。

## 注

ログの保存期間は無制限です。ログはテーブルから自動的には削除されません。古いログの削除自分で整理する必要があります。

パーティショニングキーを指定できます。`system.query_thread_log` のテーブル **query\_thread\_log** サーバ設定(参照 `partition_by` 変数)。

## システム**trace\_log**

を含むスタックトレースの収集、サンプリングクロフライ。

ClickHouseはこのテーブルを作成します。**trace\_log** サーバの設定が設定されます。また、**query\_profiler\_real\_time\_period\_ns** と **query\_profiler\_cpu\_time\_period\_ns** 設定は設定する必要があります。

ログを分析するには、`addressToLine`, `addressToSymbol` と `demangle` イントロスペクション関数。

列:

- `event_date` (**日付**) — Date of sampling moment.
- `event_time` (**DateTime**) — Timestamp of the sampling moment.

- `timestamp_ns` (`UInt64`) — Timestamp of the sampling moment in nanoseconds.
- `revision` (`UInt32`) — ClickHouse server build revision.  
サーバーに接続するとき `clickhouse-client` のような文字列を参照してください `Connected to ClickHouse server version 19.18.1 revision 54429..` このフィールドには `revision` ではなく、`version` サーバーの。
- `timer_type` (`Enum8`) — Timer type:
  - `Real` 壁時計の時間を表します。
  - `CPU` CPU時間を表します。
- `thread_number` (`UInt32`) — Thread identifier.
- `query_id` (`文字列`) — Query identifier that can be used to get details about a query that was running from the `query_log` システムテーブル。
- `trace` (`配列(UInt64)`) — Stack trace at the moment of sampling. Each element is a virtual memory address inside ClickHouse server process.

例

```
SELECT * FROM system.trace_log LIMIT 1 \G
```

Row 1:

```
event_date: 2019-11-15
event_time: 2019-11-15 15:09:38
revision: 54428
timer_type: Real
thread_number: 48
query_id: acc4d61f-5bd1-4a3e-bc91-2180be37c915
trace:
[94222141367858,94222152240175,94222152325351,94222152329944,94222152330796,94222151449980,94222144088167,94222151682763,94222144088167,94222151682763,94222144088167,94222144058283,94222144059248,94222091840750,94222091842302,94222091831228,94222189631488,140509950166747,140509942945935]
```

## システムレプリカ

情報および状況を再現しテーブル在住の地元のサーバーです。  
このテーブルは監視に使用することができる。のテーブルが含まれて行毎に再現\*ます。

例:

```
SELECT *
FROM system.replicas
WHERE table = 'visits'
FORMAT Vertical
```

## Row 1:

```
database:      merge
table:        visits
engine:       ReplicatedCollapsingMergeTree
is_leader:    1
can_become_leader: 1
is_READONLY: 0
is_session_expired: 0
future_parts: 1
parts_to_check: 0
zookeeper_path: /clickhouse/tables/01-06/visits
replica_name: example01-06-1.yandex.ru
replica_path: /clickhouse/tables/01-06/visits/replicas/example01-06-1.yandex.ru
columns_version: 9
queue_size: 1
inserts_in_queue: 0
merges_in_queue: 1
part_mutations_in_queue: 0
queue_oldest_time: 2020-02-20 08:34:30
inserts_oldest_time: 1970-01-01 00:00:00
merges_oldest_time: 2020-02-20 08:34:30
part_mutations_oldest_time: 1970-01-01 00:00:00
oldest_part_to_get:
oldest_part_to_merge_to: 20200220_20284_20840_7
oldest_part_to_mutate_to:
log_max_index: 596273
log_pointer: 596274
last_queue_update: 2020-02-20 08:34:32
absolute_delay: 0
total_replicas: 2
active_replicas: 2
```

列:

- **database (String)**-データベース名
- **table (String)**-テーブル名
- **engine (String)**-テーブルエンジン名
- **is\_leader (UInt8)** -レプリカがリーダーであるかどうか。  
一度に一つのレプリカだけがリーダーになります。リーダーは、実行するバックグラウンドマージを選択します。書き込みは、リーダーであるかどうかに関係なく、使用可能でZKにセッションがある任意のレプリカに対して実行できます。
- **can\_become\_leader (UInt8)** -レプリカがリーダーとして選出できるかどうか。
- **is\_READONLY (UInt8)**-レプリカが読み取り専用モードであるかどうか。  
このモードは、設定にZooKeeperのセクションがない場合、zookeeperでセッションを再初期化するとき、およびZooKeeperでセッションを再初期化するときに不明なエラーが発生
- **is\_session\_expired (UInt8)** -ZooKeeperとのセッションが終了しました。基本的には **is\_READONLY**.
- **future\_parts (UInt32)**-まだ行われていない挿入またはマージの結果として表示されるデータパートの数。
- **parts\_to\_check (UInt32)**-検証のためのキュー内のデータ部分の数。破損の疑いがある場合は、部品を検証キューに入れます。
- **zookeeper\_path (String)** -ZooKeeperのテーブルデータへのパス。
- **replica\_name (String)** -飼育係のレプリカ名。同じテーブルの異なるレプリカの名前は異なります。
- **replica\_path (String)** -ZooKeeperのレプリカデータへのパス。連結と同じ 'zookeeper\_path/replicas/replica\_path'.

- `columns_version` (`Int32`)-テーブル構造のバージョン番号。変更が実行された回数を示します。場合にレプリカは異なるバージョンで一部のレプリカさんのすべての変更はまだない。
- `queue_size` (`UInt32`)-実行待ちの操作のキューのサイズ。操作には、データのブロックの挿入、マージ、その他の特定の操作が含まれます。それは通常と一致します `future_parts`.
- `inserts_in_queue` (`UInt32`)-必要なデータブロックの挿入数。挿入は通常、かなり迅速に複製されます。この数が大きい場合は、何かが間違っていることを意味します。
- `merges_in_queue` (`UInt32`)-行われるのを待っているマージの数。時にはマージが長いので、この値は長い間ゼロより大きくなることがあります。
- `part_mutations_in_queue` (`UInt32`) -作られるのを待っている突然変異の数。
- `queue_oldest_time` (`DateTime`)-`queue_size` 0 より大きい場合は、最も古い操作がいつキューに追加されたかを示します。
- `inserts_oldest_time` (`DateTime`) -参照 `queue_oldest_time`
- `merges_oldest_time` (`DateTime`) -参照 `queue_oldest_time`
- `part_mutations_oldest_time` (`DateTime`) -参照 `queue_oldest_time`

次の4つの列は、zkとのアクティブなセッションがある場合にのみゼロ以外の値を持ちます。

- `log_max_index` (`UInt64`) -一般的な活動のログ内の最大エントリ番号。
- `log_pointer` (`UInt64`)-レプリカがその実行キューにコピーした一般的なアクティビティのログ内の最大エントリ番号を加えたもの。もし `log_pointer` はるかに小さい `log_max_index` 何かおかしい
- `last_queue_update` (`DateTime`)-キューが前回updatedされたとき。
- `absolute_delay` (`UInt64`) -現在のレプリカが持っている秒単位の大きな遅れ。
- `total_replicas` (`UInt8`)-このテーブルの既知のレプリカの総数。
- `active_replicas` (`UInt8`)-ZooKeeperにセッションがあるこのテーブルのレプリカの数(つまり、機能しているレプリカの数)。

すべての列を要求すると、ZooKeeperからのいくつかの読み取りが行ごとに行われるため、テーブルは少し遅く動作する可能性があります。

最後の4つの列 (`log_max_index`、`log_pointer`、`total_replicas`、`active_replicas`) を要求しないと、テーブルはすぐに機能します。

たとえば、次のようにすべてが正常に動作していることを確認できます:

```

SELECT
    database,
    table,
    is_leader,
    is_readonly,
    is_session_expired,
    future_parts,
    parts_to_check,
    columns_version,
    queue_size,
    inserts_in_queue,
    merges_in_queue,
    log_max_index,
    log_pointer,
    total_replicas,
    active_replicas
FROM system.replicas
WHERE
    is_readonly
    OR is_session_expired
    OR future_parts > 20
    OR parts_to_check > 10
    OR queue_size > 20
    OR inserts_in_queue > 10
    OR log_max_index - log_pointer > 10
    OR total_replicas < 2
    OR active_replicas < total_replicas

```

このクエリが何も返さない場合は、すべて正常であることを意味します。

## システム設定

現在のユ

列:

- **name** (文字列) — Setting name.
- **value** (文字列) — Setting value.
- **changed** (UInt8) — Shows whether a setting is changed from its default value.
- **description** (文字列) — Short setting description.
- **min** (Null可能(文字列)) — Minimum value of the setting, if any is set via 制約. 最小値が設定されていない場合は、以下を含みます **NULL**.
- **max** (Null可能(文字列)) — Maximum value of the setting, if any is set via 制約. 最大値が設定されていない場合は、以下を含みます **NULL**.
- **readonly** (UInt8) — Shows whether the current user can change the setting:
  - 0 — Current user can change the setting.
  - 1 — Current user can't change the setting.

例

次の例は、名前に含まれる設定に関する情報を取得する方法を示しています **min\_i**.

```

SELECT *
FROM system.settings
WHERE name LIKE '%min_i%'

```

name	value	changed	description
	min	max	readonly
min_insert_block_size_rows in rows, if blocks are not big enough.	1048576	0   Squash blocks passed to INSERT query to specified size	NULL   NULL   0
min_insert_block_size_bytes in bytes, if blocks are not big enough.	268435456	0   Squash blocks passed to INSERT query to specified size	NULL   NULL   0
read_backoff_min_interval_between_events_ms	1000	0   Settings to reduce the number of threads in	
case of slow reads. Do not pay attention to the event, if the previous one has passed less than a certain amount of time.	NULL   NULL   0		

の使用 WHERE changed たとえば、チェックしたいときに便利です:

- 構成ファイルの設定が正しく読み込まれ、使用されているかどうか。
- 現在のセッションで変更された設定。

```
SELECT * FROM system.settings WHERE changed AND name='load_balancing'
```

も参照。

- [設定](#)
- [クエリの権限](#)
- [設定の制約](#)

## システムtable\_engines

name	value
max_threads	8
use_uncompressed_cache	0
load_balancing	random
max_memory_usage	100000000000

## システムmerge\_tree\_settings

の設定に関する情報が含まれます MergeTree テーブル

列:

- `name` (String) — Setting name.
- `value` (String) — Setting value.
- `description` (String) — Setting description.
- `type` (String) — Setting type (implementation specific string value).
- `changed` (UInt8) — Whether the setting was explicitly defined in the config or explicitly changed.

## システムtable\_engines

を含むの記述のテーブルエンジンをサポートサーバーとその特徴を支援す。

このテーブル以下のカラムのカラムタイプはブレケット):

- `name` (String) — The name of table engine.
- `supports_settings` (UInt8) — Flag that indicates if table engine supports `SETTINGS` 句。
- `supports_skipping_indices` (UInt8) — Flag that indicates if table engine supports 索引のスキップ。
- `supports_ttl` (UInt8) — Flag that indicates if table engine supports `TTL`.
- `supports_sort_order` (UInt8) — Flag that indicates if table engine supports clauses `PARTITION_BY`, `PRIMARY_KEY`, `ORDER_BY` と `SAMPLE_BY`.
- `supports_replication` (UInt8) — Flag that indicates if table engine supports データ複製。
- `supports_deduplication` (UInt8) — Flag that indicates if table engine supports data deduplication.

例:

```
SELECT *
FROM system.table_engines
WHERE name in ('Kafka', 'MergeTree', 'ReplicatedCollapsingMergeTree')
```

name	supports_ttl	supports_replication	supports_deduplication	supports_settings	supports_skipping_indices	supports_sort_order
Kafka	1	0	0	0	1	0
MergeTree	1	1	1	1	1	0
ReplicatedCollapsingMergeTree	1	1	1	1	1	1

も参照。

- メルゲツリー族 クエリ句
- カフカ 設定
- 参加 設定

## システムテーブル

を含むメタデータは各テーブルサーバーに知っています。デタッチされたテーブルは `system.tables`。

このテーブル以下のカラムのカラムタイプは括弧内:

- `database` (String) — The name of the database the table is in.
- `name` (String) — Table name.
- `engine` (String) — Table engine name (without parameters).
- `is_temporary` (UInt8)-テーブルが一時的かどうかを示すフラグ。
- `data_path` (String)-ファイルシステム内のテーブルデータへのパス。
- `metadata_path` (String)-ファイルシステム内のテーブルメタデータへのパス。
- `metadata_modification_time` (DateTime)-テーブルメタデータの最新の変更時刻。
- `dependencies_database` (Array(String))-データベースの依存関係。
- `dependencies_table` (Array(String))-テーブルの依存関係 (**マテリアライズドビュー** 現在のテーブルに基づくテーブル)。

- `create_table_query` (String)-テーブルの作成に使用されたクエリ。
- `engine_full` (String)-テーブルエンジンのパラメータ。
- `partition_key` (String)-テーブルで指定されたパーティションキー式。
- `sorting_key` (String)-テーブルで指定されたソートキー式。
- `primary_key` (String)-テーブルで指定された主キー式。
- `sampling_key` (String)-テーブルで指定されたサンプリングキー式。
- `storage_policy` (String)-ストレージポリシー:
  - メルゲツリー
  - 分散
- `total_rows` (Nullable(UInt64))-テーブル内の正確な行数をすばやく決定できる場合は、行の合計数です。 Null (アンダーリングを含む `Buffer` 表)。
- `total_bytes` (Nullable(UInt64))-ストレージ上のテーブルの正確なバイト数を迅速に決定できる場合は、合計バイト数。 Null (ない 基になるストレージを含む)。
  - If the table stores data on disk, returns used space on disk (i.e. compressed).
  - テーブルがメモリにデータを格納している場合は、メモリ内の使用バイト数の概算を返します。

その `system.tables` テーブルはで使用されます `SHOW TABLES` クエリの実装。

## システム飼育係

ZooKeeperが設定されていない場合、テーブルは存在しません。できるデータを読み込んで飼育係クラスタで定義され、config.

クエリには ‘path’ WHERE 句の等価条件。これは、データを取得する子供のためのZooKeeperのパスです。

クエリ `SELECT * FROM system.zookeeper WHERE path = '/clickhouse'` すべての子のデータを出力します。 /clickhouse ノード  
すべてのルートノードのデータを出力するには、`path= '/'`.  
指定されたパスの場合 ‘path’ 存在しない場合、例外がスローされます。

例:

- `name` (String) — The name of the node.
- `path` (String) — The path to the node.
- `value` (String) — Node value.
- `dataLength` (Int32) — Size of the value.
- `numChildren` (Int32) — Number of descendants.
- `czxid` (Int64) — ID of the transaction that created the node.
- `mzxid` (Int64) — ID of the transaction that last changed the node.
- `pzxid` (Int64) — ID of the transaction that last deleted or added descendants.
- `ctime` (DateTime) — Time of node creation.
- `mtime` (DateTime) — Time of the last modification of the node.

- `version` (Int32) — Node version: the number of times the node was changed.
- `cversion` (Int32) — Number of added or removed descendants.
- `aversion` (Int32) — Number of changes to the ACL.
- `ephemeralOwner` (Int64) — For ephemeral nodes, the ID of the session that owns this node.

例:

```
SELECT *
FROM system.zookeeper
WHERE path = '/clickhouse/tables/01-08/visits/replicas'
FORMAT Vertical
```

Row 1:

```
name: example01-08-1.yandex.ru
value:
czxid: 932998691229
mzxid: 932998691229
ctime: 2015-03-27 16:49:51
mtime: 2015-03-27 16:49:51
version: 0
cversion: 47
aversion: 0
ephemeralOwner: 0
dataLength: 0
numChildren: 7
pzxid: 987021031383
path: /clickhouse/tables/01-08/visits/replicas
```

Row 2:

```
name: example01-08-2.yandex.ru
value:
czxid: 933002738135
mzxid: 933002738135
ctime: 2015-03-27 16:57:01
mtime: 2015-03-27 16:57:01
version: 0
cversion: 37
aversion: 0
ephemeralOwner: 0
dataLength: 0
numChildren: 7
pzxid: 987021252247
path: /clickhouse/tables/01-08/visits/replicas
```

## システム突然変異

のテーブルについての情報が含まれて **突然変異** マージツリーテーブルとその進捗状況。各突然変異コマンドは单一の行で表されます。テーブルには次の列があります:

**データ**, テーブル - 突然変異が適用されたデータベースとテーブルの名前。

**mutation\_id** - 突然変異のID。レプリケートされたテーブルの場合、これらのIdは

`<table_path_in_zookeeper>/mutations/` 飼育係のディレクトリ。未複製テーブルの場合、Idはテーブルのデータディレクトリ内のファイル名に対応します。

**コマンド** - 突然変異コマンド文字列 (後のクエリの部分 `ALTER TABLE [db.]table`).

**create\_time** - この突然変異コマンドが実行のために提出されたとき。

ロック番号 **partition\_id**, ブロック番号番号 -入れ子になった列。パーティションIDと、その変異によって取得されたロック番号(各パーティションでは、そのパーティション内の変異によって取得されたロック番号 非複製のテーブル、ロック番号の全ての仕切りがひとつのシーケンスです。こないということを意味している変異体再現し、テーブルの列として展開しているのが記録するとともにシングルロック番号の取得による突然変異が原因です。

**parts\_to\_do** -突然変異が完了するために突然変異する必要があるデータ部分の数。

**is\_done** -突然変異は？たとえ `parts_to_do = 0` 変更する必要がある新しいデータパートを作成する長時間実行される `INSERT`のために、複製されたテーブルの突然変異がまだ行われていない可能性があり

一部のパートの変更に問題がある場合は、次の列に追加情報が含まれています：

**latest\_failed\_part** -変異することができなかった最新の部分の名前。

**latest\_fail\_time** -最も最近の部分突然変異の失敗の時間。

**latest\_fail\_reason** -最新の部品突然変異の失敗を引き起こした例外メッセージ。

## システムディスク

ディスク サーバー構成。

列:

- `name` (文字列) — Name of a disk in the server configuration.
- `path` (文字列) — Path to the mount point in the file system.
- `free_space` (UInt64) — Free space on disk in bytes.
- `total_space` (UInt64) — Disk volume in bytes.
- `keep_free_space` (UInt64) — Amount of disk space that should stay free on disk in bytes. Defined in the `keep_free_space_bytes` ディスク構成のパラメータ。

## システムストレージポリシー

ストレージポリシ サーバー構成。

列:

- `policy_name` (文字列) — Name of the storage policy.
- `volume_name` (文字列) — Volume name defined in the storage policy.
- `volume_priority` (UInt64) — Volume order number in the configuration.
- `disks` (配列(文字列)) — Disk names, defined in the storage policy.
- `max_data_part_size` (UInt64) — Maximum size of a data part that can be stored on volume disks (0 — no limit).
- `move_factor` (Float64) — Ratio of free disk space. When the ratio exceeds the value of configuration parameter, ClickHouse start to move data to the next volume in order.

ストレージポリシーに複数のボリュームが含まれている場合は、各ボリュームの情報がテーブルの個々の行に格納されます。

## サンプリングクロファイル

ClickHouse運転サンプリングプロファイラでの分析クエリを実行します。 Profilerを使用すると、クエリの実行中に最も頻繁に使用されるソースコードルーチンを検索できます。 CPU時間とアイドル時間を含む壁時計時間をトレースできます。

Profilerを使用するには:

- セットアップ `trace_log` サーバー構成のセクション。

このセクションでは、`trace_log` プロファイラ機能の結果を含むシステムテーブル。これは既定で構成されています。この表のデータは、実行中のサーバーに対してのみ有効です。後、サーバを再起動ClickHouseないクリーンのテーブルに格納された仮想メモリアドレスが無効になります。

- セットアップ `query_profiler_cpu_time_period_ns` または `query_profiler_real_time_period_ns` 設定。両方の設定を同時に使用できます。

これらの設定を許可する設定プロファイラータイマー。これらはセッション設定であるため、サーバー全体、個々のユーザーまたはユーザープロファイル、対話式セッション、および個々のクエリごとに異なるサンプリング周波数

デフォルトのサンプリング周波数はサンプルや、CPU、リアルタイマーが有効になっています。この頻度により、ClickHouse clusterに関する十分な情報を収集できます。同時に、この頻度で作業すると、profilerはClickHouse serverのパフォーマンスには影響しません。が必要な場合にプロファイル毎に個別のクエリを利用して高サンプリング周波数です。

分析するには `trace_log` システム表:

- インストール `clickhouse-common-static-dbg` パッケージ。見る [DEBパッケージから](#)。
- によるintrospection関数を許可する。`allow_introspection_functions` 設定。

セキュリティ上の理由から、introspection関数は既定で無効になっています。

- 使用する `addressToLine`, `addressToSymbol` と `demangle` 内観関数 ClickHouseコードで関数名とその位置を取得するには。いくつかのクエリのプロファイルを取得するには、`trace_log` テーブル。個々の関数またはスタックトレース全体でデータを集計できます。

視覚化する必要がある場合 `trace_log` 情報、試して [フランメグラフ](#) と [スピードスコープ](#)。

## 例

この例では、:

- フィルタ処理 `trace_log` クエリ識別子と現在の日付によるデータ。
- スタックトレースによる集計。
- イントロスペクション関数を使用して、我々はのレポートを取得します:
  - シンボルおよび対応するソースコード関数の名前。
  - これらの関数のソースコードの場所。

```
SELECT
    count(),
    arrayStringConcat(arrayMap(x -> concat(demangle(addressToSymbol(x)), '\n  ', addressToLine(x)), trace), '\n') AS
sym
FROM system.trace_log
WHERE (query_id = 'ebca3574-ad0a-400a-9cbc-dca382f5998c') AND (event_date = today())
GROUP BY trace
ORDER BY count() DESC
LIMIT 10
```

Row 1:

```
count(): 6344
sym: StackTrace::StackTrace(ucontext_t const&)
/home/milovidov/ClickHouse/build_gcc9/../src/Common/StackTrace.cpp:208
DB::(anonymous namespace)::writeTraceInfo(DB::TimerType, int, siginfo_t*, void*) [clone .isra.0]
/home/milovidov/ClickHouse/build_gcc9/../src/IO/BufferBase.h:99

read

DB::ReadBufferFromFileDescriptor::nextImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/IO/ReadBufferFromFileDescriptor.cpp:56
DB::CompressedReadBufferBase::readCompressedData(unsigned long&, unsigned long&)
/home/milovidov/ClickHouse/build_gcc9/../src/IO/ReadBuffer.h:54
DB::CompressedReadBufferFromFile::nextImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/Compression/CompressedReadBufferFromFile.cpp:22
DB::CompressedReadBufferFromFile::seek(unsigned long, unsigned long)
/home/milovidov/ClickHouse/build_gcc9/../src/Compression/CompressedReadBufferFromFile.cpp:63
DB::MergeTreeReaderStream::seekToMark(unsigned long)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeReaderStream.cpp:200
std::_Function_handler<DB::ReadBuffer* (std::vector<DB::IDataType::Substream,
std::allocator<DB::IDataType::Substream> > const&),
DB::MergeTreeReader::readData(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >
const&, DB::IDataType const&, DB::IColumn&, unsigned long, bool, unsigned long, bool)::
{lambda(bool)#1}::operator()(bool) const::{lambda(std::vector<DB::IDataType::Substream,
std::allocator<DB::IDataType::Substream> > const&)#1}>::_M_invoke(std::_Any_data const&,
std::vector<DB::IDataType::Substream, std::allocator<DB::IDataType::Substream> > const&)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeReader.cpp:212
DB::IDataType::deserializeBinaryBulkWithMultipleStreams(DB::IColumn&, unsigned long,
DB::IDataType::DeserializeBinaryBulkSettings&, std::shared_ptr<DB::IDataType::DeserializeBinaryBulkState>&) const
/usr/local/include/c++/9.1.0/bits/std_function.h:690
DB::MergeTreeReader::readData(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >
const&, DB::IDataType const&, DB::IColumn&, unsigned long, bool, unsigned long, bool)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeReader.cpp:232
DB::MergeTreeReader::readRows(unsigned long, bool, unsigned long, DB::Block&)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeReader.cpp:111
DB::MergeTreeRangeReader::DelayedStream::finalize(DB::Block&)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeRangeReader.cpp:35
DB::MergeTreeRangeReader::continueReadingChain(DB::MergeTreeRangeReader::ReadResult&)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeRangeReader.cpp:219
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeRangeReader.cpp:487
DB::MergeTreeBaseSelectBlockInputStream::readFromPartImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeBaseSelectBlockInputStream.cpp:158
DB::MergeTreeBaseSelectBlockInputStream::readImpl()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ExpressionBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/ExpressionBlockInputStream.cpp:34
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::PartialSortingBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/PartialSortingBlockInputStream.cpp:13
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::loop(unsigned long)
/usr/local/include/c++/9.1.0/bits/atomic_base.h:419
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::thread(std::shared_ptr<DB::ThreadGroupStatus
>, unsigned long)
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/ParallelInputsProcessor.h:215
ThreadFromGlobalPool::ThreadFromGlobalPool<void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*)(std::shared_ptr<DB::ThreadGroupStatus>,
unsigned long), DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*>,
std::shared_ptr<DB::ThreadGroupStatus>, unsigned long&)(void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*&&)
(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long),
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*&&,
std::shared_ptr<DB::ThreadGroupStatus>&&, unsigned long)::{lambda()#1}::operator()() const
/usr/local/include/c++/9.1.0/bits/shared_ptr_base.h:729
ThreadPoolImpl<std::thread>::worker(std::list<std::thread>)
/usr/local/include/c++/9.1.0/bits/unique_lock.h:69
execute_native_thread_routine
/home/milovidov/ClickHouse/ci/workspace/gcc/gcc-build/x86_64-pc-linux-gnu/libstdc++-
v3/include/bits/unique_ptr.h:81
start_thread
```

\_clone

Row 2:

count(): 3295  
sym: StackTrace::StackTrace(ucontext\_t const&)  
/home/milovidov/ClickHouse/build\_gcc9/../src/Common/StackTrace.cpp:208  
DB::(anonymous namespace)::writeTraceInfo(DB::TimerType, int, siginfo\_t\*, void\*) [clone .isra.0]  
/home/milovidov/ClickHouse/build\_gcc9/../src/IO/BufferBase.h:99

\_pthread\_cond\_wait

std::condition\_variable::wait(std::unique\_lock<std::mutex>&)  
/home/milovidov/ClickHouse/ci/workspace/gcc/gcc-build/x86\_64-pc-linux-gnu/libstdc++-v3/src/c++11/../../../../gcc-9.1.0/libstdc++-v3/src/c++11/condition\_variable.cc:55  
Poco::Semaphore::wait()  
/home/milovidov/ClickHouse/build\_gcc9/../../../contrib/poco/Foundation/src/Semaphore.cpp:61  
DB::UnionBlockInputStream::readImpl()  
/usr/local/include/c++/9.1.0/x86\_64-pc-linux-gnu/bits/gthr-default.h:748  
DB::IBlockInputStream::read()  
/usr/local/include/c++/9.1.0/bits/stl\_vector.h:108  
DB::MergeSortingBlockInputStream::readImpl()  
/home/milovidov/ClickHouse/build\_gcc9/../../src/Core/Block.h:90  
DB::IBlockInputStream::read()  
/usr/local/include/c++/9.1.0/bits/stl\_vector.h:108  
DB::ExpressionBlockInputStream::readImpl()  
/home/milovidov/ClickHouse/build\_gcc9/../../src/DataStreams/ExpressionBlockInputStream.cpp:34  
DB::IBlockInputStream::read()  
/usr/local/include/c++/9.1.0/bits/stl\_vector.h:108  
DB::LimitBlockInputStream::readImpl()  
/usr/local/include/c++/9.1.0/bits/stl\_vector.h:108  
DB::IBlockInputStream::read()  
/usr/local/include/c++/9.1.0/bits/stl\_vector.h:108  
DB::AsynchronousBlockInputStream::calculate()  
/usr/local/include/c++/9.1.0/bits/stl\_vector.h:108  
std::\_Function\_handler<void (), DB::AsynchronousBlockInputStream::next()>::  
{lambda()#1}>::\_M\_invoke(std::Any\_data const&)  
/usr/local/include/c++/9.1.0/bits/atomic\_base.h:551  
ThreadPoolImpl<ThreadFromGlobalPool>::worker(std::list<ThreadFromGlobalPool>::iterator)<ThreadFromGlobalPool>  
/usr/local/include/c++/9.1.0/x86\_64-pc-linux-gnu/bits/gthr-default.h:748  
ThreadFromGlobalPool::ThreadFromGlobalPool<ThreadPoolImpl<ThreadFromGlobalPool>::scheduleImpl<void>(std::function<void ()>, int, std::optional<unsigned long>){lambda()#3}>  
(ThreadPoolImpl<ThreadFromGlobalPool>::scheduleImpl<void>(std::function<void ()>, int, std::optional<unsigned long>){lambda()#3} &&){lambda()#1}::operator()() const  
/home/milovidov/ClickHouse/build\_gcc9/../../src/Common/ThreadPool.h:146  
ThreadPoolImpl<std::thread>::worker(std::list<std::thread>::iterator)<std::thread>  
/usr/local/include/c++/9.1.0/bits/unique\_lock.h:69  
execute\_native\_thread\_routine  
/home/milovidov/ClickHouse/ci/workspace/gcc/gcc-build/x86\_64-pc-linux-gnu/libstdc++-v3/include/bits/unique\_ptr.h:81  
start\_thread

\_clone

Row 3:

count(): 1978  
sym: StackTrace::StackTrace(ucontext\_t const&)  
/home/milovidov/ClickHouse/build\_gcc9/../src/Common/StackTrace.cpp:208  
DB::(anonymous namespace)::writeTraceInfo(DB::TimerType, int, siginfo\_t\*, void\*) [clone .isra.0]  
/home/milovidov/ClickHouse/build\_gcc9/../src/IO/BufferBase.h:99

DB::VolnitskyBase<true, true, DB::StringSearcher<true, true>>::search(unsigned char const\*, unsigned long) const  
/opt/milovidov/ClickHouse/build\_gcc9/programs/clickhouse  
DB::MatchImpl<true, false>::vector\_constant(DB::PODArray<unsigned char, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul> const&, DB::PODArray<unsigned long, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul> const&, std::cxx11::basic\_string<char, std::char\_traits<char>, std::allocator<char> > const&, DB::PODArray<unsigned char, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul> &)  
/opt/milovidov/ClickHouse/build\_gcc9/programs/clickhouse  
DB::FunctionsStringSearch<DB::MatchImpl<true, false>, DB::NameLike>::executeImpl(DB::Block&, std::vector<unsigned long, std::allocator<unsigned long> > const&, unsigned long, unsigned long)  
/opt/milovidov/ClickHouse/build\_gcc9/programs/clickhouse

```

DB::PreparedFunctionImpl::execute(DB::Block&, std::vector<unsigned long, std::allocator<unsigned long> > const&, unsigned long, unsigned long, bool)
/home/milovidov/ClickHouse/build_gcc9/../src/Functions/IFunction.cpp:464
DB::ExpressionAction::execute(DB::Block&, bool) const
/usr/local/include/c++/9.1.0/bits/stl_vector.h:677
DB::ExpressionActions::execute(DB::Block&, bool) const
/home/milovidov/ClickHouse/build_gcc9/../src/Interpreters/ExpressionActions.cpp:739
DB::MergeTreeRangeReader::executePrewhereActionsAndFilterColumns(DB::MergeTreeRangeReader::ReadResult&)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeRangeReader.cpp:660
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeRangeReader.cpp:546
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::MergeTreeBaseSelectBlockInputStream::readFromPartImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeBaseSelectBlockInputStream.cpp:158
DB::MergeTreeBaseSelectBlockInputStream::readImpl()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ExpressionBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/ExpressionBlockInputStream.cpp:34
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::PartialSortingBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/PartialSortingBlockInputStream.cpp:13
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::loop(unsigned long)
/usr/local/include/c++/9.1.0/bits/atomic_base.h:419
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::thread(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long)
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/ParallelInputsProcessor.h:215
ThreadFromGlobalPool::ThreadFromGlobalPool<void>
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*)(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long), DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*, std::shared_ptr<DB::ThreadGroupStatus>, unsigned long&)(void)
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*&&)(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long),
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*&&, std::shared_ptr<DB::ThreadGroupStatus>&&, unsigned long&:{lambda()#1}::operator()() const
/usr/local/include/c++/9.1.0/bits/shared_ptr_base.h:729
ThreadPoolImpl<std::thread>::worker(std::list<std::thread>*)
/usr/local/include/c++/9.1.0/bits/unique_lock.h:69
execute_native_thread_routine
/home/milovidov/ClickHouse/ci/workspace/gcc/gcc-build/x86_64-pc-linux-gnu/libstdc++-v3/include/bits/unique_ptr.h:81
start_thread

__clone

```

Row 4:

---

```

count(): 1913
sym: StackTrace::StackTrace(ucontext_t const&)
/home/milovidov/ClickHouse/build_gcc9/../src/Common/StackTrace.cpp:208
DB::(anonymous namespace)::writeTraceInfo(DB::TimerType, int, siginfo_t*, void*) [clone .isra.0]
/home/milovidov/ClickHouse/build_gcc9/../src/IO/BufferBase.h:99

```

```

DB::VolnitskyBase<true, true, DB::StringSearcher<true, true> >::search(unsigned char const*, unsigned long) const
/opt/milovidov/ClickHouse/build_gcc9/programs/clickhouse
DB::MatchImpl<true, false>::vector_constant(DB::PODArray<unsigned char, 4096ul, AllocatorWithHint<false>, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul> const&, DB::PODArray<unsigned long, 4096ul, AllocatorWithHint<false>, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul> const&, std::cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, DB::PODArray<unsigned char, 4096ul, AllocatorWithHint<false>, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul>&)
/opt/milovidov/ClickHouse/build_gcc9/programs/clickhouse
DB::FunctionsStringSearch<DB::MatchImpl<true, false>, DB::NameLike>::executeImpl(DB::Block&, std::vector<unsigned long, std::allocator<unsigned long> > const&, unsigned long, unsigned long)
/opt/milovidov/ClickHouse/build_gcc9/programs/clickhouse
DB::PreparedFunctionImpl::execute(DB::Block&, std::vector<unsigned long, std::allocator<unsigned long> > const&, unsigned long, unsigned long, bool)
/home/milovidov/ClickHouse/build_gcc9/../src/Functions/IFunction.cpp:464
DB::ExpressionAction::execute(DB::Block&, bool) const
/usr/local/include/c++/9.1.0/bits/stl_vector.h:677
DB::ExpressionActions::execute(DB::Block&, bool) const

```

```
/home/milovidov/ClickHouse/build_gcc9/../src/Interpreters/ExpressionActions.cpp:739
DB::MergeTreeRangeReader::executePrewhereActionsAndFilterColumns(DB::MergeTreeRangeReader::ReadResult&
 /home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeRangeReader.cpp:660
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&
 /home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeRangeReader.cpp:546
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&
 /usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::MergeTreeBaseSelectBlockInputStream::readFromPartImpl()
 /home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeBaseSelectBlockInputStream.cpp:158
DB::MergeTreeBaseSelectBlockInputStream::readImpl()
 /usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::IBlockInputStream::read()
 /usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ExpressionBlockInputStream::readImpl()
 /home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/ExpressionBlockInputStream.cpp:34
DB::IBlockInputStream::read()
 /usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::PartialSortingBlockInputStream::readImpl()
 /home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/PartialSortingBlockInputStream.cpp:13
DB::IBlockInputStream::read()
 /usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::loop(unsigned long)
 /usr/local/include/c++/9.1.0/bits/atomic_base.h:419
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::thread(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long)
 /home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/ParallelInputsProcessor.h:215
ThreadFromGlobalPool::ThreadFromGlobalPool<void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*)(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long), DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*, std::shared_ptr<DB::ThreadGroupStatus>, unsigned long&>(void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*&&)
(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long),
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*&&, std::shared_ptr<DB::ThreadGroupStatus>&&, unsigned long&:{lambda()#1}::operator()() const
 /usr/local/include/c++/9.1.0/bits/shared_ptr_base.h:729
ThreadPoolImpl<std::thread>::worker(std::list<std::thread>)
 /usr/local/include/c++/9.1.0/bits/unique_lock.h:69
execute_native_thread_routine
 /home/milovidov/ClickHouse/ci/workspace/gcc/gcc-build/x86_64-pc-linux-gnu/libstdc++-v3/include/bits/unique_ptr.h:81
start_thread
```

\_clone

Row 5:

```
count(): 1672
sym: StackTrace::StackTrace(ucontext_t const&
 /home/milovidov/ClickHouse/build_gcc9/../src/Common/StackTrace.cpp:208
DB::(anonymous namespace)::writeTraceInfo(DB::TimerType, int, siginfo_t*, void*) [clone .isra.0]
 /home/milovidov/ClickHouse/build_gcc9/../src/IO/BufferBase.h:99
```

```
DB::VolnitskyBase<true, true, DB::StringSearcher<true, true> >::search(unsigned char const*, unsigned long) const
 /opt/milovidov/ClickHouse/build_gcc9/programs/clickhouse
DB::MatchImpl<true, false>::vector_constant(DB::PODArray<unsigned char, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul> const&, DB::PODArray<unsigned long, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul> const&, std::cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, DB::PODArray<unsigned char, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul>&)
 /opt/milovidov/ClickHouse/build_gcc9/programs/clickhouse
DB::FunctionsStringSearch<DB::MatchImpl<true, false>, DB::NameLike>::executeImpl(DB::Block&, std::vector<unsigned long, std::allocator<unsigned long> > const&, unsigned long, unsigned long)
 /opt/milovidov/ClickHouse/build_gcc9/programs/clickhouse
DB::PreparedFunctionImpl::execute(DB::Block&, std::vector<unsigned long, std::allocator<unsigned long> > const&, unsigned long, unsigned long, bool)
 /home/milovidov/ClickHouse/build_gcc9/../src/Functions/IFunction.cpp:464
DB::ExpressionAction::execute(DB::Block&, bool) const
 /usr/local/include/c++/9.1.0/bits/stl_vector.h:677
DB::ExpressionActions::execute(DB::Block&, bool) const
 /home/milovidov/ClickHouse/build_gcc9/../src/Interpreters/ExpressionActions.cpp:739
DB::MergeTreeRangeReader::executePrewhereActionsAndFilterColumns(DB::MergeTreeRangeReader::ReadResult&
 /home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeRangeReader.cpp:660
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&
 /home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeRangeReader.cpp:546
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&
```

```
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::MergeTreeBaseSelectBlockInputStream::readFromPartImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeBaseSelectBlockInputStream.cpp:158
DB::MergeTreeBaseSelectBlockInputStream::readImpl()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ExpressionBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/ExpressionBlockInputStream.cpp:34
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::PartialSortingBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/PartialSortingBlockInputStream.cpp:13
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::loop(unsigned long)
/usr/local/include/c++/9.1.0/bits/atomic_base.h:419
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::thread(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long)
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/ParallelInputsProcessor.h:215
ThreadFromGlobalPool::ThreadFromGlobalPool<void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*)(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long), DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*, std::shared_ptr<DB::ThreadGroupStatus>, unsigned long&>(void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*&&)
(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long),
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*&&, std::shared_ptr<DB::ThreadGroupStatus>&&, unsigned long&){lambda(#1)::operator()()} const
/usr/local/include/c++/9.1.0/bits/shared_ptr_base.h:729
ThreadPoolImpl<std::thread>::worker(std::list<std::thread>)
/usr/local/include/c++/9.1.0/bits/unique_lock.h:69
execute_native_thread_routine
/home/milovidov/ClickHouse/ci/workspace/gcc/gcc-build/x86_64-pc-linux-gnu/libstdc++-v3/include/bits/unique_ptr.h:81
start_thread

__clone
```

## Row 6:

```
count(): 1531
sym: StackTrace::StackTrace(ucontext_t const&
 /home/milovidov/ClickHouse/build_gcc9/../src/Common/StackTrace.cpp:208
DB::(anonymous namespace)::writeTraceInfo(DB::TimerType, int, siginfo_t*, void*) [clone .isra.0]
 /home/milovidov/ClickHouse/build_gcc9/../src/IO/BufferBase.h:99
```

## read

```
DB::ReadBufferFromFileDescriptor::nextImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/IO/ReadBufferFromFileDescriptor.cpp:56
DB::CompressedReadBufferBase::readCompressedData(unsigned long&, unsigned long&)
/home/milovidov/ClickHouse/build_gcc9/../src/IO/ReadBuffer.h:54
DB::CompressedReadBufferFromFile::nextImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/Compression/CompressedReadBufferFromFile.cpp:22
void DB::deserializeBinarySSE2<4>(DB::PODArray<unsigned char, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul>&, DB::PODArray<unsigned long, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul>&, DB::ReadBuffer&, unsigned long)
/home/milovidov/ClickHouse/build_gcc9/../src/IO/ReadBuffer.h:53
DB::DataTypeString::deserializeBinaryBulk(DB::IColumn&, DB::ReadBuffer&, unsigned long, double) const
/home/milovidov/ClickHouse/build_gcc9/../src/DataTypes/DataTypeString.cpp:202
DB::MergeTreeReader::readData(std::cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, DB::IDataType const&, DB::IColumn&, unsigned long, bool, unsigned long, bool)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeReader.cpp:232
DB::MergeTreeReader::readRows(unsigned long, bool, unsigned long, DB::Block&)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeReader.cpp:111
DB::MergeTreeRangeReader::DelayedStream::finalize(DB::Block&)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeRangeReader.cpp:35
DB::MergeTreeRangeReader::startReadingChain(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeRangeReader.cpp:219
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::MergeTreeBaseSelectBlockInputStream::readFromPartImpl()
```

```
/home/milovidov/ClickHouse/build_gcc9/..src/Storages/MergeTree/MergeTreeBaseSelectBlockInputStream.cpp:158
DB::MergeTreeBaseSelectBlockInputStream::readImpl()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ExpressionBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/..src/DataStreams/ExpressionBlockInputStream.cpp:34
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::PartialSortingBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/..src/DataStreams/PartialSortingBlockInputStream.cpp:13
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::loop(unsigned long)
/usr/local/include/c++/9.1.0/bits/atomic_base.h:419
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::thread(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long)
/home/milovidov/ClickHouse/build_gcc9/..src/DataStreams/ParallelInputsProcessor.h:215
ThreadFromGlobalPool::ThreadFromGlobalPool<void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*)(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long), DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*, std::shared_ptr<DB::ThreadGroupStatus>, unsigned long&>(void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*&&)
(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long),
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*&&, std::shared_ptr<DB::ThreadGroupStatus>&&, unsigned long&){lambda()#1}::operator()() const
/usr/local/include/c++/9.1.0/bits/shared_ptr_base.h:729
ThreadPoolImpl<std::thread>::worker(std::list<std::thread>)
/usr/local/include/c++/9.1.0/bits/unique_lock.h:69
execute_native_thread_routine
/home/milovidov/ClickHouse/ci/workspace/gcc/gcc-build/x86_64-pc-linux-gnu/libstdc++-v3/include/bits/unique_ptr.h:81
start_thread

_clone
```

Row 7:

```
count(): 1034
sym: StackTrace::StackTrace(ucontext_t const&
 /home/milovidov/ClickHouse/build_gcc9/..src/Common/StackTrace.cpp:208
DB::(anonymous namespace)::writeTraceInfo(DB::TimerType, int, siginfo_t*, void*) [clone .isra.0]
 /home/milovidov/ClickHouse/build_gcc9/..src/IO/BufferBase.h:99
```

```
DB::VolnitskyBase<true, true, DB::StringSearcher<true, true> >::search(unsigned char const*, unsigned long) const
 /opt/milovidov/ClickHouse/build_gcc9/programs/clickhouse
DB::MatchImpl<true, false>::vector_constant(DB::PODArray<unsigned char, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul> const&, DB::PODArray<unsigned long, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul> const&, std::cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, DB::PODArray<unsigned char, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul>&)
 /opt/milovidov/ClickHouse/build_gcc9/programs/clickhouse
DB::FunctionsStringSearch<DB::MatchImpl<true, false>, DB::NameLike>::executImpl(DB::Block&, std::vector<unsigned long, std::allocator<unsigned long> > const&, unsigned long, unsigned long)
 /opt/milovidov/ClickHouse/build_gcc9/programs/clickhouse
DB::PreparedFunctionImpl::execute(DB::Block&, std::vector<unsigned long, std::allocator<unsigned long> > const&, unsigned long, unsigned long, bool)
 /home/milovidov/ClickHouse/build_gcc9/..src/Functions/IFunction.cpp:464
DB::ExpressionAction::execute(DB::Block&, bool) const
 /usr/local/include/c++/9.1.0/bits/stl_vector.h:677
DB::ExpressionActions::execute(DB::Block&, bool) const
 /home/milovidov/ClickHouse/build_gcc9/..src/Interpreters/ExpressionActions.cpp:739
DB::MergeTreeRangeReader::executePrewhereActionsAndFilterColumns(DB::MergeTreeRangeReader::ReadResult&)
 /home/milovidov/ClickHouse/build_gcc9/..src/Storages/MergeTree/MergeTreeRangeReader.cpp:660
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
 /home/milovidov/ClickHouse/build_gcc9/..src/Storages/MergeTree/MergeTreeRangeReader.cpp:546
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
 /usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::MergeTreeBaseSelectBlockInputStream::readFromPartImpl()
 /home/milovidov/ClickHouse/build_gcc9/..src/Storages/MergeTree/MergeTreeBaseSelectBlockInputStream.cpp:158
DB::MergeTreeBaseSelectBlockInputStream::readImpl()
 /usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::IBlockInputStream::read()
 /usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ExpressionBlockInputStream::readImpl()
```

```
DB::IBlockInputStream::read()
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/ExpressionBlockInputStream.cpp:34
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::PartialSortingBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/PartialSortingBlockInputStream.cpp:13
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::loop(unsigned long)
/usr/local/include/c++/9.1.0/bits/atomic_base.h:419
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::thread(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long)
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/ParallelInputsProcessor.h:215
ThreadFromGlobalPool::ThreadFromGlobalPool<void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*)(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long), DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*, std::shared_ptr<DB::ThreadGroupStatus>, unsigned long&)(void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*&&)
(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long),
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*&&, std::shared_ptr<DB::ThreadGroupStatus>&&, unsigned long&){lambda()#1}::operator()() const
/usr/local/include/c++/9.1.0/bits/shared_ptr_base.h:729
ThreadPoolImpl<std::thread>::worker(std::list<std::unique_lock>)
/usr/local/include/c++/9.1.0/bits/unique_lock.h:69
execute_native_thread_routine
/home/milovidov/ClickHouse/ci/workspace/gcc/gcc-build/x86_64-pc-linux-gnu/libstdc++-v3/include/bits/unique_ptr.h:81
start_thread

__clone
```

Row 8:

```
count(): 989
sym: StackTrace::StackTrace(ucontext_t const&
/home/milovidov/ClickHouse/build_gcc9/../src/Common/StackTrace.cpp:208
DB::(anonymous namespace)::writeTraceInfo(DB::TimerType, int, siginfo_t*, void*) [clone .isra.0]
/home/milovidov/ClickHouse/build_gcc9/../src/I/O/BufferBase.h:99
```

\_\_ll\_lock\_wait

pthread\_mutex\_lock

```
DB::MergeTreeReaderStream::loadMarks()
/usr/local/include/c++/9.1.0/bits/std_mutex.h:103
DB::MergeTreeReaderStream::MergeTreeReaderStream(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> > const&, DB::MarkCache*, bool, DB::UncompressedCache*, unsigned long, unsigned long, unsigned long, DB::MergeTreeIndexGranularityInfo const*, std::function<void (DB::ReadBufferFromFileBase::ProfileInfo)> const&, int)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeReaderStream.cpp:107
std::function<void (std::vector<DB::IDataType::Substream, std::allocator<DB::IDataType::Substream> > const&), DB::MergeTreeReader::addStreams(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, DB::IDataType const&, std::function<void (DB::ReadBufferFromFileBase::ProfileInfo)> const&, int){lambda(std::vector<DB::IDataType::Substream, std::allocator<DB::IDataType::Substream> > const&#1>}::M_invoke(std::any_data const&, std::vector<DB::IDataType::Substream, std::allocator<DB::IDataType::Substream> > const&)
/usr/local/include/c++/9.1.0/bits/unique_ptr.h:147
DB::MergeTreeReader::addStreams(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, DB::IDataType const&, std::function<void (DB::ReadBufferFromFileBase::ProfileInfo)> const&, int)
/usr/local/include/c++/9.1.0/bits/stl_vector.h:677
DB::MergeTreeReader::MergeTreeReader(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, std::shared_ptr<DB::MergeTreeDataPart const> const&, DB::NamesAndTypesList const&, DB::UncompressedCache*, DB::MarkCache*, bool, DB::MergeTreeData const&, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> > const&, unsigned long, unsigned long, std::map<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, double, std::less<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > >, std::allocator<std::pair<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const, double> > > const&, std::function<void (DB::ReadBufferFromFileBase::ProfileInfo)> const&, int)
/usr/local/include/c++/9.1.0/bits/stl_list.h:303
DB::MergeTreeThreadSelectBlockInputStream::getNewTask()
/usr/local/include/c++/9.1.0/bits/std_function.h:259
DB::MergeTreeBaseSelectBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeBaseSelectBlockInputStream.cpp:54
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
```

```
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ExpressionBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/..src/DataStreams/ExpressionBlockInputStream.cpp:34
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::PartialSortingBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/..src/DataStreams/PartialSortingBlockInputStream.cpp:13
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::loop(unsigned long)
/usr/local/include/c++/9.1.0/bits/atomic_base.h:419
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::thread(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long)
/home/milovidov/ClickHouse/build_gcc9/..src/DataStreams/ParallelInputsProcessor.h:215
ThreadFromGlobalPool::ThreadFromGlobalPool<void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*)(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long), DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*, std::shared_ptr<DB::ThreadGroupStatus>, unsigned long&)(void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*&&)
(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long),
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*&&, std::shared_ptr<DB::ThreadGroupStatus>&&, unsigned long&)::{lambda()#1}::operator()() const
/usr/local/include/c++/9.1.0/bits/shared_ptr_base.h:729
ThreadPoolImpl<std::thread>::worker(std::list<std::thread>)
/usr/local/include/c++/9.1.0/bits/unique_lock.h:69
execute_native_thread_routine
/home/milovidov/ClickHouse/ci/workspace/gcc/gcc-build/x86_64-pc-linux-gnu/libstdc++-v3/include/bits/unique_ptr.h:81
start_thread
```

\_clone

Row 9:

```
count(): 779
sym: StackTrace::StackTrace(ucontext_t const&)
/home/milovidov/ClickHouse/build_gcc9/..src/Common/StackTrace.cpp:208
DB::(anonymous namespace)::writeTraceInfo(DB::TimerType, int, siginfo_t*, void*) [clone .isra.0]
/home/milovidov/ClickHouse/build_gcc9/..src/IO/BufferBase.h:99
```

```
void DB::deserializeBinarySSE2<4>(DB::PODArray<unsigned char, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul>&, DB::PODArray<unsigned long, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul>&, DB::ReadBuffer&, unsigned long)
/usr/local/lib/gcc/x86_64-pc-linux-gnu/9.1.0/include/emmintrin.h:727
DB::DataTypeString::deserializeBinaryBulk(DB::IColumn&, DB::ReadBuffer&, unsigned long, double) const
/home/milovidov/ClickHouse/build_gcc9/..src/DataTypes/DataTypeString.cpp:202
DB::MergeTreeReader::readData(std::cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, DB::IDataType const&, DB::IColumn&, unsigned long, bool, unsigned long, bool)
/home/milovidov/ClickHouse/build_gcc9/..src/Storages/MergeTree/MergeTreeReader.cpp:232
DB::MergeTreeReader::readRows(unsigned long, bool, unsigned long, DB::Block&)
/home/milovidov/ClickHouse/build_gcc9/..src/Storages/MergeTree/MergeTreeReader.cpp:111
DB::MergeTreeRangeReader::DelayedStream::finalize(DB::Block&)
/home/milovidov/ClickHouse/build_gcc9/..src/Storages/MergeTree/MergeTreeRangeReader.cpp:35
DB::MergeTreeRangeReader::startReadingChain(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
/home/milovidov/ClickHouse/build_gcc9/..src/Storages/MergeTree/MergeTreeRangeReader.cpp:219
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::MergeTreeBaseSelectBlockInputStream::readFromPartImpl()
/home/milovidov/ClickHouse/build_gcc9/..src/Storages/MergeTree/MergeTreeBaseSelectBlockInputStream.cpp:158
DB::MergeTreeBaseSelectBlockInputStream::readImpl()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ExpressionBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/..src/DataStreams/ExpressionBlockInputStream.cpp:34
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::PartialSortingBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/..src/DataStreams/PartialSortingBlockInputStream.cpp:13
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::loop(unsigned long)
/usr/local/include/c++/9.1.0/bits/atomic_base.h:419
```

```

/usr/local/include/c++/9.1.0/bits/atomic_base.h:419
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::thread(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long)
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/ParallelInputsProcessor.h:215
ThreadFromGlobalPool::ThreadFromGlobalPool<void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*)(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long), DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*, std::shared_ptr<DB::ThreadGroupStatus>, unsigned long&>(void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*&&)
(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long),
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*&&, std::shared_ptr<DB::ThreadGroupStatus>&&, unsigned long&:{lambda()#1}::operator()() const
/usr/local/include/c++/9.1.0/bits/shared_ptr_base.h:729
ThreadPoolImpl<std::thread>::worker(std::list<std::thread>)
/usr/local/include/c++/9.1.0/bits/unique_lock.h:69
execute_native_thread_routine
/home/milovidov/ClickHouse/ci/workspace/gcc/gcc-build/x86_64-pc-linux-gnu/libstdc++-v3/include/bits/unique_ptr.h:81
start_thread

__clone

```

Row 10:

```

count(): 666
sym: StackTrace::StackTrace(ucontext_t const&)
/home/milovidov/ClickHouse/build_gcc9/../src/Common/StackTrace.cpp:208
DB::(anonymous namespace)::writeTraceInfo(DB::TimerType, int, siginfo_t*, void*) [clone .isra.0]
/home/milovidov/ClickHouse/build_gcc9/../src/I/O/BufferBase.h:99

void DB::deserializeBinarySSE2<4>(DB::PODArray<unsigned char, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul>&, DB::PODArray<unsigned long, 4096ul, AllocatorWithHint<false, AllocatorHints::DefaultHint, 67108864ul>, 15ul, 16ul>&, DB::ReadBuffer&, unsigned long)
/usr/local/lib/gcc/x86_64-pc-linux-gnu/9.1.0/include/emmintrin.h:727
DB::DataTypeString::deserializeBinaryBulk(DB::IColumn&, DB::ReadBuffer&, unsigned long, double) const
/home/milovidov/ClickHouse/build_gcc9/../src/DataTypes/DataTypeString.cpp:202
DB::MergeTreeReader::readData(std::cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > const&, DB::IDataType const&, DB::IColumn&, unsigned long, bool, unsigned long, bool)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeReader.cpp:232
DB::MergeTreeReader::readRows(unsigned long, bool, unsigned long, DB::Block&)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeReader.cpp:111
DB::MergeTreeRangeReader::DelayedStream::finalize(DB::Block&)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeRangeReader.cpp:35
DB::MergeTreeRangeReader::startReadingChain(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeRangeReader.cpp:219
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::MergeTreeRangeReader::read(unsigned long, std::vector<DB::MarkRange, std::allocator<DB::MarkRange> >&)
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::MergeTreeBaseSelectBlockInputStream::readFromPartImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/Storages/MergeTree/MergeTreeBaseSelectBlockInputStream.cpp:158
DB::MergeTreeBaseSelectBlockInputStream::readImpl()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ExpressionBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/ExpressionBlockInputStream.cpp:34
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::PartialSortingBlockInputStream::readImpl()
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/PartialSortingBlockInputStream.cpp:13
DB::IBlockInputStream::read()
/usr/local/include/c++/9.1.0/bits/stl_vector.h:108
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::loop(unsigned long)
/usr/local/include/c++/9.1.0/bits/atomic_base.h:419
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::thread(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long)
/home/milovidov/ClickHouse/build_gcc9/../src/DataStreams/ParallelInputsProcessor.h:215
ThreadFromGlobalPool::ThreadFromGlobalPool<void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*)(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long), DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*, std::shared_ptr<DB::ThreadGroupStatus>, unsigned long&>(void
(DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>::*&&)
(std::shared_ptr<DB::ThreadGroupStatus>, unsigned long),
DB::ParallelInputsProcessor<DB::UnionBlockInputStream::Handler>*&&, std::shared_ptr<DB::ThreadGroupStatus>&&, unsigned long&:{lambda()#1}::operator()() const
/usr/local/include/c++/9.1.0/bits/shared_ptr_base.h:729
ThreadPoolImpl<std::thread>::worker(std::list<std::thread>)
/usr/local/include/c++/9.1.0/bits/unique_lock.h:69
execute_native_thread_routine
/home/milovidov/ClickHouse/ci/workspace/gcc/gcc-build/x86_64-pc-linux-gnu/libstdc++-v3/include/bits/unique_ptr.h:81
start_thread

__clone

```

```
DB::ParallelInputProcessor::DB::UnionBlockInputStream::handle>`&&,
std::shared_ptr<DB::ThreadGroupStatus>&&, unsigned long&){lambda()#1}::operator()() const
/usr/local/include/c++/9.1.0/bits/shared_ptr_base.h:729
ThreadPoolImpl<std::thread>::worker(std::list<std::thread>)
/usr/local/include/c++/9.1.0/bits/unique_lock.h:69
execute_native_thread_routine
/home/milovidov/ClickHouse/ci/workspace/gcc/gcc-build/x86_64-pc-linux-gnu/libstdc++-
v3/include/bits/unique_ptr.h:81
start_thread

__clone
```

## サーバ構成パラメータ

ここで記述が含まれてサーバーの設定を変更することはできないのセッションor検索。

これらの設定は `config.xml` ClickHouseサーバー上のファイル。

その他の設定については、“[設定](#)” セクション

設定を勉強する前に、[設定ファイル](#) セクションと置換の使用に注意してください( `incl` と `optional` 属性)。

## サーバー設定

### `builtin_dictionaries_reload_interval`

組み込み辞書を再ロードするまでの秒単位の間隔。

ClickHouseはx秒ごとに組み込みの辞書を再読み込みします。これにより、辞書の編集が可能になります “on the fly” サーバーを再起動せずに。

デフォルト値は3600です。

例

```
<builtin_dictionaries_reload_interval>3600</builtin_dictionaries_reload_interval>
```

## 圧縮

データ圧縮の設定 [メルゲツリー](#)-エンジンテーブル。

### 警告

ClickHouseの使用を開始したばかりの場合は、使用しないでください。

構成テンプレート：

```
<compression>
  <case>
    <min_part_size>...</min_part_size>
    <min_part_size_ratio>...</min_part_size_ratio>
    <method>...</method>
  </case>
  ...
</compression>
```

`<case>` フィールド：

- `min_part_size` – The minimum size of a data part.
- `min_part_size_ratio` – The ratio of the data part size to the table size.
- `method` – Compression method. Acceptable values: `lz4` または `zstd`.

複数の設定が可能です `<case>` セクション

条件を満たした場合のアクション:

- データ部分が条件セットに一致する場合、ClickHouseは指定された圧縮方法を使用します。
- データパートが複数の条件セットに一致する場合、ClickHouseは最初に一致した条件セットを使用します。

データパートの条件が満たされていない場合、ClickHouseは `lz4` 圧縮。

例

```
<compression incl="clickhouse_compression">
  <case>
    <min_part_size>10000000000</min_part_size>
    <min_part_size_ratio>0.01</min_part_size_ratio>
    <method>zstd</method>
  </case>
</compression>
```

## default\_database

既定のデータベース。

データベースのリストを取得するには、**SHOW DATABASES** クエリ。

例

```
<default_database>default</default_database>
```

## default\_profile

既定の設定プロファイル。

設定プロファイル内のファイルで指定されたパラメータ `user_config`.

例

```
<default_profile>default</default_profile>
```

## dictionaries\_config

外部辞書の設定ファイルへのパス。

パス:

- 絶対パスまたはサーバー設定ファイルに対する相対パスを指定します。
- のパスを含むことができワイルドカード\*や?.

も参照。“**外部辞書**”。

例

```
<dictionaries_config>*_dictionary.xml</dictionaries_config>
```

## dictionaries\_lazy\_load

辞書の遅延読み込み。

もし `true` その後、各辞書は、最初の使用時に作成されます。 辞書の作成に失敗した場合、辞書を使用していた関数は例外をスローします。

もし `false` すべての辞書は、サーバーの起動時に作成され、エラーがある場合は、サーバーがシャットダウンされます。

既定値は次のとおりです `true`.

例

```
<dictionaries_lazy_load>true</dictionaries_lazy_load>
```

## format\_schema\_path

入力データのスキーマを含むディレクトリへのパス。`CapnProto` 形式。

例

```
<!-- Directory containing schema files for various input formats. -->
<format_schema_path>format_schemas/</format_schema_path>
```

## 黒鉛

データの送信先 `黒鉛`.

設定:

- host - The Graphite server.
- port - The port on the Graphite server.
- interval - The interval for sending, in seconds.
- timeout - The timeout for sending data, in seconds.
- root\_path - Prefix for keys.
- metrics - Sending data from the システムメトリック テーブル。
- events - Sending deltas data accumulated for the time period from the システムイベント テーブル。
- events\_cumulative - Sending cumulative data from the システムイベント テーブル。
- asynchronous\_metrics - Sending data from the システムasynchronous\_metrics テーブル。

複数の設定が可能です `<graphite>` 句。 たとえば、異なる間隔で異なるデータを送信するためにこれを使用できます。

例

```
<graphite>
  <host>localhost</host>
  <port>42000</port>
  <timeout>0.1</timeout>
  <interval>60</interval>
  <root_path>one_min</root_path>
  <metrics>true</metrics>
  <events>true</events>
  <events_cumulative>false</events_cumulative>
  <asynchronous_metrics>true</asynchronous_metrics>
</graphite>
```

## graphite\_rollup

グラフファイルのデータを薄くする設定。

詳細は、を参照してください [GraphiteMergeTree](#).

例

```
<graphite_rollup_example>
  <default>
    <function>max</function>
    <retention>
      <age>0</age>
      <precision>60</precision>
    </retention>
    <retention>
      <age>3600</age>
      <precision>300</precision>
    </retention>
    <retention>
      <age>86400</age>
      <precision>3600</precision>
    </retention>
  </default>
</graphite_rollup_example>
```

## http\_port/https\_port

HTTP経由でサーバーに接続するためのポート。

もし `https_port` 指定される, [openSSL](#) 設定する必要があります。

もし `http_port` が指定されている場合、OpenSSL設定が設定されていても無視されます。

例

```
<https_port>9999</https_port>
```

## http\_server\_default\_response

ClickHouse HTTP(s)サーバーにアクセスするときにデフォルトで表示されるページ。

既定値は次のとおりです “Ok.” (最後に改行があります)

例

開く `https://tabix.io/` アクセス時 `http://localhost: http_port`.

```
<http_server_default_response>
<![CDATA[<html ng-app="SMI2"><head><base href="http://ui.tabix.io/"></head><body><div ui-view="" class="content-ui"></div><script src="http://loader.tabix.io/master.js"></script></body></html>]]>
</http_server_default_response>
```

## include\_from

置換されたファイルへのパス。

詳細については “[設定ファイル](#)”。

例

```
<include_from>/etc/metrica.xml</include_from>
```

## interserver\_http\_port

ClickHouseサーバー間でデータを交換するポート。

例

```
<interserver_http_port>9009</interserver_http_port>
```

## interserver\_http\_host

このサーバーへのアクセスに他のサーバーが使用できるホスト名。

省略された場合、これは `hostname -f` コマンド

特定のネット

例

```
<interserver_http_host>example.yandex.ru</interserver_http_host>
```

## interserver\_http\_credentials

認証時に使用されるユーザー名とパスワード [複製](#) 複製された\*エンジンで。これらの資格情報は、レプリカ間の通信にのみ使用され、ClickHouseクライアントの資格情報とは無関係です。サーバーにあるチェックにこれらの資格の接続にはレプリカと同じ資格を接続する場合はその他のレプリカ。なので、これらの資格を設定する同じすべてのレプリカ、クラスター

既定では、認証は使用されません。

このセクションでは以下のパラメータ:

- `user` — username.
- `password` — password.

例

```
<interserver_http_credentials>
  <user>admin</user>
  <password>222</password>
</interserver_http_credentials>
```

## keep\_alive\_timeout

ClickHouseが接続を閉じる前に受信要求を待機する秒数。既定値は10秒です。

例

```
<keep_alive_timeout>10</keep_alive_timeout>
```

## listen\_host

要求元のホストに対する制限。したい場合はサーバーの回答をしているが、それらを指定し:::

例:

```
<listen_host>::1</listen_host>
<listen_host>127.0.0.1</listen_host>
```

## ロガ一

ログ設定。

キ一:

- level – Logging level. Acceptable values: trace, debug, information, warning, error.
- log – The log file. Contains all the entries according to level.
- errorlog – Error log file.
- size – Size of the file. Applies to logとerrorlog. ファイルが到達すると size、ClickHouseはアーカイブし、その名前を変更し、その場所に新しいログファイルを作成します。
- count – The number of archived log files that ClickHouse stores.

例

```
<logger>
  <level>trace</level>
  <log>/var/log/clickhouse-server/clickhouse-server.log</log>
  <errorlog>/var/log/clickhouse-server/clickhouse-server.err.log</errorlog>
  <size>1000M</size>
  <count>10</count>
</logger>
```

Syslogへの書き込みもサポートされています。設定の例:

```
<logger>
  <use_syslog>1</use_syslog>
  <syslog>
    <address>syslog.remote:10514</address>
    <hostname>myhost.local</hostname>
    <facility>LOG_LOCAL6</facility>
    <format>syslog</format>
  </syslog>
</logger>
```

キ一:

- use\_syslog — Required setting if you want to write to the syslog.

- address — The host[:port] of syslogd. If omitted, the local daemon is used.
- hostname — Optional. The name of the host that logs are sent from.
- facility — **Syslog機能キーワード** 大文字では “LOG\_” 接頭辞: (LOG\_USER, LOG\_DAEMON, LOG\_LOCAL3、というように)。  
デフォルト値: `LOG_USER` もし `address` 指定される, `LOG_DAEMON` otherwise.
- format - Message format. Possible values: `bsd` と `syslog`.

## マクロ

複製されたテーブルのパラメーター置換。

ければ省略することができ複製のテーブルは使用しておりません。

詳細については “[複製テーブルの作成](#)”.

例

```
<macros incl="macros" optional="true" />
```

## mark\_cache\_size

テーブルエンジンが使用するマークのキャッシュのおおよそのサイズ(バイト単位) **メルゲツリー** 家族だ

キャッシュの共有のサーバーメモリが割り当てられます。キャッシュサイズは、少なくとも5368709120である必要が  
あります。

例

```
<mark_cache_size>5368709120</mark_cache_size>
```

## max\_concurrent\_queries

同時に処理される要求の最大数。

例

```
<max_concurrent_queries>100</max_concurrent_queries>
```

## max\_connections

受信接続の最大数。

例

```
<max_connections>4096</max_connections>
```

## max\_open\_files

開いているファイルの最大数。

既定では: `maximum`.

このオプションをMac OS Xで使用することをお勧めします。`getrlimit()` 関数は、誤った値を返します。

例

```
<max_open_files>262144</max_open_files>
```

## max\_table\_size\_to\_drop

テーブルの削除に関する制限。

のサイズが メルゲツリー テーブル超過 `max_table_size_to_drop` (バイト単位)、ドロップクエリを使用して削除することはできません。

ClickHouseサーバーを再起動せずにテーブルを削除する必要がある場合は、`<clickhouse-path>/flags/force_drop_table` DROPクエリをファイルして実行します。

デフォルト値:50GB。

値0は、制限なしですべてのテーブルを削除できることを意味します。

例

```
<max_table_size_to_drop>0</max_table_size_to_drop>
```

## merge\_tree

テーブルの微調整 メルゲツリー。

詳細については、MergeTreeSettingsを参照してください。hヘッダファイル。

例

```
<merge_tree>
  <max_suspicious_broken_parts>5</max_suspicious_broken_parts>
</merge_tree>
```

## openSSL

SSLクライアント/サーバー構成。

SSLのサポートは以下によって提供されます libpoco 図書館 ユーザーインターフェイスはファイルに記述 `SSLManager.h`

サーバー/クライアント設定のキー:

- `privateKeyFile` – The path to the file with the secret key of the PEM certificate. The file may contain a key and certificate at the same time.
- `certificateFile` – The path to the client/server certificate file in PEM format. You can omit it if `privateKeyFile` 証明書が含まれています。
- `caConfig` – The path to the file or directory that contains trusted root certificates.
- `verificationMode` – The method for checking the node's certificates. Details are in the description of the 文脈 クラス 可能な値: `none`, `relaxed`, `strict`, `once`.
- `verificationDepth` – The maximum length of the verification chain. Verification will fail if the certificate chain length exceeds the set value.

- loadDefaultCAFile – Indicates that built-in CA certificates for OpenSSL will be used. Acceptable values: true, false. |
- cipherList – Supported OpenSSL encryptions. For example: ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH.
- cacheSessions – Enables or disables caching sessions. Must be used in combination with sessionIdContext.  
許容値: true, false.
- sessionIdContext – A unique set of random characters that the server appends to each generated identifier. The length of the string must not exceed SSL\_MAX\_SSL\_SESSION\_ID\_LENGTH. このパラメータは常にお勧めです問題を避けることになるだろう両方の場合はサーバのキャッシュのセッションがクライアントの要望はキャッシュ. デフォルト値: \${application.name}.
- sessionCacheSize – The maximum number of sessions that the server caches. Default value: 1024\*20. 0 – Unlimited sessions.
- sessionTimeout – Time for caching the session on the server.
- extendedVerification – Automatically extended verification of certificates after the session ends. Acceptable values: true, false.
- requireTLSv1 – Require a TLSv1 connection. Acceptable values: true, false.
- requireTLSv1\_1 – Require a TLSv1.1 connection. Acceptable values: true, false.
- requireTLSv1\_2 – Require a TLSv1.2 connection. Acceptable values: true, false.
- fips – Activates OpenSSL FIPS mode. Supported if the library's OpenSSL version supports FIPS.
- privateKeyPassphraseHandler – Class (PrivateKeyPassphraseHandler subclass) that requests the passphrase for accessing the private key. For example: <privateKeyPassphraseHandler>, <name>KeyFileHandler</name>, <options><password>test</password></options>, </privateKeyPassphraseHandler>.
- invalidCertificateHandler – Class (a subclass of CertificateHandler) for verifying invalid certificates. For example: <invalidCertificateHandler> <name>ConsoleCertificateHandler</name> </invalidCertificateHandler> .
- disableProtocols – Protocols that are not allowed to use.
- preferServerCiphers – Preferred server ciphers on the client.

設定例:

```

<openSSL>
  <server>
    <!-- openssl req -subj "/CN=localhost" -new -newkey rsa:2048 -days 365 -nodes -x509 -keyout /etc/clickhouse-server/server.key -out /etc/clickhouse-server/server.crt -->
    <certificateFile>/etc/clickhouse-server/server.crt</certificateFile>
    <privateKeyFile>/etc/clickhouse-server/server.key</privateKeyFile>
    <!-- openssl dhparam -out /etc/clickhouse-server/dhparam.pem 4096 -->
    <dhParamsFile>/etc/clickhouse-server/dhparam.pem</dhParamsFile>
    <verificationMode>none</verificationMode>
    <loadDefaultCAFile>true</loadDefaultCAFile>
    <cacheSessions>true</cacheSessions>
    <disableProtocols>sslv2,sslv3</disableProtocols>
    <preferServerCiphers>true</preferServerCiphers>
  </server>
  <client>
    <loadDefaultCAFile>true</loadDefaultCAFile>
    <cacheSessions>true</cacheSessions>
    <disableProtocols>sslv2,sslv3</disableProtocols>
    <preferServerCiphers>true</preferServerCiphers>
    <!-- Use for self-signed: <verificationMode>none</verificationMode> -->
    <invalidCertificateHandler>
      <!-- Use for self-signed: <name>AcceptCertificateHandler</name> -->
      <name>RejectCertificateHandler</name>
    </invalidCertificateHandler>
  </client>
</openSSL>

```

## part\_log

関連付けられたイベントのログ記録 メルゲツリー。たとえば、データの追加やマージなどです。利用できるログを統合アルゴリズムと比較しています。マージプロセスを視覚化できます。

クエリは システム part\_log 別のファイルではなく、テーブル。このテーブルの名前は、table パラメータ(下記参照)。

以下のパラメータの設定ロギング:

- database – Name of the database.
- table – Name of the system table.
- partition\_by – Sets a カスタム分割キー.
- flush\_interval\_milliseconds – Interval for flushing data from the buffer in memory to the table.

例

```

<part_log>
  <database>system</database>
  <table>part_log</table>
  <partition_by>toMonday(event_date)</partition_by>
  <flush_interval_milliseconds>7500</flush_interval_milliseconds>
</part_log>

```

パス

データを含むディレクトリへのパス。

### 注

末尾のスラッシュは必須です。

例

```
<path>/var/lib/clickhouse/</path>
```

## プロメテウス

スクレイピングの指標データの公開 プロメテウス。

設定:

- `endpoint` – HTTP endpoint for scraping metrics by prometheus server. Start from '/'.
- `port` – Port for `endpoint`.
- `metrics` – Flag that sets to expose metrics from the システムメトリック テーブル。
- `events` – Flag that sets to expose metrics from the システムイベント テーブル。
- `asynchronous_metrics` – Flag that sets to expose current metrics values from the システム asynchronous\_metrics テーブル。

例

```
<prometheus>
  <endpoint>/metrics</endpoint>
  <port>8001</port>
  <metrics>true</metrics>
  <events>true</events>
  <asynchronous_metrics>true</asynchronous_metrics>
</prometheus>
```

## query\_log

クエリをログに記録するための設定 `log_queries=1` 設定。

クエリは システム `query_log` 別のファイルではなく、テーブル。 テーブルの名前を変更することができます。`table` パラメータ(下記参照)。

以下のパラメータの設定ロギング:

- `database` – Name of the database.
- `table` – Name of the system table the queries will be logged in.
- `partition_by` – Sets a カスタム分割キー テーブルのために。
- `flush_interval_milliseconds` – Interval for flushing data from the buffer in memory to the table.

テーブルが存在しない場合、ClickHouseはそれを作成します。 ClickHouseサーバーが更新されたときにクエリログの構造が変更された場合、古い構造のテーブルの名前が変更され、新しいテーブルが自動的に作成されます。

例

```
<query_log>
  <database>system</database>
  <table>query_log</table>
  <partition_by>toMonday(event_date)</partition_by>
  <flush_interval_milliseconds>7500</flush_interval_milliseconds>
</query_log>
```

## query\_thread\_log

クエリのスレッドをログに記録するための設定 `log_query_threads=1` 設定。

クエリは システムquery\_thread\_log 別のファイルではなく、テーブル。 テーブルの名前を変更することができます。  
table パラメータ(下記参照)。

以下のパラメータの設定ロギング:

- database – Name of the database.
- table – Name of the system table the queries will be logged in.
- partition\_by – Sets a カスタム分割キー システムテーブルの場合。
- flush\_interval\_milliseconds – Interval for flushing data from the buffer in memory to the table.

テーブルが存在しない場合、ClickHouseはそれを作成します。 ClickHouseサーバーの更新時にクエリスレッドログの構造が変更された場合、古い構造のテーブルの名前が変更され、新しいテーブルが自動的に作成されます。

例

```
<query_thread_log>
  <database>system</database>
  <table>query_thread_log</table>
  <partition_by>toMonday(event_date)</partition_by>
  <flush_interval_milliseconds>7500</flush_interval_milliseconds>
</query_thread_log>
```

## trace\_log

の設定 trace\_log システムテーブル操作。

パラメータ:

- database — Database for storing a table.
- table — Table name.
- partition\_by — カスタム分割キー システムテーブルの場合。
- flush\_interval\_milliseconds — Interval for flushing data from the buffer in memory to the table.

既定のサーバー構成ファイル config.xml 次の設定セクションがあります:

```
<trace_log>
  <database>system</database>
  <table>trace_log</table>
  <partition_by>toYYYYMM(event_date)</partition_by>
  <flush_interval_milliseconds>7500</flush_interval_milliseconds>
</trace_log>
```

## query\_masking\_rules

サーバーログに格納する前に、すべてのログメッセージだけでなく、クエリにも適用されます,  
system.query\_log, system.text\_log, system.processes クライアントに送信されたログ。 これにより  
SQLクエリからの機密データ漏洩(名前、電子メール、個人など  
ログへの識別子またはクレジットカード番号)。

例

```
<query_masking_rules>
  <rule>
    <name>hide SSN</name>
    <regexp>(^|\D)\d{3}-\d{2}-\d{4}($|\D)</regexp>
    <replace>000-00-0000</replace>
  </rule>
</query_masking_rules>
```

設定フィールド:

- **name** - ルールの名前(オプション)
- **regexp** - RE2互換の正規表現(必須)
- **replace** - 機密データの置換文字列(オプション、デフォルトでは-六つのアスタリスク)

マスキングルールは、クエリ全体に適用されます(不正な形式/解析不可能なクエリから機密データが漏れるのを防ぐため)。

`system.events` テーブルカウンター `QueryMaskingRulesMatch` して全体のマスキングルール。

分散クエリの場合は、各サーバーを個別に構成する必要があります。

ノードはマスクなしで保存されます。

## remote\_servers

によって使用されるクラスタの構成 **分散** テーブルエンジンと `cluster` テーブル関数。

例

```
<remote_servers incl="clickhouse_remote_servers" />
```

の値に対して `incl` 属性は、節を参照してください “[設定ファイル](#)”.

も参照。

- [skip\\_unavailable\\_shards](#)

## タイムゾーン

サーバーのタイムゾーン。

UTCタイムゾーンまたは地理的位置(たとえば、Africa/Abidjan)のIANA識別子として指定します。

タイムゾーンは、`DateTime`フィールドをテキスト形式(画面またはファイルに出力)に出力するとき、および文字列から`DateTime`を取得するときに、文字列と`DateTime`形式の間 また、タイムゾーンは、入力パラメーターでタイムゾーンを受信しなかった場合、時刻と日付を扱う関数で使用されます。

例

```
<timezone>Europe/Moscow</timezone>
```

## tcp\_port

TCPプロトコル経由でクライアントと通信するポート。

例

```
<tcp_port>9000</tcp_port>
```

## tcp\_port\_secure

クライアントとの安全な通信用のTCPポート。それを使用して OpenSSL 設定。

可能な値

正の整数。

デフォルト値

```
<tcp_port_secure>9440</tcp_port_secure>
```

## mysql\_port

MySQLプロトコ

可能な値

正の整数。

例

```
<mysql_port>9004</mysql_port>
```

## tmp\_path

大規模なクエリを処理するための一時データへのパス。

### 注

末尾のスラッシュは必須です。

例

```
<tmp_path>/var/lib/clickhouse/tmp/</tmp_path>
```

## tmp\_policy

ポリシーから storage\_configuration 一時ファイルを格納する。  
設定されていない場合 tmp\_path それ以外の場合は無視されます。

### 注

■ move\_factor 無視される

■ keep\_free\_space\_bytes 無視される

■ max\_data\_part\_size\_bytes 無視される  
なければならない同一数量の政策

## uncompressed\_cache\_size

テーブルエンジンが使用する非圧縮データのキャッシュサイズ(バイト単位) メルゲツリー。

サーバーの共有キャッシュが一つあります。メモリが割り当てられます。キャッシュが使用されるのは **use\_uncompressed\_cache** 有効です。

非圧縮キャッシュは、個々のケースで非常に短いクエリで有利です。

例

```
<uncompressed_cache_size>8589934592</uncompressed_cache_size>
```

## user\_files\_path

ユーテーブル関数で使用されます **ファイル()**.

例

```
<user_files_path>/var/lib/clickhouse/user_files/</user_files_path>
```

## users\_config

以下のファイルへのパス:

- ユーザー構成。
- アクセス権。
- 設定プロファイル。
- クオータ設定。

例

```
<users_config>users.xml</users_config>
```

## 飼育係

ClickHouseと対話できるようにする設定が含まれています。**飼育係** クラスター。

ClickHouse用飼育係の保存メタデータのレプリカの使用時に再現します。場合は複製のテーブルを使用していないので、このパラメータを省略することができます。

このセクションでは以下のパラメータ:

- **node** — ZooKeeper endpoint. You can set multiple endpoints.

例えば:

```
<node index="1">
  <host>example_host</host>
  <port>2181</port>
</node>
```

The `index` attribute specifies the node order when trying to connect to the ZooKeeper cluster.

- **session\_timeout** — Maximum timeout for the client session in milliseconds.
- **root** — The **znode** これはClickHouseサーバーで使用されるznodeのルートとして使用されます。任意。

- **identity** — User and password, that can be required by ZooKeeper to give access to requested znodes. Optional.

#### 設定例

```
<zookeeper>
  <node>
    <host>example1</host>
    <port>2181</port>
  </node>
  <node>
    <host>example2</host>
    <port>2181</port>
  </node>
  <session_timeout_ms>30000</session_timeout_ms>
  <operation_timeout_ms>10000</operation_timeout_ms>
  <!-- Optional. Chroot suffix. Should exist. -->
  <root>/path/to/zookeeper/node</root>
  <!-- Optional. Zookeeper digest ACL string. -->
  <identity>user:password</identity>
</zookeeper>
```

も参照。

- [複製](#)
- [ZooKeeperプログラマガイド](#)

## use\_minimalistic\_part\_header\_in\_zookeeper

ZooKeeperのデータ部分ヘッダーの格納方法。

この設定は、[MergeTree](#) 家族だ 指定できます:

- グローバルに [merge\\_tree](#) のセクション `config.xml` ファイル

ClickHouseは、サーバー上のすべてのテーブルの設定を使用します。 設定はいつでも変更できます。 既存のテーブルは、設定が変更されると動作を変更します。

- 各テーブルのため。

テーブルを作成するときは、対応する [エンジン設定](#)。 この設定を持つ既存のテーブルの動作は、グローバル設定が変更されても変更されません。

#### 可能な値

- 0 — Functionality is turned off.
- 1 — Functionality is turned on.

もし `use_minimalistic_part_header_in_zookeeper = 1` その後 [複製](#) テーブルは、単一のデータパーティのヘッダーをコンパクトに格納します `znode`。 テーブルに多数の列が含まれている場合、この格納方法はZooKeeperに格納されるデータの量を大幅に削減します。

## 注意

申請後 `use_minimalistic_part_header_in_zookeeper = 1` ClickHouseサーバーをこの設定をサポートしないバージョンにダウングレードすることはできません。 するとアップグレード時に注意ClickHouseサーバーにクラスターなアップの全てのサーバーです。 ClickHouseの新しいバージョンをテストするには、テスト環境またはクラスターの少数のサーバーでテストする方が安全です。

Data part headers already stored with this setting can't be restored to their previous (non-compact) representation.

デフォルト値: 0.

## disable\_internal\_dns\_cache

内部DNSキャッシュを無効にします。システムの作動のClickHouseのために推薦される頻繁に変化するインフラなどのKubernetes。

デフォルト値: 0.

## dns\_cache\_update\_period

ClickHouse内部DNSキャッシュに格納されているIPアドレスの更新期間(秒単位)。

更新は、別のシステムスレッドで非同期に実行されます。

デフォルト値: 15.

## access\_control\_path

パフォルダがClickHouseサーバー店舗ユーザーの役割構成で作成したSQLコマンド。

デフォルト値: /var/lib/clickhouse/access/.

も参照。

- アクセス制御とアカウント管理

## ClickHouseでハードウェアをテストする方法

この命令を実行できますが基本的なClickHouse性能試験はサーバーなしでの設置ClickHouseパッケージ。

1. に行く “commits” ページ:<https://github.com/ClickHouse/ClickHouse/commits/master>
2. 最初の緑色のチェックマークまたは緑色の赤十字をクリックします “ClickHouse Build Check” をクリックして “Details” 近くのリンク “ClickHouse Build Check”. いくつかのコミットにはそのようなリンクはありません。この場合、このリンクを持つ最も近いコミットを選択します。
3. リンクをコピーする “clickhouse” amd64 またはaarch64のバイナリ。
4. サーバーにsshし、wgetでダウンロードします:

```
# For amd64:  
wget  
https://clickhouse-builds.s3.yandex.net/0/00ba767f5d2a929394ea3be193b1f79074a1c4bc/1578163263_binary/clickhouse  
# For aarch64:  
wget  
https://clickhouse-builds.s3.yandex.net/0/00ba767f5d2a929394ea3be193b1f79074a1c4bc/1578161264_binary/clickhouse  
# Then do:  
chmod a+x clickhouse
```

1. ダウンロードconfigs:

```
 wget https://raw.githubusercontent.com/ClickHouse/ClickHouse/master/programs/server/config.xml  
 wget https://raw.githubusercontent.com/ClickHouse/ClickHouse/master/programs/server/users.xml  
 mkdir config.d  
 wget https://raw.githubusercontent.com/ClickHouse/ClickHouse/master/programs/server/config.d/path.xml -O config.d/path.xml  
 wget https://raw.githubusercontent.com/ClickHouse/ClickHouse/master/programs/server/config.d/log_to_console.xml -O config.d/log_to_console.xml
```

#### 1. ダウンロードファイルのベンチマーク:

```
 wget https://raw.githubusercontent.com/ClickHouse/ClickHouse/master/benchmark/clickhouse/benchmark-new.sh  
 chmod a+x benchmark-new.sh  
 wget https://raw.githubusercontent.com/ClickHouse/ClickHouse/master/benchmark/clickhouse/queries.sql
```

#### 1. に従うダウンロードテストデータ **Yandex.メトリカデータセット** 命令 ("hits" 100万行を含むテーブル)。

```
 wget https://datasets.clickhouse.com/hits/partitions/hits_100m_obfuscated_v1.tar.xz  
 tar xvf hits_100m_obfuscated_v1.tar.xz -C .  
 mv hits_100m_obfuscated_v1/* .
```

#### 1. サーバーの実行:

```
 ./clickhouse server
```

#### 1. データを確認する：別の端末のサーバーへのssh

```
 ./clickhouse client --query "SELECT count() FROM hits_100m_obfuscated"  
 100000000
```

#### 1. 編集benchmark-new.sh, 変更 clickhouse-client に ./clickhouse client と追加 --max\_memory\_usage 100000000000 パラメータ。

```
mcedit benchmark-new.sh
```

#### 1. ベンチマークの実行:

```
 ./benchmark-new.sh hits_100m_obfuscated
```

#### 1. ハードウェア構成に関する番号と情報を次の宛先に送信します **clickhouse-feedback@yandex-team.com**

すべての結果をこちらに発表します:<https://clickhouse.技術/基準/ハードウェア/>

## 設定

複数あるというものですすべての設定は、このセクションで説明する文書

設定はレイヤーで構成されるため、後続の各レイヤーは以前の設定を再定義します。

優先順位の順に設定する方法:

- の設定 `users.xml` サーバー構成ファイル。

要素に設定 `<profiles>`.

- セッション設定。

送信 `SET setting=value` 対話モードでClickHouseコンソールクライアントから。

同様に、HttpプロトコルでClickHouseセッションを使用できます。これを行うには、以下を指定する必要があります `session_id` HTTPパラメータ。

- クエリ設定。

- ClickHouse console クライアントを非対話モードで起動するときは、`startup`パラメータを設定します `--setting=value`.
- HTTP APIを使用する場合は、CGIパラメータを渡します (`URL?setting_1=value&setting_2=value...`).

サーバー設定ファイルでのみ行うことができる設定は、このセクションでは説明しません。

## クエリの権限

問合せClickHouse大きく分けて複数の種類:

1. データを読み込むためのクエリー: `SELECT`, `SHOW`, `DESCRIBE`, `EXISTS`.
2. データクエリの記述: `INSERT`, `OPTIMIZE`.
3. 設定の変更クエリ: `SET`, `USE`.
4. **DDL** クエリ: `CREATE`, `ALTER`, `RENAME`, `ATTACH`, `DETACH`, `DROP` `TRUNCATE`.
5. `KILL QUERY`.

次の設定では、クエリの種類によってユーザー権限を調整します:

- **読み取り専用** — Restricts permissions for all types of queries except DDL queries.
- **allow\_ddl** — Restricts permissions for DDL queries.

`KILL QUERY` 任意の設定で実行できます。

### 読み取り専用

制限のアクセス権読書データの書き込みデータや設定の変更ます。

クエリを型に分割する方法を参照してください [上記](#).

可能な値:

- 0 — All queries are allowed.
- 1 — Only read data queries are allowed.
- 2 — Read data and change settings queries are allowed.

設定後 `readonly = 1` ユーザーは変更できません `readonly` と `allow_ddl` 現在のセッションの設定。

を使用する場合 `GET` の方法 **HTTPインターフェ**, `readonly = 1` 自動的に設定されます。データを変更するには、`POST` 方法。

設定 `readonly = 1` ユーザーがすべての設定を変更するのを禁止します。ユーザーを禁止する方法がありますからの変更は、特定の設定の詳細を見る [設定の制約](#).

デフォルト値:0

### allow\_ddl

許可または拒否 **DDL** クエリ。

クエリを型に分割する方法を参照してください [上記](#).

可能な値:

- 0 — DDL queries are not allowed.
- 1 — DDL queries are allowed.

実行できない `SET allow_ddl = 1` もし `allow_ddl = 0` 現在のセッションの場合。

デフォルト値:1

## クエリの複雑さの制限

クエリの複雑さに関する制限は、設定の一部です。

これらをより安全な実行のユーザーインターフェースです。

ほとんどすべての制限が適用されます `SELECT`. 分散クエリ処理では、各サーバーに個別に制限が適用されます。

ClickHouseは、各行ではなく、データ部分の制限をチェックします。これは、データ部分のサイズで制限の値を超えることができることを意味します。

の制限 “maximum amount of something” 値0を取ることができます。“unrestricted”.

ほとんどの制限には ‘overflow\_mode’ 設定、制限を超えたときに何をすべきかを意味します。

での値: `throw` または `break`. 集計の制限(`group_by_overflow_mode`)にも値があります `any`.

`throw` – Throw an exception (default).

`break` – Stop executing the query and return the partial result, as if the source data ran out.

`any` (only for `group_by_overflow_mode`) – Continuing aggregation for the keys that got into the set, but don't add new keys to the set.

## max\_memory\_usage

単一サーバーでクエリを実行するために使用するRAMの最大量。

既定の構成ファイルでは、最大10GBです。

この設定では、使用可能なメモリの容量やマシン上のメモリの合計容量は考慮されません。

この制限は、単一サーバー内の単一のクエリに適用されます。

以下を使用できます `SHOW PROCESSLIST` 各クエリの現在のメモリ消費量を確認します。

さらに、ピークメモリ消費は各クエリに対して追跡され、ログに書き込まれます。

特定の集計関数の状態については、メモリ使用量は監視されません。

集計関数の状態に対してメモリ使用量が完全に追跡されません `min`, `max`, `any`, `anyLast`, `argMin`, `argMax` から `String` と `Array` 引数。

メモリ消費もパラメータによって制限されます `max_memory_usage_for_user` と `max_memory_usage_for_all_queries`.

## max\_memory\_usage\_for\_user

単一サーバー上でユーザーのクエリを実行するために使用するRAMの最大量。

デフォルト値は [設定](#)。`h`. デフォルトでは、金額は制限されません (`max_memory_usage_for_user = 0`).

の説明も参照してください [max\\_memory\\_usage](#).

## max\_memory\_usage\_for\_all\_queries

单一路由器上ですべてのクエリを実行するために使用するRAMの最大量。

デフォルト値は [設定](#)。[h](#). デフォルトでは、金額は制限されません (`max_memory_usage_for_all_queries = 0`). の説明も参照してください [max\\_memory\\_usage](#).

## max\_rows\_to\_read

次の制限は、各ロック（各行ではなく）で確認できます。つまり、制限は少し壊れる可能性があります。複数のスレッドでクエリを実行する場合、次の制限は各スレッドに個別に適用されます。

クエリの実行時にテーブルから読み取ることができる最大行数。

## max\_bytes\_to\_read

クエリの実行時にテーブルから読み取ることができる最大バイト数(非圧縮データ)。

## read\_overflow\_mode

読み込まれるデータ量がいずれかの制限を超えた場合の対処方法: ‘throw’ または ‘break’. デフォルトでは、throw。

## max\_rows\_to\_group\_by

集計から受け取った一意のキーの最大数。この設定を使用すると、集計時のメモリ消費量を制限できます。

## group\_by\_overflow\_mode

集計の一意キーの数が制限を超えた場合の対処方法: ‘throw’, ‘break’, または ‘any’. デフォルトでは、throw。を使用して ‘any’ value では、GROUP BY の近似を実行できます。この近似の品質は、データの統計的性質に依存します。

## max\_bytes\_before\_external\_group\_by

の実行を有効または無効にします。 `GROUP BY` 外部メモリ内の句。見る [外部メモリのGROUP BY](#).

可能な値:

- シングルで使用できるRAMの最大ボリューム(バイト単位) [GROUP BY](#) 作戦だ
- 0 — `GROUP BY` 外部メモリでは無効です。

デフォルト値は0です。

## max\_rows\_to\_sort

並べ替え前の最大行数。これにより、ソート時のメモリ消費を制限できます。

## max\_bytes\_to\_sort

並べ替え前の最大バイト数。

## sort\_overflow\_mode

ソート前に受信した行数がいずれかの制限を超えた場合の対処方法: ‘throw’ または ‘break’. デフォルトでは、throw。

## max\_result\_rows

結果の行数を制限します。またチェックサブクエリは、windows アプリケーションの実行時にパートの分散を返します。

## max\_result\_bytes

結果のバイト数を制限します。前の設定と同じです。

## result\_overflow\_mode

結果の量が制限のいずれかを超えた場合の対処方法: 'throw' または 'break'. デフォルトでは、throw。

を使用して 'break' LIMIT の使用に似ています。Break ブロックレベルでのみ実行を中断します。これは、返される行の量が max\_result\_rows の倍数 max\_block\_size そして依存する max\_threads.

例:

```
SET max_threads = 3, max_block_size = 3333;
SET max_result_rows = 3334, result_overflow_mode = 'break';

SELECT *
FROM numbers_mt(100000)
FORMAT Null;
```

結果:

```
6666 rows in set. ...
```

## max\_execution\_time

クエリの最大実行時間を秒単位で指定します。

現時点では、ソートステージのいずれか、または集計関数のマージおよびファイナライズ時にはチェックされません。

## timeout\_overflow\_mode

クエリが実行される時間よりも長い場合の対処方法 'max\_execution\_time': 'throw' または 'break'. デフォルトでは、throw。

## min\_execution\_speed

毎秒行単位の最小実行速度。すべてのデータブロックで 'timeout\_before\_checking\_execution\_speed' 有効期限が切れます。実行速度が低い場合は、例外がスローされます。

## min\_execution\_speed\_bytes

秒あたりの最小実行バイト数。すべてのデータブロックで 'timeout\_before\_checking\_execution\_speed' 有効期限が切れます。実行速度が低い場合は、例外がスローされます。

## max\_execution\_speed

毎秒の実行行の最大数。すべてのデータブロックで 'timeout\_before\_checking\_execution\_speed' 有効期限が切れます。実行速度が高い場合は、実行速度が低下します。

## max\_execution\_speed\_bytes

毎秒の実行バイト数の最大値。すべてのデータブロックで 'timeout\_before\_checking\_execution\_speed' 有効期限が切れます。実行速度が高い場合は、実行速度が低下します。

## timeout\_before\_checking\_execution\_speed

実行速度が遅すぎないことをチェックします 'min\_execution\_speed' )、指定された時間が秒単位で経過した後。

## **max\_columns\_to\_read**

単一のクエリ内のテーブルから読み取ることができる列の最大数。 クエリでより多くの列を読み取る必要がある場合は、例外がスローされます。

## **max\_temporary\_columns**

定数列を含む、クエリを実行するときに同時にRAMに保持する必要がある一時列の最大数。 これよりも多くの一時列がある場合、例外がスローされます。

## **max\_temporary\_non\_const\_columns**

同じことと ‘max\_temporary\_columns’ しかし、定数列を数えずに。

定数列は、クエリを実行するときにかなり頻繁に形成されますが、計算リソースはほぼゼロです。

## **max\_subquery\_depth**

サブクエリの最大ネスト深さ。 サブクエリが深い場合は、例外がスローされます。 既定では100です。

## **max\_pipeline\_depth**

最大パイプライン深さ。 クエリ処理中に各データブロックが処理する変換の数に対応します。 単一サーバーの範囲内でのカウントされます。 パイプラインの深さが大きい場合は、例外がスローされます。 既定では、1000です。

## **max\_ast\_depth**

クエリ構文ツリーの最大ネスト深さ。 超過すると、例外がスローされます。

現時点では、解析中にはチェックされず、クエリの解析後にのみチェックされます。 つまり、解析中に深すぎる構文ツリーを作成することができますが、クエリは失敗します。 既定では、1000です。

## **max\_ast\_elements**

クエリ構文ツリー内の要素の最大数。 超過すると、例外がスローされます。

前の設定と同じように、クエリを解析した後にのみチェックされます。 既定では、50,000です。

## **max\_rows\_in\_set**

サブクエリから作成されたIN句内のデータ-セットの最大行数。

## **max\_bytes\_in\_set**

サブクエリから作成されたIN句のセットで使用される最大バイト数(非圧縮データ)。

## **set\_overflow\_mode**

データ量がいずれかの制限を超えた場合の対処方法: ‘throw’ または ‘break’. デフォルトでは、throw。

## **max\_rows\_in\_distinct**

DISTINCTを使用する場合の最大行数。

## **max\_bytes\_in\_distinct**

DISTINCTを使用するときにハッシュテーブルで使用される最大バイト数。

## **distinct\_overflow\_mode**

データ量がいずれかの制限を超えた場合の対処方法: ‘throw’ または ‘break’. デフォルトでは、throw。

## **max\_rows\_to\_transfer**

グローバルINを使用するときに、リモートサーバーに渡すか、一時テーブルに保存できる最大行数。

## **max\_bytes\_to\_transfer**

グローバルINを使用するときに、リモートサーバーに渡すか、一時テーブルに保存できる最大バイト数(非圧縮データ)。

## **transfer\_overflow\_mode**

データ量がいずれかの制限を超えた場合の対処方法: 'throw' または 'break'. デフォルトでは、throw。

## **max\_rows\_in\_join**

テーブルを結合するときに使用されるハッシュテーブル内の行数を制限します。

この設定は以下に適用されます **SELECT ... JOIN** 操作および 参加 テーブルエンジン。

クエリに複数の結合が含まれている場合、ClickHouseはこの設定で中間結果をすべてチェックします。

ClickHouseは、制限に達したときにさまざまなアクションを続行できます。 使用する **join\_overflow\_mode** アクションを選択する設定。

可能な値:

- 正の整数。
- 0 — Unlimited number of rows.

デフォルト値は0です。

## **max\_bytes\_in\_join**

制限サイズをバイトのハッシュテーブルが参加す。

この設定は以下に適用されます **SELECT ... JOIN** 操作および 結合テーブルエンジン。

クエリに結合が含まれている場合、ClickHouseは中間結果ごとにこの設定をチェックします。

ClickHouseは、制限に達したときにさまざまなアクションを続行できます。 使用 **join\_overflow\_mode** アクションを選択する設定。

可能な値:

- 正の整数。
- 0 — Memory control is disabled.

デフォルト値は0です。

## **join\_overflow\_mode**

次のいずれかの結合制限に達したときにClickHouseが実行するアクションを定義します:

- **max\_bytes\_in\_join**
- **max\_rows\_in\_join**

可能な値:

- **THROW** — ClickHouse throws an exception and breaks operation.

- **BREAK** — ClickHouse breaks operation and doesn't throw an exception.

デフォルト値: **THROW**.

も参照。

- **JOIN句**

- **結合テーブルエンジン**

## max\_partitions\_per\_insert\_block

单一挿入ブロック内のパーティションの最大数を制限します。

- 正の整数。
- 0 — Unlimited number of partitions.

デフォルト値は100です。

### 詳細

を挿入する際、データClickHouse計算パーティションの数に挿入されます。パーティションの数が **max\_partitions\_per\_insert\_block**, ClickHouseは、次のテキストで例外をスローします:

"Too many partitions for single INSERT block (more than " +toString(max\_parts)+ "). The limit is controlled by 'max\_partitions\_per\_insert\_block' setting. A large number of partitions is a common misconception. It will lead to severe negative performance impact, including slow server startup, slow INSERT queries and slow SELECT queries. Recommended total number of partitions for a table is under 1000..10000. Please note, that partitioning is not intended to speed up SELECT queries (ORDER BY key is sufficient to make range queries fast). Partitions are intended for data manipulation (DROP PARTITION, etc)."

## 設定プロファイル

設定プロファイルは、同じ名前でグループ化された設定の集合です。

### 情報

ClickHouseはまた支えます **SQL駆動型ワークフロー** 設定プロファイルを管理する。お勧めいたします。

プロファイルのどれでも持つことができます。プロファイルのどれでも持つ事ができます。異なるユーザーに同じプロファイルを指定できます。最も重要なことが書ける設定プロフィール **readonly=1** 読み取り専用アクセスを保証します。

設定プロファイルは相互に継承できます。継承を使用するには、一つまたは複数を指定します **profile** プロファイルリストされている他の設定の前の設定。ある設定が異なるプロファイルで定義されている場合は、定義された最新の設定が使用されます。

プロファイル内のすべての設定を適用するには、**profile** 設定。

例:

インストール **web** プロフィール

```
SET profile = 'web'
```

設定プロファイルで宣言されたユーザのconfigファイルです。これは通常です `users.xml`.

例:

```
<!-- Settings profiles -->
<profiles>
  <!-- Default settings -->
  <default>
    <!-- The maximum number of threads when running a single query. -->
    <max_threads>8</max_threads>
  </default>

  <!-- Settings for queries from the user interface -->
  <web>
    <max_rows_to_read>1000000000</max_rows_to_read>
    <max_bytes_to_read>1000000000000</max_bytes_to_read>

    <max_rows_to_group_by>1000000</max_rows_to_group_by>
    <group_by_overflow_mode>any</group_by_overflow_mode>

    <max_rows_to_sort>1000000</max_rows_to_sort>
    <max_bytes_to_sort>10000000000</max_bytes_to_sort>

    <max_result_rows>100000</max_result_rows>
    <max_result_bytes>1000000000</max_result_bytes>
    <result_overflow_mode>break</result_overflow_mode>

    <max_execution_time>600</max_execution_time>
    <min_execution_speed>1000000</min_execution_speed>
    <timeout_before_checking_execution_speed>15</timeout_before_checking_execution_speed>

    <max_columns_to_read>25</max_columns_to_read>
    <max_temporary_columns>100</max_temporary_columns>
    <max_temporary_non_const_columns>50</max_temporary_non_const_columns>

    <max_subquery_depth>2</max_subquery_depth>
    <max_pipeline_depth>25</max_pipeline_depth>
    <max_ast_depth>50</max_ast_depth>
    <max_ast_elements>100</max_ast_elements>

    <readonly>1</readonly>
  </web>
</profiles>
```

この例では、`: default` と `web`.

その `default` プロファイルには特別な目的があります。つまり、`default` オプションの設定デフォルトを設定します。

その `web` プロファイルは通常のプロファイルです。SET クエリまたはHTTPクエリでURLパラメータを使用する。

## 設定の制約

設定に関する制約は、以下で定義することができます。`profiles` のセクション `user.xml` 設定ファイルとユーザーが設定の一部を変更することを禁じます。SET クエリ。

制約は次のように定義されます:

```

<profiles>
  <user_name>
    <constraints>
      <setting_name_1>
        <min>lower_boundary</min>
      </setting_name_1>
      <setting_name_2>
        <max>upper_boundary</max>
      </setting_name_2>
      <setting_name_3>
        <min>lower_boundary</min>
        <max>upper_boundary</max>
      </setting_name_3>
      <setting_name_4>
        <readonly/>
      </setting_name_4>
    </constraints>
  </user_name>
</profiles>

```

ユーザーが制約に違反しようとすると、例外がスローされ、設定は変更されません。

サポートされている制約は以下の通りです: `min`, `max`, `readonly`. その `min` と `max` 制約は、数値設定の上限と下限を指定し、組み合わせて使用できます。その `readonly` 制約を指定すると、ユーザーは変更できませんので、対応する設定です。

例: さあ `users.xml` 行を含む:

```

<profiles>
  <default>
    <max_memory_usage>10000000000</max_memory_usage>
    <force_index_by_date>0</force_index_by_date>
    ...
    <constraints>
      <max_memory_usage>
        <min>5000000000</min>
        <max>20000000000</max>
      </max_memory_usage>
      <force_index_by_date>
        <readonly/>
      </force_index_by_date>
    </constraints>
  </default>
</profiles>

```

以下のクエリすべての例外をスロー:

```

SET max_memory_usage=20000000001;
SET max_memory_usage=4999999999;
SET force_index_by_date=1;

```

```

Code: 452, e.displayText() = DB::Exception: Setting max_memory_usage should not be greater than 20000000000.
Code: 452, e.displayText() = DB::Exception: Setting max_memory_usage should not be less than 5000000000.
Code: 452, e.displayText() = DB::Exception: Setting force_index_by_date should not be changed.

```

注: その `default` プロファイルには特別な処理があります。 `default` プロのデフォルトの制約、その制限するすべてのユーザーまでの彼らのメソッドを明示的にこれらのユーザー

## ユーザー設定

その `users` のセクション `user.xml` 設定ファイルにユーザを設定します。

## 情報

ClickHouseはまた支えます **SQL駆動型ワークフロー** ユーザーを管理するため。お勧めいたします。

の構造 users セクション：

```
<users>
  <!-- If user name was not specified, 'default' user is used. -->
  <user_name>
    <password></password>
    <!-- Or -->
    <password_sha256_hex></password_sha256_hex>

    <access_management>0|1</access_management>

    <networks incl="networks" replace="replace">
    </networks>

    <profile>profile_name</profile>

    <quota>default</quota>

    <databases>
      <database_name>
        <table_name>
          <filter>expression</filter>
        <table_name>
      </database_name>
    </databases>
  </user_name>
  <!-- Other users settings -->
</users>
```

### user\_name/パスワード

パスワードは、平文またはSHA256(hex形式)で指定できます。

- 平文でパスワードを割り当てるには(推奨されない)、それをaに置きます **password** 要素。

例えば、`<password>qwerty</password>`。パスワードは空白のままにできます。

- SHA256ハッシュを使用してパスワードを割り当てるには、パスワードを **password\_sha256\_hex** 要素。

例えば、

```
<password_sha256_hex>65e84be33532fb784c48129675f9eff3a682b27168c0ea744b2cf58ee02337c5</password_sha256_hex>
```

シェルからパスワードを生成する方法の例：

```
PASSWORD=$(base64 < /dev/urandom | head -c8); echo "$PASSWORD"; echo -n "$PASSWORD" | sha256sum | tr -d '-'
```

結果の最初の行はパスワードです。第二の行は、対応するSHA256ハッシュです。

- MySQLクライアントとの互換性のために、パスワードは二重SHA1ハッシュで指定できます。それを置く `password_double_sha1_hex` 要素。

例えば、

```
<password_double_sha1_hex>08b4a0f1de6ad37da17359e592c8d74788a83eb0</password_double_sha1_hex>.
```

シェルからパスワードを生成する方法の例:

```
PASSWORD=$(base64 < /dev/urandom | head -c8); echo "$PASSWORD"; echo -n "$PASSWORD" | sha1sum | tr -d '-' | xxd -r -p | sha1sum | tr -d '-'
```

結果の最初の行はパスワードです。第二の行は、対応する二重SHA1ハッシュです。

## access\_management

この設定では、SQLドリブンの使用を無効にできます [アクセス制御とアカウント管理](#) ユーザーのために。

可能な値:

- 0 — Disabled.
- 1 — Enabled.

デフォルト値は0です。

## user\_name/ネットワーク

ユーザーがClickHouseサーバーに接続できるネットワークのリスト。

リストの各要素には、次のいずれかの形式を使用できます:

- <ip> — IP address or network mask.

例: 213.180.204.3, 10.0.0.1/8, 10.0.0.1/255.255.255.0, 2a02:6b8::3, 2a02:6b8::3/64, 2a02:6b8::3/ffff:ffff:ffff:ffff::.

- <host> — Hostname.

例: example01.host.ru.

チェックアクセス、DNS問い合わせを行い、すべて返されたIPアドレスと比べてピアがアドレスです。

- <host\_regexp> — Regular expression for hostnames.

例, ^example\d\d-\d\d-\d\.\host\.ru\$

アクセスを確認するには、[DNS PTRクエリ](#) ピアアドレスに対して実行され、指定された正規表現が適用されます。次に、PTRクエリの結果に対して別のDNSクエリが実行され、受信したすべてのアドレスがピアアドレスと比較されます。正規表現は\$で終わることを強くお勧めします。

すべての結果のDNSの要求をキャッシュまでのサーバが再起動してしまいます。

### 例

オーブンアクセスのためのユーザーからネットワークのいずれかを指定し:

```
<ip>::/0</ip>
```

**警告**

この不安にオープンアクセスからネットワークを持っていない場合、ファイアウォールを適切に設定されたサーバーに直接接続されます。

オープンアクセスのみからlocalhostを指定し:

```
<ip>::1</ip>
<ip>127.0.0.1</ip>
```

## user\_name/プロファイル

を割り当てることができる設定プロファイルをユーザーです。 設定プロファイルは、`users.xml` ファイル 詳細については、[設定のプロファイル](#)。

## user\_name/クオータ

クオータを使用すると、一定期間のリソース使用量を追跡または制限できます。 クオータは `quotas` のセクション `users.xml` 設定ファイル。

ユーザにクオータセットを割り当てることができます。 クオータ設定の詳細については、以下を参照してください [クオータ](#)。

## user\_name/データベース

このセクションでは、ClickHouseによって返される行を以下の目的で制限することができます `SELECT` クエリーによる、現在のユーザが実施基本列レベルです。

### 例

以下の構成力がユーザー `user1` の行だけを見ることができます `table1` の結果として `SELECT` クエリの値は、次のとおりです。`id` フィールドは `1000` です。

```
<user1>
  <databases>
    <database_name>
      <table1>
        <filter>id = 1000</filter>
      </table1>
    </database_name>
  </databases>
</user1>
```

その `filter` 任意の式にすることができます。 `UInt8`-タイプ値。 通常、比較演算子と論理演算子が含まれています。 からの行 `database_name.table1` る結果をフィルターを `0` においても返却いたしませんこのユーザーです。 このフィルタリングは `PREWHERE` 操作と無効 `WHERE→PREWHERE` 最適化。

## MergeTree tables settings

The values of `merge_tree` settings (for all MergeTree tables) can be viewed in the table `system.merge_tree_settings`, they can be overridden in `config.xml` in the `merge_tree` section, or set in the `SETTINGS` section of each table.

Override example in `config.xml`:

```
<merge_tree>
  <max_suspicious_broken_parts>5</max_suspicious_broken_parts>
</merge_tree>
```

An example to set in `SETTINGS` for a particular table:

```
CREATE TABLE foo
(
    `A` Int64
)
ENGINE = MergeTree
ORDER BY tuple()
SETTINGS max_suspicious_broken_parts = 500;
```

An example of changing the settings for a specific table with the `ALTER TABLE ... MODIFY SETTING` command:

```
ALTER TABLE foo
MODIFY SETTING max_suspicious_broken_parts = 100;
```

## parts\_to\_throw\_insert

If the number of active parts in a single partition exceeds the `parts_to_throw_insert` value, `INSERT` is interrupted with the `Too many parts (N)`. Merges are processing significantly slower than `insertexception`.

Possible values:

- Any positive integer.

Default value: 300.

To achieve maximum performance of `SELECT` queries, it is necessary to minimize the number of parts processed, see [Merge Tree](#).

You can set a larger value to 600 (1200), this will reduce the probability of the `Too many parts` error, but at the same time `SELECT` performance might degrade. Also in case of a merge issue (for example, due to insufficient disk space) you will notice it later than it could be with the original 300.

## parts\_to\_delay\_insert

If the number of active parts in a single partition exceeds the `parts_to_delay_insert` value, an `INSERT` artificially slows down.

Possible values:

- Any positive integer.

Default value: 150.

ClickHouse artificially executes `INSERT` longer (adds 'sleep') so that the background merge process can merge parts faster than they are added.

## inactive\_parts\_to\_throw\_insert

If the number of inactive parts in a single partition more than the `inactive_parts_to_throw_insert` value, `INSERT` is interrupted with the "Too many inactive parts (N)". Parts cleaning are processing significantly slower than `inserts`" exception.

Possible values:

- Any positive integer.

Default value: 0 (unlimited).

## inactive\_parts\_to\_delay\_insert

If the number of inactive parts in a single partition in the table at least that many the `inactive_parts_to_delay_insert` value, an `INSERT` artificially slows down. It is useful when a server fails to clean up parts quickly enough.

Possible values:

- Any positive integer.

Default value: 0 (unlimited).

## max\_delay\_to\_insert

The value in seconds, which is used to calculate the `INSERT` delay, if the number of active parts in a single partition exceeds the `parts_to_delay_insert` value.

Possible values:

- Any positive integer.

Default value: 1.

The delay (in milliseconds) for `INSERT` is calculated by the formula:

```
max_k = parts_to_throw_insert - parts_to_delay_insert
k = 1 + parts_count_in_partition - parts_to_delay_insert
delay_milliseconds = pow(max_delay_to_insert * 1000, k / max_k)
```

For example if a partition has 299 active parts and `parts_to_throw_insert` = 300, `parts_to_delay_insert` = 150, `max_delay_to_insert` = 1, `INSERT` is delayed for `pow( 1 * 1000, (1 + 299 - 150) / (300 - 150) ) = 1000` milliseconds.

## max\_parts\_in\_total

If the total number of active parts in all partitions of a table exceeds the `max_parts_in_total` value `INSERT` is interrupted with the `Too many parts (N)` exception.

Possible values:

- Any positive integer.

Default value: 100000.

A large number of parts in a table reduces performance of ClickHouse queries and increases ClickHouse boot time. Most often this is a consequence of an incorrect design (mistakes when choosing a partitioning strategy - too small partitions).

## replicated\_deduplication\_window

The number of most recently inserted blocks for which Zookeeper stores hash sums to check for duplicates.

Possible values:

- Any positive integer.
- 0 (disable deduplication)

Default value: 100.

The `Insert` command creates one or more blocks (parts). When inserting into Replicated tables, ClickHouse for `insert deduplication` writes the hash sums of the created parts into Zookeeper. Hash sums are stored only for the most recent `replicated_deduplication_window` blocks. The oldest hash sums are removed from Zookeeper.

A large number of `replicated_deduplication_window` slows down `Inserts` because it needs to compare more entries.

The hash sum is calculated from the composition of the field names and types and the data of the inserted part (stream of bytes).

## `non_replicated_deduplication_window`

The number of the most recently inserted blocks in the non-replicated `MergeTree` table for which hash sums are stored to check for duplicates.

Possible values:

- Any positive integer.
- 0 (disable deduplication).

Default value: 0.

A deduplication mechanism is used, similar to replicated tables (see `replicated_deduplication_window` setting). The hash sums of the created parts are written to a local file on a disk.

## `replicated_deduplication_window_seconds`

The number of seconds after which the hash sums of the inserted blocks are removed from Zookeeper.

Possible values:

- Any positive integer.

Default value: 604800 (1 week).

Similar to `replicated_deduplication_window`, `replicated_deduplication_window_seconds` specifies how long to store hash sums of blocks for insert deduplication. Hash sums older than `replicated_deduplication_window_seconds` are removed from Zookeeper, even if they are less than `replicated_deduplication_window`.

## `replicated_fetches_http_connection_timeout`

HTTP connection timeout (in seconds) for part fetch requests. Inherited from default profile `http_connection_timeout` if not set explicitly.

Possible values:

- Any positive integer.
- 0 - Use value of `http_connection_timeout`.

Default value: 0.

## `replicated_fetches_http_send_timeout`

HTTP send timeout (in seconds) for part fetch requests. Inherited from default profile `http_send_timeout` if not set explicitly.

Possible values:

- Any positive integer.

- 0 - Use value of `http_send_timeout`.

Default value: 0.

## `replicated_fetches_http_receive_timeout`

HTTP receive timeout (in seconds) for fetch part requests. Inherited from default profile `http_receive_timeout` if not set explicitly.

Possible values:

- Any positive integer.
- 0 - Use value of `http_receive_timeout`.

Default value: 0.

## `max_replicated_fetches_network_bandwidth`

Limits the maximum speed of data exchange over the network in bytes per second for `replicated` fetches.

This setting is applied to a particular table, unlike the

`max_replicated_fetches_network_bandwidth_for_server` setting, which is applied to the server.

You can limit both server network and network for a particular table, but for this the value of the table-level setting should be less than server-level one. Otherwise the server considers only the `max_replicated_fetches_network_bandwidth_for_server` setting.

The setting isn't followed perfectly accurately.

Possible values:

- Positive integer.
- 0 — Unlimited.

Default value: 0.

### **Usage**

Could be used for throttling speed when replicating data to add or replace new nodes.

## `max_replicated_sends_network_bandwidth`

Limits the maximum speed of data exchange over the network in bytes per second for `replicated` sends.

This setting is applied to a particular table, unlike the

`max_replicated_sends_network_bandwidth_for_server` setting, which is applied to the server.

You can limit both server network and network for a particular table, but for this the value of the table-level setting should be less than server-level one. Otherwise the server considers only the `max_replicated_sends_network_bandwidth_for_server` setting.

The setting isn't followed perfectly accurately.

Possible values:

- Positive integer.
- 0 — Unlimited.

Default value: 0.

### **Usage**

Could be used for throttling speed when replicating data to add or replace new nodes.

## old\_parts\_lifetime

The time (in seconds) of storing inactive parts to protect against data loss during spontaneous server reboots.

Possible values:

- Any positive integer.

Default value: 480.

After merging several parts into a new part, ClickHouse marks the original parts as inactive and deletes them only after `old_parts_lifetime` seconds.

Inactive parts are removed if they are not used by current queries, i.e. if the `refcount` of the part is zero.

`fsync` is not called for new parts, so for some time new parts exist only in the server's RAM (OS cache). If the server is rebooted spontaneously, new parts can be lost or damaged.

To protect data inactive parts are not deleted immediately.

During startup ClickHouse checks the integrity of the parts.

If the merged part is damaged ClickHouse returns the inactive parts to the active list, and later merges them again. Then the damaged part is renamed (the `broken_` prefix is added) and moved to the `detached` folder.

If the merged part is not damaged, then the original inactive parts are renamed (the `ignored_` prefix is added) and moved to the `detached` folder.

The default `dirty_expire_centisecs` value (a Linux kernel setting) is 30 seconds (the maximum time that written data is stored only in RAM), but under heavy loads on the disk system data can be written much later. Experimentally, a value of 480 seconds was chosen for `old_parts_lifetime`, during which a new part is guaranteed to be written to disk.

## max\_bytes\_to\_merge\_at\_max\_space\_in\_pool

The maximum total parts size (in bytes) to be merged into one part, if there are enough resources available.

`max_bytes_to_merge_at_max_space_in_pool` -- roughly corresponds to the maximum possible part size created by an automatic background merge.

Possible values:

- Any positive integer.

Default value: 161061273600 (150 GB).

The merge scheduler periodically analyzes the sizes and number of parts in partitions, and if there is enough free resources in the pool, it starts background merges. Merges occur until the total size of the source parts is less than `max_bytes_to_merge_at_max_space_in_pool`.

Merges initiated by `OPTIMIZE FINAL` ignore `max_bytes_to_merge_at_max_space_in_pool` and merge parts only taking into account available resources (free disk's space) until one part remains in the partition.

## max\_bytes\_to\_merge\_at\_min\_space\_in\_pool

The maximum total part size (in bytes) to be merged into one part, with the minimum available resources in the background pool.

Possible values:

- Any positive integer.

Default value: 1048576 (1 MB)

`max_bytes_to_merge_at_min_space_in_pool` defines the maximum total size of parts which can be merged despite the lack of available disk space (in pool). This is necessary to reduce the number of small parts and the chance of `Too many parts` errors.

Merges book disk space by doubling the total merged parts sizes. Thus, with a small amount of free disk space, a situation may happen that there is free space, but this space is already booked by ongoing large merges, so other merges unable to start, and the number of small parts grows with every insert.

## merge\_max\_block\_size

The number of rows that are read from the merged parts into memory.

Possible values:

- Any positive integer.

Default value: 8192

Merge reads rows from parts in blocks of `merge_max_block_size` rows, then merges and writes the result into a new part. The read block is placed in RAM, so `merge_max_block_size` affects the size of the RAM required for the merge. Thus, merges can consume a large amount of RAM for tables with very wide rows (if the average row size is 100kb, then when merging 10 parts,  $(100\text{kb} * 10 * 8192) = \sim 8\text{GB}$  of RAM). By decreasing `merge_max_block_size`, you can reduce the amount of RAM required for a merge but slow down a merge.

## max\_part\_loading\_threads

The maximum number of threads that read parts when ClickHouse starts.

Possible values:

- Any positive integer.

Default value: auto (number of CPU cores).

During startup ClickHouse reads all parts of all tables (reads files with metadata of parts) to build a list of all parts in memory. In some systems with a large number of parts this process can take a long time, and this time might be shortened by increasing `max_part_loading_threads` (if this process is not CPU and disk I/O bound).

## max\_partitions\_to\_read

Limits the maximum number of partitions that can be accessed in one query.

The setting value specified when the table is created can be overridden via query-level setting.

Possible values:

- Any positive integer.

Default value: -1 (unlimited).

## allow\_floating\_point\_partition\_key

Enables to allow floating-point number as a partition key.

Possible values:

- 0 — Floating-point partition key not allowed.
- 1 — Floating-point partition key allowed.

Default value: 0.

## check\_sample\_column\_is\_correct

Enables the check at table creation, that the data type of a column for sampling or sampling expression is correct. The data type must be one of unsigned [integer types](#): UInt8, UInt16, UInt32, UInt64.

Possible values:

- true — The check is enabled.
- false — The check is disabled at table creation.

Default value: true.

By default, the ClickHouse server checks at table creation the data type of a column for sampling or sampling expression. If you already have tables with incorrect sampling expression and do not want the server to raise an exception during startup, set `check_sample_column_is_correct` to false.

## 設定

## distributed\_product\_mode

の動作を変更します [分散サブクエリ](#).

ClickHouse applies this setting when the query contains the product of distributed tables, i.e. when the query for a distributed table contains a non-GLOBAL subquery for the distributed table.

制限:

- INおよびJOINサブクエリにのみ適用されます。
- FROMセクションが複数のシャードを含む分散テーブルを使用する場合のみ。
- サブクエリが複数のシャードを含む分散テーブルに関係する場合。
- テーブル値には使用されません [リモート](#) 機能。

可能な値:

- `deny` — Default value. Prohibits using these types of subqueries (returns the “Double-distributed in/JION subqueries is denied”例外)。
- `local` — Replaces the database and table in the subquery with local ones for the destination server (shard), leaving the normal IN/JOIN.
- `global` — Replaces the IN/JION クエリ GLOBAL IN/GLOBAL JOIN.
- `allow` — Allows the use of these types of subqueries.

## enable\_optimize\_predicate\_expression

述語プッシュダウンをオンにする [SELECT](#) クエリ。

述語プッシュダウン

可能な値:

- 0 — Disabled.
- 1 — Enabled.

デフォルト値:1。

#### 使用法

次のクエリを検討します:

1. `SELECT count() FROM test_table WHERE date = '2018-10-10'`
2. `SELECT count() FROM (SELECT * FROM test_table) WHERE date = '2018-10-10'`

もし `enable_optimize_predicate_expression = 1` ClickHouseが適用されるため、これらのクエリの実行時間は等しくなります `WHERE` それを処理するときにサブクエリに。

もし `enable_optimize_predicate_expression = 0` 次に、第二のクエリの実行時間ははるかに長くなります。 `WHERE` 句は、サブクエリの終了後にすべてのデータに適用されます。

## フォールバック\_to\_stale\_replicas\_for\_distributed\_queries

更新されたデータが使用できない場合は、クエリを古いレプリカに強制します。見る [複製](#).

ClickHouseは、テーブルの古いレプリカから最も関連性の高いものを選択します。

実行時使用 `SELECT` 複製されたテーブルを指す分散テーブルから。

デフォルトでは、1(有効)です。

## force\_index\_by\_date

インデックスを日付で使用できない場合は、クエリの実行を無効にします。

MergeTreeファミリ内のテーブルで動作します。

もし `force_index_by_date=1`, ClickHouseは、クエリにデータ範囲を制限するために使用できる日付キー条件があるかどうかをチェックします。適切な条件がない場合は、例外をスローします。ただし、条件によって読み取るデータ量が減少するかどうかはチェックされません。たとえば、次の条件 `Date != '2000-01-01'` テーブル内のすべてのデータと一致する場合でも許容されます（つまり、クエリを実行するにはフルスキヤンが必要です）。MergeTreeテーブル内のデータ範囲の詳細については、以下を参照してください [メルゲツリー](#).

## force\_primary\_key

主キーによる索引付けが不可能な場合は、クエリの実行を無効にします。

MergeTreeファミリ内のテーブルで動作します。

もし `force_primary_key=1`, ClickHouseは、クエリにデータ範囲を制限するために使用できる主キー条件があるかどうかを確認します。適切な条件がない場合は、例外をスローします。ただし、条件によって読み取るデータ量が減少するかどうかはチェックされません。MergeTreeテーブルのデータ範囲の詳細については、以下を参照してください [メルゲツリー](#).

## format\_schema

このパラメーターは、次のようなスキーマ定義を必要とする形式を使用している場合に便利です [Cap'N Proto](#) または [プロトコル](#). 値は形式によって異なります。

## fsync\_metadata

有効または無効にします `fsync` 書くとき `.sql` ファイル既定で有効になっています。

ことは、あってはならないことで無効にすることもできれば、サーバは、数百万の小さなテーブルが続々と生まれてくると破壊されました。

## enable\_http\_compression

HTTP要求に対する応答のデータ圧縮を有効または無効にします。

詳細については、[HTTP](#).

可能な値:

- 0 — Disabled.
- 1 — Enabled.

デフォルト値は0です。

## http\_zlib\_compression\_level

HTTP要求に対する応答のデータ圧縮のレベルを次の場合に設定します [enable\\_http\\_compression=1](#).

可能な値:1から9までの数値。

デフォルト値は3です。

## http\_native\_compression\_disable\_checksumming\_on\_decompression

クライア ClickHouseネイティブ圧縮フォーマットにのみ使用されます([gzip](#) または [deflate](#)).

詳細については、[HTTP](#).

可能な値:

- 0 — Disabled.
- 1 — Enabled.

デフォルト値は0です。

## send\_progress\_in\_http\_headers

有効または無効にします X-ClickHouse-Progress HTTP応答ヘッダー `clickhouse-server` 応答。

詳細については、[HTTP](#).

可能な値:

- 0 — Disabled.
- 1 — Enabled.

デフォルト値は0です。

## max\_http\_get\_redirects

HTTP GETリダイレクトホップの最大数を制限します。URL-エンジンテーブル。この設定は、両方のタイプのテーブルに適用されます。[CREATE TABLE](#) クエリとによって `url` テーブル関数。

可能な値:

- 任意の正の整数ホップ数。
- 0 — No hops allowed.

デフォルト値は0です。

## input\_format\_allow\_errors\_num

テキスト形式 (CSV、TSVなど) から読み取るときに許容されるエラーの最大数を設定します。).

既定値は0です。

常にそれをペアにします `input_format_allow_errors_ratio`.

行の読み取り中にエラーが発生したが、エラーカウンタがそれより小さい場合

`input_format_allow_errors_num`, ClickHouseは行を無視し、次の行に移動します。

両方の場合 `input_format_allow_errors_num` と `input_format_allow_errors_ratio` を超えた場合、ClickHouseは例外をスローします。

## input\_format\_allow\_errors\_ratio

テキスト形式 (CSV、TSVなど) から読み取るときに許容されるエラーの最大割合を設定します。).

エラーの割合は、0から1の間の浮動小数点数として設定されます。

既定値は0です。

常にそれをペアにします `input_format_allow_errors_num`.

行の読み取り中にエラーが発生したが、エラーカウンタがそれより小さい場合

`input_format_allow_errors_ratio`, ClickHouseは行を無視し、次の行に移動します。

両方の場合 `input_format_allow_errors_num` と `input_format_allow_errors_ratio` を超えた場合、ClickHouseは例外をスローします。

## input\_format\_values\_interpret\_expressions

を有効または無効にしのSQLのパーサの場合の高速ストリームのパーサで構文解析のデータです。この設定は、[値](#) データ挿入時の書式。構文解析の詳細については、以下を参照してください [構文](#) セクション

可能な値:

- 0 — Disabled.

この場合、提供しなければなりません形式のデータです。を参照。[形式](#) セクション

- 1 — Enabled.

この場合、SQL式を値として使用できますが、データの挿入はこの方法ではるかに遅くなります。書式設定されたデータのみを挿入すると、ClickHouseは設定値が0であるかのように動作します。

デフォルト値:1。

使用例

挿入 [DateTime](#) 異なる設定で値を入力します。

```
SET input_format_values_interpret_expressions = 0;
INSERT INTO datetime_t VALUES (now())
```

```
Exception on client:
Code: 27. DB::Exception: Cannot parse input: expected ) before: now()): (at row 1)
```

```
SET input_format_values_interpret_expressions = 1;
INSERT INTO datetime_t VALUES (now())
```

Ok.

最後のクエリは次のものと同じです:

```
SET input_format_values_interpret_expressions = 0;
INSERT INTO datetime_t SELECT now()
```

Ok.

## input\_format\_values\_deduce\_templates\_of\_expressions

SQL式のテンプレート控除を有効または無効にします。 値 形式。 この解析と解釈表現 Values 連続する行の式が同じ構造を持つ場合、はるかに高速です。 ClickHouseは式のテンプレートを推論し、このテンプレートを使用して次の行を解析し、正常に解析された行のパッチで式を評価しようとします。

可能な値:

- 0 — Disabled.
- 1 — Enabled.

デフォルト値:1。

次のクエリの場合:

```
INSERT INTO test VALUES (lower('Hello')), (lower('world')), (lower('INSERT')), (upper('Values')), ...
```

- もし `input_format_values_interpret_expressions=1` と `format_values_deduce_templates_of_expressions=0`、式は各行ごとに別々に解釈されます（これは多数の行では非常に遅いです）。
- もし `input_format_values_interpret_expressions=0` と `format_values_deduce_templates_of_expressions=1`、第一、第二および第三の行の式は、テンプレートを使用して解析されます `lower(String)` そして一緒に解釈されると、forth行の式は別のテンプレートで解析されます (`upper(String)`)。
- もし `input_format_values_interpret_expressions=1` と `format_values_deduce_templates_of_expressions=1`、前の場合と同じですが、テンプレートを推論できない場合は、式を別々に解釈することもできます。

## input\_format\_values\_accurate\_types\_of\_literals

この設定は次の場合にのみ使用されます `input_format_values_deduce_templates_of_expressions = 1`。 いくつかの列の式は同じ構造を持ちますが、異なる型の数値リテラルが含まれています。

```
(..., abs(0), ...),      -- UInt64 literal
(..., abs(3.141592654), ...), -- Float64 literal
(..., abs(-1), ...),     -- Int64 literal
```

可能な値:

- 0 — Disabled.

In this case, ClickHouse may use a more general type for some literals (e.g., `Float64` または `Int64` 代わりに `UInt64` のために 42が、その原因となりオーバーフローおよび精度の問題です。

- 0 — Disabled.

この場合、ClickHouseはリテラルの実際の型をチェックし、対応する型の式テンプレートを使用します。場合によつては、`Values`.

デフォルト値:1。

## input\_format\_defaults\_for\_omitted\_fields

実行するとき `INSERT` 省略された入力列の値を、それぞれの列のデフォルト値に置き換えます。このオプションは `JSONEachRow`, `CSV` と `TabSeparated` フォーマット。

### 注

このオプション 消費量で追加的に計算資源をサーバーを低減できる。

可能な値:

- 0 — Disabled.
- 1 — Enabled.

デフォルト値:1。

## input\_format\_tsv\_empty\_as\_default

有効にすると、TSVの空の入力フィールドを既定値に置き換えます。複雑な既定の式の場合 `input_format_defaults_for_omitted_fields` 有効にする必要があります。

既定では無効です。

## input\_format\_null\_as\_default

を有効または無効にし用のデフォルト値が入力データを含む `NULL` しかし、`not`の対応する列のデータ型 `Nullable(T)` (テキスト入力形式の場合)。

## input\_format\_skip\_unknown\_fields

追加データのスキップ挿入を有効または無効にします。

書き込みデータClickHouseが例外をスローした場合入力データを含むカラム特別な権限は必要ありません使用します。スキップが有効な場合、ClickHouseは余分なデータを挿入せず、例外をスローしません。

対応形式:

- `JSONEachRow`
- `CSVWithNames`
- `TabSeparatedWithNames`
- `TSKV`

可能な値:

- 0 — Disabled.
- 1 — Enabled.

デフォルト値は0です。

## `input_format_import_nested_json`

を有効または無効にし、挿入のJSONデータをネストしたオブジェクト。

対応形式:

- `JSONEachRow`

可能な値:

- 0 — Disabled.
- 1 — Enabled.

デフォルト値は0です。

も参照。:

- 入れ子構造の使用 と `JSONEachRow` 形式。

## `input_format_with_names_use_header`

データ挿入時の列順序の確認を有効または無効にします。

挿入のパフォーマンスを向上させるために、入力データの列の順序がターゲットテーブルと同じであることが確実な場合は、このチェックを無効にするこ

対応形式:

- `CSVWithNames`
- `TabSeparatedWithNames`

可能な値:

- 0 — Disabled.
- 1 — Enabled.

デフォルト値:1。

## `date_time_input_format`

日付と時刻のテキスト表現のパーサーを選択できます。

この設定は次の場合には適用されません [日付と時刻の関数](#).

可能な値:

- `'best_effort'` — Enables extended parsing.

ClickHouseは基本を解析できます `YYYY-MM-DD HH:MM:SS` 書式とすべて [ISO 8601](#) 日付と時刻の形式。例えば, `'2018-06-08T01:02:03.000Z'`.

- `'basic'` — Use basic parser.

ClickHouseは基本のみを解析できます `YYYY-MM-DD HH:MM:SS` 形式。例えば, `'2019-08-20 10:18:56'`.

デフォルト値: `'basic'`.

も参照。:

- `DateTime`データ型。

- 日付と時刻を操作するための関数。

## join\_default\_strictness

デフォルトの厳密さを [結合句](#)。

可能な値:

- ALL — If the right table has several matching rows, ClickHouse creates a [デカルト積](#) 一致する行から。これは正常です [JOIN 標準SQL](#)からの動作。
- ANY — If the right table has several matching rows, only the first one found is joined. If the right table has only one matching row, the results of ANY と ALL 同じです。
- ASOF — For joining sequences with an uncertain match.
- Empty string — If ALL または ANY クエリで指定されていない場合、ClickHouseは例外をスローします。

デフォルト値: ALL。

## join\_any\_take\_last\_row

結合操作の動作を ANY 厳密さ。

### 注意

この設定は [JOIN](#) との操作 参加 エンジンテーブル。

可能な値:

- 0 — If the right table has more than one matching row, only the first one found is joined.
- 1 — If the right table has more than one matching row, only the last one found is joined.

デフォルト値は0です。

も参照。:

- [JOIN句](#)
- [結合テーブルエンジン](#)
- [join\\_default\\_strictness](#)

## join\_use\_nulls

のタイプを設定します。 [JOIN](#) 行動。 際融合のテーブル、空細胞が表示される場合があります。 ClickHouseは、この設定に基づいて異なる塗りつぶします。

可能な値:

- 0 — The empty cells are filled with the default value of the corresponding field type.
- 1 — [JOIN 標準SQL](#)と同じように動作します。 対応するフィールドの型は次のように変換されます [Null可能](#) 空のセルは [NULL](#)。

デフォルト値は0です。

## max\_block\_size

ClickHouseでは、データはブロック（列部分のセット）によって処理されます。 単一ブロックの内部処理サイクルは十分に効率的ですが、各ブロックには顕著な支出があります。 その `max_block_size` 設定は、テーブルからロードするブロックのサイズ（行数）の推奨事項です。 ブロックサイズは小さすぎるので、各ブロックの支出はまだ目立ちますが、最初のブロックの後に完了したLIMITのクエリがすばやく処理されるよう 目標は、複数のスレッドで多数の列を抽出するときにメモリを消費するのを避け、少なくともいくつかのキャッシュの局所性を保持することです。

デフォルト値は65,536です。

ブロックのサイズ `max_block_size` ていなから読み込まれます。 場合ことは明らかであることなくデータを取得され、さらに小型のブロックを処理します。

## `preferred_block_size_bytes`

と同じ目的のために使用される `max_block_size` しかし、ブロック内の行数に適応させることによって、推奨されるブロックサイズをバイト単位で設定します。

ただし、ブロックサイズは次のようにになります `max_block_size` 行。

既定では、1,000,000です。 MergeTreeエンジンから読み取るときにのみ動作します。

## `merge_tree_min_rows_for_concurrent_read`

のファイルから読み込まれる行数が メルゲツリー テーブル超過 `merge_tree_min_rows_for_concurrent_read` その後 ClickHouseしようとして行な兼職の状況からの読み出しこのファイルに複数のスレッド）。

可能な値:

- 任意の正の整数。

デフォルト値は163840です。

## `merge_tree_min_bytes_for_concurrent_read`

のファイルから読み取るバイト数が メルゲツリー-エンジン表超過 `merge_tree_min_bytes_for_concurrent_read` 次に、 ClickHouseはこのファイルから複数のスレッドで同時に読み取ろうとします。

可能な値:

- 任意の正の整数。

デフォルト値は251658240です。

## `merge_tree_min_rows_for_seek`

一つのファイルに読み込まれる二つのデータブロック間の距離が `merge_tree_min_rows_for_seek` その後、ClickHouseはファイルをシークせず、データを順番に読み込みます。

可能な値:

- 任意の正の整数。

デフォルト値は0です。

## `merge_tree_min_bytes_for_seek`

一つのファイルに読み込まれる二つのデータブロック間の距離が `merge_tree_min_bytes_for_seek` その後、ClickHouseは両方のブロックを含むファイルの範囲を順番に読み込むため、余分なシークを避けます。

可能な値:

- 任意の正の整数。

デフォルト値は0です。

## `merge_tree_coarse_index_granularity`

する場合のデータClickHouseチェックのデータにファイルです。必要なキーがある範囲にあることをClickHouseが検出すると、この範囲を次のように分割します `merge_tree_coarse_index_granularity` 必要なキーを再帰的に検索します。

可能な値:

- 任意の正の偶数の整数。

デフォルト値:8。

## `merge_tree_max_rows_to_use_cache`

ClickHouseがより多くを読むべきであれば `merge_tree_max_rows_to_use_cache` あるクエリでは、非圧縮ブロックのキャッシュを使用しません。

のキャッシュされた、圧縮解除されたブロックの店舗データを抽出したためます。ClickHouseこのキャッシュの高速化対応小の繰り返します。この設定は、大量のデータを読み取るクエリによってキャッシュが破損するのを防ぎます。その `uncompressed_cache_size` サーバー設定は、非圧縮ブロックのキャッシュのサイズを定義します。

可能な値:

- 任意の正の整数。

Default value:  $128 \times 8192$ .

## `merge_tree_max_bytes_to_use_cache`

ClickHouseがより多くを読むべきであれば `merge_tree_max_bytes_to_use_cache` 一つのクエリでは、非圧縮ブロックのキャッシュを使用しません。

のキャッシュされた、圧縮解除されたブロックの店舗データを抽出したためます。ClickHouseこのキャッシュの高速化対応小の繰り返します。この設定は、大量のデータを読み取るクエリによってキャッシュが破損するのを防ぎます。その `uncompressed_cache_size` サーバー設定は、非圧縮ブロックのキャッシュのサイズを定義します。

可能な値:

- 任意の正の整数。

既定値:2013265920。

## `min_bytes_to_use_direct_io`

ストレージディスクへの直接I/Oアクセスを使用するために必要な最小データ量。

ClickHouseこの設定からデータを読み込むとします。読み取るすべてのデータの合計ストレージ容量が超過した場合 `min_bytes_to_use_direct_io` その後、ClickHouseはストレージディスクからデータを読み取ります。`O_DIRECT` オプション

可能な値:

- 0 — Direct I/O is disabled.
- 正の整数。

デフォルト値は0です。

## `log_queries`

クエリログの設定。

この設定でClickHouseに送信されたクエリは、[query\\_log](#) サーバー構成パラメータ。

例:

```
log_queries=1
```

## log\_queries\_min\_type

`query_log` ログに記録する最小タイプ。

可能な値:

- `QUERY_START` (=1)
- `QUERY_FINISH` (=2)
- `EXCEPTION_BEFORE_START` (=3)
- `EXCEPTION_WHILE_PROCESSING` (=4)

デフォルト値: `QUERY_START`.

どのエンティリーが行くかを制限するために使用できます `query_log` んだ興味深いだけ誤差を利用することができます  
`EXCEPTION_WHILE_PROCESSING`:

```
log_queries_min_type='EXCEPTION_WHILE_PROCESSING'
```

## log\_query\_threads

クエリスレッドログの設定。

この設定でClickHouseによって実行されたクエリのスレッドは、[query\\_thread\\_log](#) サーバー構成パラメータ。

例:

```
log_query_threads=1
```

## max\_insert\_block\_size

テーブルに挿入するために形成するブロックのサイズ。

この設定は、サーバーがブロックを形成する場合にのみ適用されます。

たとえば、HTTPインターフェイスを介した挿入の場合、サーバーはデータ形式を解析し、指定されたサイズのブロックを形成します。

しかし、clickhouse-clientを使用すると、クライアントはデータ自体を解析し、「`max_insert_block_size`」サーバーでの設定は、挿入されたブロックのサイズには影響しません。

データはSELECT後に形成されるのと同じブロックを使用して挿入されるため、INSERT SELECTを使用するときにもこの設定には目的がありません。

既定値は1,048,576です。

デフォルトは、`max_block_size`。この理由は、特定のテーブルエンジンが原因です (`*MergeTree`) 挿入されたブロックごとにディスク上にデータ部分を形成します。同様に、`*MergeTree` テーブルデータを並べ替え時の挿入やるのに十分な大きさのブロックサイズを選別データにアプライです。

## min\_insert\_block\_size\_rows

テーブルに挿入できるブロック内の最小行数を設定します。`INSERT` クエリ。 小さめサイズのブロックをつぶし入れます。

可能な値:

- 正の整数。
- 0 — Squashing disabled.

デフォルト値は1048576です。

## min\_insert\_block\_size\_bytes

テーブルに挿入できるブロック内の最小バイト数を設定します。 `INSERT` クエリ。 小さめサイズのブロックをつぶしります。

可能な値:

- 正の整数。
- 0 — Squashing disabled.

デフォルト値:268435456。

## max\_replica\_delay\_for\_distributed\_queries

分散クエリの遅延レプリカを無効にします。 見る [複製](#).

時間を秒単位で設定します。 レプリカが設定値より遅れている場合、このレプリカは使用されません。

デフォルト値は300です。

実行時使用 `SELECT` 複製されたテーブルを指す分散テーブルから。

## max\_threads

リモートサーバーからデータを取得するためのスレッドを除く、クエリ処理スレッドの最大数  
'max\_distributed\_connections' 変数)。

このパラメータに適用されるスレッドは、それらのスレッドが同じ段階での問合せ処理パイプライン。  
たとえば、テーブルから読み込むときに、関数式を評価することができる場合は、少なくともを使用して `WHERE` と  
`GROUP BY` の事前集計を並列に使用してフィル 'max\_threads' その後、スレッドの数 'max\_threads' 使用されます。

デフォルト値:物理CPUコアの数。

サーバーで同時に実行される `SELECT` クエリが通常より少ない場合は、このパラメーターを実際のプロセッサコア数より  
わずかに小さい値に設定します。

制限のために迅速に完了するクエリの場合は、低い値を設定できます 'max\_threads'. たとえば、必要な数のエントリ  
がすべてのブロックにあり、`max_threads=8`の場合、8つのブロックが取得されます。

小さいほど `max_threads` 値は、より少ないメモリが消費されます。

## max\_insert\_threads

実行するスレッドの最大数 `INSERT SELECT` クエリ。

可能な値:

- 0 (or 1) — `INSERT SELECT` 並列実行はありません。
- 正の整数。 1より大きい。

デフォルト値は0です。

平行 `INSERT SELECT` のみ効果があります。 `SELECT` パーツは並列で実行されます。 `max_threads` 設定。  
値を大きくすると、メモリ使用量が増えます。

## **max\_compress\_block\_size**

テーブルに書き込むための圧縮前の非圧縮データのブロックの最大サイズ。既定では、1,048,576(1MiB)です。サイズを小さくすると、圧縮率が大幅に低下し、キャッシュの局所性のために圧縮と解凍の速度がわずかに増加し、メモリ消費が減少します。通常、この設定を変更する理由はありません。

圧縮のためのブロック（バイトで構成されるメモリの塊）とクエリ処理のためのブロック（テーブルからの行のセット）を混同しないでください。

## **min\_compress\_block\_size**

のために **メルゲツリー**"テーブル。削減のため、遅延が処理クエリーのブロックの圧縮を書くとき、次のマークがそのサイズは少なくとも 'min\_compress\_block\_size'。既定では、65,536です。

圧縮されていないデータが以下の場合、ブロックの実際のサイズ 'max\_compress\_block\_size' は、この値以上であり、一つのマークのデータ量以上である。

例を見てみましょう。仮定すると 'index\_granularity' テーブル作成時に8192に設定されました。

UInt32型の列（値あたり4バイト）を書いています。8192行を書き込むと、合計は32KBのデータになります。  
Min\_compress\_block\_size=65,536なので、圧縮されたブロックはすべてのマークに対して形成されます。

文字列タイプ（値あたり60バイトの平均サイズ）のURL列を作成しています。8192行を書き込むと、平均は500KBのデータよりわずかに小さくなります。これは65,536以上であるため、各マークに圧縮されたブロックが形成されます。この場合、単一のマークの範囲でディスクからデータを読み取るとき、余分なデータは解凍されません。

通常、この設定を変更する理由はありません。

## **max\_query\_size**

SQLパーサーを使用して解析するためにRAMに取り込むことができるクエリの最大部分。

INSERTクエリには、別のストリームパーサー( $O(1)$ RAMを消費する)によって処理されるINSERTのデータも含まれていますが、この制限には含まれていません。

デフォルト値:256KiB。

## **interactive\_delay**

区間マイクロ秒単位で確認を行うための要求実行中止となり、送信を行います。

デフォルト値:100,000(キャンセルをチェックし、進行状況を秒単位で送信します)。

## **connect\_timeout,receive\_timeout,send\_timeout**

クライアントとの通信に使用されるソケットの秒単位のタイムアウト。

デフォルト値:10,300,300。

## **cancel\_http\_READONLY\_queries\_on\_client\_close**

Cancels HTTP read-only queries (e.g. SELECT) when a client closes the connection without waiting for the response.

デフォルト値:0

## **poll\_interval**

指定された秒数の待機ループをロックします。

デフォルト値は10です。

## **max\_distributed\_connections**

単一の分散テーブルへの単一のクエリの分散処理のためのリモートサーバーとの同時接続の最大数。 クラスター内のサーバー数以上の値を設定することをお勧めします。

デフォルト値:1024。

次のパラメーターは、分散テーブルを作成するとき(およびサーバーを起動するとき)にのみ使用されるため、実行時に変更する理由はありません。

## **distributed\_connections\_pool\_size**

単一の分散テーブルに対するすべてのクエリの分散処理のためのリモートサーバーとの同時接続の最大数。 クラスター内のサーバー数以上の値を設定することをお勧めします。

デフォルト値:1024。

## **connect\_timeout\_with\_failover\_ms**

分散テーブルエンジンのリモートサーバーに接続するためのタイムアウト時間(ミリ秒)。'shard' と 'replica' セクションは、クラスタ定義で使用されます。

失敗した場合は、さまざまなレプリカへの接続が試行されます。

デフォルト値は50です。

## **connections\_with\_failover\_max\_tries**

分散テーブルエンジンの各レプリカとの接続試行の最大数。

デフォルト値は3です。

## **極端な**

極端な値(クエリ結果の列の最小値と最大値)を数えるかどうか。 0または1を受け入れます。 既定では、0(無効)です。 詳細については "Extreme values".

## **use\_uncompressed\_cache**

非圧縮ブロックのキャッシュを使用するかどうか。 0または1を受け入れます。 既定では、0(無効)です。

非圧縮キャッシュ(MergeTreeファミリ内のテーブルのみ)を使用すると、多数の短いクエリを処理する場合に、待ち時間を大幅に削減してスループットを向上させ この設定を有効にユーザーに送信頻繁に短います。 またに注意を払って下さい [uncompressed\\_cache\\_size configuration parameter \(only set in the config file\)](#) - the size of uncompressed cache blocks. By default, it is 8 GiB. The uncompressed cache is filled in as needed and the least-used data is automatically deleted.

少なくとも幾分大きな量のデータ (百万行以上) を読み取るクエリの場合、圧縮されていないキャッシュは自動的に無効になります。 本当に小さなクエリの これは保つことができる意味する 'use\_uncompressed\_cache' 設定は常に1に設定します。

## **replace\_running\_query**

HTTPインターフェイスを使用する場合、「query\_id」変数は渡すことができます。 これは、クエリ識別子として機能する任意の文字列です。

同じユーザーからのクエリが同じ場合 'query\_id' この時点で既に存在しているので、動作は 'replace\_running\_query' パラメータ。

0 (default) - Throw an exception (don't allow the query to run if a query with the same 'query\_id' すでに実行されています)。

## 1 – Cancel the old query and start running the new one.

Yandex.Metricaこのパラメータセットが1の実施のための提案のための分割ます。次の文字を入力した後、古いクエリがまだ終了していない場合は、キャンセルする必要があります。

## stream\_flush\_interval\_ms

作品のテーブルストリーミングの場合はタイムアウトした場合、またはスレッドを生成す **max\_insert\_block\_size** 行。

既定値は7500です。

値が小さいほど、データがテーブルにフラッシュされる頻度が高くなります。値を小さく設定すると、パフォーマンスが低下します。

## load\_balancing

分散クエリ処理に使用されるレプリカ選択のアルゴリズムを指定します。

ClickHouse対応し、以下のようなアルゴリズムの選択のレプリカ:

- **ランダム** (デフォルトでは)
- **最寄りのホスト名**
- **順番に**
- **最初またはランダム**

### ランダム(デフォルト)

```
load_balancing = random
```

エラーの数は、レプリカごとにカウントされます。クエリは、エラーが最も少ないレプリカに送信されます。  
レプリカのデータが異なる場合は、異なるデータも取得されます。

### 最寄りのホスト名

```
load_balancing = nearest_hostname
```

The number of errors is counted for each replica. Every 5 minutes, the number of errors is integrally divided by 2. Thus, the number of errors is calculated for a recent time with exponential smoothing. If there is one replica with a minimal number of errors (i.e. errors occurred recently on the other replicas), the query is sent to it. If there are multiple replicas with the same minimal number of errors, the query is sent to the replica with a hostname that is most similar to the server's hostname in the config file (for the number of different characters in identical positions, up to the minimum length of both hostnames).

例えば、example01-01-1やexample01-01-2.yandex.ru 一つの場所で異なるが、example01-01-1とexample01-02-2は二つの場所で異なる。

この方法は原始的に見えるかもしれません、ネットワークトポジに関する外部データを必要とせず、Ipアドレスを比較することもありません。

したがって、同等のレプリカがある場合は、名前で最も近いレプリカが優先されます。

また、同じサーバーにクエリを送信するときに、障害がない場合、分散クエリも同じサーバーに移動すると仮定することもできます。したがって、レプリカに異なるデータが配置されても、クエリはほとんど同じ結果を返します。

### 順番に

```
load_balancing = in_order
```

同じ数のエラーを持つレプリカは、構成で指定されている順序と同じ順序でアクセスされます。  
この方法は、適切なレプリカを正確に把握している場合に適しています。

## 最初またはランダム

```
load_balancing = first_or_random
```

このアルゴリズムは、セット内の最初のレプリカを選択します。クロスレプリケーショントポロジの設定では有効ですが、他の構成では役に立ちません。

その `first_or_random` アルゴリズムはの問題を解決します `in_order` アルゴリズムと `in_order` あるレプリカがダウンした場合、残りのレプリカは通常のトラフィック量を処理しますが、次のレプリカは二重負荷を受けます。を使用する場合 `first_or_random` アルゴリズムでは、負荷はまだ利用可能なレプリカ間で均等に分散されます。

## prefer\_localhost\_replica

を有効/無効にしが好みのlocalhostレプリカ処理時に分布します。

可能な値:

- 1 — ClickHouse always sends a query to the localhost replica if it exists.
- 0 — ClickHouse uses the balancing strategy specified by the `load_balancing` 設定。

デフォルト値:1。

### 警告

使用する場合は、この設定を無効にします `max_parallel_replicas`.

## totals\_mode

有するときの合計の計算方法、および`max_rows_to_group_by`および`group_by_overflow_mode= 'any'`存在する。  
セクションを参照 “WITH TOTALS modifier”。

## totals\_auto\_threshold

のしきい値 `totals_mode = 'auto'`.  
セクションを参照 “WITH TOTALS modifier”.

## max\_parallel\_replicas

クエリ実行時の各シャードのレプリカの最大数。

のための一貫性を異なる部分に同じデータを分割)、このオプションにしているときだけサンプリングキーを設定します。

レプリカラグは制御されません。

## output\_format\_json\_quote\_64bit\_integers

値がtrueの場合、`json*Int64`および`UInt64`形式（ほとんどのJavaScript実装との互換性のため）を使用するときに整数が引用符で表示されます。

## format\_csv\_delimiter

CSVデータの区切り文字として解釈される文字。デフォルトでは、区切り文字は`,`。

## `input_format_csv_unquoted_null_literal_as_null`

CSV入力形式の場合、引用符なしの解析を有効または無効にします `NUL` リテラルとして(のシノニム`\N`)。

## `output_format_csv_crlf_end_of_line`

CSVでは、UNIXスタイル(LF)の代わりにDOS/Windowsスタイルの行区切り記号(CRLF)を使用します。

## `output_format_tsv_crlf_end_of_line`

TSVでは、UNIXスタイル(LF)の代わりにDOC/Windowsスタイルの行区切り記号(CRLF)を使用します。

## `insert_quorum`

を決議の定足数を書き込みます。

- もし `insert_quorum < 2` クオーラム書き込みは無効です。
- もし `insert_quorum >= 2` クオーラム書き込みが有効になります。

デフォルト値は0です。

定足数書き込み

`INSERT` 承ClickHouse管理を正しく書き込みデータの `insert_quorum` の間のレプリカの `insert_quorum_timeout`. 何らかの理由で書き込みが成功したレプリカの数が `insert_quorum` を書くのは失敗したとClickHouseを削除するに挿入したブロックからすべてのレプリカがデータがすでに記されています。

クオーラム内のすべてのレプリカは一貫性があります。`INSERT` クエリ。その `INSERT` シーケンスは線形化されます。

から書き込まれたデータを読み取ると `insert_quorum` を使用することができます `select_sequential_consistency` オプション

ClickHouseは例外を生成します

- クエリの時点で使用可能なレプリカの数が `insert_quorum`.
- 前のブロックがまだ挿入されていないときにデータを書き込もうとすると `insert_quorum` レプリカの。この状況は、ユーザーが `INSERT` 前のもの前に `insert_quorum` 完了です。

も参照。:

- `insert_quorum_timeout`
- `select_sequential_consistency`

## `insert_quorum_timeout`

書き込み数が定員タイムアウトを秒で指定します。タイムアウトが経過し、まだ書き込みが行われていない場合、ClickHouseは例外を生成し、クライアントは同じまたは他のレプリカに同じブロックを書き込む

デフォルト値は60秒です。

も参照。:

- `insert_quorum`
- `select_sequential_consistency`

## `select_sequential_consistency`

の順次整合性を有効または無効にします `SELECT` クエリ:

可能な値:

- 0 — Disabled.
- 1 — Enabled.

デフォルト値は0です。

使用法

順次整合性が有効になっている場合、ClickHouseはクライアントが `SELECT` 以前のすべてのデータを含むレプリカのみを照会します `INSERT` 以下で実行されるクエリ `insert_quorum`. クライアントが部分レプリカを参照している場合、ClickHouseは例外を生成します。 `SELECT` クエリには、レプリカのクオーラムにまだ書き込まれていないデータは含まれません。

も参照。:

- [insert\\_quorum](#)
- [insert\\_quorum\\_timeout](#)

## `insert_deduplicate`

重複除外のブロックを有効または無効にします。 `INSERT` (複製された\*テーブルの場合)。

可能な値:

- 0 — Disabled.
- 1 — Enabled.

デフォルト値:1。

デフォルトでは、ブロックは `INSERT` ステートメントは重複排除されます ([データ複製](#)).

## `deduplicate_blocks_in_dependent_materialized_views`

を有効または無効にし、重複排除圧縮をチェックを実現し意見を受け取るデータから複製\*ます。

可能な値:

0 — Disabled.  
1 — Enabled.

デフォルト値は0です。

使用法

デフォルトでは、重複除外はマテリアライズドビューでは実行されませんが、ソーステーブルの上流で実行されます。ソーステーブルの重複除外のために挿入されたロックがスキップされた場合、添付されたマテリアライズドビューには挿入されません。この動作は、マテリアライズドビューの集計後に挿入されたロックが同じで、ソーステーブルへの異なる挿入から派生した場合に、高度に集計されたデータ

同時に、この動作 “breaks” `INSERT` べき等性。もし `INSERT` メインテーブルに成功したと `INSERT into a materialized view failed (e.g. because of communication failure with Zookeeper) a client will get an error and can retry the operation. However, the materialized view won't receive the second insert because it will be discarded by deduplication in the main (source) table. The setting deduplicate_blocks_in_dependent_materialized_views こ`

の動作を変更できます。再試行すると、マテリアライズドビューは繰り返し挿入を受け取り、重複除外チェックを単独で実行します。

ソーステーブルのチェック結果を無視し、最初の失敗のために失われた行を挿入します。

## max\_network\_bytes

クエリの実行時にネットワーク経由で受信または送信されるデータ量(バイト単位)を制限します。この設定は、個々のクエリごとに適用されます。

可能な値:

- 正の整数。
- 0 — Data volume control is disabled.

デフォルト値は0です。

## max\_network\_bandwidth

ネットワーク上でのデータ交換の速度を毎秒バイト単位で制限します。この設定はすべての照会に適用されます。

可能な値:

- 正の整数。
- 0 — Bandwidth control is disabled.

デフォルト値は0です。

## max\_network\_bandwidth\_for\_user

ネットワーク上でのデータ交換の速度を毎秒バイト単位で制限します。この設定は、単一ユーザーが同時に実行するすべてのクエリに適用されます。

可能な値:

- 正の整数。
- 0 — Control of the data speed is disabled.

デフォルト値は0です。

## max\_network\_bandwidth\_for\_all\_users

ネットワーク経由でデータが交換される速度を毎秒バイト単位で制限します。この設定が適用されるのはすべての同時走行に関するお問い合わせです。

可能な値:

- 正の整数。
- 0 — Control of the data speed is disabled.

デフォルト値は0です。

## count\_distinctImplementation

これは、`uniq*` を実行するために使用する必要があります。`COUNT(DISTINCT ...)` 建設。

可能な値:

- `uniq`

- uniqCombined
- uniqCombined64
- uniqHLL12
- uniqExact

デフォルト値: uniqExact.

## skip\_unavailable\_shards

を有効または無効にし静キの不可欠片.

ザ-シャーがある場合にはご利用できないそのすべてのレプリカのためご利用いただけません。次の場合、レプリカは使用できません:

- ClickHouseは何らかの理由でレプリカに接続できません。

レプリカに接続すると、ClickHouseはいくつかの試行を実行します。すべてこれらの試みが失敗し、レプリカとはできます。

- レプリカはDNSで解決できません。

レプリカのホスト名をDNS経由で解決できない場合は、次の状況を示すことがあります:

- レプリカのホストにDNSレコードがない。これは、動的DNSを持つシステムで発生する可能性があります。, **Kubernetes** ここで、ノードはダウンタイム中に解決できず、これはエラーではありません。
- 設定エラー。 ClickHouse設定ファイルが含まれて間違ったホスト名。

可能な値:

- 1 — skipping enabled.

シャードが使用できない場合、ClickHouseは部分的なデータに基づいて結果を返し、ノードの可用性の問題は報告しません。

- 0 — skipping disabled.

シャードが使用できない場合、ClickHouseは例外をスローします。

デフォルト値は0です。

## optimize\_skip\_unused\_shards

PREWHERE/WHEREにシャーディングキー条件があるSELECTクエリの未使用的のシャードのスキップを有効または無効にします(データがシャーディングキーによって配布される

デフォルト値:0

## force\_optimize\_skip\_unused\_shards

を有効または無効にしクエリの実行の場合 optimize\_skip\_unused\_shards 未使用的のシャードを有効にしてスキップすることはできません。スキップが不可能で、設定が有効になっている場合は例外がスローされます。

可能な値:

- 0-無効(スローしない)
- 1-除クエリの実行の場合のみ表はshardingキー

- 2-を無効にクエリの実行に関わらずshardingキ-一定義のテーブル

デフォルト値:0

## optimize\_throw\_if\_noop

例外のスローを有効または無効にします。 **OPTIMIZE** クエリがマージを実行しませんでした。

既定では、**OPTIMIZE** 何もしなくても正常に戻ります。 この設定を使用すると、これらの状況を区別し、例外メッセージで理由を取得できます。

可能な値:

- 1 — Throwing an exception is enabled.
- 0 — Throwing an exception is disabled.

デフォルト値は0です。

## ディストリビューター

- タイプ:秒
- デフォルト値:60秒

分散テーブルのエラーをゼロにする速度を制御します。 レプリカがしばらく使用できず、5つのエラーが蓄積され、**distributed\_replica\_error\_half\_life**が1秒に設定されている場合、レプリカは最後のエラーの3秒後に正常と見なされま  
も参照。:

- 分散テーブルエンジン
- ディストリビューター

## ディストリビューター

- 型:unsigned int
- デフォルト値:1000

各レプリカのエラ

も参照。:

- 分散テーブルエンジン
- ディストリビューター

## distributed\_directory\_monitor\_sleep\_time\_ms

の基本区間 **分散** データを送信する表エンジン。 実際の間隔は、エラーが発生した場合に指數関数的に増加します。

可能な値:

- ミリ秒の正の整数。

既定値は100ミリ秒です。

## distributed\_directory\_monitor\_max\_sleep\_time\_ms

の最大間隔 **分散** データを送信する表エンジン。 の区間の指數関数的成長を制限する。

**distributed\_directory\_monitor\_sleep\_time\_ms** 設定。

可能な値:

- ミリ秒の正の整数。

デフォルト値:30000ミリ秒(30秒)。

## distributed\_directory\_monitor\_batch\_inserts

挿入されたデータのパッチ送信を有効または無効にします。

パッチ送信が有効になっている場合、[分散](#) テーブルエンジンをお送り複数のファイルの挿入データを移動するようになっていますの代わりに送信します。一括送信の改善にクラスターの性能をより活用してサーバやネットワーク資源です。

可能な値:

- 1 — Enabled.
- 0 — Disabled.

デフォルト値は0です。

## os\_thread\_priority

優先度を設定します ([ニース](#))クエリを実行するスレッドの場合。OSスケジューラは、使用可能な各CPUコアで実行する次のスレッドを選択する際に、この優先順位を考慮します。

### 警告

この設定を使用するには、[CAP\\_SYS\\_NICE](#) 能力。その `clickhouse-server` パッケ いくつかの仮想環境では、[CAP\\_SYS\\_NICE](#) 能力。この場合, `clickhouse-server` 開始時にそれに関するメッセージを表示します。

可能な値:

- 範囲の値を設定できます [-20, 19].

値が低いほど優先度が高くなります。低いの系 `nice` 優先度の値は、高い値のスレッドよりも頻繁に実行されます。長時間実行される非対話型クエリでは、短い対話型クエリが到着したときにリソースをすばやく放棄できるため、高い値が望ましいです。

デフォルト値は0です。

## query\_profiler\_real\_time\_period\_ns

の実クロックタイマーの期間を設定します。 [クエリプロファイラ](#). リアルクロックタイマーカウント壁掛時計。

可能な値:

- ナノ秒単位の正の整数です。

推奨値:

- 10000000 (100 times a second) nanoseconds and less for single queries.  
- 1000000000 (once a second) for cluster-wide profiling.

- タイマーをオフにする場合は0。

タイプ: [UInt64](#).

デフォルト値:10000000000ナノ秒(秒)。

も参照。:

- システム表 [trace\\_log](#)

## query\_profiler\_cpu\_time\_period\_ns

のCPUクロックタイマーの期間を設定します。 [クエリプロファイラ](#). このタイマーカウントのみのCPU時間。

可能な値:

- ナノ秒の正の整数。

推奨値:

```
- 10000000 (100 times a second) nanoseconds and more for single queries.  
- 10000000000 (once a second) for cluster-wide profiling.
```

- タイマーをオフにする場合は0。

タイプ: [UInt64](#).

デフォルト値:10000000000ナノ秒。

も参照。:

- システム表 [trace\\_log](#)

## allow\_introspection\_functions

ディスエーブルの有効 [introspection関数](#) クエリプロファイル用。

可能な値:

- 1 — Introspection functions enabled.
- 0 — Introspection functions disabled.

デフォルト値は0です。

も参照。

- [サンプリングクロファイラ](#)
- システム表 [trace\\_log](#)

## input\_format\_parallel\_parsing

- タイプ:bool
- 既定値:True

データ形式の順序保持並列解析を有効にします。 TSV、TKSV、CSVおよびJSONEachRow形式でのみサポートされます。

## min\_chunk\_bytes\_for\_parallel\_parsing

- 型:unsigned int
- デフォルト値:1MiB

各スレッドが並列に解析する最小チャンクサイズをバイト単位で表します。

## output\_format\_avro\_codec

出力Avroファイルに使用する圧縮コーデックを設定します。

タイプ:文字列

可能な値:

- `null` — No compression
- `deflate` — Compress with Deflate (zlib)
- `snappy` — Compress with [スナッピー](#)

デフォルト値: `snappy` (利用可能な場合) または `deflate`.

## output\_format\_avro\_sync\_interval

出力Avroファイルの同期マーカー間の最小データサイズ(バイト単位)を設定します。

型:unsigned int

使用可能な値:32(32バイト)-1073741824(1GiB)

デフォルト値:32768(32KiB)

## format\_avro\_schema\_registry\_url

使用するConfluentスキーマレジストリURLを設定します [アプロコンフルエント](#) 形式

タイプ:URL

既定値:空

## background\_pool\_size

セットのスレッド数を行う背景事業のテーブルエンジン (例えば、合併に [MergeTreeエンジン](#) テーブル)。この設定はClickHouseサーバーの起動時に適用され、ユーザーセッションでは変更できません。この設定を調整することで、CPUとディスクの負荷を管理します。小さなプールサイズを以下のCPUやディスクの資源が背景のプロセスの事前の遅れが影響をクエリす。

可能な値:

- 任意の正の整数。

デフォルト値は16です。

[元の記事](#)

# ClickHouseユーティリティ

- [ツツイ姪"ツ債ツケ](#) — Allows running SQL queries on data without stopping the ClickHouse server, similar to how `awk` これを行います。
- [クリックハウス-複写機](#) — Copies (and reshards) data from one cluster to another cluster.
- [clickhouse-ベンチマーク](#) — Loads server with the custom queries and settings.

# クリックハウス-複写機

コピーデータからのテーブルを一つクラスターテーブルの他の同クラスター

複数実行できます `clickhouse-copier` インスタンスの異なるサーバーを行う仕事です。 ZooKeeperはプロセスの同期に使用されます。

開始後, `clickhouse-copier`:

- ZooKeeperに接続し、受信します:
  - ジョブのコピー。
  - コピージョブの状態。
- これは、ジョブを実行します。

各実行中のプロセスは、“closest”ザ-シャーのソースクラスターのデータ転送先のクラスター `resharding` 場合はそのデータが必要です。

`clickhouse-copier` ZooKeeperの変更を追跡し、その場でそれらを適用します。

ネットワークトライ `clickhouse-copier` ソースデータが配置されている同じサーバー上。

## クリックハウスコピー機の実行

このユーテ:

```
$ clickhouse-copier copier --daemon --config zookeeper.xml --task-path /task/path --base-dir /path/to/dir
```

パラメータ:

- `daemon` — Starts `clickhouse-copier` デーモンモードで。
- `config` — The path to the `zookeeper.xml` ZooKeeperへの接続のためのパラメータを持つファイル。
- `task-path` — The path to the ZooKeeper node. This node is used for syncing `clickhouse-copier` プロセスとタスクの保存。タスクは `$task-path/description`.
- `task-file` — Optional path to file with task configuration for initial upload to ZooKeeper.
- `task-upload-force` — Force upload `task-file` ノードが既に存在していても。
- `base-dir` — The path to logs and auxiliary files. When it starts, `clickhouse-copier` 作成 `clickhouse-copier_YYYYMMHHSS_<PID>` サブディレクトリ `$base-dir`. このパラメーターを省略すると、ディレクトリは次の場所に作成されます `clickhouse-copier` 発売された。

## 飼育係の形式。xml

```

<clickhouse>
  <logger>
    <level>trace</level>
    <size>100M</size>
    <count>3</count>
  </logger>

  <zookeeper>
    <node index="1">
      <host>127.0.0.1</host>
      <port>2181</port>
    </node>
  </zookeeper>
</clickhouse>

```

## コピー一タスクの構成

```

<clickhouse>
  <!-- Configuration of clusters as in an ordinary server config -->
  <remote_servers>
    <source_cluster>
      <shard>
        <internal_replication>false</internal_replication>
        <replica>
          <host>127.0.0.1</host>
          <port>9000</port>
        </replica>
      </shard>
      ...
    </source_cluster>

    <destination_cluster>
    ...
  </destination_cluster>
  </remote_servers>

  <!-- How many simultaneously active workers are possible. If you run more workers superfluous workers will sleep.
-->
  <max_workers>2</max_workers>

  <!-- Setting used to fetch (pull) data from source cluster tables -->
  <settings_pull>
    <readonly>1</readonly>
  </settings_pull>

  <!-- Setting used to insert (push) data to destination cluster tables -->
  <settings_push>
    <readonly>0</readonly>
  </settings_push>

  <!-- Common setting for fetch (pull) and insert (push) operations. Also, copier process context uses it.
      They are overlaid by <settings_pull/> and <settings_push/> respectively. -->
  <settings>
    <connect_timeout>3</connect_timeout>
    <!-- Sync insert is set forcibly, leave it here just in case. -->
    <insert_distributed_sync>1</insert_distributed_sync>
  </settings>

  <!-- Copying tasks description.
      You could specify several table task in the same task description (in the same ZooKeeper node), they will be
      performed
      sequentially.
-->
  <tables>
    <!-- A table task, copies one table. -->
    <table_hits>
      <!-- Source cluster name (from <remote_servers/> section) and tables in it that should be copied -->
      <cluster_pull>source_cluster</cluster_pull>
      <database_pull>test</database_pull>
      <table_pull>hits</table_pull>

      <!-- Destination cluster name and tables in which the data should be inserted -->
      <cluster_push>destination_cluster</cluster_push>
    </table_hits>
  </tables>
</clickhouse>

```

```

<database_push>test</database_push>
<table_push>hits2</table_push>

<!-- Engine of destination tables.
    If destination tables have not be created, workers create them using columns definition from source tables
and engine
    definition from here.

    NOTE: If the first worker starts insert data and detects that destination partition is not empty then the
partition will
        be dropped and refilled, take it into account if you already have some data in destination tables. You could
directly
        specify partitions that should be copied in <enabled_partitions/>, they should be in quoted format like
partition column of
        system.parts table.

-->
<engine>
ENGINE=ReplicatedMergeTree('/clickhouse/tables/{cluster}/{shard}/hits2', '{replica}')
PARTITION BY toMonday(date)
ORDER BY (CounterID, EventDate)
</engine>

<!-- Sharding key used to insert data to destination cluster -->
<sharding_key>jumpConsistentHash(intHash64(userID), 2)</sharding_key>

<!-- Optional expression that filter data while pull them from source servers -->
<where_condition>CounterID != 0</where_condition>

<!-- This section specifies partitions that should be copied, other partition will be ignored.
    Partition names should have the same format as
    partition column of system.parts table (i.e. a quoted text).
    Since partition key of source and destination cluster could be different,
    these partition names specify destination partitions.

    NOTE: In spite of this section is optional (if it is not specified, all partitions will be copied),
it is strictly recommended to specify them explicitly.
    If you already have some ready partitions on destination cluster they
will be removed at the start of the copying since they will be interpreted
as unfinished data from the previous copying!!!
-->
<enabled_partitions>
    <partition>'2018-02-26'</partition>
    <partition>'2018-03-05'</partition>
    ...
</enabled_partitions>
</table_hits>

<!-- Next table to copy. It is not copied until previous table is copying. -->
</table_visits>
...
</table_visits>
...
</tables>
</clickhouse>

```

`clickhouse-copier` の変更を追跡します `/task/path/description` そしてその場でそれらを適用します。たとえば、次の値を変更すると `max_workers`、タスクを実行しているプロセスの数も変更されます。

## ツツイツ姪"ツ債ツツケ

その `clickhouse-local` プログラムで行う高速処理が地元のファイルを展開に合わせて構成されています ClickHouse サーバーです。

データを受け入れを表すテーブル、クエリを利用して [クリックハウスSQL方言](#)。

`clickhouse-local` ClickHouse serverと同じコアを使用するため、ほとんどの機能と同じフォーマットとテーブルエンジンのセットをサポートします。

既定では `clickhouse-local` なアクセスデータを同じホストにも対応して搭載サーバの設定を使用 `--config-file` 引数。

## 警告

運用サーバーの構成をロードすることはお勧めしません `clickhouse-local` ヒューマンエラーの場合にデータが破損する可能性があるため。

## 使用法

基本的な使用法:

```
$ clickhouse-local --structure "table_structure" --input-format "format_of_incoming_data" -q "query"
```

引数:

- `-S, --structure` — table structure for input data.
- `-if, --input-format` — input format, `TSV` デフォルトでは。
- `-f, --file` — path to data, `stdin` デフォルトでは。
- `-q --query` — queries to execute with ; デリミテーターとして。
- `-N, --table` — table name where to put output data, `table` デフォルトでは。
- `-of, --format, --output-format` — output format, `TSV` デフォルトでは。
- `--stacktrace` — whether to dump debug output in case of exception.
- `--verbose` — more details on query execution.
- `-s` — disables `stderr` ロギング
- `--config-file` — path to configuration file in same format as for ClickHouse server, by default the configuration empty.
- `--help` — arguments references for `clickhouse-local`.

また、ClickHouse設定変数ごとに引数があります。`--config-file`.

## 例

```
$ echo -e "1,2\n3,4" | clickhouse-local -S "a Int64, b Int64" -if "CSV" -q "SELECT * FROM table"  
Read 2 rows, 32.00 B in 0.000 sec., 5182 rows/sec., 80.97 KiB/sec.  
1 2  
3 4
```

前の例と同じです:

```
$ echo -e "1,2\n3,4" | clickhouse-local -q "CREATE TABLE table (a Int64, b Int64) ENGINE = File(CSV, stdin); SELECT a, b FROM table; DROP TABLE table"  
Read 2 rows, 32.00 B in 0.000 sec., 4987 rows/sec., 77.93 KiB/sec.  
1 2  
3 4
```

次に、各Unixユーザのメモリユーザを出力しましょう:

```
$ ps aux | tail -n +2 | awk '{ printf("%s\t%s\n", $1, $4) }' | clickhouse-local -S "user String, mem Float64" -q "SELECT user, round(sum(mem), 2) as memTotal FROM table GROUP BY user ORDER BY memTotal DESC FORMAT Pretty"
```

```
Read 186 rows, 4.15 KiB in 0.035 sec., 5302 rows/sec., 118.34 KiB/sec.
```

user	memTotal
bayonet	113.5
root	8.8
...	

## clickhouse-ベンチマーク

ClickHouseサーバーに接続し、指定したクエリを繰り返し送信します。

構文:

```
$ echo "single query" | clickhouse-benchmark [keys]
```

または

```
$ clickhouse-benchmark [keys] <<< "single query"
```

一連のクエリを送信する場合は、テキストファイルを作成し、各クエリをこのファイル内の個々の文字列に配置します。例えば:

```
SELECT * FROM system.numbers LIMIT 10000000
SELECT 1
```

次に、このファイルを標準入力に渡します。 `clickhouse-benchmark`.

```
clickhouse-benchmark [keys] < queries_file
```

### キー

- `-c N, --concurrency=N` — Number of queries that `clickhouse-benchmark` 同時に送信します。 デフォルト値:1。
- `-d N, --delay=N` — Interval in seconds between intermediate reports (set 0 to disable reports). Default value: 1.
- `-h WORD, --host=WORD` — Server host. Default value: `localhost`. のために 比較モード 複数を使用できます `-h` 鍵だ
- `-p N, --port=N` — Server port. Default value: 9000. For the 比較モード 複数を使用できます `-p` 鍵だ
- `-i N, --iterations=N` — Total number of queries. Default value: 0.
- `-r, --randomize` — Random order of queries execution if there is more then one input query.
- `-s, --secure` — Using TLS connection.
- `-t N, --timelimit=N` — Time limit in seconds. `clickhouse-benchmark` 指定された時間制限に達すると、クエリの送信を停止します。 デフォルト値:0(制限時間無効)。
- `--confidence=N` — Level of confidence for T-test. Possible values: 0 (80%), 1 (90%), 2 (95%), 3 (98%), 4 (99%), 5 (99.5%). Default value: 5. In the 比較モード `clickhouse-benchmark` を実行します。 独立した二つのサンプル学生のtテスト これらの分布が選択された信頼度と異ならないかどうかを判定します。

- `--cumulative` — Printing cumulative data instead of data per interval.
- `--database=DATABASE_NAME` — ClickHouse database name. Default value: `default`.
- `--json=FILEPATH` — JSON output. When the key is set, `clickhouse-benchmark` 指定されたJSONファイルにレポートを出力します。
- `--user=USERNAME` — ClickHouse user name. Default value: `default`.
- `--password=PSWD` — ClickHouse user password. Default value: empty string.
- `--stacktrace` — Stack traces output. When the key is set, `clickhouse-benchmark` 出力スタックトレースの例外をスローしました。
- `--stage=WORD` — Query processing stage at server. ClickHouse stops query processing and returns answer to `clickhouse-benchmark` 指定された段階で。 可能な値: `complete`, `fetch_columns`, `with_mergeable_state`. デフォルト値: `complete`.
- `--help` — Shows the help message.

あなたはいくつかを適用したい場合 **設定** クエリの場合は、キーとして渡します `--<session setting name>=SETTING_VALUE`. 例えば, `--max_memory_usage=1048576`.

## 出力

既定では, `clickhouse-benchmark` 各レポート `--delay` インターバル

レポートの例:

Queries executed: 10.

localhost:9000, queries 10, QPS: 6.772, RPS: 67904487.440, MiB/s: 518.070, result RPS: 67721584.984, result MiB/s: 516.675.

```
0.000% 0.145 sec.
10.000% 0.146 sec.
20.000% 0.146 sec.
30.000% 0.146 sec.
40.000% 0.147 sec.
50.000% 0.148 sec.
60.000% 0.148 sec.
70.000% 0.148 sec.
80.000% 0.149 sec.
90.000% 0.150 sec.
95.000% 0.150 sec.
99.000% 0.150 sec.
99.900% 0.150 sec.
99.990% 0.150 sec.
```

このレポートでは:

- クエリの数 `Queries executed:` フィールド

- ステータス文字列を含む(順):
  - ClickHouseサーバーのエンドポイント。
  - 処理されたクエリの数。
  - QPS:QPS:で指定された期間に毎秒実行されるクエリサーバーの数 `--delay` 引数。
  - RPS:サーバーが指定された期間に毎秒読み取った行数 `--delay` 引数。
  - MiB/s:指定された期間に毎秒読み取られるmebibytesサーバーの数 `--delay` 引数。
  - 結果RPS:サーバーによって指定された期間内にクエリの結果に対して毎秒どのくらいの行が配置されますか `--delay` 引数。
  - クエリの結果に対してサーバによって指定された期間内に秒あたりのメビバイト数 `--delay` 引数。
- クエリの実行時間の百分位数。

## 比較モード

`clickhouse-benchmark` ツツイツ姪"ツツ"ツ債ツづュツケ

比較モードを使用するには、両方のサーバーのエンドポイントを `--host`, `--port` 鍵だけ引数リスト内の位置によって一致するキーは、最初の `-host` 最初のものと一致します `--port` など。`clickhouse-benchmark` 両方のサーバーへの接続を確立し、クエリを送信します。ランダムに選択されたサーバー宛の各クエリ。結果は、各サーバーごとに個別に表示されます。

### 例

```
$ echo "SELECT * FROM system.numbers LIMIT 10000000 OFFSET 10000000" | clickhouse-benchmark -i 10
```

Loaded 1 queries.

Queries executed: 6.

localhost:9000, queries 6, QPS: 6.153, RPS: 123398340.957, MiB/s: 941.455, result RPS: 61532982.200, result MiB/s: 469.459.

```
0.000% 0.159 sec.  
10.000% 0.159 sec.  
20.000% 0.159 sec.  
30.000% 0.160 sec.  
40.000% 0.160 sec.  
50.000% 0.162 sec.  
60.000% 0.164 sec.  
70.000% 0.165 sec.  
80.000% 0.166 sec.  
90.000% 0.166 sec.  
95.000% 0.167 sec.  
99.000% 0.167 sec.  
99.900% 0.167 sec.  
99.990% 0.167 sec.
```

Queries executed: 10.

localhost:9000, queries 10, QPS: 6.082, RPS: 121959604.568, MiB/s: 930.478, result RPS: 60815551.642, result MiB/s: 463.986.

```
0.000% 0.159 sec.  
10.000% 0.159 sec.  
20.000% 0.160 sec.  
30.000% 0.163 sec.  
40.000% 0.164 sec.  
50.000% 0.165 sec.  
60.000% 0.166 sec.  
70.000% 0.166 sec.  
80.000% 0.167 sec.  
90.000% 0.167 sec.  
95.000% 0.170 sec.  
99.000% 0.172 sec.  
99.900% 0.172 sec.  
99.990% 0.172 sec.
```

## clickhouse-format

Allows formatting input queries.

Keys:

- `--help` or `-h` — Produce help message.
- `--hilite` — Add syntax highlight with ANSI terminal escape sequences.
- `--oneline` — Format in single line.
- `--quiet` or `-q` — Just check syntax, no output on success.
- `--multiquery` or `-n` — Allow multiple queries in the same file.
- `--obfuscate` — Obfuscate instead of formatting.
- `--seed <string>` — Seed arbitrary string that determines the result of obfuscation.
- `--backslash` — Add a backslash at the end of each line of the formatted query. Can be useful when you copy a query from web or somewhere else with multiple lines, and want to execute it in command line.

# Examples

1. Highlighting and single line:

```
$ clickhouse-format --oneline --hilite <<< "SELECT sum(number) FROM numbers(5);"
```

Result:

```
SELECT sum(number) FROM numbers(5)
```

2. Multiqueries:

```
$ clickhouse-format -n <<< "SELECT * FROM (SELECT 1 AS x UNION ALL SELECT 1 UNION DISTINCT SELECT 3);"
```

Result:

```
SELECT *
FROM
(
    SELECT 1 AS x
    UNION ALL
    SELECT 1
    UNION DISTINCT
    SELECT 3
)
;
```

3. Obfuscating:

```
$ clickhouse-format --seed Hello --obfuscate <<< "SELECT cost_first_screen BETWEEN a AND b, CASE WHEN x >= 123 THEN y ELSE NULL END;"
```

Result:

```
SELECT treasury_mammoth_hazelnut BETWEEN nutmeg AND span, CASE WHEN chive >= 116 THEN switching ELSE ANYTHING END;
```

Same query and another seed string:

```
$ clickhouse-format --seed World --obfuscate <<< "SELECT cost_first_screen BETWEEN a AND b, CASE WHEN x >= 123 THEN y ELSE NULL END;"
```

Result:

```
SELECT horse_tape_summer BETWEEN folklore AND moccasins, CASE WHEN intestine >= 116 THEN nonconformist ELSE FORESTRY END;
```

4. Adding backslash:

```
$ clickhouse-format --backslash <<< "SELECT * FROM (SELECT 1 AS x UNION ALL SELECT 1 UNION DISTINCT SELECT 3);"
```

Result:

```
SELECT * \
FROM \
(\ \
  SELECT 1 AS x \
UNION ALL \
  SELECT 1 \
UNION DISTINCT \
  SELECT 3 \
)
```

## ClickHouse compressor

Simple program for data compression and decompression.

### Examples

Compress data with LZ4:

```
$ ./clickhouse-compressor < input_file > output_file
```

Decompress data from LZ4 format:

```
$ ./clickhouse-compressor --decompress < input_file > output_file
```

Compress data with ZSTD at level 5:

```
$ ./clickhouse-compressor --codec 'ZSTD(5)' < input_file > output_file
```

Compress data with Delta of four bytes and ZSTD level 10.

```
$ ./clickhouse-compressor --codec 'Delta(4)' --codec 'ZSTD(10)' < input_file > output_file
```

## ClickHouse obfuscator

A simple tool for table data obfuscation.

It reads an input table and produces an output table, that retains some properties of input, but contains different data.

It allows publishing almost real production data for usage in benchmarks.

It is designed to retain the following properties of data:

- cardinalities of values (number of distinct values) for every column and every tuple of columns;
- conditional cardinalities: number of distinct values of one column under the condition on the value of another column;
- probability distributions of the absolute value of integers; the sign of signed integers; exponent and sign for floats;
- probability distributions of the length of strings;
- probability of zero values of numbers; empty strings and arrays, `NULLs`;
- data compression ratio when compressed with LZ77 and entropy family of codecs;
- continuity (magnitude of difference) of time values across the table; continuity of floating-point values;
- date component of `DateTime` values;

- UTF-8 validity of string values;
- string values look natural.

Most of the properties above are viable for performance testing:

reading data, filtering, aggregation, and sorting will work at almost the same speed as on original data due to saved cardinalities, magnitudes, compression ratios, etc.

It works in a deterministic fashion: you define a seed value and the transformation is determined by input data and by seed.

Some transformations are one to one and could be reversed, so you need to have a large seed and keep it in secret.

It uses some cryptographic primitives to transform data but from the cryptographic point of view, it does not do it properly, that is why you should not consider the result as secure unless you have another reason. The result may retain some data you don't want to publish.

It always leaves 0, 1, -1 numbers, dates, lengths of arrays, and null flags exactly as in source data. For example, you have a column `IsMobile` in your table with values 0 and 1. In transformed data, it will have the same value.

So, the user will be able to count the exact ratio of mobile traffic.

Let's give another example. When you have some private data in your table, like user email and you don't want to publish any single email address.

If your table is large enough and contains multiple different emails and no email has a very high frequency than all others, it will anonymize all data. But if you have a small number of different values in a column, it can reproduce some of them.

You should look at the working algorithm of this tool works, and fine-tune its command line parameters.

This tool works fine only with an average amount of data (at least 1000s of rows).

---

## clickhouse-odbc-bridge

Simple HTTP-server which works like a proxy for ODBC driver. The main motivation was possible segfaults or another faults in ODBC implementations, which can crash whole clickhouse-server process.

This tool works via HTTP, not via pipes, shared memory, or TCP because:

- It's simpler to implement
- It's simpler to debug
- jdbc-bridge can be implemented in the same way

## Usage

`clickhouse-server` use this tool inside odbc table function and StorageODBC.

However it can be used as standalone tool from command line with the following parameters in POST-request URL:

- `connection_string` -- ODBC connection string.
  - `columns` -- columns in ClickHouse NamesAndTypesList format, name in backticks, type as string. Name and type are space separated, rows separated with newline.
  - `max_block_size` -- optional parameter, sets maximum size of single block.
- Query is send in post body. Response is returned in RowBinary format.

## Example:

```
$ clickhouse-odbc-bridge --http-port 9018 --daemon

$ curl -d "query=SELECT PageID, ImplID, AdType FROM Keys ORDER BY PageID, ImplID" --data-urlencode
"connection_string=DSN=ClickHouse;DATABASE=stat" --data-urlencode "columns=columns format version: 1
3 columns:
\`PageID\` String
\`ImplID\` String
\`AdType\` String
" "http://localhost:9018/" > result.txt

$ cat result.txt
12246623837185725195925621517
```

## 使用法の推奨事項

### CPUのスケール知事

常に `performance` スケーリング知事。その `on-demand` スケーリング総裁の作品もないと常に高いです。

```
$ echo 'performance' | sudo tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
```

### CPUの制限

プロセッサでの過熱を防止します。使用 `dmesg cpu` のクロックレートが過熱により制限されているかどうかを確認します。

制限は、データセンターレベルで外部で設定することもできます。以下を使用できます `turbostat` 負荷の下でそれを監視する。

### RAM

少量のデータ（最大200GB圧縮）の場合は、データ量と同じくらいのメモリを使用することをお勧めします。

大量のデータの場合、および対話型(オンライン)クエリを処理する場合は、ホットデータサブセットがページのキャッシュに収まるように、妥当な量のRAM(128GB)

でもデータ量の50TBサーバ用のもの128GB RAMを大幅に向上的クエリの性能に比べて64GBにサンプルがあります。

オーバーコミットを無効にしません。値 `cat /proc/sys/vm/overcommit_memory 0` または `1` にする必要があります。走れ。

```
$ echo 0 | sudo tee /proc/sys/vm/overcommit_memory
```

### 巨大なページ

常に透明な巨大ページを無効にします。これはメモリアロケータと干渉し、パフォーマンスが大幅に低下します。

```
$ echo 'never' | sudo tee /sys/kernel/mm/transparent_hugepage/enabled
```

使用 `perf top` メモリ管理のためにカーネルで費やされた時間を監視する。

永続的な巨大なページも割り当てる必要はありません。

### Storageサブシステム

予算でSSDを使用できる場合は、SSDを使用してください。

そうでない場合は、HDDを使用します。SATA Hdd7200RPMが行います。

優先のサーバー地域のハードディスク上に小さな複数のサーバーが付属ディスクが増す。ものの保存アーカイブでクエリー、棚します。

## RAID

HDDを使用する場合は、RAID-10、RAID-5、RAID-6またはRAID-50を組み合わせることができます。

Linuxでは、ソフトウェアRAIDが優れています（`mdadm`）。LVMの使用はお勧めしません。

RAID-10を作成するときは、`far` レイアウト。

予算が許せば、RAID-10を選択します。

4つ以上のディスクがある場合は、RAID-6(優先)またはRAID-50を使用してください。

RAID-5、RAID-6、またはRAID-50を使用する場合は、常に`stripe_cache_size`を増やしてください。

```
$ echo 4096 | sudo tee /sys/block/md2/md/stripe_cache_size
```

式を使用して、デバイスの数とブロックサイズから正確な数を計算します: `2 * num_devices * chunk_size_in_bytes / 4096.`

すべてのRAID構成では、1024KBのブロックサイズで十分です。

ないセットのブロックサイズでは多すぎます。

SSDではRAID-0を使用できます。

に関わらずRAIDの利用、使用複製のためのデータです。

長いキューでNCQを有効にします。 HDDの場合はCFQスケジューラを選択し、SSDの場合はnoopを選択します。 減らさないで下さい ‘readahead’ 設定。

HDDの場合、ライトキャッシュを有効にします。

## ファイルシステム

Ext4は最も信頼性の高いオプションです。 マウントオプションの設定 `noatime`.

XFSも適していますが、ClickHouseで徹底的にテストされていません。

他のほとんどのファイルシステム仕様。 ファイルシステムの遅配ます。

## Linuxカーネル

古いLinuxカーネルを使用しないでください。

## ネット

IPv6を使用している場合は、ルートキャッシュのサイズを増やします。

3.2より前のLinuxカーネルには、IPv6実装に関して多くの問題がありました。

可能であれば、10GB以上のネットワークを使用してください。 1Gbも動作しますが、数十テラバイトのデータを使用してレプリカにパッチを適用する場合や、大量の中間データを使用して分散クエリを処理する場合

## 飼育係

おそらく既に他の目的のためにZooKeeperを使用しています。 まだ過負荷になっていない場合は、ZooKeeperと同じインストールを使用できます。

It's best to use a fresh version of ZooKeeper – 3.4.9 or later. The version in stable Linux distributions may be outdated.

異なるZooKeeperクラスター間でデータを転送するには、手動で書かれたスクリプトを使用しないでください。 決して使用しない “zkcopy” 同じ理由でユーティリティ：<https://github.com/ksprojects/zkcopy/issues/15>

既存のZooKeeperクラスタを二つに分割する場合、正しい方法は、そのレプリカの数を増やし、それを二つの独立したクラスタとして再構成することです。

ClickHouseと同じサーバー上でZooKeeperを実行しないでください。で飼育係が非常に敏感なために時間遅れとClickHouseを利用することも可能で利用可能なすべてシステム資源です。

デフォルト設定では、飼育係は時限爆弾です:

ZooKeeperサーバーは、デフォルト設定を使用するときに古いスナップショットとログからファイルを削除しません(autopurgeを参照)。

この爆弾は取り除かなければならない

以下の飼育係（3.5.1）設定はYandexで使用されています。月のようMetricaの生産環境20,2017:

動物園cfg:

```
## http://hadoop.apache.org/zookeeper/docs/current/zookeeperAdmin.html

## The number of milliseconds of each tick
tickTime=2000
## The number of ticks that the initial
## synchronization phase can take
initLimit=30000
## The number of ticks that can pass between
## sending a request and getting an acknowledgement
syncLimit=10

maxClientCnxns=2000

maxSessionTimeout=60000000
## the directory where the snapshot is stored.
dataDir=/opt/zookeeper/{{ cluster['name'] }}/data
## Place the dataLogDir to a separate physical disc for better performance
dataLogDir=/opt/zookeeper/{{ cluster['name'] }}/logs

autopurge.snapRetainCount=10
autopurge.purgeInterval=1

## To avoid seeks ZooKeeper allocates space in the transaction log file in
## blocks of preAllocSize kilobytes. The default block size is 64M. One reason
## for changing the size of the blocks is to reduce the block size if snapshots
## are taken more often. (Also, see snapCount).
preAllocSize=131072

## Clients can submit requests faster than ZooKeeper can process them,
## especially if there are a lot of clients. To prevent ZooKeeper from running
## out of memory due to queued requests, ZooKeeper will throttle clients so that
## there is no more than globalOutstandingLimit outstanding requests in the
## system. The default limit is 1,000.ZooKeeper logs transactions to a
## transaction log. After snapCount transactions are written to a log file a
## snapshot is started and a new transaction log file is started. The default
## snapCount is 10,000.
snapCount=3000000

## If this option is defined, requests will be will logged to a trace file named
## traceFile.year.month.day.
##traceFile=

## Leader accepts client connections. Default value is "yes". The leader machine
## coordinates updates. For higher update throughput at the slight expense of
## read throughput the leader can be configured to not accept clients and focus
## on coordination.
leaderServes=yes

standaloneEnabled=false
dynamicConfigFile=/etc/zookeeper-{{ cluster['name'] }}/conf/zoo.cfg.dynamic
```

Javaバージョン:

```
Java(TM) SE Runtime Environment (build 1.8.0_25-b17)
Java HotSpot(TM) 64-Bit Server VM (build 25.25-b02, mixed mode)
```

JVMパラメータ:

```
NAME=zookeeper-{{ cluster['name'] }}
ZOOCFGDIR=/etc/$NAME/conf

## TODO this is really ugly
## How to find out, which jars are needed?
## seems, that log4j requires the log4j.properties file to be in the classpath
CLASSPATH="$ZOOCFGDIR:/usr/build/classes:/usr/build/lib/*.jar:/usr/share/zookeeper/zookeeper-3.5.1-
metrika.jar:/usr/share/zookeeper/slf4j-log4j12-1.7.5.jar:/usr/share/zookeeper/slf4j-api-
1.7.5.jar:/usr/share/zookeeper/servlet-api-2.5-20081211.jar:/usr/share/zookeeper/netty-
3.7.0.Final.jar:/usr/share/zookeeper/log4j-1.2.16.jar:/usr/share/zookeeper/jline-2.11.jar:/usr/share/zookeeper/jetty-util-
6.1.26.jar:/usr/share/zookeeper/jetty-6.1.26.jar:/usr/share/zookeeper/javacc.jar:/usr/share/zookeeper/jackson-mapper-
asl-1.9.11.jar:/usr/share/zookeeper/jackson-core-asl-1.9.11.jar:/usr/share/zookeeper/commons-cli-
1.2.jar:/usr/src/java/lib/*.jar:/usr/etc/zookeeper"

ZOOCFG="$ZOOCFGDIR/zoo.cfg"
ZOO_LOG_DIR=/var/log/$NAME
USER=zookeeper
GROUP=zookeeper
PIDDIR=/var/run/$NAME
PIDFILE=$PIDDIR/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME
JAVA=/usr/bin/java
ZOOMAIN="org.apache.zookeeper.server.quorum.QuorumPeerMain"
ZOO_LOG4J_PROP="INFO,ROLLINGFILE"
JMXLOCALONLY=false
JAVA_OPTS="-Xms{{ cluster.get('xms','128M') }} \
-Xmx{{ cluster.get('xmx','1G') }} \
-Xloggc:/var/log/$NAME/zookeeper-gc.log \
-XX:+UseGCLogFileRotation \
-XX:NumberOfGCLogFiles=16 \
-XX:GCLogFileSize=16M \
-verbose:gc \
-XX:+PrintGCTimeStamps \
-XX:+PrintGCDateStamps \
-XX:+PrintGCDetails \
-XX:+PrintTenuringDistribution \
-XX:+PrintGCApplicationStoppedTime \
-XX:+PrintGCApplicationConcurrentTime \
-XX:+PrintSafepointStatistics \
-XX:+UseParNewGC \
-XX:+UseConcMarkSweepGC \
-XX:+CMSParallelRemarkEnabled"
```

監init:

```

description "zookeeper-{{ cluster['name'] }} centralized coordination service"
start on runlevel [2345]
stop on runlevel [!2345]
respawn
limit nofile 8192 8192
pre-start script
  [ -r "/etc/zookeeper-{{ cluster['name'] }}/conf/environment" ] || exit 0
  . /etc/zookeeper-{{ cluster['name'] }}/conf/environment
  [ -d $ZOO_LOG_DIR ] || mkdir -p $ZOO_LOG_DIR
  chown $USER:$GROUP $ZOO_LOG_DIR
end script

script
  . /etc/zookeeper-{{ cluster['name'] }}/conf/environment
  [ -r /etc/default/zookeeper ] && . /etc/default/zookeeper
  if [ -z "$JMXDISABLE" ]; then
    JAVA_OPTS="$JAVA_OPTS -Dcom.sun.management.jmxremote -
Dcom.sun.management.jmxremote.local.only=$JMXLOCALONLY"
    fi
  exec start-stop-daemon --start -c $USER --exec $JAVA --name zookeeper-{{ cluster['name'] }} \
--cp $CLASSPATH $JAVA_OPTS -Dzookeeper.log.dir=${ZOO_LOG_DIR} \
-Dzookeeper.root.logger=${ZOO_LOG4J_PROP} $ZOOMAIN $ZOOCFG
end script

```

## [experimental] OpenTelemetry Support

[OpenTelemetry](#) is an open standard for collecting traces and metrics from the distributed application. ClickHouse has some support for OpenTelemetry.

### Warning

This is an experimental feature that will change in backwards-incompatible ways in future releases.

## Supplying Trace Context to ClickHouse

ClickHouse accepts trace context HTTP headers, as described by the [W3C recommendation](#). It also accepts trace context over a native protocol that is used for communication between ClickHouse servers or between the client and server. For manual testing, trace context headers conforming to the Trace Context recommendation can be supplied to `clickhouse-client` using `--opentelemetry-traceparent` and `--opentelemetry-tracestate` flags.

If no parent trace context is supplied, ClickHouse can start a new trace, with probability controlled by the `opentelemetry_start_trace_probability` setting.

## Propagating the Trace Context

The trace context is propagated to downstream services in the following cases:

- Queries to remote ClickHouse servers, such as when using [Distributed](#) table engine.
- `url` table function. Trace context information is sent in HTTP headers.

## Tracing the ClickHouse Itself

ClickHouse creates trace spans for each query and some of the query execution stages, such as query planning or distributed queries.

To be useful, the tracing information has to be exported to a monitoring system that supports OpenTelemetry, such as [Jaeger](#) or [Prometheus](#). ClickHouse avoids a dependency on a particular monitoring system, instead only providing the tracing data through a system table. OpenTelemetry trace span information required by the standard is stored in the `system.opentelemetry_span_log` table.

The table must be enabled in the server configuration, see the `opentelemetry_span_log` element in the default config file `config.xml`. It is enabled by default.

The tags or attributes are saved as two parallel arrays, containing the keys and values. Use [ARRAY JOIN](#) to work with them.

## Integration with monitoring systems

At the moment, there is no ready tool that can export the tracing data from ClickHouse to a monitoring system.

For testing, it is possible to setup the export using a materialized view with the [URL](#) engine over the `system.opentelemetry_span_log` table, which would push the arriving log data to an HTTP endpoint of a trace collector. For example, to push the minimal span data to a Zipkin instance running at `http://localhost:9411`, in Zipkin v2 JSON format:

```
CREATE MATERIALIZED VIEW default.zipkin_spans
ENGINE = URL('http://127.0.0.1:9411/api/v2/spans', 'JSONEachRow')
SETTINGS output_format_json_named_tuples_as_objects = 1,
    output_format_json_array_of_rows = 1 AS
SELECT
    lower(hex(reinterpretAsFixedString(trace_id))) AS traceId,
    lower(hex(parent_span_id)) AS parentId,
    lower(hex(span_id)) AS id,
    operation_name AS name,
    start_time_us AS timestamp,
    finish_time_us - start_time_us AS duration,
    cast(tuple('clickhouse'), 'Tuple(serviceName text)') AS localEndpoint,
    cast(tuple(
        attribute.values[indexOf(attribute.names, 'db.statement')]),
        'Tuple("db.statement" text)') AS tags
FROM system.opentelemetry_span_log
```

In case of any errors, the part of the log data for which the error has occurred will be silently lost. Check the server log for error messages if the data does not arrive.

## Cache Types

When performing queries, ClickHouse uses different caches.

Main cache types:

- `mark_cache` — Cache of marks used by table engines of the [MergeTree](#) family.
- `uncompressed_cache` — Cache of uncompressed data used by table engines of the [MergeTree](#) family.

Additional cache types:

- DNS cache.
- [Regexp](#) cache.
- Compiled expressions cache.
- [Avro format](#) schemas cache.
- [Dictionaries](#) data cache.

Indirectly used:

- OS page cache.

To drop cache, use `SYSTEM DROP ... CACHE` statements.

## [pre-production] ClickHouse Keeper

ClickHouse server uses [ZooKeeper](#) coordination system for data [replication](#) and [distributed DDL](#) queries execution. ClickHouse Keeper is an alternative coordination system compatible with ZooKeeper.

### Warning

This feature is currently in the pre-production stage. We test it in our CI and on small internal installations.

## Implementation details

ZooKeeper is one of the first well-known open-source coordination systems. It's implemented in Java, has quite a simple and powerful data model. ZooKeeper's coordination algorithm called ZAB (ZooKeeper Atomic Broadcast) doesn't provide linearizability guarantees for reads, because each ZooKeeper node serves reads locally. Unlike ZooKeeper ClickHouse Keeper is written in C++ and uses [RAFT algorithm implementation](#). This algorithm allows to have linearizability for reads and writes, has several open-source implementations in different languages.

By default, ClickHouse Keeper provides the same guarantees as ZooKeeper (linearizable writes, non-linearizable reads). It has a compatible client-server protocol, so any standard ZooKeeper client can be used to interact with ClickHouse Keeper. Snapshots and logs have an incompatible format with ZooKeeper, but `clickhouse-keeper-converter` tool allows to convert ZooKeeper data to ClickHouse Keeper snapshot. Interserver protocol in ClickHouse Keeper is also incompatible with ZooKeeper so mixed ZooKeeper / ClickHouse Keeper cluster is impossible.

## Configuration

ClickHouse Keeper can be used as a standalone replacement for ZooKeeper or as an internal part of the ClickHouse server, but in both cases configuration is almost the same `.xml` file. The main ClickHouse Keeper configuration tag is `<keeper_server>`. Keeper configuration has the following parameters:

- `tcp_port` — Port for a client to connect (default for ZooKeeper is `2181`).
- `tcp_port_secure` — Secure port for a client to connect.
- `server_id` — Unique server id, each participant of the ClickHouse Keeper cluster must have a unique number (1, 2, 3, and so on).
- `log_storage_path` — Path to coordination logs, better to store logs on the non-busy device (same for ZooKeeper).
- `snapshot_storage_path` — Path to coordination snapshots.

Other common parameters are inherited from the ClickHouse server config (`listen_host`, `logger`, and so on).

Internal coordination settings are located in `<keeper_server>. <coordination_settings>` section:

- `operation_timeout_ms` — Timeout for a single client operation (ms) (default: 10000).
- `session_timeout_ms` — Timeout for client session (ms) (default: 30000).

- `dead_session_check_period_ms` — How often ClickHouse Keeper check dead sessions and remove them (ms) (default: 500).
- `heart_beat_interval_ms` — How often a ClickHouse Keeper leader will send heartbeats to followers (ms) (default: 500).
- `election_timeout_lower_bound_ms` — If the follower didn't receive heartbeats from the leader in this interval, then it can initiate leader election (default: 1000).
- `election_timeout_upper_bound_ms` — If the follower didn't receive heartbeats from the leader in this interval, then it must initiate leader election (default: 2000).
- `rotate_log_storage_interval` — How many log records to store in a single file (default: 100000).
- `reserved_log_items` — How many coordination log records to store before compaction (default: 100000).
- `snapshot_distance` — How often ClickHouse Keeper will create new snapshots (in the number of records in logs) (default: 100000).
- `snapshots_to_keep` — How many snapshots to keep (default: 3).
- `stale_log_gap` — Threshold when leader considers follower as stale and sends the snapshot to it instead of logs (default: 10000).
- `fresh_log_gap` — When node became fresh (default: 200).
- `max_requests_batch_size` - Max size of batch in requests count before it will be sent to RAFT (default: 100).
- `force_sync` — Call `fsync` on each write to coordination log (default: true).
- `quorum_reads` — Execute read requests as writes through whole RAFT consensus with similar speed (default: false).
- `raft_logs_level` — Text logging level about coordination (trace, debug, and so on) (default: system default).
- `auto_forwarding` — Allow to forward write requests from followers to the leader (default: true).
- `shutdown_timeout` — Wait to finish internal connections and shutdown (ms) (default: 5000).
- `startup_timeout` — If the server doesn't connect to other quorum participants in the specified timeout it will terminate (ms) (default: 30000).

Quorum configuration is located in `<keeper_server>.<raft_configuration>` section and contain servers description. The only parameter for the whole quorum is `secure`, which enables encrypted connection for communication between quorum participants. The main parameters for each `<server>` are:

- `id` — Server identifier in a quorum.
- `hostname` — Hostname where this server is placed.
- `port` — Port where this server listens for connections.

Examples of configuration for quorum with three nodes can be found in [integration tests](#) with `test_keeper_` prefix. Example configuration for server #1:

```

<keeper_server>
  <tcp_port>2181</tcp_port>
  <server_id>1</server_id>
  <log_storage_path>/var/lib/clickhouse/coordination/log</log_storage_path>
  <snapshot_storage_path>/var/lib/clickhouse/coordination/snapshots</snapshot_storage_path>

  <coordination_settings>
    <operation_timeout_ms>10000</operation_timeout_ms>
    <session_timeout_ms>30000</session_timeout_ms>
    <raft_logs_level>trace</raft_logs_level>
  </coordination_settings>

  <raft_configuration>
    <server>
      <id>1</id>
      <hostname>zoo1</hostname>
      <port>9444</port>
    </server>
    <server>
      <id>2</id>
      <hostname>zoo2</hostname>
      <port>9444</port>
    </server>
    <server>
      <id>3</id>
      <hostname>zoo3</hostname>
      <port>9444</port>
    </server>
  </raft_configuration>
</keeper_server>

```

## How to run

ClickHouse Keeper is bundled into the ClickHouse server package, just add configuration of `<keeper_server>` and start ClickHouse server as always. If you want to run standalone ClickHouse Keeper you can start it in a similar way with:

```
clickhouse-keeper --config /etc/your_path_to_config/config.xml --daemon
```

## [experimental] Migration from ZooKeeper

Seamlessly migration from ZooKeeper to ClickHouse Keeper is impossible you have to stop your ZooKeeper cluster, convert data and start ClickHouse Keeper. `clickhouse-keeper-converter` tool allows converting ZooKeeper logs and snapshots to ClickHouse Keeper snapshot. It works only with ZooKeeper > 3.4. Steps for migration:

1. Stop all ZooKeeper nodes.
2. Optional, but recommended: find ZooKeeper leader node, start and stop it again. It will force ZooKeeper to create a consistent snapshot.
3. Run `clickhouse-keeper-converter` on a leader, for example:

```
clickhouse-keeper-converter --zookeeper-logs-dir /var/lib/zookeeper/version-2 --zookeeper-snapshots-dir /var/lib/zookeeper/version-2 --output-dir /path/to/clickhouse/keeper/snapshots
```

4. Copy snapshot to ClickHouse server nodes with a configured `keeper` or start ClickHouse Keeper instead of ZooKeeper. The snapshot must persist on all nodes, otherwise, empty nodes can be faster and one of them can become a leader.

## External Disks for Storing Data

Data, processed in ClickHouse, is usually stored in the local file system — on the same machine with the ClickHouse server. That requires large-capacity disks, which can be expensive enough. To avoid that you can store the data remotely — on [Amazon S3](#) disks or in the Hadoop Distributed File System ([HDFS](#)).

To work with data stored on [Amazon S3](#) disks use [S3](#) table engine, and to work with data in the Hadoop Distributed File System — [HDFS](#) table engine.

To load data from a web server with static files use a disk with type [web](#).

## Zero-copy Replication

ClickHouse supports zero-copy replication for [S3](#) and [HDFS](#) disks, which means that if the data is stored remotely on several machines and needs to be synchronized, then only the metadata is replicated (paths to the data parts), but not the data itself.

## Configuring HDFS

[MergeTree](#) and [Log](#) family table engines can store data to HDFS using a disk with type [HDFS](#).

Configuration markup:

```
<clickhouse>
  <storage_configuration>
    <disks>
      <hdfs>
        <type>hdfs</type>
        <endpoint>hdfs://hdfs1:9000/clickhouse/</endpoint>
      </hdfs>
    </disks>
    <policies>
      <hdfs>
        <volumes>
          <main>
            <disk>hdfs</disk>
          </main>
        </volumes>
      </hdfs>
    </policies>
  </storage_configuration>

  <merge_tree>
    <min_bytes_for_wide_part>0</min_bytes_for_wide_part>
  </merge_tree>
</clickhouse>
```

Required parameters:

- `endpoint` — HDFS endpoint URL in `path` format. Endpoint URL should contain a root path to store data.

Optional parameters:

- `min_bytes_for_seek` — The minimal number of bytes to use seek operation instead of sequential read.  
Default value: `1 Mb`.

## Using Virtual File System for Data Encryption

You can encrypt the data stored on [S3](#), or [HDFS](#) external disks, or on a local disk. To turn on the encryption mode, in the configuration file you must define a disk with the type `encrypted` and choose a disk on which the data will be saved. An `encrypted` disk ciphers all written files on the fly, and when you read files from an `encrypted` disk it deciphers them automatically. So you can work with an `encrypted` disk like with a normal one.

Example of disk configuration:

```

<disks>
  <disk1>
    <type>local</type>
    <path>/path1/</path>
  </disk1>
  <disk2>
    <type>encrypted</type>
    <disk>disk1</disk>
    <path>path2/</path>
    <key>_16_ascii_chars_</key>
  </disk2>
</disks>

```

For example, when ClickHouse writes data from some table to a file `store/all_1_1_0/data.bin` to `disk1`, then in fact this file will be written to the physical disk along the path `/path1/store/all_1_1_0/data.bin`.

When writing the same file to `disk2`, it will actually be written to the physical disk at the path `/path1/path2/store/all_1_1_0/data.bin` in encrypted mode.

Required parameters:

- `type` — encrypted. Otherwise the encrypted disk is not created.
- `disk` — Type of disk for data storage.
- `key` — The key for encryption and decryption. Type: [UInt64](#). You can use `key_hex` parameter to encrypt in hexadecimal form.  
You can specify multiple keys using the `id` attribute (see example above).

Optional parameters:

- `path` — Path to the location on the disk where the data will be saved. If not specified, the data will be saved in the root directory.
- `current_key_id` — The key used for encryption. All the specified keys can be used for decryption, and you can always switch to another key while maintaining access to previously encrypted data.
- `algorithm` — [Algorithm](#) for encryption. Possible values: `AES_128_CTR`, `AES_192_CTR` or `AES_256_CTR`. Default value: `AES_128_CTR`. The key length depends on the algorithm: `AES_128_CTR` — 16 bytes, `AES_192_CTR` — 24 bytes, `AES_256_CTR` — 32 bytes.

Example of disk configuration:

```

<clickhouse>
  <storage_configuration>
    <disks>
      <disk_s3>
        <type>s3</type>
        <endpoint>...
      </disk_s3>
      <disk_s3_encrypted>
        <type>encrypted</type>
        <disk>disk_s3</disk>
        <algorithm>AES_128_CTR</algorithm>
        <key_hex id="0">00112233445566778899aabccddeeff</key_hex>
        <key_hex id="1">ffeedddccbbaa99887766554433221100</key_hex>
        <current_key_id>1</current_key_id>
      </disk_s3_encrypted>
    </disks>
  </storage_configuration>
</clickhouse>

```

## Storing Data on Web Server

There is a tool `clickhouse-static-files-uploader`, which prepares a data directory for a given table (`SELECT data_paths FROM system.tables WHERE name = 'table_name'`). For each table you need, you get a directory of files. These files can be uploaded to, for example, a web server with static files. After this preparation, you can load this table into any ClickHouse server via DiskWeb.

This is a read-only disk. Its data is only read and never modified. A new table is loaded to this disk via `ATTACH TABLE` query (see example below). Local disk is not actually used, each `SELECT` query will result in a http request to fetch required data. All modification of the table data will result in an exception, i.e. the following types of queries are not allowed: `CREATE TABLE`, `ALTER TABLE`, `RENAME TABLE`, `DETACH TABLE` and `TRUNCATE TABLE`.

Web server storage is supported only for the `MergeTree` and `Log` engine families. To access the data stored on a web disk, use the `storage_policy` setting when executing the query. For example, `ATTACH TABLE table_web UUID '{}' (id Int32) ENGINE = MergeTree() ORDER BY id SETTINGS storage_policy = 'web'`.

A ready test case. You need to add this configuration to config:

```
<clickhouse>
  <storage_configuration>
    <disks>
      <web>
        <type>web</type>
        <endpoint>https://clickhouse-datasets.s3.yandex.net/disk-with-static-files-tests/test-hits/</endpoint>
      </web>
    </disks>
    <policies>
      <web>
        <volumes>
          <main>
            <disk>web</disk>
          </main>
        </volumes>
      </web>
    </policies>
  </storage_configuration>
</clickhouse>
```

And then execute this query:

```
ATTACH TABLE test_hits UUID '1ae36516-d62d-4218-9ae3-6516d62da218'
(
    WatchID UInt64,
    JavaEnable UInt8,
    Title String,
    GoodEvent Int16,
    EventTime DateTime,
    EventDate Date,
    CounterID UInt32,
    ClientIP UInt32,
    ClientIP6 FixedString(16),
    RegionID UInt32,
    UserID UInt64,
    CounterClass Int8,
    OS UInt8,
    UserAgent UInt8,
    URL String,
    Referer String,
    URLDomain String,
    RefererDomain String,
    Refresh UInt8,
    IsRobot UInt8,
    RefererCategories Array(UInt16),
    URLCategories Array(UInt16),
    URLRegions Array(UInt32),
    RefererRegions Array(UInt32),
    ResolutionWidth UInt16,
    ResolutionHeight UInt16,
    ResolutionDepth UInt8,
    FlashMajor UInt8
```

```
FlashMajor UInt8,
FlashMinor UInt8,
FlashMinor2 String,
NetMajor UInt8,
NetMinor UInt8,
UserAgentMajor UInt16,
UserAgentMinor FixedString(2),
CookieEnable UInt8,
JavascriptEnable UInt8,
IsMobile UInt8,
MobilePhone UInt8,
MobilePhoneModel String,
Params String,
IPNetworkID UInt32,
TraficSourceID Int8,
SearchEngineID UInt16,
SearchPhrase String,
AdvEngineID UInt8,
IsArtifical UInt8,
WindowClientWidth UInt16,
WindowClientHeight UInt16,
ClientTimeZone Int16,
ClientEventTime DateTime,
SilverlightVersion1 UInt8,
SilverlightVersion2 UInt8,
SilverlightVersion3 UInt32,
SilverlightVersion4 UInt16,
PageCharset String,
CodeVersion UInt32,
IsLink UInt8,
IsDownload UInt8,
 IsNotBounce UInt8,
FUniqID UInt64,
HID UInt32,
IsOldCounter UInt8,
IsEvent UInt8,
IsParameter UInt8,
DontCountHits UInt8,
WithHash UInt8,
HitColor FixedString(1),
UTCEventTime DateTime,
Age UInt8,
Sex UInt8,
Income UInt8,
Interests UInt16,
Robotness UInt8,
GeneralInterests Array(UInt16),
RemoteIP UInt32,
RemoteIP6 FixedString(16),
WindowName Int32,
OpenerName Int32,
HistoryLength Int16,
BrowserLanguage FixedString(2),
BrowserCountry FixedString(2),
SocialNetwork String,
SocialAction String,
HTTPError UInt16,
SendTiming Int32,
DNSTiming Int32,
ConnectTiming Int32,
ResponseStartTiming Int32,
ResponseEndTiming Int32,
FetchTiming Int32,
RedirectTiming Int32,
DOMInteractiveTiming Int32,
DOMContentLoadedTiming Int32,
DOMCompleteTiming Int32,
LoadEventStartTiming Int32,
LoadEventEndTiming Int32,
NSToDOMContentLoadedTiming Int32,
FirstPaintTiming Int32,
RedirectCount Int8,
SocialSourceNetworkID UInt8,
SocialSourcePage String,
ParamPrice Int64,
ParamOrderID String,
ParamCurrency FixedString(3),
ParamCurrencyID UInt16
```

```

ParanCurrencyID UInt16,
GoalsReached Array(UInt32),
OpenstatServiceName String,
OpenstatCampaignID String,
OpenstatAdID String,
OpenstatSourceID String,
UTMSource String,
UTMMedium String,
UTMCampaign String,
UTMContent String,
UTMTerm String,
FromTag String,
HasGCLID UInt8,
RefererHash UInt64,
URLHash UInt64,
CLID UInt32,
YCLID UInt64,
ShareService String,
ShareURL String,
ShareTitle String,
ParsedParams Nested(
    Key1 String,
    Key2 String,
    Key3 String,
    Key4 String,
    Key5 String,
    ValueDouble Float64),
IslandID FixedString(16),
RequestNum UInt32,
RequestTry UInt8
)
ENGINE = MergeTree()
PARTITION BY toYYYYMM(EventDate)
ORDER BY (CounterID, EventDate, intHash32(UserID))
SAMPLE BY intHash32(UserID)
SETTINGS storage_policy='web';

```

Required parameters:

- `type` — `web`. Otherwise the disk is not created.
- `endpoint` — The endpoint URL in `path` format. Endpoint URL must contain a root path to store data, where they were uploaded.

Optional parameters:

- `min_bytes_for_seek` — The minimal number of bytes to use seek operation instead of sequential read. Default value: `1 Mb`.
- `remote_fs_read_backoff_threshold` — The maximum wait time when trying to read data for remote disk. Default value: `10000` seconds.
- `remote_fs_read_backoff_max_tries` — The maximum number of attempts to read with backoff. Default value: `5`.

If a query fails with an exception `DB:Exception Unreachable URL`, then you can try to adjust the settings: `http_connection_timeout`, `http_receive_timeout`, `keep_alive_timeout`.

To get files for upload run:

```
clickhouse static-files-disk-uploader --metadata-path <path> --output-dir <dir> (--metadata-path can be found in query SELECT data_paths FROM system.tables WHERE name = 'table_name').
```

When loading files by `endpoint`, they must be loaded into `<endpoint>/store/` path, but config must contain only `endpoint`.

If URL is not reachable on disk load when the server is starting up tables, then all errors are caught. If in this case there were errors, tables can be reloaded (become visible) via `DETACH TABLE table_name -> ATTACH TABLE table_name`. If metadata was successfully loaded at server startup, then tables are available straight away.

Use `http_max_single_read_retries` setting to limit the maximum number of retries during a single HTTP read.

## ClickHouse開発

ClickHouseのビルドはLinux、FreeBSD、Mac OS Xでサポートされています。

### Windowsを使用する場合

Windowsを使用する場合は、Ubuntuで仮想マシンを作成する必要があります。 するには、仮想マシンをインストールしてくださいVirtualBox. ダウンロードできますUbuntuのウェブサイト：<https://www.ubuntu.com/#download>. を作成してください仮想マシンからダウンロードした画像を保少なくとも4GB RAMめます。 Ubuntuでコマンドライン端末を実行するには、その単語を含むプログラムを見つけてください “terminal” その名前で（gnome-terminal、konsoleなど。）または単にCtrl+Alt+Tを押します。

### 32ビットシステムを使用する場合

ClickHouseできない仕事を32ビットのシステム。 きの獲得へのアクセスでは、64ビットのシステムを継続できる。

### GitHubでのリポジトリの作成

ClickHouse repositoryで作業を開始するには、GitHubアカウントが必要です。

おそらく既に持っていますが、そうでない場合は、登録してください<https://github.com>.SSHキーがない場合は、それらを生成してGitHubにアップロードする必要があります。 あなたのパッチを送信するために必要です。 他のSSHサーバーで使用するのと同じSSHキーを使用することも可能です。

ClickHouseリポジトリのフォークを作成します。 それを行うには、“fork” 右上のボタン <https://github.com/ClickHouse/ClickHouse>. それはあなたのアカウントにClickHouse/ClickHouseの独自のコピーをフォークします。

開発プロセスは、最初に意図した変更をClickHouseのフォークにコミットし、次に “pull request” これらの変更をメインリポジトリ（ClickHouse/ClickHouse）に受け入れる。

作gitリポジトリをインストールしてください git.

Ubuntuでこれを行うには、コマンドラインターミナルで実行します：

```
sudo apt update  
sudo apt install git
```

簡単なマニュアルを使用Gitで、できるだけ早く送ってください <https://education.github.com/git-cheat-sheet-education.pdf>.

詳細なマニュアルGit見 <https://git-scm.com/book/en/v2>.

### 開発マシンへのリポジトリの複製

次に、ソースファイルを作業マシンにダウンロードする必要があります。 これは “to clone a repository” 作業マシン上にリポジトリのローカルコピーを作成するためです。

コマンドラインインターミナルで実行:

```
git clone git@github.com:your_github_username/ClickHouse.git  
cd ClickHouse
```

注:、代理して下さい `your_github_username` 適切なもので！

このコマンドディレクトリの作成 `ClickHouse` プロジェクトの作業コピーを含む。

ビルドシステムの実行に問題が生じる可能性があるため、作業ディレクトリへのパスに空白が含まれていないことが重要です。

`ClickHouse` リポジトリは以下を使用します `submodules`. That is what the references to additional repositories are called (i.e. external libraries on which the project depends). It means that when cloning the repository you need to specify the `--recursive` 上記の例のようにフラグ。場合のリポジトリにてクローニングな`submodules`、ダウンロードを実行する必要があります:

```
git submodule init  
git submodule update
```

このコマンドでステータスを確認できます: `git submodule status`.

次のエラーメッセージが表示される場合:

```
Permission denied (publickey).  
fatal: Could not read from remote repository.  
  
Please make sure you have the correct access rights  
and the repository exists.
```

一般に、GitHubに接続するためのSSHキーがないことを意味します。これらのキーは、通常、`~/.ssh`. SSHキーを受け入れるには、GitHub UIの設定セクションにアップロードする必要があります。

またクローンをリポジトリによ`https`プロトコル:

```
git clone https://github.com/ClickHouse/ClickHouse.git
```

ただし、変更をサーバーに送信することはできません。にもそのままお使いいただけで一時的にメモリの使用範囲のサイズはSSHキーの後に交換し、リモートアドレスのリポジトリ `git remote` コマンド

元の`ClickHouse`レポのアドレスをローカルリポジトリに追加して、そこから更新を取得することもできます:

```
git remote add upstream git@github.com:ClickHouse/ClickHouse.git
```

このコマンドを正常に実行すると、次のようにしてメインの`ClickHouse`リポジトリから更新を取得できます `git pull upstream master`.

## サブモジュールの操作

Gitでサブモジュールを操作するのは苦痛です。次のコマンドは管理に役立ちます:

```
# ! each command accepts
# Update remote URLs for submodules. Barely rare case
git submodule sync
# Add new submodules
git submodule init
# Update existing submodules to the current state
git submodule update
# Two last commands could be merged together
git submodule update --init
```

次のコマンドは、すべてのサブモジュールを初期状態にリセットするのに役立ちます（！ツヅツキ。 - 内部の変更は削除されます）：

```
# Synchronizes submodules' remote URL with .gitmodules
git submodule sync
# Update the registered submodules with initialize not yet initialized
git submodule update --init
# Reset all changes done after HEAD
git submodule foreach git reset --hard
# Clean files from .gitignore
git submodule foreach git clean -xfd
# Repeat last 4 commands for all submodule
git submodule foreach git submodule sync
git submodule foreach git submodule update --init
git submodule foreach git submodule foreach git reset --hard
git submodule foreach git submodule foreach git clean -xfd
```

## ビルドシステム

ClickHouseは、構築のためのCMakeと忍者を使用しています。

CMake-忍者ファイル（ビルドタスク）を生成することができるメタビルドシステム。

忍者-これらのcmake生成されたタスクを実行するために使用される速度に焦点を当てた小さなビルドシステム。

Ubuntu、DebianまたはMint runにインストールするには `sudo apt install cmake ninja-build`.

セントスでは、レッドハットラン `sudo yum install cmake ninja-build`.

ArchまたはGentooを使用する場合は、おそらくCMakeのインストール方法を自分で知っています。

そのためのCMakeおよび忍者Mac OS X初めて自作としてインストールインストールさんによbrew:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
brew install cmake ninja
```

次に、CMakeのバージョンを確認します: `cmake --version`. 3.3未満の場合は、webサイトから新しいバージョンをインストールしてください。<https://cmake.org/download/>.

## 任意の外部ライブラリ

ClickHouseはビルドに複数の外部ライブラリを使用します。それらのすべては、サブモジュールにあるソースからClickHouseと一緒に構築されているので、別々にインストールする必要はありません。リストは次の場所で確認できます `contrib`.

## C++ Compiler

Compilers Clang starting from version 11 is supported for building ClickHouse.

Clang should be used instead of gcc. Though, our continuous integration (CI) platform runs checks for about a dozen of build combinations.

On Ubuntu/Debian you can use the automatic installation script (check [official webpage](#))

```
sudo bash -c "$(wget -O - https://apt.llvm.org/llvm.sh)"
```

Mac OS X build is also supported. Just run `brew install llvm`

## 建築プロセス

ClickHouseを構築する準備ができたので、別のディレクトリを作成することをお勧めします `build` 内部 ClickHouse それはすべてのビルド人工物が含まれています:

```
mkdir build  
cd build
```

いくつかの異なるディレクトリ (`build_release`、`build_debug`など) を持つことができます。) ビルドの異なるタイプのために。

中の間 `build cmake` を実行してビルドを構成します。最初の実行の前に、コンパイラ (この例ではバージョン9gccコンパイラ) を指定する環境変数を定義する必要があります。

```
export CC=clang CXX=clang++  
cmake ..
```

その `CC` 変数は、Cのコンパイラを指定します (Cコンパイラの略)。 `CXX variable` は、どのC++コンパイラをビルドに使用するかを指示します。

より高速なビルドのために、あなたは `debug` ビルドタイプ-最適化のないビルド。その供給は、以下のパラメータ `-D CMAKE_BUILD_TYPE=Debug`:

```
cmake -D CMAKE_BUILD_TYPE=Debug ..
```

ビルドのタイプを変更するには、次のコマンドを実行します。`build` ディレクトリ。

忍者を実行して構築する:

```
ninja clickhouse-server clickhouse-client
```

この例では、必要なバイナリのみを構築します。

必要な場合は、構築すべてのバイナリ(光熱費および試験)を動かして行く必要がある忍者のないパラメータ:

```
ninja
```

フルの構築が必要約30GBのディスクスペースまたは15GBの主binaries.

ビルドマシン上で大量のRAMが利用可能な場合は、以下と並行して実行されるビルドタスクの数を制限する必要があります `-j param`:

```
ninja -j 1 clickhouse-server clickhouse-client
```

4GBのRAMを搭載しているマシンでは、1を8GBのRAMに指定することをお勧めします `-j 2` 推奨されます。

メッセージが届いたら: `ninja: error: loading 'build.ninja': No such file or directory` これは、ビルド構成の生成が失敗し、上記のメッセージを検査する必要があることを意味します。

ビルドプロセスが正常に開始されると、ビルドの進行状況、つまり処理されたタスクの数とタスクの総数が表示されます。

ながらメッセージについてprotobufファイルlibhdfs2図書館のような libprotobuf WARNING 現れるかも 彼らは何も影響を与えず、無視されても安全です。

ビルドが成功すると、実行可能ファイルを取得します ClickHouse/<build\_dir>/programs/clickhouse:

```
ls -l programs/clickhouse
```

## ClickHouseのビルドされた実行可能ファイルの実行

現在のユーザーの下でサーバーを実行するには、次の場所に移動します ClickHouse/programs/server/（外にあります build)と実行:

```
../../build/programs/clickhouse server
```

この場合、ClickHouseは現在のディレクトリにある設定ファイルを使用します。実行できます clickhouse server からのディレクトリのパスを指定し、設定ファイルとしてコマンドラインパラメータ --config-file.

別のターミナルでclickhouse-clientを使用してClickHouseに接続するには、次の場所に移動します ClickHouse/build/programs/ と実行 ./clickhouse client.

あなたが得る場合 Connection refused メッセージMac OS XまたはFreeBSDでは、ホストアドレス127.0.0.1を指定してみます:

```
clickhouse client --host 127.0.0.1
```

に置き換えることができ生産版ClickHouseバイナリインストールされるシステムのカスタム構築ClickHouseバイナリー. いるイ ClickHouse利用するマシンの指示に従って公式サイトから 次に、以下を実行します:

```
sudo service clickhouse-server stop  
sudo cp ClickHouse/build/programs/clickhouse /usr/bin/  
sudo service clickhouse-server start
```

なお clickhouse-client, clickhouse-server そして、他のものは、一般的に共有される clickhouse バイナリ を運営することも可能ですカスタム構築ClickHouseバイナリのコンフィグファイルからのClickHouseパッケージをインストールシステム:

```
sudo service clickhouse-server stop  
sudo -u clickhouse ClickHouse/build/programs/clickhouse server --config-file /etc/clickhouse-server/config.xml
```

## IDE(統合開発環境)

使用するIDEがわからない場合は、CLionを使用することをお勧めします。CLionは商用ソフトウェアですが、30日間の無料試用期間を提供しています。また、学生のための無料です。CLionはLinuxとMac OS Xの両方で使用できます。

KDevelopとQTCreatorは、ClickHouseを開発するためのIDEの他の優れた選択肢です。KDevelopは非常に便利なIDEとして提供されますが、不安定です。まKDevelopクラッシュが開設するプロジェクト、クリック“Stop All”プロジェクトのファイルのリストを開くとすぐにボタンを押します。そうした後、KDevelopはうまくいくはずです。

単純なコードエディターとして、Sublime TextまたはVisual Studio Code、またはKate(すべてLinuxで利用可能)を使用できます。

念のため、CLionが作成することに言及する価値があります build 独自のパス、それはまた、独自の選択に debug ビルドタイプの場合、構成のために、CLionで定義されているCMakeのバージョンを使用し、インストールされているものではありません。make ビルドタスクを実行するには ninja。これは通常の動作ですが、混乱を避けるためにそれを念頭に置いてください。

## コードの作成

の説明ClickHouse建築で、できるだけ早く送ってください <https://clickhouse.com/docs/en/開発/アーキテクチャ/>

コードのスタイルガイド：<https://clickhouse.com/docs/en/開発/スタイル/>

筆記試験：<https://clickhouse.com/docs/en/development/tests/>

タスクのリスト：[https://github.com/ClickHouse/ClickHouse/issues?  
q=is%3Aopen+is%3Aissue+label%3A%22easy+task%22](https://github.com/ClickHouse/ClickHouse/issues?q=is%3Aopen+is%3Aissue+label%3A%22easy+task%22)

## テストデータ

開発ClickHouseが必要となり載荷実ックスです。パフォーマンステストでは特に重要です。して特定の匿名化データからのYandex. メトリカ さらに3GBの空きディスク領域が必要です。このデータは、ほとんどの開発タスクを実行するためには必要ありません。

```
sudo apt install wget xz-utils

wget https://datasets.clickhouse.com/hits/tsv/hits_v1.tsv.xz
wget https://datasets.clickhouse.com/visits/tsv/visits_v1.tsv.xz

xz -v -d hits_v1.tsv.xz
xz -v -d visits_v1.tsv.xz

clickhouse-client

CREATE DATABASE IF NOT EXISTS test

CREATE TABLE test.hits ( WatchID UInt64, JavaEnable UInt8, Title String, GoodEvent Int16, EventTime DateTime,
EventDate Date, CounterID UInt32, ClientIP UInt32, ClientIP6 FixedString(16), RegionID UInt32, UserID UInt64,
CounterClass Int8, OS UInt8, UserAgent UInt8, URL String, Referer String, URLDomain String, RefererDomain String,
Refresh UInt8, IsRobot UInt8, RefererCategories Array(UInt16), URLCategories Array(UInt16), URLRegions
Array(UInt32), RefererRegions Array(UInt32), ResolutionWidth UInt16, ResolutionHeight UInt16, ResolutionDepth
UInt8, FlashMajor UInt8, FlashMinor UInt8, FlashMinor2 String, NetMajor UInt8, NetMinor UInt8, UserAgentMajor
UInt16, UserAgentMinor FixedString(2), CookieEnable UInt8, JavascriptEnable UInt8, IsMobile UInt8, MobilePhone
UInt8, MobilePhoneModel String, Params String, IPNetworkID UInt32, TraficSourceID Int8, SearchEngineID UInt16,
SearchPhrase String, AdvEngineID UInt8, IsArtifical UInt8, WindowClientWidth UInt16, WindowClientHeight UInt16,
ClientTimeZone Int16, ClientEventTime DateTime, SilverlightVersion1 UInt8, SilverlightVersion2 UInt8,
SilverlightVersion3 UInt32, SilverlightVersion4 UInt16, PageCharset String, CodeVersion UInt32, IsLink UInt8,
IsDownload UInt8, IsNotBounce UInt8, FUniqID UInt64, HID UInt32, IsOldCounter UInt8, IsEvent UInt8, IsParameter
UInt8, DontCountHits UInt8, WithHash UInt8, HitColor FixedString(1), UTCEventTime DateTime, Age UInt8, Sex
UInt8, Income UInt8, Interests UInt16, Robotness UInt8, GeneralInterests Array(UInt16), RemoteIP UInt32,
RemoteIP6 FixedString(16), WindowName Int32, OpenerName Int32, HistoryLength Int16, BrowserLanguage
FixedString(2), BrowserCountry FixedString(2), SocialNetwork String, SocialAction String, HTTPError UInt16,
SendTiming Int32, DNSTiming Int32, ConnectTiming Int32, ResponseStartTiming Int32, ResponseEndTiming Int32,
FetchTiming Int32, RedirectTiming Int32, DOMInteractiveTiming Int32, DOMContentLoadedTiming Int32,
DOMCompleteTiming Int32, LoadEventStartTiming Int32, LoadEventEndTiming Int32,
NSToDOMContentLoadedTiming Int32, FirstPaintTiming Int32, RedirectCount Int8, SocialSourceNetworkID UInt8,
SocialSourcePage String, ParamPrice Int64, ParamOrderID String, ParamCurrency FixedString(3), ParamCurrencyID
UInt16, GoalsReached Array(UInt32), OpenstatServiceName String, OpenstatCampaignID String, OpenstatAdID
String, OpenstatSourceID String, UTMSource String, UTMMedium String, UTMCampaign String, UTMContent String,
UTMTerm String, FromTag String, HasGCLID UInt8, RefererHash UInt64, URLHash UInt64, CLID UInt32, YCLID
UInt64, ShareService String, ShareURL String, ShareTitle String, `ParsedParams.Key1` Array(String),
`ParsedParams.Key2` Array(String), `ParsedParams.Key3` Array(String), `ParsedParams.Key4` Array(String),
`ParsedParams.Key5` Array(String), `ParsedParams.ValueDouble` Array(Float64), IslandID FixedString(16),
RequestNum UInt32, RequestTry UInt8) ENGINE = MergeTree PARTITION BY toYYYYMM(EventDate) SAMPLE BY
intHash32(UserID) ORDER BY (CounterID, EventDate, intHash32(UserID), EventTime);

CREATE TABLE test.visits ( CounterID UInt32, StartDate Date, Sign Int8, IsNew UInt8, VisitID UInt64, UserID UInt64,
StartTime DateTime, Duration UInt32, UTCStartTime DateTime, PageViews Int32, Hits Int32, IsBounce UInt8,
Referer String, StartURL String, RefererDomain String, StartURLDomain String, EndURL String, LinkURL String,
IsDownload UInt8, TraficSourceID Int8, SearchEngineID UInt16, SearchPhrase String, AdvEngineID UInt8, PlaceID
UInt32, RefererCategories Array(UInt16), URLCategories Array(UInt16), URLRegions Array(UInt32), RefererRegions
```

```

Array(UInt32), IsYandex UInt8, GoalReachesDepth UInt32, GoalReachesURL UInt32, GoalReachesAny Int32,
SocialSourceNetworkID UInt8, SocialSourcePage String, MobilePhoneModel String, ClientEventTime DateTime,
RegionID UInt32, ClientIP UInt32, ClientIP6 FixedString(16), RemoteIP UInt32, RemoteIP6 FixedString(16),
IPNetworkID UInt32, SilverlightVersion3 UInt32, CodeVersion UInt32, ResolutionWidth UInt16, ResolutionHeight
UInt16, UserAgentMajor UInt16, UserAgentMinor UInt16, WindowClientWidth UInt16, WindowClientHeight UInt16,
SilverlightVersion2 UInt8, SilverlightVersion4 UInt16, FlashVersion3 UInt16, FlashVersion4 UInt16, ClientTimeZone
Int16, OS UInt8, UserAgent UInt8, ResolutionDepth UInt8, FlashMajor UInt8, FlashMinor UInt8, NetMajor UInt8,
NetMinor UInt8, MobilePhone UInt8, SilverlightVersion1 UInt8, Age UInt8, Sex UInt8, Income UInt8, JavaEnable
UInt8, CookieEnable UInt8, JavascriptEnable UInt8, IsMobile UInt8, BrowserLanguage UInt16, BrowserCountry
UInt16, Interests UInt16, Robotness UInt8, GeneralInterests Array(UInt16), Params Array(String), `Goals.ID`  

Array(UInt32), `Goals.Serial` Array(UInt32), `Goals.EventTime` Array(DateTime), `Goals.Price` Array(Int64),
`Goals.OrderID` Array(String), `Goals.CurrencyID` Array(UInt32), WatchIDs Array(UInt64), ParamSumPrice Int64,  

ParamCurrency FixedString(3), ParamCurrencyID UInt16, ClickLogID UInt64, ClickEventID Int32, ClickGoodEvent
Int32, ClickEventTime DateTime, ClickPriorityID Int32, ClickPhraseID Int32, ClickPageID Int32, ClickPlaceID Int32,  

ClickTypeID Int32, ClickResourceID Int32, ClickCost UInt32, ClickClientIP UInt32, ClickDomainID UInt32, ClickURL
String, ClickAttempt UInt8, ClickOrderID UInt32, ClickBannerID UInt32, ClickMarketCategoryID UInt32,  

ClickMarketPP UInt32, ClickMarketCategoryName String, ClickMarketPPName String, ClickAWAPSCampaignName
String, ClickPageName String, ClickTargetType UInt16, ClickTargetPhraseID UInt64, ClickContextType UInt8,  

ClickSelectType Int8, ClickOptions String, ClickGroupBannerID Int32, OpenstatServiceName String,  

OpenstatCampaignID String, OpenstatAdID String, OpenstatSourceID String, UTMSource String, UTMMedium String,  

UTMCampaign String, UTMCContent String, UTMTerm String, FromTag String, HasGCLID UInt8, FirstVisit DateTime,  

PredLastVisit Date, LastVisit Date, TotalVisits UInt32, `TraficSource.ID` Array(Int8), `TraficSource.SearchEngineID`  

Array(UInt16), `TraficSource.AdvEngineID` Array(UInt8), `TraficSource.PlaceID` Array(UInt16),  

`TraficSource.SocialSourceNetworkID` Array(UInt8), `TraficSource.Domain` Array(String),  

`TraficSource.SearchPhrase` Array(String), `TraficSource.SocialSourcePage` Array(String), Attendance  

FixedString(16), CLID UInt32, YCLID UInt64, NormalizedRefererHash UInt64, SearchPhraseHash UInt64,  

RefererDomainHash UInt64, NormalizedStartURLHash UInt64, StartURLDomainHash UInt64, NormalizedEndURLHash  

UInt64, TopLevelDomain UInt64, URLScheme UInt64, OpenstatServiceNameHash UInt64, OpenstatCampaignIDHash  

UInt64, OpenstatAdIDHash UInt64, OpenstatSourceIDHash UInt64, UTMSourceHash UInt64, UTMMediumHash  

UInt64, UTMCampaignHash UInt64, UTMCContentHash UInt64, UTMTermHash UInt64, FromHash UInt64,  

WebVisorEnabled UInt8, WebVisorActivity UInt32, `ParsedParams.Key1` Array(String), `ParsedParams.Key2`  

Array(String), `ParsedParams.Key3` Array(String), `ParsedParams.Key4` Array(String), `ParsedParams.Key5`  

Array(String), `ParsedParams.ValueDouble` Array(Float64), `Market.Type` Array(UInt8), `Market.GoalID`  

Array(UInt32), `Market.OrderID` Array(String), `Market.OrderPrice` Array(UInt64), `Market.PP` Array(UInt32),  

`Market.DirectPlaceID` Array(UInt32), `Market.DirectOrderID` Array(UInt32), `Market.DirectBannerID` Array(UInt32),  

`Market.GoodID` Array(String), `Market.GoodName` Array(String), `Market.GoodQuantity` Array(Int32),  

`Market.GoodPrice` Array(Int64), IslandID FixedString(16)) ENGINE = CollapsingMergeTree(Sign) PARTITION BY  

toYYYYMM(StartDate) SAMPLE BY intHash32(UserID) ORDER BY (CounterID, StartDate, intHash32(UserID), VisitID);

```

```

clickhouse-client --max_insert_block_size 100000 --query "INSERT INTO test.hits FORMAT TSV" < hits_v1.tsv
clickhouse-client --max_insert_block_size 100000 --query "INSERT INTO test.visits FORMAT TSV" < visits_v1.tsv

```

## プル要求の作成

GitHubのUIでforkリポジトリに移動します。 ブランチで開発している場合は、そのブランチを選択する必要があります。 があるでしょう “Pull request” 画面上にあるボタン。 本質的に、これは “create a request for accepting my changes into the main repository”。

プル要求は、作業がまだ完了していない場合でも作成できます。 この場合、単語を入れてください “WIP”（進行中の作業）タイトルの先頭に、それは後で変更することができます。 これは、変更の協調的なレビューと議論、および利用可能なすべてのテストの実行に役立ちます。 変更の簡単な説明を提供することが重要です。

Yandexの従業員がタグあなたのPRにラベルを付けるとすぐにテストが開始されます “can be tested”. The results of some first checks (e.g. code style) will come in within several minutes. Build check results will arrive within half an hour. And the main set of tests will report itself within an hour.

システムは、プル要求用にClickHouseバイナリビルドを個別に準備します。 これらのビルドを取得するには “Details” 次のリンク “ClickHouse build check” 小切手のリストのエントリ。 そこには、ビルドへの直接リンクがあります。 ClickHouseのdebパッケージは、本番サーバーにも展開できます（恐れがない場合）。

ほとんどの場合、ビルドの一部は最初に失敗します。 これは、gccとclangの両方でビルドをチェックするという事実によるものです。 -Werror flag)clangを有効にします。 その同じページで、すべてのビルドログを見つけることができる所以、ClickHouseをすべての可能な方法でビルドする必要はありません。

## Continuous Integration Checks

When you submit a pull request, some automated checks are ran for your code by the ClickHouse [continuous integration \(CI\) system](#).

This happens after a repository maintainer (someone from ClickHouse team) has screened your code and added the can be tested label to your pull request.

The results of the checks are listed on the GitHub pull request page as described in the [GitHub checks documentation](#).

If a check is failing, you might be required to fix it. This page gives an overview of checks you may encounter, and what you can do to fix them.

If it looks like the check failure is not related to your changes, it may be some transient failure or an infrastructure problem. Push an empty commit to the pull request to restart the CI checks:

```
git reset  
git commit --allow-empty  
git push
```

If you are not sure what to do, ask a maintainer for help.

## Merge With Master

Verifies that the PR can be merged to master. If not, it will fail with the message 'Cannot fetch mergecommit'. To fix this check, resolve the conflict as described in the [GitHub documentation](#),

or merge the `master` branch to your pull request branch using git.

## Docs check

Tries to build the ClickHouse documentation website. It can fail if you changed something in the documentation. Most probable reason is that some cross-link in the documentation is wrong. Go to the check report and look for `ERROR` and `WARNING` messages.

## Report Details

- [Status page example](#)
- `docs_output.txt` contains the building log. [Successful result example](#)

## Description Check

Check that the description of your pull request conforms to the template [PULL\\_REQUEST\\_TEMPLATE.md](#).

You have to specify a changelog category for your change (e.g., Bug Fix), and write a user-readable message describing the change for [CHANGELOG.md](#)

## Push To Dockerhub

Builds docker images used for build and tests, then pushes them to DockerHub.

## Marker Check

This check means that the CI system started to process the pull request. When it has 'pending' status, it means that not all checks have been started yet. After all checks have been started, it changes status to 'success'.

## Style Check

Performs some simple regex-based checks of code style, using the `utils/check-style/check-style` binary (note that it can be run locally).

If it fails, fix the style errors following the [code style guide](#).

### Report Details

- [Status page example](#)
- `output.txt` contains the check resulting errors (invalid tabulation etc), blank page means no errors.  
[Successful result example](#).

## PVS Check

Check the code with [PVS-studio](#), a static analysis tool. Look at the report to see the exact errors. Fix them if you can, if not -- ask a ClickHouse maintainer for help.

### Report Details

- [Status page example](#)
- `test_run.txt.out.log` contains the building and analyzing log file. It includes only parsing or not-found errors.
- `HTML report` contains the analysis results. For its description visit PVS's [official site](#).

## Fast Test

Normally this is the first check that is ran for a PR. It builds ClickHouse and runs most of [stateless functional tests](#), omitting some. If it fails, further checks are not started until it is fixed. Look at the report to see which tests fail, then reproduce the failure locally as described [here](#).

### Report Details

#### [Status page example](#)

#### Status Page Files

- `runlog.out.log` is the general log that includes all other logs.
- `test_log.txt`
- `submodule_log.txt` contains the messages about cloning and checkout needed submodules.
- `stderr.log`
- `stdout.log`
- `clickhouse-server.log`
- `clone_log.txt`
- `install_log.txt`
- `clickhouse-server.err.log`
- `build_log.txt`
- `cmake_log.txt` contains messages about the C/C++ and Linux flags check.

## Status Page Columns

- *Test name* contains the name of the test (without the path e.g. all types of tests will be stripped to the name).
- *Test status* -- one of *Skipped*, *Success*, or *Fail*.
- *Test time, sec.* -- empty on this test.

## Build Check

Builds ClickHouse in various configurations for use in further steps. You have to fix the builds that fail. Build logs often has enough information to fix the error, but you might have to reproduce the failure locally. The `cmake` options can be found in the build log, grepping for `cmake`. Use these options and follow the [general build process](#).

## Report Details

[Status page example](#).

- **Compiler:** `gcc-9` or `clang-10` (or `clang-10-xx` for other architectures e.g. `clang-10-freebsd`).
- **Build type:** `Debug` or `RelWithDebInfo` (`cmake`).
- **Sanitizer:** `none` (without sanitizers), `address` (ASan), `memory` (MSan), `undefined` (UBSan), or `thread` (TSan).
- **Bundled:** `bundled` build uses libraries from `contrib` folder, and `unbundled` build uses system libraries.
- **Splitted** splitted is a [split build](#)
- **Status:** `success` or `fail`
- **Build log:** link to the building and files copying log, useful when build failed.
- **Build time.**
- **Artifacts:** build result files (with `XXX` being the server version e.g. `20.8.1.4344`).
  - `clickhouse-client_XXX_all.deb`
  - `clickhouse-common-static-dbg_XXX[+asan, +msan, +ubsan, +tsan]_amd64.deb`
  - `clickhouse-common-staticXXX_amd64.deb`
  - `clickhouse-server_XXX_all.deb`
  - `clickhouse-test_XXX_all.deb`
  - `clickhouse_XXX_amd64.buildinfo`
  - `clickhouse_XXX_amd64.changes`
  - `clickhouse`: Main built binary.
  - `clickhouse-odbc-bridge`
  - `unit_tests_dbms`: GoogleTest binary with ClickHouse unit tests.
  - `shared_build.tgz`: build with shared libraries.
  - `performance.tgz`: Special package for performance tests.

## Special Build Check

Performs static analysis and code style checks using clang-tidy. The report is similar to the [build check](#). Fix the errors found in the build log.

## Functional Stateless Tests

Runs [stateless functional tests](#) for ClickHouse binaries built in various configurations -- release, debug, with sanitizers, etc. Look at the report to see which tests fail, then reproduce the failure locally as described [here](#). Note that you have to use the correct build configuration to reproduce -- a test might fail under AddressSanitizer but pass in Debug. Download the binary from [CI build checks page](#), or build it locally.

## Functional Stateful Tests

Runs [stateful functional tests](#). Treat them in the same way as the functional stateless tests. The difference is that they require `hits` and `visits` tables from the [Yandex.Metrica dataset](#) to run.

## Integration Tests

Runs [integration tests](#).

## Testflows Check

Runs some tests using Testflows test system. See [here](#) how to run them locally.

## Stress Test

Runs stateless functional tests concurrently from several clients to detect concurrency-related errors. If it fails:

- \* Fix all other test failures first;
- \* Look at the report to find the server logs and check them for possible causes of error.

## Split Build Smoke Test

Checks that the server build in [split build](#) configuration can start and run simple queries. If it fails:

- \* Fix other test errors first;
- \* Build the server in [split build](#development-build-md) configuration locally and check whether it can start and run `select 1`.

## Compatibility Check

Checks that clickhouse binary runs on distributions with old libc versions. If it fails, ask a maintainer for help.

## AST Fuzzer

Runs randomly generated queries to catch program errors. If it fails, ask a maintainer for help.

## Performance Tests

Measure changes in query performance. This is the longest check that takes just below 6 hours to run. The performance test report is described in detail [here](#).

# QA

What is a Task (private network) item on status pages?

It's a link to the Yandex's internal job system. Yandex employees can see the check's start time and its more verbose status.

Where the tests are run

Somewhere on Yandex internal infrastructure.

## クリックハウス建築の概要

ClickHouseは眞の列指向のDBMSです。データは、列によって、および配列（列のベクトルまたはチャンク）の実行中に格納されます。可能であれば、個々の値ではなく配列に対して演算が送出されます。それは呼ばれます“vectorized query execution,” そしてそれは実際のデータ処理の費用を下げるのを助けます。

この考え方は、新しいものではない。それはにさかのぼります APL プログラミング言語とその子孫: A+, J, K, and Q. 配列プログラミングは科学データ処理で使用されます。どちらもこの考え方は関係データベースで新しいものではありません。Vectorwise システム

クエリ処理の高速化には、ベクトル化されたクエリ実行とランタイムコード生成という二つの方法があります。後者は、すべての間接および動的ディスパッチを削除します。それのアプローチは厳重によります。ランタイムコード生成は、多くの操作を融合し、CPU実行単位とパイプラインを完全に利用する場合に優れています。Vectorizedクエリを実行できる実用的では一時的ベクトルを明記のことは、キャッシュを読みます。一時データがL2キャッシュに収まらない場合、これが問題になります。しかし、ベクトル化されたクエリの実行は、CPUのSIMD機能をより簡単に利用します。A 研究論文 書面による友人達しないことであり、両アプローチ。ClickHouse用vectorizedクエリを実行して初期支援のためにランタイムコード。

## 列

`IColumn interface`は、メモリ内の列（実際には列のチャンク）を表すために使用されます。この元の列を変更するのではなく、新しい変更された列を作成します。例えば、`IColumn :: filter` 法を受け入れフィルタのバイトマスクです。それはのために使用されます `WHERE` と `HAVING` 関係演算子。追加の例：`IColumn :: permute` サポートする方法 `ORDER BY` は、`IColumn :: cut` サポートする方法 `LIMIT`。

各種 `IColumn` 実装 (`ColumnUInt8`, `ColumnString` というように) は、列のメモリレイアウトを担当しています。メモリレイアウトは、通常、連続した配列です。整数型の列の場合は、次のように連続した配列にすぎません `std :: vector`. のために `String` と `Array` すべての配列要素に対して一つ、連続して配置され、各配列の先頭へのオフセットに対して二つのベクトルです。また `ColumnConst` これはメモリに一つの値を格納しますが、列のように見えます。

## フィールド

それにもかかわらず、個々の価値を扱うことも可能です。個々の値を表すために、`Field` が使用される。`Field` ただの識別された組合である `UInt64`, `Int64`, `Float64`, `String` と `Array`. `IColumn` は、`operator[]` n番目の値をaとして取得するメソッド `Field` そして、`insert a`を追加するメソッド `Field` 列の最後まで。これらの方法は、一時的な処理を必要とするため、あまり効率的ではありません `Field` 個々の値を表すオブジェクト。次のようなより効率的な方法があります `insertFrom`, `insertRangeFrom`、というように。

`Field` テーブルの特定のデータ型に関する十分な情報がありません。例えば, `UInt8`, `UInt16`, `UInt32`, and `UInt64` すべてとして表されます `UInt64` で `Field`.

## 漏れやすい抽象化

`IColumn` は方法のための共通の関係変容のデータもあるんですが、そのすべて満たす。例えば、`ColumnUInt64` 二つの列の合計を計算する方法がありません。`ColumnString` 部分文字列検索を実行するメソッドがありません。これらの無数のルーチンは `IColumn`。

列のさまざまな関数は、次のようにして、一般的で非効率的な方法で実装できます `IColumn` 抽出する方法 `Field` 値、または特定のデータの内部メモリレイアウトの知識を使用して特殊な方法で `IColumn` 実装。で実施する铸造機能を特定 `IColumn` 内部表現を直接入力して処理します。例えば、`ColumnUInt64` は、`getData` 内部配列への参照を返すメソッドは、別のルーチンが直接その配列を読み取るか、または塗りつぶします。我々は持っている “leaky abstractions” さまざまなルーチンの効率的な専門化を可能にする。

## データ型

`IDataType` 列または個々の値のチャネルをバイナリ形式またはテキスト形式で読み書きするためのものです。`IDataType` テーブル内のデータ型に直接対応します。例えば、次のものがあります `DataTypeUInt32`, `DataTypeDateTime`, `DataTypeString` など。

`IDataType` と `IColumn` 互いに緩やかに関連しているだけです。異なるデータ型は、同じメモリで表すことができます `IColumn` 実装。例えば、`DataTypeUInt32` と `DataTypeDateTime` で表される。`ColumnUInt32` または `ColumnConstUInt32` また、同じデータ型で表現することができ `IColumn` 実装。例えば、`DataTypeUInt8` で表すことができる `ColumnUInt8` または `ColumnConstUInt8`。

`IDataType` 貨物のメタデータを指すものとします。例えば、`DataTypeUInt8` 何も保存しません(`vptr`を除きます)。`DataTypeFixedString` ちょうど店 `N` (固定サイズの文字列のサイズ)。

`IDataType` はヘルパーの方法のための様々なデータフォーマット たとえば、クオート可能な値をシリアル化したり、JSONの値をシリアル化したり、XML形式の一部として値をシリアル化したりするメソッドがあります。データ形式への直接の対応はありません。たとえば、異なるデータ形式 `Pretty` と `TabSeparated` 同じを使用できます `serializeTextEscaped` からのヘルパーメソッド `IDataType` インタフェース

## ブロック

A Block メモリ内のテーブルのサブセット（チャネル）を表すコンテナです。それは単なるトリプルのセットです: (`IColumn`, `IDataType`, `column name`)。クエリの実行中、データは次の方法で処理されます `Blocks`。私達に `a` があれば `Block`、我々はデータを持っている(で `IColumn` そのタイプに関する情報があります `IDataType`) それはその列をどのように扱うかを教えてくれます。これは、テーブルの元の列名か、一時的な計算結果を取得するために割り当てられた人工的な名前のいずれかです。

ブロック内の列に対して関数を計算するとき、その結果を含む別の列をブロックに追加します。後で、不要な列はブロックから削除できますが、変更はできません。共通の部分式を排除するのに便利です。

ブロックの作成のための各処理チャネルのデータです。同じタイプの計算では、列名と型は異なるブロックで同じままであり、列データのみが変更されることに注意してください。ブロックサイズが小さいと、`shared_ptrs` と列名をコピーするための一時的な文字列のオーバーヘッドが高くなるため、ブロックヘッダーからブロックデータを分割

## ブロックの流れ

ブロックストリームのための加工データです。を使用していま流のブロックからデータを読み込むためのどこかに、データ変換、または書き込みデータをどこかということです。`IBlockInputStream` は、`read` 利用可能な状態で次のブロックを取得するメソッド。`IBlockOutputStream` は、`write` どこかにブロックをプッシュする方法。

ストリームは:

1. テーブルへの読み書き。のテーブルだけを返しますストリームを読み取りまたは書き込みブロックとなります。
2. データ形式の実装。たとえば、端末にデータを出力する場合は `Pretty` 書式設定すると、ブロックをプッシュするブロック出力ストリームを作成し、書式設定します。

3. データ変換の実行。あなたが持っているとしよう `IBlockInputStream` いを作ろうというストリームです。作成する `FilterBlockInputStream` ストリームで初期化します その後、ブロックを引っ張るときから `FilterBlockInputStream` で引きプロックからストリーム、フィルタでは、フィルタを返しますプロックします。クエリの実行パイプラインで表現しました。

より洗練された変換があります。例えば、`AggregatingBlockInputStream` で読み込みのすべてのデータからのフレームワークを利用して、集合体で、それを返しますストリームの集約トヘすでに使用されています。別の例:

`UnionBlockInputStream` コンストラクタ内の多くの入力ソースと多数のスレッドを受け入れます。複数のスレッドを起動し、複数のソースから並列に読み込みます。

ブロックストリームは “pull” 制御フローへのアプローチ：最初のストリームからブロックをプルすると、ネストされたストリームから必要なブロックがプルされ、実行パイプライン全体 どちらも “pull” また “push” 制御フローは暗黙的であり、複数のクエリの同時実行（多くのパイプラインをマージする）などのさまざまな機能の実装を制限するため、最良の解決策でこの制限は、コルーチンまたはお互いを待つ余分なスレッドを実行するだけで克服できます。つまり、ある計算単位から別の計算単位の外部にデータを渡すロジックを見つけると、制御フローを明示的にすると、より多くの可能性があります。これを読む [記事](#) より多くの思考のため。

クエリ実行パイプラインでは、各ステップで一時データが作成されます。一時データがCPUキャッシュに収まるように、ブロックサイズを十分に小さくしておきます。その前提では、一時データの書き込みと読み込みは、他の計算と比較してほとんど自由です。これは、パイプライン内の多くの操作と一緒に融合させることです。パイプラインができるだけ短くし、一時データの多くを削除することができますが、これは利点ですが、欠点もあります。たとえば、分割パイプラインを使用すると、中間データのキャッシュ、同時に実行される類似クエリからの中間データの盗み、類似クエリのパイプライン

## 形式

データフォーマットにて実施しブロックわれている。そこには “presentational” 次のように、クライアントへのデータ出力にのみ適した形式になります `Pretty` のみを提供する形式 `IBlockOutputStream`. そして入出力フォーマットが、のようあります `TabSeparated` または `JSONEachRow`.

行ストリームもあります: `IRowInputStream` と `IRowOutputStream`. ブロックではなく、個々の行でデータをプル/プッシュすることができます。また、行指向形式の実装を簡素化するためにのみ必要です。ラッパー `BlockInputStreamFromRowInputStream` と `BlockOutputStreamFromRowOutputStream` 行指向のストリームを通常のブロック指向のストリームに変換できます。

## I/O

バイト指向の入出力には、次のようなものがあります `ReadBuffer` と `WriteBuffer` 抽象クラス。それらはC++の代わりに使用されます `iostream` 心配しないでください：すべての成熟したC++プロジェクトは、`iostream` 正当な理由のためのS。

`ReadBuffer` と `WriteBuffer` 単なる連続したバッファであり、そのバッファ内の位置を指すカーソルです。実装にはない独自のメモリにバッファです。以下のデータでバッファを埋める仮想メソッドがあります `ReadBuffer` (またはバッファをどこかにフラッシュする (`WriteBuffer`)). 仮想メソッドはまれに呼び出されます。

の実装 `ReadBuffer/WriteBuffer` 圧縮を実装するために、ファイルとファイル記述子とネットワークソケット (`CompressedWriteBuffer` is initialized with another `WriteBuffer` and performs compression before writing data to it), and for other purposes - the names `ConcatReadBuffer`, `LimitReadBuffer`, and `HashingWriteBuffer` 自分のためには話す。

`Read/WriteBuffers` はバイトのみを扱います。からの関数があります `ReadHelpers` と `WriteHelpers` 入力/出力のフォーマットに役立つヘッダファイル。たとえば、小数の形式で数値を書くヘルパーがあります。

結果セットを書きたいときに何が起こるかを見てみましょう `JSON` 標準出力にフォーマットします。結果セットを取得する準備ができています `IBlockInputStream`. 作成する `WriteBufferFromFileDescriptor(STDOUT_FILENO)` `stdout` にバイトを書き込む。作成する `JSONRowOutputStream`、それで初期化されます `WriteBuffer`、行を書き込むには `JSON` 標準出力に。作成する `BlockOutputStreamFromRowOutputStream` その上に、それを次のように表します `IBlockOutputStream`. それから電話する `copyData` データを転送するには `IBlockInputStream` に `IBlockOutputStream` そして、すべてが動作します。内部的には、`JSONRowOutputStream` さまざまな JSON 区切り文字を書き、`IDataType::serializeTextJSON` を参照するメソッド `IColumn` 引数として行番号を指定します。その結果、`IDataType::serializeTextJSON` からメソッドを呼び出します `WriteHelpers.h`: 例えば、`writeText` 数値型および `writeJSONString` のために `DataToString`.

## テーブル

その `IStorage` インタフェースです。異なる実装のインターフェースの異なるテーブルエンジンです。例としては `StorageMergeTree`, `StorageMemory`、というように。これらのクラスのインスタンス

キー `IStorage` メソッドは `read` と `write`. また、`alter`, `rename`, `drop`、というように。その `read` このメソッドは、次の引数を受け入れます。AST 考慮すべきクエリ、および返すストリームの必要な数。一つまたは複数を返します `IBlockInputStream` クエリの実行中にテーブルエンジン内で完了したデータ処理のステージに関するオブジェクトと情報。

ほとんどの場合、`read` メソッドは、指定された列をテーブルから読み取るだけで、それ以降のデータ処理は行いません。すべてのデータ処理が行われるクエリの通訳や外部の責任 `IStorage`.

しかし、顕著な例外があります:

- AST クエリは `read` 法により処理し、テーブルエンジンを使用できる指の利用と読みの少ないデータを表示します。
- 時々のテーブルエンジンを処理できるデータそのものである場合でも特定の段階にある。例えば、`StorageDistributed` リモートサーバーにクエリを送信し、異なるリモートサーバーからのデータをマージできるステージにデータを処理するように依頼し、その前処理されたデータを返すこのクエリの通訳を仕上げ加工のデータです。

テーブルの `read` メソッドは、複数の `IBlockInputStream` 並列データ処理を可能にするオブジェクト。これらの複数のブロックの入力ストリームでテーブルから行なった。次に、これらのストリームを、独立して計算できるさまざまな変換(式の評価やフィルタリングなど)でラップして、`UnionBlockInputStream` それらの上に、並列に複数のストリームから読み込みます。

また、`TableFunction` これらは一時的なものを返す関数です `IStorage` で使用するオブジェクト `FROM` クエリの句。

テーブルエンジンの実装方法の簡単なアイデアを得るには、次のような単純なものを見てください `StorageMemory` または `StorageTinyLog`.

の結果として `read` 方法、`IStorage` ツヅケ。 `QueryProcessingStage` – information about what parts of the query were already calculated inside storage.

## パーサー

手書きの再帰降下パーサーは、クエリを解析します。例えば、`ParserSelectQuery` クエリのさまざまな部分に対して基になるパーサを再帰的に呼び出すだけです。パーサーは `AST`. その `AST` ノードによって表されます。`IAST`.

パーサジェネレータは、使用しない歴史的な理由があります。

## 通訳者

インタプリタは、クエリ実行パイプラインの作成を担当します。AST. 以下のような簡単な通訳があります InterpreterExistsQuery と InterpreterDropQuery またはより洗練された InterpreterSelectQuery. クエリの実行パイプラインの組み合わせたブロック入力または出力ストリーム. たとえば、SELECT クエリは IBlockInputStream 結果セットを読

み取るために、`INSERT`クエリの結果は次のようにになります。`IBlockOutputStream`に挿入するためのデータを書き込むために、および解釈の結果 `INSERT SELECT` クエリは `IBlockInputStream` これは、最初の読み取り時に空の結果セットを返しますが、データをコピーします `SELECT` に `INSERT` 同時に。

`InterpreterSelectQuery` 用途 `ExpressionAnalyzer` と `ExpressionActions` クエリ分析と変換のための機械。ここでは、ほとんどのルールベースのクエリの最適化が行われます。`ExpressionAnalyzer` モジュラー変換やクエリを可能にするために、さまざまなクエリ変換と最適化を別々のクラスに抽出する必要があります。

## 関数

通常の関数と集計関数があります。集計関数については、次の節を参照してください。

Ordinary functions don't change the number of rows – they work as if they are processing each row independently. In fact, functions are not called for individual rows, but for Blockベクトル化されたクエリ実行を実装するためのデータ。

いくつかのその他の機能があります。 ブロックサイズ, `rowNumberInBlock`, and `runningAccumulate`、それはプロック処理を悪用し、行の独立性に違反します。

ClickHouseには強い型指定があるため、暗黙的な型変換はありません。関数が特定の型の組み合わせをサポートしていない場合、例外がスローされます。ものの機能で作業する過負荷のもとに多くの異なる組み合わせます。例えば、`plus` 関数（実装するには + 演算子）数値型の任意の組み合わせに対して動作します: `UInt8 + Float32`, `UInt16 + Int8`、というように。また、一部の可変引数関数は、以下のような任意の数の引数を受け入れることができます。`concat` 機能。

実施の機能が少し不便での機能を明示的に派遣サポートされているデータの種類と対応 `IColumns`. 例えば、`plus` 関数は、数値型の組み合わせごとにC++テンプレートのインスタンス化によって生成されたコード、および定数または非定数左と右の引数を持っていま

テンプレートコードの膨張を避けるために、実行時コード生成を実装するのに最適な場所です。また、`fused multiply-add`のような`fused`関数を追加したり、一つのループ反復で多重比較を行うこともできます。

ベクトル化されたクエリの実行により、関数は短絡されません。たとえば、`WHERE f(x) AND g(y)`, 両側が計算されます、でも行について、とき `f(x)` がゼロである（場合を除く `f(x)` はゼロ定数式である）。しかし、の選択性が `f(x)` 条件は高く、計算の `f(x)` よりもはるかに安いです `g(y)`、マルチパス計算を実装する方が良いでしょう。最初に計算します `f(x)` 次に、結果によって列をフィルター処理し、次に計算します `g(y)` フィルター処理された小さなデータのチャンクのみ。

## 集計関数

集計関数はステートフル関数です。渡された値のある状態に蓄積し、その状態から結果を得ることができます。それらはと管理されます `IAggregateFunction` インタフェース 状態はかなり単純にすることができます（`AggregateFunctionCount` ただの单一です `UInt64` 値）または非常に複雑な（の状態 `AggregateFunctionUniqCombined` 線形配列、ハッシュテーブル、およびaの組み合わせです `HyperLogLog` 確率データ構造）。

状態は `Arena` (メモリプール)高カーディナリティの実行中に複数の状態を処理する `GROUP BY` クエリ。たとえば、複雑な集約状態では、追加のメモリを割り当てることができます。これは、作成し、状態を破壊し、適切にその所有権と破壊命令を渡すためにいくつかの注意が必要です。

集約状態をシリアル化および逆シリアル化して、分散クエリの実行中にネットワーク経由で渡したり、十分なRAMがないディスクに書き込んだりできま それらはあるテーブルで貯えることができます `DataTypeAggregateFunction` データの増分集計を可能にする。

集計関数状態のシリアル化されたデータ形式は、現在バージョン管理されていません。集約状態が一時的にのみ格納されていればokです。しかし、我々は持っている `AggregatingMergeTree` テーブルエンジンが増えた場合の集約、人々に基づき使用されている。これは、将来集計関数のシリアル化形式を変更するときに下位互換性が必要な理由です。

## サーバ

サーバを実装し複数の複数のインターフェース:

- 外部クライアント用のHTTPインターフェイス。
- TCPインターフェースのネイティブClickHouseクライアントとクロス-サーバー通信中に分散クエリを実行します。
- インターフェース転送データレプリケーション。

内部的には、コルーチンやファイバーのない原始的なマルチスレッドサーバーです。サーバーは、単純なクエリの割合が高いのではなく、比較的低い複雑なクエリの割合を処理するように設計されているため、それぞれが分析のために膨大

サーバーは初期化します **Context** クエリ実行に必要な環境を持つクラス: 使用可能なデータベース、ユーザーとアクセス権、設定、クラスター、プロセスリスト、クエリログなどのリスト。通訳者はこの環境を利用します。

古いクライアントは新しいサーバーと話すことができ、新しいクライアントは古いサーバーと話すことができます。しかし、我々は永遠にそれを維持したくない、と我々は約一年後に古いバージョンのサポートを削除しています。

## 注

ほとんどの外部アプリケーションでは、HTTPインターフェイスを使用することをお勧めします。TCPプロトコルは、データのブロックを渡すために内部形式を使用し、圧縮されたデータにはカスタムフレーミングを使用します。まだ公表したCライブラリのためのこのプロトコールが必要なことから、リンクのClickHouseコードベース、るのは現実的ではありません。

## 分散クエリの実行

サーバーにクラスター設定アップがほぼ独立しています。を作成することができます **Distributed** クラスター内のサーバーの表。その **Distributed table does not store data itself - it only provides a "view"** すべての地方のテーブルに複数のノードのクラスター Aから選択すると **Distributed** テーブルは、そのクエリを書き換え、負荷分散設定に従ってリモートノードを選択し、それらにクエリを送信します。その **Distributed** テーブル要求をリモートサーバー処理クエリーだけで最大のペースでの中間結果から異なるサーバできます。その後、中間結果を受信してマージします。のテーブルを配布してできる限りの仕事へのリモートサーバーを送信しない多くの中間データのネットワーク。

INまたはJOIN句にサブクエリがあり、それが **Distributed** テーブル。これらのクエリの実行には、さまざまな戦略があります。

分散クエリ実行用のグローバルクエリプランはありません。各ノードには、ジョブの一部のローカルクエリプランがあります。リモートノードに対してクエリを送信し、結果をマージします。しかし、基底の多いグループBYsを持つ複雑なクエリや、結合のための大量の一時データを持つクエリでは、これは不可能です。そのような場合には、「reshuffle」追加の調整が必要なサーバー間のデータ。ClickHouseはそのようなクエリの実行をサポートしています。

## マージツリー

**MergeTree** 主キーは、列または式の任意のタプルにすることができます。Aのデータ **MergeTree** テーブルは“parts”。各パートはデータを主キーの順序で格納するため、データは主キータプルによって辞書順に並べ替えられます。すべてのテーブル列は別々に格納されます **column.bin** これらの部分のファイル。ファイルは圧縮ブロックで構成されます。各ブロックは、平均値のサイズに応じて、通常64KBから1MBの非圧縮データです。ブロックは、列の値が連続して次々に配置されています。列の値は各列で同じ順序になるため（主キーによって順序が定義されます）、多くの列で反復処理すると、対応する行の値が取得されます。

主キー自体は次のとおりです “sparse”。すべての行に対処するのではなく、いくつかの範囲のデータのみを扱います。別の **primary.idx** fileには、N番目の行ごとに主キーの値があります。**index\_granularity** (通常、N=8192)。また、各列について、我々は持っている **column.mrk** ファイル “marks,” これは、データファイル内のN番目の行ごとにオフセットされます。各マークは、ファイル内の圧縮ブロックの先頭までのオフセットと、圧縮解除ブロック内のデータの先頭までのオフセットのペアです。通常、圧縮されたブロックはマークで整列され、解凍されたブロックのオフセットはゼロです。データのための **primary.idx** 常にメモリ内に存在し、**column.mrk** ファイ

我々は一部から何かを読むつもりですと同時に **MergeTree** 私たちは **primary.idx** データと要求されたデータを含む可能性のある範囲を見つけて、次に **column.mrk** これらの範囲の読み取りを開始する場所のデータと計算オフセット。希薄さのために、余分なデータが読み取られることがあります。ClickHouseは、単純なポイントクエリの高負荷には適していません。**index\_granularity** キーごとに行を読み取り、圧縮されたブロック全体を列ごとに解凍する必要があります。私たちは、インデックスの顕著なメモリ消費なしに单一のサーバーごとに数兆行を維持できる必要があるため、インデックスを疎にしました。また、主キーは疎であるため、一意ではありません。テーブルに同じキーを持つ多くの行を持つことができます。

あなたが **INSERT** データの束に **MergeTree**、その束は主キーの順序でソートされ、新しい部分を形成します。が背景のスレッドを定期的に、選択部分と統合して单一のソート部の部品点数が比較的低い。それが呼び出される理由です **MergeTree**。もちろん、マージは “write amplification”. すべてのパートは不变であり、作成および削除のみが行われますが、変更は行われません。SELECTが実行されると、テーブルのスナップショット(パートのセット)が保持されます。マージ後もしばらくの間、古いパートを保持して、障害後の回復を容易にするため、マージされたパートが壊れている可能性があることがわかつたら、ソースパート

**MergeTree** LSMツリーではありません。“memtable” と “log”: inserted data is written directly to the filesystem. This makes it suitable only to INSERT data in batches, not by individual row and not very frequently – about once per second is ok, but a thousand times a second is not. We did it this way for simplicity's sake, and because we are already inserting data in batches in our applications.

**MergeTree** テーブルには、一つの(プライマリ)インデックスしか持つことができません。たとえば、複数の物理的順序でデータを格納したり、事前に集計されたデータと元のデータを含む表現を許可したりすることもできます。

バックグラウンドマージ中に追加の作業を行っている**MergeTree**エンジンがあります。例としては **CollapsingMergeTree** と **AggregatingMergeTree**。この処理として特別支援しました。なぜなら、ユーザーは通常、バックグラウンドマージが実行される時間とデータを制御することができないからです。**MergeTree** テーブルは、ほとんどの場合、完全にマージされた形式ではなく、複数の部分に格納されます。

## 複製

ClickHouseでのレプリケーションは、テーブルごとに構成できます。きも複製されない一部の複製のテーブルと同じサーバです。また、二要素複製のテーブルと三要素複製のテーブルなど、さまざまな方法でテーブルをレプリケートすることもできます。

レプリケーションは **ReplicatedMergeTree** ストレージエンジンのパス **ZooKeeper** ストレージエンジンのパラメータとして指定します。同じパスを持つすべてのテーブル **ZooKeeper** 互いのレプリカになる：彼らはデータを同期し、一貫性を維持する。レプリカは、テーブルを作成または削除するだけで動的に追加および削除できます。

複製を使用して非同期マルチマスタースキームです。次のセッションを持つ任意のレプリカにデータを挿入できます **ZooKeeper** データを複製、その他すべてのレプリカは非同期的に。ClickHouseは更新をサポートしていないため、複製は競合しません。挿入のクオーラム確認がないため、あるノードに障害が発生すると、挿入されただけのデータが失われる可能性があります。

複製のメタデータはZooKeeperに格納されます。実行するアクションを示すレプリケーションログがあります。アクションは次のとおりです。各レプリカコピー、複製のログをキューにその行動からのキューに挿入します 例えば、挿入では、“get the part” actionを作成し、ログイン、レプリカのダウンロードいます。マージはレプリカ間で調整され、バイトが同一の結果が得られます。すべての部品を合併した場合と同様にすべてのレプリカ。では達成を補い、一つのレプリカのリーダーとして、レプリカを始めと融合し、書き込みます “merge parts” ログへのアクション。

圧縮された部分のみがノード間で転送され、クエリは転送されません。合併処理され、各レプリカ多くの場合、自主的に下げるネットワークコストを回避することによるネットワークが増幅。大合併をさらにネットワークする場合に限り重複製に遅れて波及してきています。

さらに、各レプリカは、その状態をパートとそのチェックサムのセットとしてZooKeeperに格納します。ローカルファイルシステム上の状態がZooKeeperの参照状態から乖離すると、レプリカは他のレプリカから欠落している部分と壊れている部分をダウンロード ローカルファイルシステムに予期しないデータや壊れたデータがある場合、ClickHouseはそれを削除しませんが、別のディレクトリに移動して忘れます。

## 注

ClickHouseクラスターは独立したシャードで構成され、各シャードはレプリカで構成されます。クラスターは彈性ではないしたがって、新しいシャードを追加した後、データはシャード間で自動的に再調整されません。代わりに、クラスタ負荷は不均一に調整されることになっています。この実装はより多くの制御を提供し、数十のノードなどの比較的小さなクラスタでもokです。しかし、運用環境で使用している数百のノードを持つクラスターでは、このアプローチは重大な欠点になります。"を実行すべきである"と述べていテーブルエンジンで広がる、クラスターを動的に再現れる可能性がある地域分割のバランスとクラスターの動します。

## ClickHouseのソースコードを参照

以下を使用できます **Woboq** オンラインのコードブラウザをご利用 [ここに](#)。このコードナビゲーションや意味のハイライト表示、検索インデックス。コードのスナップショットは隨時更新中です。

また、ソースを参照することもできます [GitHub](#) いつものように

使用するIDEに興味がある場合は、CLion、QT Creator、VS Code、KDevelop（注意点あり）をお勧めします。任意の好きなIDEを使用できます。VimとEmacsもカウントされます。

## 開発のためのClickHouseを構築する方法

次のチュートリアルはUbuntu Linuxシステムに基づいています。

適切な変更により、他のLinuxディストリビューションでも動作するはずです。

サポートされるプラットフォーム:x86\_64およびAArch64。Power9のサポートは実験的です。

## Git、CMake、Pythonと忍者をインストールします

```
$ sudo apt-get install git cmake python ninja-build
```

古いシステムではcmakeの代わりにcmake3。

## Clang 11 のインストール

On Ubuntu/Debian you can use the automatic installation script (check [official webpage](#))

```
sudo bash -c "$(wget -O - https://apt.llvm.org/llvm.sh)"
```

```
$ export CC=clang  
$ export CXX=clang++
```

## ツツイツ姪"ツ債ツツケ

```
$ git clone --recursive git@github.com:ClickHouse/ClickHouse.git
```

または

```
$ git clone --recursive https://github.com/ClickHouse/ClickHouse.git
```

## ビルド ClickHouse

```
$ cd ClickHouse  
$ mkdir build  
$ cd build  
$ cmake ..  
$ ninja  
$ cd ..
```

実行可能ファイルを作成するには、`ninja clickhouse`。

これは作成します `programs/clickhouse` 実行可能ファイル `client` または `server` 引数。

## 任意のLinux上でClickHouseを構築する方法

の構築が必要で以下のコンポーネント：

- Git（ソースをチェックアウトするためにのみ使用され、ビルドには必要ありません）
- CMake3.10以降
- 忍者（推奨）または作る
- C++コンパイラ:clang11以降
- リンカ:lldまたはgold(古典的なGNU ldは動作しません)
- Python(LLVMビルド内でのみ使用され、オプションです)

すべてのコンポーネントがインストールされている場合、上記の手順と同じ方法でビルドできます。

Ubuntu Eoanの例：

```
sudo apt update  
sudo apt install git cmake ninja-build g++ python  
git clone --recursive https://github.com/ClickHouse/ClickHouse.git  
mkdir build && cd build  
cmake .../ClickHouse  
ninja
```

OpenSUSEタンブルウィードの例：

```
sudo zypper install git cmake ninja gcc-c++ python lld  
git clone --recursive https://github.com/ClickHouse/ClickHouse.git  
mkdir build && cd build  
cmake .../ClickHouse  
ninja
```

Fedora Rawhideの例：

```
sudo yum update  
yum --nogpg install git cmake make gcc-c++ python3  
git clone --recursive https://github.com/ClickHouse/ClickHouse.git  
mkdir build && cd build  
cmake .../ClickHouse  
make -j $(nproc)
```

## ClickHouseを構築する必要はありません

ClickHouseは、事前に構築されたバイナリとパッケージで利用可能です。バイナリは移植性があり、任意のLinuxフレーバーで実行できます。

これらのために、安定したprestable-試験スリリースして毎にコミットマスターすべてを引きます。

から新鮮なビルドを見つけるには master,に行く [コミットページ](#) 最初の緑色のチェックマークまたはコミットの近くにある赤い十字をクリックし、“Details”右の後にリンク “ClickHouse Build Check”.

## ClickHouse Debianパッケージのビルド方法

### GitとPbuilderのインストール

```
$ sudo apt-get update  
$ sudo apt-get install git python pbuilder debhelper lsb-release fakeroot sudo debian-archive-keyring debian-keyring
```

### ツツイツ姪ツツイツケ

```
$ git clone --recursive --branch master https://github.com/ClickHouse/ClickHouse.git  
$ cd ClickHouse
```

### 解放スクリプトの実行

```
$ ./release
```

## Mac OS XでClickHouseを構築する方法

ビルドはMac OS X10.15(Catalina)

### 自作のインストール

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

### 設置に必要なコンパイラ、ツール、図書館

```
$ brew install cmake ninja libtool gettext
```

### ツツイツ姪ツツイツケ

```
$ git clone --recursive git@github.com:ClickHouse/ClickHouse.git
```

または

```
$ git clone --recursive https://github.com/ClickHouse/ClickHouse.git  
$ cd ClickHouse
```

### ビルド ClickHouse

```
$ mkdir build  
$ cd build  
$ cmake .. -DCMAKE_CXX_COMPILER=`which clang++` -DCMAKE_C_COMPILER=`which clang`  
$ ninja  
$ cd ..
```

警告

Clickhouse-serverを実行する場合は、システムのmaxfiles変数を増やしてください。

## 注

Sudoを使用する必要があります。

これを行うには、次のファイルを作成します:

/ライブラリ/LaunchDaemons/limit.マックスファイルリスト:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
  "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>limit.maxfiles</string>
  <key>ProgramArguments</key>
  <array>
    <string>launchctl</string>
    <string>limit</string>
    <string>maxfiles</string>
    <string>524288</string>
    <string>524288</string>
  </array>
  <key>RunAtLoad</key>
  <true/>
  <key>ServiceIPC</key>
  <false/>
</dict>
</plist>
```

次のコマンドを実行します:

```
$ sudo chown root:wheel /Library/LaunchDaemons/limit.maxfiles.plist
```

再起動しろ

チェックの場合は、利用できる `ulimit -n` コマンド

## Mac OS X用のLinux上でClickHouseを構築する方法

これは、Linuxマシンを使用してビルドする場合のためのものです `clickhouse` これは、Linuxサーバー上で実行される継続的な統合チェックを目的としています。 Mac OS X上でClickHouseを直接ビルドする場合は、次の手順に進みます [別の命令](#)。

Mac OS X用のクロスビルドは [ビルド命令](#) 先について来い

## Clang-8をインストール

の指示に従ってください<https://apt.llvm.org/>あなたのUbuntuまたはDebianのセットアップ用。

例えば、コマンドバイオニックのような:

```
sudo echo "deb [trusted=yes] http://apt.llvm.org/bionic/ llvm-toolchain-bionic-8 main" >> /etc/apt/sources.list
sudo apt-get install clang-8
```

## クロスコンパイルツールセット

インストール先のパスを覚えてみましょう `cctools` として\${CCTOOLS}

```
mkdir ${CCTOOLS}

git clone https://github.com/tpoechtrager/apple-libtapi.git
cd apple-libtapi
INSTALLPREFIX=${CCTOOLS} ./build.sh
./install.sh
cd ..

git clone https://github.com/tpoechtrager/cctools-port.git
cd cctools-port/cctools
./configure --prefix=${CCTOOLS} --with-libtapi=${CCTOOLS} --target=x86_64-apple-darwin
make install
```

また、作業ツリーにmacOS X SDKをダウンロードする必要があります。

```
cd ClickHouse
wget 'https://github.com/phracker/MacOSX-SDKs/releases/download/10.15/MacOSX10.15.sdk.tar.xz'
mkdir -p build-darwin/cmake/toolchain/darwin-x86_64
tar xJf MacOSX10.15.sdk.tar.xz -C build-darwin/cmake/toolchain/darwin-x86_64 --strip-components=1
```

## ビルド ClickHouse

```
cd ClickHouse
mkdir build-osx
CC=clang-8 CXX=clang++-8 cmake . -Bbuild-osx -DCMAKE_TOOLCHAIN_FILE=cmake/darwin/toolchain-x86_64.cmake \
-DCMAKE_AR:FILEPATH=${CCTOOLS}/bin/x86_64-apple-darwin-ar \
-DCMAKE_RANLIB:FILEPATH=${CCTOOLS}/bin/x86_64-apple-darwin-ranlib \
-DLINKER_NAME=${CCTOOLS}/bin/x86_64-apple-darwin-ld
ninja -C build-osx
```

結果のバイナリはmach-O実行可能フォーマットを持ち、Linux上で実行することはできません。

## Aarch64(ARM64)アーキテクチャ用のLinux上で ClickHouseを構築する方法

これは、Linuxマシンを使用してビルドする場合のためのものです `clickhouse AARCH64CPU`アーキテクチャを持つ別のLinuxマシン上で実行されるバイナリ。この目的のために継続的インテグレーションをチェックを実行Linuxサーバー

AARCH64のクロスピルドは [ビルド命令](#) 先について来い

## Clang-8をインストール

の指示に従ってください<https://apt.llvm.org/>あなたのUbuntuまたはDebianのセットアップ用。  
たとえば、Ubuntu Bionicでは、次のコマンドを使用できます:

```
echo "deb [trusted=yes] http://apt.llvm.org/bionic/ llvm-toolchain-bionic-8 main" | sudo tee
/etc/apt/sources.list.d/llvm.list
sudo apt-get update
sudo apt-get install clang-8
```

## クロスコンパイルツールセット

```
cd ClickHouse
mkdir -p build-aarch64/cmake/toolchain/linux-aarch64
wget 'https://developer.arm.com/-/media/Files/downloads/gnu-a/8.3-2019.03/binrel/gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu.tar.xz?revision=2e88a73f-d233-4f96-b1f4-d8b36e9bb0b9&la=en' -O gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu.tar.xz
tar xf gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu.tar.xz -C build-aarch64/cmake/toolchain/linux-aarch64 --strip-components=1
```

## ビルド ClickHouse

```
cd ClickHouse
mkdir build-arm64
CC=clang-8 CXX=clang++-8 cmake . -Bbuild-arm64 -DCMAKE_TOOLCHAIN_FILE=cmake/linux/toolchain-aarch64.cmake
ninja -C build-arm64
```

結果のバイナリは、Aarch64CPUアーキテクチャを持つLinux上でのみ実行されます。

## C++コードの書き方

### 一般的な推奨事項

1. 以下は推奨事項であり、要件ではありません。
2. コードを編集する場合は、既存のコードの書式に従うことが理にかなっています。
3. 一貫性のためにコードスタイルが必要です。一貫性により、コードを読みやすくなり、コードの検索も簡単になります。
4. ルールの多くは論理的な理由を持っていない;彼らは確立された慣行によって決定されます。

### 書式設定

1. 多くのフォーマットは自動的に実行されるので `clang-format`.
2. インデントは4スペースです。タブにスペースが追加されるように開発環境を構成します。
3. 中括弧の開始と終了は別の行にする必要があります。

```
inline void readBoolText(bool & x, ReadBuffer & buf)
{
    char tmp = '0';
    readChar(tmp, buf);
    x = tmp != '0';
}
```

4. 関数本体全体が单一の場合 `statement`、それは單一行に置くことができます。中括弧の周りにスペースを配置します(行末のスペースのほかに)。

```
inline size_t mask() const { return buf_size() - 1; }
inline size_t place(HashValue x) const { return x & mask(); }
```

5. 機能のため。をかけないスットに固定して使用します。

```
void reinsert(const Value & x)
```

```
memcpy(&buf[place_value], &x, sizeof(x));
```

6. で `if`, `for`, `while` 他の式では、（関数呼び出しとは対照的に）開始括弧の前にスペースが挿入されます。

```
for (size_t i = 0; i < rows; i += storage.index_granularity)
```

7. 二項演算子の周りにスペースを追加 (+, -, \*, /, %, ...) and the ternary operator ?:.

```
UInt16 year = (s[0] - '0') * 1000 + (s[1] - '0') * 100 + (s[2] - '0') * 10 + (s[3] - '0');
UInt8 month = (s[5] - '0') * 10 + (s[6] - '0');
UInt8 day = (s[8] - '0') * 10 + (s[9] - '0');
```

8. 改行が入力されている場合は、演算子を新しい行に置き、その前にインデントを増やします。

```
if (elapsed_ns)
    message << "("
        << rows_read_on_server * 1000000000 / elapsed_ns << " rows/s., "
        << bytes_read_on_server * 1000.0 / elapsed_ns << " MB/s.) ";
```

9. 必要に応じて、行内の整列にスペースを使用できます。

```
dst.ClickLogID      = click.LogID;
dst.ClickEventID    = click.EventID;
dst.ClickGoodEvent  = click.GoodEvent;
```

10. 演算子の周りにスペースを使用しない `, ->`.

必要に応じて、演算子を次の行に折り返すことができます。この場合、その前のオフセットが増加する。

11. 単項演算子の区切りにスペースを使用しないでください `(-, ++, *, &, ...)` from the argument.

12. 入れの後に空白、コンマといえるものです。同じルールは、内部のセミコロンのために行く `for` 式。

13. 区切りにスペースを使用しないで下さい `[]` オペレーター

14. で `template <...>` 式は、間にスペースを使用します `template` と `<`;後にスペースなし `<` または前に `>`.

```
template <typename TKey, typename TValue>
struct AggregatedStatElement
{}
```

15. クラスと構造体では、`public`, `private`, and `protected` と同じレベルで `class/struct` コードの残りの部分をインデントします。

```
template <typename T>
class MultiVersion
{
public:
    /// Version of object for usage. shared_ptr manage lifetime of version.
    using Version = std::shared_ptr<const T>;
    ...
}
```

16. 同じ場合 `namespace` ファイル全体に使用され、他に重要なものはありません。`namespace`.

17. のブロックが `if`, `for`, `while`、または他の式は、单一の `statement`、中括弧は省略可能です。を置く `statement` 別の行で、代わりに。この規則は、入れ子に対しても有効です `if`, `for`, `while`, ...

しかし、内側の場合 `statement` 中括弧または `else`、外部ブロックは中括弧で記述する必要があります。

```
/// Finish write.  
for (auto & stream : streams)  
    stream.second->finalize();
```

**18.** 行の終わりにスペースがあつてはなりません。

**19.** ソースファイルはUTF-8エンコードです。

**20.** ASCII以外の文字は、文字列リテラルで使用できます。

```
<< ", " << (timer.elapsed() / chunks_stats.hits) << " µsec(hit.);
```

**21.** 単一行に複数の式を記述しないでください。

**22.** 関数内のコードのセクションをグループ化し、空行を複数以下で区切ります。

**23.** 関数、クラスなどを空行で区切ります。

**24.** A `const` (値に関連する)型名の前に記述する必要があります。

```
//correct  
const char * pos  
const std::string & s  
//incorrect  
char const * pos
```

**25.** ポインタまたは参照を宣言するとき、`*` と `&` シンボルは、両側のスペースで区切る必要があります。

```
//correct  
const char * pos  
//incorrect  
const char* pos  
const char *pos
```

**26.** テンプレートタイプを使用するときは、それらを `using` キーワード(最も単純な場合を除く)。

つまり、テンプレートパラメーターは、`using` コードでは繰り返されません。

`using` 関数内など、ローカルで宣言できます。

```
//correct  
using FileStreams = std::map<std::string, std::shared_ptr<Stream>>;  
FileStreams streams;  
//incorrect  
std::map<std::string, std::shared_ptr<Stream>> streams;
```

**27.** 一つの文で異なる型の変数を複数宣言しないでください。

```
//incorrect  
int x, *y;
```

**28.** Cスタイルのキャストは使用しないでください。

```
//incorrect  
std::cerr << (int)c << std::endl;  
//correct  
std::cerr << static_cast<int>(c) << std::endl;
```

**29.** クラスと構造体では、各可視スコープ内でメンバーと関数を個別にグループ化します。

**30.** 小さなクラスと構造体の場合、メソッド宣言を実装から分離する必要はありません。

同じことが、クラスや構造体の小さなメソッドにも当てはまります。

テンプレート化されたクラスと構造体の場合、メソッド宣言を実装から分離しないでください（そうでない場合は、同じ翻訳単位で定義する必要があります）

**31.** 行は140文字で、80文字ではなく折り返すことができます。

**32.** Postfixが必要ない場合は、常に接頭辞の増分/減分演算子を使用します。

```
for (Names::const_iterator it = column_names.begin(); it != column_names.end(); ++it)
```

## コメント

**1.** コードのすべての非自明な部分にコメントを追加してください。

これは非常に重要です。書面でのコメントだけを更新したいのですが--このコードに必要な、又は間違っています。

```
/** Part of piece of memory, that can be used.  
 * For example, if internal_buffer is 1MB, and there was only 10 bytes loaded to buffer from file for reading,  
 * then working_buffer will have size of only 10 bytes  
 * (working_buffer.end() will point to position right after those 10 bytes available for read).  
 */
```

**2.** コメントは必要に応じて詳細に設定できます。

**3.** 記述するコードの前にコメントを配置します。まれに、コメントが同じ行のコードの後に来ることがあります。

```
/** Parses and executes the query.  
 */  
void executeQuery(  
    ReadBuffer & istr, /// Where to read the query from (and data for INSERT, if applicable)  
    WriteBuffer & ostr, /// Where to write the result  
    Context & context, /// DB, tables, data types, engines, functions, aggregate functions...  
    BlockInputStreamPtr & query_plan, /// Here could be written the description on how query was executed  
    QueryProcessingStage::Enum stage = QueryProcessingStage::Complete /// Up to which stage process the SELECT  
    query  
)
```

**4.** コメントは英語のみで書く必要があります。

**5.** を書いていて図書館を含む詳細なコメントで説明を主なヘッダファイルです。

**6.** 追加情報を提供しないコメントは追加しないでください。特に放置しないでください空のコメントこのような：

```
/*
 * Procedure Name:
 * Original procedure name:
 * Author:
 * Date of creation:
 * Dates of modification:
 * Modification authors:
 * Original file name:
 * Purpose:
 * Intent:
 * Designation:
 * Classes used:
 * Constants:
 * Local variables:
 * Parameters:
 * Date of creation:
 * Purpose:
 */
```

この例はリソースから借用されています <http://home.tamk.fi/~jaalto/course/coding-style/doc/unmainainable-code/>.

**7.** ごみのコメントを書かないでください（作成者、作成日。.)各ファイルの先頭にある。

**8.** 単一行のコメントはスラッシュで始まります: `///` 複数行のコメントは `/**`. これらのコメントは、“documentation”.

注:Doxxygenを使用すると、これらのコメントからドキュメントを生成できます。しかし、IDEでコードをナビゲートする方が便利なので、Doxxygenは一般的には使用されません。

**9.** 複数行コメントの先頭と末尾に空行を含めることはできません(複数行コメントを閉じる行を除きます)。

**10.** コメント行コードは、基本的なコメントは、“documenting” コメント。

**11.** 削除のコメントアウトされていバーツのコード深い。

**12.** コメントやコードで冒流を使用しないでください。

**13.** 大文字は使用しないでください。過度の句読点を使用しないでください。

```
/// WHAT THE FAIL???
```

**14.** 使用しないコメントをdelimeters.

```
///*****
```

**15.** コメントで議論を開始しないでください。

```
/// Why did you do this stuff?
```

**16.** それが何であったかを説明するブロックの最後にコメントを書く必要はありません。

```
/// for
```

## 名前

**1.** 変数とクラスメンバーの名前にアンダースコア付きの小文字を使用します。

```
size_t max_block_size;
```

**2.** 関数（メソッド）の名前には、小文字で始まるcamelCaseを使用します。

```
std::string getName() const override { return "Memory"; }
```

**3.** クラス(構造体)の名前には、大文字で始まるCamelCaseを使用します。I以外の接頭辞はインターフェイスには使用されません。

```
class StorageMemory : public IStorage
```

**4.** using クラスと同じように、または\_t 最後に。

**5.** テンプレート型引数の名前:単純な場合は、次のようにします T; T, U; T1, T2.

より複雑な場合は、クラス名の規則に従うか、プレフィックスを追加します T.

```
template <typename TKey, typename TValue>
struct AggregatedStatElement
```

**6.** テンプレート定数引数の名前:変数名の規則に従うか、または N 簡単なケースでは。

```
template <bool without_www>
struct ExtractDomain
```

**7.** 抽象クラス(インターフェイス)については、I プレフィックス

```
class IBlockInputStream
```

**8.** 変数をローカルで使用する場合は、短い名前を使用できます。

それ以外の場合は、意味を説明する名前を使用します。

```
bool info_successfully_loaded = false;
```

**9.** の名前 defines およびグローバル定数を使用 ALL\_CAPS をアンダースコア(\_).

```
##define MAX_SRC_TABLE_NAMES_TO_STORE 1000
```

**10.** ファイル名は内容と同じスタイルを使用する必要があります。

ファイルに单一のクラスが含まれている場合は、クラス(CamelCase)と同じ方法でファイルに名前を付けます。

ファイルに单一の関数が含まれている場合は、関数(camelCase)と同じ方法でファイルに名前を付けます。

**11.** 名前に略語が含まれている場合は、:

- 変数名の場合、省略形は小文字を使用する必要があります mysql\_connection (ない mySQL\_connection).
- クラスおよび関数の名前については、省略形に大文字を使用します MySQLConnection (ない MySqlConnection).

**12.** クラスメンバーを初期化するためだけに使用されるコンストラクター引数には、クラスメンバーと同じ名前を付ける必要がありますが、最後にアンダースコ

```
FileQueueProcessor(
    const std::string & path_,
    const std::string & prefix_,
    std::shared_ptr<FileHandler> handler_
    : path(path_),
    prefix(prefix_),
    handler(handler_),
    log(&Logger::get("FileQueueProcessor"))
{
}
```

引数がコンストラクタ本体で使用されていない場合は、アンダースコアの接尾辞を省略できます。

**13.** ローカル変数とクラスメンバーの名前に違いはありません（接頭辞は必要ありません）。

```
timer (not m_timer)
```

**14.** の定数に対して `enum`、大文字でキャメルケースを使用します。 `ALL_CAPS` も許容されます。もし `enum` は非ローカルである。`enum class`。

```
enum class CompressionMethod
{
    QuickLZ = 0,
    LZ4      = 1,
};
```

**15.** すべての名前は英語である必要があります。ロシア語の音訳は許可されません。

```
not Stroka
```

**16.** 略語は、よく知られている場合（Wikipediaや検索エンジンで略語の意味を簡単に見つけることができる場合）に許容されます。

```
`AST`, `SQL`.
```

```
Not `NVDH` (some random letters)
```

短縮版が一般的に使用されている場合、不完全な単語は許容されます。

コメントの横にフルネームが含まれている場合は、略語を使用することもできます。

**17.** C++のソースコードを持つファイル名は、`.cpp` 延長。ヘッダファイルには `.h` 延長。

## コードの書き方

**1.** メモリ管理。

手動メモリ解放 (`delete`) ライブドリコードでのみ使用できます。

ライブドリコードでは、`delete operator` はデストラクタでのみ使用できます。

アプリケーション

例:

- 最も簡単な方法は、スタック上にオブジェクトを配置するか、別のクラスのメンバーにすることです。
- 多数の小さなオブジェクトの場合は、コンテナを使用します。

- ヒープ内に存在する少數のオブジェクトの自動割り当て解除には、以下を使用します `shared_ptr/unique_ptr`.

## 2. リソース管理。

使用 `RAII` 上記を参照してください。

## 3. エラー処理。

例外を使用します。ほとんどの場合、例外をスローするだけで、それをキャッチする必要はありません（`RAII`）。

オフライ

ユーザー要求を処理するサーバーでは、通常、接続ハンドラの最上位レベルで例外をキャッチするだけで十分です。

スレッド機能、とくすべての例外 `rethrow` のメインスレッド後 `join`.

```
/// If there weren't any calculations yet, calculate the first block synchronously
if (!started)
{
    calculate();
    started = true;
}
else // If calculations are already in progress, wait for the result
    pool.wait();

if (exception)
    exception->rethrow();
```

ない非表示の例外なります。すべての例外を盲目的に記録するだけではありません。

```
//Not correct
catch (...) {}
```

いくつかの例外を無視する必要がある場合は、特定の例外に対してのみ実行し、残りを再スローします。

```
catch (const DB::Exception & e)
{
    if (e.code() == ErrorCodes::UNKNOWN_AGGREGATE_FUNCTION)
        return nullptr;
    else
        throw;
}
```

応答コードまたは関数を使用する場合 `errno`、常に結果をチェックし、エラーの場合は例外をスローします。

```
if (0 != close(fd))
    throwError("Cannot close file " + file_name, ErrorCodes::CANNOT_CLOSE_FILE);
```

Do not use `assert`.

## 4. 例外タイプ。

アプリケーションコードで複雑な例外階層を使用する必要はありません。例外テキストは、システム管理者が理解できるはずです。

## 5. 投げから例外がスローされる場合 `destructors`.

これは推奨されませんが、許可されています。

次のオプションを使用しま:

- 関数の作成 (`done()` または `finalize()`) それは例外につながる可能性のあるすべての作業を事前に行います。その関数が呼び出された場合、後でデストラクタに例外はないはずです。
- タスクも複雑で(メッセージを送信するなどのネットワーク)を置くことができます別の方法は、クラスのユーザーを呼び出す前に破壊。
- デストラクタに例外がある場合は、それを非表示にするよりもログに記録する方が良いでしょう（ロガーが利用可能な場合）。
- 簡単な適用では、頼ることは受諾可能です `std::terminate` (以下の場合 `noexcept` デフォルトでは`c++11`) 例外を処理する。

## 6. 匿名コードブロック。

特定の変数をローカルにするために、単一の関数内に別のコードブロックを作成して、ブロックを終了するときにデストラクターが呼び出されるようにす

```
Block block = data.in->read();

{
    std::lock_guard<std::mutex> lock(mutex);
    data.ready = true;
    data.block = block;
}

ready_any.set();
```

## 7. マルチスレッド

オフライ：

- 単一のCPUコアで可能な限り最高のパフォーマンスを得ようとします。必要に応じて、コードを並列化できます。

サーバーアプリ：

- スレッドプールを使用して要求を処理します。この時点で、まだタスクを必要とした管理コストイッキング時の値です。※

`Fork`は並列化には使用されません。

## 8. 同期スレッド。

くすることが可能で別のスレッドが別のメモリー細胞により異なるキャッシュ回線)を使用していないスレッドが同期を除く `joinAll`).

同期が必要な場合は、ほとんどの場合、以下の条件で`mutex`を使用すれば十分です `lock_guard`.

他の場合は、システム同期プリミティブを使用します。`Busy wait`は使用しないでください。

原子演算は、最も単純な場合にのみ使用する必要があります。

主な専門分野でない限り、ロックフリーのデータ構造を実装しようとしてください。

## 9. ポインタ参照。

ほとんどの場合、参照を好む。

## 10. `const`.

定数参照、定数へのポインタを使用する、`const_iterator`、および`const`メソッド。

考慮する `const` デフォルトで非を使用するには`-const`必要なときだけ。

変数を値で渡すときは、`const` 通常は意味がありません。

## 11. 無署名

使用 `unsigned` 必要に応じて。

## 12. 数値型。

タイプを使用する `UInt8, UInt16, UInt32, UInt64, Int8, Int16, Int32, and Int64` だけでなく、`size_t, ssize_t, and ptrdiff_t`.

これらの型を数値に使用しないでください: `signed/unsigned long, long long, short, signed/unsigned char, char`.

## 13. 引数を渡す。

参照による複素数値の渡し(含む `std::string`).

関数がヒープ内に作成されたオブジェクトの所有権を取得する場合は、引数の型を作成します `shared_ptr` または `unique_ptr`.

## 14. 戻り値。

ほとんどの場合、`return`. 書かない `return std::move(res)`.

関数がオブジェクトをヒープに割り当てて返す場合は、次のようにします `shared_ptr` または `unique_ptr`.

まれに、引数を使用して値を返す必要がある場合があります。この場合、引数は参照でなければなりません。

```
using AggregateFunctionPtr = std::shared_ptr<IAggregateFunction>;  
  
/** Allows creating an aggregate function by its name.  
 */  
class AggregateFunctionFactory  
{  
public:  
    AggregateFunctionFactory();  
    AggregateFunctionPtr get(const String & name, const DataTypes & argument_types) const;
```

## 15. 名前空間。

別のものを使用する必要はありません `namespace` 適用コードのため。

小さな図書館でもこれは必要ありません。

中規模から大規模のライブラリの場合は、すべてを `namespace`.

図書館の `.h` ファイル、使用できます `namespace detail` アプリケーションコードに必要な実装の詳細を非表示にする。

で `.cpp` を使用することができます `static` またはシンボルを非表示にする匿名の名前空間。

また、`namespace` に使用することができます `enum` 対応する名前が外部に落ちないようにするには `namespace` (しかし、それを使用する方が良いです `enum class`).

## 16. 遅延初期化。

初期化に引数が必要な場合は、通常は既定のコンストラクタを記述しないでください。

後で初期化を遅らせる必要がある場合は、無効なオブジェクトを作成する既定のコンストラクターを追加できます。または、少数のオブジェクトの場合は、以下を使用できます `shared_ptr/unique_ptr`.

```
Loader(DB::Connection * connection_, const std::string & query, size_t max_block_size_);  
  
/// For deferred initialization  
Loader() {}
```

## 17. 仮想関数。

クラスが多態的な使用を意図していない場合、関数を仮想にする必要はありません。これはデストラクタにも当てはまります。

## 18. エンコーディング

どこでもUTF-8を使用します。使用 `std::string` と `char*`。使用しない `std::wstring` と `wchar_t`。

## 19. ロギング

コードのどこでも例を参照してください。

コミットする前に、無意味なログやデバッグログ、その他のデバッグ出力をすべて削除します。

トレースレベルでも、サイクルのログインは避けるべきです。

ログには必読でログインです。

ログインできるアプリケーションコードにすることができます。

ログメッセージは英語で書く必要があります。

ログは、システム管理者が理解できるようにしてください。

ログに冒頭を使用しないでください。

ログでUTF-8エンコーディングを使用します。まれに、ログに非ASCII文字を使用できます。

## 20. 入出力。

使用しない `iostreams` 内部のサイクルにおいて不可欠な存在であるためのアプリケーション性能(い利用 `stringstream`)。

使用する `DB/IO` 代わりに図書館。

## 21. 日付と時刻。

を参照。`DateLUT` 図書館

## 22. 含める。

常に使用 `#pragma once` 代わりに警備員を含めます。

## 23. を使用して。

`using namespace` は使用されません。以下を使用できます `using` 特定の何かと。しかし、クラスや関数内でローカルにします。

## 24. 使用しない `trailing return type` 必要がない限り機能のため。

```
auto f() -> void
```

## 25. 変数の宣言と初期化。

```
//right way
std::string s = "Hello";
std::string s{"Hello"};

//wrong way
auto s = std::string{"Hello"};
```

## 26. のための仮想関数を書く `virtual` 基本クラスでは、`override` 代わりに `virtual` 子孫クラスで。

# C++の未使用の機能

1. 仮想継承は使用されません。
2. C++03の例外指定子は使用されません。

## プラット

1. を書いていますコードの特定の。

それが同じ場合には、クロス-プラットフォームまたは携帯コードが好ましい。

2. 言語:C++20.
3. コンパイラ: `gcc`. 2020年現在、コードはバージョン9.3を使用してコンパイルされている。（以下を使ってコンパイルできます `clang 8`。）

標準ライブラリが使用されます (`libc++`).

- 4.OS : LinuxのUbuntuの、正確よりも古いではありません。

- 5.コードはx86\_64CPUアーキテクチャ用に書かれている。

CPU命令セットは、サーバー間でサポートされる最小のセットです。現在、SSE4.2です。

6. 使用 `-Wall -Wextra -Werror` コンパイルフラグ。
7. 静的に接続することが困難なライブラリを除くすべてのライブラリとの静的リンクを使用します。`ldd` コマンド）。
8. コードはリリース設定で開発およびデバッグされます。

## ツール

1. KDevelopは良いIDEです。
2. デバッグの使用 `gdb`, `valgrind (memcheck)`, `strace`, `-fsanitize=...`, または `tcmalloc_minimal_debug`.
3. のためのプロファイリングを使用 `Linux Perf`, `valgrind (callgrind)` 、または `strace -cf`.
4. ソースはGitにあります。
5. アセンブリの使用 `CMake`.
6. プログラムは `deb` パッケージ。
7. ることを約束し、マスターが破ってはいけないの。

選択したリビジョンのみが実行可能とみなされます。

8. コードが部分的にしか準備できていなくても、できるだけ頻繁にコミットを行います。
- この目的のために分岐を使用します。
- あなたのコードが `master branch` はまだビルド可能ではありません。`push`. あなたはそれを終了するか、数日以内にそれを削除する必要があります。
9. 些細な変更ではない場合は、ブランチを使用してサーバーに公開します。
10. 未使用的コードがリポジトリから削除されます。

## 図書館

1. The C++20 standard library is used (experimental extensions are allowed), as well as boost and Poco frameworks.
2. It is not allowed to use libraries from OS packages. It is also not allowed to use pre-installed libraries. All libraries should be placed in form of source code in contrib directory and built with ClickHouse.
3. Preference is always given to libraries that are already in use.

## 一般的な推奨事項

1. できるだけ少ないコードを書く。
2. う最も単純な解決策です。
3. どのように動作し、内部ループがどのように機能するかを知るまで、コードを記述しないでください。
4. 最も単純なケースでは、`using` クラスや構造体の代わりに。
5. 可能であれば、コピー構造子、代入演算子、デストラクター(クラスに少なくとも一つの仮想関数が含まれている場合は仮想関数を除く)、コンストラクタつまり、コンパイラで生成された関数は正しく動作する必要があります。以下を使用できます `default`.
6. コードの簡素化が推奨されます。可能であれば、コードのサイズを小さくします。

## その他の推奨事項

1. 明示的に指定する `std::` からのタイプのため `stddef.h`

推奨されません。つまり、書くことをお勧めします `size_t` 代わりに `std::size_t` それは短いので。

追加することは許容されます `std::`.

2. 明示的に指定する `std::` 標準Cライブラリの関数の場合

推奨されません。言い換えれば、`memcpy` 代わりに `std::memcpy`.

その理由は、次のような非標準的な機能があるからです `memmem`. 私たちは機会にこれらの機能を使用します。これらの関数は、`namespace std`.

あなたが書く場合 `std::memcpy` 代わりに `memcpy` どこでも、その後、`memmem` なし `std::` 奇妙に見えます。

それにもかかわらず、`std::` あなたがそれを好むなら。

3. 標準C++ライブラリで同じ関数が使用できる場合、Cからの関数を使用します。

これは、より効率的であれば許容されます。

たとえば、次を使用します `memcpy` 代わりに `std::copy` メモリの大きな塊をコピーするため。

4. 複数行の関数の引数。

の他の包装のスタイルを許可:

```
function(  
    T1 x1,  
    T2 x2)
```

```
function(  
    size_t left, size_t right,  
    const & RangesInDataParts ranges,  
    size_t limit)
```

```
function(size_t left, size_t right,
        const & RangesInDataParts ranges,
        size_t limit)
```

```
function(size_t left, size_t right,
        const & RangesInDataParts ranges,
        size_t limit)
```

```
function(
    size_t left,
    size_t right,
    const & RangesInDataParts ranges,
    size_t limit)
```

## ClickHouse Testing

### Functional Tests

Functional tests are the most simple and convenient to use. Most of ClickHouse features can be tested with functional tests and they are mandatory to use for every change in ClickHouse code that can be tested that way.

Each functional test sends one or multiple queries to the running ClickHouse server and compares the result with reference.

Tests are located in `queries` directory. There are two subdirectories: `stateless` and `stateful`. Stateless tests run queries without any preloaded test data - they often create small synthetic datasets on the fly, within the test itself. Stateful tests require preloaded test data from Yandex.Metrica and it is available to general public.

Each test can be one of two types: `.sql` and `.sh`. `.sql` test is the simple SQL script that is piped to `clickhouse-client --multิquery --testmode`. `.sh` test is a script that is run by itself. SQL tests are generally preferable to `.sh` tests. You should use `.sh` tests only when you have to test some feature that cannot be exercised from pure SQL, such as piping some input data into `clickhouse-client` or testing `clickhouse-local`.

### Running a Test Locally

Start the ClickHouse server locally, listening on the default port (9000). To run, for example, the test `01428_hash_set_nan_key`, change to the repository folder and run the following command:

```
PATH=$PATH:<path to clickhouse-client> tests/clickhouse-test 01428_hash_set_nan_key
```

For more options, see `tests/clickhouse-test --help`. You can simply run all tests or run subset of tests filtered by substring in test name: `./clickhouse-test` substring. There are also options to run tests in parallel or in randomized order.

### Adding a New Test

To add new test, create a `.sql` or `.sh` file in `queries/0_stateless` directory, check it manually and then generate `.reference` file in the following way: `clickhouse-client -n --testmode < 00000_test.sql > 00000_test.reference` or `./00000_test.sh > ./00000_test.reference`.

Tests should use (create, drop, etc) only tables in `test` database that is assumed to be created beforehand; also tests can use temporary tables.

## Choosing the Test Name

The name of the test starts with a five-digit prefix followed by a descriptive name, such as `00422_hash_function_constexpr.sql`. To choose the prefix, find the largest prefix already present in the directory, and increment it by one. In the meantime, some other tests might be added with the same numeric prefix, but this is OK and does not lead to any problems, you don't have to change it later.

Some tests are marked with `zookeeper`, `shard` or `long` in their names. `zookeeper` is for tests that are using ZooKeeper. `shard` is for tests that require the server to listen `127.0.0.*`; `distributed` or `global` have the same meaning. `long` is for tests that run slightly longer than one second. You can disable these groups of tests using `--no-zookeeper`, `--no-shard` and `--no-long` options, respectively. Make sure to add a proper prefix to your test name if it needs ZooKeeper or distributed queries.

## Checking for an Error that Must Occur

Sometimes you want to test that a server error occurs for an incorrect query. We support special annotations for this in SQL tests, in the following form:

```
select x; -- { serverError 49 }
```

This test ensures that the server returns an error with code 49 about unknown column `x`. If there is no error, or the error is different, the test will fail. If you want to ensure that an error occurs on the client side, use `clientError` annotation instead.

Do not check for a particular wording of error message, it may change in the future, and the test will needlessly break. Check only the error code. If the existing error code is not precise enough for your needs, consider adding a new one.

## Testing a Distributed Query

If you want to use distributed queries in functional tests, you can leverage `remote table` function with `127.0.0.{1..2}` addresses for the server to query itself; or you can use predefined test clusters in server configuration file like `test_shard_localhost`. Remember to add the words `shard` or `distributed` to the test name, so that it is run in CI in correct configurations, where the server is configured to support distributed queries.

## Known Bugs

If we know some bugs that can be easily reproduced by functional tests, we place prepared functional tests in `tests/queries/bugs` directory. These tests will be moved to `tests/queries/0_stateless` when bugs are fixed.

## Integration Tests

Integration tests allow testing ClickHouse in clustered configuration and ClickHouse interaction with other servers like MySQL, Postgres, MongoDB. They are useful to emulate network splits, packet drops, etc. These tests are run under Docker and create multiple containers with various software.

See `tests/integration/README.md` on how to run these tests.

Note that integration of ClickHouse with third-party drivers is not tested. Also, we currently do not have integration tests with our JDBC and ODBC drivers.

## Unit Tests

Unit tests are useful when you want to test not the ClickHouse as a whole, but a single isolated library or class. You can enable or disable build of tests with `ENABLE_TESTS` CMake option. Unit tests (and other test programs) are located in `tests` subdirectories across the code. To run unit tests, type `ninja test`. Some tests use `gtest`, but some are just programs that return non-zero exit code on test failure.

It's not necessary to have unit tests if the code is already covered by functional tests (and functional tests are usually much more simple to use).

You can run individual `gtest` checks by calling the executable directly, for example:

```
$ ./src/unit_tests_dbms --gtest_filter=LocalAddress*
```

## Performance Tests

Performance tests allow to measure and compare performance of some isolated part of ClickHouse on synthetic queries. Tests are located at `tests/performance`. Each test is represented by `.xml` file with description of test case. Tests are run with `docker/tests/performance-comparison` tool . See the `readme` file for invocation.

Each test run one or multiple queries (possibly with combinations of parameters) in a loop. Some tests can contain preconditions on preloaded test dataset.

If you want to improve performance of ClickHouse in some scenario, and if improvements can be observed on simple queries, it is highly recommended to write a performance test. It always makes sense to use `perf top` or other perf tools during your tests.

## Test Tools and Scripts

Some programs in `tests` directory are not prepared tests, but are test tools. For example, for `Lexer` there is a tool `src/Parsers/tests/lexer` that just do tokenization of `stdin` and writes colorized result to `stdout`. You can use these kind of tools as a code examples and for exploration and manual testing.

## Miscellaneous Tests

There are tests for machine learned models in `tests/external_models`. These tests are not updated and must be transferred to integration tests.

There is separate test for quorum inserts. This test runs ClickHouse cluster on separate servers and emulate various failure cases: network split, packet drop (between ClickHouse nodes, between ClickHouse and ZooKeeper, between ClickHouse server and client, etc.), `kill -9`, `kill -STOP` and `kill -CONT` , like `Jepsen`. Then the test checks that all acknowledged inserts were written and all rejected inserts were not.

Quorum test was written by separate team before ClickHouse was open-sourced. This team no longer work with ClickHouse. Test was accidentally written in Java. For these reasons, quorum test must be rewritten and moved to integration tests.

## Manual Testing

When you develop a new feature, it is reasonable to also test it manually. You can do it with the following steps:

Build ClickHouse. Run ClickHouse from the terminal: change directory to `programs/clickhouse-server` and run it with `./clickhouse-server`. It will use configuration (`config.xml`, `users.xml` and files within `config.d` and `users.d` directories) from the current directory by default. To connect to ClickHouse server, run `programs/clickhouse-client/clickhouse-client`.

Note that all clickhouse tools (server, client, etc) are just symlinks to a single binary named `clickhouse`. You can find this binary at `programs clickhouse`. All tools can also be invoked as `clickhouse tool` instead of `clickhouse-tool`.

Alternatively you can install ClickHouse package: either stable release from Yandex repository or you can build package for yourself with `./release` in ClickHouse sources root. Then start the server with `sudo service clickhouse-server start` (or stop to stop the server). Look for logs at `/etc/clickhouse-server/clickhouse-server.log`.

When ClickHouse is already installed on your system, you can build a new `clickhouse` binary and replace the existing binary:

```
$ sudo service clickhouse-server stop  
$ sudo cp ./clickhouse /usr/bin/  
$ sudo service clickhouse-server start
```

Also you can stop system `clickhouse-server` and run your own with the same configuration but with logging to terminal:

```
$ sudo service clickhouse-server stop  
$ sudo -u clickhouse /usr/bin/clickhouse server --config-file /etc/clickhouse-server/config.xml
```

Example with `gdb`:

```
$ sudo -u clickhouse gdb --args /usr/bin/clickhouse server --config-file /etc/clickhouse-server/config.xml
```

If the system `clickhouse-server` is already running and you do not want to stop it, you can change port numbers in your `config.xml` (or override them in a file in `config.d` directory), provide appropriate data path, and run it.

`clickhouse` binary has almost no dependencies and works across wide range of Linux distributions. To quick and dirty test your changes on a server, you can simply `scp` your fresh built `clickhouse` binary to your server and then run it as in examples above.

## Testing Environment

Before publishing release as stable we deploy it on testing environment. Testing environment is a cluster that process 1/39 part of [Yandex.Metrica](#) data. We share our testing environment with Yandex.Metrica team. ClickHouse is upgraded without downtime on top of existing data. We look at first that data is processed successfully without lagging from realtime, the replication continue to work and there is no issues visible to Yandex.Metrica team. First check can be done in the following way:

```
SELECT hostName() AS h, any(version()), any(uptime()), max(UTCEventTime), count() FROM remote('example01-01-{1..3}t', merge, hits) WHERE EventDate >= today() - 2 GROUP BY h ORDER BY h;
```

In some cases we also deploy to testing environment of our friend teams in Yandex: Market, Cloud, etc. Also we have some hardware servers that are used for development purposes.

## Load Testing

After deploying to testing environment we run load testing with queries from production cluster. This is done manually.

Make sure you have enabled `query_log` on your production cluster.

Collect query log for a day or more:

```
$ clickhouse-client --query="SELECT DISTINCT query FROM system.query_log WHERE event_date = today() AND query LIKE '%ym:%' AND query NOT LIKE '%system.query_log%' AND type = 2 AND is_initial_query" > queries.tsv
```

This is a way complicated example. `type = 2` will filter queries that are executed successfully. `query LIKE '%ym:%'` is to select relevant queries from Yandex.Metrica. `is_initial_query` is to select only queries that are initiated by client, not by ClickHouse itself (as parts of distributed query processing).

scp this log to your testing cluster and run it as following:

```
$ clickhouse benchmark --concurrency 16 < queries.tsv
```

(probably you also want to specify a `--user`)

Then leave it for a night or weekend and go take a rest.

You should check that `clickhouse-server` does not crash, memory footprint is bounded and performance not degrading over time.

Precise query execution timings are not recorded and not compared due to high variability of queries and environment.

## Build Tests

Build tests allow to check that build is not broken on various alternative configurations and on some foreign systems. These tests are automated as well.

Examples:

- cross-compile for Darwin x86\_64 (Mac OS X)
- cross-compile for FreeBSD x86\_64
- cross-compile for Linux AArch64
- build on Ubuntu with libraries from system packages (discouraged)
- build with shared linking of libraries (discouraged)

For example, build with system packages is bad practice, because we cannot guarantee what exact version of packages a system will have. But this is really needed by Debian maintainers. For this reason we at least have to support this variant of build. Another example: shared linking is a common source of trouble, but it is needed for some enthusiasts.

Though we cannot run all tests on all variant of builds, we want to check at least that various build variants are not broken. For this purpose we use build tests.

We also test that there are no translation units that are too long to compile or require too much RAM.

We also test that there are no too large stack frames.

## Testing for Protocol Compatibility

When we extend ClickHouse network protocol, we test manually that old `clickhouse-client` works with new `clickhouse-server` and new `clickhouse-client` works with old `clickhouse-server` (simply by running binaries from corresponding packages).

We also test some cases automatically with integrational tests:

- if data written by old version of ClickHouse can be successfully read by the new version;
- do distributed queries work in a cluster with different ClickHouse versions.

## Help from the Compiler

Main ClickHouse code (that is located in `dbms` directory) is built with `-Wall -Wextra -Werror` and with some additional enabled warnings. Although these options are not enabled for third-party libraries.

Clang has even more useful warnings - you can look for them with `-Weverything` and pick something to default build.

For production builds, clang is used, but we also test make gcc builds. For development, clang is usually more convenient to use. You can build on your own machine with debug mode (to save battery of your laptop), but please note that compiler is able to generate more warnings with `-O3` due to better control flow and inter-procedure analysis. When building with clang in debug mode, debug version of `libc++` is used that allows to catch more errors at runtime.

## Sanitizers

### Address sanitizer

We run functional, integration, stress and unit tests under ASan on per-commit basis.

### Thread sanitizer

We run functional, integration, stress and unit tests under TSan on per-commit basis.

### Memory sanitizer

We run functional, integration, stress and unit tests under MSan on per-commit basis.

### Undefined behaviour sanitizer

We run functional, integration, stress and unit tests under UBSan on per-commit basis. The code of some third-party libraries is not sanitized for UB.

### Valgrind (Memcheck)

We used to run functional tests under Valgrind overnight, but don't do it anymore. It takes multiple hours. Currently there is one known false positive in `re2` library, see [this article](#).

## Fuzzing

ClickHouse fuzzing is implemented both using `libFuzzer` and random SQL queries.  
All the fuzz testing should be performed with sanitizers (Address and Undefined).

LibFuzzer is used for isolated fuzz testing of library code. Fuzzers are implemented as part of test code and have “\_fuzzer” name postfixes.

Fuzzer example can be found at `src/Parsers/tests/lexer_fuzzer.cpp`. LibFuzzer-specific configs, dictionaries and corpus are stored at `tests/fuzz`.

We encourage you to write fuzz tests for every functionality that handles user input.

Fuzzers are not built by default. To build fuzzers both `-DENABLE_FUZZING=1` and `-DENABLE_TESTS=1` options should be set.

We recommend to disable Jemalloc while building fuzzers. Configuration used to integrate ClickHouse fuzzing to

Google OSS-Fuzz can be found at `docker/fuzz`.

We also use simple fuzz test to generate random SQL queries and to check that the server does not die executing them.

You can find it in `00746_sql_fuzzy.pl`. This test should be run continuously (overnight and longer).

We also use sophisticated AST-based query fuzzer that is able to find huge amount of corner cases. It does random permutations and substitutions in queries AST. It remembers AST nodes from previous tests to use them for fuzzing of subsequent tests while processing them in random order. You can learn more about this fuzzer in [this blog article](#).

## Stress test

Stress tests are another case of fuzzing. It runs all functional tests in parallel in random order with a single server. Results of the tests are not checked.

It is checked that:

- server does not crash, no debug or sanitizer traps are triggered;
- there are no deadlocks;
- the database structure is consistent;
- server can successfully stop after the test and start again without exceptions.

There are five variants (Debug, ASan, TSan, MSan, UBSan).

## Thread Fuzzer

Thread Fuzzer (please don't mix up with Thread Sanitizer) is another kind of fuzzing that allows to randomize thread order of execution. It helps to find even more special cases.

## Security Audit

People from Yandex Security Team do some basic overview of ClickHouse capabilities from the security standpoint.

## Static Analyzers

We run clang-tidy and PVS-Studio on per-commit basis. clang-static-analyzer checks are also enabled. clang-tidy is also used for some style checks.

We have evaluated clang-tidy, Coverity, cppcheck, PVS-Studio, tscancode, CodeQL. You will find instructions for usage in tests/instructions/ directory. Also you can read [the article in russian](#).

If you use CLion as an IDE, you can leverage some clang-tidy checks out of the box.

We also use shellcheck for static analysis of shell scripts.

## Hardening

In debug build we are using custom allocator that does ASLR of user-level allocations.

We also manually protect memory regions that are expected to be readonly after allocation.

In debug build we also involve a customization of libc that ensures that no "harmful" (obsolete, insecure, not thread-safe) functions are called.

Debug assertions are used extensively.

In debug build, if exception with "logical error" code (implies a bug) is being thrown, the program is terminated prematurely. It allows to use exceptions in release build but make it an assertion in debug build.

Debug version of jemalloc is used for debug builds.

Debug version of libc++ is used for debug builds.

## Runtime Integrity Checks

Data stored on disk is checksummed. Data in MergeTree tables is checksummed in three ways simultaneously\* (compressed data blocks, uncompressed data blocks, the total checksum across blocks). Data transferred over network between client and server or between servers is also checksummed. Replication ensures bit-identical data on replicas.

It is required to protect from faulty hardware (bit rot on storage media, bit flips in RAM on server, bit flips in RAM of network controller, bit flips in RAM of network switch, bit flips in RAM of client, bit flips on the wire). Note that bit flips are common and likely to occur even for ECC RAM and in presence of TCP checksums (if you manage to run thousands of servers processing petabytes of data each day). [See the video \(russian\)](#).

ClickHouse provides diagnostics that will help ops engineers to find faulty hardware.

\* and it is not slow.

## Code Style

Code style rules are described [here](#).

To check for some common style violations, you can use `utils/check-style` script.

To force proper style of your code, you can use `clang-format`. File `.clang-format` is located at the sources root. It mostly corresponds with our actual code style. But it's not recommended to apply `clang-format` to existing files because it makes formatting worse. You can use `clang-format-diff` tool that you can find in clang source repository.

Alternatively you can try `unrustify` tool to reformat your code. Configuration is in `unrustify.cfg` in the sources root. It is less tested than `clang-format`.

`CLion` has its own code formatter that has to be tuned for our code style.

We also use `codespell` to find typos in code. It is automated as well.

## Metrica B2B Tests

Each ClickHouse release is tested with Yandex Metrica and AppMetrica engines. Testing and stable versions of ClickHouse are deployed on VMs and run with a small copy of Metrica engine that is processing fixed sample of input data. Then results of two instances of Metrica engine are compared together.

These tests are automated by separate team. Due to high number of moving parts, tests fail most of the time by completely unrelated reasons, that are very difficult to figure out. Most likely these tests have negative value for us. Nevertheless these tests were proved to be useful in about one or two times out of hundreds.

## Test Coverage

We also track test coverage but only for functional tests and only for `clickhouse-server`. It is performed on daily basis.

## Tests for Tests

There is automated check for flaky tests. It runs all new tests 100 times (for functional tests) or 10 times (for integration tests). If at least one time the test failed, it is considered flaky.

## Testflows

[Testflows](#) is an enterprise-grade testing framework. It is used by Altinity for some of the tests and we run these tests in our CI.

# Yandex Checks (only for Yandex employees)

These checks are importing ClickHouse code into Yandex internal monorepository, so ClickHouse codebase can be used as a library by other products at Yandex (YT and YDB). Note that clickhouse-server itself is not being build from internal repo and unmodified open-source build is used for Yandex applications.

## Test Automation

We run tests with Yandex internal CI and job automation system named “Sandbox”.

Build jobs and tests are run in Sandbox on per commit basis. Resulting packages and test results are published in GitHub and can be downloaded by direct links. Artifacts are stored for several months. When you send a pull request on GitHub, we tag it as “can be tested” and our CI system will build ClickHouse packages (release, debug, with address sanitizer, etc) for you.

We do not use Travis CI due to the limit on time and computational power.

We do not use Jenkins. It was used before and now we are happy we are not using Jenkins.

## サードパーティ製ライブラリを使用

ライブラリ	ライセンス
base64	BSD2条項ライセンス
ブースト	Boost Software License1.0
プロトコル	MIT
capnproto	MIT
cctz	Apacheライセンス2.0
二重変換	BSD3条項ライセンス
FastMemcpy	MIT
googletest	BSD3条項ライセンス
h3	Apacheライセンス2.0
hyperscan	BSD3条項ライセンス
libcxxabi	BSD + MIT
libdivide	Zlibライセンス
libgsasl	LGPL v2.1
libhdfs3	Apacheライセンス2.0
libmetrohash	Apacheライセンス2.0
libpcg-ランダム	Apacheライセンス2.0

ライブラリ	ライセンス
libressl	OpenSSLライセンス
リブ ド カフ カ	BSD2条項ライセンス
libwidechar_width	CC0 1.0ユニバーサル
llvm	BSD3条項ライセンス
lz4	BSD2条項ライセンス
mariadb-コネクタ-c	LGPL v2.1
murmurhash	パブリック
pdqsort	Zlibライセンス
ポコ	Boost Software License-バージョン1.0
プロトプロト	BSD3条項ライセンス
re2	BSD3条項ライセンス
UnixODBC	LGPL v2.1
zlib-ng	Zlibライセンス
zstd	BSD3条項ライセンス

## CMake in ClickHouse

TL; DR How to make ClickHouse compile and link faster?

Minimal ClickHouse build example:

```
cmake .. \
-DCMAKE_C_COMPILER=$(which clang-11) \
-DCMAKE_CXX_COMPILER=$(which clang++-11) \
-DCMAKE_BUILD_TYPE=Debug \
-DENABLE_CLICKHOUSE_ALL=OFF \
-ENABLE_CLICKHOUSE_SERVER=ON \
-ENABLE_CLICKHOUSE_CLIENT=ON \
-ENABLE_LIBRARIES=OFF \
-DUSE_UNWIND=ON \
-ENABLE_UTILS=OFF \
-ENABLE_TESTS=OFF
```

## CMake files types

1. ClickHouse's source CMake files (located in the root directory and in /src).
2. Arch-dependent CMake files (located in /cmake/\*os\_name\*).
3. Libraries finders (search for contrib libraries, located in /cmake/find).
4. Contrib build CMake files (used instead of libraries' own CMake files, located in /cmake/modules)

# List of CMake flags

- This list is auto-generated by [this Python script](#).
- The flag name is a link to its position in the code.
- If an option's default value is itself an option, it's also a link to its position in this list.

## ClickHouse modes

Name	Default value	Description	Comment
<a href="#">ENABLE_CLICKHOUSE_ALL</a>	ON	Enable all ClickHouse modes by default	The clickhouse tool multiple executors may be built at once to know what mode to use for SERVER and CLIENT.
<a href="#">ENABLE_CLICKHOUSE_BENCHMARK</a>	<a href="#">ENABLE_CLICKHOUSE_ALL</a>	Queries benchmarking mode	<a href="https://clickhouse.com/docs/en/interfaces/benchmark/">https://clickhouse.com/docs/en/interfaces/benchmark/</a>
<a href="#">ENABLE_CLICKHOUSE_CLIENT</a>	<a href="#">ENABLE_CLICKHOUSE_ALL</a>	Client mode (interactive tui/shell that connects to the server)	
<a href="#">ENABLE_CLICKHOUSE_COMPRESSOR</a>	<a href="#">ENABLE_CLICKHOUSE_ALL</a>	Data compressor and decompressor	<a href="https://clickhouse.com/docs/en/interfaces/compressor/">https://clickhouse.com/docs/en/interfaces/compressor/</a>
<a href="#">ENABLE_CLICKHOUSE_COPIER</a>	<a href="#">ENABLE_CLICKHOUSE_ALL</a>	Inter-cluster data copying mode	<a href="https://clickhouse.com/docs/en/interfaces/copier/">https://clickhouse.com/docs/en/interfaces/copier/</a>
<a href="#">ENABLE_CLICKHOUSE_EXTRACT_FROM_CONFIG</a>	<a href="#">ENABLE_CLICKHOUSE_ALL</a>	Configs processor (extract values etc.)	
<a href="#">ENABLE_CLICKHOUSE_FORMAT</a>	<a href="#">ENABLE_CLICKHOUSE_ALL</a>	Queries pretty-printer and formatter with syntax highlighting	
<a href="#">ENABLE_CLICKHOUSE_GIT_IMPORT</a>	<a href="#">ENABLE_CLICKHOUSE_ALL</a>	A tool to analyze Git repositories	<a href="https://presentations.clickhouse.com/doc/2020-07-01-clickhouse-github-import.html">https://presentations.clickhouse.com/doc/2020-07-01-clickhouse-github-import.html</a>

Name	Default value	Description	Comment
ENABLE_CLICKHOUSE_INSTALL	OFF	Install ClickHouse without .deb/.rpm/.tgz packages (having the binary only)	
ENABLE_CLICKHOUSE_KEEPER	ENABLE_CLICKHOUSE_ALL	ClickHouse alternative to ZooKeeper	
ENABLE_CLICKHOUSE_KEEPER_CONVERTER	ENABLE_CLICKHOUSE_ALL	Util allows to convert ZooKeeper logs and snapshots into clickhouse-keeper snapshot	
ENABLE_CLICKHOUSE_LIBRARY_BRIDGE	ENABLE_CLICKHOUSE_ALL	HTTP-server working like a proxy to Library dictionary source	
ENABLE_CLICKHOUSE_LOCAL	ENABLE_CLICKHOUSE_ALL	Local files fast processing mode	<a href="https://clickhouse.local/">https://clickhouse.local/</a>
ENABLE_CLICKHOUSE_OBFUSCATOR	ENABLE_CLICKHOUSE_ALL	Table data obfuscator (convert real data to benchmark-ready one)	<a href="https://clickhouse.obfuscator/">https://clickhouse.obfuscator/</a>
ENABLE_CLICKHOUSE_ODBC_BRIDGE	ENABLE_CLICKHOUSE_ALL	HTTP-server working like a proxy to ODBC driver	
ENABLE_CLICKHOUSE_SERVER	ENABLE_CLICKHOUSE_ALL	Server mode (main mode)	

Name	Default value	Description	Comment
ENABLE_CLICKHOUSE_STATIC_FILES_DISK_UPLOADER	ENABLE_CLICKHOUSE_ALL	A tool to export table data files to be later put to a static files web server	

## External libraries

Note that ClickHouse uses forks of these libraries, see <https://github.com/ClickHouse-Extras>.

Name	Default value	Description	Comment
ENABLE_AMQPCPP	ENABLE_LIBRARIES	Enable AMQP-CPP	
ENABLE_AVRO	ENABLE_LIBRARIES	Enable Avro	Needed when serialization
ENABLE_AVX	0	Use AVX instructions on x86_64	
ENABLE_AVX2	0	Use AVX2 instructions on x86_64	
ENABLE_AVX2_FOR_SPEC_OP	0	Use avx2 instructions for specific operations on x86_64	
ENABLE_AVX512	0	Use AVX512 instructions on x86_64	
ENABLE_AVX512_FOR_SPEC_OP	0	Use avx512 instructions for specific operations on x86_64	
ENABLE_BASE64	ENABLE_LIBRARIES	Enable base64	
ENABLE_BMI	0	Use BMI instructions on x86_64	
ENABLE_BROTLI	ENABLE_LIBRARIES	Enable brotli	
ENABLE_BZIP2	ENABLE_LIBRARIES	Enable bzip2 compression support	
ENABLE_CAPNP	ENABLE_LIBRARIES	Enable Cap'n Proto	

Name	Default value	Description	Comment
ENABLE_CASSANDRA	ENABLE_LIBRARIES	Enable Cassandra	
ENABLE_CCACHE	ENABLE_CCACHE_BY_DEFAULT	Speedup re-compilations using ccache (external tool)	<a href="https://ccache.suckless.org">https://ccache.suckless.org</a>
ENABLE_CLANG_TIDY	OFF	Use clang-tidy static analyzer	<a href="https://clang-tidy.readthedocs.io/">https://clang-tidy/</a>
ENABLE_CURL	ENABLE_LIBRARIES	Enable curl	
ENABLE_DATASKETCHES	ENABLE_LIBRARIES	Enable DataSketches	
ENABLE_EMBEDDED_COMPILER	ENABLE_EMBEDDED_COMPILER_DEFAULT	Enable support for 'compile_expressions' option for query execution	
ENABLE_FASTOPS	ENABLE_LIBRARIES	Enable fast vectorized mathematical functions library by Mikhail Parakhin	
ENABLE_FILELOG	ON	Enable FILELOG	
ENABLE_GPERF	ENABLE_LIBRARIES	Use gperf function hash generator tool	
ENABLE_GRPC	ENABLE_GRPC_DEFAULT	Use gRPC	
ENABLE_GSASL_LIBRARY	ENABLE_LIBRARIES	Enable gsasl library	
ENABLE_H3	ENABLE_LIBRARIES	Enable H3	
ENABLE_HDFS	ENABLE_LIBRARIES	Enable HDFS	
ENABLE_ICU	ENABLE_LIBRARIES	Enable ICU	
ENABLE_LDAP	ENABLE_LIBRARIES	Enable LDAP	
ENABLE_LIBPQXX	ENABLE_LIBRARIES	Enable libpqxx	
ENABLE_MSGPACK	ENABLE_LIBRARIES	Enable msgpack library	
ENABLE_MYSQL	ENABLE_LIBRARIES	Enable MySQL	

Name	Default value	Description	Comment
ENABLE_NLP	ENABLE_LIBRARIES	Enable NLP functions support	
ENABLE_NURAIT	ENABLE_LIBRARIES	Enable NuRaft	
ENABLE_ODBC	ENABLE_LIBRARIES	Enable ODBC library	
ENABLE_ORC	ENABLE_LIBRARIES	Enable ORC	
ENABLE_PARQUET	ENABLE_LIBRARIES	Enable parquet	
ENABLE_PCLMULQDQ	1	Use pclmulqdq instructions on x86_64	
ENABLE_POPCNT	1	Use popcnt instructions on x86_64	
ENABLE_PROTOBUF	ENABLE_LIBRARIES	Enable protobuf	
ENABLE_RAPIDJSON	ENABLE_LIBRARIES	Use rapidjson	
ENABLE_RDKAFKA	ENABLE_LIBRARIES	Enable kafka	
ENABLE_ROCKSDB	ENABLE_LIBRARIES	Enable ROCKSDB	
ENABLE_S2_GEOMETRY	ENABLE_LIBRARIES	Enable S2 geometry library	
ENABLE_S3	ENABLE_LIBRARIES	Enable S3	
ENABLE_SQLITE	ENABLE_LIBRARIES	Enable sqlite	
ENABLE_SSE41	1	Use SSE4.1 instructions on x86_64	
ENABLE_SSE42	1	Use SSE4.2 instructions on x86_64	
ENABLE_SSL	ENABLE_LIBRARIES	Enable ssl	Needed when connecting to e.g. clickhouse secure
ENABLE_SSSE3	1	Use SSSE3 instructions on x86_64	

Name	Default value	Description	Comment
ENABLE_STATS	ENABLE_LIBRARIES	Enable StatsLib library	

## External libraries system/bundled mode

Name	Default value	Description	Comment
USE_INTERNAL_AVRO_LIBRARY	ON	Set to FALSE to use system avro library instead of bundled	
USE_INTERNAL_AWS_S3_LIBRARY	ON	Set to FALSE to use system S3 instead of bundled (experimental set to OFF on your own risk)	
USE_INTERNAL_BROTLI_LIBRARY	USE_STATIC_LIBRARIES	Set to FALSE to use system libbrotli library instead of bundled	Many system brotly libraries bundled by
USE_INTERNAL_CAPNP_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system capnproto library instead of bundled	
USE_INTERNAL_CURL	NOT_UNBUNDLED	Use internal curl library	
USE_INTERNAL_DATASKETCHES_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system DataSketches library instead of bundled	

Name	Default value	Description	Comment
USE_INTERNAL_GRPC_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system gRPC library instead of bundled. (Experimental) Set to OFF on your own risk)	Normally w framework USE_INTERNAL_GRPC_LIBRARY set to OFF to force gRPC framework installed in The external lib be installed running su libgrpc++- grpc
USE_INTERNAL_GTEST_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system Google Test instead of bundled	
USE_INTERNAL_H3_LIBRARY	ON	Set to FALSE to use system h3 library instead of bundled	
USE_INTERNAL_HDFS3_LIBRARY	ON	Set to FALSE to use system HDFS3 instead of bundled (experimental - set to OFF on your own risk)	
USE_INTERNAL_ICU_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system ICU library instead of bundled	
USE_INTERNAL_LDAP_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system *LDAP library instead of bundled	
USE_INTERNAL_LIBCXX_LIBRARY	USE_INTERNAL_LIBCXX_LIBRARY_DEFAULT	Disable to use system libcxx and libcxxabi libraries instead of bundled	

Name	Default value	Description	Comment
USE_INTERNAL_LIBGSASL_LIBRARY	USE_STATIC_LIBRARIES	Set to FALSE to use system libgsasl library instead of bundled	when USE_ usually need dependency
USE_INTERNAL_LIBXML2_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system libxml2 library instead of bundled	
USE_INTERNAL_MSGPACK_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system msgpack library instead of bundled	
USE_INTERNAL_MYSQL_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system mysqlclient library instead of bundled	
USE_INTERNAL_ODBC_LIBRARY	NOT_UNBUNDLED	Use internal ODBC library	
USE_INTERNAL_ORC_LIBRARY	ON	Set to FALSE to use system ORC instead of bundled (experimental set to OFF on your own risk)	
USE_INTERNAL_PARQUET_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system parquet library instead of bundled	
USE_INTERNAL_POCO_LIBRARY	ON	Use internal Poco library	

Name	Default value	Description	Comment
USE_INTERNAL_PROTOBUF_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system protobuf instead of bundled. (Experimental. Set to OFF to force protobuf lib installed in The external installed in sudo apt-get install protobuf-compiler)	
USE_INTERNAL_RAPIDJSON_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system rapidjson library instead of bundled	
USE_INTERNAL_RDKAFKA_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system librdkafka instead of the bundled	
USE_INTERNAL_RE2_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system re2 library instead of bundled [slower]	
USE_INTERNAL_ROCKSDB_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system ROCKSDB library instead of bundled	
USE_INTERNAL_SNAPPY_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system snappy library instead of bundled	
USE_INTERNAL_SPARSEHASH_LIBRARY	ON	Set to FALSE to use system sparsehash library instead of bundled	

Name	Default value	Description	Comment
USE_INTERNAL_SSL_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system *ssl library instead of bundled	
USE_INTERNAL_XZ_LIBRARY	NOT_UNBUNDLED	Set to OFF to use system xz (lzma) library instead of bundled	
USE_INTERNAL_ZLIB_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system zlib library instead of bundled	
USE_INTERNAL_ZSTD_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system zstd library instead of bundled	

## Other flags

Name	Default value	Description	Comment
ADD_GDB_INDEX_FOR_GOLD	OFF	Add .gdb-index to resulting binaries for gold linker.	Ignored if <code>lld</code> is used

Name	Default value	Description	Comment
ARCH_NATIVE	0	Add -march=native compiler flag. This makes your binaries non-portable but more performant code may be generated. This option overrides ENABLE_* options for specific instruction set. Highly not recommended to use.	
CLICKHOUSE_SPLIT_BINARY	OFF	Make several binaries (clickhouse-server, clickhouse-client etc.) instead of one bundled	
COMPILER_PIPE	ON	-pipe compiler option	Less /tmp usage, mc
ENABLE_CHECK_HEAVY_BUILDS	OFF	Don't allow C++ translation units to compile too long or to take too much memory while compiling.	Take care to add pr thinks that prlimit is to work with multipl 31T18:06:32.65532 as=10000000000 --..... std=gnu++2a .src/CMakeFiles/dbm MF src/CMakeFiles/dbm -O src/CMakeFiles/dbm c ..../src/Storages/Mer 31T18:06:32.65670 ..../src/Storages/Merg to use --ccache-skip

Name	Default value	Description	Comment
ENABLE_EXAMPLES	OFF	Build all example programs in 'examples' subdirectories	
ENABLE_FUZZING	OFF	Fuzzy testing using libfuzzer	
ENABLE_LIBRARIES	ON	Enable all external libraries by default	Turns on all external libraries by default.
ENABLE_MULTITARGET_CODE	ON	Enable platform-dependent code	ClickHouse development macro (e.g. <code>ifdef ENABLE_LIBRARY</code> ) such macro. See <code>src/Makefile</code> .
ENABLE_TESTS	ON	Provide unit_test_dbms target with Google.Test unit tests	If turned ON, assumes bundled one.
ENABLE_THINLTO	ON	Clang-specific link time optimization	<a href="https://clang.llvm.org/docs/ThinLTO.html">https://clang.llvm.org/docs/ThinLTO.html</a> when building with Clang.
FAIL_ON_UNSUPPORTED_OPTIONS_COMBINATION	ON	Stop/Fail CMake configuration if some ENABLE_XXX option is defined (either ON or OFF) but is not possible to satisfy	If turned off: e.g. with <code>cmake -DENABLE_LIBRARY=OFF</code> the CMake will continue to build.
GLIBC_COMPATIBILITY	ON	Enable compatibility with older glibc libraries.	Only for Linux, x86_64.
LINKER_NAME	OFF	Linker name or full path	Example values: <code>lld-link</code> , <code>ld</code> .
MAKE_STATIC_LIBRARIES	USE_STATIC_LIBRARIES	Disable to make shared libraries	

Name	Default value	Description	Comment
PARALLEL_COMPILE_JOBS	""	Maximum number of concurrent compilation jobs	1 if not set
PARALLEL_LINK_JOBS	""	Maximum number of concurrent link jobs	1 if not set
SANITIZE	""	Enable one of the code sanitizers	Possible values: - <code>address</code> (ASan) - <code>undefined</code> (UBSan) - <code>leak</code> (Lsan)
SPLIT_SHARED_LIBRARIES	OFF	Keep all internal libraries as separate .so files	DEVELOPER ONLY. If enabled, ClickHouse will use shared libraries.
STRIP_DEBUG_SYMBOLS_FUNCTIONS	STRIP_DSF_DEFAULT	Do not generate debugger info for ClickHouse functions	Provides faster linking (but requires debug some source variables)."
UNBUNDLED	OFF	Use system libraries instead of ones in contrib/	We recommend avc can't guarantee all issues exists for enthusiasts. Whole idea of using is deeply flawed. Use contrib/
USE_INCLUDE_WHAT_YOU_USE	OFF	Automatically reduce unneeded includes in source code (external tool)	<a href="https://github.com/include-what-you-use/include-what-you-use">https://github.com/include-what-you-use/include-what-you-use</a>
USE_LIBCXX	NOT_UNBUNDLED	Use libc++ and libc++abi instead of libstdc++	
USE_LIBPROTOBUF_MUTATOR	ENABLE_FUZZING	Enable libprotobuf-mutator	
USE_SENTRY	ENABLE_LIBRARIES	Use Sentry	

Name	Default value	Description	Comment
USE_SIMDJSON	ENABLE_LIBRARIES	Use simdjson	
USE_SNAPPY	ENABLE_LIBRARIES	Enable snappy library	
USE_STATIC_LIBRARIES	ON	Disable to use shared libraries	
USE_UNWIND	ENABLE_LIBRARIES	Enable libunwind (better stacktraces)	
USE_YAML_CPP	ENABLE_LIBRARIES	Enable yaml-cpp	
WERROR	OFF	Enable -Werror compiler option	Using system libs can cause linking issues.
WEVERYTHING	ON	Enable -Weverything option with some exceptions.	Add some warnings. Intended for pedantic code. Intended to be found useful. API changes.
WITH_COVERAGE	OFF	Profile the resulting binary/binaries	Compiler-specific command line options.

## Developer's guide for adding new CMake options

Don't be obvious. Be informative.

Bad:

```
option(ENABLE_TESTS "Enables testing" OFF)
```

This description is quite useless as it neither gives the viewer any additional information nor explains the option purpose.

Better:

```
option(ENABLE_TESTS "Provide unit_test_dbms target with Google.test unit tests" OFF)
```

If the option's purpose can't be guessed by its name, or the purpose guess may be misleading, or option has some pre-conditions, leave a comment above the `option()` line and explain what it does. The best way would be linking the docs page (if it exists). The comment is parsed into a separate column (see below).

Even better:

```
## implies ${TESTS_ARE_ENABLED}
## see tests/CMakeLists.txt for implementation detail.
option(ENABLE_TESTS "Provide unit_test_dbms target with Google.test unit tests" OFF)
```

## If the option's state could produce unwanted (or unusual) result, explicitly warn the user.

Suppose you have an option that may strip debug symbols from the ClickHouse's part. This can speed up the linking process, but produces a binary that cannot be debugged. In that case, prefer explicitly raising a warning telling the developer that he may be doing something wrong.

Also, such options should be disabled if applies.

Bad:

```
option(STRIPE_DEBUG_SYMBOLS_FUNCTIONS
    "Do not generate debugger info for ClickHouse functions.
    ${STRIPE_DSF_DEFAULT}")

if (STRIPE_DEBUG_SYMBOLS_FUNCTIONS)
    target_compile_options(clickhouse_functions PRIVATE "-g0")
endif()
```

Better:

```
## Provides faster linking and lower binary size.
## Tradeoff is the inability to debug some source files with e.g. gdb
## (empty stack frames and no local variables)."
option(STRIPE_DEBUG_SYMBOLS_FUNCTIONS
    "Do not generate debugger info for ClickHouse functions."
    ${STRIPE_DSF_DEFAULT})

if (STRIPE_DEBUG_SYMBOLS_FUNCTIONS)
    message(WARNING "Not generating debugger info for ClickHouse functions")
    target_compile_options(clickhouse_functions PRIVATE "-g0")
endif()
```

## In the option's description, explain WHAT the option does rather than WHY it does something.

The WHY explanation should be placed in the comment.  
You may find that the option's name is self-descriptive.

Bad:

```
option(ENABLE_THINLTO "Enable Thin LTO. Only applicable for clang. It's also suppressed when building with tests or
sanitizers." ON)
```

Better:

```
## Only applicable for clang.
## Turned off when building with tests or sanitizers.
option(ENABLE_THINLTO "Clang-specific link time optimisation" ON).
```

Don't assume other developers know as much as you do.

In ClickHouse, there are many tools used that an ordinary developer may not know. If you are in doubt, give a link to the tool's docs. It won't take much of your time.

Bad:

```
option(ENABLE_THINLTO "Enable Thin LTO. Only applicable for clang. It's also suppressed when building with tests or sanitizers." ON)
```

Better (combined with the above hint):

```
## https://clang.llvm.org/docs/ThinLTO.html
## Only applicable for clang.
## Turned off when building with tests or sanitizers.
option(ENABLE_THINLTO "Clang-specific link time optimisation" ON).
```

Other example, bad:

```
option (USE_INCLUDE_WHAT_YOU_USE "Use 'include-what-you-use' tool" OFF)
```

Better:

```
## https://github.com/include-what-you-use/include-what-you-use
option (USE_INCLUDE_WHAT_YOU_USE "Reduce unneeded #include s (external tool)" OFF)
```

## Prefer consistent default values.

CMake allows you to pass a plethora of values representing boolean true/false, e.g. 1, ON, YES, .... Prefer the ON/OFF values, if possible.

# How to add test queries to ClickHouse CI

ClickHouse has hundreds (or even thousands) of features. Every commit gets checked by a complex set of tests containing many thousands of test cases.

The core functionality is very well tested, but some corner-cases and different combinations of features can be uncovered with ClickHouse CI.

Most of the bugs/regressions we see happen in that 'grey area' where test coverage is poor.

And we are very interested in covering most of the possible scenarios and feature combinations used in real life by tests.

## Why adding tests

Why/when you should add a test case into ClickHouse code:

- 1) you use some complicated scenarios / feature combinations / you have some corner case which is probably not widely used
- 2) you see that certain behavior gets changed between version w/o notifications in the changelog
- 3) you just want to help to improve ClickHouse quality and ensure the features you use will not be broken in the future releases
- 4) once the test is added/accepted, you can be sure the corner case you check will never be accidentally broken.

- 5) you will be a part of great open-source community
- 6) your name will be visible in the `system.contributors` table!
- 7) you will make a world bit better :)

## Steps to do

### Prerequisite

I assume you run some Linux machine (you can use docker / virtual machines on other OS) and any modern browser / internet connection, and you have some basic Linux & SQL skills.

Any highly specialized knowledge is not needed (so you don't need to know C++ or know something about how ClickHouse CI works).

### Preparation

1) [create GitHub account](#) (if you haven't one yet)

2) [setup git](#)

```
## for Ubuntu
sudo apt-get update
sudo apt-get install git

git config --global user.name "John Doe" # fill with your name
git config --global user.email "email@example.com" # fill with your email
```

3) [fork ClickHouse project](#) - just open <https://github.com/ClickHouse/ClickHouse> and press fork button in the top right corner:

 Watch ▾

35

 Star

45

 Fork

37

4) clone your fork to some folder on your PC, for example, `~/workspace/ClickHouse`

```
mkdir ~/workspace && cd ~/workspace
git clone https://github.com/< your GitHub username>/ClickHouse
cd ClickHouse
git remote add upstream https://github.com/ClickHouse/ClickHouse
```

### New branch for the test

1) create a new branch from the latest clickhouse master

```
cd ~/workspace/ClickHouse
git fetch upstream
git checkout -b name_for_a_branch_with_my_test upstream/master
```

### Install & run clickhouse

1) install `clickhouse-server` (follow [official docs](#))

2) install test configurations (it will use Zookeeper mock implementation and adjust some settings)

```
cd ~/workspace/ClickHouse/tests/config
sudo ./install.sh
```

3) run `clickhouse-server`

```
sudo systemctl restart clickhouse-server
```

## Creating the test file

1) find the number for your test - find the file with the biggest number in `tests/queries/0_stateless/`

```
$ cd ~/workspace/ClickHouse  
$ ls tests/queries/0_stateless/[0-9]*.reference | tail -n 1  
tests/queries/0_stateless/01520_client_print_query_id.reference
```

Currently, the last number for the test is `01520`, so my test will have the number `01521`

2) create an SQL file with the next number and name of the feature you test

```
touch tests/queries/0_stateless/01521_dummy_test.sql
```

3) edit SQL file with your favorite editor (see hint of creating tests below)

```
vim tests/queries/0_stateless/01521_dummy_test.sql
```

4) run the test, and put the result of that into the reference file:

```
clickhouse-client -nmT < tests/queries/0_stateless/01521_dummy_test.sql | tee  
tests/queries/0_stateless/01521_dummy_test.reference
```

5) ensure everything is correct, if the test output is incorrect (due to some bug for example), adjust the reference file using text editor.

## How to create a good test

- A test should be
  - minimal - create only tables related to tested functionality, remove unrelated columns and parts of query
  - fast - should not take longer than a few seconds (better subseconds)
  - correct - fails then feature is not working
    - deterministic
  - isolated / stateless
    - don't rely on some environment things
    - don't rely on timing when possible
- try to cover corner cases (zeros / Nulls / empty sets / throwing exceptions)
- to test that query return errors, you can put special comment after the query: `-- { serverError 60 }` or `-- { clientError 20 }`
- don't switch databases (unless necessary)
- you can create several table replicas on the same node if needed
- you can use one of the test cluster definitions when needed (see `system.clusters`)
- use `number` / `numbers_mt` / `zeros` / `zeros_mt` and similar for queries / to initialize data when applicable
- clean up the created objects after test and before the test (`DROP IF EXISTS`) - in case of some dirty state

- prefer sync mode of operations (mutations, merges, etc.)
- use other SQL files in the `0_stateless` folder as an example
- ensure the feature / feature combination you want to test is not yet covered with existing tests

## Test naming rules

It's important to name tests correctly, so one could turn some tests subset off in `clickhouse-test` invocation.

Tester flag  What should be in test name   When flag should be added
--- --- --- ---
--[no-]zookeeper  "zookeeper" or "replica"   Test uses tables from ReplicatedMergeTree family
--[no-]shard   "shard" or "distributed" or "global"   Test using connections to 127.0.0.2 or similar
--[no-]long   "long" or "deadlock" or "race"   Test runs longer than 60 seconds

Commit / push / create PR.

- 1) commit & push your changes

```
cd ~/workspace/ClickHouse
git add tests/queries/0_stateless/01521_dummy_test.sql
git add tests/queries/0_stateless/01521_dummy_test.reference
git commit # use some nice commit message when possible
git push origin HEAD
```

- 2) use a link which was shown during the push, to create a PR into the main repo

- 3) adjust the PR title and contents, in Changelog category (leave one) keep Build/Testing/Packaging Improvement, fill the rest of the fields if you want.

## ClickHouse Cloud Service

### Info

Detailed public description for ClickHouse cloud services is not ready yet, please [contact us](#) to learn more.

## ClickHouse Commercial Support Service

### Info

Detailed public description for ClickHouse support services is not ready yet, please [contact us](#) to learn more.

## ClickHouse Commercial Services

Service categories:

- [Cloud](#)
- [Support](#)

# ClickHouse release v21.10, 2021-10-16

## Backward Incompatible Change

- Now the following MergeTree table-level settings: `replicated_max_parallel_sends`, `replicated_max_parallel_sends_for_table`, `replicated_max_parallel_fetches`, `replicated_max_parallel_fetches_for_table` do nothing. They never worked well and were replaced with `max_replicated_fetches_network_bandwidth`, `max_replicated_sends_network_bandwidth` and `background_fetches_pool_size`. [#28404 \(alesapin\)](#).

## New Feature

- Add feature for creating user-defined functions (UDF) as lambda expressions. Syntax `CREATE FUNCTION {function_name} as ({parameters}) -> {function core}`. Example `CREATE FUNCTION plus_one as (a) -> a + 1`. Authors @Realist007. [#27796 \(Maksim Kita\)](#) [#23978 \(Realist007\)](#).
- Added Executable storage engine and `executable` table function. It enables data processing with external scripts in streaming fashion. [#28102 \(Maksim Kita\)](#) (`ruct`).
- Added `ExecutablePool` storage engine. Similar to `Executable` but it's using a pool of long running processes. [#28518 \(Maksim Kita\)](#).
- Add `ALTER TABLE ... MATERIALIZE COLUMN` query. [#27038 \(Vladimir Chebotarev\)](#).
- Support for partitioned write into `s3` table function. [#23051 \(Vladimir Chebotarev\)](#).
- Support `lz4` compression format (in addition to `gz`, `bz2`, `xz`, `zstd`) for data import / export. [#25310 \(Bharat Nallan\)](#).
- Allow positional arguments under setting `enable_positional_arguments`. Closes [#2592](#). [#27530 \(Ksenia Sumarokova\)](#).
- Accept user settings related to file formats in `SETTINGS` clause in `CREATE` query for `s3` tables. This closes [#27580](#). [#28037 \(Nikita Mikhaylov\)](#).
- Allow SSL connection for `RabbitMQ` engine. [#28365 \(Ksenia Sumarokova\)](#).
- Add `getServerPort` function to allow getting server port. When the port is not used by the server, throw an exception. [#27900 \(Amos Bird\)](#).
- Add conversion functions between "snowflake id" and `DateTime`, `DateTime64`. See [#27058](#). [#27704 \(jasine\)](#).
- Add function `SHA512`. [#27830 \(zhanglistar\)](#).
- Add `log_queries_probability` setting that allows user to write to `query_log` only a sample of queries. Closes [#16609](#). [#27527 \(Nikolay Degterinsky\)](#).

## Experimental Feature

- `web` type of disks to store readonly tables on web server in form of static files. See [#23982](#). [#25251 \(Ksenia Sumarokova\)](#). This is mostly needed to facilitate testing of operation on shared storage and for easy importing of datasets. Not recommended to use before release 21.11.
- Added new commands `BACKUP` and `RESTORE`. [#21945 \(Vitaly Baranov\)](#). This is under development and not intended to be used in current version.

## Performance Improvement

- Speed up `sumIf` and `countIf` aggregation functions. [#28272 \(Raúl Marín\)](#).

- Create virtual projection for `minmax` indices. Now, when `allow_experimental_projection_optimization` is enabled, queries will use minmax index instead of reading the data when possible. [#26286 \(Amos Bird\)](#).
- Introducing two checks in `sequenceMatch` and `sequenceCount` that allow for early exit when some deterministic part of the sequence pattern is missing from the events list. This change unlocks many queries that would previously fail due to reaching operations cap, and generally speeds up the pipeline. [#27729 \(Jakub Kuklis\)](#).
- Enhance primary key analysis with always monotonic information of binary functions, notably non-zero constant division. [#28302 \(Amos Bird\)](#).
- Make `hasAll` filter condition leverage bloom filter data-skipping indexes. [#27984 \(Braulio Valdivielso Martínez\)](#).
- Speed up data parts loading by delaying table startup process. [#28313 \(Amos Bird\)](#).
- Fixed possible excessive number of conditions moved from `WHERE` to `PREWHERE` (optimization controlled by settings `optimize_move_to_prewhere`). [#28139 \(lthaooo\)](#).
- Enable `optimize_distributed_group_by_sharding_key` by default. [#28105 \(Azat Khuzhin\)](#).

## Improvement

- Check cluster name before creating `Distributed` table, do not allow to create a table with incorrect cluster name. Fixes [#27832](#). [#27927 \(tavplubix\)](#).
- Add aggregate function `quantileBFloat16Weighted` similarly to other quantile...Weighted functions. This closes [#27745](#). [#27758 \(Ivan Novitskiy\)](#).
- Allow to create dictionaries with empty attributes list. [#27905 \(Maksim Kita\)](#).
- Add interactive documentation in `clickhouse-client` about how to reset the password. This is useful in scenario when user has installed ClickHouse, set up the password and instantly forget it. See [#27750](#). [#27903 \(alexey-milovidov\)](#).
- Support the case when the data is enclosed in array in `JSONAsString` input format. Closes [#25517](#). [#25633 \(Kruglov Pavel\)](#).
- Add new column `last_queue_update_exception` to `system.replicas` table. [#26843 \(nvartolomei\)](#).
- Support reconnections on failover for `MaterializedPostgreSQL` tables. Closes [#28529](#). [#28614 \(Kseniia Sumarokova\)](#).
- Generate a unique server UUID on first server start. [#20089 \(Bharat Nallan\)](#).
- Introduce `connection_wait_timeout` (default to 5 seconds, 0 - do not wait) setting for `MySQL` engine. [#28474 \(Azat Khuzhin\)](#).
- Do not allow creating `MaterializedPostgreSQL` with bad arguments. Closes [#28423](#). [#28430 \(Kseniia Sumarokova\)](#).
- Use real tmp file instead of predefined "rows\_sources" for vertical merges. This avoids generating garbage directories in tmp disks. [#28299 \(Amos Bird\)](#).
- Added `libhdfs3_conf` in server config instead of export env `LIBHDFS3_CONF` in `clickhouse-server.service`. This is for configuration of interaction with HDFS. [#28268 \(Zhichang Yu\)](#).
- Fix removing of parts in a Temporary state which can lead to an unexpected exception (Part %name% doesn't exist). Fixes [#23661](#). [#28221 #28221 \(Azat Khuzhin\)](#).

- Fix `zookeeper_log.address` (before the first patch in this PR the address was always `::`) and reduce number of calls `getpeername(2)` for this column (since each time entry for `zookeeper_log` is added `getpeername()` is called, cache this address in the zookeeper client to avoid this). #28212 (Azat Khuzhin).
- Support implicit conversions between index in operator `[]` and key of type `Map` (e.g. different `Int` types, `String` and `FixedString`). #28096 (Anton Popov).
- Support `ON CONFLICT` clause when inserting into PostgreSQL table engine or table function. Closes #27727. #28081 (Kseniia Sumarokova).
- Lower restrictions for `Enum` data type to allow attaching compatible data. Closes #26672, #28028 (Dmitry Novik).
- Add a setting `empty_result_for_aggregation_by_constant_keys_on_empty_set` to control the behavior of grouping by constant keys on empty set. This is to bring back the old behaviour of #6842. #27932 (Amos Bird).
- Added `replication_wait_for_inactive_replica_timeout` setting. It allows to specify how long to wait for inactive replicas to execute `ALTER/OPTIMZE/TRUNCATE` query (default is 120 seconds). If `replication_alter_partitions_sync` is 2 and some replicas are not active for more than `replication_wait_for_inactive_replica_timeout` seconds, then `UNFINISHED` will be thrown. #27931 (tavplubix).
- Support lambda argument for `APPLY` column transformer which allows applying functions with more than one argument. This is for #27877, #27901 (Amos Bird).
- Enable `tcp_keep_alive_timeout` by default. #27882 (Azat Khuzhin).
- Improve remote query cancelation (in case of remote server abnormaly terminated). #27881 (Azat Khuzhin).
- Use Multipart copy upload for large S3 objects. #27858 (ianton-ru).
- Allow symlink traversal for library dictionary path. #27815 (Kseniia Sumarokova).
- Now `ALTER MODIFY COLUMN T TO Nullable(T)` doesn't require mutation. #27787 (victorgao).
- Don't silently ignore errors and don't count delays in `ReadBufferFromS3`. #27484 (Vladimir Chebotarev).
- Improve `ALTER ... MATERIALIZE TTL` by recalculating metadata only without actual TTL action. #27019 (lthaooo).
- Allow reading the list of custom top level domains without a new line at EOF. #28213 (Azat Khuzhin).

## Bug Fix

- Fix cases, when reading compressed data from `carbon-clickhouse` fails with 'attempt to read after end of file'. Closes #26149, #28150 (FArthur-cmd).
- Fix checking access grants when executing `GRANT WITH REPLACE` statement with `ON CLUSTER` clause. This PR improves fix #27001. #27983 (Vitaly Baranov).
- Allow selecting with `extremes = 1` from a column of the type `LowCardinality(UUID)`. #27918 (Vitaly Baranov).
- Fix PostgreSQL-style cast (`::` operator) with negative numbers. #27876 (Anton Popov).
- After #26864. Fix shutdown of `NamedSessionStorage`: session contexts stored in `NamedSessionStorage` are now destroyed before destroying the global context. #27875 (Vitaly Baranov).
- Bugfix for `windowFunnel` "strict" mode. This fixes #27469, #27563 (achimbab).
- Fix infinite loop while reading truncated `bzip2` archive. #28543 (Azat Khuzhin).

- Fix UUID overlap in `DROP TABLE` for internal DDL from `MaterializedMySQL`. `MaterializedMySQL` is an experimental feature. [#28533 \(Azat Khuzhin\)](#).
- Fix `There is no subcolumn` error, while select from tables, which have `Nested` columns and scalar columns with dot in name and the same prefix as `Nested` (e.g. `n.id UInt32, n.arr1 Array(UInt64), n.arr2 Array(UInt64)`). [#28531 \(Anton Popov\)](#).
- Fix bug which can lead to error `Existing table metadata in ZooKeeper differs in sorting key expression`.after `ALTER` of `ReplicatedVersionedCollapsingMergeTree`. Fixes [#28515](#). [#28528 \(alesapin\)](#).
- Fixed possible ZooKeeper watches leak (minor issue) on background processing of distributed DDL queue. Closes [#26036](#). [#28446 \(tavplubix\)](#).
- Fix missing quoting of table names in `MaterializedPostgreSQL` engine. Closes [#28316](#). [#28433 \(Ksenia Sumarokova\)](#).
- Fix the wrong behaviour of non joined rows from nullable column. Close [#27691](#). [#28349 \(vdimir\)](#).
- Fix NOT-IN index optimization when not all key columns are used. This fixes [#28120](#). [#28315 \(Amos Bird\)](#).
- Fix intersecting parts due to new part had been replaced with an empty part. [#28310 \(Azat Khuzhin\)](#).
- Fix inconsistent result in queries with `ORDER BY` and `Merge` tables with enabled setting `optimize_read_in_order`. [#28266 \(Anton Popov\)](#).
- Fix possible read of uninitialized memory for queries with `Nullable(LowCardinality)` type and the setting `extremes` set to 1. Fixes [#28165](#). [#28205 \(Nikolai Kochetov\)](#).
- Multiple small fixes for projections. See detailed description in the PR. [#28178 \(Amos Bird\)](#).
- Fix extremely rare segfaults on shutdown due to incorrect order of context/config reloader shutdown. [#28088 \(nvartolomei\)](#).
- Fix handling null value with type of `Nullable(String)` in function `JSONExtract`. This fixes [#27929](#) and [#27930](#). This was introduced in <https://github.com/ClickHouse/ClickHouse/pull/25452>. [#27939 \(Amos Bird\)](#).
- Multiple fixes for the new `clickhouse-keeper` tool. Fix a rare bug in `clickhouse-keeper` when the client can receive a watch response before request-response. [#28197 \(alesapin\)](#). Fix incorrect behavior in `clickhouse-keeper` when list watches (`getChildren`) triggered with `set` requests for children. [#28190 \(alesapin\)](#). Fix rare case when changes of `clickhouse-keeper` settings may lead to lost logs and server hung. [#28360 \(alesapin\)](#). Fix bug in `clickhouse-keeper` which can lead to endless logs when `rotate_logs_interval` decreased. [#28152 \(alesapin\)](#).

## Build/Testing/Packaging Improvement

- Enable Thread Fuzzer in Stress Test. Thread Fuzzer is ClickHouse feature that allows to test more permutations of thread scheduling and discover more potential issues. This closes [#9813](#). This closes [#9814](#). This closes [#9515](#). This closes [#9516](#). [#27538 \(alexey-milovidov\)](#).
- Add new log level `test` for testing environments. It is even more verbose than the default `trace`. [#28559 \(alesapin\)](#).
- Print out git status information at CMake configure stage. [#28047 \(Braulio Valdivielso Martínez\)](#).
- Temporarily switched ubuntu apt repository to mirror ru.archive.ubuntu.com as the default one (archive.ubuntu.com) is not responding from our CI. [#28016 \(Ilya Yatsishin\)](#).

ClickHouse release v21.9, 2021-09-09

## Backward Incompatible Change

- Do not output trailing zeros in text representation of `Decimal` types. Example: `1.23` will be printed instead of `1.230000` for decimal with scale 6. This closes [#15794](#). It may introduce slight incompatibility if your applications somehow relied on the trailing zeros. Serialization in output formats can be controlled with the setting `output_format_decimal_trailing_zeros`. Implementation of `toString` and casting to `String` is changed unconditionally. [#27680](#) ([alexey-milovidov](#)).
- Do not allow to apply parametric aggregate function with `-Merge` combinator to aggregate function state if state was produced by aggregate function with different parameters. For example, state of `fooState(42)(x)` cannot be finalized with `fooMerge(s)` or `fooMerge(123)(s)`, parameters must be specified explicitly like `fooMerge(42)(s)` and must be equal. It does not affect some special aggregate functions like `quantile` and `sequence*` that use parameters for finalization only. [#26847](#) ([tavplubix](#)).
- Under `clickhouse-local`, always treat local addresses with a port as remote. [#26736](#) ([Raúl Marín](#)).
- Fix the issue that in case of some sophisticated query with column aliases identical to the names of expressions, bad cast may happen. This fixes [#25447](#). This fixes [#26914](#). This fix may introduce backward incompatibility: if there are different expressions with identical names, exception will be thrown. It may break some rare cases when `enable_optimize_predicate_expression` is set. [#26639](#) ([alexey-milovidov](#)).
- Now, scalar subquery always returns `Nullable` result if it's type can be `Nullable`. It is needed because in case of empty subquery it's result should be `Null`. Previously, it was possible to get error about incompatible types (type deduction does not execute scalar subquery, and it could use not-nullable type). Scalar subquery with empty result which can't be converted to `Nullable` (like `Array` or `Tuple`) now throws error. Fixes [#25411](#). [#26423](#) ([Nikolai Kochetov](#)).
- Introduce syntax for here documents. Example `SELECT $doc$ VALUE $doc$`. [#26671](#) ([Maksim Kita](#)). This change is backward incompatible if in query there are identifiers that contain `$`. [#28768](#).
- Now indices can handle `Nullable` types, including `isNull` and `isNotNull`. [#12433](#) and [#12455](#) ([Amos Bird](#)) and [#27250](#) ([Azat Khuzhin](#)). But this was done with on-disk format changes, and even though new server can read old data, old server cannot. Also, in case you have `MINMAX` data skipping indices, you may get `Data after mutation/merge is not byte-identical` error, since new index will have `.idx2` extension while before it was `.idx`. That said, that you should not delay updating all existing replicas, in this case, otherwise, if old replica (<21.9) will download data from new replica with 21.9+ it will not be able to apply index for downloaded part.

## New Feature

- Implementation of short circuit function evaluation, closes [#12587](#). Add settings `short_circuit_function_evaluation` to configure short circuit function evaluation. [#23367](#) ([Kruglov Pavel](#)).
- Add support for `INTERSECT`, `EXCEPT`, `ANY`, `ALL` operators. [#24757](#) ([Kirill Ershov](#)). ([Ksenia Sumarokova](#)).
- Add support for encryption at the virtual file system level (data encryption at rest) using AES-CTR algorithm. [#24206](#) ([Latysheva Alexandra](#)). ([Vitaly Baranov](#)) [#26733](#) [#26377](#) [#26465](#).
- Added natural language processing (NLP) functions for tokenization, stemming, lemmatizing and search in synonyms extensions. [#24997](#) ([Nikolay Degterinsky](#)).
- Added integration with S2 geometry library. [#24980](#) ([Andr0901](#)). ([Nikita Mikhaylov](#)).
- Add SQLite table engine, table function, database engine. [#24194](#) ([Arslan Gumerov](#)). ([Ksenia Sumarokova](#)).
- Added support for custom query for MySQL, PostgreSQL, ClickHouse, JDBC, Cassandra dictionary source. Closes [#1270](#). [#26995](#) ([Maksim Kita](#)).

- Add shared (replicated) storage of user, roles, row policies, quotas and settings profiles through ZooKeeper. #27426 (Kevin Michel).
- Add compression for `INTO OUTFILE` that automatically choose compression algorithm. Closes #3473. #27134 (Filatenkov Artur).
- Add `INSERT ... FROM INFILE` similarly to `SELECT ... INTO OUTFILE`. #27655 (Filatenkov Artur).
- Added `complex_key_range_hashed` dictionary. Closes #22029. #27629 (Maksim Kita).
- Support expressions in `JOIN ON` section. Close #21868. #24420 (Vladimir C).
- When client connects to server, it receives information about all warnings that are already were collected by server. (It can be disabled by using option `--no-warnings`). Add `system.warnings` table to collect warnings about server configuration. #26246 (Filatenkov Artur). #26282 (Filatenkov Artur).
- Allow using constant expressions from with and select in aggregate function parameters. Close #10945. #27531 (abel-cheng).
- Add `tupleToNameValuePairs`, a function that turns a named tuple into an array of pairs. #27505 (Braulio Valdivielso Martínez).
- Add support for `bzip2` compression method for import/export. Closes #22428. #27377 (Nikolay Degterinsky).
- Added `bitmapSubsetOffsetLimit(bitmap, offset, cardinality_limit)` function. It creates a subset of bitmap limit the results to `cardinality_limit` with offset of `offset`. #27234 (DHBin).
- Add column `default_database` to `system.users`. #27054 (kevin wan).
- Supported `cluster` macros inside table functions 'cluster' and 'clusterAllReplicas'. #26913 (polyprogrammist).
- Add new functions `currentRoles()`, `enabledRoles()`, `defaultRoles()`. #26780 (Vitaly Baranov).
- New functions `currentProfiles()`, `enabledProfiles()`, `defaultProfiles()`. #26714 (Vitaly Baranov).
- Add functions that return `(initial_)query_id` of the current query. This closes #23682. #26410 (Alexey Boykov).
- Add `REPLACE GRANT` feature. #26384 (Caspian).
- `EXPLAIN` query now has `EXPLAIN ESTIMATE ...` mode that will show information about read rows, marks and parts from MergeTree tables. Closes #23941. #26131 (fastio).
- Added `system.zookeeper_log` table. All actions of ZooKeeper client are logged into this table. Implements #25449. #26129 (tavplubix).
- Zero-copy replication for `ReplicatedMergeTree` over HDFS storage. #25918 (Zhichang Yu).
- Allow to insert Nested type as array of structs in `Arrow`, `ORC` and `Parquet` input format. #25902 (Kruglov Pavel).
- Add a new datatype `Date32` (store data as `Int32`), support date range same with `DateTime64` support load parquet date32 to ClickHouse `Date32` Add new function `toDate32` like `toDate`. #25774 (LiuNeng).
- Allow setting default database for users. #25268. #25687 (kevin wan).
- Add an optional parameter to `MongoDB` engine to accept connection string options and support SSL connection. Closes #21189. Closes #21041. #22045 (Omar Bazaraa).

## Experimental Feature

- Added a compression codec `AES_128_GCM_SIV` which encrypts columns instead of compressing them. [#19896 \(PHO\)](#). Will be rewritten, do not use.
- Rename `MaterializeMySQL` to `MaterializedMySQL`. [#26822 \(tavplubix\)](#).

## Performance Improvement

- Improve the performance of fast queries when `max_execution_time = 0` by reducing the number of `clock_gettime` system calls. [#27325 \(filimonov\)](#).
- Specialize date time related comparison to achieve better performance. This fixes [#27083](#) . [#27122 \(Amos Bird\)](#).
- Share file descriptors in concurrent reads of the same files. There is no noticeable performance difference on Linux. But the number of opened files will be significantly (10..100 times) lower on typical servers and it makes operations easier. See [#26214](#). [#26768 \(alexey-milovidov\)](#).
- Improve latency of short queries, that require reading from tables with large number of columns. [#26371 \(Anton Popov\)](#).
- Don't build sets for indices when analyzing a query. [#26365 \(Raúl Marín\)](#).
- Vectorize the SUM of Nullable integer types with native representation ([David Manzanares](#), [Raúl Marín](#)). [#26248 \(Raúl Marín\)](#).
- Compile expressions involving columns with `Enum` types. [#26237 \(Maksim Kita\)](#).
- Compile aggregate functions `groupBitOr`, `groupBitAnd`, `groupBitXor`. [#26161 \(Maksim Kita\)](#).
- Improved memory usage with better block size prediction when reading empty DEFAULT columns. Closes [#17317](#). [#25917 \(Vladimir Chebotarev\)](#).
- Reduce memory usage and number of read rows in queries with `ORDER BY primary_key`. [#25721 \(Anton Popov\)](#).
- Enable `distributed_push_down_limit` by default. [#27104 \(Azat Khuzhin\)](#).
- Make `toTimeZone` monotonicity when `timeZone` is a constant value to support partition purging when use sql like:. [#26261 \(huangzhaowei\)](#).

## Improvement

- Mark window functions as ready for general use. Remove the `allow_experimental_window_functions` setting. [#27184 \(Alexander Kuzmenkov\)](#).
- Improve compatibility with non-whole-minute timezone offsets. [#27080 \(Raúl Marín\)](#).
- If file descriptor in `File` table is regular file - allow to read multiple times from it. It allows `clickhouse-local` to read multiple times from `stdin` (with multiple `SELECT` queries or subqueries) if `stdin` is a regular file like `clickhouse-local --query "SELECT * FROM table UNION ALL SELECT * FROM table" ... < file` This closes [#11124](#). Co-authored with ([alexey-milovidov](#)). [#25960 \(BoloniniD\)](#).
- Remove duplicate index analysis and avoid possible invalid limit checks during projection analysis. [#27742 \(Amos Bird\)](#).
- Enable query parameters to be passed in the body of HTTP requests. [#27706 \(Hermano Lustosa\)](#).
- Disallow `arrayJoin` on partition expressions. [#27648 \(Raúl Marín\)](#).
- Log client IP address if authentication fails. [#27514 \(Misko Lee\)](#).

- Use bytes instead of strings for binary data in the GRPC protocol. #27431 (Vitaly Baranov).
- Send response with error message if HTTP port is not set and user tries to send HTTP request to TCP port. #27385 (Braulio Valdivielso Martínez).
- Add `_CAST` function for internal usage, which will not preserve type nullability, but non-internal cast will preserve according to setting `cast_keep_nullable`. Closes #12636. #27382 (Kseniia Sumarokova).
- Add setting `log_formatted_queries` to log additional formatted query into `system.query_log`. It's useful for normalized query analysis because functions like `normalizeQuery` and `normalizeQueryKeepNames` don't parse/format queries in order to achieve better performance. #27380 (Amos Bird).
- Add two settings `max_hyperscan_regexp_length` and `max_hyperscan_regexp_total_length` to prevent huge regexp being used in hyperscan related functions, such as `multiMatchAny`. #27378 (Amos Bird).
- Memory consumed by bitmap aggregate functions now is taken into account for memory limits. This closes #26555. #27252 (alexey-milovidov).
- Add 10 seconds cache for S3 proxy resolver. #27216 (ianton-ru).
- Split global mutex into individual regexp construction. This helps avoid huge regexp construction blocking other related threads. #27211 (Amos Bird).
- Support schema for PostgreSQL database engine. Closes #27166. #27198 (Kseniia Sumarokova).
- Track memory usage in clickhouse-client. #27191 (Filatenkov Artur).
- Try recording `query_kind` in `system.query_log` even when query fails to start. #27182 (Amos Bird).
- Added columns `replica_is_active` that maps replica name to is replica active status to table `system.replicas`. Closes #27138. #27180 (Maksim Kita).
- Allow to pass query settings via server URI in Web UI. #27177 (kolsys).
- Add a new metric called `MaxPushedDDLEntryID` which is the maximum ddl entry id that current node push to zookeeper. #27174 (Fuwang Hu).
- Improved the existence condition judgment and empty string node judgment when `clickhouse-keeper` creates znode. #27125 (小路).
- Merge JOIN correctly handles empty set in the right. #27078 (Vladimir C).
- Now functions can be shard-level constants, which means if it's executed in the context of some distributed table, it generates a normal column, otherwise it produces a constant value. Notable functions are: `hostName()`, `tcpPort()`, `version()`, `buildId()`, `uptime()`, etc. #27020 (Amos Bird).
- Updated `extractAllGroupsHorizontal` - upper limit on the number of matches per row can be set via optional third argument. #26961 (Vasily Nemkov).
- Expose RocksDB statistics via `system.rocksdb` table. Read rocksdb options from ClickHouse config (`rocksdb...` keys). NOTE: ClickHouse does not rely on RocksDB, it is just one of the additional integration storage engines. #26821 (Azat Khuzhin).
- Less verbose internal RocksDB logs. NOTE: ClickHouse does not rely on RocksDB, it is just one of the additional integration storage engines. This closes #26252. #26789 (alexey-milovidov).
- Changing default roles affects new sessions only. #26759 (Vitaly Baranov).
- Watchdog is disabled in docker by default. Fix for not handling `ctrl+c`. #26757 (Mikhail f. Shiryaev).
- SET PROFILE now applies constraints too if they're set for a passed profile. #26730 (Vitaly Baranov).

- Improve handling of `KILL QUERY` requests. [#26675](#) ([Raúl Marín](#)).
- `mapPopulatesSeries` function supports `Map` type. [#26663](#) ([Ildus Kurbangaliev](#)).
- Fix excessive (x2) connect attempts with `skip_unavailable_shards`. [#26658](#) ([Azat Khuzhin](#)).
- Avoid hanging `clickhouse-benchmark` if connection fails (i.e. on `EMFILE`). [#26656](#) ([Azat Khuzhin](#)).
- Allow more threads to be used by the Kafka engine. [#26642](#) ([feihengye](#)).
- Add round-robin support for `clickhouse-benchmark` (it does not differ from the regular multi host/port run except for statistics report). [#26607](#) ([Azat Khuzhin](#)).
- Executable dictionaries (`executable`, `executable_pool`) enable creation with DDL query using `clickhouse-local`. Closes [#22355](#). [#26510](#) ([Maksim Kita](#)).
- Set client query kind for `mysql` and `postgresql` compatibility protocol handlers. [#26498](#) ([anneji-dev](#)).
- Apply `LIMIT` on the shards for queries like `SELECT * FROM dist ORDER BY key LIMIT 10` w/  
`distributed_push_down_limit=1`. Avoid running `Distinct/LIMIT BY` steps for queries like `SELECT DISTINCT shading_key FROM dist ORDER BY key`. Now `distributed_push_down_limit` is respected by `optimize_distributed_group_by_sharding_key` optimization. [#26466](#) ([Azat Khuzhin](#)).
- Updated protobuf to 3.17.3. Changelogs are available on  
<https://github.com/protocolbuffers/protobuf/releases>. [#26424](#) ([Ilya Yatsishin](#)).
- Enable `use_hedged_requests` setting that allows to mitigate tail latencies on large clusters. [#26380](#) ([alexey-milovidov](#)).
- Improve behaviour with non-existing host in user allowed host list. [#26368](#) ([ianton-ru](#)).
- Add ability to set `Distributed` directory monitor settings via `CREATE TABLE` (i.e. `CREATE TABLE dist (key Int Engine=Distributed(cluster, db, table) SETTINGS monitor_batch_inserts=1` and similar). [#26336](#) ([Azat Khuzhin](#)).
- Save server address in history URLs in web UI if it differs from the origin of web UI. This closes [#26044](#).  
[#26322](#) ([alexey-milovidov](#)).
- Add events to profile calls to `sleep` / `sleepEachRow`. [#26320](#) ([Raúl Marín](#)).
- Allow to reuse connections of shards among different clusters. It also avoids creating new connections when using `cluster` table function. [#26318](#) ([Amos Bird](#)).
- Control the execution period of clear old temporary directories by parameter with default value.  
[#26212](#). [#26313](#) ([fastio](#)).
- Add a setting `function_range_max_elements_in_block` to tune the safety threshold for data volume generated by function `range`. This closes [#26303](#). [#26305](#) ([alexey-milovidov](#)).
- Check hash function at table creation, not at sampling. Add settings for MergeTree, if someone create a table with incorrect sampling column but sampling never be used, disable this settings for starting the server without exception. [#26256](#) ([zhaoyu](#)).
- Added `output_format_avro_string_column_pattern` setting to put specified String columns to Avro as string instead of default bytes. Implements [#22414](#). [#26245](#) ([Ilya Golshtein](#)).
- Add information about column sizes in `system.columns` table for `Log` and `TinyLog` tables. This closes [#9001](#).  
[#26241](#) ([Nikolay Degterinsky](#)).
- Don't throw exception when querying `system.detached_parts` table if there is custom disk configuration and `detached` directory does not exist on some disks. This closes [#26078](#). [#26236](#) ([alexey-milovidov](#)).

- Check for non-deterministic functions in keys, including constant expressions like `now()`, `today()`. This closes #25875. This closes #11333. #26235 (alexey-milovidov).
- convert timestamp and timestamptz data types to `DateTime64` in PostgreSQL table engine. #26234 (jasine).
- Apply aggressive IN index analysis for projections so that better projection candidate can be selected. #26218 (Amos Bird).
- Remove GLOBAL keyword for IN when scalar function is passed. In previous versions, if user specified `GLOBAL IN f(x)` exception was thrown. #26217 (Amos Bird).
- Add error id (like `BAD_ARGUMENTS`) to exception messages. This closes #25862. #26172 (alexey-milovidov).
- Fix incorrect output with --progress option for clickhouse-local. Progress bar will be cleared once it gets to 100% - same as it is done for clickhouse-client. Closes #17484. #26128 (Ksenia Sumarokova).
- Add `merge_selecting_sleep_ms` setting. #26120 (lthaooo).
- Remove complicated usage of Linux AIO with one block readahead and replace it with plain simple synchronous IO with `O_DIRECT`. In previous versions, the setting `min_bytes_to_use_direct_io` may not work correctly if `max_threads` is greater than one. Reading with direct IO (that is disabled by default for queries and enabled by default for large merges) will work in less efficient way. This closes #25997. #26003 (alexey-milovidov).
- Flush Distributed table on `REPLACE TABLE` query. Resolves #24566 - Do not replace (or create) table on `[CREATE OR] REPLACE TABLE ... AS SELECT` query if insertion into new table fails. Resolves #23175. #25895 (tavplubix).
- Add `views` column to `system.query_log` containing the names of the (materialized or live) views executed by the query. Adds a new log table (`system.query_views_log`) that contains information about each view executed during a query. Modifies view execution: When an exception is thrown while executing a view, any view that has already started will continue running until it finishes. This used to be the behaviour under `parallel_view_processing=true` and now it's always the same behaviour. - Dependent views now report reading progress to the context. #25714 (Raúl Marín).
- Do connection draining asynchronously upon finishing executing distributed queries. A new server setting is added `max_threads_for_connection_collector` which specifies the number of workers to recycle connections in background. If the pool is full, connection will be drained synchronously but a bit different than before: It's drained after we send EOS to client, query will succeed immediately after receiving enough data, and any exception will be logged instead of throwing to the client. Added setting `drain_timeout` (3 seconds by default). Connection draining will disconnect upon timeout. #25674 (Amos Bird).
- Support for multiple includes in configuration. It is possible to include users configuration, remote servers configuration from multiple sources. Simply place `<include />` element with `from_zk`, `from_env` or `incl` attribute and it will be replaced with the substitution. #24404 (nvartolomei).
- Fix multiple block insertion into distributed table with `insert_distributed_one_random_shard = 1`. This is a marginal feature. Mark as improvement. #23140 (Amos Bird).
- Support `LowCardinality` and `FixedString` keys/values for `Map` type. #21543 (hexiaoting).
- Enable reloading of local disk config. #19526 (taiyang-li).

## Bug Fix

- Fix a couple of bugs that may cause replicas to diverge. #27808 (tavplubix).

- Fix a rare bug in `DROP PART` which can lead to the error `Unexpected merged part intersects drop range`. [#27807 \(alesapin\)](#).
- Prevent crashes for some formats when `NULL` (tombstone) message was coming from Kafka. Closes [#19255](#). [#27794 \(filimonov\)](#).
- Fix column filtering with union distinct in subquery. Closes [#27578](#). [#27689 \(Kseniia Sumarokova\)](#).
- Fix bad type cast when functions like `arrayHas` are applied to arrays of `LowCardinality` of `Nullable` of different non-numeric types like `DateTime` and `DateTime64`. In previous versions bad cast occurs. In new version it will lead to exception. This closes [#26330](#). [#27682 \(alexey-milovidov\)](#).
- Fix postgresql table function resulting in non-closing connections. Closes [#26088](#). [#27662 \(Kseniia Sumarokova\)](#).
- Fixed another case of `Unexpected merged part ... intersecting drop range ...` error. [#27656 \(tavplubix\)](#).
- Fix an error with aliased column in `Distributed` table. [#27652 \(Vladimir C\)](#).
- After setting `max_memory_usage*` to non-zero value it was not possible to reset it back to 0 (unlimited). It's fixed. [#27638 \(tavplubix\)](#).
- Fixed underflow of the time value when constructing it from components. Closes [#27193](#). [#27605 \(Vasily Nemkov\)](#).
- Fix crash during projection materialization when some parts contain missing columns. This fixes [#27512](#). [#27528 \(Amos Bird\)](#).
- fix metric `BackgroundMessageBrokerSchedulePoolTask`, maybe mistyped. [#27452 \(Ben\)](#).
- Fix distributed queries with zero shards and aggregation. [#27427 \(Azat Khuzhin\)](#).
- Compatibility when `/proc/meminfo` does not contain KB suffix. [#27361 \(Mike Kot\)](#).
- Fix incorrect result for query with row-level security, `PREWHERE` and `LowCardinality` filter. Fixes [#27179](#). [#27329 \(Nikolai Kochetov\)](#).
- Fixed incorrect validation of partition id for `MergeTree` tables that created with old syntax. [#27328 \(tavplubix\)](#).
- Fix MySQL protocol when using parallel formats (CSV / TSV). [#27326 \(Raúl Marín\)](#).
- Fix `Cannot find column` error for queries with sampling. Was introduced in [#24574](#). Fixes [#26522](#). [#27301 \(Nikolai Kochetov\)](#).
- Fix errors like `Expected ColumnLowCardinality, gotUInt8` or `Bad cast from type DB::ColumnVector<char8_t> to DB::ColumnLowCardinality` for some queries with `LowCardinality` in `PREWHERE`. And more importantly, fix the lack of whitespace in the error message. Fixes [#23515](#). [#27298 \(Nikolai Kochetov\)](#).
- Fix `distributed_group_by_no_merge = 2` with `distributed_push_down_limit = 1` or `optimize_distributed_group_by_sharding_key = 1` with `LIMIT BY` and `LIMIT OFFSET`. [#27249 \(Azat Khuzhin\)](#). These are obscure combination of settings that no one is using.
- Fix mutation stuck on invalid partitions in non-replicated `MergeTree`. [#27248 \(Azat Khuzhin\)](#).
- In case of ambiguity, lambda functions prefer its arguments to other aliases or identifiers. [#27235 \(Raúl Marín\)](#).
- Fix column structure in merge join, close [#27091](#). [#27217 \(Vladimir C\)](#).

- In rare cases `system.detached_parts` table might contain incorrect information for some parts, it's fixed. Fixes #27114. #27183 (tavplubix).
- Fix uninitialized memory in functions `multiSearch*` with empty array, close #27169. #27181 (Vladimir C).
- Fix synchronization in GRPCServer. This PR fixes #27024. #27064 (Vitaly Baranov).
- Fixed `cache`, `complex_key_cache`, `ssd_cache`, `complex_key_ssd_cache` configuration parsing. Options `allow_expired_keys`, `max_update_queue_size`, `update_queue_push_timeout_milliseconds`, `query_wait_timeout_milliseconds` were not parsed for dictionaries with non `cache` type. #27032 (Maksim Kita).
- Fix possible mutation stack due to race with `DROP_RANGE`. #27002 (Azat Khuzhin).
- Now partition ID in queries like `ALTER TABLE ... PARTITION ID xxx` validates for correctness. Fixes #25718. #26963 (alesapin).
- Fix "Unknown column name" error with multiple JOINS in some cases, close #26899. #26957 (Vladimir C).
- Fix reading of custom TLDs (stops processing with lower buffer or bigger file). #26948 (Azat Khuzhin).
- Fix error `Missing columns: 'xxx'` when `DEFAULT` column references other non materialized column without `DEFAULT` expression. Fixes #26591. #26900 (alesapin).
- Fix loading of dictionary keys in library-bridge for library dictionary source. #26834 (Kseniia Sumarokova).
- Aggregate function parameters might be lost when applying some combinator causing exceptions like `Conversion from AggregateFunction(topKArray, Array(String)) to AggregateFunction(topKArray(10), Array(String)) is not supported.` It's fixed. Fixes #26196 and #26433. #26814 (tavplubix).
- Add `event_time_microseconds` value for `REMOVE_PART` in `system.part_log`. In previous versions is was not set. #26720 (Azat Khuzhin).
- Do not remove data on ReplicatedMergeTree table shutdown to avoid creating data to metadata inconsistency. #26716 (nvartolomei).
- Sometimes `SET ROLE` could work incorrectly, this PR fixes that. #26707 (Vitaly Baranov).
- Some fixes for parallel formatting (<https://github.com/ClickHouse/ClickHouse/issues/26694>). #26703 (Raúl Marín).
- Fix potential nullptr dereference in window functions. This fixes #25276. #26668 (Alexander Kuzmenkov).
- Fix clickhouse-client history file conversion (when upgrading from the format of 3 years old version of clickhouse-client) if file is empty. #26589 (Azat Khuzhin).
- Fix incorrect function names of `groupBitmapAnd/Or/Xor` (can be displayed in some occasions). This fixes. #26557 (Amos Bird).
- Update chown cmd check in clickhouse-server docker entrypoint. It fixes the bug that cluster pod restart failed (or timeout) on kubernetes. #26545 (Ky Li).
- Fix crash in `RabbitMQ` shutdown in case `RabbitMQ` setup was not started. Closes #26504. #26529 (Kseniia Sumarokova).
- Fix issues with `CREATE DICTIONARY` query if dictionary name or database name was quoted. Closes #26491. #26508 (Maksim Kita).

- Fix broken column name resolution after rewriting column aliases. This fixes #26432. #26475 (Amos Bird).
- Fix some fuzzed msan crash. Fixes #22517. #26428 (Nikolai Kochetov).
- Fix infinite non joined block stream in `partial_merge_join` close #26325. #26374 (Vladimir C).
- Fix possible crash when login as dropped user. This PR fixes #26073. #26363 (Vitaly Baranov).
- Fix `optimize_distributed_group_by_sharding_key` for multiple columns (leads to incorrect result w/ `optimize_skip_unused_shards=1/allow_nondeterministic_optimize_skip_unused_shards=1` and multiple columns in sharding key expression). #26353 (Azat Khuzhin).
- Fixed rare bug in lost replica recovery that may cause replicas to diverge. #26321 (tavplubix).
- Fix zstd decompression (for import/export in zstd framing format that is unrelated to tables data) in case there are escape sequences at the end of internal buffer. Closes #26013. #26314 (Kseniia Sumarokova).
- Fix logical error on join with totals, close #26017. #26250 (Vladimir C).
- Remove excessive newline in `thread_name` column in `system.stack_trace` table. This fixes #24124. #26210 (alexey-milovidov).
- Fix potential crash if more than one `untuple` expression is used. #26179 (alexey-milovidov).
- Don't throw exception in `toString` for Nullable Enum if Enum does not have a value for zero, close #25806. #26123 (Vladimir C).
- Fixed incorrect `sequence_id` in MySQL protocol packets that ClickHouse sends on exception during query execution. It might cause MySQL client to reset connection to ClickHouse server. Fixes #21184. #26051 (tavplubix).
- Fix for the case that `cutToFirstSignificantSubdomainCustom()`/`cutToFirstSignificantSubdomainCustomWithWWW()`/`firstSignificantSubdomainCustom()` returns incorrect type for consts, and hence `optimize_skip_unused_shards` does not work:. #26041 (Azat Khuzhin).
- Fix possible mismatched header when using normal projection with prewhere. This fixes #26020. #26038 (Amos Bird).
- Fix sharding\_key from column w/o function for remote() (before `select * from remote('127.1', system.one, dummy)` leads to Unknown column: `dummy`, there are only columns `.error`). #25824 (Azat Khuzhin).
- Fixed Not found column ... and Missing column ... errors when selecting from `MaterializeMySQL`. Fixes #23708, #24830, #25794. #25822 (tavplubix).
- Fix `optimize_skip_unused_shards_rewrite_in` for non-UInt64 types (may select incorrect shards eventually or throw Cannot infer type of an empty tuple or Function tuple requires at least one argument). #25798 (Azat Khuzhin).

## Build/Testing/Packaging Improvement

- Now we ran stateful and stateless tests in random timezones. Fixes #12439. Reading String as DateTime and writing DateTime as String in Protobuf format now respect timezone. Reading UInt16 as DateTime in Arrow and Parquet formats now treat it as Date and then converts to DateTime with respect to DateTime's timezone, because Date is serialized in Arrow and Parquet as UInt16. GraphiteMergeTree now respect time zone for rounding of times. Fixes #5098. Author: @alexey-milovidov. #15408 (alesapin).
- `clickhouse-test` supports SQL tests with `Jinja2` templates. #26579 (Vladimir C).

- Add support for build with clang-13. This closes #27705. #27714 (alexey-milovidov). #27777 (Sergei Semin).
- Add CMake options to build with or without specific CPU instruction set. This is for #17469 and #27509. #27508 (alexey-milovidov).
- Fix linking of auxiliar programs when using dynamic libraries. #26958 (Raúl Marín).
- Update RocksDB to 2021-07-16 master. #26411 (alexey-milovidov).

## ClickHouse release v21.8, 2021-08-12

### Upgrade Notes

- New version is using `Map` data type for system logs tables (`system.query_log`, `system.query_thread_log`, `system.processes`, `system.opentelemetry_span_log`). These tables will be auto-created with new data types. Virtual columns are created to support old queries. Closes #18698. #23934, #25773 (hexiaoting, sundy-li, Maksim Kita). If you want to *downgrade* from version 21.8 to older versions, you will need to cleanup system tables with logs manually. Look at `/var/lib/clickhouse/data/system/*_log`.

### New Features

- Add support for a part of SQL/JSON standard. #24148 (l1tsolaiki, Kseniia Sumarokova).
- Collect common system metrics (in `system.asynchronous_metrics` and `system.asynchronous_metric_log`) on CPU usage, disk usage, memory usage, IO, network, files, load average, CPU frequencies, thermal sensors, EDAC counters, system uptime; also added metrics about the scheduling jitter and the time spent collecting the metrics. It works similar to atop in ClickHouse and allows access to monitoring data even if you have no additional tools installed. Close #9430. #24416 (alexey-milovidov, Yegor Levankov).
- Add MaterializedPostgreSQL table engine and database engine. This database engine allows replicating a whole database or any subset of database tables. #20470 (Kseniia Sumarokova).
- Add new functions `leftPad()`, `rightPad()`, `leftPadUTF8()`, `rightPadUTF8()`. #26075 (Vitaly Baranov).
- Add the `FIRST` keyword to the `ADD INDEX` command to be able to add the index at the beginning of the indices list. #25904 (xjewer).
- Introduce `system.data_skipping_indices` table containing information about existing data skipping indices. Close #7659. #25693 (Dmitry Novik).
- Add `bin/unbin` functions. #25609 (zhaoyu).
- Support `Map` and `UInt128`, `Int128`, `UInt256`, `Int256` types in `mapAdd` and `mapSubtract` functions. #25596 (Ildus Kurbangaliev).
- Support `DISTINCT ON (columns)` expression, close #25404. #25589 (Zijie Lu).
- Add an ability to reset a custom setting to default and remove it from the table's metadata. It allows rolling back the change without knowing the system/config's default. Closes #14449. #17769 (xjewer).
- Render pipelines as graphs in Web UI if `EXPLAIN PIPELINE graph = 1` query is submitted. #26067 (alexey-milovidov).

### Performance Improvements

- Compile aggregate functions. Use option `compile_aggregate_expressions` to enable it. #24789 (Maksim Kita).

- Improve latency of short queries that require reading from tables with many columns. #26371 (Anton Popov).

## Improvements

- Use Map data type for system logs tables (system.query\_log, system.query\_thread\_log, system.processes, system.opentelemetry\_span\_log). These tables will be auto-created with new data types. Virtual columns are created to support old queries. Closes #18698. #23934, #25773 (hexiaoting, sundy-li, Maksim Kita).
- For a dictionary with a complex key containing only one attribute, allow not wrapping the key expression in tuple for functions dictGet, dictHas. #26130 (Maksim Kita).
- Implement function bin/hex from AggregateFunction states. #26094 (zhaoyu).
- Support arguments of UUID type for empty and notEmpty functions. UUID is empty if it is all zeros (nil UUID). Closes #3446. #25974 (zhaoyu).
- Add support for SET SQL\_SELECT\_LIMIT in MySQL protocol. Closes #17115. #25972 (Kseniia Sumarokova).
- More instrumentation for network interaction: add counters for recv/send bytes; add gauges for recvs/sends. Added missing documentation. Close #5897. #25962 (alexey-milovidov).
- Add setting optimize\_move\_to\_prewhere\_if\_final. If query has FINAL, the optimization move\_to\_prewhere will be enabled only if both optimize\_move\_to\_prewhere and optimize\_move\_to\_prewhere\_if\_final are enabled. Closes #8684. #25940 (Kseniia Sumarokova).
- Allow complex quoted identifiers of JOINed tables. Close #17861. #25924 (alexey-milovidov).
- Add support for Unicode (e.g. Chinese, Cyrillic) components in Nested data types. Close #25594. #25923 (alexey-milovidov).
- Allow quantiles\* functions to work with aggregate\_functions\_null\_for\_empty. Close #25892. #25919 (alexey-milovidov).
- Allow parameters for parametric aggregate functions to be arbitrary constant expressions (e.g., `1 + 2`), not just literals. It also allows using the query parameters (in parameterized queries like `{param:UInt8}`) inside parametric aggregate functions. Closes #11607. #25910 (alexey-milovidov).
- Correctly throw the exception on the attempt to parse an invalid Date. Closes #6481. #25909 (alexey-milovidov).
- Support for multiple includes in configuration. It is possible to include users configuration, remote server configuration from multiple sources. Simply place `<include />` element with `from_zk`, `from_env` or `incl` attribute, and it will be replaced with the substitution. #24404 (nvartolomei).
- Support for queries with a column named "null" (it must be specified in back-ticks or double quotes) and ON CLUSTER. Closes #24035. #25907 (alexey-milovidov).
- Support LowCardinality, Decimal, and UUID for JSONExtract. Closes #24606. #25900 (Kseniia Sumarokova).
- Convert history file from `readline` format to `replxx` format. #25888 (Azat Khuzhin).
- Fix an issue which can lead to intersecting parts after `DROP PART` or background deletion of an empty part. #25884 (alesapin).
- Better handling of lost parts for ReplicatedMergeTree tables. Fixes rare inconsistencies in ReplicationQueue. Fixes #10368. #25820 (alesapin).
- Allow starting clickhouse-client with unreadable working directory. #25817 (ianton-ru).
- Fix "No available columns" error for Merge storage. #25801 (Azat Khuzhin).

- MySQL Engine now supports the exchange of column comments between MySQL and ClickHouse. #25795 (Storozhuk Kostiantyn).
- Fix inconsistent behaviour of `GROUP BY` constant on empty set. Closes #6842. #25786 (Ksenia Sumarokova).
- Cancel already running merges in partition on `DROP PARTITION` and `TRUNCATE` for `ReplicatedMergeTree`. Resolves #17151. #25684 (tavplubix).
- Support `ENUM`` data type for `MaterializeMySQL`. #25676 (Storozhuk Kostiantyn).
- Support materialized and aliased columns in `JOIN`, close #13274. #25634 (Vladimir C).
- Fix possible logical race condition between `ALTER TABLE ... DETACH` and background merges. #25605 (Azat Khuzhin).
- Make `NetworkReceiveElapsedMicroseconds` metric to correctly include the time spent waiting for data from the client to `INSERT`. Close #9958. #25602 (alexey-milovidov).
- Support `TRUNCATE TABLE` for S3 and HDFS. Close #25530. #25550 (Ksenia Sumarokova).
- Support for dynamic reloading of config to change number of threads in pool for background jobs execution (merges, mutations, fetches). #25548 (Nikita Mikhaylov).
- Allow extracting of non-string element as string using `JSONExtract`. This is for #25414. #25452 (Amos Bird).
- Support regular expression in `Database` argument for `StorageMerge`. Close #776. #25064 (flynn).
- Web UI: if the value looks like a URL, automatically generate a link. #25965 (alexey-milovidov).
- Make `sudo service clickhouse-server start` to work on systems with `systemd` like Centos 8. Close #14298. Close #17799. #25921 (alexey-milovidov).

## Bug Fixes

- Fix incorrect `SET ROLE` in some cases. #26707 (Vitaly Baranov).
- Fix potential `nullptr` dereference in window functions. Fix #25276. #26668 (Alexander Kuzmenkov).
- Fix incorrect function names of `groupBitmapAnd/Or/Xor`. Fix #26557 (Amos Bird).
- Fix crash in RabbitMQ shutdown in case RabbitMQ setup was not started. Closes #26504. #26529 (Ksenia Sumarokova).
- Fix issues with `CREATE DICTIONARY` query if dictionary name or database name was quoted. Closes #26491. #26508 (Maksim Kita).
- Fix broken name resolution after rewriting column aliases. Fix #26432. #26475 (Amos Bird).
- Fix infinite non-joined block stream in `partial_merge_join` close #26325. #26374 (Vladimir C).
- Fix possible crash when login as dropped user. Fix #26073. #26363 (Vitaly Baranov).
- Fix `optimize_distributed_group_by_sharding_key` for multiple columns (leads to incorrect result w/ `optimize_skip_unused_shards=1/allow_nondeterministic_optimize_skip_unused_shards=1` and multiple columns in sharding key expression). #26353 (Azat Khuzhin).
- `CAST` from `Date` to `DateTime` (or `DateTime64`) was not using the timezone of the `DateTime` type. It can also affect the comparison between `Date` and `DateTime`. Inference of the common type for `Date` and `DateTime` also was not using the corresponding timezone. It affected the results of function `if` and array construction. Closes #24128. #24129 (Maksim Kita).

- Fixed rare bug in lost replica recovery that may cause replicas to diverge. #26321 (tavplubix).
- Fix zstd decompression in case there are escape sequences at the end of internal buffer. Closes #26013. #26314 (Kseniia Sumarokova).
- Fix logical error on join with totals, close #26017. #26250 (Vladimir C).
- Remove excessive newline in `thread_name` column in `system.stack_trace` table. Fix #24124. #26210 (alexey-milovidov).
- Fix `joinGet` with `LowCarinality` columns, close #25993. #26118 (Vladimir C).
- Fix possible crash in `pointInPolygon` if the setting `validate_polygons` is turned off. #26113 (alexey-milovidov).
- Fix throwing exception when iterate over non-existing remote directory. #26087 (ianton-ru).
- Fix rare server crash because of `abort` in ZooKeeper client. Fixes #25813. #26079 (alesapin).
- Fix wrong thread count estimation for right subquery join in some cases. Close #24075. #26052 (Vladimir C).
- Fixed incorrect `sequence_id` in MySQL protocol packets that ClickHouse sends on exception during query execution. It might cause MySQL client to reset connection to ClickHouse server. Fixes #21184. #26051 (tavplubix).
- Fix possible mismatched header when using normal projection with `PREWHERE`. Fix #26020. #26038 (Amos Bird).
- Fix formatting of type `Map` with integer keys to `JSON`. #25982 (Anton Popov).
- Fix possible deadlock during query profiler stack unwinding. Fix #25968. #25970 (Maksim Kita).
- Fix crash on call `dictGet()` with bad arguments. #25913 (Vitaly Baranov).
- Fixed `scram-sha-256` authentication for PostgreSQL engines. Closes #24516. #25906 (Kseniia Sumarokova).
- Fix extremely long backoff for background tasks when the background pool is full. Fixes #25836. #25893 (alesapin).
- Fix ARM exception handling with non default page size. Fixes #25512, #25044, #24901, #23183, #20221, #19703, #19028, #18391, #18121, #17994, #12483. #25854 (Maksim Kita).
- Fix sharding\_key from column w/o function for `remote()` (before `select * from remote('127.1', system.one, dummy)` leads to `Unknown column: dummy, there are only columns .error`). #25824 (Azat Khuzhin).
- Fixed Not found column ... and Missing column ... errors when selecting from `MaterializeMySQL`. Fixes #23708, #24830, #25794. #25822 (tavplubix).
- Fix `optimize_skip_unused_shards_rewrite_in` for non-UInt64 types (may select incorrect shards eventually or throw Cannot infer type of an empty tuple or Function tuple requires at least one argument). #25798 (Azat Khuzhin).
- Fix rare bug with `DROP PART` query for `ReplicatedMergeTree` tables which can lead to error message `Unexpected merged part intersecting drop range`. #25783 (alesapin).
- Fix bug in `TTL` with `GROUP BY` expression which refuses to execute `TTL` after first execution in part. #25743 (alesapin).
- Allow `StorageMerge` to access tables with aliases. Closes #6051. #25694 (Kseniia Sumarokova).

- Fix slow dict join in some cases, close #24209. #25618 (Vladimir C).
- Fix ALTER MODIFY COLUMN of columns, which participates in TTL expressions. #25554 (Anton Popov).
- Fix assertion in PREWHERE with non-UInt8 type, close #19589. #25484 (Vladimir C).
- Fix some fuzzed msan crash. Fixes #22517. #26428 (Nikolai Kochetov).
- Update chown cmd check in clickhouse-server docker entrypoint. It fixes error 'cluster pod restart failed (or timeout)' on kubernetes. #26545 (Ky Li).

## ClickHouse release v21.7, 2021-07-09

### Backward Incompatible Change

- Improved performance of queries with explicitly defined large sets. Added compatibility setting `legacy_column_name_of_tuple_literal`. It makes sense to set it to `true`, while doing rolling update of cluster from version lower than 21.7 to any higher version. Otherwise distributed queries with explicitly defined sets at `IN` clause may fail during update. #25371 (Anton Popov).
- Forward/backward incompatible change of maximum buffer size in clickhouse-keeper (an experimental alternative to ZooKeeper). Better to do it now (before production), than later. #25421 (alesapin).

### New Feature

- Support configuration in YAML format as alternative to XML. This closes #3607. #21858 (BoloniniD).
- Provides a way to restore replicated table when the data is (possibly) present, but the ZooKeeper metadata is lost. Resolves #13458. #13652 (Mike Kot).
- Support structs and maps in Arrow/Parquet/ORC and dictionaries in Arrow input/output formats. Present new setting `output_format_arrow_low_cardinality_as_dictionary`. #24341 (Kruglov Pavel).
- Added support for `Array` type in dictionaries. #25119 (Maksim Kita).
- Added function `bitPositionsToArray`. Closes #23792. Author [Kevin Wan] (@MaxWk). #25394 (Maksim Kita).
- Added function `dateName` to return names like 'Friday' or 'April'. Author [Daniil Kondratyev] (@dankondr). #25372 (Maksim Kita).
- Add `toJSONString` function to serialize columns to their JSON representations. #25164 (Amos Bird).
- Now `query_log` has two new columns: `initial_query_start_time`, `initial_query_start_time_microsecond` that record the starting time of a distributed query if any. #25022 (Amos Bird).
- Add aggregate function `segmentLengthSum`. #24250 (flynn).
- Add a new boolean setting `prefer_global_in_and_join` which defaults all IN/JOIN as GLOBAL IN/JOIN. #23434 (Amos Bird).
- Support ALTER DELETE queries for Join table engine. #23260 (foolchi).
- Add `quantileBFloat16` aggregate function as well as the corresponding `quantilesBFloat16` and `medianBFloat16`. It is very simple and fast quantile estimator with relative error not more than 0.390625%. This closes #16641. #23204 (Ivan Novitskiy).
- Implement `sequenceNextNode()` function useful for flow analysis. #19766 (achimbab).

### Experimental Feature

- Add support for virtual filesystem over HDFS. #11058 (overshov) (Ksenia Sumarokova).

- Now clickhouse-keeper (an experimental alternative to ZooKeeper) supports ZooKeeper-like digest ACLs. #24448 (alesapin).

## Performance Improvement

- Added optimization that transforms some functions to reading of subcolumns to reduce amount of read data. E.g., statement `col IS NULL` is transformed to reading of subcolumn `col.null`. Optimization can be enabled by setting `optimize_functions_to_subcolumns` which is currently off by default. #24406 (Anton Popov).
- Rewrite more columns to possible alias expressions. This may enable better optimization, such as projections. #24405 (Amos Bird).
- Index of type `bloom_filter` can be used for expressions with `hasAny` function with constant arrays. This closes: #24291. #24900 (Vasily Nemkov).
- Add exponential backoff to reschedule read attempt in case RabbitMQ queues are empty. (ClickHouse has support for importing data from RabbitMQ). Closes #24340. #24415 (Kseniia Sumarokova).

## Improvement

- Allow to limit bandwidth for replication. Add two Replicated\*MergeTree settings: `max_replicated_fetches_network_bandwidth` and `max_replicated_sends_network_bandwidth` which allows to limit maximum speed of replicated fetches/sends for table. Add two server-wide settings (in `default` user profile): `max_replicated_fetches_network_bandwidth_for_server` and `max_replicated_sends_network_bandwidth_for_server` which limit maximum speed of replication for all tables. The settings are not followed perfectly accurately. Turned off by default. Fixes #1821. #24573 (alesapin).
- Resource constraints and isolation for ODBC and Library bridges. Use separate `clickhouse-bridge` group and user for bridge processes. Set `oom_score_adj` so the bridges will be first subjects for OOM killer. Set set maximum RSS to 1 GiB. Closes #23861. #25280 (Kseniia Sumarokova).
- Add standalone `clickhouse-keeper` symlink to the main `clickhouse` binary. Now it's possible to run coordination without the main `clickhouse` server. #24059 (alesapin).
- Use global settings for query to `VIEW`. Fixed the behavior when queries to `VIEW` use local settings, that leads to errors if setting on `CREATE VIEW` and `SELECT` were different. As for now, `VIEW` won't use these modified settings, but you can still pass additional settings in `SETTINGS` section of `CREATE VIEW` query. Close #20551. #24095 (Vladimir).
- On server start, parts with incorrect partition ID would not be ever removed, but always detached. #25070. #25166 (Nikolai Kochetov).
- Increase size of background schedule pool to 128 (`background_schedule_pool_size` setting). It allows avoiding replication queue hung on slow zookeeper connection. #25072 (alesapin).
- Add merge tree setting `max_parts_to_merge_at_once` which limits the number of parts that can be merged in the background at once. Doesn't affect `OPTIMIZE FINAL` query. Fixes #1820. #24496 (alesapin).
- Allow `NOT IN` operator to be used in partition pruning. #24894 (Amos Bird).
- Recognize IPv4 addresses like `127.0.1.1` as local. This is controversial and closes #23504. Michael Filimonov will test this feature. #24316 (alexey-milovidov).
- ClickHouse database created with MaterializeMySQL (it is an experimental feature) now contains all column comments from the MySQL database that materialized. #25199 (Storozhuk Kostiantyn).

- Add settings (`connection_auto_close`/`connection_max_tries`/`connection_pool_size`) for MySQL storage engine. #24146 (Azat Khuzhin).
- Improve startup time of Distributed engine. #25663 (Azat Khuzhin).
- Improvement for Distributed tables. Drop replicas from dirname for `internal_replication=true` (allows `INSERT` into Distributed with cluster from any number of replicas, before only 15 replicas was supported, everything more will fail with `ENAMETOOLONG` while creating directory for async blocks). #25513 (Azat Khuzhin).
- Added support `Interval` type for `LowCardinality`. It is needed for intermediate values of some expressions. Closes #21730. #25410 (Vladimir).
- Add `==` operator on time conditions for `sequenceMatch` and `sequenceCount` functions. For eg: `sequenceMatch('(?1)(?t==1)(?2)')(time, data = 1, data = 2)`. #25299 (Christophe Kalenzaga).
- Add settings `http_max_fields`, `http_max_field_name_size`, `http_max_field_value_size`. #25296 (Ivan).
- Add support for function `if` with `Decimal` and `Int` types on its branches. This closes #20549. This closes #10142. #25283 (alexey-milovidov).
- Update prompt in `clickhouse-client` and display a message when reconnecting. This closes #10577. #25281 (alexey-milovidov).
- Correct memory tracking in aggregate function `topK`. This closes #25259. #25260 (alexey-milovidov).
- Fix `topLevelDomain` for IDN hosts (i.e. `example.pф`), before it returns empty string for such hosts. #25103 (Azat Khuzhin).
- Detect Linux kernel version at runtime (for worked nested epoll, that is required for `async_socket_for_remote/use_hedged_requests`, otherwise remote queries may stuck). #25067 (Azat Khuzhin).
- For distributed query, when `optimize_skip_unused_shards=1`, allow to skip shard with condition like (`sharding key`) `IN` (one-element-tuple). (Tuples with many elements were supported. Tuple with single element did not work because it is parsed as literal). #24930 (Amos Bird).
- Improved log messages of S3 errors, no more double whitespaces in case of empty keys and buckets. #24897 (Vladimir Chebotarev).
- Some queries require multi-pass semantic analysis. Try reusing built sets for `IN` in this case. #24874 (Amos Bird).
- Respect `max_distributed_connections` for `insert_distributed_sync` (otherwise for huge clusters and sync insert it may run out of `max_thread_pool_size`). #24754 (Azat Khuzhin).
- Avoid hiding errors like `Limit for rows or bytes to read exceeded` for scalar subqueries. #24545 (nvartolomei).
- Make String-to-Int parser stricter so that `toInt64('+')` will throw. #24475 (Amos Bird).
- If `SSD_CACHE` is created with DDL query, it can be created only inside `user_files` directory. #24466 (Maksim Kita).
- PostgreSQL support for specifying non default schema for insert queries. Closes #24149. #24413 (Kseniia Sumarokova).
- Fix IPv6 addresses resolving (i.e. fixes `select * from remote('[:1]', system.one)`). #24319 (Azat Khuzhin).
- Fix trailing whitespaces in `FROM` clause with subqueries in multiline mode, and also changes the output of the queries slightly in a more human friendly way. #24151 (Azat Khuzhin).

- Improvement for Distributed tables. Add ability to split distributed batch on failures (i.e. due to memory limits, corruptions), under `distributed_directory_monitor_split_batch_on_failure` (OFF by default). #23864 (Azat Khuzhin).
- Handle column name clashes for Join table engine. Closes #20309. #23769 (Vladimir).
- Display progress for File table engine in `clickhouse-local` and on INSERT query in `clickhouse-client` when data is passed to stdin. Closes #18209. #23656 (Ksenia Sumarokova).
- Bugfixes and improvements of `clickhouse-copier`. Allow to copy tables with different (but compatible schemas). Closes #9159. Added test to copy ReplacingMergeTree. Closes #22711. Support TTL on columns and Data Skipping Indices. It simply removes it to create internal Distributed table (underlying table will have TTL and skipping indices). Closes #19384. Allow to copy MATERIALIZED and ALIAS columns. There are some cases in which it could be helpful (e.g. if this column is in PRIMARY KEY). Now it could be allowed by setting `allow_to_copy_alias_and_materialized_columns` property to true in task configuration. Closes #9177. Closes [#11007] (<https://github.com/ClickHouse/ClickHouse/issues/11007>). Closes #9514. Added a property `allow_to_drop_target_partitions` in task configuration to drop partition in original table before moving helping tables. Closes #20957. Get rid of `OPTIMIZE DEDUPLICATE` query. This hack was needed, because `ALTER TABLE MOVE PARTITION` was retried many times and plain MergeTree tables don't have deduplication. Closes #17966. Write progress to ZooKeeper node on path `task_path + /status` in JSON format. Closes #20955. Support for ReplicatedTables without arguments. Closes #24834. #23518 (Nikita Mikhaylov).
- Added sleep with backoff between read retries from S3. #23461 (Vladimir Chebotarev).
- Respect `insert_allow_materialized_columns` (allows materialized columns) for INSERT into Distributed table. #23349 (Azat Khuzhin).
- Add ability to push down LIMIT for distributed queries. #23027 (Azat Khuzhin).
- Fix zero-copy replication with several S3 volumes (Fixes #22679). #22864 (ianton-ru).
- Resolve the actual port number bound when a user requests any available port from the operating system to show it in the log message. #25569 (bnaecker).
- Fixed case, when sometimes conversion of postgres arrays resulted in String data type, not n-dimensional array, because `atndims` works incorrectly in some cases. Closes #24804. #25538 (Ksenia Sumarokova).
- Fix conversion of DateTime with timezone for MySQL, PostgreSQL, ODBC. Closes #5057. #25528 (Ksenia Sumarokova).
- Distinguish KILL MUTATION for different tables (fixes unexpected `Cancelled mutating parts` error). #25025 (Azat Khuzhin).
- Allow to declare S3 disk at root of bucket (S3 virtual filesystem is an experimental feature under development). #24898 (Vladimir Chebotarev).
- Enable reading of subcolumns (e.g. components of Tuples) for distributed tables. #24472 (Anton Popov).
- A feature for MySQL compatibility protocol: make `user` function to return correct output. Closes #25697. #25697 (sundyli).

## Bug Fix

- Improvement for backward compatibility. Use old modulo function version when used in partition key. Closes #23508. #24157 (Ksenia Sumarokova).
- Fix extremely rare bug on low-memory servers which can lead to the inability to perform merges without restart. Possibly fixes #24603. #24872 (alesapin).

- Fix extremely rare error Tagging already tagged part in replication queue during concurrent `alter move/replace partition`. Possibly fixes #22142. #24961 (alesapin).
- Fix potential crash when calculating aggregate function states by aggregation of aggregate function states of other aggregate functions (not a practical use case). See #24523. #25015 (alexey-milovidov).
- Fixed the behavior when query `SYSTEM RESTART REPLICA` or `SYSTEM SYNC REPLICA` does not finish. This was detected on server with extremely low amount of RAM. #24457 (Nikita Mikhaylov).
- Fix bug which can lead to ZooKeeper client hung inside clickhouse-server. #24721 (alesapin).
- If ZooKeeper connection was lost and replica was cloned after restoring the connection, its replication queue might contain outdated entries. Fixed failed assertion when replication queue contains intersecting virtual parts. It may rarely happen if some data part was lost. Print error in log instead of terminating. #24777 (tavplubix).
- Fix lost WHERE condition in expression-push-down optimization of query plan (setting `query_plan_filter_push_down = 1` by default). Fixes #25368. #25370 (Nikolai Kochetov).
- Fix bug which can lead to intersecting parts after merges with TTL: Part `all_40_40_0` is covered by `all_40_40_1` but should be merged into `all_40_41_1`. This shouldn't happen often.. #25549 (alesapin).
- On ZooKeeper connection loss `ReplicatedMergeTree` table might wait for background operations to complete before trying to reconnect. It's fixed, now background operations are stopped forcefully. #25306 (tavplubix).
- Fix error `Key expression contains comparison between incompatible types` for queries with `ARRAY JOIN` in case if array is used in primary key. Fixes #8247. #25546 (Anton Popov).
- Fix wrong totals for query `WITH TOTALS` and `WITH FILL`. Fixes #20872. #25539 (Anton Popov).
- Fix data race when querying `system.clusters` while reloading the cluster configuration at the same time. #25737 (Amos Bird).
- Fixed `No such file or directory` error on moving `Distributed` table between databases. Fixes #24971. #25667 (tavplubix).
- `REPLACE PARTITION` might be ignored in rare cases if the source partition was empty. It's fixed. Fixes #24869. #25665 (tavplubix).
- Fixed a bug in `Replicated` database engine that might rarely cause some replica to skip enqueued DDL query. #24805 (tavplubix).
- Fix null pointer dereference in `EXPLAIN AST` without query. #25631 (Nikolai Kochetov).
- Fix waiting of automatic dropping of empty parts. It could lead to full filling of background pool and stuck of replication. #23315 (Anton Popov).
- Fix restore of a table stored in S3 virtual filesystem (it is an experimental feature not ready for production). #25601 (ianton-ru).
- Fix nullptr dereference in Arrow format when using `Decimal256`. Add `Decimal256` support for Arrow format. #25531 (Kruglov Pavel).
- Fix excessive underscore before the names of the preprocessed configuration files. #25431 (Vitaly Baranov).
- A fix for `clickhouse-copier` tool: Fix segfault when sharding\_key is absent in task config for copier. #25419 (Nikita Mikhaylov).

- Fix `REPLACE` column transformer when used in DDL by correctly quoting the formatted query. This fixes #23925. #25391 (Amos Bird).
- Fix the possibility of non-deterministic behaviour of the `quantileDeterministic` function and similar. This closes #20480. #25313 (alexey-milovidov).
- Support `SimpleAggregateFunction(LowCardinality)` for `SummingMergeTree`. Fixes #25134. #25300 (Nikolai Kochetov).
- Fix logical error with exception message "Cannot sum Array/Tuple in min/maxMap". #25298 (Kruglov Pavel).
- Fix error `Bad cast from type DB::ColumnLowCardinality to DB::ColumnVector<char8_t>` for queries where `LowCardinality` argument was used for IN (this bug appeared in 21.6). Fixes #25187. #25290 (Nikolai Kochetov).
- Fix incorrect behaviour of `joinGetOrNull` with not-nullable columns. This fixes #24261. #25288 (Amos Bird).
- Fix incorrect behaviour and UBSan report in big integers. In previous versions `CAST(1e19 AS UInt128)` returned zero. #25279 (alexey-milovidov).
- Fixed an error which occurred while inserting a subset of columns using `CSVWithNames` format. Fixes #25129. #25169 (Nikita Mikhaylov).
- Do not use table's projection for `SELECT` with `FINAL`. It is not supported yet. #25163 (Amos Bird).
- Fix possible parts loss after updating up to 21.5 in case table used `UUID` in partition key. (It is not recommended to use `UUID` in partition key). Fixes #25070. #25127 (Nikolai Kochetov).
- Fix crash in query with cross join and `joined_subquery_requires_alias = 0`. Fixes #24011. #25082 (Nikolai Kochetov).
- Fix bug with constant maps in `mapContains` function that lead to error `empty column was returned by function mapContains`. Closes #25077. #25080 (Kruglov Pavel).
- Remove possibility to create tables with columns referencing themselves like `a UInt32 ALIAS a + 1` or `b UInt32 MATERIALIZED b`. Fixes #24910, #24292. #25059 (alesapin).
- Fix wrong result when using aggregate projection with *not empty* `GROUP BY` key to execute query with `GROUP BY` by empty key. #25055 (Amos Bird).
- Fix serialization of splitted nested messages in Protobuf format. This PR fixes #24647. #25000 (Vitaly Baranov).
- Fix limit/offset settings for distributed queries (ignore on the remote nodes). #24940 (Azat Khuzhin).
- Fix possible heap-buffer-overflow in Arrow format. #24922 (Kruglov Pavel).
- Fixed possible error 'Cannot read from istream at offset 0' when reading a file from DiskS3 (S3 virtual filesystem is an experimental feature under development that should not be used in production). #24885 (Pavel Kovalenko).
- Fix "Missing columns" exception when joining Distributed Materialized View. #24870 (Azat Khuzhin).
- Allow `NONE` values in postgresql compatibility protocol. Closes #22622. #24857 (Ksenia Sumarokova).
- Fix bug when exception `Mutation` was killed can be thrown to the client on mutation wait when mutation not loaded into memory yet. #24809 (alesapin).

- Fixed bug in deserialization of random generator state with might cause some data types such as `AggregateFunction(groupArraySample(N), T)` to behave in a non-deterministic way. #24538 ([tavplubix](#)).
- Disallow building uniqXXXXStates of other aggregation states. #24523 ([Raúl Marín](#)). Then allow it back by actually eliminating the root cause of the related issue. ([alexey-milovidov](#)).
- Fix usage of tuples in `CREATE .. AS SELECT` queries. #24464 ([Anton Popov](#)).
- Fix computation of total bytes in Buffer table. In current ClickHouse version `total_writes.bytes` counter decreases too much during the buffer flush. It leads to counter overflow and `totalBytes` return something around 17.44 EB some time after the flush. #24450 ([DimasKovas](#)).
- Fix incorrect information about the monotonicity of `toWeek` function. This fixes #24422 . This bug was introduced in <https://github.com/ClickHouse/ClickHouse/pull/5212> , and was exposed later by smarter partition pruner. #24446 ([Amos Bird](#)).
- When user authentication is managed by LDAP. Fixed potential deadlock that can happen during LDAP role (re)mapping, when LDAP group is mapped to a nonexistent local role. #24431 ([Denis Glazachev](#)).
- In "multipart/form-data" message consider the CRLF preceding a boundary as part of it. Fixes #23905. #24399 ([Ivan](#)).
- Fix drop partition with intersect fake parts. In rare cases there might be parts with mutation version greater than current block number. #24321 ([Amos Bird](#)).
- Fixed a bug in moving Materialized View from Ordinary to Atomic database (RENAME TABLE query). Now inner table is moved to new database together with Materialized View. Fixes #23926. #24309 ([tavplubix](#)).
- Allow empty HTTP headers. Fixes #23901. #24285 ([Ivan](#)).
- Correct processing of mutations (ALTER UPDATE/DELETE) in Memory tables. Closes #24274. #24275 ([flynn](#)).
- Make column LowCardinality property in JOIN output the same as in the input, close #23351, close #20315. #24061 ([Vladimir](#)).
- A fix for Kafka tables. Fix the bug in failover behavior when Engine = Kafka was not able to start consumption if the same consumer had an empty assignment previously. Closes #21118. #21267 ([filimonov](#)).

## Build/Testing/Packaging Improvement

- Add `darwin-aarch64` (Mac M1 / Apple Silicon) builds in CI #25560 ([Ivan](#)) and put the links to the docs and website ([alexey-milovidov](#)).
- Adds cross-platform embedding of binary resources into executables. It works on Illumos. #25146 ([bnaecker](#)).
- Add join related options to stress tests to improve fuzzing. #25200 ([Vladimir](#)).
- Enable build with s3 module in osx #25217. #25218 ([kevin wan](#)).
- Add integration test cases to cover JDBC bridge. #25047 ([Zhichun Wu](#)).
- Integration tests configuration has special treatment for dictionaries. Removed remaining dictionaries manual setup. #24728 ([Ilya Yatsishin](#)).
- Add libfuzzer tests for YAMLParser class. #24480 ([BoloniniD](#)).

- Ubuntu 20.04 is now used to run integration tests, docker-compose version used to run integration tests is updated to 1.28.2. Environment variables now take effect on docker-compose. Rework `test_dictionaries_all_layouts_separate_sources` to allow parallel run. [#20393](#) ([Ilya Yatsishin](#)).
- Fix TOCTOU error in installation script. [#25277](#) ([alexey-milovidov](#)).

## ClickHouse release 21.6, 2021-06-05

### Upgrade Notes

- `zstd` compression library is updated to v1.5.0. You may get messages about "checksum does not match" in replication. These messages are expected due to update of compression algorithm and you can ignore them. These messages are informational and do not indicate any kinds of undesired behaviour.
- The setting `compile_expressions` is enabled by default. Although it has been heavily tested on variety of scenarios, if you find some undesired behaviour on your servers, you can try turning this setting off.
- Values of `UUID` type cannot be compared with integer. For example, instead of writing `uuid != 0` type `uuid != '00000000-0000-0000-0000-000000000000'`.

### New Feature

- Add Postgres-like cast operator `(::)`. E.g.: `[1, 2]::Array(UInt8)`, `0.1::Decimal(4, 4)`, `number::UInt16`. [#23871](#) ([Anton Popov](#)).
- Make big integers production ready. Add support for `UInt128` data type. Fix known issues with the `Decimal256` data type. Support big integers in dictionaries. Support `gcd/lcm` functions for big integers. Support big integers in array search and conditional functions. Support `LowCardinality(UUID)`. Support big integers in `generateRandom` table function and `clickhouse-obfuscator`. Fix error with returning `UUID` from scalar subqueries. This fixes [#7834](#). This fixes [#23936](#). This fixes [#4176](#). This fixes [#24018](#). Backward incompatible change: values of `UUID` type cannot be compared with integer. For example, instead of writing `uuid != 0` type `uuid != '00000000-0000-0000-0000-000000000000'`. [#23631](#) ([alexey-milovidov](#)).
- Support `Array` data type for inserting and selecting data in `Arrow`, `Parquet` and `ORC` formats. [#21770](#) ([taylor12805](#)).
- Implement table comments. Closes [#23225](#). [#23548](#) ([flynn](#)).
- Support creating dictionaries with DDL queries in `clickhouse-local`. Closes [#22354](#). Added support for `DETACH DICTIONARY PERMANENTLY`. Added support for `EXCHANGE DICTIONARIES` for `Atomic` database engine. Added support for moving dictionaries between databases using `RENAME DICTIONARY`. [#23436](#) ([Maksim Kita](#)).
- Add aggregate function `uniqTheta` to support `Theta Sketch` in ClickHouse. [#23894](#). [#22609](#) ([Ping Yu](#)).
- Add function `splitByRegexp`. [#24077](#) ([abel-cheng](#)).
- Add function `arrayProduct` which accept an array as the parameter, and return the product of all the elements in array. Closes [#21613](#). [#23782](#) ([Maksim Kita](#)).
- Add `thread_name` column in `system.stack_trace`. This closes [#23256](#). [#24124](#) ([abel-cheng](#)).
- If `insert_null_as_default = 1`, insert default values instead of NULL in `INSERT ... SELECT` and `INSERT ... SELECT ... UNION ALL ...` queries. Closes [#22832](#). [#23524](#) ([Kseniia Sumarokova](#)).
- Add support for progress indication in `clickhouse-local` with `--progress` option. [#23196](#) ([Egor Savin](#)).
- Add support for HTTP compression (determined by Content-Encoding HTTP header) in `http` dictionary source. This fixes [#8912](#). [#23946](#) ([FArthur-cmd](#)).

- Added `SYSTEM QUERY RELOAD MODEL`, `SYSTEM QUERY RELOAD MODELS`. Closes #18722. #23182 (Maksim Kita).
- Add setting `json` (boolean, 0 by default) for `EXPLAIN PLAN` query. When enabled, query output will be a single JSON row. It is recommended to use `TSVRaw` format to avoid unnecessary escaping. #23082 (Nikolai Kochetov).
- Add setting `indexes` (boolean, disabled by default) to `EXPLAIN PIPELINE` query. When enabled, shows used indexes, number of filtered parts and granules for every index applied. Supported for `MergeTree*` tables. #22352 (Nikolai Kochetov).
- LDAP: implemented user DN detection functionality to use when mapping Active Directory groups to ClickHouse roles. #22228 (Denis Glazachev).
- New aggregate function `deltaSumTimestamp` for summing the difference between consecutive rows while maintaining ordering during merge by storing timestamps. #21888 (Russ Frank).
- Added less secure IMDS credentials provider for S3 which works under docker correctly. #21852 (Vladimir Chebotarev).
- Add back `indexHint` function. This is for #21238. This reverts #9542. This fixes #9540. #21304 (Amos Bird).

## Experimental Feature

- Add `PROJECTION` support for `MergeTree*` tables. #20202 (Amos Bird).

## Performance Improvement

- Enable `compile_expressions` setting by default. When this setting enabled, compositions of simple functions and operators will be compiled to native code with LLVM at runtime. #8482 (Maksim Kita, alexey-milovidov). Note: if you feel in trouble, turn this option off.
- Update `re2` library. Performance of regular expressions matching is improved. Also this PR adds compatibility with gcc-11. #24196 (Raúl Marín).
- ORC input format reading by stripe instead of reading entire table into memory by once which is cost memory when file size is huge. #23102 (Chao Ma).
- Fusion of aggregate functions `sum`, `count` and `avg` in a query into single aggregate function. The optimization is controlled with the `optimize_fuse_sum_count_avg` setting. This is implemented with a new aggregate function `sumCount`. This function returns a tuple of two fields: `sum` and `count`. #21337 (hexiaoting).
- Update `zstd` to v1.5.0. The performance of compression is improved for single digits percentage. #24135 (Raúl Marín). Note: you may get messages about "checksum does not match" in replication. These messages are expected due to update of compression algorithm and you can ignore them.
- Improved performance of `Buffer` tables: do not acquire lock for `total_bytes/total_rows` for `Buffer` engine. #24066 (Azat Khuzhin).
- Preallocate support for hashed/sparse\_hashed dictionaries is returned. #23979 (Azat Khuzhin).
- Enable `async_socket_for_remote` by default (lower amount of threads in querying Distributed tables with large fanout). #23683 (Nikolai Kochetov).

## Improvement

- Add `_partition_value` virtual column to `MergeTree` table family. It can be used to prune partition in a deterministic way. It's needed to implement partition matcher for mutations. #23673 (Amos Bird).

- Added `region` parameter for S3 storage and disk. [#23846](#) ([Vladimir Chebotarev](#)).
- Allow configuring different log levels for different logging channels. Closes [#19569](#). [#23857](#) ([filimonov](#)).
- Keep default timezone on `DateTime` operations if it was not provided explicitly. For example, if you add one second to a value of `DateTime` type without timezone it will remain `DateTime` without timezone. In previous versions the value of default timezone was placed to the returned data type explicitly so it becomes `DateTime('something')`. This closes [#4854](#). [#23392](#) ([alexey-milovidov](#)).
- Allow user to specify empty string instead of database name for `MySQL` storage. Default database will be used for queries. In previous versions it was working for `SELECT` queries and not support for `INSERT` was also added. This closes [#19281](#). This can be useful working with `Sphinx` or other MySQL-compatible foreign databases. [#23319](#) ([alexey-milovidov](#)).
- Fixed `quantile(s)TDigest`. Added special handling of singleton centroids according to tdunning/t-digest 3.2+. Also a bug with over-compression of centroids in implementation of earlier version of the algorithm was fixed. [#23314](#) ([Vladimir Chebotarev](#)).
- Function `now64` now supports optional timezone argument. [#24091](#) ([Vasily Nemkov](#)).
- Fix the case when a progress bar in interactive mode in `clickhouse-client` that appear in the middle of the data may rewrite some parts of visible data in terminal. This closes [#19283](#). [#23050](#) ([alexey-milovidov](#)).
- Fix crash when memory allocation fails in `simdjson`. <https://github.com/simdjson/simdjson/pull/1567> . Mark as improvement because it's a very rare bug. [#24147](#) ([Amos Bird](#)).
- Preserve dictionaries until storage shutdown (this will avoid possible `external dictionary 'DICT' not found` errors at server shutdown during final flush of the `Buffer` engine). [#24068](#) ([Azat Khuzhin](#)).
- Flush `Buffer` tables before shutting down tables (within one database), to avoid discarding blocks due to underlying table had been already detached (and `Destination table default.a_data_01870 doesn't exist. Block of data is discarded` error in the log). [#24067](#) ([Azat Khuzhin](#)).
- Now `prefer_column_name_to_alias = 1` will also favor column names for `group by`, `having` and `order by`. This fixes [#23882](#). [#24022](#) ([Amos Bird](#)).
- Add support for `ORDER BY WITH FILL` with `DateTime64`. [#24016](#) ([kevin wan](#)).
- Enable `DateTime64` to be a version column in `ReplacingMergeTree`. [#23992](#) ([kevin wan](#)).
- Log information about OS name, kernel version and CPU architecture on server startup. [#23988](#) ([Azat Khuzhin](#)).
- Support specifying table schema for `postgresql` dictionary source. Closes [#23958](#). [#23980](#) ([Kseniia Sumarokova](#)).
- Add hints for names of `Enum` elements (suggest names in case of typos). Closes [#17112](#). [#23919](#) ([flynn](#)).
- Measure found rate (the percentage for which the value was found) for dictionaries (see `found_rate` in `system.dictionaries`). [#23916](#) ([Azat Khuzhin](#)).
- Allow to add specific queue settings via table setting `rabbitmq_queue_settings_list`. (Closes [#23737](#) and [#23918](#)). Allow user to control all RabbitMQ setup: if table setting `rabbitmq_queue_consume` is set to `1` - RabbitMQ table engine will only connect to specified queue and will not perform any RabbitMQ consumer-side setup like declaring exchange, queues, bindings. (Closes [#21757](#)). Add proper cleanup when RabbitMQ table is dropped - delete queues, which the table has declared and all bound exchanges - if they were created by the table. [#23887](#) ([Kseniia Sumarokova](#)).

- Add `broken_data_files/broken_data_compressed_bytes` into `system.distribution_queue`. Add metric for number of files for asynchronous insertion into Distributed tables that has been marked as broken (`BrokenDistributedFilesToInsert`). [#23885](#) ([Azat Khuzhin](#)).
- Querying `system.tables` does not go to ZooKeeper anymore. [#23793](#) ([Fuwang Hu](#)).
- Respect `lock_acquire_timeout_for_background_operations` for `OPTIMIZE` queries. [#23623](#) ([Azat Khuzhin](#)).
- Possibility to change S3 disk settings in runtime via new `SYSTEM RESTART DISK` SQL command. [#23429](#) ([Pavel Kovalenko](#)).
- If user applied a misconfiguration by mistakenly setting `max_distributed_connections` to value zero, every query to a `Distributed` table will throw exception with a message containing "logical error". But it's really an expected behaviour, not a logical error, so the exception message was slightly incorrect. It also triggered checks in our CI environment that ensures that no logical errors ever happen. Instead we will treat `max_distributed_connections` misconfigured to zero as the minimum possible value (one). [#23348](#) ([Azat Khuzhin](#)).
- Disable `min_bytes_to_use_mmap_io` by default. [#23322](#) ([Azat Khuzhin](#)).
- Support `LowCardinality` nullability with `join_use_nulls`, close [#15101](#). [#23237](#) ([vdimir](#)).
- Added possibility to restore `MergeTree` parts to `detached` directory for S3 disk. [#23112](#) ([Pavel Kovalenko](#)).
- Retries on HTTP connection drops in S3. [#22988](#) ([Vladimir Chebotarev](#)).
- Add settings `external_storage_max_read_rows` and `external_storage_max_write_rows` for MySQL table engine, dictionary source and MaterializeMySQL minor data fetches. [#22697](#) ([TCeason](#)).
- MaterializeMySQL (experimental feature): Previously, MySQL 5.7.9 was not supported due to SQL incompatibility. Now leave MySQL parameter verification to the MaterializeMySQL. [#23413](#) ([TCeason](#)).
- Enable reading of subcolumns for distributed tables. [#24472](#) ([Anton Popov](#)).
- Fix usage of tuples in `CREATE .. AS SELECT` queries. [#24464](#) ([Anton Popov](#)).
- Support for `Parquet` format in `Kafka` tables. [#23412](#) ([Chao Ma](#)).

## Bug Fix

- Use old modulo function version when used in partition key and primary key. Closes [#23508](#). [#24157](#) ([Kseniia Sumarokova](#)). It was a source of backward incompatibility in previous releases.
- Fixed the behavior when query `SYSTEM RESTART REPLICA` or `SYSTEM SYNC REPLICA` is being processed infinitely. This was detected on server with extremely little amount of RAM. [#24457](#) ([Nikita Mikhaylov](#)).
- Fix incorrect monotonicity of `toWeek` function. This fixes [#24422](#). This bug was introduced in [#5212](#), and was exposed later by smarter partition pruner. [#24446](#) ([Amos Bird](#)).
- Fix drop partition with intersect fake parts. In rare cases there might be parts with mutation version greater than current block number. [#24321](#) ([Amos Bird](#)).
- Fixed a bug in moving Materialized View from Ordinary to Atomic database (`RENAME TABLE` query). Now inner table is moved to new database together with Materialized View. Fixes [#23926](#). [#24309](#) ([tavplubix](#)).
- Allow empty HTTP headers in client requests. Fixes [#23901](#). [#24285](#) ([Ivan](#)).
- Set `max_threads = 1` to fix mutation fail of `Memory` tables. Closes [#24274](#). [#24275](#) ([flynn](#)).

- Fix typo in implementation of `Memory` tables, this bug was introduced at #15127. Closes #24192. #24193 (张中南).
- Fix abnormal server termination due to `HDFS` becoming not accessible during query execution. Closes #24117. #24191 (Ksenia Sumarokova).
- Fix crash on updating of `Nested` column with const condition. #24183 (hexiaoting).
- Fix race condition which could happen in RBAC under a heavy load. This PR fixes #24090, #24134, #24176 (Vitaly Baranov).
- Fix a rare bug that could lead to a partially initialized table that can serve write requests (insert/alter/so on). Now such tables will be in readonly mode. #24122 (alesapin).
- Fix an issue: `EXPLAIN PIPELINE` with `SELECT xxx FINAL` showed a wrong pipeline. (hexiaoting).
- Fixed using const `DateTime` value vs `DateTime64` column in `WHERE`. #24100 (Vasily Nemkov).
- Fix crash in merge JOIN, closes #24010. #24013 (vdimir).
- Some `ALTER PARTITION` queries might cause `Part A` intersects previous part `B` and Unexpected merged part `C` intersecting drop range `D` errors in replication queue. It's fixed. Fixes #23296. #23997 (tavplubix).
- Fix SIGSEGV for external GROUP BY and overflow row (i.e. queries like `SELECT FROM GROUP BY WITH TOTALS SETTINGS max_bytes_before_external_group_by>0, max_rows_to_group_by>0, group_by_overflow_mode='any', totals_mode='before_having'`). #23962 (Azat Khuzhin).
- Fix keys metrics accounting for `CACHE` dictionary with duplicates in the source (leads to `DictCacheKeysRequestedMiss` overflows). #23929 (Azat Khuzhin).
- Fix implementation of connection pool of PostgreSQL engine. Closes #23897. #23909 (Ksenia Sumarokova).
- Fix `distributed_group_by_no_merge = 2` with GROUP BY and aggregate function wrapped into regular function (had been broken in #23546). Throw exception in case of someone trying to use `distributed_group_by_no_merge = 2` with window functions. Disable `optimize_distributed_group_by_sharding_key` for queries with window functions. #23906 (Azat Khuzhin).
- A fix for `s3` table function: better handling of HTTP errors. Response bodies of HTTP errors were being ignored earlier. #23844 (Vladimir Chebotarev).
- A fix for `s3` table function: better handling of URI's. Fixed an incompatibility with URLs containing + symbol, data with such keys could not be read previously. #23822 (Vladimir Chebotarev).
- Fix error `Can't initialize pipeline with empty pipe` for queries with `GLOBAL IN/JOIN` and `use_hedged_requests`. Fixes #23431. #23805 (Nikolai Kochetov).
- Fix `CLEAR COLUMN` does not work when it is referenced by materialized view. Close #23764. #23781 (flynn).
- Fix heap use after free when reading from HDFS if `Values` format is used. #23761 (Ksenia Sumarokova).
- Avoid possible "Cannot schedule a task" error (in case some exception had been occurred) on INSERT into Distributed. #23744 (Azat Khuzhin).
- Fixed a bug in recovery of staled ReplicatedMergeTree replica. Some metadata updates could be ignored by staled replica if `ALTER` query was executed during downtime of the replica. #23742 (tavplubix).
- Fix a bug with `Join` and `WITH TOTALS`, close #17718. #23549 (vdimir).

- Fix possible `Block` structure mismatch error for queries with `UNION` which could possibly happen after filter-pushdown optimization. Fixes #23029. #23359 (Nikolai Kochetov).
- Add type conversion when the setting `optimize_skip_unused_shards_rewrite_in` is enabled. This fixes MSan report. #23219 (Azat Khuzhin).
- Add a missing check when updating nested subcolumns, close issue: #22353. #22503 (hexiaoting).

## Build/Testing/Packaging Improvement

- Support building on Illumos. #24144. Adds support for building on Solaris-derived operating systems. #23746 (bnaecker).
- Add more benchmarks for hash tables, including the Swiss Table from Google (that appeared to be slower than ClickHouse hash map in our specific usage scenario). #24111 (Maksim Kita).
- Update librdkafka 1.6.0-RC3 to 1.6.1. #23874 (filimonov).
- Always enable `asynchronous-unwind-tables` explicitly. It may fix query profiler on AArch64. #23602 (alexey-milovidov).
- Avoid possible build dependency on locale and filesystem order. This allows reproducible builds. #23600 (alexey-milovidov).
- Remove a source of nondeterminism from build. Now builds at different point of time will produce byte-identical binaries. Partially addressed #22113. #23559 (alexey-milovidov).
- Add simple tool for benchmarking (Zoo)Keeper. #23038 (alesapin).

## ClickHouse release 21.5, 2021-05-20

### Backward Incompatible Change

- Change comparison of integers and floating point numbers when integer is not exactly representable in the floating point data type. In new version comparison will return false as the rounding error will occur. Example: `9223372036854775808.0 != 9223372036854775808`, because the number `9223372036854775808` is not representable as floating point number exactly (and `9223372036854775808.0` is rounded to `9223372036854776000.0`). But in previous version the comparison will return as the numbers are equal, because if the floating point number `9223372036854776000.0` get converted back to `UInt64`, it will yield `9223372036854775808`. For the reference, the Python programming language also treats these numbers as equal. But this behaviour was dependend on CPU model (different results on AMD64 and AArch64 for some out-of-range numbers), so we make the comparison more precise. It will treat int and float numbers equal only if int is represented in floating point type exactly. #22595 (alexey-milovidov).
- Remove support for `argMin` and `argMax` for single `Tuple` argument. The code was not memory-safe. The feature was added by mistake and it is confusing for people. These functions can be reintroduced under different names later. This fixes #22384 and reverts #17359. #23393 (alexey-milovidov).

### New Feature

- Added functions `dictGetChildren(dictionary, key)`, `dictGetDescendants(dictionary, key, level)`. Function `dictGetChildren` return all children as an array of indexes. It is a inverse transformation for `dictGetHierarchy`. Function `dictGetDescendants` return all descendants as if `dictGetChildren` was applied `level` times recursively. Zero `level` value is equivalent to infinity. Improved performance of `dictGetHierarchy`, `dictIsIn` functions. Closes #14656. #22096 (Maksim Kita).
- Added function `dictGetOrNull`. It works like `dictGet`, but return `Null` in case key was not found in dictionary. Closes #22375. #22413 (Maksim Kita).

- Added a table function `s3Cluster`, which allows to process files from `s3` in parallel on every node of a specified cluster. [#22012 \(Nikita Mikhaylov\)](#).
- Added support for replicas and shards in MySQL/PostgreSQL table engine / table function. You can write `SELECT * FROM mysql('host{1,2}-{1|2}', ...)`. Closes [#20969](#). [#22217 \(Ksenia Sumarokova\)](#).
- Added `ALTER TABLE ... FETCH PART ...query`. It's similar to `FETCH PARTITION`, but fetches only one part. [#22706 \(turbo jason\)](#).
- Added a setting `max_distributed_depth` that limits the depth of recursive queries to `Distributed` tables. Closes [#20229](#). [#21942 \(flynn\)](#).

## Performance Improvement

- Improved performance of `intDiv` by dynamic dispatch for AVX2. This closes [#22314](#). [#23000 \(alexey-milovidov\)](#).
- Improved performance of reading from `ArrowStream` input format for sources other than local file (e.g. URL). [#22673 \(nvartolomei\)](#).
- Disabled compression by default when interacting with localhost (with `clickhouse-client` or server to server with distributed queries) via native protocol. It may improve performance of some import/export operations. This closes [#22234](#). [#22237 \(alexey-milovidov\)](#).
- Exclude values that does not belong to the shard from right part of IN section for distributed queries (under `optimize_skip_unused_shards_rewrite_in`, enabled by default, since it still requires `optimize_skip_unused_shards`). [#21511 \(Azat Khuzhin\)](#).
- Improved performance of reading a subset of columns with File-like table engine and column-oriented format like Parquet, Arrow or ORC. This closes [#issue:20129](#). [#21302 \(keenwolf\)](#).
- Allow to move more conditions to `PREWHERE` as it was before version 21.1 (adjustment of internal heuristics). Insufficient number of moved conditions could lead to worse performance. [#23397 \(Anton Popov\)](#).
- Improved performance of ODBC connections and fixed all the outstanding issues from the backlog. Using `nanodbc` library instead of `Poco::ODBC`. Closes [#9678](#). Add support for `DateTime64` and `Decimal*` for ODBC table engine. Closes [#21961](#). Fixed issue with cyrillic text being truncated. Closes [#16246](#). Added connection pools for odbc bridge. [#21972 \(Ksenia Sumarokova\)](#).

## Improvement

- Increase `max_uri_size` (the maximum size of URL in HTTP interface) to 1 MiB by default. This closes [#21197](#). [#22997 \(alexey-milovidov\)](#).
- Set `background_fetches_pool_size` to 8 that is better for production usage with frequent small insertions or slow ZooKeeper cluster. [#22945 \(alexey-milovidov\)](#).
- FlatDictionary added `initial_array_size`, `max_array_size` options. [#22521 \(Maksim Kita\)](#).
- Add new setting `non_replicated_deduplication_window` for non-replicated MergeTree inserts deduplication. [#22514 \(alesapin\)](#).
- Update paths to the `CatBoost` model configs in config reloading. [#22434 \(Kruglov Pavel\)](#).
- Added `Decimal256` type support in dictionaries. `Decimal256` is experimental feature. Closes [#20979](#). [#22960 \(Maksim Kita\)](#).
- Enabled `async_socket_for_remote` by default (using less amount of OS threads for distributed queries). [#23683 \(Nikolai Kochetov\)](#).

- Fixed `quantile(s)TDigest`. Added special handling of singleton centroids according to tdunning/t-digest 3.2+. Also a bug with over-compression of centroids in implementation of earlier version of the algorithm was fixed. [#23314](#) ([Vladimir Chebotarev](#)).
- Make function name `unhex` case insensitive for compatibility with MySQL. [#23229](#) ([alexey-milovidov](#)).
- Implement functions `arrayHasAny`, `arrayHasAll`, `has`, `indexOf`, `countEqual` for generic case when types of array elements are different. In previous versions the functions `arrayHasAny`, `arrayHasAll` returned false and `has`, `indexOf`, `countEqual` thrown exception. Also add support for `Decimal` and big integer types in functions `has` and similar. This closes [#20272](#). [#23044](#) ([alexey-milovidov](#)).
- Raised the threshold on max number of matches in result of the function `extractAllGroupsHorizontal`. [#23036](#) ([Vasily Nemkov](#)).
- Do not perform `optimize_skip_unused_shards` for cluster with one node. [#22999](#) ([Azat Khuzhin](#)).
- Added ability to run clickhouse-keeper (experimental drop-in replacement to ZooKeeper) with SSL. Config settings `keeper_server.tcp_port_secure` can be used for secure interaction between client and keeper-server. `keeper_server.raft_configuration.secure` can be used to enable internal secure communication between nodes. [#22992](#) ([alesapin](#)).
- Added ability to flush buffer only in background for `Buffer` tables. [#22986](#) ([Azat Khuzhin](#)).
- When selecting from MergeTree table with NULL in WHERE condition, in rare cases, exception was thrown. This closes [#20019](#). [#22978](#) ([alexey-milovidov](#)).
- Fix error handling in Poco HTTP Client for AWS. [#22973](#) ([kreuzerkrieg](#)).
- Respect `max_part_removal_threads` for `ReplicatedMergeTree`. [#22971](#) ([Azat Khuzhin](#)).
- Fix obscure corner case of MergeTree settings `inactive_parts_to_throw_insert = 0` with `inactive_parts_to_delay_insert > 0`. [#22947](#) ([Azat Khuzhin](#)).
- `dateDiff` now works with `DateTime64` arguments (even for values outside of `DateTime` range) [#22931](#) ([Vasily Nemkov](#)).
- MaterializeMySQL (experimental feature): added an ability to replicate MySQL databases containing views without failing. This is accomplished by ignoring the views. [#22760](#) ([Christian](#)).
- Allow RBAC row policy via postgresql protocol. Closes [#22658](#). PostgreSQL protocol is enabled in configuration by default. [#22755](#) ([Kseniia Sumarokova](#)).
- Add metric to track how much time is spend during waiting for Buffer layer lock. [#22725](#) ([Azat Khuzhin](#)).
- Allow to use CTE in VIEW definition. This closes [#22491](#). [#22657](#) ([Amos Bird](#)).
- Clear the rest of the screen and show cursor in `clickhouse-client` if previous program has left garbage in terminal. This closes [#16518](#). [#22634](#) ([alexey-milovidov](#)).
- Make `round` function to behave consistently on non-x86\_64 platforms. Rounding half to nearest even (Banker's rounding) is used. [#22582](#) ([alexey-milovidov](#)).
- Correctly check structure of blocks of data that are sending by Distributed tables. [#22325](#) ([Azat Khuzhin](#)).
- Allow publishing Kafka errors to a virtual column of Kafka engine, controlled by the `kafka_handle_error_mode` setting. [#21850](#) ([fastio](#)).
- Add aliases `simpleJSONExtract/simpleJSONHas` to `visitParam/visitParamExtract{UInt, Int, Bool, Float, Raw, String}`. Fixes [#21383](#). [#21519](#) ([fastio](#)).

- Add `clickhouse-library-bridge` for library dictionary source. Closes #9502. #21509 (Ksenia Sumarokova).
- Forbid to drop a column if it's referenced by materialized view. Closes #21164. #21303 (flynn).
- Support dynamic interserver credentials (rotating credentials without downtime). #14113 (johnskopis).
- Add support for Kafka storage with `Arrow` and `ArrowStream` format messages. #23415 (Chao Ma).
- Fixed missing semicolon in exception message. The user may find this exception message unpleasant to read. #23208 (alexey-milovidov).
- Fixed missing whitespace in some exception messages about `LowCardinality` type. #23207 (alexey-milovidov).
- Some values were formatted with alignment in center in table cells in `Markdown` format. Not anymore. #23096 (alexey-milovidov).
- Remove non-essential details from suggestions in `clickhouse-client`. This closes #22158. #23040 (alexey-milovidov).
- Correct calculation of `bytes_allocated` field in `system.dictionaries` for `sparse_hashed` dictionaries. #22867 (Azat Khuzhin).
- Fixed approximate total rows accounting for reverse reading from `MergeTree`. #22726 (Azat Khuzhin).
- Fix the case when it was possible to configure dictionary with `clickhouse` source that was looking to itself that leads to infinite loop. Closes #14314. #22479 (Maksim Kita).

## Bug Fix

- Multiple fixes for hedged requests. Fixed an error `Can't initialize pipeline with empty pipe` for queries with `GLOBAL IN/JOIN` when the setting `use_hedged_requests` is enabled. Fixes #23431. #23805 (Nikolai Kochetov). Fixed a race condition in hedged connections which leads to crash. This fixes #22161. #22443 (Kruglov Pavel). Fix possible crash in case if `unknown` packet was received from remote query (with `async_socket_for_remote` enabled). Fixes #21167. #23309 (Nikolai Kochetov).
- Fixed the behavior when disabling `input_format_with_names_use_header` setting discards all the input with `CSVWithNames` format. This fixes #22406. #23202 (Nikita Mikhaylov).
- Fixed remote JDBC bridge timeout connection issue. Closes #9609. #23771 (Maksim Kita, alexey-milovidov).
- Fix the logic of initial load of `complex_key_hashed` if `update_field` is specified. Closes #23800. #23824 (Maksim Kita).
- Fixed crash when `PREWHERE` and row policy filter are both in effect with empty result. #23763 (Amos Bird).
- Avoid possible "Cannot schedule a task" error (in case some exception had been occurred) on `INSERT` into `Distributed`. #23744 (Azat Khuzhin).
- Added an exception in case of completely the same values in both samples in aggregate function `mannWhitneyUTest`. This fixes #23646. #23654 (Nikita Mikhaylov).
- Fixed server fault when inserting data through HTTP caused an exception. This fixes #23512. #23643 (Nikita Mikhaylov).
- Fixed misinterpretation of some `LIKE` expressions with escape sequences. #23610 (alexey-milovidov).
- Fixed restart / stop command hanging. Closes #20214. #23552 (filimonov).

- Fixed `COLUMNS` matcher in case of multiple JOINS in select query. Closes #22736. #23501 (Maksim Kita).
- Fixed a crash when modifying column's default value when a column itself is used as `ReplacingMergeTree`'s parameter. #23483 (hexiaoting).
- Fixed corner cases in vertical merges with `ReplacingMergeTree`. In rare cases they could lead to fails of merges with exceptions like `Incomplete granules are not allowed while blocks are granules size` #23459 (Anton Popov).
- Fixed bug that does not allow cast from empty array literal, to array with dimensions greater than 1, e.g. `CAST([] AS Array(Array(String)))`. Closes #14476. #23456 (Maksim Kita).
- Fixed a bug when `deltaSum` aggregate function produced incorrect result after resetting the counter. #23437 (Russ Frank).
- Fixed `Cannot unlink file` error on unsuccessful creation of `ReplicatedMergeTree` table with multidisk configuration. This closes #21755. #23433 (tavplubix).
- Fixed incompatible constant expression generation during partition pruning based on virtual columns. This fixes [https://github.com/ClickHouse/ClickHouse/pull/21401#discussion\\_r611888913](https://github.com/ClickHouse/ClickHouse/pull/21401#discussion_r611888913). #23366 (Amos Bird).
- Fixed a crash when setting `join_algorithm` is set to 'auto' and Join is performed with a Dictionary. Close #23002. #23312 (Vladimir).
- Don't relax NOT conditions during partition pruning. This fixes #23305 and #21539. #23310 (Amos Bird).
- Fixed very rare race condition on background cleanup of old blocks. It might cause a block not to be deduplicated if it's too close to the end of deduplication window. #23301 (tavplubix).
- Fixed very rare (distributed) race condition between creation and removal of `ReplicatedMergeTree` tables. It might cause exceptions like `node doesn't exist` on attempt to create replicated table. Fixes #21419. #23294 (tavplubix).
- Fixed simple key dictionary from DDL creation if primary key is not first attribute. Fixes #23236. #23262 (Maksim Kita).
- Fixed reading from ODBC when there are many long column names in a table. Closes #8853. #23215 (Kseniia Sumarokova).
- MaterializeMySQL (experimental feature): fixed `Not found column` error when selecting from `MaterializeMySQL` with condition on key column. Fixes #22432. #23200 (tavplubix).
- Correct aliases handling if subquery was optimized to constant. Fixes #22924. Fixes #10401. #23191 (Maksim Kita).
- Server might fail to start if `data_type_default_nullable` setting is enabled in default profile, it's fixed. Fixes #22573. #23185 (tavplubix).
- Fixed a crash on shutdown which happened because of wrong accounting of current connections. #23154 (Vitaly Baranov).
- Fixed `Table .inner_id... doesn't exist` error when selecting from Materialized View after detaching it from Atomic database and attaching back. #23047 (tavplubix).
- Fix error `Cannot find column` in ActionsDAG result which may happen if subquery uses `untuple`. Fixes #22290. #22991 (Nikolai Kochetov).
- Fix usage of constant columns of type `Map` with nullable values. #22939 (Anton Popov).

- fixed `formatDateTime()` on `DateTime64` and "%C" format specifier fixed `toDateTime64()` for large values and non-zero scale. #22937 ([Vasily Nemkov](#)).
- Fixed a crash when using `mannWhitneyUTest` and `rankCorr` with window functions. This fixes #22728. #22876 ([Nikita Mikhaylov](#)).
- LIVE VIEW (experimental feature): fixed possible hanging in concurrent DROP/CREATE of TEMPORARY LIVE VIEW in `TemporaryLiveViewCleaner`, see. #22858 ([Vitaly Baranov](#)).
- Fixed pushdown of `HAVING` in case, when filter column is used in aggregation. #22763 ([Anton Popov](#)).
- Fixed possible hangs in Zookeeper requests in case of OOM exception. Fixes #22438. #22684 ([Nikolai Kochetov](#)).
- Fixed wait for mutations on several replicas for ReplicatedMergeTree table engines. Previously, mutation/alter query may finish before mutation actually executed on other replicas. #22669 ([alesapin](#)).
- Fixed exception for Log with nested types without columns in the SELECT clause. #22654 ([Azat Khuzhin](#)).
- Fix unlimited wait for auxiliary AWS requests. #22594 ([Vladimir Chebotarev](#)).
- Fixed a crash when client closes connection very early #22579. #22591 ([nvartolomei](#)).
- Map data type (experimental feature): fixed an incorrect formatting of function `map` in distributed queries. #22588 ([foolchi](#)).
- Fixed deserialization of empty string without newline at end of TSV format. This closes #20244. Possible workaround without version update: set `input_format_null_as_default` to zero. It was zero in old versions. #22527 ([alexey-milovidov](#)).
- Fixed wrong cast of a column of `LowCardinality` type in Merge Join algorithm. Close #22386, close #22388. #22510 ([Vladimir](#)).
- Buffer overflow (on read) was possible in `tokenbf_v1` full text index. The excessive bytes are not used but the read operation may lead to crash in rare cases. This closes #19233. #22421 ([alexey-milovidov](#)).
- Do not limit HTTP chunk size. Fixes #21907. #22322 ([Ivan](#)).
- Fixed a bug, which leads to underaggregation of data in case of enabled `optimize_aggregation_in_order` and many parts in table. Slightly improve performance of aggregation with enabled `optimize_aggregation_in_order`. #21889 ([Anton Popov](#)).
- Check if table function view is used as a column. This complements #20350. #21465 ([Amos Bird](#)).
- Fix "unknown column" error for tables with `Merge` engine in queris with `JOIN` and aggregation. Closes #18368, close #22226. #21370 ([Vladimir](#)).
- Fixed name clashes in pushdown optimization. It caused incorrect `WHERE` filtration after FULL JOIN. Close #20497. #20622 ([Vladimir](#)).
- Fixed very rare bug when quorum insert with `quorum_parallel=1` is not really "quorum" because of deduplication. #18215 ([filimonov](#) - reported, [alesapin](#) - fixed).

## Build/Testing/Packaging Improvement

- Run stateless tests in parallel in CI. #22300 ([alesapin](#)).
- Simplify debian packages. This fixes #21698. #22976 ([alexey-milovidov](#)).
- Added support for ClickHouse build on Apple M1. #21639 ([changvzb](#)).

- Fixed ClickHouse Keeper build for MacOS. #22860 ([alesapin](#)).
- Fixed some tests on AArch64 platform. #22596 ([alexey-milovidov](#)).
- Added function alignment for possibly better performance. #21431 ([Danila Kutenin](#)).
- Adjust some tests to output identical results on amd64 and aarch64 (qemu). The result was depending on implementation specific CPU behaviour. #22590 ([alexey-milovidov](#)).
- Allow query profiling only on x86\_64. See #15174 and #15638. This closes #15638. #22580 ([alexey-milovidov](#)).
- Allow building with unbundled xz (lzma) using `USE_INTERNAL_XZ_LIBRARY=OFF` CMake option. #22571 ([Kfir Itzhak](#)).
- Enable bundled `openldap` on `ppc64le` #22487 ([Kfir Itzhak](#)).
- Disable incompatible libraries (platform specific typically) on `ppc64le` #22475 ([Kfir Itzhak](#)).
- Add Jepsen test in CI for clickhouse Keeper. #22373 ([alesapin](#)).
- Build `jemalloc` with support for `heap profiling`. #22834 ([nvartolomei](#)).
- Avoid UB in `*Log` engines for rwlock unlock due to unlock from another thread. #22583 ([Azat Khuzhin](#)).
- Fixed UB by unlocking the rwlock of the TinyLog from the same thread. #22560 ([Azat Khuzhin](#)).

## ClickHouse release 21.4

### ClickHouse release 21.4.1 2021-04-12

#### Backward Incompatible Change

- The `toStartOfIntervalFunction` will align hour intervals to the midnight (in previous versions they were aligned to the start of unix epoch). For example, `toStartOfInterval(x, INTERVAL 11 HOUR)` will split every day into three intervals: `00:00:00..10:59:59`, `11:00:00..21:59:59` and `22:00:00..23:59:59`. This behaviour is more suited for practical needs. This closes #9510. #22060 ([alexey-milovidov](#)).
- `Age` and `Precision` in graphite rollup configs should increase from retention to retention. Now it's checked and the wrong config raises an exception. #21496 ([Mikhail f. Shiryaev](#)).
- Fix `cutToFirstSignificantSubdomainCustom()/firstSignificantSubdomainCustom()` returning wrong result for 3+ level domains present in custom top-level domain list. For input domains matching these custom top-level domains, the third-level domain was considered to be the first significant one. This is now fixed. This change may introduce incompatibility if the function is used in e.g. the sharding key. #21946 ([Azat Khuzhin](#)).
- Column `keys` in table `system.dictionaries` was replaced to columns `key.names` and `key.types`. Columns `key.names`, `key.types`, `attribute.names`, `attribute.types` from `system.dictionaries` table does not require dictionary to be loaded. #21884 ([Maksim Kita](#)).
- Now replicas that are processing the `ALTER TABLE ATTACH PART[ITION]` command search in their `detached` folders before fetching the data from other replicas. As an implementation detail, a new command `ATTACH_PART` is introduced in the replicated log. Parts are searched and compared by their checksums. #18978 ([Mike Kot](#)). **Note:**
- `ATTACH PART[ITION]` queries may not work during cluster upgrade.
- It's not possible to rollback to older ClickHouse version after executing `ALTER ... ATTACH` query in new version as the old servers would fail to pass the `ATTACH_PART` entry in the replicated log.

- In this version, empty `<remote_url_allow_hosts></remote_url_allow_hosts>` will block all access to remote hosts while in previous versions it did nothing. If you want to keep old behaviour and you have empty `remote_url_allow_hosts` element in configuration file, remove it. #20058 (Vladimir Chebotarev).

## New Feature

- Extended range of `DateTime64` to support dates from year 1925 to 2283. Improved support of `DateTime` around zero date (1970-01-01). #9404 (alexey-milovidov, Vasily Nemkov). Not every time and date functions are working for extended range of dates.
- Added support of Kerberos authentication for preconfigured users and HTTP requests (GSS-SPNEGO). #14995 (Denis Glazachev).
- Add `prefer_column_name_to_alias` setting to use original column names instead of aliases. It is needed to be more compatible with common databases' aliasing rules. This is for #9715 and #9887. #22044 (Amos Bird).
- Added functions `dictGetChildren(dictionary, key)`, `dictGetDescendants(dictionary, key, level)`. Function `dictGetChildren` return all children as an array of indexes. It is a inverse transformation for `dictGetHierarchy`. Function `dictGetDescendants` return all descendants as if `dictGetChildren` was applied `level` times recursively. Zero `level` value is equivalent to infinity. Closes #14656. #22096 (Maksim Kita).
- Added `executable_pool` dictionary source. Close #14528. #21321 (Maksim Kita).
- Added table function `dictionary`. It works the same way as `Dictionary` engine. Closes #21560. #21910 (Maksim Kita).
- Support `Nullable` type for `PolygonDictionary` attribute. #21890 (Maksim Kita).
- Functions `dictGet`, `dictHas` use current database name if it is not specified for dictionaries created with DDL. Closes #21632. #21859 (Maksim Kita).
- Added function `dictGetOrNull`. It works like `dictGet`, but return `Null` in case key was not found in dictionary. Closes #22375. #22413 (Maksim Kita).
- Added async update in `ComplexKeyCache`, `SSDCache`, `SSDComplexKeyCache` dictionaries. Added support for `Nullable` type in `Cache`, `ComplexKeyCache`, `SSDCache`, `SSDComplexKeyCache` dictionaries. Added support for multiple attributes fetch with `dictGet`, `dictGetOrDefault` functions. Fixes #21517. #20595 (Maksim Kita).
- Support `dictHas` function for `RangeHashedDictionary`. Fixes #6680. #19816 (Maksim Kita).
- Add function `timezoneOf` that returns the timezone name of `DateTime` or `DateTime64` data types. This does not close #9959. Fix inconsistencies in function names: add aliases `timezone` and `timeZone` as well as `toTimezone` and `toTimeZone` and `timezoneOf` and `timeZoneOf`. #22001 (alexey-milovidov).
- Add new optional clause `GRANTEES` for `CREATE/ALTER USER` commands. It specifies users or roles which are allowed to receive grants from this user on condition this user has also all required access granted with grant option. By default `GRANTEES ANY` is used which means a user with grant option can grant to anyone. Syntax: `CREATE USER ... GRANTEES {user | role | ANY | NONE} [...] [EXCEPT {user | role} [...]]` #21641 (Vitaly Baranov).
- Add new column `slowdowns_count` to `system.clusters`. When using hedged requests, it shows how many times we switched to another replica because this replica was responding slowly. Also show actual value of `errors_count` in `system.clusters`. #21480 (Kruglov Pavel).
- Add `_partition_id` virtual column for `MergeTree*` engines. Allow to prune partitions by `_partition_id`. Add `partitionID()` function to calculate partition id string. #21401 (Amos Bird).
- Add function `isIPAddressInRange` to test if an IPv4 or IPv6 address is contained in a given CIDR network prefix. #21329 (PHO).

- Added new SQL command `ALTER TABLE 'table_name' UNFREEZE [PARTITION 'part_expr'] WITH NAME 'backup_name'`. This command is needed to properly remove 'freezed' partitions from all disks. [#21142](#) ([Pavel Kovalenko](#)).
- Supports implicit key type conversion for JOIN. [#19885](#) ([Vladimir](#)).

## Experimental Feature

- Support `RANGE OFFSET` frame (for window functions) for floating point types. Implement `lagInFrame/leadInFrame` window functions, which are analogous to `lag/lead`, but respect the window frame. They are identical when the frame is `between unbounded preceding and unbounded following`. This closes [#5485](#). [#21895](#) ([Alexander Kuzmenkov](#)).
- Zero-copy replication for `ReplicatedMergeTree` over S3 storage. [#16240](#) ([ianton-ru](#)).
- Added possibility to migrate existing S3 disk to the schema with backup-restore capabilities. [#22070](#) ([Pavel Kovalenko](#)).

## Performance Improvement

- Supported parallel formatting in `clickhouse-local` and everywhere else. [#21630](#) ([Nikita Mikhaylov](#)).
- Support parallel parsing for `CSVWithNames` and `TSVWithNames` formats. This closes [#21085](#). [#21149](#) ([Nikita Mikhaylov](#)).
- Enable read with mmap IO for file ranges from 64 MiB (the settings `min_bytes_to_use_mmap_io`). It may lead to moderate performance improvement. [#22326](#) ([alexey-milovidov](#)).
- Add cache for files read with `min_bytes_to_use_mmap_io` setting. It makes significant (2x and more) performance improvement when the value of the setting is small by avoiding frequent mmap/munmap calls and the consequent page faults. Note that mmap IO has major drawbacks that makes it less reliable in production (e.g. hung or SIGBUS on faulty disks; less controllable memory usage). Nevertheless it is good in benchmarks. [#22206](#) ([alexey-milovidov](#)).
- Avoid unnecessary data copy when using codec `NONE`. Please note that codec `NONE` is mostly useless - it's recommended to always use compression (`LZ4` is by default). Despite the common belief, disabling compression may not improve performance (the opposite effect is possible). The `NONE` codec is useful in some cases: - when data is uncompressable; - for synthetic benchmarks. [#22145](#) ([alexey-milovidov](#)).
- Faster `GROUP BY` with small `max_rows_to_group_by` and `group_by_overflow_mode='any'`. [#21856](#) ([Nikolai Kochetov](#)).
- Optimize performance of queries like `SELECT ... FINAL ... WHERE`. Now in queries with `FINAL` it's allowed to move to `PREDWHERE` columns, which are in sorting key. [#21830](#) ([foolchi](#)).
- Improved performance by replacing `memcpy` to another implementation. This closes [#18583](#). [#21520](#) ([alexey-milovidov](#)).
- Improve performance of aggregation in order of sorting key (with enabled setting `optimize_aggregation_in_order`). [#19401](#) ([Anton Popov](#)).

## Improvement

- Add connection pool for PostgreSQL table/database engine and dictionary source. Should fix [#21444](#). [#21839](#) ([Kseniia Sumarokova](#)).
- Support non-default table schema for postgres storage/table-function. Closes [#21701](#). [#21711](#) ([Kseniia Sumarokova](#)).
- Support replicas priority for postgres dictionary source. [#21710](#) ([Kseniia Sumarokova](#)).

- Introduce a new merge tree setting `min_bytes_to_rebalance_partition_over_jbod` which allows assigning new parts to different disks of a JBOD volume in a balanced way. [#16481 \(Amos Bird\)](#).
- Added `Grant`, `Revoke` and `System` values of `query_kind` column for corresponding queries in `system.query_log`. [#21102 \(Vasily Nemkov\)](#).
- Allow customizing timeouts for HTTP connections used for replication independently from other HTTP timeouts. [#20088 \(nvartolomei\)](#).
- Better exception message in client in case of exception while server is writing blocks. In previous versions client may get misleading message like `Data compressed with different methods`. [#22427 \(alexey-milovidov\)](#).
- Fix error `Directory tmp_fetch_XXX already exists` which could happen after failed fetch part. Delete temporary fetch directory if it already exists. Fixes [#14197](#). [#22411 \(nvartolomei\)](#).
- Fix MSan report for function `range` with `UInt256` argument (support for large integers is experimental). This closes [#22157](#). [#22387 \(alexey-milovidov\)](#).
- Add `current_database` column to `system.processes` table. It contains the current database of the query. [#22365 \(Alexander Kuzmenkov\)](#).
- Add case-insensitive history search/navigation and subword movement features to `clickhouse-client`. [#22105 \(Amos Bird\)](#).
- If tuple of NULLs, e.g. `(NULL, NULL)` is on the left hand side of `IN` operator with tuples of non-NULLs on the right hand side, e.g. `SELECT (NULL, NULL) IN ((0, 0), (3, 1))` return 0 instead of throwing an exception about incompatible types. The expression may also appear due to optimization of something like `SELECT (NULL, NULL) = (8, 0) OR (NULL, NULL) = (3, 2) OR (NULL, NULL) = (0, 0) OR (NULL, NULL) = (3, 1)`. This closes [#22017](#). [#22063 \(alexey-milovidov\)](#).
- Update used version of `simdjson` to 0.9.1. This fixes [#21984](#). [#22057 \(Vitaly Baranov\)](#).
- Added case insensitive aliases for `CONNECTION_ID()` and `VERSION()` functions. This fixes [#22028](#). [#22042 \(Eugene Klimov\)](#).
- Add option `strict_increase` to `windowFunnel` function to calculate each event once (resolve [#21835](#)). [#22025 \(Vladimir\)](#).
- If partition key of a `MergeTree` table does not include `Date` or `DateTime` columns but includes exactly one `DateTime64` column, expose its values in the `min_time` and `max_time` columns in `system.parts` and `system.parts_columns` tables. Add `min_time` and `max_time` columns to `system.parts_columns` table (these was inconsistency to the `system.parts` table). This closes [#18244](#). [#22011 \(alexey-milovidov\)](#).
- Supported `replication_alter_partitions_sync=1` setting in `clickhouse-copier` for moving partitions from helping table to destination. Decreased default timeouts. Fixes [#21911](#). [#21912 \(turbo jason\)](#).
- Show path to data directory of `EmbeddedRocksDB` tables in system tables. [#21903 \(tavplubix\)](#).
- Add profile event `HedgedRequestsChangeReplica`, change read data timeout from sec to ms. [#21886 \(Kruglov Pavel\)](#).
- DiskS3 (experimental feature under development). Fixed bug with the impossibility to move directory if the destination is not empty and cache disk is used. [#21837 \(Pavel Kovalenko\)](#).
- Better formatting for `Array` and `Map` data types in Web UI. [#21798 \(alexey-milovidov\)](#).
- Update clusters only if their configurations were updated. [#21685 \(Kruglov Pavel\)](#).

- Propagate query and session settings for distributed DDL queries. Set `distributed_ddl_entry_format_version` to 2 to enable this. Added `distributed_ddl_output_mode` setting. Supported modes: `none`, `throw` (default), `null_status_on_timeout` and `never_throw`. Miscellaneous fixes and improvements for Replicated database engine. #21535 (tavplubix).
- If `PODArray` was instantiated with element size that is neither a fraction or a multiple of 16, buffer overflow was possible. No bugs in current releases exist. #21533 (alexey-milovidov).
- Add `last_error_time`/`last_error_message`/`last_error_stacktrace`/`remote` columns for `system.errors`. #21529 (Azat Khuzhin).
- Add aliases `simpleJSONExtract`/`simpleJSONHas` to `visitParam`/`visitParamExtract{UInt, Int, Bool, Float, Raw, String}`. Fixes #21383. #21519 (fastio).
- Add setting `optimize_skip_unused_shards_limit` to limit the number of sharding key values for `optimize_skip_unused_shards`. #21512 (Azat Khuzhin).
- Improve `clickhouse-format` to not throw exception when there are extra spaces or comment after the last query, and throw exception early with readable message when format `ASTInsertQuery` with data . #21311 (flynn).
- Improve support of integer keys in data type `Map`. #21157 (Anton Popov).
- MaterializeMySQL: attempt to reconnect to MySQL if the connection is lost. #20961 (Håvard Kvålen).
- Support more cases to rewrite CROSS JOIN to INNER JOIN. #20392 (Vladimir).
- Do not create empty parts on INSERT when `optimize_on_insert` setting enabled. Fixes #20304. #20387 (Kruglov Pavel).
- MaterializeMySQL: add minmax skipping index for `_version` column. #20382 (Stig Bakken).
- Add option `--backslash` for `clickhouse-format`, which can add a backslash at the end of each line of the formatted query. #21494 (flynn).
- Now clickhouse will not throw `LOGICAL_ERROR` exception when we try to mutate the already covered part. Fixes #22013. #22291 (alesapin).

## Bug Fix

- Remove socket from epoll before cancelling packet receiver in `HedgedConnections` to prevent possible race. Fixes #22161. #22443 (Kruglov Pavel).
- Add (missing) memory accounting in parallel parsing routines. In previous versions OOM was possible when the resultset contains very large blocks of data. This closes #22008. #22425 (alexey-milovidov).
- Fix exception which may happen when `SELECT` has constant `WHERE` condition and source table has columns which names are digits. #22270 (LiuNeng).
- Fix query cancellation with `use_hedged_requests=0` and `async_socket_for_remote=1`. #22183 (Azat Khuzhin).
- Fix uncaught exception in `InterserverIOHTTPHandler`. #22146 (Azat Khuzhin).
- Fix docker entrypoint in case `http_port` is not in the config. #22132 (Ewout).
- Fix error `Invalid number of rows in Chunk` in `JOIN` with `TOTALS` and `arrayJoin`. Closes #19303. #22129 (Vladimir).
- Fix the background thread pool name which used to poll message from Kafka. The Kafka engine with the broken thread pool will not consume the message from message queue. #22122 (fastio).

- Fix waiting for `OPTIMIZE` and `ALTER` queries for `ReplicatedMergeTree` table engines. Now the query will not hang when the table was detached or restarted. #22118 (alesapin).
- Disable `async_socket_for_remote/use_hedged_requests` for buggy Linux kernels. #22109 (Azat Khuzhin).
- Docker entrypoint: avoid chown of `.` in case when `LOG_PATH` is empty. Closes #22100. #22102 (filimonov).
- The function `decrypt` was lacking a check for the minimal size of data encrypted in AEAD mode. This closes #21897. #22064 (alexey-milovidov).
- In rare case, merge for `CollapsingMergeTree` may create granule with `index_granularity + 1` rows. Because of this, internal check, added in #18928 (affects 21.2 and 21.3), may fail with error `Incomplete granules are not allowed while blocks are granules size`. This error did not allow parts to merge. #21976 (Nikolai Kochetov).
- Reverted #15454 that may cause significant increase in memory usage while loading external dictionaries of hashed type. This closes #21935. #21948 (Maksim Kita).
- Prevent hedged connections overlaps (`Unknown packet 9 from server error`). #21941 (Azat Khuzhin).
- Fix reading the HTTP POST request with "multipart/form-data" content type in some cases. #21936 (Ivan).
- Fix wrong `ORDER BY` results when a query contains window functions, and optimization for reading in primary key order is applied. Fixes #21828. #21915 (Alexander Kuzmenkov).
- Fix deadlock in first catboost model execution. Closes #13832. #21844 (Kruglov Pavel).
- Fix incorrect query result (and possible crash) which could happen when `WHERE` or `HAVING` condition is pushed before `GROUP BY`. Fixes #21773. #21841 (Nikolai Kochetov).
- Better error handling and logging in `WriteBufferFromS3`. #21836 (Pavel Kovalenko).
- Fix possible crashes in aggregate functions with combinator `Distinct`, while using two-level aggregation. This is a follow-up fix of #18365 . Can only reproduced in production env. #21818 (Amos Bird).
- Fix scalar subquery index analysis. This fixes #21717 , which was introduced in #18896. #21766 (Amos Bird).
- Fix bug for `ReplicatedMerge` table engines when `ALTER MODIFY COLUMN` query doesn't change the type of `Decimal` column if its size (32 bit or 64 bit) doesn't change. #21728 (alesapin).
- Fix possible infinite waiting when concurrent `OPTIMIZE` and `DROP` are run for `ReplicatedMergeTree`. #21716 (Azat Khuzhin).
- Fix function `arrayElement` with type `Map` for constant integer arguments. #21699 (Anton Popov).
- Fix SIGSEGV on not existing attributes from `ip_trie` with `access_to_key_from_attributes`. #21692 (Azat Khuzhin).
- Server now start accepting connections only after `DDLWorker` and dictionaries initialization. #21676 (Azat Khuzhin).
- Add type conversion for keys of tables of type `Join` (previously led to SIGSEGV). #21646 (Azat Khuzhin).
- Fix distributed requests cancellation (for example simple select from multiple shards with limit, i.e. `select * from remote('127.{2,3}', system.numbers) limit 100`) with `async_socket_for_remote=1`. #21643 (Azat Khuzhin).
- Fix `fsync_part_directory` for horizontal merge. #21642 (Azat Khuzhin).

- Remove unknown columns from joined table in `WHERE` for queries to external database engines (MySQL, PostgreSQL). close #14614, close #19288 (dup), close #19645 (dup). #21640 (Vladimir).
- `std::terminate` was called if there is an error writing data into s3. #21624 (Vladimir).
- Fix possible error `Cannot find column` when `optimize_skip_unused_shards` is enabled and zero shards are used. #21579 (Azat Khuzhin).
- In case if query has constant `WHERE` condition, and setting `optimize_skip_unused_shards` enabled, all shards may be skipped and query could return incorrect empty result. #21550 (Amos Bird).
- Fix table function `clusterAllReplicas` returns wrong `_shard_num`. close #21481. #21498 (flynn).
- Fix that S3 table holds old credentials after config update. #21457 (Grigory Pervakov).
- Fixed race on SSL object inside `SecureSocket` in Poco. #21456 (Nikita Mikhaylov).
- Fix Avro format parsing for Kafka. Fixes #21437. #21438 (Ilya Golshtein).
- Fix receive and send timeouts and non-blocking read in secure socket. #21429 (Kruglov Pavel).
- `force_drop_table` flag didn't work for `MATERIALIZED VIEW`, it's fixed. Fixes #18943. #20626 (tavplubix).
- Fix name clashes in `PredicateRewriteVisitor`. It caused incorrect `WHERE` filtration after full join. Close #20497. #20622 (Vladimir).

## Build/Testing/Packaging Improvement

- Add `Jepsen` tests for ClickHouse Keeper. #21677 (alesapin).
- Run stateless tests in parallel in CI. Depends on #22181. #22300 (alesapin).
- Enable status check for `SQLancer` CI run. #22015 (Ilya Yatsishin).
- Multiple preparations for PowerPC builds: Enable the bundled `openldap` on `ppc64le`. #22487 (Kfir Itzhak). Enable compiling on `ppc64le` with Clang. #22476 (Kfir Itzhak). Fix compiling boost on `ppc64le`. #22474 (Kfir Itzhak). Fix CMake error about internal CMake variable `CMAKE_ASM_COMPILE_OBJECT` not set on `ppc64le`. #22469 (Kfir Itzhak). Fix Fedora/RHEL/CentOS not finding `libclang_rt.builtins` on `ppc64le`. #22458 (Kfir Itzhak). Enable building with `jemalloc` on `ppc64le`. #22447 (Kfir Itzhak). Fix ClickHouse's config embedding and cctz's timezone embedding on `ppc64le`. #22445 (Kfir Itzhak). Fixed compiling on `ppc64le` and use the correct instruction pointer register on `ppc64le`. #22430 (Kfir Itzhak).
- Re-enable the S3 (AWS) library on `aarch64`. #22484 (Kfir Itzhak).
- Add `tzdata` to Docker containers because reading `ORC` formats requires it. This closes #14156. #22000 (alexey-milovidov).
- Introduce 2 arguments for `clickhouse-server` image Dockerfile: `deb_location` & `single_binary_location`. #21977 (filimonov).
- Allow to use clang-tidy with release builds by enabling assertions if it is used. #21914 (alexey-milovidov).
- Add `llvm-12` binaries name to search in cmake scripts. Implicit constants conversions to mute clang warnings. Updated submodules to build with CMake 3.19. Mute recursion in macro expansion in `readpassphrase` library. Deprecated `-fuse-ld` changed to `--ld-path` for clang. #21597 (Ilya Yatsishin).
- Updating `docker/test/testflows/runner/dockerd-entrypoint.sh` to use Yandex dockerhub-proxy, because Docker Hub has enabled very restrictive rate limits #21551 (vzakaznikov).
- Fix macOS shared lib build. #20184 (nvartolomei).

- Add `ctime` option to `zookeeper-dump-tree`. It allows to dump node creation time. #21842 (Ilya).

# ClickHouse release 21.3 (LTS)

## ClickHouse release v21.3, 2021-03-12

### Backward Incompatible Change

- Now it's not allowed to create MergeTree tables in old syntax with table TTL because it's just ignored. Attach of old tables is still possible. #20282 (alesapin).
- Now all case-insensitive function names will be rewritten to their canonical representations. This is needed for projection query routing (the upcoming feature). #20174 (Amos Bird).
- Fix creation of `TTL` in cases, when its expression is a function and it is the same as `ORDER BY` key. Now it's allowed to set custom aggregation to primary key columns in `TTL` with `GROUP BY`. Backward incompatible: For primary key columns, which are not in `GROUP BY` and aren't set explicitly now is applied function `any` instead of `max`, when `TTL` is expired. Also if you use `TTL` with `WHERE` or `GROUP BY` you can see exceptions at merges, while making rolling update. #15450 (Anton Popov).

### New Feature

- Add file engine settings: `engine_file_empty_if_not_exists` and `engine_file_truncate_on_insert`. #20620 (M0r64n).
- Add aggregate function `deltaSum` for summing the differences between consecutive rows. #20057 (Russ Frank).
- New `event_time_microseconds` column in `system.part_log` table. #20027 (Bharat Nallan).
- Added `timezoneOffset(datetime)` function which will give the offset from UTC in seconds. This close #issue:19850. #19962 (keenwolf).
- Add setting `insert_shard_id` to support insert data into specific shard from distributed table. #19961 (flynn).
- Function `reinterpretAs` updated to support big integers. Fixes #19691. #19858 (Maksim Kita).
- Added Server Side Encryption Customer Keys (the `x-amz-server-side-encryption-customer-(key/md5)` header) support in S3 client. See the link. Closes #19428. #19748 (Vladimir Chebotarev).
- Added `implicit_key` option for `executable` dictionary source. It allows to avoid printing key for every record if records comes in the same order as the input keys. Implements #14527. #19677 (Maksim Kita).
- Add quota type `query_selects` and `query_inserts`. #19603 (JackyWoo).
- Add function `extractTextFromHTML` #19600 (zlx19950903), (alexey-milovidov).
- Tables with `MergeTree*` engine now have two new table-level settings for query concurrency control. Setting `max_concurrent_queries` limits the number of concurrently executed queries which are related to this table. Setting `min_marks_to_honor_max_concurrent_queries` tells to apply previous setting only if query reads at least this number of marks. #19544 (Amos Bird).
- Added `file` function to read file from `user_files` directory as a String. This is different from the `file` table function. This implements #issue:18851. #19204 (keenwolf).

### Experimental feature

- Add experimental `Replicated` database engine. It replicates DDL queries across multiple hosts. #16193 (tavplubix).

- Introduce experimental support for window functions, enabled with `allow_experimental_window_functions = 1`. This is a preliminary, alpha-quality implementation that is not suitable for production use and will change in backward-incompatible ways in future releases. Please see [the documentation](#) for the list of supported features. [#20337 \(Alexander Kuzmenkov\)](#).
- Add the ability to backup/restore metadata files for DiskS3. [#18377 \(Pavel Kovalenko\)](#).

## Performance Improvement

- Hedged requests for remote queries. When setting `use_hedged_requests` enabled (off by default), allow to establish many connections with different replicas for query. New connection is enabled in case existent connection(s) with replica(s) were not established within `hedged_connection_timeout` or no data was received within `receive_data_timeout`. Query uses the first connection which send non empty progress packet (or data packet, if `allow_changing_replica_until_first_data_packet`); other connections are cancelled. Queries with `max_parallel_replicas > 1` are supported. [#19291 \(Kruglov Pavel\)](#). This allows to significantly reduce tail latencies on very large clusters.
- Added support for `PREWHERE` (and enable the corresponding optimization) when tables have row-level security expressions specified. [#19576 \(Denis Glazachev\)](#).
- The setting `distributed_aggregation_memory_efficient` is enabled by default. It will lower memory usage and improve performance of distributed queries. [#20599 \(alexey-milovidov\)](#).
- Improve performance of GROUP BY multiple fixed size keys. [#20472 \(alexey-milovidov\)](#).
- Improve performance of aggregate functions by more strict aliasing. [#19946 \(alexey-milovidov\)](#).
- Speed up reading from `Memory` tables in extreme cases (when reading speed is in order of 50 GB/sec) by simplification of pipeline and (consequently) less lock contention in pipeline scheduling. [#20468 \(alexey-milovidov\)](#).
- Partially reimplement HTTP server to make it making less copies of incoming and outgoing data. It gives up to 1.5 performance improvement on inserting long records over HTTP. [#19516 \(Ivan\)](#).
- Add `compress` setting for `Memory` tables. If it's enabled the table will use less RAM. On some machines and datasets it can also work faster on `SELECT`, but it is not always the case. This closes [#20093](#). Note: there are reasons why `Memory` tables can work slower than `MergeTree`: (1) lack of compression (2) static size of blocks (3) lack of indices and prewhere... [#20168 \(alexey-milovidov\)](#).
- Slightly better code in aggregation. [#20978 \(alexey-milovidov\)](#).
- Add back `intDiv/modulo` specializations for better performance. This fixes [#21293](#). The regression was introduced in <https://github.com/ClickHouse/ClickHouse/pull/18145>. [#21307 \(Amos Bird\)](#).
- Do not squash blocks too much on `INSERT SELECT` if inserting into `Memory` table. In previous versions inefficient data representation was created in `Memory` table after `INSERT SELECT`. This closes [#13052](#). [#20169 \(alexey-milovidov\)](#).
- Fix at least one case when `DataType` parser may have exponential complexity (found by fuzzer). This closes [#20096](#). [#20132 \(alexey-milovidov\)](#).
- Parallelize `SELECT` with `FINAL` for single part with `level > 0` when `do_not_merge_across_partitions_select_final` setting is 1. [#19375 \(Kruglov Pavel\)](#).
- Fill only requested columns when querying `system.parts` and `system.parts_columns`. Closes [#19570](#). [#21035 \(Anmol Arora\)](#).
- Perform algebraic optimizations of arithmetic expressions inside `avg` aggregate function. close [#20092](#). [#20183 \(flynn\)](#).

## Improvement

- Case-insensitive compression methods for table functions. Also fixed LZMA compression method which was checked in upper case. #21416 ([Vladimir Chebotarev](#)).
- Add two settings to delay or throw error during insertion when there are too many inactive parts. This is useful when server fails to clean up parts quickly enough. #20178 ([Amos Bird](#)).
- Provide better compatibility for mysql clients. 1. mysql jdbc 2. mycli. #21367 ([Amos Bird](#)).
- Forbid to drop a column if it's referenced by materialized view. Closes #21164. #21303 ([flynn](#)).
- MySQL dictionary source will now retry unexpected connection failures (Lost connection to MySQL server during query) which sometimes happen on SSL/TLS connections. #21237 ([Alexander Kazakov](#)).
- Usability improvement: more consistent `DateTime64` parsing: recognize the case when unix timestamp with subsecond resolution is specified as scaled integer (like `1111111111222` instead of `1111111111.222`). This closes #13194. #21053 ([alexey-milovidov](#)).
- Do only merging of sorted blocks on initiator with `distributed_group_by_no_merge`. #20882 ([Azat Khuzhin](#)).
- When loading config for mysql source ClickHouse will now randomize the list of replicas with the same priority to ensure the round-robin logic of picking mysql endpoint. This closes #20629. #20632 ([Alexander Kazakov](#)).
- Function 'reinterpretAs(x, Type)' renamed into 'reinterpret(x, Type)'. #20611 ([Maksim Kita](#)).
- Support vhost for RabbitMQ engine #20576. #20596 ([Kseniia Sumarokova](#)).
- Improved serialization for data types combined of Arrays and Tuples. Improved matching enum data types to protobuf enum type. Fixed serialization of the `Map` data type. Omitted values are now set by default. #20506 ([Vitaly Baranov](#)).
- Fixed race between execution of distributed DDL tasks and cleanup of DDL queue. Now DDL task cannot be removed from ZooKeeper if there are active workers. Fixes #20016. #20448 ([tavplubix](#)).
- Make FQDN and other DNS related functions work correctly in alpine images. #20336 ([filimonov](#)).
- Do not allow early constant folding of explicitly forbidden functions. #20303 ([Azat Khuzhin](#)).
- Implicit conversion from integer to Decimal type might succeed if integer value does not fit into Decimal type. Now it throws `ARGUMENT_OUT_OF_BOUND`. #20232 ([tavplubix](#)).
- Lockless `SYSTEM FLUSH DISTRIBUTED`. #20215 ([Azat Khuzhin](#)).
- Normalize `count(constant)`, `sum(1)` to `count()`. This is needed for projection query routing. #20175 ([Amos Bird](#)).
- Support all native integer types in bitmap functions. #20171 ([Amos Bird](#)).
- Updated `CacheDictionary`, `ComplexCacheDictionary`, `SSDCacheDictionary`, `SSDComplexKeyDictionary` to use `LRUHashMap` as underlying index. #20164 ([Maksim Kita](#)).
- The setting `access_management` is now configurable on startup by providing `CLICKHOUSE_DEFAULT_ACCESS_MANAGEMENT`, defaults to disabled (0) which was the prior value. #20139 ([Marquitos](#)).
- Fix `toDateTime64(toDate()/toDateTime())` for `DateTime64` - Implement `DateTime64` clamping to match `DateTime` behaviour. #20131 ([Azat Khuzhin](#)).

- Quota improvements: SHOW TABLES is now considered as one query in the quota calculations, not two queries. SYSTEM queries now consume quota. Fix calculation of interval's end in quota consumption. #20106 (Vitaly Baranov).
- Supports path IN (set) expressions for system.zookeeper table. #20105 (小路).
- Show full details of MaterializeMySQL tables in system.tables. #20051 (Stig Bakken).
- Fix data race in executable dictionary that was possible only on misuse (when the script returns data ignoring its input). #20045 (alexey-milovidov).
- The value of MYSQL\_OPT\_RECONNECT option can now be controlled by "opt\_reconnect" parameter in the config section of mysql replica. #19998 (Alexander Kazakov).
- If user calls JSONExtract function with Float32 type requested, allow inaccurate conversion to the result type. For example the number 0.1 in JSON is double precision and is not representable in Float32, but the user still wants to get it. Previous versions return 0 for non-Nullable type and NULL for Nullable type to indicate that conversion is imprecise. The logic was 100% correct but it was surprising to users and leading to questions. This closes #13962. #19960 (alexey-milovidov).
- Add conversion of block structure for INSERT into Distributed tables if it does not match. #19947 (Azat Khuzhin).
- Improvement for the system.distributed\_ddl\_queue table. Initialize MaxDDLEntryID to the last value after restarting. Before this PR, MaxDDLEntryID will remain zero until a new DDLTask is processed. #19924 (Amos Bird).
- Show MaterializeMySQL tables in system.parts. #19770 (Stig Bakken).
- Add separate config directive for Buffer profile. #19721 (Azat Khuzhin).
- Move conditions that are not related to JOIN to WHERE clause. #18720. #19685 (hexiaoting).
- Add ability to throttle INSERT into Distributed based on amount of pending bytes for async send (bytes\_to\_delay\_insert/max\_delay\_to\_insert and bytes\_to\_throw\_insert settings for Distributed engine has been added). #19673 (Azat Khuzhin).
- Fix some rare cases when write errors can be ignored in destructors. #19451 (Azat Khuzhin).
- Print inline frames in stack traces for fatal errors. #19317 (Ivan).

## Bug Fix

- Fix redundant reconnects to ZooKeeper and the possibility of two active sessions for a single clickhouse server. Both problems introduced in #14678. #21264 (alesapin).
- Fix error Bad cast from type ... to DB::ColumnLowCardinality while inserting into table with LowCardinality column from Values format. Fixes #21140 #21357 (Nikolai Kochetov).
- Fix a deadlock in ALTER DELETE mutations for non replicated MergeTree table engines when the predicate contains the table itself. Fixes #20558. #21477 (alesapin).
- Fix SIGSEGV for distributed queries on failures. #21434 (Azat Khuzhin).
- Now ALTER MODIFY COLUMN queries will correctly affect changes in partition key, skip indices, TTLs, and so on. Fixes #13675. #21334 (alesapin).
- Fix bug with join\_use\_nulls and joining TOTALS from subqueries. This closes #19362 and #21137. #21248 (vdimir).
- Fix crash in EXPLAIN for query with UNION. Fixes #20876, #21170. #21246 (flynn).

- Now mutations allowed only for table engines that support them (MergeTree family, Memory, MaterializedView). Other engines will report a more clear error. Fixes #21168. #21183 (alesapin).
- Fixes #21112. Fixed bug that could cause duplicates with insert query (if one of the callbacks came a little too late). #21138 (Kseniia Sumarokova).
- Fix `input_format_null_as_default` take effective when types are nullable. This fixes #21116 . #21121 (Amos Bird).
- fix bug related to cast Tuple to Map. Closes #21029. #21120 (hexiaoting).
- Fix the metadata leak when the Replicated\*MergeTree with custom (non default) ZooKeeper cluster is dropped. #21119 (fastio).
- Fix type mismatch issue when using LowCardinality keys in joinGet. This fixes #21114. #21117 (Amos Bird).
- fix `default_replica_path` and `default_replica_name` values are useless on Replicated(\*)MergeTree engine when the engine needs specify other parameters. #21060 (mxzlxy).
- Out of bound memory access was possible when formatting specifically crafted out of range value of type `DateTime64`. This closes #20494. This closes #20543. #21023 (alexey-milovidov).
- Block parallel insertions into storage join. #21009 (vdimir).
- Fixed behaviour, when `ALTER MODIFY COLUMN` created mutation, that will knowingly fail. #21007 (Anton Popov).
- Closes #9969. Fixed Brotli http compression error, which reproduced for large data sizes, slightly complicated structure and with json output format. Update Brotli to the latest version to include the "fix rare access to uninitialized data in ring-buffer". #20991 (Kseniia Sumarokova).
- Fix 'Empty task was returned from async task queue' on query cancellation. #20881 (Azat Khuzhin).
- USE database; query did not work when using MySQL 5.7 client to connect to ClickHouse server, it's fixed. Fixes #18926. #20878 (tavplubix).
- Fix usage of `-Distinct` combinator with `-State` combinator in aggregate functions. #20866 (Anton Popov).
- Fix subquery with union distinct and limit clause. close #20597. #20610 (flynn).
- Fixed inconsistent behavior of dictionary in case of queries where we look for absent keys in dictionary. #20578 (Nikita Mikhaylov).
- Fix the number of threads for scalar subqueries and subqueries for index (after #19007 single thread was always used). Fixes #20457, #20512. #20550 (Nikolai Kochetov).
- Fix crash which could happen if unknown packet was received from remove query (was introduced in #17868). #20547 (Azat Khuzhin).
- Add proper checks while parsing directory names for async INSERT (fixes SIGSEGV). #20498 (Azat Khuzhin).
- Fix function `transform` does not work properly for floating point keys. Closes #20460. #20479 (flynn).
- Fix infinite loop when propagating WITH aliases to subqueries. This fixes #20388. #20476 (Amos Bird).
- Fix abnormal server termination when http client goes away. #20464 (Azat Khuzhin).
- Fix `LOGICAL_ERROR` for `join_use_nulls=1` when JOIN contains const from SELECT. #20461 (Azat Khuzhin).

- Check if table function `view` is used in expression list and throw an error. This fixes #20342. #20350 ([Amos Bird](#)).
- Avoid invalid dereference in `RANGE_HASHED()` dictionary. #20345 ([Azat Khuzhin](#)).
- Fix null dereference with `join_use_nulls=1`. #20344 ([Azat Khuzhin](#)).
- Fix incorrect result of binary operations between two constant decimals of different scale. Fixes #20283. #20339 ([Maksim Kita](#)).
- Fix too often retries of failed background tasks for `ReplicatedMergeTree` table engines family. This could lead to too verbose logging and increased CPU load. Fixes #20203. #20335 ([alesapin](#)).
- Restrict to `DROP` or `RENAME` version column of `*CollapsingMergeTree` and `ReplacingMergeTree` table engines. #20300 ([alesapin](#)).
- Fixed the behavior when in case of broken JSON we tried to read the whole file into memory which leads to exception from the allocator. Fixes #19719. #20286 ([Nikita Mikhaylov](#)).
- Fix exception during vertical merge for `MergeTree` table engines family which don't allow to perform vertical merges. Fixes #20259. #20279 ([alesapin](#)).
- Fix rare server crash on config reload during the shutdown. Fixes #19689. #20224 ([alesapin](#)).
- Fix CTE when using in `INSERT SELECT`. This fixes #20187, fixes #20195. #20211 ([Amos Bird](#)).
- Fixes #19314. #20156 ([Ivan](#)).
- fix `toMinute` function to handle special timezone correctly. #20149 ([keenwolf](#)).
- Fix server crash after query with `if` function with `Tuple` type of `then/else` branches result. `Tuple` type must contain `Array` or another complex type. Fixes #18356. #20133 ([alesapin](#)).
- The `MongoDB` table engine now establishes connection only when it's going to read data. `ATTACH TABLE` won't try to connect anymore. #20110 ([Vitaly Baranov](#)).
- Bugfix in `StorageJoin`. #20079 ([vdimir](#)).
- Fix the case when calculating modulo of division of negative number by small divisor, the resulting data type was not large enough to accomodate the negative result. This closes #20052. #20067 ([alexey-milovidov](#)).
- MaterializeMySQL: Fix replication for statements that update several tables. #20066 ([Håvard Kvålen](#)).
- Prevent "Connection refused" in docker during initialization script execution. #20012 ([filimonov](#)).
- `EmbeddedRocksDB` is an experimental storage. Fix the issue with lack of proper type checking. Simplified code. This closes #19967. #19972 ([alexey-milovidov](#)).
- Fix a segfault in function `fromModifiedJulianDay` when the argument type is `Nullable(T)` for any integral types other than `Int32`. #19959 ([PHO](#)).
- BloomFilter index crash fix. Fixes #19757. #19884 ([Maksim Kita](#)).
- Deadlock was possible if `system.text_log` is enabled. This fixes #19874. #19875 ([alexey-milovidov](#)).
- Fix starting the server with tables having default expressions containing `dictGet()`. Allow getting return type of `dictGet()` without loading dictionary. #19805 ([Vitaly Baranov](#)).
- Fix clickhouse-client abort exception while executing only `select`. #19790 ([taiyang-li](#)).

- Fix a bug that moving pieces to destination table may failed in case of launching multiple clickhouse-copiers. [#19743](#) ([madianjun](#)).
- Background thread which executes `ON CLUSTER` queries might hang waiting for dropped replicated table to do something. It's fixed. [#19684](#) ([yiguolei](#)).

## Build/Testing/Packaging Improvement

- Allow to build ClickHouse with AVX-2 enabled globally. It gives slight performance benefits on modern CPUs. Not recommended for production and will not be supported as official build for now. [#20180](#) ([alexey-milovidov](#)).
- Fix some of the issues found by Coverity. See [#19964](#). [#20010](#) ([alexey-milovidov](#)).
- Allow to start up with modified binary under gdb. In previous version if you set up breakpoint in gdb before start, server will refuse to start up due to failed integrity check. [#21258](#) ([alexey-milovidov](#)).
- Add a test for different compression methods in Kafka. [#21111](#) ([filimonov](#)).
- Fixed port clash from `test_storage_kerberized_hdfs` test. [#19974](#) ([Ilya Yatsishin](#)).
- Print `stdout` and `stderr` to log when failed to start docker in integration tests. Before this PR there was a very short error message in this case which didn't help to investigate the problems. [#20631](#) ([Vitaly Baranov](#)).

## ClickHouse release 21.2

### ClickHouse release v21.2.2.8-stable, 2021-02-07

#### Backward Incompatible Change

- Bitwise functions (`bitAnd`, `bitOr`, etc) are forbidden for floating point arguments. Now you have to do explicit cast to integer. [#19853](#) ([Azat Khuzhin](#)).
- Forbid `lcm/gcd` for floats. [#19532](#) ([Azat Khuzhin](#)).
- Fix memory tracking for `OPTIMIZE TABLE/merges`; account query memory limits and sampling for `OPTIMIZE TABLE/merges`. [#18772](#) ([Azat Khuzhin](#)).
- Disallow floating point column as partition key, see [#18421](#). [#18464](#) ([hexiaoting](#)).
- Excessive parenthesis in type definitions no longer supported, example: `Array((UInt8))`.

#### New Feature

- Added `PostgreSQL` table engine (both select/insert, with support for multidimensional arrays), also as table function. Added `PostgreSQL` dictionary source. Added `PostgreSQL` database engine. [#18554](#) ([Kseniia Sumarokova](#)).
- Data type `Nested` now supports arbitrary levels of nesting. Introduced subcolumns of complex types, such as `size0` in `Array`, `null` in `Nullable`, names of `Tuple` elements, which can be read without reading of whole column. [#17310](#) ([Anton Popov](#)).
- Added `Nullable` support for `FlatDictionary`, `HashedDictionary`, `ComplexKeyHashedDictionary`, `DirectDictionary`, `ComplexKeyDirectDictionary`, `RangeHashedDictionary`. [#18236](#) ([Maksim Kita](#)).
- Adds a new table called `system.distributed_ddl_queue` that displays the queries in the DDL worker queue. [#17656](#) ([Bharat Nallan](#)).

- Added support of mapping LDAP group names, and attribute values in general, to local roles for users from Ildap user directories. #17211 (Denis Glazachev).
- Support insert into table function `cluster`, and for both table functions `remote` and `cluster`, support distributing data across nodes by specify sharding key. Close #16752. #18264 (flynn).
- Add function `decodeXMLComponent` to decode characters for XML. Example: `SELECT decodeXMLComponent('Hello,"world"!')` #17659. #18542 (nauta).
- Added functions `parseDateTimeBestEffortUSOrZero`, `parseDateTimeBestEffortUSOrNull`. #19712 (Maksim Kita).
- Add `sign` math function. #19527 (flynn).
- Add information about used features (functions, table engines, etc) into `system.query_log`. #18495. #19371 (Ksenia Sumarokova).
- Function `formatDateTime` support the `%Q` modification to format date to quarter. #19224 (Jianmei Zhang).
- Support MetaKey+Enter hotkey binding in play UI. #19012 (sundyli).
- Add three functions for map data type: 1. `mapContains(map, key)` to check weather map.keys include the second parameter key. 2. `mapKeys(map)` return all the keys in Array format 3. `mapValues(map)` return all the values in Array format. #18788 (hexiaoting).
- Add `log_comment` setting related to #18494. #18549 (Zijie Lu).
- Add support of tuple argument to `argMin` and `argMax` functions. #17359 (Ildus Kurbangaliev).
- Support `EXISTS VIEW` syntax. #18552 (Du Chuan).
- Add `SELECT ALL` syntax. closes #18706. #18723 (flynn).

## Performance Improvement

- Faster parts removal by lowering the number of `stat` syscalls. This returns the optimization that existed while ago. More safe interface of `IDisk`. This closes #19065. #19086 (alexey-milovidov).
- Aliases declared in `WITH` statement are properly used in index analysis. Queries like `WITH column AS alias SELECT ... WHERE alias = ...` may use index now. #18896 (Amos Bird).
- Add `optimize_alias_column_prediction` (on by default), that will: - Respect aliased columns in `WHERE` during partition pruning and skipping data using secondary indexes; - Respect aliased columns in `WHERE` for trivial count queries for `optimize_trivial_count`; - Respect aliased columns in `GROUP BY/ORDER BY` for `optimize_aggregation_in_order/optimize_read_in_order`. #16995 (sundyli).
- Speed up aggregate function `sum`. Improvement only visible on synthetic benchmarks and not very practical. #19216 (alexey-milovidov).
- Update `libc++` and use another ABI to provide better performance. #18914 (Danila Kutenin).
- Rewrite `sumIf()` and `sum(if())` function to `countIf()` function when logically equivalent. #17041 (flynn).
- Use a connection pool for S3 connections, controlled by the `s3_max_connections` settings. #13405 (Vladimir Chebotarev).
- Add support for zstd long option for better compression of string columns to save space. #17184 (ygrek).
- Slightly improve server latency by removing access to configuration on every connection. #19863 (alexey-milovidov).
- Reduce lock contention for multiple layers of the `Buffer` engine. #19379 (Azat Khuzhin).

- Support splitting `Filter` step of query plan into `Expression + Filter` pair. Together with `Expression + Expression` merging optimization (#17458) it may delay execution for some expressions after `Filter` step. #19253 (Nikolai Kochetov).

## Improvement

- `SELECT count() FROM table` now can be executed if only one any column can be selected from the `table`. This PR fixes #10639. #18233 (Vitaly Baranov).
- Set `charset` to `utf8mb4` when interacting with remote MySQL servers. Fixes #19795. #19800 (alexey-milovidov).
- `S3` table function now supports `auto` compression mode (`autodetect`). This closes #18754. #19793 (Vladimir Chebotarev).
- Correctly output infinite arguments for `formatReadableTimeDelta` function. In previous versions, there was implicit conversion to implementation specific integer value. #19791 (alexey-milovidov).
- Table function `S3` will use global region if the region can't be determined exactly. This closes #10998. #19750 (Vladimir Chebotarev).
- In distributed queries if the setting `async_socket_for_remote` is enabled, it was possible to get stack overflow at least in debug build configuration if very deeply nested data type is used in table (e.g. `Array(Array(Array(...more...))))`). This fixes #19108. This change introduces minor backward incompatibility: excessive parenthesis in type definitions no longer supported, example: `Array((UInt8))`. #19736 (alexey-milovidov).
- Add separate pool for message brokers (RabbitMQ and Kafka). #19722 (Azat Khuzhin).
- Fix rare `max_number_of_merges_with_ttl_in_pool` limit overrun (more merges with TTL can be assigned) for non-replicated MergeTree. #19708 (alesapin).
- Dictionary: better error message during attribute parsing. #19678 (Maksim Kita).
- Add an option to disable validation of checksums on reading. Should never be used in production. Please do not expect any benefits in disabling it. It may only be used for experiments and benchmarks. The setting only applicable for tables of MergeTree family. Checksums are always validated for other table engines and when receiving data over network. In my observations there is no performance difference or it is less than 0.5%. #19588 (alexey-milovidov).
- Support constant result in function `multilf`. #19533 (Maksim Kita).
- Enable function `length/empty/notEmpty` for datatype `Map`, which returns keys number in `Map`. #19530 (taiyang-li).
- Add `--reconnect` option to `clickhouse-benchmark`. When this option is specified, it will reconnect before every request. This is needed for testing. #19872 (alexey-milovidov).
- Support using the new location of `.debug` file. This fixes #19348. #19520 (Amos Bird).
- `toIPv6` function parses `IPv4` addresses. #19518 (Bharat Nallan).
- Add `http_referer` field to `system.query_log`, `system.processes`, etc. This closes #19389. #19390 (alexey-milovidov).
- Improve MySQL compatibility by making more functions case insensitive and adding aliases. #19387 (Daniil Kondratyev).
- Add metrics for MergeTree parts (Wide/Compact/InMemory) types. #19381 (Azat Khuzhin).
- Allow docker to be executed with arbitrary uid. #19374 (filimonov).

- Fix wrong alignment of values of `IPv4` data type in Pretty formats. They were aligned to the right, not to the left. This closes [#19184](#). [#19339](#) ([alexey-milovidov](#)).
- Allow change `max_server_memory_usage` without restart. This closes [#18154](#). [#19186](#) ([alexey-milovidov](#)).
- The exception when function `bar` is called with certain NaN argument may be slightly misleading in previous versions. This fixes [#19088](#). [#19107](#) ([alexey-milovidov](#)).
- Explicitly set uid / gid of clickhouse user & group to the fixed values (101) in clickhouse-server images. [#19096](#) ([filimonov](#)).
- Fixed `.PeekableReadBuffer`: Memory limit exceed error when inserting data with huge strings. Fixes [#18690](#). [#18979](#) ([tavplubix](#)).
- Docker image: several improvements for clickhouse-server entrypoint. [#18954](#) ([filimonov](#)).
- Add `normalizeQueryKeepNames` and `normalizedQueryHashKeepNames` to normalize queries without masking long names with ?. This helps better analyze complex query logs. [#18910](#) ([Amos Bird](#)).
- Check per-block checksum of the distributed batch on the sender before sending (without reading the file twice, the checksums will be verified while reading), this will avoid stuck of the INSERT on the receiver (on truncated .bin file on the sender). Avoid reading .bin files twice for batched INSERT (it was required to calculate rows/bytes to take squashing into account, now this information included into the header, backward compatible is preserved). [#18853](#) ([Azat Khuzhin](#)).
- Fix issues with RIGHT and FULL JOIN of tables with aggregate function states. In previous versions exception about `cloneResized` method was thrown. [#18818](#) ([templarzq](#)).
- Added prefix-based S3 endpoint settings. [#18812](#) ([Vladimir Chebotarev](#)).
- Add [`UInt8`, `UInt16`, `UInt32`, `UInt64`] arguments types support for `bitmapTransform`, `bitmapSubsetInRange`, `bitmapSubsetLimit`, `bitmapContains` functions. This closes [#18713](#). [#18791](#) ([sundyli](#)).
- Allow CTE (Common Table Expressions) to be further aliased. Propagate CSE (Common Subexpressions Elimination) to subqueries in the same level when `enable_global_with_statement = 1`. This fixes [#17378](#). This fixes <https://github.com/ClickHouse/ClickHouse/pull/16575#issuecomment-753416235>. [#18684](#) ([Amos Bird](#)).
- Update librdkafka to v1.6.0-RC2. Fixes [#18668](#). [#18671](#) ([filimonov](#)).
- In case of unexpected exceptions automatically restart background thread which is responsible for execution of distributed DDL queries. Fixes [#17991](#). [#18285](#) ([徐忻](#)).
- Updated AWS C++ SDK in order to utilize global regions in S3. [#17870](#) ([Vladimir Chebotarev](#)).
- Added support for `WITH ... [AND] [PERIODIC] REFRESH [interval_in_sec]` clause when creating LIVE VIEW tables. [#14822](#) ([vzakaznikov](#)).
- Restrict `MODIFY TTL` queries for `MergeTree` tables created in old syntax. Previously the query succeeded, but actually it had no effect. [#19064](#) ([Anton Popov](#)).

## Bug Fix

- Fix index analysis of binary functions with constant argument which leads to wrong query results. This fixes [#18364](#). [#18373](#) ([Amos Bird](#)).
- Fix starting the server with tables having default expressions containing `dictGet()`. Allow getting return type of `dictGet()` without loading dictionary. [#19805](#) ([Vitaly Baranov](#)).

- Fix server crash after query with `if` function with `Tuple` type of then/else branches result. `Tuple` type must contain `Array` or another complex type. Fixes #18356. #20133 (alesapin).
- MaterializeMySQL (experimental feature): Fix replication for statements that update several tables. #20066 (Håvard Kvålen).
- Prevent "Connection refused" in docker during initialization script execution. #20012 (filimonov).
- EmbeddedRocksDB is an experimental storage. Fix the issue with lack of proper type checking. Simplified code. This closes #19967. #19972 (alexey-milovidov).
- Fix a segfault in function `fromModifiedJulianDay` when the argument type is `Nullable(T)` for any integral types other than `Int32`. #19959 (PHO).
- The function `greatCircleAngle` returned inaccurate results in previous versions. This closes #19769. #19789 (alexey-milovidov).
- Fix rare bug when some replicated operations (like mutation) cannot process some parts after data corruption. Fixes #19593. #19702 (alesapin).
- Background thread which executes `ON CLUSTER` queries might hang waiting for dropped replicated table to do something. It's fixed. #19684 (yiguolei).
- Fix wrong deserialization of columns description. It makes `INSERT` into a table with a column named \ impossible. #19479 (alexey-milovidov).
- Mark distributed batch as broken in case of empty data block in one of files. #19449 (Azat Khuzhin).
- Fixed very rare bug that might cause mutation to hang after `DROP/DETACH/REPLACE/MOVE PARTITION`. It was partially fixed by #15537 for the most cases. #19443 (tavplubix).
- Fix possible error `Extremes` transform was already added to pipeline Fixes #14100. #19430 (Nikolai Kochetov).
- Fix default value in join types with non-zero default (e.g. some Enums). Closes #18197. #19360 (vdimir).
- Do not mark file for distributed send as broken on EOF. #19290 (Azat Khuzhin).
- Fix leaking of pipe fd for `async_socket_for_remote`. #19153 (Azat Khuzhin).
- Fix infinite reading from file in `ORC` format (was introduced in #10580). Fixes #19095. #19134 (Nikolai Kochetov).
- Fix issue in merge tree data writer which can lead to marks with bigger size than fixed granularity size. Fixes #18913. #19123 (alesapin).
- Fix startup bug when clickhouse was not able to read compression codec from `LowCardinality(Nullable(...))` and throws exception `Attempt to read after EOF`. Fixes #18340. #19101 (alesapin).
- Simplify the implementation of `tupleHammingDistance`. Support for tuples of any equal length. Fixes #19029. #19084 (Nikolai Kochetov).
- Make sure `groupUniqArray` returns correct type for argument of `Enum` type. This closes #17875. #19019 (alexey-milovidov).
- Fix possible error `Expected single dictionary argument for function` if use function `ignore` with `LowCardinality` argument. Fixes #14275. #19016 (Nikolai Kochetov).
- Fix inserting of `LowCardinality` column to table with `TinyLog` engine. Fixes #18629. #19010 (Nikolai Kochetov).

- Fix minor issue in JOIN: Join tries to materialize const columns, but our code waits for them in other places. [#18982](#) ([Nikita Mikhaylov](#)).
- Disable `optimize_move_functions_out_of_any` because optimization is not always correct. This closes [#18051](#). This closes [#18973](#). [#18981](#) ([alexey-milovidov](#)).
- Fix possible exception `QueryPipeline` stream: different number of columns caused by merging of query plan's Expression steps. Fixes [#18190](#). [#18980](#) ([Nikolai Kochetov](#)).
- Fixed very rare deadlock at shutdown. [#18977](#) ([tavplubix](#)).
- Fixed rare crashes when server run out of memory. [#18976](#) ([tavplubix](#)).
- Fix incorrect behavior when `ALTER TABLE ... DROP PART 'part_name'` query removes all deduplication blocks for the whole partition. Fixes [#18874](#). [#18969](#) ([alesapin](#)).
- Fixed issue [#18894](#) Add a check to avoid exception when long column alias('table.column' style, usually auto-generated by BI tools like Looker) equals to long table name. [#18968](#) ([Daniel Qin](#)).
- Fix error `Task was not found in task queue` (possible only for remote queries, with `async_socket_for_remote = 1`). [#18964](#) ([Nikolai Kochetov](#)).
- Fix bug when mutation with some escaped text (like `ALTER ... UPDATE e = CAST('foo', 'Enum8(\\'foo\' = 1')`) serialized incorrectly. Fixes [#18878](#). [#18944](#) ([alesapin](#)).
- ATTACH PARTITION will reset mutations. [#18804](#). [#18935](#) ([fastio](#)).
- Fix issue with `bitmapOrCardinality` that may lead to nullptr dereference. This closes [#18911](#). [#18912](#) ([sundyli](#)).
- Fixed Attempt to read after eof error when trying to `CAST NULL` from `Nullable(String)` to `Nullable(Decimal(P, S))`. Now function `CAST` returns `NULL` when it cannot parse decimal from nullable string. Fixes [#7690](#). [#18718](#) ([Winter Zhang](#)).
- Fix data type convert issue for MySQL engine. [#18124](#) ([bo zeng](#)).
- Fix clickhouse-client abort exception while executing only `select`. [#19790](#) ([taiyang-li](#)).

## Build/Testing/Packaging Improvement

- Run `SQLancer` (logical SQL fuzzer) in CI. [#19006](#) ([Ilya Yatsishin](#)).
- Query Fuzzer will fuzz newly added tests more extensively. This closes [#18916](#). [#19185](#) ([alexey-milovidov](#)).
- Integrate with `Big List of Naughty Strings` for better fuzzing. [#19480](#) ([alexey-milovidov](#)).
- Add integration tests run with MSan. [#18974](#) ([alesapin](#)).
- Fixed MemorySanitizer errors in cyrus-sasl and musl. [#19821](#) ([Ilya Yatsishin](#)).
- Insufficient arguments check in `positionCaseInsensitiveUTF8` function triggered address sanitizer. [#19720](#) ([alexey-milovidov](#)).
- Remove --project-directory for docker-compose in integration test. Fix logs formatting from docker container. [#19706](#) ([Ilya Yatsishin](#)).
- Made generation of macros.xml easier for integration tests. No more excessive logging from dicttoxml. dicttoxml project is not active for 5+ years. [#19697](#) ([Ilya Yatsishin](#)).
- Allow to explicitly enable or disable watchdog via environment variable `CLICKHOUSE_WATCHDOG_ENABLE`. By default it is enabled if server is not attached to terminal. [#19522](#) ([alexey-milovidov](#)).

- Allow building ClickHouse with Kafka support on arm64. #19369 (filimonov).
- Allow building librdkafka without ssl. #19337 (filimonov).
- Restore Kafka input in FreeBSD builds. #18924 (Alexandre Snarskii).
- Fix potential nullptr dereference in table function VALUES. #19357 (alexey-milovidov).
- Avoid UBSan reports in arrayElement function, substring and arraySum. Fixes #19305. Fixes #19287. This closes #19336. #19347 (alexey-milovidov).

## ClickHouse release 21.1

### ClickHouse release v21.1.3.32-stable, 2021-02-03

#### Bug Fix

- BloomFilter index crash fix. Fixes #19757. #19884 (Maksim Kita).
- Fix crash when pushing down predicates to union distinct subquery. This fixes #19855. #19861 (Amos Bird).
- Fix filtering by UInt8 greater than 127. #19799 (Anton Popov).
- In previous versions, unusual arguments for function arrayEnumerateUniq may cause crash or infinite loop. This closes #19787. #19788 (alexey-milovidov).
- Fixed stack overflow when using accurate comparison of arithmetic type with string type. #19773 (tavplubix).
- Fix crash when nested column name was used in WHERE or PREWHERE. Fixes #19755. #19763 (Nikolai Kochetov).
- Fix a segmentation fault in bitmapAndnot function. Fixes #19668. #19713 (Maksim Kita).
- Some functions with big integers may cause segfault. Big integers is experimental feature. This closes #19667. #19672 (alexey-milovidov).
- Fix wrong result of function neighbor for LowCardinality argument. Fixes #10333. #19617 (Nikolai Kochetov).
- Fix use-after-free of the CompressedWriteBuffer in Connection after disconnect. #19599 (Azat Khuzhin).
- DROP/DETACH TABLE table ON CLUSTER cluster SYNC query might hang, it's fixed. Fixes #19568. #19572 (tavplubix).
- Query CREATE DICTIONARY id expression fix. #19571 (Maksim Kita).
- Fix SIGSEGV with  
merge\_tree\_min\_rows\_for\_concurrent\_read/merge\_tree\_min\_bytes\_for\_concurrent\_read=0/UINT64\_MAX.  
#19528 (Azat Khuzhin).
- Buffer overflow (on memory read) was possible if addMonth function was called with specifically crafted arguments. This fixes #19441. This fixes #19413. #19472 (alexey-milovidov).
- Uninitialized memory read was possible in encrypt/decrypt functions if empty string was passed as IV. This closes #19391. #19397 (alexey-milovidov).
- Fix possible buffer overflow in Uber H3 library. See <https://github.com/uber/h3/issues/392>. This closes #19219. #19383 (alexey-milovidov).

- Fix system.parts \_state column (LOGICAL\_ERROR when querying this column, due to incorrect order). [#19346](#) ([Azat Khuzhin](#)).
- Fixed possible wrong result or segfault on aggregation when Materialized View and its target table have different structure. Fixes [#18063](#). [#19322](#) ([tavplubix](#)).
- Fix error Cannot convert column now64() because it is constant but values of constants are different in source and result. Continuation of [#7156](#). [#19316](#) ([Nikolai Kochetov](#)).
- Fix bug when concurrent `ALTER` and `DROP` queries may hang while processing ReplicatedMergeTree table. [#19237](#) ([alesapin](#)).
- Fixed There is no checkpoint error when inserting data through http interface using `Template` or `CustomSeparated` format. Fixes [#19021](#). [#19072](#) ([tavplubix](#)).
- Disable constant folding for subqueries on the analysis stage, when the result cannot be calculated. [#18446](#) ([Azat Khuzhin](#)).
- Mutation might hang waiting for some non-existent part after `MOVE` or `REPLACE PARTITION` or, in rare cases, after `DETACH` or `DROP PARTITION`. It's fixed. [#15537](#) ([tavplubix](#)).

## ClickHouse release v21.1.2.15-stable 2021-01-18

### Backward Incompatible Change

- The setting `input_format_null_as_default` is enabled by default. [#17525](#) ([alexey-milovidov](#)).
- Check settings constraints for profile settings from config. Server will fail to start if `users.xml` contain settings that do not meet corresponding constraints. [#18486](#) ([tavplubix](#)).
- Restrict `ALTER MODIFY SETTING` from changing storage settings that affects data parts (`write_final_mark` and `enable_mixed_granularity_parts`). [#18306](#) ([Amos Bird](#)).
- Set `insert_quorum_parallel` to 1 by default. It is significantly more convenient to use than "sequential" quorum inserts. But if you rely to sequential consistency, you should set the setting back to zero. [#17567](#) ([alexey-milovidov](#)).
- Remove `sumburConsistentHash` function. This closes [#18120](#). [#18656](#) ([alexey-milovidov](#)).
- Removed aggregate functions `timeSeriesGroupSum`, `timeSeriesGroupRateSum` because a friend of mine said they never worked. This fixes [#16869](#). If you have luck using these functions, write a email to [clickhouse-feedback@yandex-team.com](mailto:clickhouse-feedback@yandex-team.com). [#17423](#) ([alexey-milovidov](#)).
- Prohibit `toUnixTimestamp(Date())` (before it just returns `UInt16` representation of Date). [#17376](#) ([Azat Khuzhin](#)).
- Allow using extended integer types (`Int128`, `Int256`, `UInt256`) in `avg` and `avgWeighted` functions. Also allow using different types (integer, decimal, floating point) for value and for weight in `avgWeighted` function. This is a backward-incompatible change: now the `avg` and `avgWeighted` functions always return `Float64` (as documented). Before this change the return type for `Decimal` arguments was also `Decimal`. [#15419](#) ([Mike](#)).
- Expression `toUUID(N)` no longer works. Replace with `toUUID('00000000-0000-0000-0000-000000000000')`. This change is motivated by non-obvious results of `toUUID(N)` where N is non zero.
- SSL Certificates with incorrect "key usage" are rejected. In previous versions they are used to work. See [#19262](#).

- incl references to substitutions file (`/etc/metrika.xml`) were removed from the default config (`<remote_servers>`, `<zookeeper>`, `<macros>`, `<compression>`, `<networks>`). If you were using substitutions file and were relying on those implicit references, you should put them back manually and explicitly by adding corresponding sections with `incl="..."` attributes before the update. See #18740 (alexey-milovidov).

## New Feature

- Implement gRPC protocol in ClickHouse. #15111 (Vitaly Baranov).
- Allow to use multiple zookeeper clusters. #17070 (fastio).
- Implemented `REPLACE TABLE` and `CREATE OR REPLACE TABLE` queries. #18521 (tavplubix).
- Implement `UNION DISTINCT` and treat the plain `UNION` clause as `UNION DISTINCT` by default. Add a setting `union_default_mode` that allows to treat it as `UNION ALL` or require explicit mode specification. #16338 (flynn).
- Added function `accurateCastOrNull`. This closes #10290. Add type conversions in `x IN (subquery)` expressions. This closes #10266. #16724 (Maksim Kita).
- IP Dictionary supports `IPv4` / `IPv6` types directly. #17571 (vdimir).
- IP Dictionary supports key fetching. Resolves #18241. #18480 (vdimir).
- Add `*.zst` compression/decompression support for data import and export. It enables using `*.zst` in `file()` function and `Content-encoding: zstd` in HTTP client. This closes #16791 . #17144 (Abi Palagashvili).
- Added `mannWitneyUTest`, `studentTTest` and `welchTTest` aggregate functions. Refactored `rankCorr` a bit. #16883 (Nikita Mikhaylov).
- Add functions `countMatches`/`countMatchesCaseInsensitive`. #17459 (Azat Khuzhin).
- Implement `countSubstrings()`/`countSubstringsCaseInsensitive()`/`countSubstringsCaseInsensitiveUTF8()` (Count the number of substring occurrences). #17347 (Azat Khuzhin).
- Add information about used databases, tables and columns in `system.query_log`. Add `query_kind` and `normalized_query_hash` fields. #17726 (Amos Bird).
- Add a setting `optimize_on_insert`. When enabled, do the same transformation for `INSERTED` block of data as if `merge` was done on this block (e.g. Replacing, Collapsing, Aggregating...). This setting is enabled by default. This can influence Materialized View and MaterializeMySQL behaviour (see detailed description). This closes #10683. #16954 (Kruglov Pavel).
- Kerberos Authentication for HDFS. #16621 (Ilya Golshtain).
- Support `SHOW SETTINGS` statement to show parameters in `system.settings`. `SHOW CHANGED SETTINGS` and `LIKE/ILIKE` clause are also supported. #18056 (Jianmei Zhang).
- Function `position` now supports `POSITION(needle IN haystack)` syntax for SQL compatibility. This closes #18701. ... #18779 (Jianmei Zhang).
- Now we have a new storage setting `max_partitions_to_read` for tables in the MergeTree family. It limits the max number of partitions that can be accessed in one query. A user setting `force_max_partition_limit` is also added to enforce this constraint. #18712 (Amos Bird).
- Add `query_id` column to `system.part_log` for inserted parts. Closes #10097. #18644 (flynn).
- Allow create table as select with columns specification. Example `CREATE TABLE t1 (x String) ENGINE = Memory AS SELECT 1;`. #18060 (Maksim Kita).

- Added arrayMin, arrayMax, arrayAvg aggregation functions. #18032 (Maksim Kita).
- Implemented ATTACH TABLE name FROM 'path/to/data/' (col1 Type1, ... query. It creates new table with provided structure and attaches table data from provided directory in user\_files. #17903 (tavplubix).
- Add mutation support for StorageMemory. This closes #9117. #15127 (flynn).
- Support syntax EXISTS DATABASE name. #18458 (Du Chuan).
- Support builtin function isIPv4String && isIPv6String like MySQL. #18349 (Du Chuan).
- Add a new setting insert\_distributed\_one\_random\_shard = 1 to allow insertion into multi-sharded distributed table without any distributed key. #18294 (Amos Bird).
- Add settings min\_compress\_block\_size and max\_compress\_block\_size to MergeTreeSettings, which have higher priority than the global settings and take effect when they are set. close 13890. #17867 (flynn).
- Add support for 64bit roaring bitmaps. #17858 (Andy Yang).
- Extended OPTIMIZE ... DEDUPLICATE syntax to allow explicit (or implicit with asterisk/column transformers) list of columns to check for duplicates on. ... #17846 (Vasily Nemkov).
- Added functions toModifiedJulianDay, fromModifiedJulianDay, toModifiedJulianDayOrNull, and fromModifiedJulianDayOrNull. These functions convert between Proleptic Gregorian calendar date and Modified Julian Day number. #17750 (PHO).
- Add ability to use custom TLD list: added functions firstSignificantSubdomainCustom, cutToFirstSignificantSubdomainCustom. #17748 (Azat Khuzhin).
- Add support for PROXYv1 protocol to wrap native TCP interface. Allow quotas to be keyed by proxy-forwarded IP address (applied for PROXYv1 address and for X-Forwarded-For from HTTP interface). This is useful when you provide access to ClickHouse only via trusted proxy (e.g. CloudFlare) but want to account user resources by their original IP addresses. This fixes #17268. #17707 (alexey-milovidov).
- Now clickhouse-client supports opening EDITOR to edit commands. Alt-Shift-E. #17665 (Amos Bird).
- Add function encodeXMLComponent to escape characters to place string into XML text node or attribute. #17659 (nauta).
- Introduce DETACH TABLE/VIEW ... PERMANENTLY syntax, so that after restarting the table does not reappear back automatically on restart (only by explicit request). The table can still be attached back using the short syntax ATTACH TABLE. Implements #5555. Fixes #13850. #17642 (filimonov).
- Add asynchronous metrics on total amount of rows, bytes and parts in MergeTree tables. This fix #11714. #17639 (flynn).
- Add settings limit and offset for out-of-SQL pagination: #16176 They are useful for building APIs. These two settings will affect SELECT query as if it is added like select \* from (your\_original\_select\_query) t limit xxx offset xxxx;. #17633 (hexiaoting).
- Provide a new aggregator combinator : -SimpleState to build SimpleAggregateFunction types via query. It's useful for defining MaterializedView of AggregatingMergeTree engine, and will benefit projections too. #16853 (Amos Bird).
- Added queries-file parameter for clickhouse-client and clickhouse-local. #15930 (Maksim Kita).
- Added query parameter for clickhouse-benchmark. #17832 (Maksim Kita).
- EXPLAIN AST now support queries other then SELECT. #18136 (taiyang-li).

## Experimental Feature

- Added functions for calculation of minHash and simHash of text n-grams and shingles. They are intended for semi-duplicate search. Also functions `bitHammingDistance` and `tupleHammingDistance` are added. #7649 (flynn).
- Add new data type `Map`. See #1841. First version for `Map` only supports `String` type of key and value. #15806 (hexiaoting).
- Implement alternative SQL parser based on ANTLR4 runtime and generated from EBNF grammar. #11298 (ivan).

## Performance Improvement

- New IP Dictionary implementation with lower memory consumption, improved performance for some cases, and fixed bugs. #16804 (vdimir).
- Parallel formatting for data export. #11617 (Nikita Mikhaylov).
- LDAP integration: Added `verification_cooldown` parameter in LDAP server connection configuration to allow caching of successful "bind" attempts for configurable period of time. #15988 (Denis Glazachev).
- Add `--no-system-table` option for `clickhouse-local` to run without system tables. This avoids initialization of `DateLUT` that may take noticeable amount of time (tens of milliseconds) at startup. #18899 (alexey-milovidov).
- Replace `PODArray` with `PODArrayWithStackMemory` in `AggregateFunctionWindowFunnelData` to improve `windowFunnel` function performance. #18817 (flynn).
- Don't send empty blocks to shards on synchronous `INSERT` into Distributed table. This closes #14571. #18775 (alexey-milovidov).
- Optimized read for `StorageMemory`. #18052 (Maksim Kita).
- Using Dragonbox algorithm for float to string conversion instead of ryu. This improves performance of float to string conversion significantly. #17831 (Maksim Kita).
- Speedup `IPv6CIDRToRange` implementation. #17569 (vdimir).
- Add `remerge_sort_lowered_memory_bytes_ratio` setting (If memory usage after remerge does not reduced by this ratio, remerge will be disabled). #17539 (Azat Khuzhin).
- Improve performance of `AggregatingMergeTree` with `SimpleAggregateFunction(String)` in PK. #17109 (Azat Khuzhin).
- Now the `-If` combinator is devirtualized, and `count` is properly vectorized. It is for this PR. #17043 (Amos Bird).
- Fix performance of reading from `Merge` tables over huge number of `MergeTree` tables. Fixes #7748. #16988 (Anton Popov).
- Improved performance of function `repeat`. #16937 (satanson).
- Slightly improved performance of float parsing. #16809 (Maksim Kita).
- Add possibility to skip merged partitions for `OPTIMIZE TABLE ... FINAL`. #15939 (Kruglov Pavel).
- Integrate with `fast_float` from Daniel Lemire to parse floating point numbers. #16787 (Maksim Kita). It is not enabled, because performance its performance is still lower than rough float parser in ClickHouse.
- Fix `max_distributed_connections` (affects `prefer_localhost_replica = 1` and `max_threads != max_distributed_connections`). #17848 (Azat Khuzhin).

- Adaptive choice of single/multi part upload when sending data to S3. Single part upload is controlled by a new setting `max_single_part_upload_size`. #17934 (Pavel Kovalenko).
- Support for async tasks in `PipelineExecutor`. Initial support of async sockets for remote queries. #17868 (Nikolai Kochetov).
- Allow to use `optimize_move_to_prewhere` optimization with compact parts, when sizes of columns are unknown. #17330 (Anton Popov).

## Improvement

- Avoid deadlock when executing `INSERT SELECT` into itself from a table with `TinyLog` or `Log` table engines. This closes #6802. This closes #18691. This closes #16812. This closes #14570. #15260 (alexey-milovidov).
- Support `SHOW CREATE VIEW name` syntax like MySQL. #18095 (Du Chuan).
- All queries of type `Decimal * Float` or vice versa are allowed, including aggregate ones (e.g. `SELECT sum(decimal_field * 1.1)` or `SELECT dec_col * float_col`), the result type is `Float32` or `Float64`. #18145 (Mike).
- Improved minimal Web UI: add history; add sharing support; avoid race condition of different requests; add request in-flight and ready indicators; add favicon; detect Ctrl+Enter if textarea is not in focus. #17293 #17770 (alexey-milovidov).
- clickhouse-server didn't send `close` request to ZooKeeper server. #16837 (alesapin).
- Avoid server abnormal termination in case of too low memory limits (`max_memory_usage = 1` / `max_untracked_memory = 1`). #17453 (Azat Khuzhin).
- Fix non-deterministic result of `windowFunnel` function in case of same timestamp for different events. #18884 (Fuwang Hu).
- Docker: Explicitly set uid / gid of clickhouse user & group to the fixed values (101) in clickhouse-server Docker images. #19096 (filimonov).
- Asynchronous INSERTs to Distributed tables: Two new settings (by analogy with MergeTree family) has been added: - `fsync_after_insert` - Do fsync for every inserted. Will decreases performance of inserts. - `fsync_directories` - Do fsync for temporary directory (that is used for async INSERT only) after all operations (writes, renames, etc.). #18864 (Azat Khuzhin).
- `SYSTEM KILL` command started to work in Docker. This closes #18847. #18848 (alexey-milovidov).
- Expand macros in the zk path when executing `FETCH PARTITION`. #18839 (fastio).
- Apply `ALTER TABLE <replicated_table> ON CLUSTER MODIFY SETTING ...` to all replicas. Because we don't replicate such alter commands. #18789 (Amos Bird).
- Allow column transformer `EXCEPT` to accept a string as regular expression matcher. This resolves #18685 . #18699 (Amos Bird).
- Fix SimpleAggregateFunction in SummingMergeTree. Now it works like AggregateFunction. In previous versions values were summed together regardless to the aggregate function. This fixes #18564 . #8052. #18637 (Amos Bird). Another fix of using SimpleAggregateFunction in SummingMergeTree. This fixes #18676 . #18677 (Amos Bird).
- Fixed assertion error inside allocator in case when last argument of function bar is NaN. Now simple ClickHouse's exception is being thrown. This fixes #17876. #18520 (Nikita Mikhaylov).
- Fix usability issue: no newline after exception message in some tools. #18444 (alexey-milovidov).

- Add ability to modify primary and partition key column type from `LowCardinality(Type)` to `Type` and vice versa. Also add an ability to modify primary key column type from `EnumX` to `IntX` type. Fixes #5604. #18362 (alesapin).
- Implement `untuple` field access. #18133. #18309 (hexiaoting).
- Allow to parse Array fields from CSV if it is represented as a string containing array that was serialized as nested CSV. Example: `"[\"Hello\", \"world\", \"42\" TV"]"` will parse as `['Hello', 'world', '42' TV']`. Allow to parse array in CSV in a string without enclosing braces. Example: `"Hello, 'world', '42' TV"` will parse as `['Hello', 'world', '42' TV']`. #18271 (alexey-milovidov).
- Make better adaptive granularity calculation for merge tree wide parts. #18223 (alesapin).
- Now `clickhouse install` could work on Mac. The problem was that there is no procfs on this platform. #18201 (Nikita Mikhaylov).
- Better hints for `SHOW ...` query syntax. #18183 (Du Chuan).
- Array aggregation `arrayMin`, `arrayMax`, `arraySum`, `arrayAvg` support for `Int128`, `Int256`, `UInt256`. #18147 (Maksim Kita).
- Add `disk` to Set and Join storage settings. #18112 (Grigory Pervakov).
- Access control: Now table function `merge()` requires current user to have `SELECT` privilege on each table it receives data from. This PR fixes #16964. #18104 #17983 (Vitaly Baranov).
- Temporary tables are visible in the system tables `system.tables` and `system.columns` now only in those session where they have been created. The internal database `_temporary_and_external_tables` is now hidden in those system tables; temporary tables are shown as tables with empty database with the `is_temporary` flag set instead. #18014 (Vitaly Baranov).
- Fix clickhouse-client rendering issue when the size of terminal window changes. #18009 (Amos Bird).
- Decrease log verbosity of the events when the client drops the connection from Warning to Information. #18005 (filimonov).
- Forcibly removing empty or bad metadata files from filesystem for DiskS3. S3 is an experimental feature. #17935 (Pavel Kovalenko).
- Access control: `allow_introspection_functions=0` prohibits usage of introspection functions but doesn't prohibit giving grants for them anymore (the grantee will need to set `allow_introspection_functions=1` for himself to be able to use that grant). Similarly `allow_ddl=0` prohibits usage of DDL commands but doesn't prohibit giving grants for them anymore. #17908 (Vitaly Baranov).
- Usability improvement: hints for column names. #17112. #17857 (fastio).
- Add diagnostic information when two merge tables try to read each other's data. #17854 (徐忻).
- Let the possibility to override timeout value for running script using the ClickHouse docker image. #17818 (Guillaume Tassery).
- Check system log tables' engine definition grammar to prevent some configuration errors. Notes that this grammar check is not semantical, that means such mistakes as non-existent columns / expression functions would be not found out until the table is created. #17739 (Du Chuan).
- Removed exception throwing at RabbitMQ table initialization if there was no connection (it will be reconnecting in the background). #17709 (Kseniia Sumarokova).
- Do not ignore server memory limits during Buffer flush. #17646 (Azat Khuzhin).

- Switch to patched version of RocksDB (from ClickHouse-Extras) to fix use-after-free error. [#17643](#) ([Nikita Mikhaylov](#)).
- Added an offset to exception message for parallel parsing. This fixes [#17457](#). [#17641](#) ([Nikita Mikhaylov](#)).
- Don't throw "Too many parts" error in the middle of INSERT query. [#17566](#) ([alexey-milovidov](#)).
- Allow query parameters in UPDATE statement of ALTER query. Fixes [#10976](#). [#17563](#) ([alexey-milovidov](#)).
- Query obfuscator: avoid usage of some SQL keywords for identifier names. [#17526](#) ([alexey-milovidov](#)).
- Export current max ddl entry executed by DDLWorker via server metric. It's useful to check if DDLWorker hangs somewhere. [#17464](#) ([Amos Bird](#)).
- Export asynchronous metrics of all servers current threads. It's useful to track down issues like [this](#). [#17463](#) ([Amos Bird](#)).
- Include dynamic columns like MATERIALIZED / ALIAS for wildcard query when settings `asterisk_include_materialized_columns` and `asterisk_include_alias_columns` are turned on. [#17462](#) ([Ken Chen](#)).
- Allow specifying TTL to remove old entries from [system log tables](#), using the `<ttl>` attribute in `config.xml`. [#17438](#) ([Du Chuan](#)).
- Now queries coming to the server via MySQL and PostgreSQL protocols have distinctive interface types (which can be seen in the `interface` column of the `tablesystem.query_log`): 4 for MySQL, and 5 for PostgreSQL, instead of formerly used 1 which is now used for the native protocol only. [#17437](#) ([Vitaly Baranov](#)).
- Fix parsing of SETTINGS clause of the `INSERT ... SELECT ... SETTINGS` query. [#17414](#) ([Azat Khuzhin](#)).
- Correctly account memory in RadixSort. [#17412](#) ([Nikita Mikhaylov](#)).
- Add eof check in `receiveHello` in server to prevent getting Attempt to read after eof exception. [#17365](#) ([Kruglov Pavel](#)).
- Avoid possible stack overflow in bigint conversion. Big integers are experimental. [#17269](#) ([flynn](#)).
- Now `set` indices will work with `GLOBAL IN`. This fixes [#17232](#) , [#5576](#) . [#17253](#) ([Amos Bird](#)).
- Add limit for http redirects in request to S3 storage (`s3_max_redirects`). [#17220](#) ([ianton-ru](#)).
- When `-OrNull` combinator combined `-If`, `-Merge`, `-MergeState`, `-State` combinators, we should put `-OrNull` in front. [#16935](#) ([flynn](#)).
- Support HTTP proxy and HTTPS S3 endpoint configuration. [#16861](#) ([Pavel Kovalenko](#)).
- Added proper authentication using environment, `~/.aws` and `AssumeRole` for S3 client. [#16856](#) ([Vladimir Chebotarev](#)).
- Add more OpenTelemetry spans. Add an example of how to export the span data to Zipkin. [#16535](#) ([Alexander Kuzmenkov](#)).
- Cache dictionaries: Completely eliminate callbacks and locks for acquiring them. Keys are not divided into "not found" and "expired", but stored in the same map during query. [#14958](#) ([Nikita Mikhaylov](#)).
- Fix never worked `fsync_part_directory/fsync_after_insert/in_memory_parts_insert_sync` (experimental feature). [#18845](#) ([Azat Khuzhin](#)).
- Allow using `Atomic` engine for nested database of `MaterializeMySQL` engine. [#14849](#) ([tavplubix](#)).

## Bug Fix

- Fix the issue when server can stop accepting connections in very rare cases. #17542 (Amos Bird, [alexey-milovidov](#)).
- Fix index analysis of binary functions with constant argument which leads to wrong query results. This fixes #18364. #18373 (Amos Bird).
- Fix possible wrong index analysis when the types of the index comparison are different. This fixes #17122. #17145 (Amos Bird).
- Disable write with AIO during merges because it can lead to extremely rare data corruption of primary key columns during merge. #18481 ([alesapin](#)).
- Restrict merges from wide to compact parts. In case of vertical merge it led to broken result part. #18381 (Anton Popov).
- Fix possible incomplete query result while reading from `MergeTree*` in case of read backoff (message `<Debug> MergeTreeReadPool: Will lower number of threads in logs`). Was introduced in #16423. Fixes #18137. #18216 (Nikolai Kochetov).
- Fix use after free bug in `rocksdb` library. #18862 ([sundyli](#)).
- Fix infinite reading from file in `ORC` format (was introduced in #10580). Fixes #19095. #19134 (Nikolai Kochetov).
- Fix bug in merge tree data writer which can lead to marks with bigger size than fixed granularity size. Fixes #18913. #19123 ([alesapin](#)).
- Fix startup bug when clickhouse was not able to read compression codec from `LowCardinality(Nullable(...))` and throws exception `Attempt to read after EOF`. Fixes #18340. #19101 ([alesapin](#)).
- Restrict `MODIFY TTL` queries for `MergeTree` tables created in old syntax. Previously the query succeeded, but actually it had no effect. #19064 (Anton Popov).
- Make sure `groupUniqArray` returns correct type for argument of `Enum` type. This closes #17875. #19019 ([alexey-milovidov](#)).
- Fix possible error `Expected single dictionary argument for function ignore` with `LowCardinality` argument. Fixes #14275. #19016 (Nikolai Kochetov).
- Fix inserting of `LowCardinality` column to table with `TinyLog` engine. Fixes #18629. #19010 (Nikolai Kochetov).
- Join tries to materialize const columns, but our code wants them in other places. #18982 (Nikita Mikhaylov).
- Disable `optimize_move_functions_out_of_any` because optimization is not always correct. This closes #18051. This closes #18973. #18981 ([alexey-milovidov](#)).
- Fix possible exception `QueryPipeline` stream: different number of columns caused by merging of query plan's `Expression` steps. Fixes #18190. #18980 (Nikolai Kochetov).
- Fixed very rare deadlock at shutdown. #18977 ([tavplubix](#)).
- Fix incorrect behavior when `ALTER TABLE ... DROP PART 'part_name'` query removes all deduplication blocks for the whole partition. Fixes #18874. #18969 ([alesapin](#)).
- Attach partition should reset the mutation. #18804. #18935 ([fastio](#)).

- Fix issue with `bitmapOrCardinality` that may lead to `nullptr` dereference. This closes #18911. #18912 (sundyli).
- Fix possible hang at shutdown in `clickhouse-local`. This fixes #18891. #18893 (alexey-milovidov).
- Queries for external databases (MySQL, ODBC, JDBC) were incorrectly rewritten if there was an expression in form of `x IN table`. This fixes #9756. #18876 (alexey-milovidov).
- Fix \*If combinator with unary function and Nullable types. #18806 (Azat Khuzhin).
- Fix the issue that asynchronous distributed INSERTs can be rejected by the server if the setting `network_compression_method` is globally set to non-default value. This fixes #18741. #18776 (alexey-milovidov).
- Fixed Attempt to read after eof error when trying to `CAST NULL` from `Nullable(String)` to `Nullable(Decimal(P, S))`. Now function `CAST` returns `NULL` when it cannot parse decimal from nullable string. Fixes #7690. #18718 (Winter Zhang).
- Fix minor issue with logging. #18717 (sundyli).
- Fix removing of empty parts in `ReplicatedMergeTree` tables, created with old syntax. Fixes #18582. #18614 (Anton Popov).
- Fix previous bug when date overflow with different values. Strict Date value limit to "2106-02-07", cast date > "2106-02-07" to value 0. #18565 (hexiaoting).
- Add `FixedString` data type support for replication from MySQL. Replication from MySQL is an experimental feature. This patch fixes #18450 Also fixes #6556. #18553 (awesomeleo).
- Fix possible Pipeline stuck error while using `ORDER BY` after subquery with `RIGHT` or `FULL` join. #18550 (Nikolai Kochetov).
- Fix bug which may lead to `ALTER` queries hung after corresponding mutation kill. Found by thread fuzzer. #18518 (alesapin).
- Proper support for 12AM in `parseDateTimeBestEffort` function. This fixes #18402. #18449 (vladimir-golovchenko).
- Fixed value is too short error when executing `toType(...)` functions (`toDate`, `toUInt32`, etc) with argument of type `Nullable(String)`. Now such functions return `NULL` on parsing errors instead of throwing exception. Fixes #7673. #18445 (tavplubix).
- Fix the unexpected behaviour of `SHOW TABLES`. #18431 (fastio).
- Fix -SimpleState combinator generates incompatible arugment type and return type. #18404 (Amos Bird).
- Fix possible race condition in concurrent usage of `Set` or `Join` tables and selects from `system.tables`. #18385 (alexey-milovidov).
- Fix filling table `system.settings_profile_elements`. This PR fixes #18231. #18379 (Vitaly Baranov).
- Fix possible crashes in aggregate functions with combinator `Distinct`, while using two-level aggregation. Fixes #17682. #18365 (Anton Popov).
- Fixed issue when `clickhouse-odbc-bridge` process is unreachable by server on machines with dual IPv4/IPv6 stack; Fixed issue when ODBC dictionary updates are performed using malformed queries and/or cause crashes of the odbc-bridge process; Possibly closes #14489. #18278 (Denis Glazachev).
- Access control: `SELECT count() FROM table` now can be executed if the user has access to at least single column from a table. This PR fixes #10639. #18233 (Vitaly Baranov).

- Access control: `SELECT JOIN` now requires the `SELECT` privilege on each of the joined tables. This PR fixes #17654. #18232 (Vitaly Baranov).
- Fix key comparison between `Enum` and `Int` types. This fixes #17989. #18214 (Amos Bird).
- Replication from MySQL (experimental feature). Fixes #18186 Fixes #16372 Fix unique key convert issue in MaterializeMySQL database engine. #18211 (Winter Zhang).
- Fix inconsistency for queries with both `WITH FILL` and `WITH TIES` #17466. #18188 (hexiaoting).
- Fix inserting a row with default value in case of parsing error in the last column. Fixes #17712. #18182 (Jianmei Zhang).
- Fix Unknown setting profile error on attempt to set settings profile. #18167 (tavplubix).
- Fix error when query `MODIFY COLUMN ... REMOVE TTL` doesn't actually remove column TTL. #18130 (alesapin).
- Fixed `std::out_of_range: basic_string` in S3 URL parsing. #18059 (Vladimir Chebotarev).
- Fix comparison of `DateTime64` and `Date`. Fixes #13804 and #11222. ... #18050 (Vasily Nemkov).
- Replication from MySQL (experimental feature): Fixes #15187 Fixes #17912 support convert MySQL prefix index for MaterializeMySQL. #17944 (Winter Zhang).
- When server log rotation was configured using `logger.size` parameter with numeric value larger than  $2^{32}$ , the logs were not rotated properly. This is fixed. #17905 (Alexander Kuzmenkov).
- Trivial query optimization was producing wrong result if query contains `ARRAY JOIN` (so query is actually non trivial). #17887 (sundyli).
- Fix possible segfault in `topK` aggregate function. This closes #17404. #17845 (Maksim Kita).
- WAL (experimental feature): Do not restore parts from WAL if `in_memory_parts_enable_wal` is disabled. #17802 (detaiyang).
- Exception message about max table size to drop was displayed incorrectly. #17764 (alexey-milovidov).
- Fixed possible segfault when there is not enough space when inserting into `Distributed` table. #17737 (tavplubix).
- Fixed problem when ClickHouse fails to resume connection to MySQL servers. #17681 (Alexander Kazakov).
- Windows: Fixed Function not implemented error when executing `RENAME` query in Atomic database with ClickHouse running on Windows Subsystem for Linux. Fixes #17661. #17664 (tavplubix).
- In might be determined incorrectly if cluster is circular- (cross-) replicated or not when executing `ON CLUSTER` query due to race condition when `pool_size > 1`. It's fixed. #17640 (tavplubix).
- Fix empty `system.stack_trace` table when server is running in daemon mode. #17630 (Amos Bird).
- Exception `fmt::v7::format_error` can be logged in background for MergeTree tables. This fixes #17613. #17615 (alexey-milovidov).
- When clickhouse-client is used in interactive mode with multiline queries, single line comment was erroneously extended till the end of query. This fixes #13654. #17565 (alexey-milovidov).
- Fix alter query hang when the corresponding mutation was killed on the different replica. Fixes #16953. #17499 (alesapin).

- Fix issue with memory accounting when mark cache size was underestimated by clickhouse. It may happen when there are a lot of tiny files with marks. #17496 (alesapin).
- Fix ORDER BY with enabled setting `optimize_redundant_functions_in_order_by`. #17471 (Anton Popov).
- Fix duplicates after DISTINCT which were possible because of incorrect optimization. Fixes #17294. #17296 (li chengxiang). #17439 (Nikolai Kochetov).
- Fixed high CPU usage in background tasks of \*MergeTree tables. #17416 (tavplubix).
- Fix possible crash while reading from JOIN table with LowCardinality types. Fixes #17228. #17397 (Nikolai Kochetov).
- Replication from MySQL (experimental feature): Fixes #16835 try fix miss match header with MySQL SHOW statement. #17366 (Winter Zhang).
- Fix nondeterministic functions with predicate optimizer. This fixes #17244. #17273 (Winter Zhang).
- Fix possible Unexpected packet Data received from clienterror for Distributed queries with LIMIT. #17254 (Azat Khuzhin).
- Fix set index invalidation when there are const columns in the subquery. This fixes #17246. #17249 (Amos Bird).
- clickhouse-copier: Fix for non-partitioned tables #15235. #17248 (Qi Chen).
- Fixed possible not-working mutations for parts stored on S3 disk (experimental feature). #17227 (Pavel Kovalenko).
- Bug fix for funciton `fuzzBits`, related issue: #16980. #17051 (hexiaoting).
- Fix `optimize_distributed_group_by_sharding_key` for query with OFFSET only. #16996 (Azat Khuzhin).
- Fix queries from Merge tables over Distributed tables with JOINs. #16993 (Azat Khuzhin).
- Fix order by optimization with monotonic functions. Fixes #16107. #16956 (Anton Popov).
- Fix incorrect comparison of types `DateTime64` with different scales. Fixes #16655 ... #16952 (Vasily Nemkov).
- Fix optimization of group by with enabled setting `optimize_aggregators_of_group_by_keys` and joins. Fixes #12604. #16951 (Anton Popov).
- Minor fix in SHOW ACCESS query. #16866 (tavplubix).
- Fix the behaviour with enabled `optimize_trivial_count_query` setting with partition predicate. #16767 (Azat Khuzhin).
- Return number of affected rows for INSERT queries via MySQL wire protocol. Previously ClickHouse used to always return 0, it's fixed. Fixes #16605. #16715 (Winter Zhang).
- Fix inconsistent behavior caused by `select_sequential_consistency` for optimized trivial count query and system tables. #16309 (Hao Chen).
- Throw error when `REPLACE` column transformer operates on non existing column. #16183 (hexiaoting).
- Throw exception in case of not equi-join ON expression in RIGH|FULL JOIN. #15162 (Artem Zuikov).

## Build/Testing/Packaging Improvement

- Add simple integrity check for ClickHouse binary. It allows to detect corruption due to faulty hardware (bit rot on storage media or bit flips in RAM). #18811 (alexey-milovidov).

- Change `OpenSSL` to `BoringSSL`. It allows to avoid issues with sanitizers. This fixes #12490. This fixes #17502. This fixes #12952. #18129 (alexey-milovidov).
- Simplify `Sys/V` init script. It was not working on Ubuntu 12.04 or older. #17428 (alexey-milovidov).
- Multiple improvements in `./clickhouse install` script. #17421 (alexey-milovidov).
- Now ClickHouse can pretend to be a fake ZooKeeper. Currently, storage implementation is just stored in-memory hash-table, and server partially support ZooKeeper protocol. #16877 (alesapin).
- Fix dead list watches removal for `TestKeeperStorage` (a mock for ZooKeeper). #18065 (alesapin).
- Add `SYSTEM SUSPEND` command for fault injection. It can be used to facilitate failover tests. This closes #15979. #18850 (alexey-milovidov).
- Generate build id when ClickHouse is linked with `lld`. It's appeared that `lld` does not generate it by default on my machine. Build id is used for crash reports and introspection. #18808 (alexey-milovidov).
- Fix shellcheck errors in style check. #18566 (Ilya Yatsishin).
- Update timezones info to 2020e. #18531 (alesapin).
- Fix codespell warnings. Split style checks into separate parts. Update style checks docker image. #18463 (Ilya Yatsishin).
- Automated check for leftovers of conflict markers in docs. #18332 (alexey-milovidov).
- Enable Thread Fuzzer for stateless tests flaky check. #18299 (alesapin).
- Do not use non thread-safe function `strerror`. #18204 (alexey-milovidov).
- Update `anchore/scan-action@main` workflow action (was moved from `master` to `main`). #18192 (Stig Bakken).
- Now `clickhouse-test` does `DROP/CREATE` databases with a timeout. #18098 (alesapin).
- Enable experimental support for Pytest framework for stateless tests. #17902 (Ivan).
- Now we use the fresh docker daemon version in integration tests. #17671 (alesapin).
- Send info about official build, memory, cpu and free disk space to Sentry if it is enabled. Sentry is opt-in feature to help ClickHouse developers. This closes #17279. #17543 (alexey-milovidov).
- There was an uninitialized variable in the code of `clickhouse-copier`. #17363 (Nikita Mikhaylov).
- Fix one MSan report from #17309. #17344 (Nikita Mikhaylov).
- Fix for the issue with IPv6 in Arrow Flight library. See the comments for details. #16664 (Zhanna).
- Add a library that replaces some `libc` functions to traps that will terminate the process. #16366 (alexey-milovidov).
- Provide diagnostics in server logs in case of stack overflow, send error message to `clickhouse-client`. This closes #14840. #16346 (alexey-milovidov).
- Now we can run almost all stateless functional tests in parallel. #15236 (alesapin).
- Fix corruption in `librdkafka` snappy decompression (was a problem only for gcc10 builds, but official builds uses clang already, so at least recent official releases are not affected). #18053 (Azat Khuzhin).
- If server was terminated by OOM killer, print message in log. #13516 (alexey-milovidov).
- PODArray: Avoid call to `memcpy` with `(nullptr, 0)` arguments (Fix UBSan report). This fixes #18525. #18526 (alexey-milovidov).

- Minor improvement for path concatenation of zookeeper paths inside DDLWorker. [#17767](#) ([Bharat Nallan](#)).
- Allow to reload symbols from debug file. This PR also fixes a build-id issue. [#17637](#) ([Amos Bird](#)).

## Changelog for 2020

---

### ClickHouse release 20.12

#### ClickHouse release v20.12.5.14-stable, 2020-12-28

##### Bug Fix

- Disable write with AIO during merges because it can lead to extremely rare data corruption of primary key columns during merge. [#18481](#) ([alesapin](#)).
- Fixed value is too short error when executing `toType(...)` functions (`toDate`, `toUInt32`, etc) with argument of type `Nullable(String)`. Now such functions return `NULL` on parsing errors instead of throwing exception. Fixes [#7673](#). [#18445](#) ([tavplubix](#)).
- Restrict merges from wide to compact parts. In case of vertical merge it led to broken result part. [#18381](#) ([Anton Popov](#)).
- Fix filling table `system.settings_profile_elements`. This PR fixes [#18231](#). [#18379](#) ([Vitaly Baranov](#)).
- Fix possible crashes in aggregate functions with combinator `Distinct`, while using two-level aggregation. Fixes [#17682](#). [#18365](#) ([Anton Popov](#)).
- Fix error when query `MODIFY COLUMN ... REMOVE TTL` does not actually remove column TTL. [#18130](#) ([alesapin](#)).

##### Build/Testing/Packaging Improvement

- Update timezones info to 2020e. [#18531](#) ([alesapin](#)).

#### ClickHouse release v20.12.4.5-stable, 2020-12-24

##### Bug Fix

- Fixed issue when `clickhouse-odbc-bridge` process is unreachable by server on machines with dual IPv4/IPv6 stack; - Fixed issue when ODBC dictionary updates are performed using malformed queries and/or cause crashes; Possibly closes [#14489](#). [#18278](#) ([Denis Glazachev](#)).
- Fixed key comparison between `Enum` and `Int` types. This fixes [#17989](#). [#18214](#) ([Amos Bird](#)).
- Fixed unique key convert crash in `MaterializeMySQL` database engine. This fixes [#18186](#) and fixes [#16372](#). [#18211](#) ([Winter Zhang](#)).
- Fixed `std::out_of_range`: `basic_string` in S3 URL parsing. [#18059](#) ([Vladimir Chebotarev](#)).
- Fixed the issue when some tables not synchronized to ClickHouse from MySQL caused by the fact that conversion MySQL prefix index wasn't supported for `MaterializeMySQL`. This fixes [#15187](#) and fixes [#17912](#) [#17944](#) ([Winter Zhang](#)).
- Fixed the issue when query optimization was producing wrong result if query contains `ARRAY JOIN`. [#17887](#) ([sundylj](#)).
- Fixed possible segfault in `topK` aggregate function. This closes [#17404](#). [#17845](#) ([Maksim Kita](#)).
- Fixed empty `system.stack_trace` table when server is running in daemon mode. [#17630](#) ([Amos Bird](#)).

# ClickHouse release v20.12.3.3-stable, 2020-12-13

## Backward Incompatible Change

- Enable `use_compact_format_in_distributed_parts_names` by default (see the documentation for the reference). [#16728 \(Azat Khuzhin\)](#).
- Accept user settings related to file formats (e.g. `format_csv_delimiter`) in the `SETTINGS` clause when creating a table that uses `File` engine, and use these settings in all `INSERTs` and `SELECTs`. The file format settings changed in the current user session, or in the `SETTINGS` clause of a DML query itself, no longer affect the query. [#16591 \(Alexander Kuzmenkov\)](#).

## New Feature

- add `*.xz` compression/decompression support. It enables using `*.xz` in `file()` function. This closes [#8828](#). [#16578 \(Abi Palagashvili\)](#).
- Introduce the query `ALTER TABLE ... DROP|DETACH PART 'part_name'`. [#15511 \(nvartolomei\)](#).
- Added new `ALTER UPDATE/DELETE IN PARTITION` syntax. [#13403 \(Vladimir Chebotarev\)](#).
- Allow formatting named tuples as JSON objects when using JSON input/output formats, controlled by the `output_format_json_named_tuples_as_objects` setting, disabled by default. [#17175 \(Alexander Kuzmenkov\)](#).
- Add a possibility to input enum value as its id in TSV and CSV formats by default. [#16834 \(Kruglov Pavel\)](#).
- Add COLLATE support for Nullable, LowCardinality, Array and Tuple, where nested type is String. Also refactor the code associated with collations in `ColumnString.cpp`. [#16273 \(Kruglov Pavel\)](#).
- New `tcpPort` function returns TCP port listened by this server. [#17134 \(Ivan\)](#).
- Add new math functions: `acosh`, `asinh`, `atan2`, `atanh`, `cosh`, `hypot`, `log1p`, `sinh`. [#16636 \(Konstantin Malanchev\)](#).
- Possibility to distribute the merges between different replicas. Introduces the `execute_merges_on_single_replica_time_threshold` mergetree setting. [#16424 \(filimonov\)](#).
- Add setting `aggregate_functions_null_for_empty` for SQL standard compatibility. This option will rewrite all aggregate functions in a query, adding `-OrNull` suffix to them. Implements [10273](#). [#16123 \(flynn\)](#).
- Updated `DateTime`, `DateTime64` parsing to accept string Date literal format. [#16040 \(Maksim Kita\)](#).
- Make it possible to change the path to history file in `clickhouse-client` using the `--history_file` parameter. [#15960 \(Maksim Kita\)](#).

## Bug Fix

- Fix the issue when server can stop accepting connections in very rare cases. [#17542 \(Amos Bird\)](#).
- Fixed `Function not implemented` error when executing `RENAME` query in `Atomic` database with ClickHouse running on Windows Subsystem for Linux. Fixes [#17661](#). [#17664 \(tavplubix\)](#).
- Do not restore parts from WAL if `in_memory_parts_enable_wal` is disabled. [#17802 \(detailyang\)](#).
- fix incorrect initialization of `max_compress_block_size` of `MergeTreeWriterSettings` with `min_compress_block_size`. [#17833 \(flynn\)](#).
- Exception message about max table size to drop was displayed incorrectly. [#17764 \(alexey-milovidov\)](#).
- Fixed possible segfault when there is not enough space when inserting into Distributed table. [#17737 \(tavplubix\)](#).

- Fixed problem when ClickHouse fails to resume connection to MySQL servers. #17681 (Alexander Kazakov).
- In might be determined incorrectly if cluster is circular- (cross-) replicated or not when executing `ON CLUSTER` query due to race condition when `pool_size > 1`. It's fixed. #17640 (tavplubix).
- Exception `fmt::v7::format_error` can be logged in background for MergeTree tables. This fixes #17613. #17615 (alexey-milovidov).
- When clickhouse-client is used in interactive mode with multiline queries, single line comment was erroneously extended till the end of query. This fixes #13654. #17565 (alexey-milovidov).
- Fix alter query hang when the corresponding mutation was killed on the different replica. Fixes #16953. #17499 (alesapin).
- Fix issue when mark cache size was underestimated by clickhouse. It may happen when there are a lot of tiny files with marks. #17496 (alesapin).
- Fix `ORDER BY` with enabled setting `optimize_redundant_functions_in_order_by`. #17471 (Anton Popov).
- Fix duplicates after `DISTINCT` which were possible because of incorrect optimization. Fixes #17294. #17296 (li chengxiang). #17439 (Nikolai Kochetov).
- Fix crash while reading from `JOIN` table with `LowCardinality` types. Fixes #17228. #17397 (Nikolai Kochetov).
- fix `toInt256(inf)` stack overflow. Int256 is an experimental feature. Closed #17235. #17257 (flynn).
- Fix possible `Unexpected packet Data received from clienterror` logged for Distributed queries with `LIMIT`. #17254 (Azat Khuzhin).
- Fix set index invalidation when there are const columns in the subquery. This fixes #17246. #17249 (Amos Bird).
- Fix possible wrong index analysis when the types of the index comparison are different. This fixes #17122. #17145 (Amos Bird).
- Fix ColumnConst comparison which leads to crash. This fixed #17088 . #17135 (Amos Bird).
- Multiple fixed for MaterializeMySQL (experimental feature). Fixes #16923 Fixes #15883 Fix MaterializeMySQL SYNC failure when the modify MySQL binlog\_checksum. #17091 (Winter Zhang).
- Fix bug when `ON CLUSTER` queries may hang forever for non-leader ReplicatedMergeTreeTables. #17089 (alesapin).
- Fixed crash on `CREATE TABLE ... AS some_table` query when `some_table` was created `AS table_function()` Fixes #16944. #17072 (tavplubix).
- Bug unfinished implementation for funciton fuzzBits, related issue: #16980. #17051 (hexiaoting).
- Fix LLVM's libunwind in the case when CFA register is RAX. This is the `bug` in LLVM's libunwind. We already have workarounds for this bug. #17046 (alexey-milovidov).
- Avoid unnecessary network errors for remote queries which may be cancelled while execution, like queries with `LIMIT`. #17006 (Azat Khuzhin).
- Fix `optimize_distributed_group_by_sharding_key` setting (that is disabled by default) for query with `OFFSET` only. #16996 (Azat Khuzhin).
- Fix for Merge tables over Distributed tables with `JOIN`. #16993 (Azat Khuzhin).

- Fixed wrong result in big integers (128, 256 bit) when casting from double. Big integers support is experimental. [#16986 \(Mike\)](#).
- Fix possible server crash after `ALTER TABLE ... MODIFY COLUMN ... NewType` when `SELECT` have `WHERE` expression on altering column and alter does not finished yet. [#16968 \(Amos Bird\)](#).
- Blame info was not calculated correctly in `clickhouse-git-import`. [#16959 \(alexey-milovidov\)](#).
- Fix order by optimization with monotonous functions. Fixes [#16107](#). [#16956 \(Anton Popov\)](#).
- Fix optimization of group by with enabled setting `optimize_aggregators_of_group_by_keys` and joins. Fixes [#12604](#). [#16951 \(Anton Popov\)](#).
- Fix possible error `Illegal type of argument` for queries with `ORDER BY`. Fixes [#16580](#). [#16928 \(Nikolai Kochetov\)](#).
- Fix strange code in `InterpreterShowAccessQuery`. [#16866 \(tavplubix\)](#).
- Prevent clickhouse server crashes when using the function `timeSeriesGroupSum`. The function is removed from newer ClickHouse releases. [#16865 \(filimonov\)](#).
- Fix rare silent crashes when query profiler is on and ClickHouse is installed on OS with glibc version that has (supposedly) broken asynchronous unwind tables for some functions. This fixes [#15301](#). This fixes [#13098](#). [#16846 \(alexey-milovidov\)](#).
- Fix crash when using `any` without any arguments. This is for [#16803](#). cc @azat. [#16826 \(Amos Bird\)](#).
- If no memory can be allocated while writing table metadata on disk, broken metadata file can be written. [#16772 \(alexey-milovidov\)](#).
- Fix trivial query optimization with partition predicate. [#16767 \(Azat Khuzhin\)](#).
- Fix `IN` operator over several columns and tuples with enabled `transform_null_in` setting. Fixes [#15310](#). [#16722 \(Anton Popov\)](#).
- Return number of affected rows for `INSERT` queries via MySQL protocol. Previously ClickHouse used to always return 0, it's fixed. Fixes [#16605](#). [#16715 \(Winter Zhang\)](#).
- Fix remote query failure when using 'if' suffix aggregate function. Fixes [#16574](#) Fixes [#16231](#) [#16610 \(Winter Zhang\)](#).
- Fix inconsistent behavior caused by `select_sequential_consistency` for optimized trivial count query and `system.tables`. [#16309 \(Hao Chen\)](#).

## Improvement

- Remove empty parts after they were pruned by TTL, mutation, or collapsing merge algorithm. [#16895 \(Anton Popov\)](#).
- Enable compact format of directories for asynchronous sends in Distributed tables: `use_compact_format_in_distributed_parts_names` is set to 1 by default. [#16788 \(Azat Khuzhin\)](#).
- Abort multipart upload if no data was written to S3. [#16840 \(Pavel Kovalenko\)](#).
- Reresolve the IP of the `format_avro_schema_registry_url` in case of errors. [#16985 \(filimonov\)](#).
- Mask password in `data_path` in the `system.distribution_queue`. [#16727 \(Azat Khuzhin\)](#).
- Throw error when use column transformer replaces non existing column. [#16183 \(hexiaoting\)](#).

- Turn off parallel parsing when there is no enough memory for all threads to work simultaneously. Also there could be exceptions like "Memory limit exceeded" when somebody will try to insert extremely huge rows (> min\_chunk\_bytes\_for\_parallel\_parsing), because each piece to parse has to be independent set of strings (one or more). [#16721](#) ([Nikita Mikhaylov](#)).
- Install script should always create subdirs in config folders. This is only relevant for Docker build with custom config. [#16936](#) ([filimonov](#)).
- Correct grammar in error message in JSONEachRow, JSONCompactEachRow, and RegexpRow input formats. [#17205](#) ([nico piderman](#)).
- Set default host and port parameters for SOURCE(CLICKHOUSE(...)) to current instance and set default user value to 'default'. [#16997](#) ([vdimir](#)).
- Throw an informative error message when doing ATTACH/DETACH TABLE <DICTIONARY>. Before this PR, detach table <dict> works but leads to an ill-formed in-memory metadata. [#16885](#) ([Amos Bird](#)).
- Add cutToFirstSignificantSubdomainWithWWW(). [#16845](#) ([Azat Khuzhin](#)).
- Server refused to startup with exception message if wrong config is given (metric\_log.collect\_interval\_milliseconds is missing). [#16815](#) ([Ivan](#)).
- Better exception message when configuration for distributed DDL is absent. This fixes [#5075](#). [#16769](#) ([Nikita Mikhaylov](#)).
- Usability improvement: better suggestions in syntax error message when CODEC expression is misplaced in CREATE TABLE query. This fixes [#12493](#). [#16768](#) ([alexey-milovidov](#)).
- Remove empty directories for async INSERT at start of Distributed engine. [#16729](#) ([Azat Khuzhin](#)).
- Workaround for use S3 with nginx server as proxy. Nginx currently does not accept urls with empty path like http://domain.com?delete, but vanilla aws-sdk-cpp produces this kind of urls. This commit uses patched aws-sdk-cpp version, which makes urls with "/" as path in this cases, like http://domain.com/?delete. [#16709](#) ([ianton-ru](#)).
- Allow reinterpretAs\* functions to work for integers and floats of the same size. Implements [16640](#). [#16657](#) ([flynn](#)).
- Now, <auxiliary\_zookeepers> configuration can be changed in config.xml and reloaded without server startup. [#16627](#) ([Amos Bird](#)).
- Support SNI in https connections to remote resources. This will allow to connect to Cloudflare servers that require SNI. This fixes [#10055](#). [#16252](#) ([alexey-milovidov](#)).
- Make it possible to connect to clickhouse-server secure endpoint which requires SNI. This is possible when clickhouse-server is hosted behind TLS proxy. [#16938](#) ([filimonov](#)).
- Fix possible stack overflow if a loop of materialized views is created. This closes [#15732](#). [#16048](#) ([alexey-milovidov](#)).
- Simplify the implementation of background tasks processing for the MergeTree table engines family. There should be no visible changes for user. [#15983](#) ([alesapin](#)).
- Improvement for MaterializeMySQL (experimental feature). Throw exception about right sync privileges when MySQL sync user has error privileges. [#15977](#) ([TCeason](#)).
- Made indexOf() use BloomFilter. [#14977](#) ([achimbab](#)).

## Performance Improvement

- Use Floyd-Rivest algorithm, it is the best for the ClickHouse use case of partial sorting. Benchmarks are in <https://github.com/danlark1/miniselect> and here. #16825 (Danila Kutenin).
- Now ReplicatedMergeTree tree engines family uses a separate thread pool for replicated fetches. Size of the pool limited by setting `background_fetches_pool_size` which can be tuned with a server restart. The default value of the setting is 3 and it means that the maximum amount of parallel fetches is equal to 3 (and it allows to utilize 10G network). Fixes #520. #16390 (alesapin).
- Fixed uncontrolled growth of the state of quantileTDigest. #16680 (hrissan).
- Add `VIEW` subquery description to `EXPLAIN`. Limit push down optimisation for `VIEW`. Add local replicas of `Distributed` to query plan. #14936 (Nikolai Kochetov).
- Fix `optimize_read_in_order`/`optimize_aggregation_in_order` with `max_threads > 0` and expression in `ORDER BY`. #16637 (Azat Khuzhin).
- Fix performance of reading from `Merge` tables over huge number of `MergeTree` tables. Fixes #7748. #16988 (Anton Popov).
- Now we can safely prune partitions with exact match. Useful case: Suppose table is partitioned by `intHash64(x) % 100` and the query has condition on `intHash64(x) % 100` verbatim, not on `x`. #16253 (Amos Bird).

## Experimental Feature

- Add `EmbeddedRocksDB` table engine (can be used for dictionaries). #15073 (sundyli).

## Build/Testing/Packaging Improvement

- Improvements in test coverage building images. #17233 (alesapin).
- Update embedded timezone data to version 2020d (also update cctz to the latest master). #17204 (filimonov).
- Fix UBSan report in Poco. This closes #12719. #16765 (alexey-milovidov).
- Do not instrument 3rd-party libraries with UBSan. #16764 (alexey-milovidov).
- Fix UBSan report in cache dictionaries. This closes #12641. #16763 (alexey-milovidov).
- Fix UBSan report when trying to convert infinite floating point number to integer. This closes #14190. #16677 (alexey-milovidov).

# ClickHouse release 20.11

## ClickHouse release v20.11.7.16-stable, 2021-03-02

### Improvement

- Explicitly set uid / gid of clickhouse user & group to the fixed values (101) in clickhouse-server images. #19096 (filimonov).

### Bug Fix

- BloomFilter index crash fix. Fixes #19757. #19884 (Maksim Kita).
- Deadlock was possible if `system.text_log` is enabled. This fixes #19874. #19875 (alexey-milovidov).
- In previous versions, unusual arguments for function `arrayEnumerateUniq` may cause crash or infinite loop. This closes #19787. #19788 (alexey-milovidov).

- Fixed stack overflow when using accurate comparison of arithmetic type with string type. #19773 ([tavplubix](#)).
- Fix a segmentation fault in `bitmapAndnot` function. Fixes #19668. #19713 ([Maksim Kita](#)).
- Some functions with big integers may cause segfault. Big integers is experimental feature. This closes #19667. #19672 ([alexey-milovidov](#)).
- Fix wrong result of function `neighbor` for `LowCardinality` argument. Fixes #10333. #19617 ([Nikolai Kochetov](#)).
- Fix use-after-free of the `CompressedWriteBuffer` in `Connection` after disconnect. #19599 ([Azat Khuzhin](#)).
- `DROP/DETACH TABLE` table ON CLUSTER cluster `SYNC` query might hang, it's fixed. Fixes #19568. #19572 ([tavplubix](#)).
- Query `CREATE DICTIONARY` id expression fix. #19571 ([Maksim Kita](#)).
- Fix SIGSEGV with  
`merge_tree_min_rows_for_concurrent_read/merge_tree_min_bytes_for_concurrent_read=0/UINT64_MAX.`  
#19528 ([Azat Khuzhin](#)).
- Buffer overflow (on memory read) was possible if `addMonth` function was called with specifically crafted arguments. This fixes #19441. This fixes #19413. #19472 ([alexey-milovidov](#)).
- Mark distributed batch as broken in case of empty data block in one of files. #19449 ([Azat Khuzhin](#)).
- Fix possible buffer overflow in Uber H3 library. See <https://github.com/uber/h3/issues/392>. This closes #19219. #19383 ([alexey-milovidov](#)).
- Fix system.parts \_state column (LOGICAL\_ERROR when querying this column, due to incorrect order). #19346 ([Azat Khuzhin](#)).
- Fix error `Cannot convert column now64()` because it is constant but values of constants are different in source and result. Continuation of #7156. #19316 ([Nikolai Kochetov](#)).
- Fix bug when concurrent `ALTER` and `DROP` queries may hang while processing `ReplicatedMergeTree` table. #19237 ([alesapin](#)).
- Fix infinite reading from file in `ORC` format (was introduced in #10580). Fixes #19095. #19134 ([Nikolai Kochetov](#)).
- Fix startup bug when clickhouse was not able to read compression codec from `LowCardinality(Nullable(...))` and throws exception `Attempt to read after EOF`. Fixes #18340. #19101 ([alesapin](#)).
- Fixed `There is no checkpoint` error when inserting data through http interface using `Template` or `CustomSeparated` format. Fixes #19021. #19072 ([tavplubix](#)).
- Restrict `MODIFY TTL` queries for `MergeTree` tables created in old syntax. Previously the query succeeded, but actually it had no effect. #19064 ([Anton Popov](#)).
- Make sure `groupUniqArray` returns correct type for argument of `Enum` type. This closes #17875. #19019 ([alexey-milovidov](#)).
- Fix possible error `Expected single dictionary argument for function ignore` if use function `ignore` with `LowCardinality` argument. Fixes #14275. #19016 ([Nikolai Kochetov](#)).
- Fix inserting of `LowCardinality` column to table with `TinyLog` engine. Fixes #18629. #19010 ([Nikolai Kochetov](#)).

- Disable `optimize_move_functions_out_of_any` because optimization is not always correct. This closes [#18051](#). This closes [#18973](#). [#18981](#) ([alexey-milovidov](#)).
- Fixed very rare deadlock at shutdown. [#18977](#) ([tavplubix](#)).
- Fix bug when mutation with some escaped text (like `ALTER ... UPDATE e = CAST('foo', 'Enum8(\\"foo\\" = 1')`) serialized incorrectly. Fixes [#18878](#). [#18944](#) ([alesapin](#)).
- Attach partition should reset the mutation. [#18804](#). [#18935](#) ([fastio](#)).
- Fix possible hang at shutdown in `clickhouse-local`. This fixes [#18891](#). [#18893](#) ([alexey-milovidov](#)).
- Fix \*If combinator with unary function and Nullable types. [#18806](#) ([Azat Khuzhin](#)).
- Asynchronous distributed INSERTs can be rejected by the server if the setting `network_compression_method` is globally set to non-default value. This fixes [#18741](#). [#18776](#) ([alexey-milovidov](#)).
- Fixed Attempt to read after eof error when trying to `CAST NULL` from `Nullable(String)` to `Nullable(Decimal(P, S))`. Now function `CAST` returns `NULL` when it cannot parse decimal from nullable string. Fixes [#7690](#). [#18718](#) ([Winter Zhang](#)).
- Fix Logger with unmatched arg size. [#18717](#) ([sundyli](#)).
- Add `FixedString` Data type support. I'll get this exception "Code: 50, e.displayText() = DB::Exception: Unsupported type `FixedString(1)`" when replicating data from MySQL to ClickHouse. This patch fixes bug [#18450](#) Also fixes [#6556](#). [#18553](#) ([awesomeléo](#)).
- Fix possible `Pipeline` stuck error while using `ORDER BY` after subquery with `RIGHT` or `FULL` join. [#18550](#) ([Nikolai Kochetov](#)).
- Fix bug which may lead to `ALTER` queries hung after corresponding mutation kill. Found by thread fuzzer. [#18518](#) ([alesapin](#)).
- Disable write with AIO during merges because it can lead to extremely rare data corruption of primary key columns during merge. [#18481](#) ([alesapin](#)).
- Disable constant folding for subqueries on the analysis stage, when the result cannot be calculated. [#18446](#) ([Azat Khuzhin](#)).
- Fixed value is too short error when executing `toType(...)` functions (`toDate`, `toUInt32`, etc) with argument of type `Nullable(String)`. Now such functions return `NULL` on parsing errors instead of throwing exception. Fixes [#7673](#). [#18445](#) ([tavplubix](#)).
- Restrict merges from wide to compact parts. In case of vertical merge it led to broken result part. [#18381](#) ([Anton Popov](#)).
- Fix filling table `system.settings_profile_elements`. This PR fixes [#18231](#). [#18379](#) ([Vitaly Baranov](#)).
- Fix index analysis of binary functions with constant argument which leads to wrong query results. This fixes [#18364](#). [#18373](#) ([Amos Bird](#)).
- Fix possible crashes in aggregate functions with combinator `Distinct`, while using two-level aggregation. Fixes [#17682](#). [#18365](#) ([Anton Popov](#)).
- `SELECT count()` FROM table now can be executed if only one any column can be selected from the table. This PR fixes [#10639](#). [#18233](#) ([Vitaly Baranov](#)).
- `SELECT JOIN` now requires the `SELECT` privilege on each of the joined tables. This PR fixes [#17654](#). [#18232](#) ([Vitaly Baranov](#)).

- Fix possible incomplete query result while reading from `MergeTree*` in case of read backoff (message `<Debug> MergeTreeReadPool: Will lower number of threads in logs`). Was introduced in #16423. Fixes #18137. #18216 (Nikolai Kochetov).
- Fix error when query `MODIFY COLUMN ... REMOVE TTL` does not actually remove column TTL. #18130 (alesapin).
- Fix indeterministic functions with predicate optimizer. This fixes #17244. #17273 (Winter Zhang).
- Mutation might hang waiting for some non-existent part after `MOVE` or `REPLACE PARTITION` or, in rare cases, after `DETACH` or `DROP PARTITION`. It's fixed. #15537 (tavplubix).

## Build/Testing/Packaging Improvement

- Update timezones info to 2020e. #18531 (alesapin).

## ClickHouse release v20.11.6.6-stable, 2020-12-24

### Bug Fix

- Fixed issue when `clickhouse-odbc-bridge` process is unreachable by server on machines with dual IPv4/IPv6 stack and fixed issue when ODBC dictionary updates are performed using malformed queries and/or cause crashes. This possibly closes #14489. #18278 (Denis Glazachev).
- Fixed key comparison between `Enum` and `Int` types. This fixes #17989. #18214 (Amos Bird).
- Fixed unique key convert crash in `MaterializeMySQL` database engine. This fixes #18186 and fixes #16372 #18211 (Winter Zhang).
- Fixed `std::out_of_range: basic_string` in S3 URL parsing. #18059 (Vladimir Chebotarev).
- Fixed the issue when some tables not synchronized to ClickHouse from MySQL caused by the fact that conversion MySQL prefix index wasn't supported for `MaterializeMySQL`. This fixes #15187 and fixes #17912 #17944 (Winter Zhang).
- Fixed the issue when query optimization was producing wrong result if query contains `ARRAY JOIN`. #17887 (sundyli).
- Fix possible segfault in `topK` aggregate function. This closes #17404. #17845 (Maksim Kita).
- Do not restore parts from WAL if `in_memory_parts_enable_wal` is disabled. #17802 (detailyang).
- Fixed problem when ClickHouse fails to resume connection to MySQL servers. #17681 (Alexander Kazakov).
- Fixed inconsistent behaviour of `optimize_trivial_count_query` with partition predicate. #17644 (Azat Khuzhin).
- Fixed empty `system.stack_trace` table when server is running in daemon mode. #17630 (Amos Bird).
- Fixed the behaviour when `xxception fmt::v7::format_error` can be logged in background for `MergeTree` tables. This fixes #17613. #17615 (alexey-milovidov).
- Fixed the behaviour when `clickhouse-client` is used in interactive mode with multiline queries and single line comment was erroneously extended till the end of query. This fixes #13654. #17565 (alexey-milovidov).
- Fixed the issue when server can stop accepting connections in very rare cases. #17542 (alexey-milovidov).
- Fixed alter query hang when the corresponding mutation was killed on the different replica. This fixes #16953. #17499 (alesapin).

- Fixed bug when mark cache size was underestimated by clickhouse. It may happen when there are a lot of tiny files with marks. [#17496](#) ([alesapin](#)).
- Fixed ORDER BY with enabled setting `optimize_redundant_functions_in_order_by`. [#17471](#) ([Anton Popov](#)).
- Fixed duplicates after DISTINCT which were possible because of incorrect optimization. This fixes [#17294](#). [#17296](#) ([li chengxiang](#)). [#17439](#) ([Nikolai Kochetov](#)).
- Fixed crash while reading from JOIN table with LowCardinality types. This fixes [#17228](#). [#17397](#) ([Nikolai Kochetov](#)).
- Fixed set index invalidation when there are const columns in the subquery. This fixes [#17246](#) . [#17249](#) ([Amos Bird](#)).
- Fixed possible wrong index analysis when the types of the index comparison are different. This fixes [#17122](#). [#17145](#) ([Amos Bird](#)).
- Fixed ColumnConst comparison which leads to crash. This fixes [#17088](#) . [#17135](#) ([Amos Bird](#)).
- Fixed bug when ON CLUSTER queries may hang forever for non-leader ReplicatedMergeTreeTables. [#17089](#) ([alesapin](#)).
- Fixed fuzzer-found bug in funciton fuzzBits. This fixes [#16980](#). [#17051](#) ([hexiaoting](#)).
- Avoid unnecessary network errors for remote queries which may be cancelled while execution, like queries with LIMIT. [#17006](#) ([Azat Khuzhin](#)).
- Fixed wrong result in big integers (128, 256 bit) when casting from double. [#16986](#) ([Mike](#)).
- Reresolve the IP of the `format_avro_schema_registry_url` in case of errors. [#16985](#) ([filimonov](#)).
- Fixed possible server crash after ALTER TABLE ... MODIFY COLUMN ... NewTypewhen SELECT have WHERE expression on altering column and alter does not finished yet. [#16968](#) ([Amos Bird](#)).
- Blame info was not calculated correctly in clickhouse-git-import. [#16959](#) ([alexey-milovidov](#)).
- Fixed order by optimization with monotonous functions. Fixes [#16107](#). [#16956](#) ([Anton Popov](#)).
- Fixed optimization of group by with enabled setting `optimize_aggregators_of_group_by_keys` and joins. This fixes [#12604](#). [#16951](#) ([Anton Popov](#)).
- Install script should always create subdirs in config folders. This is only relevant for Docker build with custom config. [#16936](#) ([filimonov](#)).
- Fixed possible error Illegal type of argument for queries with ORDER BY. This fixes [#16580](#). [#16928](#) ([Nikolai Kochetov](#)).
- Abort multipart upload if no data was written to WriteBufferFromS3. [#16840](#) ([Pavel Kovalenko](#)).
- Fixed crash when using `any` without any arguments. This fixes [#16803](#). [#16826](#) ([Amos Bird](#)).
- Fixed the behaviour when ClickHouse used to always return 0 insted of a number of affected rows for INSERT queries via MySQL protocol. This fixes [#16605](#). [#16715](#) ([Winter Zhang](#)).
- Fixed uncontrolled growth of TDigest. [#16680](#) ([hrissan](#)).
- Fixed remote query failure when using suffix `if` in Aggregate function. This fixes [#16574](#) fixes [#16231](#) [#16610](#) ([Winter Zhang](#)).
- Fixed inconsistent behavior caused by `select_sequential_consistency` for optimized trivial count query and system.tables. [#16309](#) ([Hao Chen](#)).

- Throw error when use ColumnTransformer replace non exist column. #16183 (hexiaoting).

## ClickHouse release v20.11.3.3-stable, 2020-11-13

### Bug Fix

- Fix rare silent crashes when query profiler is on and ClickHouse is installed on OS with glibc version that has (supposedly) broken asynchronous unwind tables for some functions. This fixes #15301. This fixes #13098. #16846 (alexey-milovidov).

## ClickHouse release v20.11.2.1, 2020-11-11

### Backward Incompatible Change

- If some `profile` was specified in `distributed_ddl` config section, then this profile could overwrite settings of `default` profile on server startup. It's fixed, now settings of distributed DDL queries should not affect global server settings. #16635 (tavplubix).
- Restrict to use of non-comparable data types (like `AggregateFunction`) in keys (Sorting key, Primary key, Partition key, and so on). #16601 (alesapin).
- Remove `ANALYZE` and `AST` queries, and make the setting `enable_debug_queries` obsolete since now it is the part of full featured `EXPLAIN` query. #16536 (Ivan).
- Aggregate functions `boundingRatio`, `rankCorr`, `retention`, `timeSeriesGroupSum`, `timeSeriesGroupRateSum`, `windowFunnel` were erroneously made case-insensitive. Now their names are made case sensitive as designed. Only functions that are specified in SQL standard or made for compatibility with other DBMS or functions similar to those should be case-insensitive. #16407 (alexey-milovidov).
- Make `rankCorr` function return nan on insufficient data #16124. #16135 (hexiaoting).
- When upgrading from versions older than 20.5, if rolling update is performed and cluster contains both versions 20.5 or greater and less than 20.5, if ClickHouse nodes with old versions are restarted and old version has been started up in presence of newer versions, it may lead to `Part ... intersects previous part` errors. To prevent this error, first install newer clickhouse-server packages on all cluster nodes and then do restarts (so, when clickhouse-server is restarted, it will start up with the new version).

### New Feature

- Added support of LDAP as a user directory for locally non-existent users. #12736 (Denis Glazachev).
- Add `system.replicated_fetches` table which shows currently running background fetches. #16428 (alesapin).
- Added setting `date_time_output_format`. #15845 (Maksim Kita).
- Added minimal web UI to ClickHouse. #16158 (alexey-milovidov).
- Allows to read/write Single protobuf message at once (w/o length-delimiters). #15199 (filimonov).
- Added initial OpenTelemetry support. ClickHouse now accepts OpenTelemetry traceparent headers over Native and HTTP protocols, and passes them downstream in some cases. The trace spans for executed queries are saved into the `system.opentelemetry_span_log` table. #14195 (Alexander Kuzmenkov).
- Allow specify primary key in column list of `CREATE TABLE` query. This is needed for compatibility with other SQL dialects. #15823 (Maksim Kita).
- Implement `OFFSET offset_row_count {ROW | ROWS} FETCH {FIRST | NEXT} fetch_row_count {ROW | ROWS} {ONLY | WITH TIES}` in `SELECT` query with `ORDER BY`. This is the SQL-standard way to specify `LIMIT`. #15855 (hexiaoting).

- `errorCodeToName` function - return variable name of the error (useful for analyzing `query_log` and similar). `system.errors` table - shows how many times errors has been happened (respects `system_events_show_zero_values`). [#16438 \(Azat Khuzhin\)](#).
- Added function `untuple` which is a special function which can introduce new columns to the `SELECT` list by expanding a named tuple. [#16242 \(Nikolai Kochetov, Amos Bird\)](#).
- Now we can provide identifiers via query parameters. And these parameters can be used as table objects or columns. [#16594 \(Amos Bird\)](#).
- Added big integers (`UInt256`, `Int128`, `Int256`) and UUID data types support for MergeTree BloomFilter index. Big integers is an experimental feature. [#16642 \(Maksim Kita\)](#).
- Add `farmFingerprint64` function (non-cryptographic string hashing). [#16570 \(Jacob Hayes\)](#).
- Add `log_queries_min_query_duration_ms`, only queries slower than the value of this setting will go to `query_log/query_thread_log` (i.e. something like `slow_query_log` in mysql). [#16529 \(Azat Khuzhin\)](#).
- Ability to create a docker image on the top of `Alpine`. Uses precompiled binary and glibc components from ubuntu 20.04. [#16479 \(filimonov\)](#).
- Added `toUUIDOrNull`, `toUUIDOrZero` cast functions. [#16337 \(Maksim Kita\)](#).
- Add `max_concurrent_queries_for_all_users` setting, see [#6636](#) for use cases. [#16154 \(nvartolomei\)](#).
- Add a new option `print_query_id` to `clickhouse-client`. It helps generate arbitrary strings with the current query id generated by the client. Also print query id in `clickhouse-client` by default. [#15809 \(Amos Bird\)](#).
- Add `tid` and `logTrace` functions. This closes [#9434](#). [#15803 \(flynn\)](#).
- Add function `formatReadableTimeDelta` that format time delta to human readable string ... [#15497 \(Filipe Caixeta\)](#).
- Added `disable_merges` option for volumes in multi-disk configuration. [#13956 \(Vladimir Chebotarev\)](#).

## Experimental Feature

- New functions `encrypt`, `aes_encrypt_mysql`, `decrypt`, `aes_decrypt_mysql`. These functions are working slowly, so we consider it as an experimental feature. [#11844 \(Vasily Nemkov\)](#).

## Bug Fix

- Mask password in `data_path` in the `system.distribution_queue`. [#16727 \(Azat Khuzhin\)](#).
- Fix `IN` operator over several columns and tuples with enabled `transform_null_in` setting. Fixes [#15310](#). [#16722 \(Anton Popov\)](#).
- The setting `max_parallel_replicas` worked incorrectly if the queried table has no sampling. This fixes [#5733](#). [#16675 \(alexey-milovidov\)](#).
- Fix `optimize_read_in_order/optimize_aggregation_in_order` with `max_threads > 0` and expression in `ORDER BY`. [#16637 \(Azat Khuzhin\)](#).
- Calculation of `DEFAULT` expressions was involving possible name collisions (that was very unlikely to encounter). This fixes [#9359](#). [#16612 \(alexey-milovidov\)](#).
- Fix `query_thread_log.query_duration_ms` unit. [#16563 \(Azat Khuzhin\)](#).
- Fix a bug when using MySQL Master -> MySQL Slave -> ClickHouse MaterializeMySQL Engine. `MaterializeMySQL` is an experimental feature. [#16504 \(TCeason\)](#).

- Specifically crafted argument of `round` function with `Decimal` was leading to integer division by zero. This fixes #13338. #16451 (alexey-milovidov).
- Fix `DROP TABLE` for Distributed (racy with `INSERT`). #16409 (Azat Khuzhin).
- Fix processing of very large entries in replication queue. Very large entries may appear in `ALTER` queries if table structure is extremely large (near 1 MB). This fixes #16307. #16332 (alexey-milovidov).
- Fixed the inconsistent behaviour when a part of return data could be dropped because the set for its filtration wasn't created. #16308 (Nikita Mikhaylov).
- Fix `dictGet` in `sharding_key` (and similar places, i.e. when the function context is stored permanently). #16205 (Azat Khuzhin).
- Fix the exception thrown in `clickhouse-local` when trying to execute `OPTIMIZE` command. Fixes #16076. #16192 (filimonov).
- Fixes #15780 regression, e.g. `indexOf([1, 2, 3], toLowCardinality(1))` now is prohibited but it should not be. #16038 (Mike).
- Fix bug with MySQL database. When MySQL server used as database engine is down some queries raise `Exception`, because they try to get tables from disabled server, while it's unnecessary. For example, query `SELECT ... FROM system.parts` should work only with MergeTree tables and don't touch MySQL database at all. #16032 (Kruglov Pavel).
- Now exception will be thrown when `ALTER MODIFY COLUMN ... DEFAULT ...` has incompatible default with column type. Fixes #15854. #15858 (alesapin).
- Fixed IPv4CIDRToRange/IPv6CIDRToRange functions to accept const IP-column values. #15856 (vladimir-golovchenko).

## Improvement

- Treat `INTERVAL '1 hour'` as equivalent to `INTERVAL 1 HOUR`, to be compatible with Postgres and similar. This fixes #15637. #15978 (flynn).
- Enable parsing enum values by their numeric ids for CSV, TSV and JSON input formats. #15685 (vivarum).
- Better read task scheduling for JBOD architecture and `MergeTree` storage. New setting `read_backoff_min_concurrency` which serves as the lower limit to the number of reading threads. #16423 (Amos Bird).
- Add missing support for `LowCardinality` in `Avro` format. #16521 (Mike).
- Workaround for use `S3` with `nginx` server as proxy. Nginx currently does not accept urls with empty path like `http://domain.com?delete`, but vanilla aws-sdk-cpp produces this kind of urls. This commit uses patched aws-sdk-cpp version, which makes urls with "/" as path in this cases, like `http://domain.com/?delete`. #16814 (ianton-ru).
- Better diagnostics on parse errors in input data. Provide row number on `Cannot read all data` errors. #16644 (alexey-milovidov).
- Make the behaviour of `minMap` and `maxMap` more desirable. It will not skip zero values in the result. Fixes #16087. #16631 (Ildus Kurbangaliev).
- Better update of ZooKeeper configuration in runtime. #16630 (sundyl).
- Apply `SETTINGS` clause as early as possible. It allows to modify more settings in the query. This closes #3178. #16619 (alexey-milovidov).

- Now `event_time_microseconds` field stores in Decimal64, not UInt64. #16617 (Nikita Mikhaylov).
- Now parameterized functions can be used in `APPLY` column transformer. #16589 (Amos Bird).
- Improve scheduling of background task which removes data of dropped tables in Atomic databases. Atomic databases do not create broken symlink to table data directory if table actually has no data directory. #16584 (tavplubix).
- Subqueries in `WITH` section (CTE) can reference previous subqueries in `WITH` section by their name. #16575 (Amos Bird).
- Add `current_database` into `system.query_thread_log`. #16558 (Azat Khuzhin).
- Allow to fetch parts that are already committed or outdated in the current instance into the detached directory. It's useful when migrating tables from another cluster and having N to 1 shards mapping. It's also consistent with the current `fetchPartition` implementation. #16538 (Amos Bird).
- Multiple improvements for RabbitMQ: Fixed bug for #16263. Also minimized event loop lifetime. Added more efficient queues setup. #16426 (Ksenia Sumarokova).
- Fix debug assertion in `quantileDeterministic` function. In previous version it may also transfer up to two times more data over the network. Although no bug existed. This fixes #15683. #16410 (alexey-milovidov).
- Add `TablesToDeleteQueueSize` metric. It's equal to number of dropped tables, that are waiting for background data removal. #16364 (tavplubix).
- Better diagnostics when client has dropped connection. In previous versions, `Attempt to read after EOF` and `Broken pipe` exceptions were logged in server. In new version, it's information message `Client has dropped the connection, cancel the query..` #16329 (alexey-milovidov).
- Add `total_rows/total_bytes` (from `system.tables`) support for Set/Join table engines. #16306 (Azat Khuzhin).
- Now it's possible to specify PRIMARY KEY without ORDER BY for MergeTree table engines family. Closes #15591. #16284 (alesapin).
- If there is no tmp folder in the system (chroot, misconfiguration etc) `clickhouse-local` will create temporary subfolder in the current directory. #16280 (filimonov).
- Add support for nested data types (like named tuple) as sub-types. Fixes #15587. #16262 (Ivan).
- Support for `database_atomic_wait_for_drop_and_detach_synchronously/NO DELAY/SYNC` for `DROP DATABASE`. #16127 (Azat Khuzhin).
- Add `allow_nondeterministic_optimize_skip_unused_shards` (to allow non deterministic like `rand()` or `dictGet()` in sharding key). #16105 (Azat Khuzhin).
- Fix `memory_profiler_step/max.untracked_memory` for queries via HTTP (test included). Fix the issue that adjusting this value globally in xml config does not help either, since those settings are not applied anyway, only default (4MB) value is used. Fix `query_id` for the most root ThreadStatus of the http query (by initializing `QueryScope` after reading `query_id`). #16101 (Azat Khuzhin).
- Now it's allowed to execute `ALTER ... ON CLUSTER` queries regardless of the `<internal_replication>` setting in cluster config. #16075 (alesapin).
- Fix rare issue when `clickhouse-client` may abort on exit due to loading of suggestions. This fixes #16035. #16047 (alexey-milovidov).
- Add support of cache layout for Redis dictionaries with complex key. #15985 (Anton Popov).

- Fix query hang (endless loop) in case of misconfiguration (`connections_with_failover_max_tries` set to 0). [#15876 \(Azat Khuzhin\)](#).
- Change level of some log messages from information to debug, so information messages will not appear for every query. This closes [#5293](#). [#15816 \(alexey-milovidov\)](#).
- Remove `MemoryTrackingInBackground*` metrics to avoid potentially misleading results. This fixes [#15684](#). [#15813 \(alexey-milovidov\)](#).
- Add reconnects to `zookeeper-dump-tree` tool. [#15711 \(alexey-milovidov\)](#).
- Allow explicitly specify columns list in `CREATE TABLE table AS table_function(...)` query. Fixes [#9249](#) Fixes [#14214](#). [#14295 \(tavplubix\)](#).

## Performance Improvement

- Do not merge parts across partitions in `SELECT FINAL`. [#15938 \(Kruglov Pavel\)](#).
- Improve performance of `-OrNull` and `-OrDefault` aggregate functions. [#16661 \(alexey-milovidov\)](#).
- Improve performance of `quantileMerge`. In previous versions it was obviously slow. This closes [#1463](#). [#16643 \(alexey-milovidov\)](#).
- Improve performance of logical functions a little. [#16347 \(alexey-milovidov\)](#).
- Improved performance of merges assignment in MergeTree table engines. Shouldn't be visible for the user. [#16191 \(alesapin\)](#).
- Speedup hashed/sparse\_hashed dictionary loading by preallocating the hash table. [#15454 \(Azat Khuzhin\)](#).
- Now trivial count optimization becomes slightly non-trivial. Predicates that contain exact partition expr can be optimized too. This also fixes [#11092](#) which returns wrong count when `max_parallel_replicas > 1`. [#15074 \(Amos Bird\)](#).

## Build/Testing/Packaging Improvement

- Add flaky check for stateless tests. It will detect potentially flaky functional tests in advance, before they are merged. [#16238 \(alesapin\)](#).
- Use proper version for `croaring` instead of amalgamation. [#16285 \(sundyli\)](#).
- Improve generation of build files for `ya.make` build system (Arcadia). [#16700 \(alexey-milovidov\)](#).
- Add MySQL BinLog file check tool for `MaterializeMySQL` database engine. `MaterializeMySQL` is an experimental feature. [#16223 \(Winter Zhang\)](#).
- Check for executable bit on non-executable files. People often accidentally commit executable files from Windows. [#15843 \(alexey-milovidov\)](#).
- Check for `#pragma once` in headers. [#15818 \(alexey-milovidov\)](#).
- Fix illegal code style `&vector[idx]` in `libhdfs3`. This fixes libcxx debug build. See also <https://github.com/ClickHouse-Extras/libhdfs3/pull/8> . [#15815 \(Amos Bird\)](#).
- Fix build of one miscellaneous example tool on Mac OS. Note that we don't build examples on Mac OS in our CI (we build only ClickHouse binary), so there is zero chance it will not break again. This fixes [#15804](#). [#15808 \(alexey-milovidov\)](#).
- Simplify Sys/V init script. [#14135 \(alexey-milovidov\)](#).

- Added `boost::program_options` to `db_generator` in order to increase its usability. This closes #15940. #15973 ([Nikita Mikhaylov](#)).

## ClickHouse release 20.10

### ClickHouse release v20.10.7.4-stable, 2020-12-24

#### Bug Fix

- Fixed issue when `clickhouse-odbc-bridge` process is unreachable by server on machines with dual IPv4/IPv6 stack and fixed issue when ODBC dictionary updates are performed using malformed queries and/or cause crashes. This possibly closes #14489. #18278 ([Denis Glazachev](#)).
- Fix key comparison between `Enum` and `Int` types. This fixes #17989. #18214 ([Amos Bird](#)).
- Fixed unique key convert crash in `MaterializeMySQL` database engine. This fixes #18186 and fixes #16372 #18211 ([Winter Zhang](#)).
- Fixed `std::out_of_range: basic_string` in S3 URL parsing. #18059 ([Vladimir Chebotarev](#)).
- Fixed the issue when some tables not synchronized to ClickHouse from MySQL caused by the fact that conversion MySQL prefix index wasn't supported for `MaterializeMySQL`. This fixes #15187 and fixes #17912 #17944 ([Winter Zhang](#)).
- Fix possible segfault in `topK` aggregate function. This closes #17404. #17845 ([Maksim Kita](#)).
- Do not restore parts from `WAL` if `in_memory_parts_enable_wal` is disabled. #17802 ([detailyang](#)).
- Fixed problem when ClickHouse fails to resume connection to MySQL servers. #17681 ([Alexander Kazakov](#)).
- Fixed empty `system.stack_trace` table when server is running in daemon mode. #17630 ([Amos Bird](#)).
- Fixed the behaviour when `clickhouse-client` is used in interactive mode with multiline queries and single line comment was erroneously extended till the end of query. This fixes #13654. #17565 ([alexey-milovidov](#)).
- Fixed the issue when server can stop accepting connections in very rare cases. #17542 ([alexey-milovidov](#)).
- Fixed `ALTER` query hang when the corresponding mutation was killed on the different replica. This fixes #16953. #17499 ([alesapin](#)).
- Fixed bug when mark cache size was underestimated by `clickhouse`. It may happen when there are a lot of tiny files with marks. #17496 ([alesapin](#)).
- Fixed `ORDER BY` with enabled setting `optimize_redundant_functions_in_order_by`. #17471 ([Anton Popov](#)).
- Fixed duplicates after `DISTINCT` which were possible because of incorrect optimization. Fixes #17294. #17296 ([li chengxiang](#)). #17439 ([Nikolai Kochetov](#)).
- Fixed crash while reading from `JOIN` table with `LowCardinality` types. This fixes #17228. #17397 ([Nikolai Kochetov](#)).
- Fixed set index invalidation when there are const columns in the subquery. This fixes #17246 . #17249 ([Amos Bird](#)).
- Fixed `ColumnConst` comparison which leads to crash. This fixed #17088 . #17135 ([Amos Bird](#)).
- Fixed bug when ON CLUSTER queries may hang forever for non-leader ReplicatedMergeTreeTables. #17089 ([alesapin](#)).

- Fixed fuzzer-found bug in function `fuzzBits`. This fixes #16980. #17051 (hexiaoting).
- Avoid unnecessary network errors for remote queries which may be cancelled while execution, like queries with `LIMIT`. #17006 (Azat Khuzhin).
- Fixed wrong result in big integers (128, 256 bit) when casting from double. #16986 (Mike).
- Reresolve the IP of the `format_avro_schema_registry_url` in case of errors. #16985 (filimonov).
- Fixed possible server crash after `ALTER TABLE ... MODIFY COLUMN ... NewType` when `SELECT` have `WHERE` expression on altering column and alter does not finished yet. #16968 (Amos Bird).
- Blame info was not calculated correctly in `clickhouse-git-import`. #16959 (alexey-milovidov).
- Fixed order by optimization with monotonous functions. This fixes #16107. #16956 (Anton Popov).
- Fixrf optimization of group by with enabled setting `optimize_aggregators_of_group_by_keys` and joins. This fixes #12604. #16951 (Anton Popov).
- Install script should always create subdirs in config folders. This is only relevant for Docker build with custom config. #16936 (filimonov).
- Fixrf possible error `Illegal type of argument` for queries with `ORDER BY`. This fixes #16580. #16928 (Nikolai Kochetov).
- Abort multipart upload if no data was written to `WriteBufferFromS3`. #16840 (Pavel Kovalenko).
- Fixed crash when using `any` without any arguments. This fixes #16803. #16826 (Amos Bird).
- Fixed the behaviour when ClickHouse used to always return 0 instead of a number of affected rows for `INSERT` queries via MySQL protocol. This fixes #16605. #16715 (Winter Zhang).
- Fixed uncontrolled growth of `TDigest`. #16680 (hrissan).
- Fixed remote query failure when using suffix `if` in Aggregate function. This fixes #16574 fixes #16231 #16610 (Winter Zhang).

## ClickHouse release v20.10.4.1-stable, 2020-11-13

### Bug Fix

- Fix rare silent crashes when query profiler is on and ClickHouse is installed on OS with glibc version that has (supposedly) broken asynchronous unwind tables for some functions. This fixes #15301. This fixes #13098. #16846 (alexey-milovidov).
- Fix `IN` operator over several columns and tuples with enabled `transform_null_in` setting. Fixes #15310. #16722 (Anton Popov).
- This will fix `optimize_read_in_order/optimize_aggregation_in_order` with `max_threads>0` and expression in `ORDER BY`. #16637 (Azat Khuzhin).
- Now when parsing AVRO from input the `LowCardinality` is removed from type. Fixes #16188. #16521 (Mike).
- Fix rapid growth of metadata when using MySQL Master -> MySQL Slave -> ClickHouse MaterializeMySQL Engine, and `slave_parallel_worker` enabled on MySQL Slave, by properly shrinking GTID sets. This fixes #15951. #16504 (TCeason).
- Fix `DROP TABLE` for Distributed (racy with `INSERT`). #16409 (Azat Khuzhin).
- Fix processing of very large entries in replication queue. Very large entries may appear in `ALTER` queries if table structure is extremely large (near 1 MB). This fixes #16307. #16332 (alexey-milovidov).

- Fix bug with MySQL database. When MySQL server used as database engine is down some queries raise Exception, because they try to get tables from disabled server, while it's unnecessary. For example, query `SELECT ... FROM system.parts` should work only with MergeTree tables and don't touch MySQL database at all. [#16032 \(Kruglov Pavel\)](#).

## Improvement

- Workaround for use S3 with nginx server as proxy. Nginx currently does not accept urls with empty path like <http://domain.com?delete>, but vanilla aws-sdk-cpp produces this kind of urls. This commit uses patched aws-sdk-cpp version, which makes urls with "/" as path in this cases, like <http://domain.com/?delete>. [#16813 \(ianton-ru\)](#).

## ClickHouse release v20.10.3.30, 2020-10-28

### Backward Incompatible Change

- Make `multiple_joins_rewriter_version` obsolete. Remove first version of joins rewriter. [#15472 \(Artem Zuikov\)](#).
- Change default value of `format_regexpEscapingRule` setting (it's related to `Regexp` format) to `Raw` (it means - read whole subpattern as a value) to make the behaviour more like to what users expect. [#15426 \(alexey-milovidov\)](#).
- Add support for nested multiline comments `/* comment /* comment */ */` in SQL. This conforms to the SQL standard. [#14655 \(alexey-milovidov\)](#).
- Added MergeTree settings (`max_replicated_merges_with_ttl_in_queue` and `max_number_of_merges_with_ttl_in_pool`) to control the number of merges with TTL in the background pool and replicated queue. This change breaks compatibility with older versions only if you use delete TTL. Otherwise, replication will stay compatible. You can avoid incompatibility issues if you update all shard replicas at once or execute `SYSTEM STOP TTL MERGES` until you finish the update of all replicas. If you'll get an incompatible entry in the replication queue, first of all, execute `SYSTEM STOP TTL MERGES` and after `ALTER TABLE ... DETACH PARTITION ...` the partition where incompatible TTL merge was assigned. Attach it back on a single replica. [#14490 \(alesapin\)](#).
- When upgrading from versions older than 20.5, if rolling update is performed and cluster contains both versions 20.5 or greater and less than 20.5, if ClickHouse nodes with old versions are restarted and old version has been started up in presence of newer versions, it may lead to `Part ... intersects previous part` errors. To prevent this error, first install newer clickhouse-server packages on all cluster nodes and then do restarts (so, when clickhouse-server is restarted, it will start up with the new version).

### New Feature

- Background data recompression. Add the ability to specify `TTL ... RECOMPRESS codec_name` for MergeTree table engines family. [#14494 \(alesapin\)](#).
- Add parallel quorum inserts. This closes [#15601](#). [#15601 \(Latysheva Alexandra\)](#).
- Settings for additional enforcement of data durability. Useful for non-replicated setups. [#11948 \(Anton Popov\)](#).
- When duplicate block is written to replica where it does not exist locally (has not been fetched from replicas), don't ignore it and write locally to achieve the same effect as if it was successfully replicated. [#11684 \(alexey-milovidov\)](#).
- Now we support `WITH <identifier> AS (subquery) ...` to introduce named subqueries in the query context. This closes [#2416](#). This closes [#4967](#). [#14771 \(Amos Bird\)](#).

- Introduce `enable_global_with_statement` setting which propagates the first select's `WITH` statements to other select queries at the same level, and makes aliases in `WITH` statements visible to subqueries. #15451 (Amos Bird).
- Secure inter-cluster query execution (with `initial_user` as current query user). #13156 (Azat Khuzhin). #15551 (Azat Khuzhin).
- Add the ability to remove column properties and table TTLs. Introduced queries `ALTER TABLE MODIFY COLUMN col_name REMOVE what_to_remove` and `ALTER TABLE REMOVE TTL`. Both operations are lightweight and executed at the metadata level. #14742 (alesapin).
- Added format `RawBLOB`. It is intended for input or output a single value without any escaping and delimiters. This closes #15349. #15364 (alexey-milovidov).
- Add the `reinterpretAsUUID` function that allows to convert a big-endian byte string to UUID. #15480 (Alexander Kuzmenkov).
- Implement `force_data_skipping_indices` setting. #15642 (Azat Khuzhin).
- Add a setting `output_format_pretty_row_numbers` to numerate the result in Pretty formats. This closes #15350. #15443 (flynn).
- Added query obfuscation tool. It allows to share more queries for better testing. This closes #15268. #15321 (alexey-milovidov).
- Add table function `null('structure')`. #14797 (vxider).
- Added `formatReadableQuantity` function. It is useful for reading big numbers by human. #14725 (Artem Hnilov).
- Add format `LineAsString` that accepts a sequence of lines separated by newlines, every line is parsed as a whole as a single String field. #14703 (Nikita Mikhaylov), #13846 (hexiaoting).
- Add `JSONStrings` format which output data in arrays of strings. #14333 (hc).
- Add support for "Raw" column format for `Regexp` format. It allows to simply extract subpatterns as a whole without any escaping rules. #15363 (alexey-milovidov).
- Allow configurable `NULL` representation for `TSV` output format. It is controlled by the setting `output_format_tsv_null_representation` which is `\N` by default. This closes #9375. Note that the setting only controls output format and `\N` is the only supported `NULL` representation for `TSV` input format. #14586 (Kruglov Pavel).
- Support Decimal data type for `MaterializeMySQL`. `MaterializeMySQL` is an experimental feature. #14535 (Winter Zhang).
- Add new feature: `SHOW DATABASES LIKE 'xxx'`. #14521 (hexiaoting).
- Added a script to import (arbitrary) git repository to ClickHouse as a sample dataset. #14471 (alexey-milovidov).
- Now insert statements can have asterisk (or variants) with column transformers in the column list. #14453 (Amos Bird).
- New query complexity limit settings `max_rows_to_read_leaf`, `max_bytes_to_read_leaf` for distributed queries to limit max rows/bytes read on the leaf nodes. Limit is applied for local reads only, *excluding* the final merge stage on the root node. #14221 (Roman Khavronenko).

- Allow user to specify settings for ReplicatedMergeTree\* storage in `<replicated_merge_tree>` section of config file. It works similarly to `<merge_tree>` section. For ReplicatedMergeTree\* storages settings from `<merge_tree>` and `<replicated_merge_tree>` are applied together, but settings from `<replicated_merge_tree>` has higher priority. Added system.replicated\_merge\_tree\_settings table. #13573 (Amos Bird).
- Add `mapPopulateSeries` function. #13166 (Ildus Kurbangaliev).
- Supporting MySQL types: `decimal` (as ClickHouse `Decimal`) and `datetime` with sub-second precision (as `DateTime64`). #11512 (Vasily Nemkov).
- Introduce `event_time_microseconds` field to `system.text_log`, `system.trace_log`, `system.query_log` and `system.query_thread_log` tables. #14760 (Bharat Nallan).
- Add `event_time_microseconds` to `system.asynchronous_metric_log` & `system.metric_log` tables. #14514 (Bharat Nallan).
- Add `query_start_time_microseconds` field to `system.query_log` & `system.query_thread_log` tables. #14252 (Bharat Nallan).

## Bug Fix

- Fix the case when memory can be overallocated regardless to the limit. This closes #14560. #16206 (alexey-milovidov).
- Fix `executable` dictionary source hang. In previous versions, when using some formats (e.g. `JSONEachRow`) data was not feed to a child process before it outputs at least something. This closes #1697. This closes #2455. #14525 (alexey-milovidov).
- Fix double free in case of exception in function `dictGet`. It could have happened if dictionary was loaded with error. #16429 (Nikolai Kochetov).
- Fix group by with totals/rollup/cube modifiers and min/max functions over group by keys. Fixes #16393. #16397 (Anton Popov).
- Fix async Distributed INSERT with `prefer_localhost_replica=0` and `internal_replication`. #16358 (Azat Khuzhin).
- Fix a very wrong code in `TwoLevelStringHashTable` implementation, which might lead to memory leak. #16264 (Amos Bird).
- Fix segfault in some cases of wrong aggregation in lambdas. #16082 (Anton Popov).
- Fix `ALTER MODIFY ... ORDER BY` query hang for `ReplicatedVersionedCollapsingMergeTree`. This fixes #15980. #16011 (alesapin).
- MaterializeMySQL (experimental feature): Fix collate name & charset name parser and support `length = 0` for string type. #16008 (Winter Zhang).
- Allow to use `direct` layout for dictionaries with complex keys. #16007 (Anton Popov).
- Prevent replica hang for 5-10 mins when replication error happens after a period of inactivity. #15987 (filimonov).
- Fix rare segfaults when inserting into or selecting from `MaterializedView` and concurrently dropping target table (for Atomic database engine). #15984 (tavplubix).
- Fix ambiguity in parsing of settings profiles: `CREATE USER ... SETTINGS profile readonly` is now considered as using a profile named `readonly`, not a setting named `profile` with the `readonly` constraint. This fixes #15628. #15982 (Vitaly Baranov).
- MaterializeMySQL (experimental feature): Fix crash on create database failure. #15954 (Winter Zhang).

- Fixed `DROP TABLE IF EXISTS` failure with `Table ... does not exist` error when table is concurrently renamed (for Atomic database engine). Fixed rare deadlock when concurrently executing some DDL queries with multiple tables (like `DROP DATABASE` and `RENAME TABLE`) - Fixed `DROP/DETACH DATABASE` failure with `Table ... does not exist` when concurrently executing `DROP/DETACH TABLE`. [#15934 \(tavplubix\)](#).
- Fix incorrect empty result for query from Distributed table if query has `WHERE`, `PREWHERE` and `GLOBAL IN`. Fixes [#15792](#). [#15933 \(Nikolai Kochetov\)](#).
- Fixes [#12513](#): difference expressions with same alias when query is reanalyzed. [#15886 \(Winter Zhang\)](#).
- Fix possible very rare deadlocks in RBAC implementation. [#15875 \(Vitaly Baranov\)](#).
- Fix exception `Block structure mismatch` in `SELECT ... ORDER BY DESC` queries which were executed after `ALTER MODIFY COLUMN` query. Fixes [#15800](#). [#15852 \(alesapin\)](#).
- MaterializeMySQL (experimental feature): Fix `select count()` inaccuracy. [#15767 \(tavplubix\)](#).
- Fix some cases of queries, in which only virtual columns are selected. Previously `Not found column _nothing in block` exception may be thrown. Fixes [#12298](#). [#15756 \(Anton Popov\)](#).
- Fix drop of materialized view with inner table in Atomic database (hangs all subsequent `DROP TABLE` due to hang of the worker thread, due to recursive `DROP TABLE` for inner table of MV). [#15743 \(Azat Khuzhin\)](#).
- Possibility to move part to another disk/volume if the first attempt was failed. [#15723 \(Pavel Kovalenko\)](#).
- Fix error `Cannot find column` which may happen at insertion into `MATERIALIZED VIEW` in case if query for `MV` contains `ARRAY JOIN`. [#15717 \(Nikolai Kochetov\)](#).
- Fixed too low default value of `max_replicated_logs_to_keep` setting, which might cause replicas to become lost too often. Improve lost replica recovery process by choosing the most up-to-date replica to clone. Also do not remove old parts from lost replica, detach them instead. [#15701 \(tavplubix\)](#).
- Fix rare race condition in dictionaries and tables from MySQL. [#15686 \(alesapin\)](#).
- Fix (benign) race condition in AMQP-CPP. [#15667 \(alesapin\)](#).
- Fix error `Cannot add simple transform to empty Pipe` which happened while reading from `Buffer` table which has different structure than destination table. It was possible if destination table returned empty result for query. Fixes [#15529](#). [#15662 \(Nikolai Kochetov\)](#).
- Proper error handling during insert into MergeTree with S3. MergeTree over S3 is an experimental feature. [#15657 \(Pavel Kovalenko\)](#).
- Fixed bug with S3 table function: region from URL was not applied to S3 client configuration. [#15646 \(Vladimir Chebotarev\)](#).
- Fix the order of destruction for resources in `ReadFromStorage` step of query plan. It might cause crashes in rare cases. Possibly connected with [#15610](#). [#15645 \(Nikolai Kochetov\)](#).
- Subtract `ReadonlyReplica` metric when detach readonly tables. [#15592 \(sundyli\)](#).
- Fixed `Element ... is not a constant expression` error when using `JSON*` function result in `VALUES`, `LIMIT` or right side of `IN` operator. [#15589 \(tavplubix\)](#).
- Query will finish faster in case of exception. Cancel execution on remote replicas if exception happens. [#15578 \(Azat Khuzhin\)](#).
- Prevent the possibility of error message `Could not calculate available disk space (statvfs), errno: 4, strerror: Interrupted system call.` This fixes [#15541](#). [#15557 \(alexey-milovidov\)](#).

- Fix Database <db> does not exist. in queries with IN and Distributed table when there's no database on initiator. [#15538](#) ([Artem Zuikov](#)).
- Mutation might hang waiting for some non-existent part after MOVE or REPLACE PARTITION or, in rare cases, after DETACH or DROP PARTITION. It's fixed. [#15537](#) ([tavplubix](#)).
- Fix bug when ILIKE operator stops being case insensitive if LIKE with the same pattern was executed. [#15536](#) ([alesapin](#)).
- Fix Missing columns errors when selecting columns which absent in data, but depend on other columns which also absent in data. Fixes [#15530](#). [#15532](#) ([alesapin](#)).
- Throw an error when a single parameter is passed to ReplicatedMergeTree instead of ignoring it. [#15516](#) ([nvartolomei](#)).
- Fix bug with event subscription in DDLWorker which rarely may lead to query hangs in ON CLUSTER. Introduced in [#13450](#). [#15477](#) ([alesapin](#)).
- Report proper error when the second argument of boundingRatio aggregate function has a wrong type. [#15407](#) ([detailyang](#)).
- Fixes [#15365](#): attach a database with MySQL engine throws exception (no query context). [#15384](#) ([Winter Zhang](#)).
- Fix the case of multiple occurrences of column transformers in a select query. [#15378](#) ([Amos Bird](#)).
- Fixed compression in S3 storage. [#15376](#) ([Vladimir Chebotarev](#)).
- Fix bug where queries like SELECT toStartOfDay(today()) fail complaining about empty time\_zone argument. [#15319](#) ([Bharat Nallan](#)).
- Fix race condition during MergeTree table rename and background cleanup. [#15304](#) ([alesapin](#)).
- Fix rare race condition on server startup when system logs are enabled. [#15300](#) ([alesapin](#)).
- Fix hang of queries with a lot of subqueries to same table of MySQL engine. Previously, if there were more than 16 subqueries to same MySQL table in query, it hang forever. [#15299](#) ([Anton Popov](#)).
- Fix MSan report in QueryLog. Uninitialized memory can be used for the field memory\_usage. [#15258](#) ([alexey-milovidov](#)).
- Fix 'Unknown identifier' in GROUP BY when query has JOIN over Merge table. [#15242](#) ([Artem Zuikov](#)).
- Fix instance crash when using joinGet with LowCardinality types. This fixes [#15214](#). [#15220](#) ([Amos Bird](#)).
- Fix bug in table engine Buffer which does not allow to insert data of new structure into Buffer after ALTER query. Fixes [#15117](#). [#15192](#) ([alesapin](#)).
- Adjust Decimal field size in MySQL column definition packet. [#15152](#) ([maqroll](#)).
- Fixes Data compressed with different methods in join\_algorithm='auto'. Keep LowCardinality as type for left table join key in join\_algorithm='partial\_merge'. [#15088](#) ([Artem Zuikov](#)).
- Update jemalloc to fix percpu\_arena with affinity mask. [#15035](#) ([Azat Khuzhin](#)). [#14957](#) ([Azat Khuzhin](#)).
- We already use padded comparison between String and FixedString (<https://github.com/ClickHouse/ClickHouse/blob/master/src/Functions/FunctionsComparison.h#L333>). This PR applies the same logic to field comparison which corrects the usage of FixedString as primary keys. This fixes [#14908](#). [#15033](#) ([Amos Bird](#)).

- If function `bar` was called with specifically crafted arguments, buffer overflow was possible. This closes [#13926](#). [#15028 \(alexey-milovidov\)](#).
- Fixed Cannot rename ... errno: 22, strerror: Invalid argument error on DDL query execution in Atomic database when running clickhouse-server in Docker on Mac OS. [#15024 \(tavplubix\)](#).
- Fix crash in RIGHT or FULL JOIN with `join_algorithm='auto'` when memory limit exceeded and we should change HashJoin with MergeJoin. [#15002 \(Artem Zuikov\)](#).
- Now settings `number_of_free_entries_in_pool_to_execute_mutation` and `number_of_free_entries_in_pool_to_lower_max_size_of_merge` can be equal to `background_pool_size`. [#14975 \(alesapin\)](#).
- Fix to make predicate push down work when subquery contains `finalizeAggregation` function. Fixes [#14847](#). [#14937 \(filimonov\)](#).
- Publish CPU frequencies per logical core in `system.asynchronous_metrics`. This fixes [#14923](#). [#14924 \(Alexander Kuzmenkov\)](#).
- MaterializeMySQL (experimental feature): Fixed `.metadata.tmp` File exists error. [#14898 \(Winter Zhang\)](#).
- Fix the issue when some invocations of `extractAllGroups` function may trigger "Memory limit exceeded" error. This fixes [#13383](#). [#14889 \(alexey-milovidov\)](#).
- Fix SIGSEGV for an attempt to INSERT into StorageFile with file descriptor. [#14887 \(Azat Khuzhin\)](#).
- Fixed segfault in `cache` dictionary [#14837](#). [#14879 \(Nikita Mikhaylov\)](#).
- MaterializeMySQL (experimental feature): Fixed bug in parsing MySQL binlog events, which causes `Attempt` to read after eof and `Packet payload` is not fully read in MaterializeMySQL database engine. [#14852 \(Winter Zhang\)](#).
- Fix rare error in `SELECT` queries when the queried column has `DEFAULT` expression which depends on the other column which also has `DEFAULT` and not present in select query and not exists on disk. Partially fixes [#14531](#). [#14845 \(alesapin\)](#).
- Fix a problem where the server may get stuck on startup while talking to ZooKeeper, if the configuration files have to be fetched from ZK (using the `from_zk` include option). This fixes [#14814](#). [#14843 \(Alexander Kuzmenkov\)](#).
- Fix wrong monotonicity detection for shrunk `Int -> Int` cast of signed types. It might lead to incorrect query result. This bug is unveiled in [#14513](#). [#14783 \(Amos Bird\)](#).
- Replace column transformer should replace identifiers with cloned ASTs. This fixes [#14695](#). [#14734 \(Amos Bird\)](#).
- Fixed missed default database name in metadata of materialized view when executing `ALTER ... MODIFY QUERY`. [#14664 \(tavplubix\)](#).
- Fix bug when `ALTER UPDATE` mutation with `Nullable` column in assignment expression and constant value (like `UPDATE x = 42`) leads to incorrect value in column or segfault. Fixes [#13634](#), [#14045](#). [#14646 \(alesapin\)](#).
- Fix wrong Decimal multiplication result caused wrong decimal scale of result column. [#14603 \(Artem Zuikov\)](#).
- Fix function `has` with `LowCardinality` of `Nullable`. [#14591 \(Mike\)](#).
- Cleanup data directory after Zookeeper exceptions during CreateQuery for StorageReplicatedMergeTree Engine. [#14563 \(Bharat Nallan\)](#).

- Fix rare segfaults in functions with combinator `-Resample`, which could appear in result of overflow with very large parameters. [#14562](#) ([Anton Popov](#)).
- Fix a bug when converting `Nullable(String)` to `Enum`. Introduced by [#12745](#). This fixes [#14435](#). [#14530](#) ([Amos Bird](#)).
- Fixed the incorrect sorting order of `Nullable` column. This fixes [#14344](#). [#14495](#) ([Nikita Mikhaylov](#)).
- Fix `currentDatabase()` function cannot be used in `ON CLUSTER` ddl query. [#14211](#) ([Winter Zhang](#)).
- `MaterializeMySQL` (experimental feature): Fixed `Packet payload is not fully read` error in `MaterializeMySQL` database engine. [#14696](#) ([BohuTANG](#)).

## Improvement

- Enable `Atomic` database engine by default for newly created databases. [#15003](#) ([tavplubix](#)).
- Add the ability to specify specialized codecs like `Delta`, `T64`, etc. for columns with subtypes. Implements [#12551](#), fixes [#11397](#), fixes [#4609](#). [#15089](#) ([alesapin](#)).
- Dynamic reload of zookeeper config. [#14678](#) ([sundyli](#)).
- Now it's allowed to execute `ALTER ... ON CLUSTER` queries regardless of the `<internal_replication>` setting in cluster config. [#16075](#) ([alesapin](#)).
- Now `joinGet` supports multi-key lookup. Continuation of [#12418](#). [#13015](#) ([Amos Bird](#)).
- Wait for `DROP/DETACH TABLE` to actually finish if `NO DELAY` or `SYNC` is specified for `Atomic` database. [#15448](#) ([tavplubix](#)).
- Now it's possible to change the type of version column for `VersionedCollapsingMergeTree` with `ALTER` query. [#15442](#) ([alesapin](#)).
- Unfold `{database}`, `{table}` and `{uuid}` macros in `zookeeper_path` on replicated table creation. Do not allow `RENAME TABLE` if it may break `zookeeper_path` after server restart. Fixes [#6917](#). [#15348](#) ([tavplubix](#)).
- The function `now` allows an argument with timezone. This closes [#15264](#). [#15285](#) ([flynn](#)).
- Do not allow connections to ClickHouse server until all scripts in `/docker-entrypoint-initdb.d/` are executed. [#15244](#) ([Aleksei Kozharin](#)).
- Added `optimize` setting to `EXPLAIN PLAN` query. If enabled, query plan level optimisations are applied. Enabled by default. [#15201](#) ([Nikolai Kochetov](#)).
- Proper exception message for wrong number of arguments of `CAST`. This closes [#13992](#). [#15029](#) ([alexey-milovidov](#)).
- Add option to disable TTL move on data part insert. [#15000](#) ([Pavel Kovalenko](#)).
- Ignore key constraints when doing mutations. Without this pull request, it's not possible to do mutations when `force_index_by_date = 1` or `force_primary_key = 1`. [#14973](#) ([Amos Bird](#)).
- Allow to drop Replicated table if previous drop attempt was failed due to ZooKeeper session expiration. This fixes [#11891](#). [#14926](#) ([alexey-milovidov](#)).
- Fixed excessive settings constraint violation when running `SELECT` with `SETTINGS` from a distributed table. [#14876](#) ([Amos Bird](#)).
- Provide a `load_balancing_first_offset` query setting to explicitly state what the first replica is. It's used together with `FIRST_OR_RANDOM` load balancing strategy, which allows to control replicas workload. [#14867](#) ([Amos Bird](#)).

- Show subqueries for `SET` and `JOIN` in `EXPLAIN` result. #14856 (Nikolai Kochetov).
- Allow using multi-volume storage configuration in storage `Distributed`. #14839 (Pavel Kovalenko).
- Construct `query_start_time` and `query_start_time_microseconds` from the same timespec. #14831 (Bharat Nallan).
- Support for disabling persistency for `StorageJoin` and `StorageSet`, this feature is controlled by setting `disable_set_and_join_persistency`. And this PR solved issue #6318. #14776 (vxider).
- Now `COLUMNS` can be used to wrap over a list of columns and apply column transformers afterwards. #14775 (Amos Bird).
- Add `merge_algorithm` to `system.merges` table to improve merging inspections. #14705 (Amos Bird).
- Fix potential memory leak caused by zookeeper exists watch. #14693 (hustnn).
- Allow parallel execution of distributed DDL. #14684 (Azat Khuzhin).
- Add `QueryMemoryLimitExceeded` event counter. This closes #14589. #14647 (fastio).
- Fix some trailing whitespaces in query formatting. #14595 (Azat Khuzhin).
- ClickHouse treats partition expr and key expr differently. Partition expr is used to construct an minmax index containing related columns, while primary key expr is stored as an expr. Sometimes user might partition a table at coarser levels, such as `partition by i / 1000`. However, binary operators are not monotonic and this PR tries to fix that. It might also benefit other use cases. #14513 (Amos Bird).
- Add an option to skip access checks for `DiskS3`. `s3` disk is an experimental feature. #14497 (Pavel Kovalenko).
- Speed up server shutdown process if there are ongoing S3 requests. #14496 (Pavel Kovalenko).
- `SYSTEM RELOAD CONFIG` now throws an exception if failed to reload and continues using the previous `users.xml`. The background periodic reloading also continues using the previous `users.xml` if failed to reload. #14492 (Vitaly Baranov).
- For `INSERTs` with inline data in `VALUES` format in the script mode of `clickhouse-client`, support semicolon as the data terminator, in addition to the new line. Closes #12288. #13192 (Alexander Kuzmenkov).
- Support custom codecs in compact parts. #12183 (Anton Popov).

## Performance Improvement

- Enable compact parts by default for small parts. This will allow to process frequent inserts slightly more efficiently (4..100 times). #11913 (alexey-milovidov).
- Improve `quantileTDigest` performance. This fixes #2668. #15542 (Kruglov Pavel).
- Significantly reduce memory usage in `AggregatingInOrderTransform/optimize_aggregation_in_order`. #15543 (Azat Khuzhin).
- Faster 256-bit multiplication. #15418 (Artem Zuikov).
- Improve performance of 256-bit types using `(u)int64_t` as base type for wide integers. Original wide integers use 8-bit types as base. #14859 (Artem Zuikov).
- Explicitly use a temporary disk to store vertical merge temporary data. #15639 (Grigory Pervakov).
- Use one S3 `DeleteObjects` request instead of multiple `DeleteObject` in a loop. No functionality changes, so covered by existing tests like `integration/test_log_family_s3`. #15238 (ianton-ru).

- Fix `DateTime <op> DateTime` mistakenly choosing the slow generic implementation. This fixes #15153. #15178 (Amos Bird).
- Improve performance of GROUP BY key of type `FixedString`. #15034 (Amos Bird).
- Only `mlock` code segment when starting clickhouse-server. In previous versions, all mapped regions were locked in memory, including debug info. Debug info is usually splitted to a separate file but if it isn't, it led to +2..3 GiB memory usage. #14929 (alexey-milovidov).
- ClickHouse binary become smaller due to link time optimization.

## Build/Testing/Packaging Improvement

- Now we use clang-11 for production ClickHouse build. #15239 (alesapin).
- Now we use clang-11 to build ClickHouse in CI. #14846 (alesapin).
- Switch binary builds (Linux, Darwin, AArch64, FreeDSD) to clang-11. #15622 (Ilya Yatsishin).
- Now all test images use `l1vm-symbolizer-11`. #15069 (alesapin).
- Allow to build with l1vm-11. #15366 (alexey-milovidov).
- Switch from `clang-tidy-10` to `clang-tidy-11`. #14922 (alexey-milovidov).
- Use LLVM's experimental pass manager by default. #15608 (Danila Kutenin).
- Don't allow any C++ translation unit to build more than 10 minutes or to use more than 10 GB of memory. This fixes #14925. #15060 (alexey-milovidov).
- Make performance test more stable and representative by splitting test runs and profile runs. #15027 (alexey-milovidov).
- Attempt to make performance test more reliable. It is done by remapping the executable memory of the process on the fly with `madvise` to use transparent huge pages - it can lower the number of iTLB misses which is the main source of instabilities in performance tests. #14685 (alexey-milovidov).
- Convert to python3. This closes #14886. #15007 (Azat Khuzhin).
- Fail early in functional tests if server failed to respond. This closes #15262. #15267 (alexey-milovidov).
- Allow to run AArch64 version of clickhouse-server without configs. This facilitates #15174. #15266 (alexey-milovidov).
- Improvements in CI docker images: get rid of ZooKeeper and single script for test configs installation. #15215 (alesapin).
- Fix CMake options forwarding in fast test script. Fixes error in #14711. #15155 (alesapin).
- Added a script to perform hardware benchmark in a single command. #15115 (alexey-milovidov).
- Splitted huge test `test_dictionaries_all_layouts_and_sources` into smaller ones. #15110 (Nikita Mikhaylov).
- Maybe fix MSan report in base64 (on servers with AVX-512). This fixes #14006. #15030 (alexey-milovidov).
- Reformat and cleanup code in all integration test \*.py files. #14864 (Bharat Nallan).
- Fix MaterializeMySQL empty transaction unstable test case found in CI. #14854 (Winter Zhang).
- Attempt to speed up build a little. #14808 (alexey-milovidov).
- Speed up build a little by removing unused headers. #14714 (alexey-milovidov).

- Fix build failure in OSX. #14761 (Winter Zhang).
- Enable ccache by default in cmake if it's found in OS. #14575 (alesapin).
- Control CI builds configuration from the ClickHouse repository. #14547 (alesapin).
- In CMake files: - Moved some options' descriptions' parts to comments above. - Replace 0 -> OFF, 1 -> ON in `options` default values. - Added some descriptions and links to docs to the options. - Replaced FUZZER option (there is another option `ENABLE_FUZZING` which also enables same functionality). - Removed `ENABLE_GTEST_LIBRARY` option as there is `ENABLE_TESTS`. See the full description in PR: #14711 (Mike).
- Make binary a bit smaller (~50 Mb for debug version). #14555 (Artem Zuikov).
- Use `std::filesystem::path` in `ConfigProcessor` for concatenating file paths. #14558 (Bharat Nallan).
- Fix debug assertion in `bitShiftLeft()` when called with negative big integer. #14697 (Artem Zuikov).

## ClickHouse release 20.9

### ClickHouse release v20.9.7.11-stable, 2020-12-07

#### Performance Improvement

- Fix performance of reading from `Merge` tables over huge number of `MergeTree` tables. Fixes #7748. #16988 (Anton Popov).

#### Bug Fix

- Do not restore parts from WAL if `in_memory_parts_enable_wal` is disabled. #17802 (detailyang).
- Fixed segfault when there is not enough space when inserting into `Distributed` table. #17737 (tavplubix).
- Fixed problem when ClickHouse fails to resume connection to MySQL servers. #17681 (Alexander Kazakov).
- Fixed `Function not implemented` error when executing `RENAME` query in `Atomic` database with ClickHouse running on Windows Subsystem for Linux. Fixes #17661. #17664 (tavplubix).
- When `clickhouse-client` is used in interactive mode with multiline queries, single line comment was erroneously extended till the end of query. This fixes #13654. #17565 (alexey-milovidov).
- Fix the issue when server can stop accepting connections in very rare cases. #17542 (alexey-milovidov).
- Fix alter query hang when the corresponding mutation was killed on the different replica. Fixes #16953. #17499 (alesapin).
- Fix bug when mark cache size was underestimated by clickhouse. It may happen when there are a lot of tiny files with marks. #17496 (alesapin).
- Fix `ORDER BY` with enabled setting `optimize_redundant_functions_in_order_by`. #17471 (Anton Popov).
- Fix duplicates after `DISTINCT` which were possible because of incorrect optimization. Fixes #17294. #17296 (li chengxiang). #17439 (Nikolai Kochetov).
- Fix crash while reading from `JOIN` table with `LowCardinality` types. Fixes #17228. #17397 (Nikolai Kochetov).
- Fix set index invalidation when there are `const` columns in the subquery. This fixes #17246 . #17249 (Amos Bird).
- Fix `ColumnConst` comparison which leads to crash. This fixed #17088 . #17135 (Amos Bird).

- Fixed crash on `CREATE TABLE ... AS some_table` query when `some_table` was created `AS table_function()`. Fixes #16944. #17072 (tavplubix).
- Bug fix for funciton fuzzBits, related issue: #16980. #17051 (hexiaoting).
- Avoid unnecessary network errors for remote queries which may be cancelled while execution, like queries with `LIMIT`. #17006 (Azat Khuzhin).
- TODO. #16866 (tavplubix).
- Return number of affected rows for `INSERT` queries via MySQL protocol. Previously ClickHouse used to always return 0, it's fixed. Fixes #16605. #16715 (Winter Zhang).

## Build/Testing/Packaging Improvement

- Update embedded timezone data to version 2020d (also update cctz to the latest master). #17204 (filimonov).

## ClickHouse release v20.9.6.14-stable, 2020-11-20

### Improvement

- Make it possible to connect to `clickhouse-server` secure endpoint which requires SNI. This is possible when `clickhouse-server` is hosted behind TLS proxy. #16938 (filimonov).
- Conditional aggregate functions (for example: `avgIf`, `sumIf`, `maxIf`) should return `NULL` when miss rows and use nullable arguments. #13964 (Winter Zhang).

### Bug Fix

- Fix bug when `ON CLUSTER` queries may hang forever for non-leader ReplicatedMergeTreeTables. #17089 (alesapin).
- Reresolve the IP of the `format_avro_schema_registry_url` in case of errors. #16985 (filimonov).
- Fix possible server crash after `ALTER TABLE ... MODIFY COLUMN ... NewType` when `SELECT` have `WHERE` expression on altering column and alter does not finished yet. #16968 (Amos Bird).
- Install script should always create subdirs in config folders. This is only relevant for Docker build with custom config. #16936 (filimonov).
- Fix possible error `Illegal type of argument` for queries with `ORDER BY`. Fixes #16580. #16928 (Nikolai Kochetov).
- Abort multipart upload if no data was written to `WriteBufferFromS3`. #16840 (Pavel Kovalenko).
- Fix crash when using `any` without any arguments. This is for #16803 . cc @azat. #16826 (Amos Bird).
- Fix `IN` operator over several columns and tuples with enabled `transform_null_in` setting. Fixes #15310. #16722 (Anton Popov).
- This will fix `optimize_read_in_order`/`optimize_aggregation_in_order` with `max_threads>0` and expression in `ORDER BY`. #16637 (Azat Khuzhin).
- fixes #16574 fixes #16231 fix remote query failure when using 'if' suffix aggregate function. #16610 (Winter Zhang).
- Query is finished faster in case of exception. Cancel execution on remote replicas if exception happens. #15578 (Azat Khuzhin).

## ClickHouse release v20.9.5.5-stable, 2020-11-13

## Bug Fix

- Fix rare silent crashes when query profiler is on and ClickHouse is installed on OS with glibc version that has (supposedly) broken asynchronous unwind tables for some functions. This fixes #15301. This fixes #13098. #16846 (alexey-milovidov).
- Now when parsing AVRO from input the LowCardinality is removed from type. Fixes #16188. #16521 (Mike).
- Fix rapid growth of metadata when using MySQL Master -> MySQL Slave -> ClickHouse MaterializeMySQL Engine, and slave\_parallel\_worker enabled on MySQL Slave, by properly shrinking GTID sets. This fixes #15951. #16504 (TCeason).
- Fix DROP TABLE for Distributed (racy with INSERT). #16409 (Azat Khuzhin).
- Fix processing of very large entries in replication queue. Very large entries may appear in ALTER queries if table structure is extremely large (near 1 MB). This fixes #16307. #16332 (alexey-milovidov).
- Fixed the inconsistent behaviour when a part of return data could be dropped because the set for its filtration wasn't created. #16308 (Nikita Mikhaylov).
- Fix bug with MySQL database. When MySQL server used as database engine is down some queries raise Exception, because they try to get tables from disabled server, while it's unnecessary. For example, query SELECT ... FROM system.parts should work only with MergeTree tables and don't touch MySQL database at all. #16032 (Kruglov Pavel).

## ClickHouse release v20.9.4.76-stable (2020-10-29)

### Bug Fix

- Fix double free in case of exception in function dictGet. It could have happened if dictionary was loaded with error. #16429 (Nikolai Kochetov).
- Fix group by with totals/rollup/cube modifiers and min/max functions over group by keys. Fixes #16393. #16397 (Anton Popov).
- Fix async Distributed INSERT w/ prefer\_localhost\_replica=0 and internal\_replication. #16358 (Azat Khuzhin).
- Fix a very wrong code in TwoLevelStringHashTable implementation, which might lead to memory leak. I'm surprised how this bug can lurk for so long.... #16264 (Amos Bird).
- Fix the case when memory can be overallocated regardless to the limit. This closes #14560. #16206 (alexey-milovidov).
- Fix ALTER MODIFY ... ORDER BY query hang for ReplicatedVersionedCollapsingMergeTree. This fixes #15980. #16011 (alesapin).
- Fix collate name & charset name parser and support length = 0 for string type. #16008 (Winter Zhang).
- Allow to use direct layout for dictionaries with complex keys. #16007 (Anton Popov).
- Prevent replica hang for 5-10 mins when replication error happens after a period of inactivity. #15987 (filimonov).
- Fix rare segfaults when inserting into or selecting from MaterializedView and concurrently dropping target table (for Atomic database engine). #15984 (tavplubix).
- Fix ambiguity in parsing of settings profiles: CREATE USER ... SETTINGS profile readonly is now considered as using a profile named readonly, not a setting named profile with the readonly constraint. This fixes #15628. #15982 (Vitaly Baranov).

- Fix a crash when database creation fails. #15954 (Winter Zhang).
- Fixed `DROP TABLE IF EXISTS` failure with `Table ... does not exist` error when table is concurrently renamed (for Atomic database engine). Fixed rare deadlock when concurrently executing some DDL queries with multiple tables (like `DROP DATABASE` and `RENAME TABLE`) Fixed `DROP/DETACH DATABASE` failure with `Table ... does not exist` when concurrently executing `DROP/DETACH TABLE`. #15934 (tavplubix).
- Fix incorrect empty result for query from Distributed table if query has `WHERE`, `PREWHERE` and `GLOBAL IN`. Fixes #15792. #15933 (Nikolai Kochetov).
- Fix possible deadlocks in RBAC. #15875 (Vitaly Baranov).
- Fix exception Block structure mismatch in `SELECT ... ORDER BY DESC` queries which were executed after `ALTER MODIFY COLUMN` query. Fixes #15800. #15852 (alesapin).
- Fix `select count()` inaccuracy for MaterializeMySQL. #15767 (tavplubix).
- Fix some cases of queries, in which only virtual columns are selected. Previously `Not found column _nothing in block` exception may be thrown. Fixes #12298. #15756 (Anton Popov).
- Fixed too low default value of `max_replicated_logs_to_keep` setting, which might cause replicas to become lost too often. Improve lost replica recovery process by choosing the most up-to-date replica to clone. Also do not remove old parts from lost replica, detach them instead. #15701 (tavplubix).
- Fix error `Cannot add simple transform to empty Pipe` which happened while reading from `Buffer` table which has different structure than destination table. It was possible if destination table returned empty result for query. Fixes #15529. #15662 (Nikolai Kochetov).
- Fixed bug with globs in S3 table function, region from URL was not applied to S3 client configuration. #15646 (Vladimir Chebotarev).
- Decrement the `ReadOnlyReplica` metric when detaching read-only tables. This fixes #15598. #15592 (sundyli).
- Throw an error when a single parameter is passed to `ReplicatedMergeTree` instead of ignoring it. #15516 (nvartolomei).

## Improvement

- Now it's allowed to execute `ALTER ... ON CLUSTER` queries regardless of the `<internal_replication>` setting in cluster config. #16075 (alesapin).
- Unfold `{database}`, `{table}` and `{uuid}` macros in `ReplicatedMergeTree` arguments on table creation. #16160 (tavplubix).

## ClickHouse release v20.9.3.45-stable (2020-10-09)

### Bug Fix

- Fix error `Cannot find column` which may happen at insertion into `MATERIALIZED VIEW` in case if query for `MV` contains `ARRAY JOIN`. #15717 (Nikolai Kochetov).
- Fix race condition in AMQP-CPP. #15667 (alesapin).
- Fix the order of destruction for resources in `ReadFromStorage` step of query plan. It might cause crashes in rare cases. Possibly connected with #15610. #15645 (Nikolai Kochetov).
- Fixed `Element ... is not a constant expression` error when using `JSON*` function result in `VALUES`, `LIMIT` or right side of `IN` operator. #15589 (tavplubix).

- Prevent the possibility of error message Could not calculate available disk space (statvfs), errno: 4, strerror: Interrupted system call. This fixes #15541. #15557 (alexey-milovidov).
- Significantly reduce memory usage in AggregatingInOrderTransform/optimize\_aggregation\_in\_order. #15543 (Azat Khuzhin).
- Mutation might hang waiting for some non-existent part after MOVE or REPLACE PARTITION or, in rare cases, after DETACH or DROP PARTITION. It's fixed. #15537 (tavplubix).
- Fix bug when ILIKE operator stops being case insensitive if LIKE with the same pattern was executed. #15536 (alesapin).
- Fix Missing columns errors when selecting columns which absent in data, but depend on other columns which also absent in data. Fixes #15530. #15532 (alesapin).
- Fix bug with event subscription in DDLWorker which rarely may lead to query hangs in ON CLUSTER. Introduced in #13450. #15477 (alesapin).
- Report proper error when the second argument of boundingRatio aggregate function has a wrong type. #15407 (detailyang).
- Fix bug where queries like SELECT toStartOfDay(today()) fail complaining about empty time\_zone argument. #15319 (Bharat Nallan).
- Fix race condition during MergeTree table rename and background cleanup. #15304 (alesapin).
- Fix rare race condition on server startup when system.logs are enabled. #15300 (alesapin).
- Fix MSan report in QueryLog. Uninitialized memory can be used for the field memory\_usage. #15258 (alexey-milovidov).
- Fix instance crash when using joinGet with LowCardinality types. This fixes #15214. #15220 (Amos Bird).
- Fix bug in table engine Buffer which does not allow to insert data of new structure into Buffer after ALTER query. Fixes #15117. #15192 (alesapin).
- Adjust decimals field size in mysql column definition packet. #15152 (maqroll).
- Fixed Cannot rename ... errno: 22, strerror: Invalid argumenterror on DDL query execution in Atomic database when running clickhouse-server in docker on Mac OS. #15024 (tavplubix).
- Fix to make predicate push down work when subquery contains finalizeAggregation function. Fixes #14847. #14937 (filimonov).
- Fix a problem where the server may get stuck on startup while talking to ZooKeeper, if the configuration files have to be fetched from ZK (using the from\_zk include option). This fixes #14814. #14843 (Alexander Kuzmenkov).

## Improvement

- Now it's possible to change the type of version column for VersionedCollapsingMergeTree with ALTER query. #15442 (alesapin).

## ClickHouse release v20.9.2.20, 2020-09-22

### Backward Incompatible Change

- When upgrading from versions older than 20.5, if rolling update is performed and cluster contains both versions 20.5 or greater and less than 20.5, if ClickHouse nodes with old versions are restarted and old version has been started up in presence of newer versions, it may lead to `Part ... intersects previous part` errors. To prevent this error, first install newer clickhouse-server packages on all cluster nodes and then do restarts (so, when clickhouse-server is restarted, it will start up with the new version).

## New Feature

- Added column transformers `EXCEPT`, `REPLACE`, `APPLY`, which can be applied to the list of selected columns (after `*` or `COLUMNS(...)`). For example, you can write `SELECT * EXCEPT(URL) REPLACE(number + 1 AS number)`. Another example: `select * apply(length) apply(max) from wide_string_table` to find out the maximum length of all string columns. [#14233 \(Amos Bird\)](#).
- Added an aggregate function `rankCorr` which computes a rank correlation coefficient. [#11769 \(antikvist\)](#) [#14411 \(Nikita Mikhaylov\)](#).
- Added table function `view` which turns a subquery into a table object. This helps passing queries around. For instance, it can be used in remote/cluster table functions. [#12567 \(Amos Bird\)](#).

## Bug Fix

- Fix bug when `ALTER UPDATE` mutation with `Nullable` column in assignment expression and constant value (like `UPDATE x = 42`) leads to incorrect value in column or segfault. Fixes [#13634](#), [#14045](#), [#14646 \(alesapin\)](#).
- Fix wrong Decimal multiplication result caused wrong decimal scale of result column. [#14603 \(Artem Zuikov\)](#).
- Fixed the incorrect sorting order of `Nullable` column. This fixes [#14344](#), [#14495 \(Nikita Mikhaylov\)](#).
- Fixed inconsistent comparison with primary key of type `FixedString` on index analysis if they're compared with a string of less size. This fixes [#14908](#), [#15033 \(Amos Bird\)](#).
- Fix bug which leads to wrong merges assignment if table has partitions with a single part. [#14444 \(alesapin\)](#).
- If function `bar` was called with specifically crafted arguments, buffer overflow was possible. This closes [#13926](#), [#15028 \(alexey-milovidov\)](#).
- Publish CPU frequencies per logical core in `system.asynchronous_metrics`. This fixes [#14923](#), [#14924 \(Alexander Kuzmenkov\)](#).
- Fixed `.metadata.tmp` File exists error when using `MaterializeMySQL` database engine. [#14898 \(Winter Zhang\)](#).
- Fix the issue when some invocations of `extractAllGroups` function may trigger "Memory limit exceeded" error. This fixes [#13383](#), [#14889 \(alexey-milovidov\)](#).
- Fix `SIGSEGV` for an attempt to `INSERT` into `StorageFile(fd)`. [#14887 \(Azat Khuzhin\)](#).
- Fix rare error in `SELECT` queries when the queried column has `DEFAULT` expression which depends on the other column which also has `DEFAULT` and not present in select query and not exists on disk. Partially fixes [#14531](#), [#14845 \(alesapin\)](#).
- Fix wrong monotonicity detection for shrunk `Int -> Int` cast of signed types. It might lead to incorrect query result. This bug is unveiled in [#14513](#), [#14783 \(Amos Bird\)](#).
- Fixed missed default database name in metadata of materialized view when executing `ALTER ... MODIFY QUERY`. [#14664 \(tavplubix\)](#).
- Fix possibly incorrect result of function `has` when LowCardinality and `Nullable` types are involved. [#14591 \(Mike\)](#).

- Cleanup data directory after Zookeeper exceptions during CREATE query for tables with ReplicatedMergeTree Engine. [#14563](#) (**Bharat Nallan**).
- Fix rare segfaults in functions with combinator `-Resample`, which could appear in result of overflow with very large parameters. [#14562](#) (**Anton Popov**).
- Check for array size overflow in `topK` aggregate function. Without this check the user may send a query with carefully crafted parameters that will lead to server crash. This closes [#14452](#). [#14467](#) (**alexey-milovidov**).
- Proxy restart/start/stop/reload of SysVinit to systemd (if it is used). [#14460](#) (**Azat Khuzhin**).
- Stop query execution if exception happened in `PipelineExecutor` itself. This could prevent rare possible query hung. [#14334](#) [#14402](#) (**Nikolai Kochetov**).
- Fix crash during `ALTER` query for table which was created `AS table_function`. Fixes [#14212](#). [#14326](#) (**alesapin**).
- Fix exception during `ALTER LIVE VIEW` query with `REFRESH` command. `LIVE VIEW` is an experimental feature. [#14320](#) (**Bharat Nallan**).
- Fix `QueryPlan` lifetime (for `EXPLAIN PIPELINE graph=1`) for queries with nested interpreter. [#14315](#) (**Azat Khuzhin**).
- Better check for tuple size in SSD cache complex key external dictionaries. This fixes [#13981](#). [#14313](#) (**alexey-milovidov**).
- Disallows `CODEC` on `ALIAS` column type. Fixes [#13911](#). [#14263](#) (**Bharat Nallan**).
- Fix `GRANT ALL` statement when executed on a non-global level. [#13987](#) (**Vitaly Baranov**).
- Fix `arrayJoin()` capturing in lambda (exception with logical error message was thrown). [#13792](#) (**Azat Khuzhin**).

## Experimental Feature

- Added `db-generator` tool for random database generation by given `SELECT` queries. It may facilitate reproducing issues when there is only incomplete bug report from the user. [#14442](#) (**Nikita Mikhaylov**) [#10973](#) (**ZeDRoman**).

## Improvement

- Allow using multi-volume storage configuration in storage Distributed. [#14839](#) (**Pavel Kovalenko**).
- Disallow empty `time_zone` argument in `toStartOf*` type of functions. [#14509](#) (**Bharat Nallan**).
- MySQL handler returns `OK` for queries like `SET @@var = value`. Such statement is ignored. It is needed because some MySQL drivers send `SET @@` query for setup after handshake  
<https://github.com/ClickHouse/ClickHouse/issues/9336#issuecomment-686222422> . [#14469](#) (**BohuTANG**).
- Now TTLs will be applied during merge if they were not previously materialized. [#14438](#) (**alesapin**).
- Now `clickhouse-obfuscator` supports `UUID` type as proposed in [#13163](#). [#14409](#) (**dimarub2000**).
- Added new setting `system_events_show_zero_values` as proposed in [#11384](#). [#14404](#) (**dimarub2000**).
- Implicitly convert primary key to not null in `MaterializeMySQL` (Same as MySQL). Fixes [#14114](#). [#14397](#) (**Winter Zhang**).
- Replace wide integers (256 bit) from boost multiprecision with implementation from <https://github.com/cerevra/int>. 256bit integers are experimental. [#14229](#) (**Artem Zuikov**).

- Add default compression codec for parts in `system.part_log` with the name `default_compression_codec`. #14116 (alesapin).
- Add precision argument for `DateTime` type. It allows to use `DateTime` name instead of `DateTime64`. #13761 (Winter Zhang).
- Added requirepass authorization for `Redis` external dictionary. #13688 (Ivan Torgashov).
- Improvements in RabbitMQ engine: added connection and channels failure handling, proper commits, insert failures handling, better exchanges, queue durability and queue resume opportunity, new queue settings. Fixed tests. #12761 (Kseniia Sumarokova).
- Support custom codecs in compact parts. #12183 (Anton Popov).

## Performance Improvement

- Optimize queries with `LIMIT/LIMIT BY/ORDER BY` for distributed with `GROUP BY sharding_key` (under `optimize_skip_unused_shards` and `optimize_distributed_group_by_sharding_key`). #10373 (Azat Khuzhin).
- Creating sets for multiple `JOIN` and `IN` in parallel. It may slightly improve performance for queries with several different `IN` subquery expressions. #14412 (Nikolai Kochetov).
- Improve Kafka engine performance by providing independent thread for each consumer. Separate thread pool for streaming engines (like Kafka). #13939 (fastio).

## Build/Testing/Packaging Improvement

- Lower binary size in debug build by removing debug info from `Functions`. This is needed only for one internal project in Yandex who is using very old linker. #14549 (alexey-milovidov).
- Prepare for build with clang 11. #14455 (alexey-milovidov).
- Fix the logic in backport script. In previous versions it was triggered for any labels of 100% red color. It was strange. #14433 (alexey-milovidov).
- Integration tests use default base config. All config changes are explicit with `main_configs`, `user_configs` and `dictionaries` parameters for instance. #13647 (Ilya Yatsishin).

# ClickHouse release 20.8

## ClickHouse release v20.8.12.2-lts, 2021-01-16

### Bug Fix

- Fix \*If combinator with unary function and Nullable types. #18806 (Azat Khuzhin).
- Restrict merges from wide to compact parts. In case of vertical merge it led to broken result part. #18381 (Anton Popov).

## ClickHouse release v20.8.11.17-lts, 2020-12-25

### Bug Fix

- Disable write with AIO during merges because it can lead to extremely rare data corruption of primary key columns during merge. #18481 (alesapin).
- Fixed `value is too short` error when executing `toType(...)` functions (`toDate`, `toUInt32`, etc) with argument of type `Nullable(String)`. Now such functions return `NULL` on parsing errors instead of throwing exception. Fixes #7673. #18445 (tavplubix).
- Fix possible crashes in aggregate functions with combinator `Distinct`, while using two-level aggregation. Fixes #17682. #18365 (Anton Popov).

# ClickHouse release v20.8.10.13-lts, 2020-12-24

## Bug Fix

- When server log rotation was configured using `logger.size` parameter with numeric value larger than  $2^{32}$ , the logs were not rotated properly. [#17905](#) ([Alexander Kuzmenkov](#)).
- Fixed incorrect initialization of `max_compress_block_size` in `MergeTreeWriterSettings` with `min_compress_block_size`. [#17833](#) ([flynn](#)).
- Fixed problem when ClickHouse fails to resume connection to MySQL servers. [#17681](#) ([Alexander Kazakov](#)).
- Fixed ALTER query hang when the corresponding mutation was killed on the different replica. This fixes [#16953](#). [#17499](#) ([alesapin](#)).
- Fixed a bug when mark cache size was underestimated by ClickHouse. It may happen when there are a lot of tiny files with marks. [#17496](#) ([alesapin](#)).
- Fixed ORDER BY with enabled setting `optimize_redundant_functions_in_order_by`. [#17471](#) ([Anton Popov](#)).
- Fixed `ColumnConst` comparison which leads to crash. This fixed [#17088](#) . [#17135](#) ([Amos Bird](#)).
- Fixed bug when `ON CLUSTER` queries may hang forever for non-leader ReplicatedMergeTreeTables. [#17089](#) ([alesapin](#)).
- Avoid unnecessary network errors for remote queries which may be cancelled while execution, like queries with `LIMIT`. [#17006](#) ([Azat Khuzhin](#)).
- Reresolve the IP of the `format_avro_schema_registry_url` in case of errors. [#16985](#) ([filimonov](#)).
- Fixed possible server crash after `ALTER TABLE ... MODIFY COLUMN ... NewType` when `SELECT` have `WHERE` expression on altering column and alter does not finished yet. [#16968](#) ([Amos Bird](#)).
- Install script should always create subdirs in config folders. This is only relevant for Docker build with custom config. [#16936](#) ([filimonov](#)).
- Fixed possible error `Illegal type of argument` for queries with ORDER BY. Fixes [#16580](#). [#16928](#) ([Nikolai Kochetov](#)).
- Abort multipart upload if no data was written to `WriteBufferFromS3`. [#16840](#) ([Pavel Kovalenko](#)).
- Fixed crash when using `any` without any arguments. This fixes [#16803](#). [#16826](#) ([Amos Bird](#)).
- Fixed `IN` operator over several columns and tuples with enabled `transform_null_in` setting. Fixes [#15310](#). [#16722](#) ([Anton Popov](#)).
- Fixed inconsistent behaviour of `optimize_read_in_order/optimize_aggregation_in_order` with `max_threads > 0` and expression in ORDER BY. [#16637](#) ([Azat Khuzhin](#)).
- Fixed the issue when query optimization was producing wrong result if query contains `ARRAY JOIN`. [#17887](#) ([sundyli](#)).
- Query is finished faster in case of exception. Cancel execution on remote replicas if exception happens. [#15578](#) ([Azat Khuzhin](#)).

# ClickHouse release v20.8.6.6-lts, 2020-11-13

## Bug Fix

- Fix rare silent crashes when query profiler is on and ClickHouse is installed on OS with glibc version that has (supposedly) broken asynchronous unwind tables for some functions. This fixes #15301. This fixes #13098. #16846 ([alexey-milovidov](#)).
- Now when parsing AVRO from input the LowCardinality is removed from type. Fixes #16188. #16521 ([Mike](#)).
- Fix rapid growth of metadata when using MySQL Master -> MySQL Slave -> ClickHouse MaterializeMySQL Engine, and `slave_parallel_worker` enabled on MySQL Slave, by properly shrinking GTID sets. This fixes #15951. #16504 ([TCeason](#)).
- Fix DROP TABLE for Distributed (racy with INSERT). #16409 ([Azat Khuzhin](#)).
- Fix processing of very large entries in replication queue. Very large entries may appear in ALTER queries if table structure is extremely large (near 1 MB). This fixes #16307. #16332 ([alexey-milovidov](#)).
- Fixed the inconsistent behaviour when a part of return data could be dropped because the set for its filtration wasn't created. #16308 ([Nikita Mikhaylov](#)).
- Fix bug with MySQL database. When MySQL server used as database engine is down some queries raise Exception, because they try to get tables from disabled server, while it's unnecessary. For example, query `SELECT ... FROM system.parts` should work only with MergeTree tables and don't touch MySQL database at all. #16032 ([Kruglov Pavel](#)).

## ClickHouse release v20.8.5.45-lts, 2020-10-29

### Bug Fix

- Fix double free in case of exception in function `dictGet`. It could have happened if dictionary was loaded with error. #16429 ([Nikolai Kochetov](#)).
- Fix group by with totals/rollup/cube modifiers and min/max functions over group by keys. Fixes #16393. #16397 ([Anton Popov](#)).
- Fix async Distributed INSERT w/ `prefer_localhost_replica=0` and `internal_replication`. #16358 ([Azat Khuzhin](#)).
- Fix a possible memory leak during GROUP BY with string keys, caused by an error in `TwoLevelStringHashTable` implementation. #16264 ([Amos Bird](#)).
- Fix the case when memory can be overallocated regardless to the limit. This closes #14560. #16206 ([alexey-milovidov](#)).
- Fix `ALTER MODIFY ... ORDER BY` query hang for `ReplicatedVersionedCollapsingMergeTree`. This fixes #15980. #16011 ([alesapin](#)).
- Fix collate name & charset name parser and support `length = 0` for string type. #16008 ([Winter Zhang](#)).
- Allow to use direct layout for dictionaries with complex keys. #16007 ([Anton Popov](#)).
- Prevent replica hang for 5-10 mins when replication error happens after a period of inactivity. #15987 ([filimonov](#)).
- Fix rare segfaults when inserting into or selecting from `MaterializedView` and concurrently dropping target table (for Atomic database engine). #15984 ([tavplubix](#)).
- Fix ambiguity in parsing of settings profiles: `CREATE USER ... SETTINGS profile readonly` is now considered as using a profile named `readonly`, not a setting named `profile` with the `readonly` constraint. This fixes #15628. #15982 ([Vitaly Baranov](#)).
- Fix a crash when database creation fails. #15954 ([Winter Zhang](#)).

- Fixed `DROP TABLE IF EXISTS` failure with `Table ... does not exist` error when table is concurrently renamed (for Atomic database engine). Fixed rare deadlock when concurrently executing some DDL queries with multiple tables (like `DROP DATABASE` and `RENAME TABLE`) Fixed `DROP/DETACH DATABASE` failure with `Table ... does not exist` when concurrently executing `DROP/DETACH TABLE`. #15934 (tavplubix).
- Fix incorrect empty result for query from Distributed table if query has `WHERE`, `PREWHERE` and `GLOBAL IN`. Fixes #15792. #15933 (Nikolai Kochetov).
- Fix possible deadlocks in RBAC. #15875 (Vitaly Baranov).
- Fix exception `Block structure mismatch` in `SELECT ... ORDER BY DESC` queries which were executed after `ALTER MODIFY COLUMN` query. Fixes #15800. #15852 (alesapin).
- Fix some cases of queries, in which only virtual columns are selected. Previously `Not found column _nothing in block` exception may be thrown. Fixes #12298. #15756 (Anton Popov).
- Fix error `Cannot find column` which may happen at insertion into `MATERIALIZED VIEW` in case if query for `MV` contains `ARRAY JOIN`. #15717 (Nikolai Kochetov).
- Fixed too low default value of `max_replicated_logs_to_keep` setting, which might cause replicas to become lost too often. Improve lost replica recovery process by choosing the most up-to-date replica to clone. Also do not remove old parts from lost replica, detach them instead. #15701 (tavplubix).
- Fix error `Cannot add simple transform to empty Pipe` which happened while reading from `Buffer` table which has different structure than destination table. It was possible if destination table returned empty result for query. Fixes #15529. #15662 (Nikolai Kochetov).
- Fixed bug with globs in S3 table function, region from URL was not applied to S3 client configuration. #15646 (Vladimir Chebotarev).
- Decrement the `ReadOnlyReplica` metric when detaching read-only tables. This fixes #15598. #15592 (sundlyi).
- Throw an error when a single parameter is passed to `ReplicatedMergeTree` instead of ignoring it. #15516 (nvartolomei).

## Improvement

- Now it's allowed to execute `ALTER ... ON CLUSTER` queries regardless of the `<internal_replication>` setting in cluster config. #16075 (alesapin).
- Unfold `{database}`, `{table}` and `{uuid}` macros in `ReplicatedMergeTree` arguments on table creation. #16159 (tavplubix).

## ClickHouse release v20.8.4.11-lts, 2020-10-09

### Bug Fix

- Fix the order of destruction for resources in `ReadFromStorage` step of query plan. It might cause crashes in rare cases. Possibly connected with #15610. #15645 (Nikolai Kochetov).
- Fixed `Element ... is not a constant expression` error when using `JSON*` function result in `VALUES`, `LIMIT` or right side of `IN` operator. #15589 (tavplubix).
- Prevent the possibility of error message `Could not calculate available disk space (statvfs), errno: 4, strerror: Interrupted system call`. This fixes #15541. #15557 (alexey-milovidov).
- Significantly reduce memory usage in `AggregatingInOrderTransform/optimize_aggregation_in_order`. #15543 (Azat Khuzhin).

- Mutation might hang waiting for some non-existent part after `MOVE` or `REPLACE PARTITION` or, in rare cases, after `DETACH` or `DROP PARTITION`. It's fixed. [#15537 \(tavplubix\)](#).
- Fix bug when `ILIKE` operator stops being case insensitive if `LIKE` with the same pattern was executed. [#15536 \(alesapin\)](#).
- Fix `Missing columns` errors when selecting columns which absent in data, but depend on other columns which also absent in data. Fixes [#15530](#). [#15532 \(alesapin\)](#).
- Fix bug with event subscription in `DDLWorker` which rarely may lead to query hangs in `ON CLUSTER`. Introduced in [#13450](#). [#15477 \(alesapin\)](#).
- Report proper error when the second argument of `boundingRatio` aggregate function has a wrong type. [#15407 \(detailyang\)](#).
- Fix race condition during `MergeTree` table rename and background cleanup. [#15304 \(alesapin\)](#).
- Fix rare race condition on server startup when `system.logs` are enabled. [#15300 \(alesapin\)](#).
- Fix MSan report in `QueryLog`. Uninitialized memory can be used for the field `memory_usage`. [#15258 \(alexey-milovidov\)](#).
- Fix instance crash when using `joinGet` with `LowCardinality` types. This fixes [#15214](#). [#15220 \(Amos Bird\)](#).
- Fix bug in table engine `Buffer` which does not allow to insert data of new structure into `Buffer` after `ALTER` query. Fixes [#15117](#). [#15192 \(alesapin\)](#).
- Adjust decimals field size in `mysql` column definition packet. [#15152 \(maqroll\)](#).
- We already use padded comparison between `String` and `FixedString` (<https://github.com/ClickHouse/ClickHouse/blob/master/src/Functions/FunctionsComparison.h#L333>). This PR applies the same logic to field comparison which corrects the usage of `FixedString` as primary keys. This fixes [#14908](#). [#15033 \(Amos Bird\)](#).
- If function `bar` was called with specifically crafted arguments, buffer overflow was possible. This closes [#13926](#). [#15028 \(alexey-milovidov\)](#).
- Fixed Cannot rename ... `errno: 22, strerror: Invalid argument` error on DDL query execution in Atomic database when running `clickhouse-server` in docker on Mac OS. [#15024 \(tavplubix\)](#).
- Now settings `number_of_free_entries_in_pool_to_execute_mutation` and `number_of_free_entries_in_pool_to_lower_max_size_of_merge` can be equal to `background_pool_size`. [#14975 \(alesapin\)](#).
- Fix to make predicate push down work when subquery contains `finalizeAggregation` function. Fixes [#14847](#). [#14937 \(filimonov\)](#).
- Publish CPU frequencies per logical core in `system.asynchronous_metrics`. This fixes [#14923](#). [#14924 \(Alexander Kuzmenkov\)](#).
- Fixed `.metadata.tmp` File exists error when using `MaterializeMySQL` database engine. [#14898 \(Winter Zhang\)](#).
- Fix a problem where the server may get stuck on startup while talking to ZooKeeper, if the configuration files have to be fetched from ZK (using the `from_zk` include option). This fixes [#14814](#). [#14843 \(Alexander Kuzmenkov\)](#).
- Fix wrong monotonicity detection for shrunk `Int -> Int` cast of signed types. It might lead to incorrect query result. This bug is unveiled in [#14513](#). [#14783 \(Amos Bird\)](#).
- Fixed the incorrect sorting order of `Nullable` column. This fixes [#14344](#). [#14495 \(Nikita Mikhaylov\)](#).

## Improvement

- Now it's possible to change the type of version column for `VersionedCollapsingMergeTree` with `ALTER` query. #15442 (alesapin).

## ClickHouse release v20.8.3.18-stable, 2020-09-18

### Bug Fix

- Fix the issue when some invocations of `extractAllGroups` function may trigger "Memory limit exceeded" error. This fixes #13383. #14889 (alexey-milovidov).
- Fix SIGSEGV for an attempt to `INSERT` into `StorageFile(fd)`. #14887 (Azat Khuzhin).
- Fix rare error in `SELECT` queries when the queried column has `DEFAULT` expression which depends on the other column which also has `DEFAULT` and not present in select query and not exists on disk. Partially fixes #14531. #14845 (alesapin).
- Fixed missed default database name in metadata of materialized view when executing `ALTER ... MODIFY QUERY`. #14664 (tavplubix).
- Fix bug when `ALTER UPDATE` mutation with Nullable column in assignment expression and constant value (like `UPDATE x = 42`) leads to incorrect value in column or segfault. Fixes #13634, #14045. #14646 (alesapin).
- Fix wrong Decimal multiplication result caused wrong decimal scale of result column. #14603 (Artem Zuikov).
- Added the checker as neither calling `lc->isNullable()` nor calling `ls->getDictionaryPtr()->isNullable()` would return the correct result. #14591 (myrrc).
- Cleanup data directory after Zookeeper exceptions during `CreateQuery` for `StorageReplicatedMergeTree` Engine. #14563 (Bharat Nallan).
- Fix rare segfaults in functions with combinator -Resample, which could appear in result of overflow with very large parameters. #14562 (Anton Popov).

## Improvement

- Speed up server shutdown process if there are ongoing S3 requests. #14858 (Pavel Kovalenko).
- Allow using multi-volume storage configuration in storage Distributed. #14839 (Pavel Kovalenko).
- Speed up server shutdown process if there are ongoing S3 requests. #14496 (Pavel Kovalenko).
- Support custom codecs in compact parts. #12183 (Anton Popov).

## ClickHouse release v20.8.2.3-stable, 2020-09-08

### Backward Incompatible Change

- Now `OPTIMIZE FINAL` query does not recalculate TTL for parts that were added before TTL was created. Use `ALTER TABLE ... MATERIALIZE TTL` once to calculate them, after that `OPTIMIZE FINAL` will evaluate TTL's properly. This behavior never worked for replicated tables. #14220 (alesapin).
- Extend `parallel_distributed_insert_select` setting, adding an option to run `INSERT` into local table. The setting changes type from `Bool` to `UInt64`, so the values `false` and `true` are no longer supported. If you have these values in server configuration, the server will not start. Please replace them with `0` and `1`, respectively. #14060 (Azat Khuzhin).

- Remove support for the `ODBCDriver` input/output format. This was a deprecated format once used for communication with the ClickHouse ODBC driver, now long superseded by the `ODBCDriver2` format. Resolves #13629. #13847 (hexiaoting).
- When upgrading from versions older than 20.5, if rolling update is performed and cluster contains both versions 20.5 or greater and less than 20.5, if ClickHouse nodes with old versions are restarted and old version has been started up in presence of newer versions, it may lead to `Part ... intersects previous part` errors. To prevent this error, first install newer `clickhouse-server` packages on all cluster nodes and then do restarts (so, when `clickhouse-server` is restarted, it will start up with the new version).

## New Feature

- Add the ability to specify `Default` compression codec for columns that correspond to settings specified in `config.xml`. Implements: #9074. #14049 (alesapin).
- Support Kerberos authentication in Kafka, using `krb5` and `cyrus-sasl` libraries. #12771 (Ilya Golshtain).
- Add function `normalizeQuery` that replaces literals, sequences of literals and complex aliases with placeholders. Add function `normalizedQueryHash` that returns identical 64bit hash values for similar queries. It helps to analyze query log. This closes #11271. #13816 (alexey-milovidov).
- Add `time_zones` table. #13880 (Bharat Nallan).
- Add function `defaultValueOfTypeName` that returns the default value for a given type. #13877 (hcz).
- Add `countDigits(x)` function that count number of decimal digits in integer or decimal column. Add `isDecimalOverflow(d, [p])` function that checks if the value in Decimal column is out of its (or specified) precision. #14151 (Artem Zuikov).
- Add `quantileExactLow` and `quantileExactHigh` implementations with respective aliases for `medianExactLow` and `medianExactHigh`. #13818 (Bharat Nallan).
- Added `date_trunc` function that truncates a date/time value to a specified date/time part. #13888 (Vladimir Golovchenko).
- Add new optional section `<user_directories>` to the main config. #13425 (Vitaly Baranov).
- Add `ALTER SAMPLE BY` statement that allows to change table sample clause. #13280 (Amos Bird).
- Function `position` now supports optional `start_pos` argument. #13237 (vdimir).

## Bug Fix

- Fix visible data clobbering by progress bar in client in interactive mode. This fixes #12562 and #13369 and #13584 and fixes #12964. #13691 (alexey-milovidov).
- Fixed incorrect sorting order if `LowCardinality` column when sorting by multiple columns. This fixes #13958. #14223 (Nikita Mikhaylov).
- Check for array size overflow in `topK` aggregate function. Without this check the user may send a query with carefully crafted parameters that will lead to server crash. This closes #14452. #14467 (alexey-milovidov).
- Fix bug which can lead to wrong merges assignment if table has partitions with a single part. #14444 (alesapin).
- Stop query execution if exception happened in `PipelineExecutor` itself. This could prevent rare possible query hung. Continuation of #14334. #14402 #14334 (Nikolai Kochetov).
- Fix crash during `ALTER` query for table which was created `AS table_function`. Fixes #14212. #14326 (alesapin).

- Fix exception during ALTER LIVE VIEW query with REFRESH command. Live view is an experimental feature. #14320 (Bharat Nallan).
- Fix QueryPlan lifetime (for EXPLAIN PIPELINE graph=1) for queries with nested interpreter. #14315 (Azat Khuzhin).
- Fix segfault in `clickhouse-odbc-bridge` during schema fetch from some external sources. This PR fixes #13861. #14267 (Vitaly Baranov).
- Fix crash in mark inclusion search introduced in #12277. #14225 (Amos Bird).
- Fix creation of tables with named tuples. This fixes #13027. #14143 (alexey-milovidov).
- Fix formatting of minimal negative decimal numbers. This fixes #14111. #14119 (Alexander Kuzmenkov).
- Fix `DistributedFilesToInsert` metric (zeroed when it should not). #14095 (Azat Khuzhin).
- Fix `pointInPolygon` with const 2d array as polygon. #14079 (Alexey Ilyukhov).
- Fixed wrong mount point in extra info for `Poco::Exception`: no space left on device #14050 (tavplubix).
- Fix GRANT ALL statement when executed on a non-global level. #13987 (Vitaly Baranov).
- Fix parser to reject create table as table function with engine. #13940 (hcz).
- Fix wrong results in select queries with `DISTINCT` keyword and subqueries with UNION ALL in case `optimize_duplicate_order_by_and_distinct` setting is enabled. #13925 (Artem Zuikov).
- Fixed potential deadlock when renaming `Distributed` table. #13922 (tavplubix).
- Fix incorrect sorting for `FixedString` columns when sorting by multiple columns. Fixes #13182. #13887 (Nikolai Kochetov).
- Fix potentially imprecise result of `topK`/`topKWeighted` merge (with non-default parameters). #13817 (Azat Khuzhin).
- Fix reading from MergeTree table with INDEX of type SET fails when comparing against NULL. This fixes #13686. #13793 (Amos Bird).
- Fix `arrayJoin` capturing in lambda (LOGICAL\_ERROR). #13792 (Azat Khuzhin).
- Add step overflow check in function `range`. #13790 (Azat Khuzhin).
- Fixed `Directory not empty` error when concurrently executing `DROP DATABASE` and `CREATE TABLE`. #13756 (alexey-milovidov).
- Add range check for `h3KRing` function. This fixes #13633. #13752 (alexey-milovidov).
- Fix race condition between DETACH and background merges. Parts may revive after detach. This is continuation of #8602 that did not fix the issue but introduced a test that started to fail in very rare cases, demonstrating the issue. #13746 (alexey-milovidov).
- Fix logging `Settings.Names/Values` when `log_queries_min_type > QUERY_START`. #13737 (Azat Khuzhin).
- Fixes `/replicas_status` endpoint response status code when `verbose=1`. #13722 (javi santana).
- Fix incorrect message in `clickhouse-server.init` while checking user and group. #13711 (ylchou).
- Do not optimize any(`arrayJoin()`) -> `arrayJoin()` under `optimize_move_functions_out_of_any` setting. #13681 (Azat Khuzhin).

- Fix crash in JOIN with StorageMerge and `set enable_optimize_predicate_expression=1`. #13679 (Artem Zuikov).
- Fix typo in error message about `The value of 'number_of_free_entries_in_pool_to_lower_max_size_of_merge'` setting. #13678 (alexey-milovidov).
- Concurrent `ALTER ... REPLACE/MOVE PARTITION ...` queries might cause deadlock. It's fixed. #13626 (tavplubix).
- Fixed the behaviour when sometimes cache-dictionary returned default value instead of present value from source. #13624 (Nikita Mikhaylov).
- Fix secondary indices corruption in compact parts. Compact parts are experimental feature. #13538 (Anton Popov).
- Fix premature `ON CLUSTER` timeouts for queries that must be executed on a single replica. Fixes #6704, #7228, #13361, #11884. #13450 (alesapin).
- Fix wrong code in function `netloc`. This fixes #13335. #13446 (alexey-milovidov).
- Fix possible race in `StorageMemory`. #13416 (Nikolai Kochetov).
- Fix missing or excessive headers in `TSV/CSVWithNames` formats in HTTP protocol. This fixes #12504. #13343 (Azat Khuzhin).
- Fix parsing row policies from `users.xml` when names of databases or tables contain dots. This fixes #5779, #12527. #13199 (Vitaly Baranov).
- Fix access to `redis` dictionary after connection was dropped once. It may happen with `cache` and `direct` dictionary layouts. #13082 (Anton Popov).
- Removed wrong auth access check when using `ClickHouseDictionarySource` to query remote tables. #12756 (sundyl).
- Properly distinguish subqueries in some cases for common subexpression elimination. #8333. #8367 (Amos Bird).

## Improvement

- Disallows `CODEC` on `ALIAS` column type. Fixes #13911. #14263 (Bharat Nallan).
- When waiting for a dictionary update to complete, use the timeout specified by `query_wait_timeout_milliseconds` setting instead of a hard-coded value. #14105 (Nikita Mikhaylov).
- Add setting `min_index_granularity_bytes` that protects against accidentally creating a table with very low `index_granularity_bytes` setting. #14139 (Bharat Nallan).
- Now it's possible to fetch partitions from clusters that use different ZooKeeper: `ALTER TABLE table_name FETCH PARTITION partition_expr FROM 'zk-name:/path-in-zookeeper'`. It's useful for shipping data to new clusters. #14155 (Amos Bird).
- Slightly better performance of Memory table if it was constructed from a huge number of very small blocks (that's unlikely). Author of the idea: [Mark Papadakis](#). Closes #14043. #14056 (alexey-milovidov).
- Conditional aggregate functions (for example: `avgIf`, `sumIf`, `maxIf`) should return `NULL` when miss rows and use nullable arguments. #13964 (Winter Zhang).
- Increase limit in -Resample combinator to 1M. #13947 (Mikhail f. Shiryaev).
- Corrected an error in AvroConfluent format that caused the Kafka table engine to stop processing messages when an abnormally small, malformed, message was received. #13941 (Gervasio Varela).

- Fix wrong error for long queries. It was possible to get syntax error other than Max query size exceeded for correct query. #13928 (Nikolai Kochetov).
- Better error message for null value of TabSeparated format. #13906 (jiang tao).
- Function arrayCompact will compare NaNs bitwise if the type of array elements is Float32/Float64. In previous versions NaNs were always not equal if the type of array elements is Float32/Float64 and were always equal if the type is more complex, like Nullable(Float64). This closes #13857. #13868 (alexey-milovidov).
- Fix data race in Igamma function. This race was caught only in tsan, no side effects a really happened. #13842 (Nikolai Kochetov).
- Avoid too slow queries when arrays are manipulated as fields. Throw exception instead. #13753 (alexey-milovidov).
- Added Redis requirepass authorization (for redis dictionary source). #13688 (Ivan Torgashov).
- Add MergeTree Write-Ahead-Log (WAL) dump tool. WAL is an experimental feature. #13640 (BohuTANG).
- In previous versions lcm function may produce assertion violation in debug build if called with specifically crafted arguments. This fixes #13368. #13510 (alexey-milovidov).
- Provide monotonicity for toDate/toDateTime functions in more cases. Monotonicity information is used for index analysis (more complex queries will be able to use index). Now the input arguments are saturated more naturally and provides better monotonicity. #13497 (Amos Bird).
- Support compound identifiers for custom settings. Custom settings is an integration point of ClickHouse codebase with other codebases (no benefits for ClickHouse itself) #13496 (Vitaly Baranov).
- Move parts from DiskLocal to DiskS3 in parallel. DiskS3 is an experimental feature. #13459 (Pavel Kovalenko).
- Enable mixed granularity parts by default. #13449 (alesapin).
- Proper remote host checking in S3 redirects (security-related thing). #13404 (Vladimir Chebotarev).
- Add QueryTimeMicroseconds, SelectQueryTimeMicroseconds and InsertQueryTimeMicroseconds to system.events. #13336 (ianton-ru).
- Fix debug assertion when Decimal has too large negative exponent. Fixes #13188. #13228 (alexey-milovidov).
- Added cache layer for DiskS3 (cache to local disk mark and index files). DiskS3 is an experimental feature. #13076 (Pavel Kovalenko).
- Fix readline so it dumps history to file now. #13600 (Amos Bird).
- Create system database with Atomic engine by default (a preparation to enable Atomic database engine by default everywhere). #13680 (tavplubix).

## Performance Improvement

- Slightly optimize very short queries with LowCardinality. #14129 (Anton Popov).
- Enable parallel INSERTs for table engines Null, Memory, Distributed and Buffer when the setting max\_insert\_threads is set. #14120 (alexey-milovidov).

- Fail fast if `max_rows_to_read` limit is exceeded on parts scan. The motivation behind this change is to skip ranges scan for all selected parts if it is clear that `max_rows_to_read` is already exceeded. The change is quite noticeable for queries over big number of parts. [#13677](#) ([Roman Khavronenko](#)).
- Slightly improve performance of aggregation by `UInt8/UInt16` keys. [#13099](#) ([alexey-milovidov](#)).
- Optimize `has()`, `indexOf()` and `countEqual()` functions for `Array(LowCardinality(T))` and constant right arguments. [#12550](#) ([myrrc](#)).
- When performing trivial `INSERT SELECT` queries, automatically set `max_threads` to 1 or `max_insert_threads`, and set `max_block_size` to `min_insert_block_size_rows`. Related to [#5907](#). [#12195](#) ([flynn](#)).

## Experimental Feature

- ClickHouse can work as MySQL replica - it is implemented by `MaterializeMySQL` database engine. Implements [#4006](#). [#10851](#) ([Winter Zhang](#)).
- Add types `Int128`, `Int256`, `UInt256` and related functions for them. Extend Decimals with `Decimal256` (precision up to 76 digits). New types are under the setting `allow_experimental_bigint_types`. It is working extremely slow and bad. The implementation is incomplete. Please don't use this feature. [#13097](#) ([Artem Zuikov](#)).

## Build/Testing/Packaging Improvement

- Added `clickhouse install` script, that is useful if you only have a single binary. [#13528](#) ([alexey-milovidov](#)).
- Allow to run `clickhouse` binary without configuration. [#13515](#) ([alexey-milovidov](#)).
- Enable check for typos in code with `codespell`. [#13513](#) [#13511](#) ([alexey-milovidov](#)).
- Enable Shellcheck in CI as a linter of .sh tests. This closes [#13168](#). [#13530](#) [#13529](#) ([alexey-milovidov](#)).
- Add a CMake option to fail configuration instead of auto-reconfiguration, enabled by default. [#13687](#) ([Konstantin](#)).
- Expose version of embedded tzdata via `TZDATA_VERSION` in `system.build_options`. [#13648](#) ([filimonov](#)).
- Improve generation of `system.time_zones` table during build. Closes [#14209](#). [#14215](#) ([filimonov](#)).
- Build ClickHouse with the most fresh tzdata from package repository. [#13623](#) ([alexey-milovidov](#)).
- Add the ability to write js-style comments in `skip_list.json`. [#14159](#) ([alesapin](#)).
- Ensure that there is no copy-pasted GPL code. [#13514](#) ([alexey-milovidov](#)).
- Switch tests docker images to use test-base parent. [#14167](#) ([Ilya Yatsishin](#)).
- Adding retry logic when bringing up docker-compose cluster; Increasing `COMPOSE_HTTP_TIMEOUT`. [#14112](#) ([vzakaznikov](#)).
- Enabled `system.text_log` in stress test to find more bugs. [#13855](#) ([Nikita Mikhaylov](#)).
- Testflows LDAP module: adding missing certificates and `dhpam.pem` for `openldap4`. [#13780](#) ([vzakaznikov](#)).
- ZooKeeper cannot work reliably in unit tests in CI infrastructure. Using unit tests for ZooKeeper interaction with real ZooKeeper is bad idea from the start (unit tests are not supposed to verify complex distributed systems). We already using integration tests for this purpose and they are better suited. [#13745](#) ([alexey-milovidov](#)).
- Added docker image for style check. Added style check that all docker and docker compose files are located in docker directory. [#13724](#) ([Ilya Yatsishin](#)).

- Fix cassandra build on Mac OS. [#13708](#) ([Ilya Yatsishin](#)).
- Fix link error in shared build. [#13700](#) ([Amos Bird](#)).
- Updating LDAP user authentication suite to check that it works with RBAC. [#13656](#) ([vzakaznikov](#)).
- Removed `-DENABLE_CURL_CLIENT` for `contrib/aws`. [#13628](#) ([Vladimir Chebotarev](#)).
- Increasing health-check timeouts for ClickHouse nodes and adding support to dump docker-compose logs if unhealthy containers found. [#13612](#) ([vzakaznikov](#)).
- Make sure [#10977](#) is invalid. [#13539](#) ([Amos Bird](#)).
- Skip PR's from robot-clickhouse. [#13489](#) ([Nikita Mikhaylov](#)).
- Move Dockerfiles from integration tests to `docker/test` directory. `docker_compose` files are available in `runner` docker container. Docker images are built in CI and not in integration tests. [#13448](#) ([Ilya Yatsishin](#)).

## ClickHouse release 20.7

### ClickHouse release v20.7.2.30-stable, 2020-08-31

#### Backward Incompatible Change

- Function `modulo` (operator `%`) with at least one floating point number as argument will calculate remainder of division directly on floating point numbers without converting both arguments to integers. It makes behaviour compatible with most of DBMS. This also applicable for Date and DateTime data types. Added alias `mod`. This closes [#7323](#). [#12585](#) ([alexey-milovidov](#)).
- Deprecate special printing of zero Date/DateTime values as `0000-00-00` and `0000-00-00 00:00:00`. [#12442](#) ([alexey-milovidov](#)).
- The function `groupArrayMoving*` was not working for distributed queries. Its result was calculated within incorrect data type (without promotion to the largest type). The function `groupArrayMovingAvg` was returning integer number that was inconsistent with the `avg` function. This fixes [#12568](#). [#12622](#) ([alexey-milovidov](#)).
- Add sanity check for MergeTree settings. If the settings are incorrect, the server will refuse to start or to create a table, printing detailed explanation to the user. [#13153](#) ([alexey-milovidov](#)).
- Protect from the cases when user may set `background_pool_size` to value lower than `number_of_free_entries_in_pool_to_execute_mutation` or `number_of_free_entries_in_pool_to_lower_max_size_of_merge`. In these cases ALTERs won't work or the maximum size of merge will be too limited. It will throw exception explaining what to do. This closes [#10897](#). [#12728](#) ([alexey-milovidov](#)).
- When upgrading from versions older than 20.5, if rolling update is performed and cluster contains both versions 20.5 or greater and less than 20.5, if ClickHouse nodes with old versions are restarted and old version has been started up in presence of newer versions, it may lead to `Part ... intersects previous part` errors. To prevent this error, first install newer clickhouse-server packages on all cluster nodes and then do restarts (so, when clickhouse-server is restarted, it will start up with the new version).

#### New Feature

- Polygon dictionary type that provides efficient "reverse geocoding" lookups - to find the region by coordinates in a dictionary of many polygons (world map). It is using carefully optimized algorithm with recursive grids to maintain low CPU and memory usage. [#9278](#) ([achulkov2](#)).
- Added support of LDAP authentication for preconfigured users ("Simple Bind" method). [#11234](#) ([Denis Glazachev](#)).

- Introduce setting `alter_partition_verbose_result` which outputs information about touched parts for some types of `ALTER TABLE ... PARTITION ...` queries (currently `ATTACH` and `FREEZE`). Closes #8076. #13017 (alesapin).
- Add `bayesAB` function for bayesian-ab-testing. #12327 (achimbab).
- Added `system.crash_log` table into which stack traces for fatal errors are collected. This table should be empty. #12316 (alexey-milovidov).
- Added http headers `X-ClickHouse-Database` and `X-ClickHouse-Format` which may be used to set default database and output format. #12981 (hcz).
- Add `minMap` and `maxMap` functions support to `SimpleAggregateFunction`. #12662 (Ildus Kurbangaliev).
- Add setting `allow_non_metadata_alters` which restricts to execute `ALTER` queries which modify data on disk. Disabled by default. Closes #11547. #12635 (alesapin).
- A function `formatRow` is added to support turning arbitrary expressions into a string via given format. It's useful for manipulating SQL outputs and is quite versatile combined with the `columns` function. #12574 (Amos Bird).
- Add `FROM_UNIXTIME` function for compatibility with MySQL, related to 12149. #12484 (flynn).
- Allow Nullable types as keys in MergeTree tables if `allow_nullable_key` table setting is enabled. Closes #5319. #12433 (Amos Bird).
- Integration with COS. #12386 (fastio).
- Add `mapAdd` and `mapSubtract` functions for adding/subtracting key-mapped values. #11735 (Ildus Kurbangaliev).

## Bug Fix

- Fix premature `ON CLUSTER` timeouts for queries that must be executed on a single replica. Fixes #6704, #7228, #13361, #11884. #13450 (alesapin).
- Fix crash in mark inclusion search introduced in #12277. #14225 (Amos Bird).
- Fix race condition in external dictionaries with cache layout which can lead server crash. #12566 (alesapin).
- Fix visible data clobbering by progress bar in client in interactive mode. This fixes #12562 and #13369 and #13584 and fixes #12964. #13691 (alexey-milovidov).
- Fixed incorrect sorting order for `LowCardinality` columns when `ORDER BY` multiple columns is used. This fixes #13958. #14223 (Nikita Mikhaylov).
- Removed hardcoded timeout, which wrongly overruled `query_wait_timeout_milliseconds` setting for cache-dictionary. #14105 (Nikita Mikhaylov).
- Fixed wrong mount point in extra info for `Poco::Exception: no space left on device` #14050 (tavplubix).
- Fix wrong query optimization of select queries with `DISTINCT` keyword when subqueries also have `DISTINCT` in case `optimize_duplicate_order_by_and_distinct` setting is enabled. #13925 (Artem Zuikov).
- Fixed potential deadlock when renaming `Distributed` table. #13922 (tavplubix).
- Fix incorrect sorting for `FixedString` columns when `ORDER BY` multiple columns is used. Fixes #13182. #13887 (Nikolai Kochetov).
- Fix potentially lower precision of `topK`/`topKWeighted` aggregations (with non-default parameters). #13817 (Azat Khuzhin).

- Fix reading from MergeTree table with INDEX of type SET fails when compared against NULL. This fixes #13686. #13793 (Amos Bird).
- Fix step overflow in function `range()`. #13790 (Azat Khuzhin).
- Fixed `Directory not empty` error when concurrently executing `DROP DATABASE` and `CREATE TABLE`. #13756 (alexey-milovidov).
- Add range check for `h3KRing` function. This fixes #13633. #13752 (alexey-milovidov).
- Fix race condition between DETACH and background merges. Parts may revive after detach. This is continuation of #8602 that did not fix the issue but introduced a test that started to fail in very rare cases, demonstrating the issue. #13746 (alexey-milovidov).
- Fix logging Settings.Names/Values when `log_queries_min_type` greater than `QUERY_START`. #13737 (Azat Khuzhin).
- Fix incorrect message in `clickhouse-server.init` while checking user and group. #13711 (ylchou).
- Do not optimize `any(arrayJoin())` to `arrayJoin()` under `optimize_move_functions_out_of_any`. #13681 (Azat Khuzhin).
- Fixed possible deadlock in concurrent `ALTER ... REPLACE/MOVE PARTITION ...` queries. #13626 (tavplubix).
- Fixed the behaviour when sometimes cache-dictionary returned default value instead of present value from source. #13624 (Nikita Mikhaylov).
- Fix secondary indices corruption in compact parts (compact parts is an experimental feature). #13538 (Anton Popov).
- Fix wrong code in function `netloc`. This fixes #13335. #13446 (alexey-milovidov).
- Fix error in `parseDateTimeBestEffort` function when unix timestamp was passed as an argument. This fixes #13362. #13441 (alexey-milovidov).
- Fix invalid return type for comparison of tuples with `NULL` elements. Fixes #12461. #13420 (Nikolai Kochetov).
- Fix wrong optimization caused aggregate function `any(x)` is found inside another aggregate function in `queryerror` with `SET optimize_move_functions_out_of_any = 1` and aliases inside `any()`. #13419 (Artem Zuikov).
- Fix possible race in `StorageMemory`. #13416 (Nikolai Kochetov).
- Fix empty output for Arrow and Parquet formats in case if query return zero rows. It was done because empty output is not valid for this formats. #13399 (hc).
- Fix select queries with constant columns and prefix of primary key in `ORDER BY` clause. #13396 (Anton Popov).
- Fix `PrettyCompactMonoBlock` for clickhouse-local. Fix extremes/totals with `PrettyCompactMonoBlock`. Fixes #7746. #13394 (Azat Khuzhin).
- Fixed deadlock in `system.text_log`. #12452 (alexey-milovidov). It is a part of #12339. This fixes #12325. #13386 (Nikita Mikhaylov).
- Fixed `File(TSVWithNames*)` (header was written multiple times), fixed `clickhouse-local --format CSVWithNames*` (lacks header, broken after #12197), fixed `clickhouse-local --format CSVWithNames*` with zero rows (lacks header). #13343 (Azat Khuzhin).
- Fix segfault when function `groupArrayMovingSum` deserializes empty state. Fixes #13339. #13341 (alesapin).

- Throw error on `arrayJoin()` function in `JOIN ON` section. #13330 (Artem Zuikov).
- Fix crash in `LEFT ASOF JOIN` with `join_use_nulls=1`. #13291 (Artem Zuikov).
- Fix possible error `Totals` having transform was already added to pipeline in case of a query from delayed replica. #13290 (Nikolai Kochetov).
- The server may crash if user passed specifically crafted arguments to the function `h3ToChildren`. This fixes #13275. #13277 (alexey-milovidov).
- Fix potentially low performance and slightly incorrect result for `uniqExact`, `topK`, `sumDistinct` and similar aggregate functions called on `Float` types with `NaN` values. It also triggered assert in debug build. This fixes #12491. #13254 (alexey-milovidov).
- Fix assertion in `KeyCondition` when primary key contains expression with monotonic function and query contains comparison with constant whose type is different. This fixes #12465. #13251 (alexey-milovidov).
- Return passed number for numbers with MSB set in function `roundUpToPowerOfTwoOrZero()`. It prevents potential errors in case of overflow of array sizes. #13234 (Azat Khuzhin).
- Fix function if with nullable `constexpr` as cond that is not literal `NULL`. Fixes #12463. #13226 (alexey-milovidov).
- Fix assert in `arrayElement` function in case of array elements are `Nullable` and array subscript is also `Nullable`. This fixes #12172. #13224 (alexey-milovidov).
- Fix `DateTime64` conversion functions with constant argument. #13205 (Azat Khuzhin).
- Fix parsing row policies from `users.xml` when names of databases or tables contain dots. This fixes #5779, #12527. #13199 (Vitaly Baranov).
- Fix access to `redis` dictionary after connection was dropped once. It may happen with `cache` and `direct` dictionary layouts. #13082 (Anton Popov).
- Fix wrong index analysis with functions. It could lead to some data parts being skipped when reading from `MergeTree` tables. Fixes #13060. Fixes #12406. #13081 (Anton Popov).
- Fix error `Cannot convert column because it is constant but values of constants are different in source and result` for remote queries which use deterministic functions in scope of query, but not deterministic between queries, like `now()`, `now64()`, `randConstant()`. Fixes #11327. #13075 (Nikolai Kochetov).
- Fix crash which was possible for queries with `ORDER BY tuple` and small `LIMIT`. Fixes #12623. #13009 (Nikolai Kochetov).
- Fix `Block structure mismatch` error for queries with `UNION` and `JOIN`. Fixes #12602. #12989 (Nikolai Kochetov).
- Corrected `merge_with_ttl_timeout` logic which did not work well when expiration affected more than one partition over one time interval. (Authored by @excitoon). #12982 (Alexander Kazakov).
- Fix columns duplication for range hashed dictionary created from DDL query. This fixes #10605. #12857 (alesapin).
- Fix unnecessary limiting for the number of threads for selects from local replica. #12840 (Nikolai Kochetov).

- Fix rare bug when `ALTER DELETE` and `ALTER MODIFY COLUMN` queries executed simultaneously as a single mutation. Bug leads to an incorrect amount of rows in `count.txt` and as a consequence incorrect data in part. Also, fix a small bug with simultaneous `ALTER RENAME COLUMN` and `ALTER ADD COLUMN`. [#12760](#) ([alesapin](#)).
- Wrong credentials being used when using `clickhouse` dictionary source to query remote tables. [#12755](#) ([sundylis](#)).
- Fix `CAST(Nullable(String), Enum())`. [#12745](#) ([Azat Khuzhin](#)).
- Fix performance with large tuples, which are interpreted as functions in `IN` section. The case when user writes `WHERE x IN tuple(1, 2, ...)` instead of `WHERE x IN (1, 2, ...)` for some obscure reason. [#12700](#) ([Anton Popov](#)).
- Fix memory tracking for `input_format_parallel_parsing` (by attaching thread to group). [#12672](#) ([Azat Khuzhin](#)).
- Fix wrong optimization `optimize_move_functions_out_of_any=1` in case of `any(func(<lambda>))`. [#12664](#) ([Artem Zuikov](#)).
- Fixed [#10572](#) fix bloom filter index with const expression. [#12659](#) ([Winter Zhang](#)).
- Fix `SIGSEGV` in `StorageKafka` when broker is unavailable (and not only). [#12658](#) ([Azat Khuzhin](#)).
- Add support for function `if` with `Array(UUID)` arguments. This fixes [#11066](#). [#12648](#) ([alexey-milovidov](#)).
- `CREATE USER IF NOT EXISTS` now does not throw exception if the user exists. This fixes [#12507](#). [#12646](#) ([Vitaly Baranov](#)).
- Exception `There is no supertype...` can be thrown during `ALTER ... UPDATE` in unexpected cases (e.g. when subtracting from `UInt64` column). This fixes [#7306](#). This fixes [#4165](#). [#12633](#) ([alexey-milovidov](#)).
- Fix possible `Pipeline` stuck error for queries with external sorting. Fixes [#12617](#). [#12618](#) ([Nikolai Kochetov](#)).
- Fix error `Output of TreeExecutor is not sorted` for `OPTIMIZE DEDUPLICATE`. Fixes [#11572](#). [#12613](#) ([Nikolai Kochetov](#)).
- Fix the issue when alias on result of function `any` can be lost during query optimization. [#12593](#) ([Anton Popov](#)).
- Remove data for Distributed tables (blocks from async `INSERTs`) on `DROP TABLE`. [#12556](#) ([Azat Khuzhin](#)).
- Now ClickHouse will recalculate checksums for parts when file `checksums.txt` is absent. Broken since [#9827](#). [#12545](#) ([alesapin](#)).
- Fix bug which lead to broken old parts after `ALTER DELETE` query when `enable_mixed_granularity_parts=1`. Fixes [#12536](#). [#12543](#) ([alesapin](#)).
- Fixing race condition in live view tables which could cause data duplication. `LIVE VIEW` is an experimental feature. [#12519](#) ([vzakaznikov](#)).
- Fix backwards compatibility in binary format of `AggregateFunction(avg, ...)` values. This fixes [#12342](#). [#12486](#) ([alexey-milovidov](#)).
- Fix crash in `JOIN` with dictionary when we are joining over expression of dictionary key: `t JOIN dict ON expr(dict.id) = t.id`. Disable dictionary join optimisation for this case. [#12458](#) ([Artem Zuikov](#)).
- Fix overflow when very large `LIMIT` or `OFFSET` is specified. This fixes [#10470](#). This fixes [#11372](#). [#12427](#) ([alexey-milovidov](#)).

- kafka: fix SIGSEGV if there is a message with error in the middle of the batch. #12302 (Azat Khuzhin).

## Improvement

- Keep smaller amount of logs in ZooKeeper. Avoid excessive growing of ZooKeeper nodes in case of offline replicas when having many servers/tables/inserts. #13100 (alexey-milovidov).
- Now exceptions forwarded to the client if an error happened during ALTER or mutation. Closes #11329. #12666 (alesapin).
- Add `QueryTimeMicroseconds`, `SelectQueryTimeMicroseconds` and `InsertQueryTimeMicroseconds` to `system.events`, along with `system.metrics`, `processes`, `query_log`, etc. #13028 (ianton-ru).
- Added `SelectedRows` and `SelectedBytes` to `system.events`, along with `system.metrics`, `processes`, `query_log`, etc. #12638 (ianton-ru).
- Added `current_database` information to `system.query_log`. #12652 (Amos Bird).
- Allow `TabSeparatedRaw` as input format. #12009 (hcZ).
- Now `joinGet` supports multi-key lookup. #12418 (Amos Bird).
- Allow `*Map` aggregate functions to work on Arrays with NULLs. Fixes #13157. #13225 (alexey-milovidov).
- Avoid overflow in parsing of DateTime values that will lead to negative unix timestamp in their timezone (for example, 1970-01-01 00:00:00 in Moscow). Saturate to zero instead. This fixes #3470. This fixes #4172. #12443 (alexey-milovidov).
- AvroConfluent: Skip Kafka tombstone records - Support skipping broken records #13203 (Andrew Onyshchuk).
- Fix wrong error for long queries. It was possible to get syntax error other than Max query size exceeded for correct query. #13928 (Nikolai Kochetov).
- Fix data race in `Igamma` function. This race was caught only in `tsan`, no side effects really happened. #13842 (Nikolai Kochetov).
- Fix a 'Week'-interval formatting for ATTACH/ALTER/CREATE QUOTA-statements. #13417 (vladimir-golovchenko).
- Now broken parts are also reported when encountered in compact part processing. Compact parts is an experimental feature. #13282 (Amos Bird).
- Fix assert in `geohashesInBox`. This fixes #12554. #13229 (alexey-milovidov).
- Fix assert in `parseDateTimeBestEffort`. This fixes #12649. #13227 (alexey-milovidov).
- Minor optimization in Processors/PipelineExecutor: breaking out of a loop because it makes sense to do so. #13058 (Mark Papadakis).
- Support TRUNCATE table without TABLE keyword. #12653 (Winter Zhang).
- Fix explain query format overwrite by default. This fixes #12541. #12541 (BohuTANG).
- Allow to set JOIN kind and type in more standad way: `LEFT SEMI JOIN` instead of `SEMI LEFT JOIN`. For now both are correct. #12520 (Artem Zuikov).
- Changes default value for `multiple_joins_rewriter_version` to 2. It enables new multiple joins rewriter that knows about column names. #12469 (Artem Zuikov).
- Add several metrics for requests to S3 storages. #12464 (ianton-ru).

- Use correct default secure port for clickhouse-benchmark with `--secure` argument. This fixes #11044. #12440 (alexey-milovidov).
- Rollback insertion errors in `Log`, `TinyLog`, `StripeLog` engines. In previous versions insertion error lead to inconsistent table state (this works as documented and it is normal for these table engines). This fixes #12402. #12426 (alexey-milovidov).
- Implement `RENAME DATABASE` and `RENAME DICTIONARY` for Atomic database engine - Add implicit `{uuid}` macro, which can be used in ZooKeeper path for `ReplicatedMergeTree`. It works with `CREATE ... ON CLUSTER ...` queries. Set `show_table_uuid_in_table_create_query_if_not_nil` to true to use it. - Make `ReplicatedMergeTree` engine arguments optional, `/clickhouse/tables/{uuid}/{shard}/` and `{replica}` are used by default. Closes #12135. - Minor fixes. - These changes break backward compatibility of Atomic database engine. Previously created Atomic databases must be manually converted to new format. Atomic database is an experimental feature. #12343 (tavplubix).
- Separated `AWSAuthV4Signer` into different logger, removed excessive `AWSClient`: `AWSClient` from log messages. #12320 (Vladimir Chebotarev).
- Better exception message in disk access storage. #12625 (alesapin).
- Better exception for function in with invalid number of arguments. #12529 (Anton Popov).
- Fix error message about adaptive granularity. #12624 (alesapin).
- Fix SETTINGS parse after `FORMAT`. #12480 (Azat Khuzhin).
- If MergeTree table does not contain `ORDER BY` or `PARTITION BY`, it was possible to request `ALTER` to `CLEAR` all the columns and `ALTER` will stuck. Fixed #7941. #12382 (alexey-milovidov).
- Avoid re-loading completion from the history file after each query (to avoid history overlaps with other client sessions). #13086 (Azat Khuzhin).

## Performance Improvement

- Lower memory usage for some operations up to 2 times. #12424 (alexey-milovidov).
- Optimize PK lookup for queries that match exact PK range. #12277 (Ivan Babrou).
- Slightly optimize very short queries with `LowCardinality`. #14129 (Anton Popov).
- Slightly improve performance of aggregation by `UInt8/UInt16` keys. #13091 and #13055 (alexey-milovidov).
- Push down `LIMIT` step for query plan (inside subqueries). #13016 (Nikolai Kochetov).
- Parallel primary key lookup and skipping index stages on parts, as described in #11564. #12589 (Ivan Babrou).
- Converting String-type arguments of function "if" and "transform" into enum if `set optimize_if_transform_strings_to_enum = 1`. #12515 (Artem Zuikov).
- Replaces monotonic functions with its argument in `ORDER BY` if `set optimize_monotonous_functions_in_order_by=1`. #12467 (Artem Zuikov).
- Add order by optimization that rewrites `ORDER BY x, f(x)` with `ORDER by x` if `set optimize_redundant_functions_in_order_by = 1`. #12404 (Artem Zuikov).
- Allow pushdown predicate when subquery contains `WITH` clause. This fixes #12293 #12663 (Winter Zhang).

- Improve performance of reading from compact parts. Compact parts is an experimental feature. #12492 (Anton Popov).
- Attempt to implement streaming optimization in `DiskS3`. DiskS3 is an experimental feature. #12434 (Vladimir Chebotarev).

## Build/Testing/Packaging Improvement

- Use `shellcheck` for sh tests linting. #13200 #13207 (alexey-milovidov).
- Add script which set labels for pull requests in GitHub hook. #13183 (alesapin).
- Remove some of recursive submodules. See #13378. #13379 (alexey-milovidov).
- Ensure that all the submodules are from proper URLs. Continuation of #13379. This fixes #13378. #13397 (alexey-milovidov).
- Added support for user-declared settings, which can be accessed from inside queries. This is needed when ClickHouse engine is used as a component of another system. #13013 (Vitaly Baranov).
- Added testing for RBAC functionality of `INSERT` privilege in TestFlows. Expanded tables on which `SELECT` is being tested. Added Requirements to match new table engine tests. #13340 (MyroTk).
- Fix timeout error during server restart in the stress test. #13321 (alesapin).
- Now fast test will wait server with retries. #13284 (alesapin).
- Function `materialize()` (the function for ClickHouse testing) will work for `NULL` as expected - by transforming it to non-constant column. #13212 (alexey-milovidov).
- Fix libunwind build in AArch64. This fixes #13204. #13208 (alexey-milovidov).
- Even more retries in zkutil gtest to prevent test flakiness. #13165 (alexey-milovidov).
- Small fixes to the RBAC TestFlows. #13152 (vzakaznikov).
- Fixing `00960_live_view_watch_events_live.py` test. #13108 (vzakaznikov).
- Improve cache purge in documentation deploy script. #13107 (alesapin).
- Rewrote some orphan tests to gtest. Removed useless includes from tests. #13073 (Nikita Mikhaylov).
- Added tests for RBAC functionality of `SELECT` privilege in TestFlows. #13061 (Ritaank Tiwari).
- Rerun some tests in fast test check. #12992 (alesapin).
- Fix MSan error in "rdkafka" library. This closes #12990. Updated `rdkafka` to version 1.5 (master). #12991 (alexey-milovidov).
- Fix UBSan report in base64 if tests were run on server with AVX-512. This fixes #12318. Author: @qoega. #12441 (alexey-milovidov).
- Fix UBSan report in HDFS library. This closes #12330. #12453 (alexey-milovidov).
- Check an ability that we able to restore the backup from an old version to the new version. This closes #8979. #12959 (alesapin).
- Do not build `helper_container` image inside integrational tests. Build docker container in CI and use pre-built `helper_container` in integration tests. #12953 (Ilya Yatsishin).
- Add a test for `ALTER TABLE CLEAR COLUMN` query for primary key columns. #12951 (alesapin).
- Increased timeouts in testflows tests. #12949 (vzakaznikov).

- Fix build of test under Mac OS X. This closes #12767. #12772 (alexey-milovidov).
- Connector-ODBC updated to mysql-connector-odbc-8.0.21. #12739 (Ilya Yatsishin).
- Adding RBAC syntax tests in TestFlows. #12642 (vzakaznikov).
- Improve performance of TestKeeper. This will speedup tests with heavy usage of Replicated tables. #12505 (alexey-milovidov).
- Now we check that server is able to start after stress tests run. This fixes #12473. #12496 (alesapin).
- Update fmtlib to master (7.0.1). #12446 (alexey-milovidov).
- Add docker image for fast tests. #12294 (alesapin).
- Rework configuration paths for integration tests. #12285 (Ilya Yatsishin).
- Add compiler option to control that stack frames are not too large. This will help to run the code in fibers with small stack size. #11524 (alexey-milovidov).
- Update gitignore-files. #13447 (vladimir-golovchenko).

## ClickHouse release 20.6

### ClickHouse release v20.6.3.28-stable

#### Backward Incompatible Change

- When upgrading from versions older than 20.5, if rolling update is performed and cluster contains both versions 20.5 or greater and less than 20.5, if ClickHouse nodes with old versions are restarted and old version has been started up in presence of newer versions, it may lead to `Part ... intersects previous part` errors. To prevent this error, first install newer clickhouse-server packages on all cluster nodes and then do restarts (so, when clickhouse-server is restarted, it will start up with the new version).

#### New Feature

- Added an initial implementation of EXPLAIN query. Syntax: `EXPLAIN SELECT ....` This fixes #1118. #11873 (Nikolai Kochetov).
- Added storage RabbitMQ. #11069 (Kseniia Sumarokova).
- Implemented PostgreSQL-like `ILIKE` operator for #11710. #12125 (Mike).
- Supported RIGHT and FULL JOIN with `SET join_algorithm = 'partial_merge'`. Only ALL strictness is allowed (ANY, SEMI, ANTI, ASOF are not). #12118 (Artem Zuikov).
- Added a function `initializeAggregation` to initialize an aggregation based on a single value. #12109 (Guillaume Tassery).
- Supported `ALTER TABLE ... [ADD|MODIFY] COLUMN ... FIRST`#4006. #12073 (Winter Zhang).
- Added function `parseDateTimeBestEffortUS`. #12028 (flynn).
- Support format ORC for output (was supported only for input). #11662 (Kruglov Pavel).

#### Bug Fix

- Fixed aggregate function `any(x)` is found inside another aggregate function in `queryerror` with `SET optimize_move_functions_out_of_any = 1` and aliases inside `any()`. #13419 (Artem Zuikov).
- Fixed PrettyCompactMonoBlock for clickhouse-local. Fixed extremes/totals with PrettyCompactMonoBlock. This fixes #7746. #13394 (Azat Khuzhin).

- Fixed possible error `Totals` having transform was already added to pipeline in case of a query from delayed replica. #13290 (Nikolai Kochetov).
- The server may crash if user passed specifically crafted arguments to the function `h3ToChildren`. This fixes #13275. #13277 (alexey-milovidov).
- Fixed potentially low performance and slightly incorrect result for `uniqExact`, `topK`, `sumDistinct` and similar aggregate functions called on `Float` types with `Nan` values. It also triggered assert in debug build. This fixes #12491. #13254 (alexey-milovidov).
- Fixed function if with nullable `constexpr` as cond that is not literal `NULL`. Fixes #12463. #13226 (alexey-milovidov).
- Fixed assert in `arrayElement` function in case of array elements are `Nullable` and array subscript is also `Nullable`. This fixes #12172. #13224 (alexey-milovidov).
- Fixed `DateTime64` conversion functions with constant argument. #13205 (Azat Khuzhin).
- Fixed wrong index analysis with functions. It could lead to pruning wrong parts, while reading from `MergeTree` tables. Fixes #13060. Fixes #12406. #13081 (Anton Popov).
- Fixed error Cannot convert column because it is constant but values of constants are different in source and result for remote queries which use deterministic functions in scope of query, but not deterministic between queries, like `now()`, `now64()`, `randConstant()`. Fixes #11327. #13075 (Nikolai Kochetov).
- Fixed unnecessary limiting for the number of threads for selects from local replica. #12840 (Nikolai Kochetov).
- Fixed rare bug when `ALTER DELETE` and `ALTER MODIFY COLUMN` queries executed simultaneously as a single mutation. Bug leads to an incorrect amount of rows in `count.txt` and as a consequence incorrect data in part. Also, fix a small bug with simultaneous `ALTER RENAME COLUMN` and `ALTER ADD COLUMN`. #12760 (alesapin).
- Fixed `CAST(Nullable(String), Enum())`. #12745 (Azat Khuzhin).
- Fixed a performance with large tuples, which are interpreted as functions in `IN` section. The case when user write `WHERE x IN tuple(1, 2, ...)` instead of `WHERE x IN (1, 2, ...)` for some obscure reason. #12700 (Anton Popov).
- Fixed memory tracking for `input_format_parallel_parsing` (by attaching thread to group). #12672 (Azat Khuzhin).
- Fixed bloom filter index with const expression. This fixes #10572. #12659 (Winter Zhang).
- Fixed `SIGSEGV` in `StorageKafka` when broker is unavailable (and not only). #12658 (Azat Khuzhin).
- Added support for function if with `Array(UUID)` arguments. This fixes #11066. #12648 (alexey-milovidov).
- `CREATE USER IF NOT EXISTS` now does not throw exception if the user exists. This fixes #12507. #12646 (Vitaly Baranov).
- Better exception message in disk access storage. #12625 (alesapin).
- The function `groupArrayMoving*` was not working for distributed queries. Its result was calculated within incorrect data type (without promotion to the largest type). The function `groupArrayMovingAvg` was returning integer number that was inconsistent with the `avg` function. This fixes #12568. #12622 (alexey-milovidov).
- Fixed lack of aliases with function `any`. #12593 (Anton Popov).

- Fixed race condition in external dictionaries with cache layout which can lead server crash. #12566 (alesapin).
- Remove data for Distributed tables (blocks from async INSERTs) on DROP TABLE. #12556 (Azat Khuzhin).
- Fixed bug which lead to broken old parts after `ALTER DELETE` query when `enable_mixed_granularity_parts=1`. Fixes #12536. #12543 (alesapin).
- Better exception for function `in` with invalid number of arguments. #12529 (Anton Popov).
- Fixing race condition in live view tables which could cause data duplication. #12519 (vzakaznikov).
- Fixed performance issue, while reading from compact parts. #12492 (Anton Popov).
- Fixed backwards compatibility in binary format of `AggregateFunction(avg, ...)` values. This fixes #12342. #12486 (alexey-milovidov).
- Fixed SETTINGS parse after FORMAT. #12480 (Azat Khuzhin).
- Fixed the deadlock if `text_log` is enabled. #12452 (alexey-milovidov).
- Fixed overflow when very large `LIMIT` or `OFFSET` is specified. This fixes #10470. This fixes #11372. #12427 (alexey-milovidov).
- Fixed possible segfault if `StorageMerge`. This fixes #12054. #12401 (tavplubix).
- Reverted change introduced in #11079 to resolve #12098. #12397 (Mike).
- Additional check for arguments of bloom filter index. This fixes #11408. #12388 (alexey-milovidov).
- Avoid exception when negative or floating point constant is used in WHERE condition for indexed tables. This fixes #11905. #12384 (alexey-milovidov).
- Allowed to `CLEAR` column even if there are depending `DEFAULT` expressions. This fixes #12333. #12378 (alexey-milovidov).
- Fix `TOTALS/ROLLUP/CUBE` for aggregate functions with `-State` and `Nullable` arguments. This fixes #12163. #12376 (alexey-milovidov).
- Fixed error message and exit codes for `ALTER RENAME COLUMN` queries, when `RENAME` is not allowed. Fixes #12301 and #12303. #12335 (alesapin).
- Fixed very rare race condition in `ReplicatedMergeTreeQueue`. #12315 (alexey-milovidov).
- When using codec `Delta` or `DoubleDelta` with non fixed width types, exception with code `LOGICAL_ERROR` was returned instead of exception with code `BAD_ARGUMENTS` (we ensure that exceptions with code logical error never happen). This fixes #12110. #12308 (alexey-milovidov).
- Fixed order of columns in `WITH FILL` modifier. Previously order of columns of `ORDER BY` statement wasn't respected. #12306 (Anton Popov).
- Avoid "bad cast" exception when there is an expression that filters data by virtual columns (like `_table` in `Merge` tables) or by "index" columns in system tables such as filtering by database name when querying from `system.tables`, and this expression returns `Nullable` type. This fixes #12166. #12305 (alexey-milovidov).
- Fixed `TTL` after renaming column, on which depends `TTL` expression. #12304 (Anton Popov).
- Fixed SIGSEGV if there is an message with error in the middle of the batch in Kafka Engine. #12302 (Azat Khuzhin).

- Fixed the situation when some threads might randomly hang for a few seconds during DNS cache updating. #12296 (tavplubix).
- Fixed typo in setting name. #12292 (alexey-milovidov).
- Show error after TrieDictionary failed to load. #12290 (Vitaly Baranov).
- The function `arrayFill` worked incorrectly for empty arrays that may lead to crash. This fixes #12263. #12279 (alexey-milovidov).
- Implement conversions to the common type for `LowCardinality` types. This allows to execute UNION ALL of tables with columns of `LowCardinality` and other columns. This fixes #8212. This fixes #4342. #12275 (alexey-milovidov).
- Fixed the behaviour on reaching redirect limit in request to S3 storage. #12256 (ianton-ru).
- Fixed the behaviour when during multiple sequential inserts in `StorageFile` header for some special types was written more than once. This fixed #6155. #12197 (Nikita Mikhaylov).
- Fixed logical functions for `UInt8` values when they are not equal to 0 or 1. #12196 (Alexander Kazakov).
- Cap `max_memory_usage*` limits to the process resident memory. #12182 (Azat Khuzhin).
- Fix dictGet arguments check during `GROUP BY` injective functions elimination. #12179 (Azat Khuzhin).
- Fixed the behaviour when `SummingMergeTree` engine sums up columns from partition key. Added an exception in case of explicit definition of columns to sum which intersects with partition key columns. This fixes #7867. #12173 (Nikita Mikhaylov).
- Don't split the dictionary source's table name into schema and table name itself if ODBC connection does not support schema. #12165 (Vitaly Baranov).
- Fixed wrong logic in `ALTER DELETE` that leads to deleting of records when condition evaluates to NULL. This fixes #9088. This closes #12106. #12153 (alexey-milovidov).
- Fixed transform of query to send to external DBMS (e.g. MySQL, ODBC) in presence of aliases. This fixes #12032. #12151 (alexey-milovidov).
- Fixed bad code in redundant ORDER BY optimization. The bug was introduced in #10067. #12148 (alexey-milovidov).
- Fixed potential overflow in integer division. This fixes #12119. #12140 (alexey-milovidov).
- Fixed potential infinite loop in `greatCircleDistance`, `geoDistance`. This fixes #12117. #12137 (alexey-milovidov).
- Normalize "pid" file handling. In previous versions the server may refuse to start if it was killed without proper shutdown and if there is another process that has the same pid as previously runned server. Also pid file may be removed in unsuccessful server startup even if there is another server running. This fixes #3501. #12133 (alexey-milovidov).
- Fixed bug which leads to incorrect table metadata in ZooKeepeer for ReplicatedVersionedCollapsingMergeTree tables. Fixes #12093. #12121 (alesapin).
- Avoid "There is no query" exception for materialized views with joins or with subqueries attached to system logs (`system.query_log`, `metric_log`, etc) or to `engine=Buffer` underlying table. #12120 (filimonov).
- Fixed handling dependency of table with `ENGINE=Dictionary` on dictionary. This fixes #10994. This fixes #10397. #12116 (Vitaly Baranov).

- Format Parquet now properly works with `LowCardinality` and `LowCardinality(Nullable)` types. Fixes #12086, #8406. #12108 (Nikolai Kochetov).
- Fixed performance for selects with `UNION` caused by wrong limit for the total number of threads. Fixes #12030. #12103 (Nikolai Kochetov).
- Fixed segfault with `-StateResample` combinators. #12092 (Anton Popov).
- Fixed empty `result_rows` and `result_bytes` metrics in `system.quey_log` for selects. Fixes #11595. #12089 (Nikolai Kochetov).
- Fixed unnecessary limiting the number of threads for selects from `VIEW`. Fixes #11937. #12085 (Nikolai Kochetov).
- Fixed SIGSEGV in StorageKafka on `DROP TABLE`. #12075 (Azat Khuzhin).
- Fixed possible crash while using wrong type for `PREWHERE`. Fixes #12053, #12060. #12060 (Nikolai Kochetov).
- Fixed error `Cannot capture column` for higher-order functions with `Tuple(LowCardinality)` argument. Fixes #9766. #12055 (Nikolai Kochetov).
- Fixed constraints check if constraint is a constant expression. This fixes #11360. #12042 (alexey-milovidov).
- Fixed wrong result and potential crash when invoking function `if` with arguments of type `FixedString` with different sizes. This fixes #11362. #12021 (alexey-milovidov).

## Improvement

- Allowed to set `JOIN` kind and type in more standard way: `LEFT SEMI JOIN` instead of `SEMI LEFT JOIN`. For now both are correct. #12520 (Artem Zuikov).
- `lifetime_rows/lifetime_bytes` for Buffer engine. #12421 (Azat Khuzhin).
- Write the detail exception message to the client instead of 'MySQL server has gone away'. #12383 (BohuTANG).
- Allows to change a charset which is used for printing grids borders. Available charsets are following: UTF-8, ASCII. Setting `output_format_pretty_grid_charset` enables this feature. #12372 (Sabyanin Maxim).
- Supported MySQL 'SELECT DATABASE()' #9336 2. Add MySQL replacement query integration test. #12314 (BohuTANG).
- Added `KILL QUERY [connection_id]` for the MySQL client/driver to cancel the long query, issue #12038. #12152 (BohuTANG).
- Added support for `%g` (two digit ISO year) and `%G` (four digit ISO year) substitutions in `formatDateTime` function. #12136 (vivarum).
- Added 'type' column in `system.disks`. #12115 (ianton-ru).
- Improved `REVOKE` command: now it requires grant/admin option for only access which will be revoked. For example, to execute `REVOKE ALL ON *.* FROM user1` now it does not require to have full access rights granted with grant option. Added command `REVOKE ALL FROM user1` - it revokes all granted roles from `user1`. #12083 (Vitaly Baranov).
- Added replica priority for `load_balancing` (for manual prioritization of the load balancing). #11995 (Azat Khuzhin).

- Switched paths in S3 metadata to relative which allows to handle S3 blobs more easily. #11892 ([Vladimir Chebotarev](#)).

## Performance Improvement

- Improved performance of 'ORDER BY' and 'GROUP BY' by prefix of sorting key (enabled with `optimize_aggregation_in_order` setting, disabled by default). #11696 ([Anton Popov](#)).
- Removed injective functions inside `uniq*()` if `set optimize_injective_functions_inside_uniq=1`. #12337 ([Ruslan Kamalov](#)).
- Index not used for IN operator with literals, performance regression introduced around v19.3. This fixes #10574. #12062 ([nvartolomei](#)).
- Implemented single part uploads for DiskS3 (experimental feature). #12026 ([Vladimir Chebotarev](#)).

## Experimental Feature

- Added new in-memory format of parts in MergeTree-family tables, which stores data in memory. Parts are written on disk at first merge. Part will be created in in-memory format if its size in rows or bytes is below thresholds `min_rows_for_compact_part` and `min_bytes_for_compact_part`. Also optional support of Write-Ahead-Log is available, which is enabled by default and is controlled by setting `in_memory_parts_enable_wal`. #10697 ([Anton Popov](#)).

## Build/Testing/Packaging Improvement

- Implement AST-based query fuzzing mode for clickhouse-client. See [this label](#) for the list of issues we recently found by fuzzing. Most of them were found by this tool, and a couple by SQLancer and `00746_sql_fuzzy.pl`. #12111 ([Alexander Kuzmenkov](#)).
- Add new type of tests based on Testflows framework. #12090 ([vzakaznikov](#)).
- Added S3 HTTPS integration test. #12412 ([Pavel Kovalenko](#)).
- Log sanitizer trap messages from separate thread. This will prevent possible deadlock under thread sanitizer. #12313 ([alexey-milovidov](#)).
- Now functional and stress tests will be able to run with old version of `clickhouse-test` script. #12287 ([alesapin](#)).
- Remove strange file creation during build in `orc`. #12258 ([Nikita Mikhaylov](#)).
- Place common docker compose files to integration docker container. #12168 ([Ilya Yatsishin](#)).
- Fix warnings from CodeQL. `CodeQL` is another static analyzer that we will use along with `clang-tidy` and `PVS-Studio` that we use already. #12138 ([alexey-milovidov](#)).
- Minor CMake fixes for UNBUNDLED build. #12131 ([Matwey V. Kornilov](#)).
- Added a showcase of the minimal Docker image without using any Linux distribution. #12126 ([alexey-milovidov](#)).
- Perform an upgrade of system packages in the `clickhouse-server` docker image. #12124 ([Ivan Blinkov](#)).
- Add `UNBUNDLED` flag to `system.build_options` table. Move skip lists for `clickhouse-test` to `clickhouse` repo. #12107 ([alesapin](#)).
- Regular check by [Anchore Container Analysis](#) security analysis tool that looks for [CVE](#) in `clickhouse-server` Docker image. Also confirms that `Dockerfile` is buildable. Runs daily on master and on pull-requests to `Dockerfile`. #12102 ([Ivan Blinkov](#)).
- Daily check by [GitHub CodeQL](#) security analysis tool that looks for [CWE](#). #12101 ([Ivan Blinkov](#)).

- Install `ca-certificates` before the first `apt-get update` in Dockerfile. #12095 (Ivan Blinkov).

## ClickHouse release 20.5

### ClickHouse release v20.5.4.40-stable 2020-08-10

#### Bug Fix

- Fixed wrong index analysis with functions. It could lead to pruning wrong parts, while reading from `MergeTree` tables. Fixes #13060. Fixes #12406. #13081 (Anton Popov).
- Fixed unnecessary limiting for the number of threads for selects from local replica. #12840 (Nikolai Kochetov).
- Fixed performance with large tuples, which are interpreted as functions in `IN` section. The case when user write `WHERE x IN tuple(1, 2, ...)` instead of `WHERE x IN (1, 2, ...)` for some obscure reason. #12700 (Anton Popov).
- Fixed memory tracking for `input_format_parallel_parsing` (by attaching thread to group). #12672 (Azat Khuzhin).
- Fixed bloom filter index with const expression. This fixes #10572. #12659 (Winter Zhang).
- Fixed `SIGSEGV` in `StorageKafka` when broker is unavailable (and not only). #12658 (Azat Khuzhin).
- Added support for function `if` with `Array(UUID)` arguments. This fixes #11066. #12648 (alexey-milovidov).
- Fixed lack of aliases with function `any`. #12593 (Anton Popov).
- Fixed race condition in external dictionaries with cache layout which can lead server crash. #12566 (alesapin).
- Remove data for Distributed tables (blocks from async INSERTs) on `DROP TABLE`. #12556 (Azat Khuzhin).
- Fixed bug which lead to broken old parts after `ALTER DELETE` query when `enable_mixed_granularity_parts=1`. Fixes #12536. #12543 (alesapin).
- Better exception for function `in` with invalid number of arguments. #12529 (Anton Popov).
- Fixed race condition in live view tables which could cause data duplication. #12519 (vzakaznikov).
- Fixed performance issue, while reading from compact parts. #12492 (Anton Popov).
- Fixed backwards compatibility in binary format of `AggregateFunction(avg, ...)` values. This fixes #12342. #12486 (alexey-milovidov).
- Fixed the deadlock if `text_log` is enabled. #12452 (alexey-milovidov).
- Fixed overflow when very large `LIMIT` or `OFFSET` is specified. This fixes #10470. This fixes #11372. #12427 (alexey-milovidov).
- Fixed possible segfault if `StorageMerge`. Closes #12054. #12401 (tavplubix).
- Reverts change introduced in #11079 to resolve #12098. #12397 (Mike).
- Avoid exception when negative or floating point constant is used in `WHERE` condition for indexed tables. This fixes #11905. #12384 (alexey-milovidov).
- Allow to `CLEAR` column even if there are depending `DEFAULT` expressions. This fixes #12333. #12378 (alexey-milovidov).

- Fixed TOTALS/ROLLUP/CUBE for aggregate functions with `-State` and `Nullable` arguments. This fixes #12163. #12376 (alexey-milovidov).
- Fixed SIGSEGV if there is an message with error in the middle of the batch in Kafka Engine. #12302 (Azat Khuzhin).
- Fixed the behaviour when `SummingMergeTree` engine sums up columns from partition key. Added an exception in case of explicit definition of columns to sum which intersects with partition key columns. This fixes #7867. #12173 (Nikita Mikhaylov).
- Fixed transform of query to send to external DBMS (e.g. MySQL, ODBC) in presence of aliases. This fixes #12032. #12151 (alexey-milovidov).
- Fixed bug which leads to incorrect table metadata in ZooKeepeer for ReplicatedVersionedCollapsingMergeTree tables. Fixes #12093. #12121 (alesapin).
- Fixed unnecessary limiting the number of threads for selects from VIEW. Fixes #11937. #12085 (Nikolai Kochetov).
- Fixed crash in JOIN with LowCardinality type with `join_algorithm=partial_merge`. #12035 (Artem Zuikov).
- Fixed wrong result for `if()` with NULLs in condition. #11807 (Artem Zuikov).

## Performance Improvement

- Index not used for IN operator with literals, performance regression introduced around v19.3. This fixes #10574. #12062 (nvartolomei).

## Build/Testing/Packaging Improvement

- Install ca-certificates before the first apt-get update in Dockerfile. #12095 (Ivan Blinkov).

# ClickHouse release v20.5.2.7-stable 2020-07-02

## Backward Incompatible Change

- Return non-`Nullable` result from COUNT(DISTINCT), and `uniq` aggregate functions family. If all passed values are NULL, return zero instead. This improves SQL compatibility. #11661 (alexey-milovidov).
- Added a check for the case when user-level setting is specified in a wrong place. User-level settings should be specified in `users.xml` inside `<profile>` section for specific user profile (or in `<default>` for default settings). The server won't start with exception message in log. This fixes #9051. If you want to skip the check, you can either move settings to the appropriate place or add `<skip_check_for_incorrect_settings>1</skip_check_for_incorrect_settings>` to config.xml. #11449 (alexey-milovidov).
- The setting `input_format_with_names_use_header` is enabled by default. It will affect parsing of input formats `-WithNames` and `-WithNamesAndTypes`. #10937 (alexey-milovidov).
- Remove `experimental_use_processors` setting. It is enabled by default. #10924 (Nikolai Kochetov).
- Update zstd to 1.4.4. It has some minor improvements in performance and compression ratio. If you run replicas with different versions of ClickHouse you may see reasonable error messages `Data after merge is not byte-identical to data on another replicas.` with explanation. These messages are Ok and you should not worry. This change is backward compatible but we list it here in changelog in case you will wonder about these messages. #10663 (alexey-milovidov).
- Added a check for meaningless codecs and a setting `allow_suspicious_codecs` to control this check. This closes #4966. #10645 (alexey-milovidov).
- Several Kafka setting changes their defaults. See #11388.

- When upgrading from versions older than 20.5, if rolling update is performed and cluster contains both versions 20.5 or greater and less than 20.5, if ClickHouse nodes with old versions are restarted and old version has been started up in presence of newer versions, it may lead to `Part ... intersects previous part` errors. To prevent this error, first install newer clickhouse-server packages on all cluster nodes and then do restarts (so, when clickhouse-server is restarted, it will start up with the new version).

## New Feature

- `TTL DELETE WHERE` and `TTL GROUP BY` for automatic data coarsening and rollup in tables. #10537 (expl0si0nn).
- Implementation of PostgreSQL wire protocol. #10242 (Movses).
- Added system tables for users, roles, grants, settings profiles, quotas, row policies; added commands `SHOW USER`, `SHOW [CURRENT|ENABLED] ROLES`, `SHOW SETTINGS PROFILES`. #10387 (Vitaly Baranov).
- Support writes in ODBC Table function #10554 (ageraab). #10901 (tavplubix).
- Add query performance metrics based on Linux `perf_events` (these metrics are calculated with hardware CPU counters and OS counters). It is optional and requires `CAP_SYS_ADMIN` to be set on clickhouse binary. #9545 Andrey Skobtsov. #11226 (Alexander Kuzmenkov).
- Now support `NULL` and `NOT NULL` modifiers for data types in `CREATE` query. #11057 (Павел Потемкин).
- Add `ArrowStream` input and output format. #11088 (hczi).
- Support Cassandra as external dictionary source. #4978 (favstovol).
- Added a new layout `direct` which loads all the data directly from the source for each query, without storing or caching data. #10622 (Artem Streltsov).
- Added new `complex_key_direct` layout to dictionaries, that does not store anything locally during query execution. #10850 (Artem Streltsov).
- Added support for MySQL style global variables syntax (stub). This is needed for compatibility of MySQL protocol. #11832 (alexey-milovidov).
- Added syntax highlighting to `clickhouse-client` using `replxx`. #11422 (Tagir Kuskarov).
- `minMap` and `maxMap` functions were added. #11603 (Ildus Kurbangaliev).
- Add the `system.asynchronous_metric_log` table that logs historical metrics from `system.asynchronous_metrics`. #11588 (Alexander Kuzmenkov).
- Add functions `extractAllGroupsHorizontal(haystack, re)` and `extractAllGroupsVertical(haystack, re)`. #11554 (Vasily Nemkov).
- Add `SHOW CLUSTER(S)` queries. #11467 (hexiaoting).
- Add `netloc` function for extracting network location, similar to `urlparse(url), netloc` in python. #11356 (Guillaume Tassery).
- Add 2 more virtual columns for `engine=Kafka` to access message headers. #11283 (filimonov).
- Add `_timestamp_ms` virtual column for Kafka engine (type is `Nullable(DateTime64(3))`). #11260 (filimonov).
- Add function `randomFixedString`. #10866 (Andrei Nekrashevich).
- Add function `fuzzBits` that randomly flips bits in a string with given probability. #11237 (Andrei Nekrashevich).

- Allow comparison of numbers with constant string in comparison operators, IN and VALUES sections. #11647 (alexey-milovidov).
- Add `round_robin` load\_balancing mode. #11645 (Azat Khuzhin).
- Add `cast_keep_nullable` setting. If set `CAST(something_nullable AS Type)` return `Nullable(Type)`. #11733 (Artem Zuikov).
- Added column position to `system.columns` table and `column_position` to `system.parts_columns` table. It contains ordinal position of a column in a table starting with 1. This closes #7744. #11655 (alexey-milovidov).
- ON CLUSTER support for SYSTEM {FLUSH DISTRIBUTED,STOP/START DISTRIBUTED SEND}. #11415 (Azat Khuzhin).
- Add `system.distribution_queue` table. #11394 (Azat Khuzhin).
- Support for all format settings in Kafka, expose some setting on table level, adjust the defaults for better performance. #11388 (filimonov).
- Add `port` function (to extract port from URL). #11120 (Azat Khuzhin).
- Now `dictGet*` functions accept table names. #11050 (Vitaly Baranov).
- The `clickhouse-format` tool is now able to format multiple queries when the `-n` argument is used. #10852 (Dario).
- Possibility to configure proxy-resolver for DiskS3. #10744 (Pavel Kovalenko).
- Make `pointInPolygon` work with non-constant polygon. `PointInPolygon` now can take `Array(Array(Tuple(..., ...)))` as second argument, array of polygon and holes. #10623 (Alexey Ilyukhov) #11421 (Alexey Ilyukhov).
- Added `move_ttl_info` to `system.parts` in order to provide introspection of move TTL functionality. #10591 (Vladimir Chebotarev).
- Possibility to work with S3 through proxies. #10576 (Pavel Kovalenko).
- Add `NCHAR` and `NVARCHAR` synonyms for data types. #11025 (alexey-milovidov).
- Resolved #7224: added `FailedQuery`, `FailedSelectQuery` and `FailedInsertQuery` metrics to `system.events` table. #11151 (Nikita Orlov).
- Add more `jemalloc` statistics to `system.asynchronous_metrics`, and ensure that we see up-to-date values for them. #11748 (Alexander Kuzmenkov).
- Allow to specify default S3 credentials and custom auth headers. #11134 (Grigory Pervakov).
- Added new functions to import/export `DateTime64` as `Int64` with various precision: `to-/fromUnixTimestamp64Milli/-Micro/-Nano`. #10923 (Vasily Nemkov).
- Allow specifying `mongodb://` URI for MongoDB dictionaries. #10915 (Alexander Kuzmenkov).
- `OFFSET` keyword can now be used without an affiliated `LIMIT` clause. #10802 (Guillaume Tassery).
- Added `system.licenses` table. This table contains licenses of third-party libraries that are located in `contrib` directory. This closes #2890. #10795 (alexey-milovidov).
- New function function `toStartOfSecond(DateTime64) -> DateTime64` that nullifies sub-second part of `DateTime64` value. #10722 (Vasily Nemkov).
- Add new input format `JSONAsString` that accepts a sequence of JSON objects separated by newlines, spaces and/or commas. #10607 (Kruglov Pavel).

- Allowed to profile memory with finer granularity steps than 4 MiB. Added sampling memory profiler to capture random allocations/deallocations. [#10598](#) ([alexey-milovidov](#)).
- SimpleAggregateFunction now also supports sumMap. [#10000](#) ([Ildus Kurbangaliev](#)).
- Support `ALTER RENAME COLUMN` for the distributed table engine. Continuation of [#10727](#). Fixes [#10747](#). [#10887](#) ([alesapin](#)).

## Bug Fix

- Fix UBSan report in Decimal parse. This fixes [#7540](#). [#10512](#) ([alexey-milovidov](#)).
- Fix potential floating point exception when parsing DateTime64. This fixes [#11374](#). [#11875](#) ([alexey-milovidov](#)).
- Fix rare crash caused by using `Nullable` column in prewhere condition. [#11895](#) [#11608](#) [#11869](#) ([Nikolai Kochetov](#)).
- Don't allow arrayJoin inside higher order functions. It was leading to broken protocol synchronization. This closes [#3933](#). [#11846](#) ([alexey-milovidov](#)).
- Fix wrong result of comparison of `FixedString` with constant String. This fixes [#11393](#). This bug appeared in version 20.4. [#11828](#) ([alexey-milovidov](#)).
- Fix wrong result for `if` with `NULLs` in condition. [#11807](#) ([Artem Zuikov](#)).
- Fix using too many threads for queries. [#11788](#) ([Nikolai Kochetov](#)).
- Fixed Scalar does not exist exception when using `WITH <scalar subquery> ...` in `SELECT ... FROM merge_tree_table ...`. [#11621](#). [#11767](#) ([Amos Bird](#)).
- Fix unexpected behaviour of queries like `SELECT *, xyz.*` which were success while an error expected. [#11753](#) ([hexiaoting](#)).
- Now replicated fetches will be cancelled during metadata alter. [#11744](#) ([alesapin](#)).
- Parse metadata stored in zookeeper before checking for equality. [#11739](#) ([Azat Khuzhin](#)).
- Fixed `LOGICAL_ERROR` caused by wrong type deduction of complex literals in Values input format. [#11732](#) ([tavplubix](#)).
- Fix `ORDER BY ... WITH FILL` over const columns. [#11697](#) ([Anton Popov](#)).
- Fix very rare race condition in `SYSTEM SYNC REPLICA`. If the replicated table is created and at the same time from the separate connection another client is issuing `SYSTEM SYNC REPLICA` command on that table (this is unlikely, because another client should be aware that the table is created), it's possible to get `nullptr` dereference. [#11691](#) ([alexey-milovidov](#)).
- Pass proper timeouts when communicating with XDBC bridge. Recently timeouts were not respected when checking bridge liveness and receiving meta info. [#11690](#) ([alexey-milovidov](#)).
- Fix `LIMIT n WITH TIES` usage together with `ORDER BY` statement, which contains aliases. [#11689](#) ([Anton Popov](#)).
- Fix possible Pipeline stuck for selects with parallel `FINAL`. Fixes [#11636](#). [#11682](#) ([Nikolai Kochetov](#)).
- Fix error which leads to an incorrect state of `system.mutations`. It may show that whole mutation is already done but the server still has `MUTATE_PART` tasks in the replication queue and tries to execute them. This fixes [#11611](#). [#11681](#) ([alesapin](#)).
- Fix syntax hilite in `CREATE USER` query. [#11664](#) ([alexey-milovidov](#)).

- Add support for regular expressions with case-insensitive flags. This fixes #11101 and fixes #11506. #11649 (alexey-milovidov).
- Remove trivial count query optimization if row-level security is set. In previous versions the user get total count of records in a table instead filtered. This fixes #11352. #11644 (alexey-milovidov).
- Fix bloom filters for String (data skipping indices). #11638 (Azat Khuzhin).
- Without -q option the database does not get created at startup. #11604 (giordyb).
- Fix error `Block structure mismatch` for queries with sampling reading from `Buffer` table. #11602 (Nikolai Kochetov).
- Fix wrong exit code of the clickhouse-client, when `exception.code() % 256 == 0`. #11601 (filimonov).
- Fix race conditions in CREATE/DROP of different replicas of ReplicatedMergeTree. Continue to work if the table was not removed completely from ZooKeeper or not created successfully. This fixes #11432. #11592 (alexey-milovidov).
- Fix trivial error in log message about "Mark cache size was lowered" at server startup. This closes #11399. #11589 (alexey-milovidov).
- Fix error `Size of offsets does not match size of column` for queries with `PREWHERE` column in (subquery) and `ARRAY JOIN`. #11580 (Nikolai Kochetov).
- Fixed rare segfault in `SHOW CREATE TABLE` Fixes #11490. #11579 (tavplubix).
- All queries in HTTP session have had the same `query_id`. It is fixed. #11578 (tavplubix).
- Now clickhouse-server docker container will prefer IPv6 checking server aliveness. #11550 (Ivan Starkov).
- Fix the error `Data compressed with different methods` that can happen if `min_bytes_to_use_direct_io` is enabled and `PREWHERE` is active and using SAMPLE or high number of threads. This fixes #11539. #11540 (alexey-milovidov).
- Fix shard\_num/replica\_num for `<node>` (breaks `use_compact_format_in_distributed_parts_names`). #11528 (Azat Khuzhin).
- Fix async INSERT into Distributed for `prefer_localhost_replica=0` and w/o internal\_replication. #11527 (Azat Khuzhin).
- Fix memory leak when exception is thrown in the middle of aggregation with `-State` functions. This fixes #8995. #11496 (alexey-milovidov).
- Fix Pipeline stuck exception for `INSERT SELECT FINAL` where `SELECT (max_threads>1)` has multiple streams but `INSERT` has only one (`max_insert_threads==0`). #11455 (Azat Khuzhin).
- Fix wrong result in queries like `select count() from t, u`. #11454 (Artem Zuikov).
- Fix return compressed size for codecs. #11448 (Nikolai Kochetov).
- Fix server crash when a column has compression codec with non-literal arguments. Fixes #11365. #11431 (alesapin).
- Fix potential uninitialized memory read in MergeTree shutdown if table was not created successfully. #11420 (alexey-milovidov).
- Fix crash in JOIN over `LowCardinality(T)` and `Nullable(T)`. #11380. #11414 (Artem Zuikov).
- Fix error code for wrong `USING` key. #11373. #11404 (Artem Zuikov).

- Fixed `geohashesInBox` with arguments outside of latitude/longitude range. #11403 (Vasily Nemkov).
- Better errors for `joinGet()` functions. #11389 (Artem Zuikov).
- Fix possible Pipeline stuck error for queries with external sort and limit. Fixes #11359. #11366 (Nikolai Kochetov).
- Remove redundant lock during parts send in ReplicatedMergeTree. #11354 (alesapin).
- Fix support for `\G` (vertical output) in clickhouse-client in multiline mode. This closes #9933. #11350 (alexey-milovidov).
- Fix potential segfault when using `Lazy` database. #11348 (alexey-milovidov).
- Fix crash in direct selects from `Join` table engine (without JOIN) and wrong nullability. #11340 (Artem Zuikov).
- Fix crash in `quantilesExactWeightedArray`. #11337 (Nikolai Kochetov).
- Now merges stopped before change metadata in `ALTER` queries. #11335 (alesapin).
- Make writing to `MATERIALIZED VIEW` with setting `parallel_view_processing = 1` parallel again. Fixes #10241. #11330 (Nikolai Kochetov).
- Fix `visitParamExtractRaw` when extracted JSON has strings with unbalanced { or [. #11318 (Ewout).
- Fix very rare race condition in ThreadPool. #11314 (alexey-milovidov).
- Fix insignificant data race in `clickhouse-copier`. Found by integration tests. #11313 (alexey-milovidov).
- Fix potential uninitialized memory in conversion. Example: `SELECT toIntervalSecond(now64())`. #11311 (alexey-milovidov).
- Fix the issue when index analysis cannot work if a table has Array column in primary key and if a query is filtering by this column with `empty` or `notEmpty` functions. This fixes #11286. #11303 (alexey-milovidov).
- Fix bug when query speed estimation can be incorrect and the limit of `min_execution_speed` may not work or work incorrectly if the query is throttled by `max_network_bandwidth`, `max_execution_speed` or `priority` settings. Change the default value of `timeout_before_checking_execution_speed` to non-zero, because otherwise the settings `min_execution_speed` and `max_execution_speed` have no effect. This fixes #11297. This fixes #5732. This fixes #6228. Usability improvement: avoid concatenation of exception message with progress bar in clickhouse-client. #11296 (alexey-milovidov).
- Fix crash when `SET DEFAULT ROLE` is called with wrong arguments. This fixes #10586. #11278 (Vitaly Baranov).
- Fix crash while reading malformed data in Protobuf format. This fixes #5957, fixes #11203. #11258 (Vitaly Baranov).
- Fixed a bug when `cache dictionary` could return default value instead of normal (when there are only expired keys). This affects only string fields. #11233 (Nikita Mikhaylov).
- Fix error `Block structure mismatch` in `QueryPipeline` while reading from `VIEW` with constants in inner query. Fixes #11181. #11205 (Nikolai Kochetov).
- Fix possible exception `Invalid status for associated output` #11200 (Nikolai Kochetov).
- Now `primary.idx` will be checked if it's defined in `CREATE` query. #11199 (alesapin).

- Fix possible error `Cannot capture column` for higher-order functions with `Array(Array(LowCardinality))` captured argument. [#11185 \(Nikolai Kochetov\)](#).
- Fixed S3 globbing which could fail in case of more than 1000 keys and some backends. [#11179 \(Vladimir Chebotarev\)](#).
- If data skipping index is dependent on columns that are going to be modified during background merge (for SummingMergeTree, AggregatingMergeTree as well as for TTL GROUP BY), it was calculated incorrectly. This issue is fixed by moving index calculation after merge so the index is calculated on merged data. [#11162 \(Azat Khuzhin\)](#).
- Fix for the hang which was happening sometimes during DROP of table engine=Kafka (or during server restarts). [#11145 \(filimonov\)](#).
- Fix excessive reserving of threads for simple queries (optimization for reducing the number of threads, which was partly broken after changes in pipeline). [#11114 \(Azat Khuzhin\)](#).
- Remove logging from mutation finalization task if nothing was finalized. [#11109 \(alesapin\)](#).
- Fixed deadlock during server startup after update with changes in structure of system log tables. [#11106 \(alesapin\)](#).
- Fixed memory leak in `registerDiskS3`. [#11074 \(Pavel Kovalenko\)](#).
- Fix error `No such name in Block::erase()` when JOIN appears with PREWHERE or `optimize_move_to_prewhere` makes PREWHERE from WHERE. [#11051 \(Artem Zuikov\)](#).
- Fixes the potential missed data during termination of Kafka engine table. [#11048 \(filimonov\)](#).
- Fixed `parseDateTime64BestEffort` argument resolution bugs. [#10925](#). [#11038 \(Vasily Nemkov\)](#).
- Now it's possible to `ADD/DROP` and `RENAME` the same one column in a single `ALTER` query. Exception message for simultaneous `MODIFY` and `RENAME` became more clear. Partially fixes [#10669](#). [#11037 \(alesapin\)](#).
- Fixed parsing of S3 URLs. [#11036 \(Vladimir Chebotarev\)](#).
- Fix memory tracking for two-level `GROUP BY` when there is a `LIMIT`. [#11022 \(Azat Khuzhin\)](#).
- Fix very rare potential use-after-free error in MergeTree if table was not created successfully. [#10986 \(alexey-milovidov\)](#).
- Fix metadata (relative path for rename) and data (relative path for symlink) handling for Atomic database. [#10980 \(Azat Khuzhin\)](#).
- Fix server crash on concurrent `ALTER` and `DROP DATABASE` queries with Atomic database engine. [#10968 \(tavplubix\)](#).
- Fix incorrect raw data size in method `getRawData()`. [#10964 \(lgr\)](#).
- Fix incompatibility of two-level aggregation between versions 20.1 and earlier. This incompatibility happens when different versions of ClickHouse are used on initiator node and remote nodes and the size of GROUP BY result is large and aggregation is performed by a single String field. It leads to several unmerged rows for a single key in result. [#10952 \(alexey-milovidov\)](#).
- Avoid sending partially written files by the `DistributedBlockOutputStream`. [#10940 \(Azat Khuzhin\)](#).
- Fix crash in `SELECT count(notNullIn(NULL, []))`. [#10920 \(Nikolai Kochetov\)](#).
- Fix for the hang which was happening sometimes during DROP of table engine=Kafka (or during server restarts). [#10910 \(filimonov\)](#).

- Now it's possible to execute multiple `ALTER RENAME` like `a TO b, c TO a`. #10895 (alesapin).
- Fix possible race which could happen when you get result from aggregate function state from multiple thread for the same column. The only way (which I found) it can happen is when you use `finalizeAggregation` function while reading from table with `Memory` engine which stores `AggregateFunction` state for `quanite*` function. #10890 (Nikolai Kochetov).
- Fix backward compatibility with tuples in Distributed tables. #10889 (Anton Popov).
- Fix SIGSEGV in `StringHashTable` (if such key does not exist). #10870 (Azat Khuzhin).
- Fixed `WATCH` hangs after `LiveView` table was dropped from database with `Atomic` engine. #10859 (tavplubix).
- Fixed bug in `ReplicatedMergeTree` which might cause some `ALTER` on `OPTIMIZE` query to hang waiting for some replica after it become inactive. #10849 (tavplubix).
- Now constraints are updated if the column participating in `CONSTRAINT` expression was renamed. Fixes #10844. #10847 (alesapin).
- Fix potential read of uninitialized memory in cache dictionary. #10834 (alexey-milovidov).
- Fix columns order after `Block::sortColumns()` (also add a test that shows that it affects some real use case - Buffer engine). #10826 (Azat Khuzhin).
- Fix the issue with ODBC bridge when no quoting of identifiers is requested. This fixes #7984. #10821 (alexey-milovidov).
- Fix UBSan and MSan report in DateLUT. #10798 (alexey-milovidov).
- Make use of `src_type` for correct type conversion in key conditions. Fixes #6287. #10791 (Andrew Onyshchuk).
- Get rid of old libunwind patches. <https://github.com/ClickHouse-Extras/libunwind/commit/500aa227911bd185a94bfc071d68f4d3b03cb3b1#r39048012> This allows to disable `-fno-omit-frame-pointer` in `clang` builds that improves performance at least by 1% in average. #10761 (Amos Bird).
- Fix avgWeighted when using floating-point weight over multiple shards. #10758 (Baudouin Giard).
- Fix `parallel_view_processing` behavior. Now all insertions into `MATERIALIZED VIEW` without exception should be finished if exception happened. Fixes #10241. #10757 (Nikolai Kochetov).
- Fix combinator `-OrNull` and `-OrDefault` when combined with `-State`. #10741 (hcz).
- Fix crash in `generateRandom` with nested types. Fixes #10583. #10734 (Nikolai Kochetov).
- Fix data corruption for `LowCardinality(FixedString)` key column in `SummingMergeTree` which could have happened after merge. Fixes #10489. #10721 (Nikolai Kochetov).
- Fix usage of primary key wrapped into a function with 'FINAL' modifier and 'ORDER BY' optimization. #10715 (Anton Popov).
- Fix possible buffer overflow in function `h3EdgeAngle`. #10711 (alexey-milovidov).
- Fix disappearing totals. Totals could have been filtered if query had had join or subquery with external where condition. Fixes #10674. #10698 (Nikolai Kochetov).
- Fix atomicity of HTTP insert. This fixes #9666. #10687 (Andrew Onyshchuk).
- Fix multiple usages of `IN` operator with the identical set in one query. #10686 (Anton Popov).

- Fixed bug, which causes http requests stuck on client close when `readonly=2` and `cancel_http_READONLY_queries_on_client_close=1`. Fixes #7939, #7019, #7736, #7091. #10684 (tavplubix).
- Fix order of parameters in `AggregateTransform` constructor. #10667 (palasonic1).
- Fix the lack of parallel execution of remote queries with `distributed_aggregation_memory_efficient` enabled. Fixes #10655. #10664 (Nikolai Kochetov).
- Fix possible incorrect number of rows for queries with `LIMIT`. Fixes #10566, #10709. #10660 (Nikolai Kochetov).
- Fix bug which locks concurrent alters when table has a lot of parts. #10659 (alesapin).
- Fix nullptr dereference in `StorageBuffer` if server was shutdown before table startup. #10641 (alexey-milovidov).
- Fix predicates optimization for distributed queries (`enable_optimize_predicate_expression=1`) for queries with `HAVING` section (i.e. when filtering on the server initiator is required), by preserving the order of expressions (and this is enough to fix), and also force aggregator use column names over indexes. Fixes: #10613, #11413. #10621 (Azat Khuzhin).
- Fix `optimize_skip_unused_shards` with `LowCardinality`. #10611 (Azat Khuzhin).
- Fix segfault in `StorageBuffer` when exception on server startup. Fixes #10550. #10609 (tavplubix).
- On `SYSTEM DROP DNS CACHE` query also drop caches, which are used to check if user is allowed to connect from some IP addresses. #10608 (tavplubix).
- Fixed incorrect scalar results inside inner query of `MATERIALIZED VIEW` in case if this query contained dependent table. #10603 (Nikolai Kochetov).
- Fixed handling condition variable for synchronous mutations. In some cases signals to that condition variable could be lost. #10588 (Vladimir Chebotarev).
- Fixes possible crash `createDictionary()` is called before `loadStoredObject()` has finished. #10587 (Vitaly Baranov).
- Fix error `the BloomFilter false positive must be a double number between 0 and 1` #10551. #10569 (Winter Zhang).
- Fix `SELECT` of column `ALIAS` which default expression type different from column type. #10563 (Azat Khuzhin).
- Implemented comparison between `DateTime64` and `String` values (just like for `DateTime`). #10560 (Vasily Nemkov).
- Fix index corruption, which may occur in some cases after merge compact parts into another compact part. #10531 (Anton Popov).
- Disable `GROUP BY sharding_key` optimization by default (`optimize_distributed_group_by_sharding_key` had been introduced and turned off by default, due to trickery of `sharding_key` analyzing, simple example is `if` in `sharding key`) and fix it for `WITH ROLLUP/CUBE/TOTALS`. #10516 (Azat Khuzhin).
- Fixes: #10263 (after that PR dist send via `INSERT` had been postponing on each `INSERT`) Fixes: #8756 (that PR breaks distributed sends with all of the following conditions met (unlikely setup for now I guess): `internal_replication == false`, multiple local shards (activates the hardlinking code) and `distributed_storage_policy` (makes `link(2)` fails on `EXDEV`)). #10486 (Azat Khuzhin).
- Fixed error with "max\_rows\_to\_sort" limit. #10268 (alexey-milovidov).

- Get dictionary and check access rights only once per each call of any function reading external dictionaries. #10928 ([Vitaly Baranov](#)).

## Improvement

- Apply TTL for old data, after `ALTER MODIFY TTL` query. This behaviour is controlled by setting `materialize_ttl_after_modify`, which is enabled by default. #11042 ([Anton Popov](#)).
- When parsing C-style backslash escapes in string literals, VALUES and various text formats (this is an extension to SQL standard that is endemic for ClickHouse and MySQL), keep backslash if unknown escape sequence is found (e.g. `\%` or `\w`) that will make usage of LIKE and match regular expressions more convenient (it's enough to write `name LIKE 'used\_cars'` instead of `name LIKE 'used\\_cars'`) and more compatible at the same time. This fixes #10922. #11208 ([alexey-milovidov](#)).
- When reading Decimal value, cut extra digits after point. This behaviour is more compatible with MySQL and PostgreSQL. This fixes #10202. #11831 ([alexey-milovidov](#)).
- Allow to DROP replicated table if the metadata in ZooKeeper was already removed and does not exist (this is also the case when using TestKeeper for testing and the server was restarted). Allow to RENAME replicated table even if there is an error communicating with ZooKeeper. This fixes #10720. #11652 ([alexey-milovidov](#)).
- Slightly improve diagnostic of reading decimal from string. This closes #10202. #11829 ([alexey-milovidov](#)).
- Fix sleep invocation in signal handler. It was sleeping for less amount of time than expected. #11825 ([alexey-milovidov](#)).
- (Only Linux) OS related performance metrics (for CPU and I/O) will work even without `CAP_NET_ADMIN` capability. #10544 ([Alexander Kazakov](#)).
- Added hostname as an alias to function `hostName`. This feature was suggested by Victor Tarnavskiy from Yandex.Metrica. #11821 ([alexey-milovidov](#)).
- Added support for distributed `DDL` (update/delete/drop partition) on cross replication clusters. #11703 ([Nikita Mikhaylov](#)).
- Emit warning instead of error in server log at startup if we cannot listen one of the listen addresses (e.g. IPv6 is unavailable inside Docker). Note that if server fails to listen all listed addresses, it will refuse to startup as before. This fixes #4406. #11687 ([alexey-milovidov](#)).
- Default user and database creation on docker image starting. #10637 ([Paramtamtam](#)).
- When multiline query is printed to server log, the lines are joined. Make it to work correct in case of multiline string literals, identifiers and single-line comments. This fixes #3853. #11686 ([alexey-milovidov](#)).
- Multiple names are now allowed in commands: CREATE USER, CREATE ROLE, ALTER USER, SHOW CREATE USER, SHOW GRANTS and so on. #11670 ([Vitaly Baranov](#)).
- Add support for distributed DDL (UPDATE/DELETE/DROP PARTITION) on cross replication clusters. #11508 ([frank lee](#)).
- Clear password from command line in `clickhouse-client` and `clickhouse-benchmark` if the user has specified it with explicit value. This prevents password exposure by `ps` and similar tools. #11665 ([alexey-milovidov](#)).
- Don't use debug info from ELF file if it does not correspond to the running binary. It is needed to avoid printing wrong function names and source locations in stack traces. This fixes #7514. #11657 ([alexey-milovidov](#)).

- Return NULL/zero when value is not parsed completely in `parseDateTimeBestEffortOrNull`/Zero functions. This fixes #7876. #11653 (alexey-milovidov).
- Skip empty parameters in requested URL. They may appear when you write `http://localhost:8123/?&a=b` or `http://localhost:8123/?a=b&&c=d`. This closes #10749. #11651 (alexey-milovidov).
- Allow using `groupArrayArray` and `groupUniqArrayArray` as `SimpleAggregateFunction`. #11650 (Volodymyr Kuznetsov).
- Allow comparison with constant strings by implicit conversions when analysing index conditions on other types. This may close #11630. #11648 (alexey-milovidov).
- <https://github.com/ClickHouse/ClickHouse/pull/7572#issuecomment-642815377> Support config default HTTPHandlers. #11628 (Winter Zhang).
- Make more input formats to work with Kafka engine. Fix the issue with premature flushes. Fix the performance issue when `kafka_num_consumers` is greater than number of partitions in topic. #11599 (filimonov).
- Improve `multiple_joins_rewriter_version=2` logic. Fix unknown columns error for lambda aliases. #11587 (Artem Zuikov).
- Better exception message when cannot parse columns declaration list. This closes #10403. #11537 (alexey-milovidov).
- Improve `enable_optimize_predicate_expression=1` logic for VIEW. #11513 (Artem Zuikov).
- Adding support for PREWHERE in live view tables. #11495 (vzakaznikov).
- Automatically update DNS cache, which is used to check if user is allowed to connect from an address. #11487 (tavplubix).
- OPTIMIZE FINAL will force merge even if concurrent merges are performed. This closes #11309 and closes #11322. #11346 (alexey-milovidov).
- Suppress output of cancelled queries in clickhouse-client. In previous versions result may continue to print in terminal even after you press Ctrl+C to cancel query. This closes #9473. #11342 (alexey-milovidov).
- Now history file is updated after each query and there is no race condition if multiple clients use one history file. This fixes #9897. #11453 (Tagir Kuskarov).
- Better log messages in while reloading configuration. #11341 (alexey-milovidov).
- Remove trailing whitespaces from formatted queries in `clickhouse-client` or `clickhouse-format` in some cases. #11325 (alexey-milovidov).
- Add setting "output\_format\_pretty\_max\_value\_width". If value is longer, it will be cut to avoid output of too large values in terminal. This closes #11140. #11324 (alexey-milovidov).
- Better exception message in case when there is shortage of memory mappings. This closes #11027. #11316 (alexey-milovidov).
- Support (U)Int8, (U)Int16, Date in ASOF JOIN. #11301 (Artem Zuikov).
- Support `kafka_client_id` parameter for Kafka tables. It also changes the default `client.id` used by ClickHouse when communicating with Kafka to be more verbose and usable. #11252 (filimonov).
- Keep the value of `DistributedFilesToInsert` metric on exceptions. In previous versions, the value was set when we are going to send some files, but it is zero, if there was an exception and some files are still pending. Now it corresponds to the number of pending files in filesystem. #11220 (alexey-milovidov).

- Add support for multi-word data type names (such as `DOUBLE PRECISION` and `CHAR VARYING`) for better SQL compatibility. #11214 (Павел Потемкин).
- Provide synonyms for some data types. #10856 (Павел Потемкин).
- The query log is now enabled by default. #11184 (Ivan Blinkov).
- Show authentication type in table `system.users` and while executing `SHOW CREATE USER` query. #11080 (Vitaly Baranov).
- Remove data on explicit `DROP DATABASE` for `Memory` database engine. Fixes #10557. #11021 (tavplubix).
- Set thread names for internal threads of rdkafka library. Make logs from rdkafka available in server logs. #10983 (Azat Khuzhin).
- Support for unicode whitespaces in queries. This helps when queries are copy-pasted from Word or from web page. This fixes #10896. #10903 (alexey-milovidov).
- Allow large `UInt` types as the index in function `tupleElement`. #10874 (hcz).
- Respect `prefer_localhost_replica/load_balancing` on `INSERT` into `Distributed`. #10867 (Azat Khuzhin).
- Introduce `min_insert_block_size_rows_for_materialized_views`, `min_insert_block_size_bytes_for_materialized_views` settings. These settings are similar to `min_insert_block_size_rows` and `min_insert_block_size_bytes`, but applied only for blocks inserted into `MATERIALIZED VIEW`. It helps to control blocks squashing while pushing to MVs and avoid excessive memory usage. #10858 (Azat Khuzhin).
- Get rid of exception from replicated queue during server shutdown. Fixes #10819. #10841 (alesapin).
- Ensure that `varSamp`, `varPop` cannot return negative results due to numerical errors and that `stddevSamp`, `stddevPop` cannot be calculated from negative variance. This fixes #10532. #10829 (alexey-milovidov).
- Better DNS exception message. This fixes #10813. #10828 (alexey-milovidov).
- Change HTTP response code in case of some parse errors to 400 Bad Request. This fix #10636. #10640 (alexey-milovidov).
- Print a message if `clickhouse-client` is newer than `clickhouse-server`. #10627 (alexey-milovidov).
- Adding support for `INSERT INTO [db.]table WATCH` query. #10498 (vzakaznikov).
- Allow to pass `quota_key` in `clickhouse-client`. This closes #10227. #10270 (alexey-milovidov).

## Performance Improvement

- Allow multiple replicas to assign merges, mutations, partition drop, move and replace concurrently. This closes #10367. #11639 (alexey-milovidov) #11795 (alexey-milovidov).
- Optimization of GROUP BY with respect to table sorting key, enabled with `optimize_aggregation_in_order` setting. #9113 (dimarub2000).
- Selects with final are executed in parallel. Added setting `max_final_threads` to limit the number of threads used. #10463 (Nikolai Kochetov).
- Improve performance for `INSERT` queries via `INSERT SELECT` or `INSERT` with `clickhouse-client` when small blocks are generated (typical case with parallel parsing). This fixes #11275. Fix the issue that `CONSTRAINTs` were not working for `DEFAULT` fields. This fixes #11273. Fix the issue that `CONSTRAINTs` were ignored for `TEMPORARY` tables. This fixes #11274. #11276 (alexey-milovidov).
- Optimization that eliminates min/max/any aggregators of GROUP BY keys in SELECT section, enabled with `optimize_aggregators_of_group_by_keys` setting. #11667 (xPoSx). #11806 (Azat Khuzhin).

- New optimization that takes all operations out of `any` function, enabled with `optimize_move_functions_out_of_any #11529` ([Ruslan](#)).
- Improve performance of `clickhouse-client` in interactive mode when Pretty formats are used. In previous versions, significant amount of time can be spent calculating visible width of UTF-8 string. This closes [#11323](#). [#11323](#) ([alexey-milovidov](#)).
- Improved performance for queries with `ORDER BY` and small `LIMIT` (less, then `max_block_size`). [#11171](#) ([Albert Kidrachev](#)).
- Add runtime CPU detection to select and dispatch the best function implementation. Add support for codegeneration for multiple targets. This closes [#1017](#). [#10058](#) ([DimasKovas](#)).
- Enable `mlock` of clickhouse binary by default. It will prevent clickhouse executable from being paged out under high IO load. [#11139](#) ([alexey-milovidov](#)).
- Make queries with `sum` aggregate function and without GROUP BY keys to run multiple times faster. [#10992](#) ([alexey-milovidov](#)).
- Improving radix sort (used in `ORDER BY` with simple keys) by removing some redundant data moves. [#10981](#) ([Arslan Gumerov](#)).
- Sort bigger parts of the left table in MergeJoin. Buffer left blocks in memory. Add `partial_merge_join_left_table_buffer_bytes` setting to manage the left blocks buffers sizes. [#10601](#) ([Artem Zuikov](#)).
- Remove duplicate `ORDER BY` and `DISTINCT` from subqueries, this optimization is enabled with `optimize_duplicate_order_by_and_distinct #10067` ([Mikhail Malafeev](#)).
- This feature eliminates functions of other keys in GROUP BY section, enabled with `optimize_group_by_function_keys #10051` ([xPoSx](#)).
- New optimization that takes arithmetic operations out of aggregate functions, enabled with `optimize_arithmetic_operations_in_aggregate_functions #10047` ([Ruslan](#)).
- Use HTTP client for S3 based on Poco instead of curl. This will improve performance and lower memory usage of s3 storage and table functions. [#11230](#) ([Pavel Kovalenko](#)).
- Fix Kafka performance issue related to reschedules based on limits, which were always applied. [#11149](#) ([filimonov](#)).
- Enable `percpu_arena:percpu` for jemalloc (This will reduce memory fragmentation due to thread pool). [#11084](#) ([Azat Khuzhin](#)).
- Optimize memory usage when reading a response from an S3 HTTP client. [#11561](#) ([Pavel Kovalenko](#)).
- Adjust the default Kafka settings for better performance. [#11388](#) ([filimonov](#)).

## Experimental Feature

- Add data type `Point` (`Tuple(Float64, Float64)`) and `Polygon` (`Array(Array(Tuple(Float64, Float64)))`). [#10678](#) ([Alexey Ilyukhov](#)).
- Add's a `hasSubstr` function that allows for look for subsequences in arrays. Note: this function is likely to be renamed without further notice. [#11071](#) ([Ryad Zenine](#)).

- Added OpenCL support and bitonic sort algorithm, which can be used for sorting integer types of data in single column. Needs to be build with flag `-DENABLE_OPENCL=1`. For using bitonic sort algorithm instead of others you need to set `bitonic_sort` for Setting's option `special_sort` and make sure that OpenCL is available. This feature does not improve performance or anything else, it is only provided as an example and for demonstration purposes. It is likely to be removed in near future if there will be no further development in this direction. [#10232 \(Ri\)](#).

## Build/Testing/Packaging Improvement

- Enable clang-tidy for programs and utils. [#10991 \(alexey-milovidov\)](#).
- Remove dependency on `tzdata`: do not fail if `/usr/share/zoneinfo` directory does not exist. Note that all timezones work in ClickHouse even without tzdata installed in system. [#11827 \(alexey-milovidov\)](#).
- Added MSan and UBSan stress tests. Note that we already have MSan, UBSan for functional tests and "stress" test is another kind of tests. [#10871 \(alexey-milovidov\)](#).
- Print compiler build id in crash messages. It will make us slightly more certain about what binary has crashed. Added new function `buildId`. [#11824 \(alexey-milovidov\)](#).
- Added a test to ensure that mutations continue to work after FREEZE query. [#11820 \(alexey-milovidov\)](#).
- Don't allow tests with "fail" substring in their names because it makes looking at the tests results in browser less convenient when you type Ctrl+F and search for "fail". [#11817 \(alexey-milovidov\)](#).
- Removes unused imports from `HTTPHandlerFactory`. [#11660 \(Bharat Nallan\)](#).
- Added a random sampling of instances where copier is executed. It is needed to avoid `Too many simultaneous queries` error. Also increased timeout and decreased fault probability. [#11573 \(Nikita Mikhaylov\)](#).
- Fix missed include. [#11525 \(Matvey V. Kornilov\)](#).
- Speed up build by removing old example programs. Also found some orphan functional test. [#11486 \(alexey-milovidov\)](#).
- Increase ccache size for builds in CI. [#11450 \(alesapin\)](#).
- Leave only `unit_tests_dbms` in deb build. [#11429 \(Ilya Yatsishin\)](#).
- Update librdkafka to version [1.4.2](#). [#11256 \(filimonov\)](#).
- Refactor CMake build files. [#11390 \(Ivan\)](#).
- Fix several flaky integration tests. [#11355 \(alesapin\)](#).
- Add support for unit tests run with UBSan. [#11345 \(alexey-milovidov\)](#).
- Remove redundant timeout from integration test `test_insertion_sync_fails_with_timeout`. [#11343 \(alesapin\)](#).
- Better check for hung queries in `clickhouse-test`. [#11321 \(alexey-milovidov\)](#).
- Emit a warning if server was build in debug or with sanitizers. [#11304 \(alexey-milovidov\)](#).
- Now `clickhouse-test` check the server aliveness before tests run. [#11285 \(alesapin\)](#).
- Fix potentially flacky test `00731_long_merge_tree_select_opened_files.sh`. It does not fail frequently but we have discovered potential race condition in this test while experimenting with ThreadFuzzer: [#9814](#) See link for the example. [#11270 \(alexey-milovidov\)](#).
- Repeat test in CI if `curl` invocation was timed out. It is possible due to system hangups for 10+ seconds that are typical in our CI infrastructure. This fixes [#11267](#). [#11268 \(alexey-milovidov\)](#).

- Add a test for Join table engine from @donmikel. This closes #9158. #11265 (alexey-milovidov).
- Fix several non significant errors in unit tests. #11262 (alesapin).
- Now parts of linker command for cctz library will not be shuffled with other libraries. #11213 (alesapin).
- Split /programs/server into actual program and library. #11186 (Ivan).
- Improve build scripts for protobuf & gRPC. #11172 (Vitaly Baranov).
- Enable performance test that was not working. #11158 (alexey-milovidov).
- Create root S3 bucket for tests before any CH instance is started. #11142 (Pavel Kovalenko).
- Add performance test for non-constant polygons. #11141 (alexey-milovidov).
- Fixing 00979\_live\_view\_watch\_continuous\_aggregates test. #11024 (vzakaznikov).
- Add ability to run zookeeper in integration tests over tmpfs. #11002 (alesapin).
- Wait for odbc-bridge with exponential backoff. Previous wait time of 200 ms was not enough in our CI environment. #10990 (alexey-milovidov).
- Fix non-deterministic test. #10989 (alexey-milovidov).
- Added a test for empty external data. #10926 (alexey-milovidov).
- Database is recreated for every test. This improves separation of tests. #10902 (alexey-milovidov).
- Added more asserts in columns code. #10833 (alexey-milovidov).
- Better cooperation with sanitizers. Print information about query\_id in the message of sanitizer failure. #10832 (alexey-milovidov).
- Fix obvious race condition in "Split build smoke test" check. #10820 (alexey-milovidov).
- Fix (false) MSan report in MergeTreeIndexFullText. The issue first appeared in #9968. #10801 (alexey-milovidov).
- Add MSan suppression for MariaDB Client library. #10800 (alexey-milovidov).
- GRPC make couldn't find protobuf files, changed make file by adding the right link. #10794 (mnkonkova).
- Enable extra warnings (-Weverything) for base, utils, programs. Note that we already have it for the most of the code. #10779 (alexey-milovidov).
- Suppressions of warnings from libraries was mistakenly declared as public in #10396. #10776 (alexey-milovidov).
- Restore a patch that was accidentally deleted in #10396. #10774 (alexey-milovidov).
- Fix performance tests errors, part 2. #10773 (alexey-milovidov).
- Fix performance test errors. #10766 (alexey-milovidov).
- Update cross-builds to use clang-10 compiler. #10724 (Ivan).
- Update instruction to install RPM packages. This was suggested by Denis (TG login @ldviolet) and implemented by Arkady Shejn. #10707 (alexey-milovidov).
- Trying to fix tests/queries/0\_stateless/01246\_insert\_into\_watch\_live\_view.py test. #10670 (vzakaznikov).

- Fixing and re-enabling 00979\_live\_view\_watch\_continuous\_aggregates.py test. #10658 (vzakaznikov).
- Fix OOM in ASan stress test. #10646 (alexey-milovidov).
- Fix UBSan report (adding zero to nullptr) in HashTable that appeared after migration to clang-10. #10638 (alexey-milovidov).
- Remove external call to `ld` (bfd) linker during tzdata processing in compile time. #10634 (alesapin).
- Allow to use `lld` to link blobs (resources). #10632 (alexey-milovidov).
- Fix UBSan report in LZ4 library. #10631 (alexey-milovidov). See also <https://github.com/lz4/lz4/issues/857>
- Update LZ4 to the latest dev branch. #10630 (alexey-milovidov).
- Added auto-generated machine-readable file with the list of stable versions. #10628 (alexey-milovidov).
- Fix `capnproto` version check for `capnp::UnalignedFlatArrayMessageReader`. #10618 (Matvey V. Kornilov).
- Lower memory usage in tests. #10617 (alexey-milovidov).
- Fixing hard coded timeouts in new live view tests. #10604 (vzakaznikov).
- Increasing timeout when opening a client in `tests/queries/0_stateless/helpers/client.py`. #10599 (vzakaznikov).
- Enable ThinLTO for clang builds, continuation of #10435. #10585 (Amos Bird).
- Adding fuzzers and preparing for oss-fuzz integration. #10546 (kyprizel).
- Fix FreeBSD build. #10150 (Ivan).
- Add new build for query tests using pytest framework. #10039 (Ivan).

## ClickHouse release v20.4

### ClickHouse release v20.4.8.99-stable 2020-08-10

#### Bug Fix

- Fixed error in `parseDateTimeBestEffort` function when unix timestamp was passed as an argument. This fixes #13362. #13441 (alexey-milovidov).
- Fixed potentially low performance and slightly incorrect result for `uniqExact`, `topK`, `sumDistinct` and similar aggregate functions called on `Float` types with `Nan` values. It also triggered assert in debug build. This fixes #12491. #13254 (alexey-milovidov).
- Fixed function `if` with nullable `constexpr` as cond that is not literal `NULL`. Fixes #12463. #13226 (alexey-milovidov).
- Fixed assert in `arrayElement` function in case of array elements are `Nullable` and array subscript is also `Nullable`. This fixes #12172. #13224 (alexey-milovidov).
- Fixed wrong index analysis with functions. It could lead to pruning wrong parts, while reading from `MergeTree` tables. Fixes #13060. Fixes #12406. #13081 (Anton Popov).
- Fixed unnecessary limiting for the number of threads for selects from local replica. #12840 (Nikolai Kochetov).
- Fixed possible extra overflow row in data which could appear for queries `WITH TOTALS`. #12747 (Nikolai Kochetov).

- Fixed performance with large tuples, which are interpreted as functions in `IN` section. The case when user write `WHERE x IN tuple(1, 2, ...)` instead of `WHERE x IN (1, 2, ...)` for some obscure reason. #12700 (Anton Popov).
- Fixed memory tracking for `input_format_parallel_parsing` (by attaching thread to group). #12672 (Azat Khuzhin).
- Fixed #12293 allow push predicate when subquery contains with clause. #12663 (Winter Zhang).
- Fixed #10572 fix bloom filter index with const expression. #12659 (Winter Zhang).
- Fixed `SIGSEGV` in `StorageKafka` when broker is unavailable (and not only). #12658 (Azat Khuzhin).
- Added support for function `if` with `Array(UUID)` arguments. This fixes #11066. #12648 (alexey-milovidov).
- Fixed race condition in external dictionaries with cache layout which can lead server crash. #12566 (alesapin).
- Removed data for Distributed tables (blocks from async INSERTs) on `DROP TABLE`. #12556 (Azat Khuzhin).
- Fixed bug which lead to broken old parts after `ALTER DELETE` query when `enable_mixed_granularity_parts=1`. Fixes #12536. #12543 (alesapin).
- Better exception for function `in` with invalid number of arguments. #12529 (Anton Popov).
- Fixed performance issue, while reading from compact parts. #12492 (Anton Popov).
- Fixed crash in JOIN with dictionary when we are joining over expression of dictionary key: `t JOIN dict ON expr(dict.id) = t.id`. Disable dictionary join optimisation for this case. #12458 (Artem Zuikov).
- Fixed possible segfault if StorageMerge. Closes #12054. #12401 (tavplubix).
- Fixed order of columns in `WITH FILL` modifier. Previously order of columns of `ORDER BY` statement wasn't respected. #12306 (Anton Popov).
- Avoid "bad cast" exception when there is an expression that filters data by virtual columns (like `_table` in Merge tables) or by "index" columns in system tables such as filtering by database name when querying from `system.tables`, and this expression returns `Nullable` type. This fixes #12166. #12305 (alexey-milovidov).
- Show error after TrieDictionary failed to load. #12290 (Vitaly Baranov).
- The function `arrayFill` worked incorrectly for empty arrays that may lead to crash. This fixes #12263. #12279 (alexey-milovidov).
- Implemented conversions to the common type for `LowCardinality` types. This allows to execute UNION ALL of tables with columns of `LowCardinality` and other columns. This fixes #8212. This fixes #4342. #12275 (alexey-milovidov).
- Fixed the behaviour when during multiple sequential inserts in `StorageFile` header for some special types was written more than once. This fixed #6155. #12197 (Nikita Mikhaylov).
- Fixed logical functions for `UInt8` values when they are not equal to 0 or 1. #12196 (Alexander Kazakov).
- Cap `max_memory_usage*` limits to the process resident memory. #12182 (Azat Khuzhin).
- Fixed `dictGet` arguments check during GROUP BY injective functions elimination. #12179 (Azat Khuzhin).
- Don't split the dictionary source's table name into schema and table name itself if ODBC connection does not support schema. #12165 (Vitaly Baranov).

- Fixed wrong logic in `ALTER DELETE` that leads to deleting of records when condition evaluates to NULL. This fixes #9088. This closes #12106. #12153 (alexey-milovidov).
- Fixed transform of query to send to external DBMS (e.g. MySQL, ODBC) in presence of aliases. This fixes #12032. #12151 (alexey-milovidov).
- Fixed potential overflow in integer division. This fixes #12119. #12140 (alexey-milovidov).
- Fixed potential infinite loop in `greatCircleDistance`, `geoDistance`. This fixes #12117. #12137 (alexey-milovidov).
- Normalize "pid" file handling. In previous versions the server may refuse to start if it was killed without proper shutdown and if there is another process that has the same pid as previously runned server. Also pid file may be removed in unsuccessful server startup even if there is another server running. This fixes #3501. #12133 (alexey-milovidov).
- Fixed handling dependency of table with `ENGINE=Dictionary` on dictionary. This fixes #10994. This fixes #10397. #12116 (Vitaly Baranov).
- Fixed performance for selects with `UNION` caused by wrong limit for the total number of threads. Fixes #12030. #12103 (Nikolai Kochetov).
- Fixed segfault with `-StateResample` combinators. #12092 (Anton Popov).
- Fixed empty `result_rows` and `result_bytes` metrics in `system.quey_log` for selects. Fixes #11595. #12089 (Nikolai Kochetov).
- Fixed unnecessary limiting the number of threads for selects from `VIEW`. Fixes #11937. #12085 (Nikolai Kochetov).
- Fixed possible crash while using wrong type for `PREDWHERE`. Fixes #12053, #12060. #12060 (Nikolai Kochetov).
- Fixed error `Expected single dictionary argument for function` for function `defaultValueOfArgumentType` with `LowCardinality` type. Fixes #11808. #12056 (Nikolai Kochetov).
- Fixed error `Cannot capture column` for higher-order functions with `Tuple(LowCardinality)` argument. Fixes #9766. #12055 (Nikolai Kochetov).
- Parse tables metadata in parallel when loading database. This fixes slow server startup when there are large number of tables. #12045 (tavplubix).
- Make `topK` aggregate function return `Enum` for `Enum` types. This fixes #3740. #12043 (alexey-milovidov).
- Fixed constraints check if constraint is a constant expression. This fixes #11360. #12042 (alexey-milovidov).
- Fixed incorrect comparison of tuples with `Nullable` columns. Fixes #11985. #12039 (Nikolai Kochetov).
- Fixed calculation of access rights when `allow_introspection_functions=0`. #12031 (Vitaly Baranov).
- Fixed wrong result and potential crash when invoking function `if` with arguments of type `FixedString` with different sizes. This fixes #11362. #12021 (alexey-milovidov).
- A query with function `neighbor` as the only returned expression may return empty result if the function is called with offset -9223372036854775808. This fixes #11367. #12019 (alexey-milovidov).
- Fixed calculation of access rights when `allow_ddl=0`. #12015 (Vitaly Baranov).

- Fixed potential array size overflow in generateRandom that may lead to crash. This fixes #11371. #12013 (alexey-milovidov).
- Fixed potential floating point exception. This closes #11378. #12005 (alexey-milovidov).
- Fixed wrong setting name in log message at server startup. #11997 (alexey-milovidov).
- Fixed Query parameter was not set in Values format. Fixes #11918. #11936 (tavplubix).
- Keep aliases for substitutions in query (parametrized queries). This fixes #11914. #11916 (alexey-milovidov).
- Fixed bug with no moves when changing storage policy from default one. #11893 (Vladimir Chebotarev).
- Fixed potential floating point exception when parsing DateTime64. This fixes #11374. #11875 (alexey-milovidov).
- Fixed memory accounting via HTTP interface (can be significant with wait\_end\_of\_query=1). #11840 (Azat Khuzhin).
- Parse metadata stored in zookeeper before checking for equality. #11739 (Azat Khuzhin).

## Performance Improvement

- Index not used for IN operator with literals, performance regression introduced around v19.3. This fixes #10574. #12062 (nvartolomei).

## Build/Testing/Packaging Improvement

- Install ca-certificates before the first apt-get update in Dockerfile. #12095 (Ivan Blinkov).

# ClickHouse release v20.4.6.53-stable 2020-06-25

## Bug Fix

- Fix rare crash caused by using Nullable column in prewhere condition. Continuation of #11608. #11869 (Nikolai Kochetov).
- Don't allow arrayJoin inside higher order functions. It was leading to broken protocol synchronization. This closes #3933. #11846 (alexey-milovidov).
- Fix wrong result of comparison of FixedString with constant String. This fixes #11393. This bug appeared in version 20.4. #11828 (alexey-milovidov).
- Fix wrong result for if() with NULLs in condition. #11807 (Artem Zuikov).
- Fix using too many threads for queries. #11788 (Nikolai Kochetov).
- Fix unexpected behaviour of queries like SELECT \*, xyz.\* which were success while an error expected. #11753 (hexiaoting).
- Now replicated fetches will be cancelled during metadata alter. #11744 (alesapin).
- Fixed LOGICAL\_ERROR caused by wrong type deduction of complex literals in Values input format. #11732 (tavplubix).
- Fix ORDER BY ... WITH FILL over const columns. #11697 (Anton Popov).
- Pass proper timeouts when communicating with XDBC bridge. Recently timeouts were not respected when checking bridge liveness and receiving meta info. #11690 (alexey-milovidov).

- Fix `LIMIT n WITH TIES` usage together with `ORDER BY` statement, which contains aliases. #11689 (Anton Popov).
- Fix error which leads to an incorrect state of `system.mutations`. It may show that whole mutation is already done but the server still has `MUTATE_PART` tasks in the replication queue and tries to execute them. This fixes #11611. #11681 (alesapin).
- Add support for regular expressions with case-insensitive flags. This fixes #11101 and fixes #11506. #11649 (alexey-milovidov).
- Remove trivial count query optimization if row-level security is set. In previous versions the user get total count of records in a table instead filtered. This fixes #11352. #11644 (alexey-milovidov).
- Fix bloom filters for String (data skipping indices). #11638 (Azat Khuzhin).
- Fix rare crash caused by using `Nullable` column in prewhere condition. (Probably it is connected with #11572 somehow). #11608 (Nikolai Kochetov).
- Fix error `Block structure mismatch` for queries with sampling reading from `Buffer` table. #11602 (Nikolai Kochetov).
- Fix wrong exit code of the clickhouse-client, when `exception.code() % 256 = 0`. #11601 (filimonov).
- Fix trivial error in log message about "Mark cache size was lowered" at server startup. This closes #11399. #11589 (alexey-milovidov).
- Fix error `Size of offsets does not match size of column` for queries with `PREWHERE` column in (subquery) and `ARRAY JOIN`. #11580 (Nikolai Kochetov).
- Fixed rare segfault in `SHOW CREATE TABLE` Fixes #11490. #11579 (tavplubix).
- All queries in HTTP session have had the same `query_id`. It is fixed. #11578 (tavplubix).
- Now clickhouse-server docker container will prefer IPv6 checking server aliveness. #11550 (Ivan Starkov).
- Fix shard\_num/replica\_num for `<node>` (breaks `use_compact_format_in_distributed_parts_names`). #11528 (Azat Khuzhin).
- Fix race condition which may lead to an exception during table drop. It's a bit tricky and not dangerous at all. If you want an explanation, just notice me in telegram. #11523 (alesapin).
- Fix memory leak when exception is thrown in the middle of aggregation with -State functions. This fixes #8995. #11496 (alexey-milovidov).
- If data skipping index is dependent on columns that are going to be modified during background merge (for SummingMergeTree, AggregatingMergeTree as well as for TTL GROUP BY), it was calculated incorrectly. This issue is fixed by moving index calculation after merge so the index is calculated on merged data. #11162 (Azat Khuzhin).
- Get rid of old libunwind patches. <https://github.com/ClickHouse-Extras/libunwind/commit/500aa227911bd185a94bfc071d68f4d3b03cb3b1#r39048012> This allows to disable `-fno-omit-frame-pointer` in clang builds that improves performance at least by 1% in average. #10761 (Amos Bird).
- Fix usage of primary key wrapped into a function with 'FINAL' modifier and 'ORDER BY' optimization. #10715 (Anton Popov).

## Build/Testing/Packaging Improvement

- Fix several non significant errors in unit tests. #11262 (alesapin).

- Fix (false) MSan report in MergeTreeIndexFullText. The issue first appeared in #9968. #10801 (alexey-milovidov).

## ClickHouse release v20.4.5.36-stable 2020-06-10

### Bug Fix

- Fix the error Data compressed with different methods that can happen if min\_bytes\_to\_use\_direct\_io is enabled and PREWHERE is active and using SAMPLE or high number of threads. This fixes #11539. #11540 (alexey-milovidov).
- Fix return compressed size for codecs. #11448 (Nikolai Kochetov).
- Fix server crash when a column has compression codec with non-literal arguments. Fixes #11365. #11431 (alesapin).
- Fix pointInPolygon with nan as point. Fixes #11375. #11421 (Alexey Ilyukhov).
- Fix potential uninitialized memory read in MergeTree shutdown if table was not created successfully. #11420 (alexey-milovidov).
- Fixed geohashesInBox with arguments outside of latitude/longitude range. #11403 (Vasily Nemkov).
- Fix possible Pipeline stuck error for queries with external sort and limit. Fixes #11359. #11366 (Nikolai Kochetov).
- Remove redundant lock during parts send in ReplicatedMergeTree. #11354 (alesapin).
- Fix support for \G (vertical output) in clickhouse-client in multiline mode. This closes #9933. #11350 (alexey-milovidov).
- Fix potential segfault when using Lazy database. #11348 (alexey-milovidov).
- Fix crash in quantilesExactWeightedArray. #11337 (Nikolai Kochetov).
- Now merges stopped before change metadata in ALTER queries. #11335 (alesapin).
- Make writing to MATERIALIZED VIEW with setting parallel\_view\_processing = 1 parallel again. Fixes #10241. #11330 (Nikolai Kochetov).
- Fix visitParamExtractRaw when extracted JSON has strings with unbalanced { or [. #11318 (Ewout).
- Fix very rare race condition in ThreadPool. #11314 (alexey-milovidov).
- Fix insignificant data race in clickhouse-copier. Found by integration tests. #11313 (alexey-milovidov).
- Fix potential uninitialized memory in conversion. Example: SELECT toIntervalSecond(now64()). #11311 (alexey-milovidov).
- Fix the issue when index analysis cannot work if a table has Array column in primary key and if a query is filtering by this column with empty or notEmpty functions. This fixes #11286. #11303 (alexey-milovidov).
- Fix bug when query speed estimation can be incorrect and the limit of min\_execution\_speed may not work or work incorrectly if the query is throttled by max\_network\_bandwidth, max\_execution\_speed or priority settings. Change the default value of timeout\_before\_checking\_execution\_speed to non-zero, because otherwise the settings min\_execution\_speed and max\_execution\_speed have no effect. This fixes #11297. This fixes #5732. This fixes #6228. Usability improvement: avoid concatenation of exception message with progress bar in clickhouse-client. #11296 (alexey-milovidov).

- Fix crash when SET DEFAULT ROLE is called with wrong arguments. This fixes #10586. #11278 (Vitaly Baranov).
- Fix crash while reading malformed data in Protobuf format. This fixes #5957, fixes #11203. #11258 (Vitaly Baranov).
- Fixed a bug when cache-dictionary could return default value instead of normal (when there are only expired keys). This affects only string fields. #11233 (Nikita Mikhaylov).
- Fix error Block structure mismatch in QueryPipeline while reading from VIEW with constants in inner query. Fixes #11181. #11205 (Nikolai Kochetov).
- Fix possible exception Invalid status for associated output #11200 (Nikolai Kochetov).
- Fix possible error Cannot capture column for higher-order functions with Array(Array(LowCardinality)) captured argument. #11185 (Nikolai Kochetov).
- Fixed S3 globbing which could fail in case of more than 1000 keys and some backends. #11179 (Vladimir Chebotarev).
- If data skipping index is dependent on columns that are going to be modified during background merge (for SummingMergeTree, AggregatingMergeTree as well as for TTL GROUP BY), it was calculated incorrectly. This issue is fixed by moving index calculation after merge so the index is calculated on merged data. #11162 (Azat Khuzhin).
- Fix Kafka performance issue related to reschedules based on limits, which were always applied. #11149 (filimonov).
- Fix for the hang which was happening sometimes during DROP of table engine=Kafka (or during server restarts). #11145 (filimonov).
- Fix excessive reserving of threads for simple queries (optimization for reducing the number of threads, which was partly broken after changes in pipeline). #11114 (Azat Khuzhin).
- Fix predicates optimization for distributed queries (enable\_optimize\_predicate\_expression=1) for queries with HAVING section (i.e. when filtering on the server initiator is required), by preserving the order of expressions (and this is enough to fix), and also force aggregator use column names over indexes. Fixes: #10613, #11413. #10621 (Azat Khuzhin).

## Build/Testing/Packaging Improvement

- Fix several flaky integration tests. #11355 (alesapin).

## ClickHouse release v20.4.4.18-stable 2020-05-26

No changes compared to v20.4.3.16-stable.

## ClickHouse release v20.4.3.16-stable 2020-05-23

### Bug Fix

- Removed logging from mutation finalization task if nothing was finalized. #11109 (alesapin).
- Fixed memory leak in registerDiskS3. #11074 (Pavel Kovalenko).
- Fixed the potential missed data during termination of Kafka engine table. #11048 (filimonov).
- Fixed `parseDateTime64BestEffort` argument resolution bugs. #11038 (Vasily Nemkov).
- Fixed very rare potential use-after-free error in MergeTree if table was not created successfully. #10986, #10970 (alexey-milovidov).

- Fixed metadata (relative path for rename) and data (relative path for symlink) handling for Atomic database. #10980 (Azat Khuzhin).
- Fixed server crash on concurrent `ALTER` and `DROP DATABASE` queries with Atomic database engine. #10968 (tavplubix).
- Fixed incorrect raw data size in `getRawData()` method. #10964 (lgr).
- Fixed incompatibility of two-level aggregation between versions 20.1 and earlier. This incompatibility happens when different versions of ClickHouse are used on initiator node and remote nodes and the size of GROUP BY result is large and aggregation is performed by a single String field. It leads to several unmerged rows for a single key in result. #10952 (alexey-milovidov).
- Fixed sending partially written files by the `DistributedBlockOutputStream`. #10940 (Azat Khuzhin).
- Fixed crash in `SELECT count(notNullIn(NULL, []))`. #10920 (Nikolai Kochetov).
- Fixed the hang which was happening sometimes during `DROP` of Kafka table engine. (or during server restarts). #10910 (filimonov).
- Fixed the impossibility of executing multiple `ALTER RENAME` like `a TO b, c TO a`. #10895 (alesapin).
- Fixed possible race which could happen when you get result from aggregate function state from multiple thread for the same column. The only way it can happen is when you use `finalizeAggregation` function while reading from table with `Memory` engine which stores `AggregateFunction` state for `quantile*` function. #10890 (Nikolai Kochetov).
- Fixed backward compatibility with tuples in Distributed tables. #10889 (Anton Popov).
- Fixed `SIGSEGV` in `StringHashTable` if such a key does not exist. #10870 (Azat Khuzhin).
- Fixed `WATCH` hangs after `LiveView` table was dropped from database with `Atomic` engine. #10859 (tavplubix).
- Fixed bug in `ReplicatedMergeTree` which might cause some `ALTER` on `OPTIMIZE` query to hang waiting for some replica after it become inactive. #10849 (tavplubix).
- Now constraints are updated if the column participating in `CONSTRAINT` expression was renamed. Fixes #10844. #10847 (alesapin).
- Fixed potential read of uninitialized memory in cache-dictionary. #10834 (alexey-milovidov).
- Fixed columns order after `Block::sortColumns()`. #10826 (Azat Khuzhin).
- Fixed the issue with `ODBC` bridge when no quoting of identifiers is requested. Fixes #7984. #10821 (alexey-milovidov).
- Fixed UBSan and MSan report in `DateLUT`. #10798 (alexey-milovidov).
- Fixed incorrect type conversion in key conditions. Fixes #6287. #10791 (Andrew Onyshchuk).
- Fixed `parallel_view_processing` behavior. Now all insertions into `MATERIALIZED VIEW` without exception should be finished if exception happened. Fixes #10241. #10757 (Nikolai Kochetov).
- Fixed combinator `-OrNull` and `-OrDefault` when combined with `-State`. #10741 (hcz).
- Fixed possible buffer overflow in function `h3EdgeAngle`. #10711 (alexey-milovidov).
- Fixed bug which locks concurrent alters when table has a lot of parts. #10659 (alesapin).
- Fixed nullptr dereference in `StorageBuffer` if server was shutdown before table startup. #10641 (alexey-milovidov).

- Fixed `optimize_skip_unused_shards` with `LowCardinality`. [#10611](#) ([Azat Khuzhin](#)).
- Fixed handling condition variable for synchronous mutations. In some cases signals to that condition variable could be lost. [#10588](#) ([Vladimir Chebotarev](#)).
- Fixed possible crash when `createDictionary()` is called before `loadStoredObject()` has finished. [#10587](#) ([Vitaly Baranov](#)).
- Fixed `SELECT` of column ALIAS which default expression type different from column type. [#10563](#) ([Azat Khuzhin](#)).
- Implemented comparison between `DateTime64` and String values. [#10560](#) ([Vasily Nemkov](#)).
- Disable `GROUP BY sharding_key` optimization by default (`optimize_distributed_group_by_sharding_key` had been introduced and turned off by default, due to trickery of `sharding_key` analyzing, simple example is `if` in sharding key) and fix it for `WITH ROLLUP/CUBE/TOTALS`. [#10516](#) ([Azat Khuzhin](#)).
- Fixed [#10263](#). [#10486](#) ([Azat Khuzhin](#)).
- Added tests about `max_rows_to_sort` setting. [#10268](#) ([alexey-milovidov](#)).
- Added backward compatibility for create bloom filter index. [#10551](#). [#10569](#) ([Winter Zhang](#)).

## ClickHouse release v20.4.2.9, 2020-05-12

### Backward Incompatible Change

- System tables (e.g. `system.query_log`, `system.trace_log`, `system.metric_log`) are using compact data part format for parts smaller than 10 MiB in size. Compact data part format is supported since version 20.3. If you are going to downgrade to version less than 20.3, you should manually delete table data for system logs in `/var/lib/clickhouse/data/system/`.
- When string comparison involves `FixedString` and compared arguments are of different sizes, do comparison as if smaller string is padded to the length of the larger. This is intended for SQL compatibility if we imagine that `FixedString` data type corresponds to SQL CHAR. This closes [#9272](#). [#10363](#) ([alexey-milovidov](#))
- Make `SHOW CREATE TABLE` multiline. Now it is more readable and more like MySQL. [#10049](#) ([Azat Khuzhin](#))
- Added a setting `validate_polygons` that is used in `pointInPolygon` function and enabled by default. [#9857](#) ([alexey-milovidov](#))

### New Feature

- Add support for secured connection from ClickHouse to Zookeeper [#10184](#) ([Konstantin Lebedev](#))
- Support custom HTTP handlers. See [#5436](#) for description. [#7572](#) ([Winter Zhang](#))
- Add MessagePack Input/Output format. [#9889](#) ([Kruglov Pavel](#))
- Add Regexp input format. [#9196](#) ([Kruglov Pavel](#))
- Added output format `Markdown` for embedding tables in markdown documents. [#10317](#) ([Kruglov Pavel](#))
- Added support for custom settings section in dictionaries. Also fixes issue [#2829](#). [#10137](#) ([Artem Streltsov](#))
- Added custom settings support in DDL-queries for `CREATE DICTIONARY` [#10465](#) ([Artem Streltsov](#))
- Add simple server-wide memory profiler that will collect allocation contexts when server memory usage becomes higher than the next allocation threshold. [#10444](#) ([alexey-milovidov](#))

- Add setting `always_fetch_merged_part` which restrict replica to merge parts by itself and always prefer dowloading from other replicas. #10379 (alesapin)
- Add function `JSONExtractKeysAndValuesRaw` which extracts raw data from JSON objects #10378 (hcz)
- Add memory usage from OS to `system.asynchronous_metrics`. #10361 (alexey-milovidov)
- Added generic variants for functions `least` and `greatest`. Now they work with arbitrary number of arguments of arbitrary types. This fixes #4767 #10318 (alexey-milovidov)
- Now ClickHouse controls timeouts of dictionary sources on its side. Two new settings added to cache dictionary configuration: `strict_max_lifetime_seconds`, which is `max_lifetime` by default, and `query_wait_timeout_milliseconds`, which is one minute by default. The first settings is also useful with `allow_read_expired_keys` settings (to forbid reading very expired keys). #10337 (Nikita Mikhaylov)
- Add `log_queries_min_type` to filter which entries will be written to `query_log` #10053 (Azat Khuzhin)
- Added function `isConstant`. This function checks whether its argument is constant expression and returns 1 or 0. It is intended for development, debugging and demonstration purposes. #10198 (alexey-milovidov)
- add `joinGetOrNull` to return NULL when key is missing instead of returning the default value. #10094 (Amos Bird)
- Consider `NULL` to be equal to `NULL` in `IN` operator, if the option `transform_null_in` is set. #10085 (achimbab)
- Add `ALTER TABLE ... RENAME COLUMN` for MergeTree table engines family. #9948 (alesapin)
- Support parallel distributed `INSERT SELECT`. #9759 (vxider)
- Add ability to query `Distributed over Distributed` (w/o `distributed_group_by_no_merge`) ... #9923 (Azat Khuzhin)
- Add function `arrayReduceInRanges` which aggregates array elements in given ranges. #9598 (hcz)
- Add Dictionary Status on prometheus exporter. #9622 (Guillaume Tassery)
- Add function `arrayAUC` #8698 (taiyang-li)
- Support `DROP VIEW` statement for better TPC-H compatibility. #9831 (Amos Bird)
- Add 'strict\_order' option to `windowFunnel()` #9773 (achimbab)
- Support `DATE` and `TIMESTAMP` SQL operators, e.g. `SELECT date '2001-01-01'` #9691 (Artem Zuikov)

## Experimental Feature

- Added experimental database engine Atomic. It supports non-blocking `DROP` and `RENAME TABLE` queries and atomic `EXCHANGE TABLES t1 AND t2` query #7512 (tavplubix)
- Initial support for ReplicatedMergeTree over S3 (it works in suboptimal way) #10126 (Pavel Kovalenko)

## Bug Fix

- Fixed incorrect scalar results inside inner query of `MATERIALIZED VIEW` in case if this query contained dependent table #10603 (Nikolai Kochetov)
- Fixed bug, which caused HTTP requests to get stuck on client closing connection when `readonly=2` and `cancel_http_READONLY_queries_on_client_close=1`. #10684 (tavplubix)
- Fix segfault in `StorageBuffer` when exception is thrown on server startup. Fixes #10550 #10609 (tavplubix)

- The query `SYSTEM DROP DNS CACHE` now also drops caches used to check if user is allowed to connect from some IP addresses [#10608 \(tavplubix\)](#)
- Fix usage of multiple `IN` operators with an identical set in one query. Fixes [#10539 #10686 \(Anton Popov\)](#)
- Fix crash in `generateRandom` with nested types. Fixes [#10583. #10734 \(Nikolai Kochetov\)](#)
- Fix data corruption for `LowCardinality(FixedString)` key column in `SummingMergeTree` which could have happened after merge. Fixes [#10489. #10721 \(Nikolai Kochetov\)](#)
- Fix logic for `aggregation_memory_efficient_merge_threads` setting. [#10667 \(palasonic1\)](#)
- Fix disappearing totals. Totals could have been filtered if query had `JOIN` or subquery with external `WHERE` condition. Fixes [#10674 #10698 \(Nikolai Kochetov\)](#)
- Fix the lack of parallel execution of remote queries with `distributed_aggregation_memory_efficient` enabled. Fixes [#10655 #10664 \(Nikolai Kochetov\)](#)
- Fix possible incorrect number of rows for queries with `LIMIT`. Fixes [#10566, #10709 #10660 \(Nikolai Kochetov\)](#)
- Fix index corruption, which may occur in some cases after merging compact parts into another compact part. [#10531 \(Anton Popov\)](#)
- Fix the situation, when mutation finished all parts, but hung up in `is_done=0`. [#10526 \(alesapin\)](#)
- Fix overflow at beginning of unix epoch for timezones with fractional offset from UTC. Fixes [#9335. #10513 \(alexey-milovidov\)](#)
- Better diagnostics for input formats. Fixes [#10204 #10418 \(tavplubix\)](#)
- Fix numeric overflow in `simpleLinearRegression()` over large integers [#10474 \(hc2\)](#)
- Fix use-after-free in Distributed shutdown, avoid waiting for sending all batches [#10491 \(Azat Khuzhin\)](#)
- Add CA certificates to clickhouse-server docker image [#10476 \(filimonov\)](#)
- Fix a rare endless loop that might have occurred when using the `addressToLine` function or `AggregateFunctionState` columns. [#10466 \(Alexander Kuzmenkov\)](#)
- Handle zookeeper "no node error" during distributed query [#10050 \(Daniel Chen\)](#)
- Fix bug when server cannot attach table after column's default was altered. [#10441 \(alesapin\)](#)
- Implicitly cast the default expression type to the column type for the ALIAS columns [#10563 \(Azat Khuzhin\)](#)
- Don't remove metadata directory if `ATTACH DATABASE` fails [#10442 \(Winter Zhang\)](#)
- Avoid dependency on system tzdata. Fixes loading of `Africa/Casablanca` timezone on CentOS 8. Fixes [#10211 #10425 \(alexey-milovidov\)](#)
- Fix some issues if data is inserted with quorum and then gets deleted (DROP PARTITION, TTL, etc.). It led to stuck of INSERTs or false-positive exceptions in SELECTs. Fixes [#9946 #10188 \(Nikita Mikhaylov\)](#)
- Check the number and type of arguments when creating BloomFilter index [#9623 #10431 \(Winter Zhang\)](#)
- Prefer `fallback_to_stale_replicas` over `skip_unavailable_shards`, otherwise when both settings specified and there are no up-to-date replicas the query will fail (patch from @alex-zaitsev ) [#10422 \(Azat Khuzhin\)](#)

- Fix the issue when a query with ARRAY JOIN, ORDER BY and LIMIT may return incomplete result. Fixes #10226. #10427 ([Vadim Plakhtinskiy](#))
- Add database name to dictionary name after DETACH/ATTACH. Fixes system.dictionaries table and SYSTEM RELOAD query #10415 ([Azat Khuzhin](#))
- Fix possible incorrect result for extremes in processors pipeline. #10131 ([Nikolai Kochetov](#))
- Fix possible segfault when the setting distributed\_group\_by\_no\_merge is enabled (introduced in 20.3.7.46 by #10131). #10399 ([Nikolai Kochetov](#))
- Fix wrong flattening of `Array(Tuple(...))` data types. Fixes #10259 #10390 ([alexey-milovidov](#))
- Fix column names of constants inside JOIN that may clash with names of constants outside of JOIN #9950 ([Alexander Kuzmenkov](#))
- Fix order of columns after `Block::sortColumns()` #10826 ([Azat Khuzhin](#))
- Fix possible Pipeline stuck error in `ConcatProcessor` which may happen in remote query. #10381 ([Nikolai Kochetov](#))
- Don't make disk reservations for aggregations. Fixes #9241 #10375 ([Azat Khuzhin](#))
- Fix wrong behaviour of datetime functions for timezones that has altered between positive and negative offsets from UTC (e.g. Pacific/Kiritimati). Fixes #7202 #10369 ([alexey-milovidov](#))
- Avoid infinite loop in `dictIsIn` function. Fixes #515 #10365 ([alexey-milovidov](#))
- Disable GROUP BY sharding\_key optimization by default and fix it for WITH ROLLUP/CUBE/TOTALS #10516 ([Azat Khuzhin](#))
- Check for error code when checking parts and don't mark part as broken if the error is like "not enough memory". Fixes #6269 #10364 ([alexey-milovidov](#))
- Show information about not loaded dictionaries in system tables. #10234 ([Vitaly Baranov](#))
- Fix nullptr dereference in StorageBuffer if server was shutdown before table startup. #10641 ([alexey-milovidov](#))
- Fixed `DROP` vs `OPTIMIZE` race in `ReplicatedMergeTree`. `DROP` could leave some garbage in replica path in ZooKeeper if there was concurrent `OPTIMIZE` query. #10312 ([tavplubix](#))
- Fix 'Logical error: CROSS JOIN has expressions' error for queries with comma and names joins mix. Fixes #9910 #10311 ([Artem Zuikov](#))
- Fix queries with `max_bytes_before_external_group_by`. #10302 ([Artem Zuikov](#))
- Fix the issue with limiting maximum recursion depth in parser in certain cases. This fixes #10283 This fix may introduce minor incompatibility: long and deep queries via clickhouse-client may refuse to work, and you should adjust settings `max_query_size` and `max_parser_depth` accordingly. #10295 ([alexey-milovidov](#))
- Allow to use `count(*)` with multiple JOINs. Fixes #9853 #10291 ([Artem Zuikov](#))
- Fix error Pipeline stuck with `max_rows_to_group_by` and `group_by_overflow_mode = 'break'`. #10279 ([Nikolai Kochetov](#))
- Fix 'Cannot add column' error while creating `range_hashed` dictionary using DDL query. Fixes #10093. #10235 ([alesapin](#))
- Fix rare possible exception Cannot drain connections: cancel first #10239 ([Nikolai Kochetov](#))

- Fixed bug where ClickHouse would throw "Unknown function lambda." error message when user tries to run ALTER UPDATE/DELETE on tables with ENGINE = Replicated\*. Check for nondeterministic functions now handles lambda expressions correctly. [#10237](#) ([Alexander Kazakov](#))
- Fixed reasonably rare segfault in StorageSystemTables that happens when SELECT ... FROM system.tables is run on a database with Lazy engine. [#10209](#) ([Alexander Kazakov](#))
- Fix possible infinite query execution when the query actually should stop on LIMIT, while reading from infinite source like system.numbers or system.zeros. [#10206](#) ([Nikolai Kochetov](#))
- Fixed "generateRandom" function for Date type. This fixes [#9973](#). Fix an edge case when dates with year 2106 are inserted to MergeTree tables with old-style partitioning but partitions are named with year 1970. [#10218](#) ([alexey-milovidov](#))
- Convert types if the table definition of a View does not correspond to the SELECT query. This fixes [#10180](#) and [#10022](#) [#10217](#) ([alexey-milovidov](#))
- Fix `parseDateTimeBestEffort` for strings in RFC-2822 when day of week is Tuesday or Thursday. This fixes [#10082](#) [#10214](#) ([alexey-milovidov](#))
- Fix column names of constants inside JOIN that may clash with names of constants outside of JOIN. [#10207](#) ([alexey-milovidov](#))
- Fix move-to-prewhere optimization in presence of arrayJoin functions (in certain cases). This fixes [#10092](#) [#10195](#) ([alexey-milovidov](#))
- Fix issue with separator appearing in SCRAMBLE for native mysql-connector-java (JDBC) [#10140](#) ([BohuTANG](#))
- Fix using the current database for an access checking when the database isn't specified. [#10192](#) ([Vitaly Baranov](#))
- Fix ALTER of tables with compact parts. [#10130](#) ([Anton Popov](#))
- Add the ability to relax the restriction on non-deterministic functions usage in mutations with `allow_nondeterministic_mutations` setting. [#10186](#) ([filimonov](#))
- Fix `DROP TABLE` invoked for dictionary [#10165](#) ([Azat Khuzhin](#))
- Convert blocks if structure does not match when doing `INSERT` into Distributed table [#10135](#) ([Azat Khuzhin](#))
- The number of rows was logged incorrectly (as sum across all parts) when inserted block is split by parts with partition key. [#10138](#) ([alexey-milovidov](#))
- Add some arguments check and support identifier arguments for MySQL Database Engine [#10077](#) ([Winter Zhang](#))
- Fix incorrect `index_granularity_bytes` check while creating new replica. Fixes [#10098](#). [#10121](#) ([alesapin](#))
- Fix bug in `CHECK TABLE` query when table contain skip indices. [#10068](#) ([alesapin](#))
- Fix Distributed-over-Distributed with the only one shard in a nested table [#9997](#) ([Azat Khuzhin](#))
- Fix possible rows loss for queries with `JOIN` and `UNION ALL`. Fixes [#9826](#), [#10113](#). ... [#10099](#) ([Nikolai Kochetov](#))
- Fix bug in dictionary when local clickhouse server is used as source. It may caused memory corruption if types in dictionary and source are not compatible. [#10071](#) ([alesapin](#))

- Fixed replicated tables startup when updating from an old ClickHouse version where `/table/replicas/replica_name/metadata` node does not exist. Fixes #10037. #10095 (alesapin)
- Fix error Cannot clone block with columns because block has 0 columns ... While executing `GroupingAggregatedTransform`. It happened when setting `distributed_aggregation_memory_efficient` was enabled, and distributed query read aggregating data with mixed single and two-level aggregation from different shards. #10063 (Nikolai Kochetov)
- Fix deadlock when database with materialized view failed attach at start #10054 (Azat Khuzhin)
- Fix a segmentation fault that could occur in GROUP BY over string keys containing trailing zero bytes (#8636, #8925). ... #10025 (Alexander Kuzmenkov)
- Fix wrong results of distributed queries when alias could override qualified column name. Fixes #9672 #9714 #9972 (Artem Zuikov)
- Fix possible deadlock in `SYSTEM RESTART REPLICAS` #9955 (tavplubix)
- Fix the number of threads used for remote query execution (performance regression, since 20.3). This happened when query from `Distributed` table was executed simultaneously on local and remote shards. Fixes #9965 #9971 (Nikolai Kochetov)
- Fixed `DeleteOnDestroy` logic in `ATTACH PART` which could lead to automatic removal of attached part and added few tests #9410 (Vladimir Chebotarev)
- Fix a bug with ON CLUSTER DDL queries freezing on server startup. #9927 (Gagan Arneja)
- Fix bug in which the necessary tables weren't retrieved at one of the processing stages of queries to some databases. Fixes #9699. #9949 (achulkov2)
- Fix 'Not found column in block' error when `JOIN` appears with `TOTALS`. Fixes #9839 #9939 (Artem Zuikov)
- Fix parsing multiple hosts set in the `CREATE USER` command #9924 (Vitaly Baranov)
- Fix `TRUNCATE` for Join table engine (#9917). #9920 (Amos Bird)
- Fix race condition between drop and optimize in `ReplicatedMergeTree`. #9901 (alesapin)
- Fix `DISTINCT` for `Distributed` when `optimize_skip_unused_shards` is set. #9808 (Azat Khuzhin)
- Fix "scalar does not exist" error in `ALTERs` (#9878). ... #9904 (Amos Bird)
- Fix error with qualified names in `distributed_product_mode='local'`. Fixes #4756 #9891 (Artem Zuikov)
- For `INSERT` queries shards now do clamp the settings from the initiator to their constraints instead of throwing an exception. This fix allows to send `INSERT` queries to a shard with another constraints. This change improves fix #9447. #9852 (Vitaly Baranov)
- Add some retries when committing offsets to Kafka broker, since it can reject commit if during `offsets.commit.timeout.ms` there were no enough replicas available for the `_consumer_offsets` topic #9884 (filimonov)
- Fix Distributed engine behavior when virtual columns of the underlying table used in WHERE #9847 (Azat Khuzhin)
- Fixed some cases when timezone of the function argument wasn't used properly. #9574 (Vasily Nemkov)
- Fix 'Different expressions with the same alias' error when query has PREWHERE and WHERE on distributed table and `SET distributed_product_mode = 'local'`. #9871 (Artem Zuikov)

- Fix mutations excessive memory consumption for tables with a composite primary key. This fixes #9850. #9860 (alesapin)
- Fix calculating grants for introspection functions from the setting `allow_introspection_functions`. #9840 (Vitaly Baranov)
- Fix max\_distributed\_connections (w/ and w/o Processors) #9673 (Azat Khuzhin)
- Fix possible exception Got 0 in totals chunk, expected 1 on client. It happened for queries with JOIN in case if right joined table had zero rows. Example: `select * from system.one t1 join system.one t2 on t1.dummy = t2.dummy limit 0 FORMAT TabSeparated;` Fixes #9777. ... #9823 (Nikolai Kochetov)
- Fix 'COMMA to CROSS JOIN rewriter is not enabled or cannot rewrite query' error in case of subqueries with COMMA JOIN out of tables lists (i.e. in WHERE). Fixes #9782 #9830 (Artem Zuikov)
- Fix server crashing when `optimize_skip_unused_shards` is set and expression for key can't be converted to its field type #9804 (Azat Khuzhin)
- Fix empty string handling in `splitByString`. #9767 (hczi)
- Fix broken `ALTER TABLE DELETE COLUMN` query for compact parts. #9779 (alesapin)
- Fixed missing `rows_before_limit_at_least` for queries over http (with processors pipeline). Fixes #9730 #9757 (Nikolai Kochetov)
- Fix excessive memory consumption in `ALTER` queries (mutations). This fixes #9533 and #9670. #9754 (alesapin)
- Fix possible permanent "Cannot schedule a task" error. #9154 (Azat Khuzhin)
- Fix bug in backquoting in external dictionaries DDL. Fixes #9619. #9734 (alesapin)
- Fixed data race in `text_log`. It does not correspond to any real bug. #9726 (alexey-milovidov)
- Fix bug in a replication that does not allow replication to work if the user has executed mutations on the previous version. This fixes #9645. #9652 (alesapin)
- Fixed incorrect internal function names for `sumKahan` and `sumWithOverflow`. It led to exception while using this functions in remote queries. #9636 (Azat Khuzhin)
- Add setting `use_compact_format_in_distributed_parts_names` which allows to write files for `INSERT` queries into `Distributed` table with more compact format. This fixes #9647. #9653 (alesapin)
- Fix RIGHT and FULL JOIN with LowCardinality in JOIN keys. #9610 (Artem Zuikov)
- Fix possible exceptions `Size of filter does not match size of column` and `Invalid number of rows in Chunk in MergeTreeRangeReader`. They could appear while executing `PREWHERE` in some cases. #9612 (Anton Popov)
- Allow `ALTER ON CLUSTER` of Distributed tables with internal replication. This fixes #3268 #9617 (shinoi2)
- Fix issue when timezone was not preserved if you write a simple arithmetic expression like `time + 1` (in contrast to an expression like `time + INTERVAL 1 SECOND`). This fixes #5743 #9323 (alexey-milovidov)

## Improvement

- Use time zone when comparing `DateTime` with string literal. This fixes #5206. #10515 (alexey-milovidov)
- Print verbose diagnostic info if Decimal value cannot be parsed from text input format. #10205 (alexey-milovidov)
- Add tasks/memory metrics for distributed/buffer schedule pools #10449 (Azat Khuzhin)

- Display result as soon as it's ready for SELECT DISTINCT queries in clickhouse-local and HTTP interface. This fixes #8951 #9559 (alexey-milovidov)
- Allow to use SAMPLE OFFSET query instead of cityHash64(PRIMARY KEY) % N == n for splitting in clickhouse-copier. To use this feature, pass --experimental-use-sample-offset 1 as a command line argument. #10414 (Nikita Mikhaylov)
- Allow to parse BOM in TSV if the first column cannot contain BOM in its value. This fixes #10301 #10424 (alexey-milovidov)
- Add Avro nested fields insert support #10354 (Andrew Onyshchuk)
- Allowed to alter column in non-modifying data mode when the same type is specified. #10382 (Vladimir Chebotarev)
- Auto distributed\_group\_by\_no\_merge on GROUP BY sharding key (if optimize\_skip\_unused\_shards is set) #10341 (Azat Khuzhin)
- Optimize queries with LIMIT/LIMIT BY/ORDER BY for distributed with GROUP BY sharding\_key #10373 (Azat Khuzhin)
- Added a setting max\_server\_memory\_usage to limit total memory usage of the server. The metric MemoryTracking is now calculated without a drift. The setting max\_memory\_usage\_for\_all\_queries is now obsolete and does nothing. This closes #10293. #10362 (alexey-milovidov)
- Add config option system\_tables\_lazy\_load. If it's set to false, then system tables with logs are loaded at the server startup. Alexander Burmak, Svyatoslav Tkhon II Pak, #9642 #10359 (alexey-milovidov)
- Use background thread pool (background\_schedule\_pool\_size) for distributed sends #10263 (Azat Khuzhin)
- Use background thread pool for background buffer flushes. #10315 (Azat Khuzhin)
- Support for one special case of removing incompletely written parts. This fixes #9940. #10221 (alexey-milovidov)
- Use isInjective() over manual list of such functions for GROUP BY optimization. #10342 (Azat Khuzhin)
- Avoid printing error message in log if client sends RST packet immediately on connect. It is typical behaviour of IPVS balancer with keepalived and VRRP. This fixes #1851 #10274 (alexey-milovidov)
- Allow to parse +inf for floating point types. This closes #1839 #10272 (alexey-milovidov)
- Implemented generateRandom table function for Nested types. This closes #9903 #10219 (alexey-milovidov)
- Provide max\_allowed\_packed in MySQL compatibility interface that will help some clients to communicate with ClickHouse via MySQL protocol. #10199 (BohuTANG)
- Allow literals for GLOBAL IN (i.e. SELECT \* FROM remote('localhost', system.one) WHERE dummy global in (0)) #10196 (Azat Khuzhin)
- Fix various small issues in interactive mode of clickhouse-client #10194 (alexey-milovidov)
- Avoid superfluous dictionaries load (system.tables, DROP SHOW CREATE TABLE) #10164 (Azat Khuzhin)
- Update to RWLock: timeout parameter for getLock() + implementation reworked to be phase fair #10073 (Alexander Kazakov)
- Enhanced compatibility with native mysql-connector-java(JDBC) #10021 (BohuTANG)

- The function `toString` is considered monotonic and can be used for index analysis even when applied in tautological cases with String or LowCardinality(String) argument. #10110 (Amos Bird)
- Add ON CLUSTER clause support to commands {CREATE|DROP} USER/ROLE/ROW POLICY/SETTINGS PROFILE/QUOTA, GRANT. #9811 (Vitaly Baranov)
- Virtual hosted-style support for S3 URI #9998 (Pavel Kovalenko)
- New layout type for dictionaries with no arguments can be specified without round brackets in dictionaries DDL-queries. Fixes #10057. #10064 (alesapin)
- Add ability to use number ranges with leading zeros in filepath #9989 (Olga Khvostikova)
- Better memory usage in CROSS JOIN. #10029 (Artem Zuikov)
- Try to connect to all shards in cluster when getting structure of remote table and `skip_unavailable_shards` is set. #7278 (nvartolomei)
- Add `total_rows/total_bytes` into the `system.tables` table. #9919 (Azat Khuzhin)
- System log tables now use polymorphic parts by default. #9905 (Anton Popov)
- Add type column into `system.settings/merge_tree_settings` #9909 (Azat Khuzhin)
- Check for available CPU instructions at server startup as early as possible. #9888 (alexey-milovidov)
- Remove ORDER BY stage from mutations because we read from a single ordered part in a single thread. Also add check that the rows in mutation are ordered by sorting key and this order is not violated. #9886 (alesapin)
- Implement operator LIKE for FixedString at left hand side. This is needed to better support TPC-DS queries. #9890 (alexey-milovidov)
- Add `force_optimize_skip_unused_shards_no_nested` that will disable `force_optimize_skip_unused_shards` for nested Distributed table #9812 (Azat Khuzhin)
- Now columns size is calculated only once for MergeTree data parts. #9827 (alesapin)
- Evaluate constant expressions for `optimize_skip_unused_shards` (i.e. `SELECT * FROM foo_dist WHERE key=xxHash32(0)`) #8846 (Azat Khuzhin)
- Check for using `Date` or `DateTime` column from TTL expressions was removed. #9967 (Vladimir Chebotarev)
- DiskS3 hard links optimal implementation. #9760 (Pavel Kovalenko)
- If `set multiple_joins_rewriter_version = 2` enables second version of multiple JOIN rewrites that keeps not clashed column names as is. It supports multiple JOINS with `USING` and allow `select *` for JOINS with subqueries. #9739 (Artem Zuikov)
- Implementation of "non-blocking" alter for StorageMergeTree #9606 (alesapin)
- Add MergeTree full support for DiskS3 #9646 (Pavel Kovalenko)
- Extend `splitByString` to support empty strings as separators. #9742 (hczi)
- Add a `timestamp_ns` column to `system.trace_log`. It contains a high-definition timestamp of the trace event, and allows to build timelines of thread profiles ("flame charts"). #9696 (Alexander Kuzmenkov)
- When the setting `send_logs_level` is enabled, avoid intermixing of log messages and query progress. #9634 (Azat Khuzhin)

- Added support of MATERIALIZE TTL IN PARTITION. #9581 (Vladimir Chebotarev)
- Support complex types inside Avro nested fields #10502 (Andrew Onyshchuk)

## Performance Improvement

- Better insert logic for right table for Partial MergeJoin. #10467 (Artem Zuikov)
- Improved performance of row-oriented formats (more than 10% for CSV and more than 35% for Avro in case of narrow tables). #10503 (Andrew Onyshchuk)
- Improved performance of queries with explicitly defined sets at right side of IN operator and tuples on the left side. #10385 (Anton Popov)
- Use less memory for hash table in HashJoin. #10416 (Artem Zuikov)
- Special HashJoin over StorageDictionary. Allow rewrite `dictGet()` functions with JOINs. It's not backward incompatible itself but could uncover #8400 on some installations. #10133 (Artem Zuikov)
- Enable parallel insert of materialized view when its target table supports. #10052 (vxider)
- Improved performance of index analysis with monotonic functions. #9607#10026 (Anton Popov)
- Using SSE2 or SSE4.2 SIMD intrinsics to speed up tokenization in bloom filters. #9968 (Vasily Nemkov)
- Improved performance of queries with explicitly defined sets at right side of IN operator. This fixes performance regression in version 20.3. #9740 (Anton Popov)
- Now clickhouse-copier splits each partition in number of pieces and copies them independently. #9075 (Nikita Mikhaylov)
- Adding more aggregation methods. For example TPC-H query 1 will now pick `FixedHashMap<UInt16, AggregateDataPtr>` and gets 25% performance gain #9829 (Amos Bird)
- Use single row counter for multiple streams in pre-limit transform. This helps to avoid uniting pipeline streams in queries with `limit` but without `order by` (like `select f(x) from (select x from t limit 1000000000)`) and use multiple threads for further processing. #9602 (Nikolai Kochetov)

## Build/Testing/Packaging Improvement

- Use a fork of AWS SDK libraries from ClickHouse-Extras #10527 (Pavel Kovalenko)
- Add integration tests for new ALTER RENAME COLUMN query. #10654 (vzakaznikov)
- Fix possible signed integer overflow in invocation of function `now64` with wrong arguments. This fixes #8973 #10511 (alexey-milovidov)
- Split fuzzer and sanitizer configurations to make build config compatible with Oss-fuzz. #10494 (kyprizel)
- Fixes for clang-tidy on clang-10. #10420 (alexey-milovidov)
- Display absolute paths in error messages. Otherwise KDevelop fails to navigate to correct file and opens a new file instead. #10434 (alexey-milovidov)
- Added `ASAN_OPTIONS` environment variable to investigate errors in CI stress tests with Address sanitizer. #10440 (Nikita Mikhaylov)
- Enable ThinLTO for clang builds (experimental). #10435 (alexey-milovidov)
- Remove accidental dependency on Z3 that may be introduced if the system has Z3 solver installed. #10426 (alexey-milovidov)

- Move integration tests docker files to docker/ directory. #10335 (Ilya Yatsishin)
- Allow to use clang-10 in CI. It ensures that #10238 is fixed. #10384 (alexey-milovidov)
- Update OpenSSL to upstream master. Fixed the issue when TLS connections may fail with the message OpenSSL SSL\_read: error:14094438:SSL routines:ssl3\_read\_bytes:tlsv1 alert internal error and SSL Exception: error:2400006E:random number generator::error retrieving entropy. The issue was present in version 20.1. #8956 (alexey-milovidov)
- Fix clang-10 build. #10238 #10370 (Amos Bird)
- Add performance test for Parallel INSERT for materialized view. #10345 (vxider)
- Fix flaky test test\_settings\_constraints\_distributed.test\_insert\_clamps\_settings. #10346 (Vitaly Baranov)
- Add util to test results upload in CI ClickHouse #10330 (Ilya Yatsishin)
- Convert test results to JSONEachRow format in junit\_to\_html tool #10323 (Ilya Yatsishin)
- Update cctz. #10215 (alexey-milovidov)
- Allow to create HTML report from the purest JUnit XML report. #10247 (Ilya Yatsishin)
- Update the check for minimal compiler version. Fix the root cause of the issue #10250 #10256 (alexey-milovidov)
- Initial support for live view tables over distributed #10179 (vzakaznikov)
- Fix (false) MSan report in MergeTreeIndexFullText. The issue first appeared in #9968. #10801 (alexey-milovidov)
- clickhouse-docker-util #10151 (filimonov)
- Update pdqsort to recent version #10171 (Ivan)
- Update libdivide to v3.0 #10169 (Ivan)
- Add check with enabled polymorphic parts. #10086 (Anton Popov)
- Add cross-compile build for FreeBSD. This fixes #9465 #9643 (Ivan)
- Add performance test for #6924 #6980 (filimonov)
- Add support of /dev/null in the File engine for better performance testing #8455 (Amos Bird)
- Move all folders inside /dbms one level up #9974 (Ivan)
- Add a test that checks that read from MergeTree with single thread is performed in order. Addition to #9670 #9762 (alexey-milovidov)
- Fix the 00964\_live\_view\_watch\_events\_heartbeat.py test to avoid race condition. #9944 (vzakaznikov)
- Fix integration test test\_settings\_constraints #9962 (Vitaly Baranov)
- Every function in its own file, part 12. #9922 (alexey-milovidov)
- Added performance test for the case of extremely slow analysis of array of tuples. #9872 (alexey-milovidov)
- Update zstd to 1.4.4. It has some minor improvements in performance and compression ratio. If you run replicas with different versions of ClickHouse you may see reasonable error messages Data after merge is not byte-identical to data on another replicas. with explanation. These messages are Ok and you should not worry. #10663 (alexey-milovidov)

- Fix TSan report in `system.stack_trace`. #9832 (alexey-milovidov)
- Removed dependency on `clock_getres`. #9833 (alexey-milovidov)
- Added identifier names check with clang-tidy. #9799 (alexey-milovidov)
- Update "builder" docker image. This image is not used in CI but is useful for developers. #9809 (alexey-milovidov)
- Remove old `performance-test` tool that is no longer used in CI. `clickhouse-performance-test` is great but now we are using way superior tool that is doing comparison testing with sophisticated statistical formulas to achieve confident results regardless to various changes in environment. #9796 (alexey-milovidov)
- Added most of clang-static-analyzer checks. #9765 (alexey-milovidov)
- Update Poco to 1.9.3 in preparation for MongoDB URI support. #6892 (Alexander Kuzmenkov)
- Fix build with `-DUSE_STATIC_LIBRARIES=0 -DENABLE_JEMALLOC=0` #9651 (Artem Zuikov)
- For change log script, if merge commit was cherry-picked to release branch, take PR name from commit description. #9708 (Nikolai Kochetov)
- Support `vX.X-conflicts` tag in backport script. #9705 (Nikolai Kochetov)
- Fix `auto-label` for backporting script. #9685 (Nikolai Kochetov)
- Use `libc++` in Darwin cross-build to make it consistent with native build. #9665 (Hui Wang)
- Fix flaky test `01017_uniqCombined_memory_usage`. Continuation of #7236. #9667 (alexey-milovidov)
- Fix build for native MacOS Clang compiler #9649 (Ivan)
- Allow to add various glitches around `pthread_mutex_lock`, `pthread_mutex_unlock` functions. #9635 (alexey-milovidov)
- Add support for clang-tidy in packager script. #9625 (alexey-milovidov)
- Add ability to use unbundled msgpack. #10168 (Azat Khuzhin)

## ClickHouse release v20.3

### ClickHouse release v20.3.21.2-lts, 2020-11-02

#### Bug Fix

- Fix dictGet in sharding\_key (and similar places, i.e. when the function context is stored permanently). #16205 (Azat Khuzhin).
- Fix incorrect empty result for query from Distributed table if query has WHERE, PREWHERE and GLOBAL IN. Fixes #15792. #15933 (Nikolai Kochetov).
- Fix missing or excessive headers in TSV/CSVWithNames formats. This fixes #12504. #13343 (Azat Khuzhin).

### ClickHouse release v20.3.20.6-lts, 2020-10-09

#### Bug Fix

- Mutation might hang waiting for some non-existent part after MOVE or REPLACE PARTITION or, in rare cases, after DETACH or DROP PARTITION. It's fixed. #15724, #15537 (tavplubix).

- Fix hang of queries with a lot of subqueries to same table of MySQL engine. Previously, if there were more than 16 subqueries to same MySQL table in query, it hang forever. #15299 (Anton Popov).
- Fix 'Unknown identifier' in GROUP BY when query has JOIN over Merge table. #15242 (Artem Zuikov).
- Fix to make predicate push down work when subquery contains finalizeAggregation function. Fixes #14847. #14937 (filimonov).
- Concurrent ALTER ... REPLACE/MOVE PARTITION ... queries might cause deadlock. It's fixed. #13626 (tavplubix).

## ClickHouse release v20.3.19.4-lts, 2020-09-18

### Bug Fix

- Fix rare error in SELECT queries when the queried column has DEFAULT expression which depends on the other column which also has DEFAULT and not present in select query and not exists on disk. Partially fixes #14531. #14845 (alesapin).
- Fix bug when ALTER UPDATE mutation with Nullable column in assignment expression and constant value (like UPDATE x = 42) leads to incorrect value in column or segfault. Fixes #13634, #14045. #14646 (alesapin).
- Fix wrong Decimal multiplication result caused wrong decimal scale of result column. #14603 (Artem Zuikov).

### Improvement

- Support custom codecs in compact parts. #12183 (Anton Popov).

## ClickHouse release v20.3.18.10-lts, 2020-09-08

### Bug Fix

- Stop query execution if exception happened in PipelineExecutor itself. This could prevent rare possible query hung. Continuation of #14334. #14402 (Nikolai Kochetov).
- Fixed the behaviour when sometimes cache-dictionary returned default value instead of present value from source. #13624 (Nikita Mikhaylov).
- Fix parsing row policies from users.xml when names of databases or tables contain dots. This fixes #5779, #12527. #13199 (Vitaly Baranov).
- Fix CAST(Nullable(String), Enum()). #12745 (Azat Khuzhin).
- Fixed data race in text\_log. It does not correspond to any real bug. #9726 (alexey-milovidov).

### Improvement

- Fix wrong error for long queries. It was possible to get syntax error other than Max query size exceeded for correct query. #13928 (Nikolai Kochetov).
- Return NULL/zero when value is not parsed completely in parseDateTimeBestEffortOrNull/Zero functions. This fixes #7876. #11653 (alexey-milovidov).

### Performance Improvement

- Slightly optimize very short queries with LowCardinality. #14129 (Anton Popov).

### Build/Testing/Packaging Improvement

- Fix UBSan report (adding zero to nullptr) in HashTable that appeared after migration to clang-10. #10638 (alexey-milovidov).

## ClickHouse release v20.3.17.173-lts, 2020-08-15

### Bug Fix

- Fix crash in JOIN with StorageMerge and `set enable_optimize_predicate_expression=1`. [#13679](#) ([Artem Zuikov](#)).
- Fix invalid return type for comparison of tuples with `NULL` elements. Fixes [#12461](#). [#13420](#) ([Nikolai Kochetov](#)).
- Fix queries with constant columns and `ORDER BY` prefix of primary key. [#13396](#) ([Anton Popov](#)).
- Return passed number for numbers with MSB set in `roundUpToPowerOfTwoOrZero()`. [#13234](#) ([Azat Khuzhin](#)).

## ClickHouse release v20.3.16.165-lts 2020-08-10

### Bug Fix

- Fixed error in `parseDateTimeBestEffort` function when unix timestamp was passed as an argument. This fixes [#13362](#). [#13441](#) ([alexey-milovidov](#)).
- Fixed potentially low performance and slightly incorrect result for `uniqExact`, `topK`, `sumDistinct` and similar aggregate functions called on Float types with `NaN` values. It also triggered assert in debug build. This fixes [#12491](#). [#13254](#) ([alexey-milovidov](#)).
- Fixed function `if` with nullable `constexpr` as cond that is not literal `NULL`. Fixes [#12463](#). [#13226](#) ([alexey-milovidov](#)).
- Fixed assert in `arrayElement` function in case of array elements are Nullable and array subscript is also Nullable. This fixes [#12172](#). [#13224](#) ([alexey-milovidov](#)).
- Fixed unnecessary limiting for the number of threads for selects from local replica. [#12840](#) ([Nikolai Kochetov](#)).
- Fixed possible extra overflow row in data which could appear for queries `WITH TOTALS`. [#12747](#) ([Nikolai Kochetov](#)).
- Fixed performance with large tuples, which are interpreted as functions in `IN` section. The case when user write `WHERE x IN tuple(1, 2, ...)` instead of `WHERE x IN (1, 2, ...)` for some obscure reason. [#12700](#) ([Anton Popov](#)).
- Fixed memory tracking for `input_format_parallel_parsing` (by attaching thread to group). [#12672](#) ([Azat Khuzhin](#)).
- Fixed [#12293](#) allow push predicate when subquery contains `with` clause. [#12663](#) ([Winter Zhang](#)).
- Fixed [#10572](#) fix bloom filter index with const expression. [#12659](#) ([Winter Zhang](#)).
- Fixed SIGSEGV in StorageKafka when broker is unavailable (and not only). [#12658](#) ([Azat Khuzhin](#)).
- Fixed race condition in external dictionaries with cache layout which can lead server crash. [#12566](#) ([alesapin](#)).
- Fixed bug which lead to broken old parts after `ALTER DELETE` query when `enable_mixed_granularity_parts=1`. Fixes [#12536](#). [#12543](#) ([alesapin](#)).
- Better exception for function `in` with invalid number of arguments. [#12529](#) ([Anton Popov](#)).
- Fixed performance issue, while reading from compact parts. [#12492](#) ([Anton Popov](#)).
- Fixed the deadlock if `text_log` is enabled. [#12452](#) ([alexey-milovidov](#)).

- Fixed possible segfault if StorageMerge. Closes #12054. #12401 (tavplubix).
- Fixed TOTALS/ROLLUP/CUBE for aggregate functions with `-State` and `Nullable` arguments. This fixes #12163. #12376 (alexey-milovidov).
- Fixed order of columns in `WITH FILL` modifier. Previously order of columns of `ORDER BY` statement wasn't respected. #12306 (Anton Popov).
- Avoid "bad cast" exception when there is an expression that filters data by virtual columns (like `_table` in `Merge` tables) or by "index" columns in system tables such as filtering by database name when querying from `system.tables`, and this expression returns `Nullable` type. This fixes #12166. #12305 (alexey-milovidov).
- Show error after `TrieDictionary` failed to load. #12290 (Vitaly Baranov).
- The function `arrayFill` worked incorrectly for empty arrays that may lead to crash. This fixes #12263. #12279 (alexey-milovidov).
- Implement conversions to the common type for `LowCardinality` types. This allows to execute UNION ALL of tables with columns of `LowCardinality` and other columns. This fixes #8212. This fixes #4342. #12275 (alexey-milovidov).
- Fixed the behaviour when during multiple sequential inserts in `StorageFile` header for some special types was written more than once. This fixed #6155. #12197 (Nikita Mikhaylov).
- Fixed logical functions for `UInt8` values when they are not equal to 0 or 1. #12196 (Alexander Kazakov).
- Fixed `dictGet` arguments check during GROUP BY injective functions elimination. #12179 (Azat Khuzhin).
- Fixed wrong logic in `ALTER DELETE` that leads to deleting of records when condition evaluates to NULL. This fixes #9088. This closes #12106. #12153 (alexey-milovidov).
- Fixed transform of query to send to external DBMS (e.g. MySQL, ODBC) in presence of aliases. This fixes #12032. #12151 (alexey-milovidov).
- Fixed potential overflow in integer division. This fixes #12119. #12140 (alexey-milovidov).
- Fixed potential infinite loop in `greatCircleDistance`, `geoDistance`. This fixes #12117. #12137 (alexey-milovidov).
- Avoid `There is no query` exception for materialized views with joins or with subqueries attached to system logs (`system.query_log`, `metric_log`, etc) or to engine=Buffer underlying table. #12120 (filimonov).
- Fixed performance for selects with `UNION` caused by wrong limit for the total number of threads. Fixes #12030. #12103 (Nikolai Kochetov).
- Fixed segfault with `-StateResample` combinators. #12092 (Anton Popov).
- Fixed unnecessary limiting the number of threads for selects from `VIEW`. Fixes #11937. #12085 (Nikolai Kochetov).
- Fixed possible crash while using wrong type for `PREWHERE`. Fixes #12053, #12060. #12060 (Nikolai Kochetov).
- Fixed error `Expected single dictionary argument for function` for function `defaultValueOfArgumentType` with `LowCardinality` type. Fixes #11808. #12056 (Nikolai Kochetov).
- Fixed error `Cannot capture column for higher-order functions with Tuple(LowCardinality) argument.` Fixes #9766. #12055 (Nikolai Kochetov).

- Parse tables metadata in parallel when loading database. This fixes slow server startup when there are large number of tables. #12045 (tavplubix).
- Make `topK` aggregate function return `Enum` for `Enum` types. This fixes #3740. #12043 (alexey-milovidov).
- Fixed constraints check if constraint is a constant expression. This fixes #11360. #12042 (alexey-milovidov).
- Fixed incorrect comparison of tuples with `Nullable` columns. Fixes #11985. #12039 (Nikolai Kochetov).
- Fixed wrong result and potential crash when invoking function `if` with arguments of type `FixedString` with different sizes. This fixes #11362. #12021 (alexey-milovidov).
- A query with function `neighbor` as the only returned expression may return empty result if the function is called with offset -9223372036854775808. This fixes #11367. #12019 (alexey-milovidov).
- Fixed potential array size overflow in `generateRandom` that may lead to crash. This fixes #11371. #12013 (alexey-milovidov).
- Fixed potential floating point exception. This closes #11378. #12005 (alexey-milovidov).
- Fixed wrong setting name in log message at server startup. #11997 (alexey-milovidov).
- Fixed Query parameter was not set in `Values` format. Fixes #11918. #11936 (tavplubix).
- Keep aliases for substitutions in query (parametrized queries). This fixes #11914. #11916 (alexey-milovidov).
- Fixed potential floating point exception when parsing `DateTime64`. This fixes #11374. #11875 (alexey-milovidov).
- Fixed memory accounting via `HTTP` interface (can be significant with `wait_end_of_query=1`). #11840 (Azat Khuzhin).
- Fixed wrong result for `if()` with `NULLs` in condition. #11807 (Artem Zuikov).
- Parse metadata stored in zookeeper before checking for equality. #11739 (Azat Khuzhin).
- Fixed `LIMIT n WITH TIES` usage together with `ORDER BY` statement, which contains aliases. #11689 (Anton Popov).
- Fix potential read of uninitialized memory in cache dictionary. #10834 (alexey-milovidov).

## Performance Improvement

- Index not used for `IN` operator with literals, performance regression introduced around v19.3. This fixes #10574. #12062 (nvartolomei).

## ClickHouse release v20.3.12.112-lts 2020-06-25

### Bug Fix

- Fix rare crash caused by using `Nullable` column in prewhere condition. Continuation of #11608. #11869 (Nikolai Kochetov).
- Don't allow `arrayJoin` inside higher order functions. It was leading to broken protocol synchronization. This closes #3933. #11846 (alexey-milovidov).
- Fix using too many threads for queries. #11788 (Nikolai Kochetov).

- Fix unexpected behaviour of queries like `SELECT *, xyz.*` which were success while an error expected. [#11753 \(hexiaoting\)](#).
- Now replicated fetches will be cancelled during metadata alter. [#11744 \(alesapin\)](#).
- Fixed `LOGICAL_ERROR` caused by wrong type deduction of complex literals in `Values` input format. [#11732 \(tavplubix\)](#).
- Fix `ORDER BY ... WITH FILL` over const columns. [#11697 \(Anton Popov\)](#).
- Pass proper timeouts when communicating with XDBC bridge. Recently timeouts were not respected when checking bridge liveness and receiving meta info. [#11690 \(alexey-milovidov\)](#).
- Fix error which leads to an incorrect state of `system.mutations`. It may show that whole mutation is already done but the server still has `MUTATE_PART` tasks in the replication queue and tries to execute them. This fixes [#11611](#). [#11681 \(alesapin\)](#).
- Add support for regular expressions with case-insensitive flags. This fixes [#11101](#) and fixes [#11506](#). [#11649 \(alexey-milovidov\)](#).
- Remove trivial count query optimization if row-level security is set. In previous versions the user get total count of records in a table instead filtered. This fixes [#11352](#). [#11644 \(alexey-milovidov\)](#).
- Fix bloom filters for String (data skipping indices). [#11638 \(Azat Khuzhin\)](#).
- Fix rare crash caused by using `Nullable` column in prewhere condition. (Probably it is connected with [#11572](#) somehow). [#11608 \(Nikolai Kochetov\)](#).
- Fix error `Block structure mismatch` for queries with sampling reading from `Buffer` table. [#11602 \(Nikolai Kochetov\)](#).
- Fix wrong exit code of the clickhouse-client, when `exception.code() % 256 = 0`. [#11601 \(filimonov\)](#).
- Fix trivial error in log message about "Mark cache size was lowered" at server startup. This closes [#11399](#). [#11589 \(alexey-milovidov\)](#).
- Fix error `Size of offsets does not match size of column` for queries with `PREWHERE` column in (subquery) and `ARRAY JOIN`. [#11580 \(Nikolai Kochetov\)](#).
- All queries in HTTP session have had the same `query_id`. It is fixed. [#11578 \(tavplubix\)](#).
- Now clickhouse-server docker container will prefer IPv6 checking server aliveness. [#11550 \(Ivan Starkov\)](#).
- Fix `shard_num/replica_num` for `<node>` (breaks `use_compact_format_in_distributed_parts_names`). [#11528 \(Azat Khuzhin\)](#).
- Fix memory leak when exception is thrown in the middle of aggregation with -State functions. This fixes [#8995](#). [#11496 \(alexey-milovidov\)](#).
- Fix wrong results of distributed queries when alias could override qualified column name. Fixes [#9672](#) [#9714](#). [#9972 \(Artem Zuikov\)](#).

## ClickHouse release v20.3.11.97-Its 2020-06-10

### New Feature

- Now ClickHouse controls timeouts of dictionary sources on its side. Two new settings added to cache dictionary configuration: `strict_max_lifetime_seconds`, which is `max_lifetime` by default and `query_wait_timeout_milliseconds`, which is one minute by default. The first setting is also useful with `allow_expired_keys` settings (to forbid reading very expired keys). [#10337 \(Nikita Mikhaylov\)](#).

## Bug Fix

- Fix the error Data compressed with different methods that can happen if `min_bytes_to_use_direct_io` is enabled and PREWHERE is active and using SAMPLE or high number of threads. This fixes #11539. #11540 (alexey-milovidov).
- Fix return compressed size for codecs. #11448 (Nikolai Kochetov).
- Fix server crash when a column has compression codec with non-literal arguments. Fixes #11365. #11431 (alesapin).
- Fix pointInPolygon with nan as point. Fixes #11375. #11421 (Alexey Ilyukhov).
- Fix crash in JOIN over LowCarinality(T) and Nullable(T). #11380. #11414 (Artem Zuikov).
- Fix error code for wrong USING key. #11373. #11404 (Artem Zuikov).
- Fixed geohashesInBox with arguments outside of latitude/longitude range. #11403 (Vasily Nemkov).
- Better errors for `joinGet()` functions. #11389 (Artem Zuikov).
- Fix possible Pipeline stuck error for queries with external sort and limit. Fixes #11359. #11366 (Nikolai Kochetov).
- Remove redundant lock during parts send in ReplicatedMergeTree. #11354 (alesapin).
- Fix support for \G (vertical output) in clickhouse-client in multiline mode. This closes #9933. #11350 (alexey-milovidov).
- Fix crash in direct selects from StorageJoin (without JOIN) and wrong nullability. #11340 (Artem Zuikov).
- Fix crash in `quantilesExactWeightedArray`. #11337 (Nikolai Kochetov).
- Now merges stopped before change metadata in ALTER queries. #11335 (alesapin).
- Make writing to MATERIALIZED VIEW with setting `parallel_view_processing = 1` parallel again. Fixes #10241. #11330 (Nikolai Kochetov).
- Fix visitParamExtractRaw when extracted JSON has strings with unbalanced { or [. #11318 (Ewout).
- Fix very rare race condition in ThreadPool. #11314 (alexey-milovidov).
- Fix potential uninitialized memory in conversion. Example: `SELECT toIntervalSecond(now64())`. #11311 (alexey-milovidov).
- Fix the issue when index analysis cannot work if a table has Array column in primary key and if a query is filtering by this column with `empty` or `notEmpty` functions. This fixes #11286. #11303 (alexey-milovidov).
- Fix bug when query speed estimation can be incorrect and the limit of `min_execution_speed` may not work or work incorrectly if the query is throttled by `max_network_bandwidth`, `max_execution_speed` or `priority` settings. Change the default value of `timeout_before_checking_execution_speed` to non-zero, because otherwise the settings `min_execution_speed` and `max_execution_speed` have no effect. This fixes #11297. This fixes #5732. This fixes #6228. Usability improvement: avoid concatenation of exception message with progress bar in clickhouse-client. #11296 (alexey-milovidov).
- Fix crash while reading malformed data in Protobuf format. This fixes #5957, fixes #11203. #11258 (Vitaly Baranov).
- Fixed a bug when cache-dictionary could return default value instead of normal (when there are only expired keys). This affects only string fields. #11233 (Nikita Mikhaylov).

- Fix error `Block structure mismatch` in `QueryPipeline` while reading from `VIEW` with constants in inner query. Fixes [#11181](#). [#11205](#) ([Nikolai Kochetov](#)).
- Fix possible exception `Invalid status` for associated output [#11200](#) ([Nikolai Kochetov](#)).
- Fix possible error `Cannot capture column` for higher-order functions with `Array(Array(LowCardinality))` captured argument. [#11185](#) ([Nikolai Kochetov](#)).
- Fixed S3 globbing which could fail in case of more than 1000 keys and some backends. [#11179](#) ([Vladimir Chebotarev](#)).
- If data skipping index is dependent on columns that are going to be modified during background merge (for `SummingMergeTree`, `AggregatingMergeTree` as well as for TTL GROUP BY), it was calculated incorrectly. This issue is fixed by moving index calculation after merge so the index is calculated on merged data. [#11162](#) ([Azat Khuzhin](#)).
- Fix excessive reserving of threads for simple queries (optimization for reducing the number of threads, which was partly broken after changes in pipeline). [#11114](#) ([Azat Khuzhin](#)).
- Fix predicates optimization for distributed queries (`enable_optimize_predicate_expression=1`) for queries with `HAVING` section (i.e. when filtering on the server initiator is required), by preserving the order of expressions (and this is enough to fix), and also force aggregator use column names over indexes. Fixes: [#10613](#), [#11413](#). [#10621](#) ([Azat Khuzhin](#)).
- Introduce commit retry logic to decrease the possibility of getting duplicates from Kafka in rare cases when offset commit was failed. [#9884](#) ([filimonov](#)).

## Performance Improvement

- Get dictionary and check access rights only once per each call of any function reading external dictionaries. [#10928](#) ([Vitaly Baranov](#)).

## Build/Testing/Packaging Improvement

- Fix several flaky integration tests. [#11355](#) ([alesapin](#)).

# ClickHouse release v20.3.10.75-Its 2020-05-23

## Bug Fix

- Removed logging from mutation finalization task if nothing was finalized. [#11109](#) ([alesapin](#)).
- Fixed `parseDateTime64BestEffort` argument resolution bugs. [#11038](#) ([Vasily Nemkov](#)).
- Fixed incorrect raw data size in method `getRawData()`. [#10964](#) ([lgr](#)).
- Fixed incompatibility of two-level aggregation between versions 20.1 and earlier. This incompatibility happens when different versions of ClickHouse are used on initiator node and remote nodes and the size of `GROUP BY` result is large and aggregation is performed by a single `String` field. It leads to several unmerged rows for a single key in result. [#10952](#) ([alexey-milovidov](#)).
- Fixed backward compatibility with tuples in Distributed tables. [#10889](#) ([Anton Popov](#)).
- Fixed `SIGSEGV` in `StringHashTable` if such a key does not exist. [#10870](#) ([Azat Khuzhin](#)).
- Fixed bug in `ReplicatedMergeTree` which might cause some `ALTER` on `OPTIMIZE` query to hang waiting for some replica after it become inactive. [#10849](#) ([tavplubix](#)).
- Fixed columns order after `Block::sortColumns()`. [#10826](#) ([Azat Khuzhin](#)).
- Fixed the issue with ODBC bridge when no quoting of identifiers is requested. Fixes [#7984](#). [#10821](#) ([alexey-milovidov](#)).

- Fixed UBSan and MSan report in DateLUT. #10798 (alexey-milovidov).
- Fixed incorrect type conversion in key conditions. Fixes #6287. #10791 (Andrew Onyshchuk)
- Fixed parallel\_view\_processing behavior. Now all insertions into MATERIALIZED VIEW without exception should be finished if exception happened. Fixes #10241. #10757 (Nikolai Kochetov).
- Fixed combinator -OrNull and -OrDefault when combined with -State. #10741 (hczi).
- Fixed crash in generateRandom with nested types. Fixes #10583. #10734 (Nikolai Kochetov).
- Fixed data corruption for LowCardinality(FixedString) key column in SummingMergeTree which could have happened after merge. Fixes #10489. #10721 (Nikolai Kochetov).
- Fixed possible buffer overflow in function h3EdgeAngle. #10711 (alexey-milovidov).
- Fixed disappearing totals. Totals could have been filtered if query had had join or subquery with external where condition. Fixes #10674. #10698 (Nikolai Kochetov).
- Fixed multiple usages of IN operator with the identical set in one query. #10686 (Anton Popov).
- Fixed bug, which causes http requests stuck on client close when readonly=2 and cancel\_http\_READONLY\_queries\_on\_client\_close=1. Fixes #7939, #7019, #7736, #7091. #10684 (tavplubix).
- Fixed order of parameters in AggregateTransform constructor. #10667 (palasonic1).
- Fixed the lack of parallel execution of remote queries with distributed\_aggregation\_memory\_efficient enabled. Fixes #10655. #10664 (Nikolai Kochetov).
- Fixed possible incorrect number of rows for queries with LIMIT. Fixes #10566, #10709. #10660 (Nikolai Kochetov).
- Fixed a bug which locks concurrent alters when table has a lot of parts. #10659 (alesapin).
- Fixed a bug when on SYSTEM DROP DNS CACHE query also drop caches, which are used to check if user is allowed to connect from some IP addresses. #10608 (tavplubix).
- Fixed incorrect scalar results inside inner query of MATERIALIZED VIEW in case if this query contained dependent table. #10603 (Nikolai Kochetov).
- Fixed SELECT of column ALIAS which default expression type different from column type. #10563 (Azat Khuzhin).
- Implemented comparison between DateTime64 and String values. #10560 (Vasily Nemkov).
- Fixed index corruption, which may occur in some cases after merge compact parts into another compact part. #10531 (Anton Popov).
- Fixed the situation, when mutation finished all parts, but hung up in is\_done=0. #10526 (alesapin).
- Fixed overflow at beginning of unix epoch for timezones with fractional offset from UTC. This fixes #9335. #10513 (alexey-milovidov).
- Fixed improper shutdown of Distributed storage. #10491 (Azat Khuzhin).
- Fixed numeric overflow in simpleLinearRegression over large integers. #10474 (hczi).

## Build/Testing/Packaging Improvement

- Fix UBSan report in LZ4 library. #10631 (alexey-milovidov).
- Fix clang-10 build. #10238. #10370 (Amos Bird).

- Added failing tests about `max_rows_to_sort` setting. #10268 (alexey-milovidov).
- Added some improvements in printing diagnostic info in input formats. Fixes #10204. #10418 (tavplubix).
- Added CA certificates to clickhouse-server docker image. #10476 (filimonov).

## Bug fix

- Fix error `the BloomFilter false positive must be a double number between 0 and 1` #10551. #10569 (Winter Zhang).

## ClickHouse release v20.3.8.53, 2020-04-23

### Bug Fix

- Fixed wrong behaviour of datetime functions for timezones that has altered between positive and negative offsets from UTC (e.g. Pacific/Kiritimati). This fixes #7202 #10369 (alexey-milovidov)
- Fix possible segfault with `distributed_group_by_no_merge` enabled (introduced in 20.3.7.46 by #10131). #10399 (Nikolai Kochetov)
- Fix wrong flattening of `Array(Tuple(...))` data types. This fixes #10259 #10390 (alexey-milovidov)
- Drop disks reservation in Aggregator. This fixes bug in disk space reservation, which may cause big external aggregation to fail even if it could be completed successfully #10375 (Azat Khuzhin)
- Fixed `DROP` vs `OPTIMIZE` race in `ReplicatedMergeTree`. `DROP` could leave some garbage in replica path in ZooKeeper if there was concurrent `OPTIMIZE` query. #10312 (tavplubix)
- Fix bug when server cannot attach table after column default was altered. #10441 (alesapin)
- Do not remove metadata directory when attach database fails before loading tables. #10442 (Winter Zhang)
- Fixed several bugs when some data was inserted with quorum, then deleted somehow (`DROP PARTITION, TTL`) and this leaded to the stuck of `INSERTs` or false-positive exceptions in `SELECTs`. This fixes #9946 #10188 (Nikita Mikhaylov)
- Fix possible `Pipeline` stuck error in `ConcatProcessor` which could have happened in remote query. #10381 (Nikolai Kochetov)
- Fixed wrong behavior in `HashTable` that caused compilation error when trying to read `HashMap` from buffer. #10386 (palasonic1)
- Allow to use `count(*)` with multiple JOINs. Fixes #9853 #10291 (Artem Zuikov)
- Prefer `fallback_to_stale_replicas` over `skip_unavailable_shards`, otherwise when both settings specified and there are no up-to-date replicas the query will fail (patch from @alex-zaitsev). Fixes: #2564. #10422 (Azat Khuzhin)
- Fix the issue when a query with `ARRAY JOIN`, `ORDER BY` and `LIMIT` may return incomplete result. This fixes #10226. Author: Vadim Plakhtinskiy. #10427 (alexey-milovidov)
- Check the number and type of arguments when creating BloomFilter index #9623 #10431 (Winter Zhang)

### Performance Improvement

- Improved performance of queries with explicitly defined sets at right side of `IN` operator and tuples in the left side. This fixes performance regression in version 20.3. #9740, #10385 (Anton Popov)

# ClickHouse release v20.3.7.46, 2020-04-17

## Bug Fix

- Fix Logical error: CROSS JOIN has expressions error for queries with comma and names joins mix. #10311 ([Artem Zuikov](#)).
- Fix queries with `max_bytes_before_external_group_by`. #10302 ([Artem Zuikov](#)).
- Fix move-to-prewhere optimization in presence of arrayJoin functions (in certain cases). This fixes #10092. #10195 ([alexey-milovidov](#)).
- Add the ability to relax the restriction on non-deterministic functions usage in mutations with `allow_nondeterministic_mutations` setting. #10186 ([filimonov](#)).

# ClickHouse release v20.3.6.40, 2020-04-16

## New Feature

- Added function `isConstant`. This function checks whether its argument is constant expression and returns 1 or 0. It is intended for development, debugging and demonstration purposes. #10198 ([alexey-milovidov](#)).

## Bug Fix

- Fix error Pipeline stuck with `max_rows_to_group_by` and `group_by_overflow_mode = 'break'`. #10279 ([Nikolai Kochetov](#)).
- Fix rare possible exception Cannot drain connections: cancel first #10239 ([Nikolai Kochetov](#)).
- Fixed bug where ClickHouse would throw "Unknown function lambda." error message when user tries to run ALTER UPDATE/DELETE on tables with ENGINE = Replicated\*. Check for nondeterministic functions now handles lambda expressions correctly. #10237 ([Alexander Kazakov](#)).
- Fixed "generateRandom" function for Date type. This fixes #9973. Fix an edge case when dates with year 2106 are inserted to MergeTree tables with old-style partitioning but partitions are named with year 1970. #10218 ([alexey-milovidov](#)).
- Convert types if the table definition of a View does not correspond to the SELECT query. This fixes #10180 and #10022. #10217 ([alexey-milovidov](#)).
- Fix `parseDateTimeBestEffort` for strings in RFC-2822 when day of week is Tuesday or Thursday. This fixes #10082. #10214 ([alexey-milovidov](#)).
- Fix column names of constants inside JOIN that may clash with names of constants outside of JOIN. #10207 ([alexey-milovidov](#)).
- Fix possible infinite query execution when the query actually should stop on LIMIT, while reading from infinite source like `system.numbers` or `system.zeros`. #10206 ([Nikolai Kochetov](#)).
- Fix using the current database for access checking when the database isn't specified. #10192 ([Vitaly Baranov](#)).
- Convert blocks if structure does not match on INSERT into Distributed(). #10135 ([Azat Khuzhin](#)).
- Fix possible incorrect result for extremes in processors pipeline. #10131 ([Nikolai Kochetov](#)).
- Fix some kinds of alters with compact parts. #10130 ([Anton Popov](#)).
- Fix incorrect `index_granularity_bytes` check while creating new replica. Fixes #10098. #10121 ([alesapin](#)).

- Fix SIGSEGV on INSERT into Distributed table when its structure differs from the underlying tables. #10105 (Azat Khuzhin).
- Fix possible rows loss for queries with JOIN and UNION ALL. Fixes #9826, #10113. #10099 (Nikolai Kochetov).
- Fixed replicated tables startup when updating from an old ClickHouse version where /table/replicas/replica\_name/metadata node does not exist. Fixes #10037. #10095 (alesapin).
- Add some arguments check and support identifier arguments for MySQL Database Engine. #10077 (Winter Zhang).
- Fix bug in clickhouse dictionary source from localhost clickhouse server. The bug may lead to memory corruption if types in dictionary and source are not compatible. #10071 (alesapin).
- Fix bug in CHECK TABLE query when table contain skip indices. #10068 (alesapin).
- Fix error Cannot clone block with columns because block has 0 columns ... While executing GroupingAggregatedTransform. It happened when setting distributed\_aggregation\_memory\_efficient was enabled, and distributed query read aggregating data with different level from different shards (mixed single and two level aggregation). #10063 (Nikolai Kochetov).
- Fix a segmentation fault that could occur in GROUP BY over string keys containing trailing zero bytes (#8636, #8925). #10025 (Alexander Kuzmenkov).
- Fix the number of threads used for remote query execution (performance regression, since 20.3). This happened when query from Distributed table was executed simultaneously on local and remote shards. Fixes #9965. #9971 (Nikolai Kochetov).
- Fix bug in which the necessary tables weren't retrieved at one of the processing stages of queries to some databases. Fixes #9699. #9949 (achulkov2).
- Fix 'Not found column in block' error when JOIN appears with TOTALS. Fixes #9839. #9939 (Artem Zuikov).
- Fix a bug with ON CLUSTER DDL queries freezing on server startup. #9927 (Gagan Arneja).
- Fix parsing multiple hosts set in the CREATE USER command, e.g. CREATE USER user6 HOST NAME REGEXP 'lo.??host', NAME REGEXP 'lo\*host'. #9924 (Vitaly Baranov).
- Fix TRUNCATE for Join table engine (#9917). #9920 (Amos Bird).
- Fix "scalar does not exist" error in ALTERs (#9878). #9904 (Amos Bird).
- Fix race condition between drop and optimize in ReplicatedMergeTree. #9901 (alesapin).
- Fix error with qualified names in distributed\_product\_mode='local'. Fixes #4756. #9891 (Artem Zuikov).
- Fix calculating grants for introspection functions from the setting 'allow\_introspection\_functions'. #9840 (Vitaly Baranov).

## Build/Testing/Packaging Improvement

- Fix integration test test\_settings\_constraints. #9962 (Vitaly Baranov).
- Removed dependency on clock\_getres. #9833 (alexey-milovidov).

ClickHouse release v20.3.5.21, 2020-03-27

Bug Fix

- Fix 'Different expressions with the same alias' error when query has PREWHERE and WHERE on distributed table and `SET distributed_product_mode = 'local'`. #9871 (Artem Zuikov).
- Fix mutations excessive memory consumption for tables with a composite primary key. This fixes #9850. #9860 (alesapin).
- For INSERT queries shard now clamps the settings got from the initiator to the shard's constraints instead of throwing an exception. This fix allows to send INSERT queries to a shard with another constraints. This change improves fix #9447. #9852 (Vitaly Baranov).
- Fix 'COMMA to CROSS JOIN rewriter is not enabled or cannot rewrite query' error in case of subqueries with COMMA JOIN out of tables lists (i.e. in WHERE). Fixes #9782. #9830 (Artem Zuikov).
- Fix possible exception `Got 0 in totals chunk, expected 1` on client. It happened for queries with `JOIN` in case if right joined table had zero rows. Example: `select * from system.one t1 join system.one t2 on t1.dummy = t2.dummy limit 0 FORMAT TabSeparated;`. Fixes #9777. #9823 (Nikolai Kochetov).
- Fix SIGSEGV with `optimize_skip_unused_shards` when type cannot be converted. #9804 (Azat Khuzhin).
- Fix broken `ALTER TABLE DELETE COLUMN` query for compact parts. #9779 (alesapin).
- Fix `max_distributed_connections` (w/ and w/o Processors). #9673 (Azat Khuzhin).
- Fixed a few cases when timezone of the function argument wasn't used properly. #9574 (Vasily Nemkov).

## Improvement

- Remove order by stage from mutations because we read from a single ordered part in a single thread. Also add check that the order of rows in mutation is ordered in sorting key order and this order is not violated. #9886 (alesapin).

## ClickHouse release v20.3.4.10, 2020-03-20

### Bug Fix

- This release also contains all bug fixes from 20.1.8.41
- Fix missing `rows_before_limit_at_least` for queries over http (with processors pipeline). This fixes #9730. #9757 (Nikolai Kochetov)

## ClickHouse release v20.3.3.6, 2020-03-17

### Bug Fix

- This release also contains all bug fixes from 20.1.7.38
- Fix bug in a replication that does not allow replication to work if the user has executed mutations on the previous version. This fixes #9645. #9652 (alesapin). It makes version 20.3 backward compatible again.
- Add setting `use_compact_format_in_distributed_parts_names` which allows to write files for `INSERT` queries into `Distributed` table with more compact format. This fixes #9647. #9653 (alesapin). It makes version 20.3 backward compatible again.

## ClickHouse release v20.3.2.1, 2020-03-12

### Backward Incompatible Change

- Fixed the issue file name too long when sending data for `Distributed` tables for a large number of replicas. Fixed the issue that replica credentials were exposed in the server log. The format of directory name on disk was changed to `[shard{shard_index}][_replica{replica_index}]]`. #8911 (Mikhail Korotov) After you upgrade to the new version, you will not be able to downgrade without manual intervention, because old server version does not recognize the new directory format. If you want to downgrade, you have to manually rename the corresponding directories to the old format. This change is relevant only if you have used asynchronous `INSERTs` to `Distributed` tables. In the version 20.3.3 we will introduce a setting that will allow you to enable the new format gradually.
- Changed the format of replication log entries for mutation commands. You have to wait for old mutations to process before installing the new version.
- Implement simple memory profiler that dumps stacktraces to `system.trace_log` every N bytes over soft allocation limit #8765 (Ivan) #9472 (alexey-milovidov) The column of `system.trace_log` was renamed from `timer_type` to `trace_type`. This will require changes in third-party performance analysis and flamegraph processing tools.
- Use OS thread id everywhere instead of internal thread number. This fixes #7477 Old `clickhouse-client` cannot receive logs that are send from the server when the setting `send_logs_level` is enabled, because the names and types of the structured log messages were changed. On the other hand, different server versions can send logs with different types to each other. When you don't use the `send_logs_level` setting, you should not care. #8954 (alexey-milovidov)
- Remove `indexHint` function #9542 (alexey-milovidov)
- Remove `findClusterIndex`, `findClusterValue` functions. This fixes #8641. If you were using these functions, send an email to `clickhouse-feedback@yandex-team.com` #9543 (alexey-milovidov)
- Now it's not allowed to create columns or add columns with `SELECT` subquery as default expression. #9481 (alesapin)
- Require aliases for subqueries in `JOIN`. #9274 (Artem Zuikov)
- Improved `ALTER MODIFY/ADD` queries logic. Now you cannot `ADD` column without type, `MODIFY` default expression does not change type of column and `MODIFY` type does not loose default expression value. Fixes #8669. #9227 (alesapin)
- Require server to be restarted to apply the changes in logging configuration. This is a temporary workaround to avoid the bug where the server logs to a deleted log file (see #8696). #8707 (Alexander Kuzmenkov)
- The setting `experimental_use_processors` is enabled by default. This setting enables usage of the new query pipeline. This is internal refactoring and we expect no visible changes. If you will see any issues, set it to back zero. #8768 (alexey-milovidov)

## New Feature

- Add `Avro` and `AvroConfluent` input/output formats #8571 (Andrew Onyshchuk) #8957 (Andrew Onyshchuk) #8717 (alexey-milovidov)
- Multi-threaded and non-blocking updates of expired keys in `cache` dictionaries (with optional permission to read old ones). #8303 (Nikita Mikhaylov)
- Add query `ALTER ... MATERIALIZE TTL` It runs mutation that forces to remove expired data by TTL and recalculates meta-information about TTL in all parts. #8775 (Anton Popov)
- Switch from `HashJoin` to `MergeJoin` (on disk) if needed #9082 (Artem Zuikov)
- Added `MOVE PARTITION` command for `ALTER TABLE` #4729 #6168 (Guillaume Tassery)

- Reloading storage configuration from configuration file on the fly. #8594 ([Vladimir Chebotarev](#))
- Allowed to change `storage_policy` to not less rich one. #8107 ([Vladimir Chebotarev](#))
- Added support for globs/wildcards for S3 storage and table function. #8851 ([Vladimir Chebotarev](#))
- Implement `bitAnd`, `bitOr`, `bitXor`, `bitNot` for `FixedString(N)` datatype. #9091 ([Guillaume Tassery](#))
- Added function `bitCount`. This fixes #8702. #8708 ([alexey-milovidov](#)) #8749 ([ikopylov](#))
- Add `generateRandom` table function to generate random rows with given schema. Allows to populate arbitrary test table with data. #8994 ([Ilya Yatsishin](#))
- `JSONEachRowFormat`: support special case when objects enclosed in top-level array. #8860 ([Kruglov Pavel](#))
- Now it's possible to create a column with `DEFAULT` expression which depends on a column with default `ALIAS` expression. #9489 ([alesapin](#))
- Allow to specify `--limit` more than the source data size in `clickhouse-obfuscator`. The data will repeat itself with different random seed. #9155 ([alexey-milovidov](#))
- Added `groupArraySample` function (similar to `groupArray`) with reservoir sampling algorithm. #8286 ([Amos Bird](#))
- Now you can monitor the size of update queue in `cache/complex_key_cache` dictionaries via system metrics. #9413 ([Nikita Mikhaylov](#))
- Allow to use CRLF as a line separator in CSV output format with setting `output_format_csv_crlf_end_of_line` is set to 1 #8934 #8935 #8963 ([Mikhail Korotov](#))
- Implement more functions of the H3 API: `h3GetBaseCell`, `h3HexAreaM2`, `h3IndexesAreNeighbors`, `h3ToChildren`, `h3ToString` and `stringToH3` #8938 ([Nico Mandery](#))
- New setting introduced: `max_parser_depth` to control maximum stack size and allow large complex queries. This fixes #6681 and #7668. #8647 ([Maxim Smirnov](#))
- Add a setting `force_optimize_skip_unused_shards` setting to throw if skipping of unused shards is not possible #8805 ([Azat Khuzhin](#))
- Allow to configure multiple disks/volumes for storing data for send in `Distributed` engine #8756 ([Azat Khuzhin](#))
- Support storage policy (`<tmp_policy>`) for storing temporary data. #8750 ([Azat Khuzhin](#))
- Added X-ClickHouse-Exception-Code HTTP header that is set if exception was thrown before sending data. This implements #4971. #8786 ([Mikhail Korotov](#))
- Added function `ifNotFinite`. It is just a syntactic sugar: `ifNotFinite(x, y) = isFinite(x) ? x : y`. #8710 ([alexey-milovidov](#))
- Added `last_successful_update_time` column in `system.dictionaries` table #9394 ([Nikita Mikhaylov](#))
- Add `blockSerializedSize` function (size on disk without compression) #8952 ([Azat Khuzhin](#))
- Add function `moduloOrZero` #9358 ([hc2](#))
- Added system tables `system.zeros` and `system.zeros_mt` as well as tale functions `zeros()` and `zeros_mt()`. Tables (and table functions) contain single column with name `zero` and type `UInt8`. This column contains zeros. It is needed for test purposes as the fastest method to generate many rows. This fixes #6604 #9593 ([Nikolai Kochetov](#))

## Experimental Feature

- Add new compact format of parts in MergeTree-family tables in which all columns are stored in one file. It helps to increase performance of small and frequent inserts. The old format (one file per column) is now called wide. Data storing format is controlled by settings `min_bytes_for_wide_part` and `min_rows_for_wide_part`. #8290 (Anton Popov)
- Support for S3 storage for Log, TinyLog and StripeLog tables. #8862 (Pavel Kovalenko)

## Bug Fix

- Fixed inconsistent whitespaces in log messages. #9322 (alexey-milovidov)
- Fix bug in which arrays of unnamed tuples were flattened as Nested structures on table creation. #8866 (achulkov2)
- Fixed the issue when "Too many open files" error may happen if there are too many files matching glob pattern in `File` table or `file` table function. Now files are opened lazily. This fixes #8857 #8861 (alexey-milovidov)
- DROP TEMPORARY TABLE now drops only temporary table. #8907 (Vitaly Baranov)
- Remove outdated partition when we shutdown the server or DETACH/ATTACH a table. #8602 (Guillaume Tassery)
- For how the default disk calculates the free space from `data` subdirectory. Fixed the issue when the amount of free space is not calculated correctly if the `data` directory is mounted to a separate device (rare case). This fixes #7441 #9257 (Mikhail Korotov)
- Allow comma (cross) join with IN () inside. #9251 (Artem Zuikov)
- Allow to rewrite CROSS to INNER JOIN if there's [NOT] LIKE operator in WHERE section. #9229 (Artem Zuikov)
- Fix possible incorrect result after GROUP BY with enabled setting `distributed_aggregation_memory_efficient`. Fixes #9134. #9289 (Nikolai Kochetov)
- Found keys were counted as missed in metrics of cache dictionaries. #9411 (Nikita Mikhaylov)
- Fix replication protocol incompatibility introduced in #8598. #9412 (alesapin)
- Fixed race condition on `queue_task_handle` at the startup of ReplicatedMergeTree tables. #9552 (alexey-milovidov)
- The token NOT did not work in SHOW TABLES NOT LIKE query #8727 #8940 (alexey-milovidov)
- Added range check to function `h3EdgeLengthM`. Without this check, buffer overflow is possible. #8945 (alexey-milovidov)
- Fixed up a bug in batched calculations of ternary logical OPs on multiple arguments (more than 10). #8718 (Alexander Kazakov)
- Fix error of PREWHERE optimization, which could lead to segfaults or Inconsistent number of columns got from `MergeTreeRangeReader` exception. #9024 (Anton Popov)
- Fix unexpected Timeout exceeded while reading from socket exception, which randomly happens on secure connection before timeout actually exceeded and when query profiler is enabled. Also add `connect_timeout_with_failover_secure_ms` settings (default 100ms), which is similar to `connect_timeout_with_failover_ms`, but is used for secure connections (because SSL handshake is slower, than ordinary TCP connection) #9026 (tavplubix)
- Fix bug with mutations finalization, when mutation may hang in state with `parts_to_do=0` and `is_done=0`. #9022 (alesapin)

- Use new ANY JOIN logic with `partial_merge_join` setting. It's possible to make ANY|ALL|SEMI LEFT and ALL INNER joins with `partial_merge_join=1` now. #8932 (Artem Zuikov)
- Shard now clamps the settings got from the initiator to the shard's constraints instead of throwing an exception. This fix allows to send queries to a shard with another constraints. #9447 (Vitaly Baranov)
- Fixed memory management problem in `MergeTreeReadPool`. #8791 (Vladimir Chebotarev)
- Fix `toDecimal*OrNull()` functions family when called with string e. Fixes #8312 #8764 (Artem Zuikov)
- Make sure that `FORMAT Null` sends no data to the client. #8767 (Alexander Kuzmenkov)
- Fix bug that timestamp in `LiveViewBlockInputStream` will not updated. LIVE VIEW is an experimental feature. #8644 (vxider) #8625 (vxider)
- Fixed `ALTER MODIFY TTL` wrong behavior which did not allow to delete old TTL expressions. #8422 (Vladimir Chebotarev)
- Fixed UBSan report in `MergeTreeIndexSet`. This fixes #9250 #9365 (alexey-milovidov)
- Fixed the behaviour of `match` and `extract` functions when haystack has zero bytes. The behaviour was wrong when haystack was constant. This fixes #9160 #9163 (alexey-milovidov) #9345 (alexey-milovidov)
- Avoid throwing from destructor in Apache Avro 3rd-party library. #9066 (Andrew Onyshchuk)
- Don't commit a batch polled from `Kafka` partially as it can lead to holes in data. #8876 (filimonov)
- Fix `joinGet` with nullable return types. #8919 #9014 (Amos Bird)
- Fix data incompatibility when compressed with T64 codec. #9016 (Artem Zuikov) Fix data type ids in T64 compression codec that leads to wrong (de)compression in affected versions. #9033 (Artem Zuikov)
- Add setting `enable_early_constant_folding` and disable it in some cases that leads to errors. #9010 (Artem Zuikov)
- Fix pushdown predicate optimizer with `VIEW` and enable the test #9011 (Winter Zhang)
- Fix segfault in `Merge` tables, that can happen when reading from `File` storages #9387 (tavplubix)
- Added a check for storage policy in `ATTACH PARTITION FROM`, `REPLACE PARTITION`, `MOVE TO TABLE`. Otherwise it could make data of part inaccessible after restart and prevent ClickHouse to start. #9383 (Vladimir Chebotarev)
- Fix alters if there is TTL set for table. #8800 (Anton Popov)
- Fix race condition that can happen when `SYSTEM RELOAD ALL DICTIONARIES` is executed while some dictionary is being modified/added/removed. #8801 (Vitaly Baranov)
- In previous versions `Memory` database engine use empty data path, so tables are created in `path` directory (e.g. `/var/lib/clickhouse/`), not in data directory of database (e.g. `/var/lib/clickhouse/db_name`). #8753 (tavplubix)
- Fixed wrong log messages about missing default disk or policy. #9530 (Vladimir Chebotarev)
- Fix `not(has())` for the `bloom_filter` index of array types. #9407 (achimbab)
- Allow first column(s) in a table with `Log` engine be an alias #9231 (Ivan)
- Fix order of ranges while reading from `MergeTree` table in one thread. It could lead to exceptions from `MergeTreeRangeReader` or wrong query results. #9050 (Anton Popov)

- Make `reinterpretAsFixedString` to return `FixedString` instead of `String`. #9052 (Andrew Onyshchuk)
- Avoid extremely rare cases when the user can get wrong error message (`Success` instead of detailed error description). #9457 (alexey-milovidov)
- Do not crash when using `Template` format with empty row template. #8785 (Alexander Kuzmenkov)
- Metadata files for system tables could be created in wrong place #8653 (tavplubix) Fixes #8581.
- Fix data race on `exception_ptr` in cache dictionary #8303. #9379 (Nikita Mikhaylov)
- Do not throw an exception for query `ATTACH TABLE IF NOT EXISTS`. Previously it was thrown if table already exists, despite the `IF NOT EXISTS` clause. #8967 (Anton Popov)
- Fixed missing closing paren in exception message. #8811 (alexey-milovidov)
- Avoid message `Possible deadlock avoided` at the startup of `clickhouse-client` in interactive mode. #9455 (alexey-milovidov)
- Fixed the issue when padding at the end of base64 encoded value can be malformed. Update base64 library. This fixes #9491, closes #9492 #9500 (alexey-milovidov)
- Prevent losing data in `Kafka` in rare cases when exception happens after reading suffix but before commit. Fixes #9378 #9507 (filimonov)
- Fixed exception in `DROP TABLE IF EXISTS` #8663 (Nikita Vasilev)
- Fix crash when a user tries to `ALTER MODIFY SETTING` for old-formatted `MergeTree` table engines family. #9435 (alesapin)
- Support for `UInt64` numbers that don't fit in `Int64` in JSON-related functions. Update SIMDJSON to master. This fixes #9209 #9344 (alexey-milovidov)
- Fixed execution of inverted predicates when non-strictly monotonic functional index is used. #9223 (Alexander Kazakov)
- Don't try to fold `IN` constant in `GROUP BY` #8868 (Amos Bird)
- Fix bug in `ALTER DELETE` mutations which leads to index corruption. This fixes #9019 and #8982. Additionally fix extremely rare race conditions in `ReplicatedMergeTree` `ALTER` queries. #9048 (alesapin)
- When the setting `compile_expressions` is enabled, you can get `unexpected column` in `LLVMEExecutableFunction` when we use `Nullable` type #8910 (Guillaume Tassery)
- Multiple fixes for `Kafka` engine: 1) fix duplicates that were appearing during consumer group rebalance. 2) Fix rare 'holes' appeared when data were polled from several partitions with one poll and committed partially (now we always process / commit the whole polled block of messages). 3) Fix flushes by block size (before that only flushing by timeout was working properly). 4) better subscription procedure (with assignment feedback). 5) Make tests work faster (with default intervals and timeouts). Due to the fact that data was not flushed by block size before (as it should according to documentation), that PR may lead to some performance degradation with default settings (due to more often & tinier flushes which are less optimal). If you encounter the performance issue after that change - please increase `kafka_max_block_size` in the table to the bigger value (for example `CREATE TABLE ... Engine=Kafka ... SETTINGS ... kafka_max_block_size=524288`). Fixes #7259 #8917 (filimonov)
- Fix Parameter out of bound exception in some queries after `PREWHERE` optimizations. #8914 (Baudouin Giard)
- Fixed the case of mixed-constness of arguments of function `arrayZip`. #8705 (alexey-milovidov)

- When executing `CREATE` query, fold constant expressions in storage engine arguments. Replace empty database name with current database. Fixes #6508, #3492 #9262 (tavplubix)
- Now it's not possible to create or add columns with simple cyclic aliases like a `DEFAULT b, b DEFAULT a.` #9603 (alesapin)
- Fixed a bug with double move which may corrupt original part. This is relevant if you use `ALTER TABLE MOVE` #8680 (Vladimir Chebotarev)
- Allow `interval` identifier to correctly parse without backticks. Fixed issue when a query cannot be executed even if the `interval` identifier is enclosed in backticks or double quotes. This fixes #9124, #9142 (alexey-milovidov)
- Fixed fuzz test and incorrect behaviour of `bitTestAll/bitTestAny` functions. #9143 (alexey-milovidov)
- Fix possible crash/wrong number of rows in `LIMIT n WITH TIES` when there are a lot of rows equal to n'th row. #9464 (tavplubix)
- Fix mutations with parts written with enabled `insert_quorum`. #9463 (alesapin)
- Fix data race at destruction of `Poco::HTTPServer`. It could happen when server is started and immediately shut down. #9468 (Anton Popov)
- Fix bug in which a misleading error message was shown when running `SHOW CREATE TABLE a_table_that_does_not_exist`. #8899 (achulkov2)
- Fixed Parameters are out of bound exception in some rare cases when we have a constant in the `SELECT` clause when we have an `ORDER BY` and a `LIMIT` clause. #8892 (Guillaume Tassery)
- Fix mutations finalization, when already done mutation can have status `is_done=0`. #9217 (alesapin)
- Prevent from executing `ALTER ADD INDEX` for MergeTree tables with old syntax, because it does not work. #8822 (Mikhail Korotov)
- During server startup do not access table, which `LIVE VIEW` depends on, so server will be able to start. Also remove `LIVE VIEW` dependencies when detaching `LIVE VIEW`. `LIVE VIEW` is an experimental feature. #8824 (tavplubix)
- Fix possible segfault in `MergeTreeRangeReader`, while executing `PREWHERE`. #9106 (Anton Popov)
- Fix possible mismatched checksums with column TTLs. #9451 (Anton Popov)
- Fixed a bug when parts were not being moved in background by TTL rules in case when there is only one volume. #8672 (Vladimir Chebotarev)
- Fixed the issue Method `createColumn()` is not implemented for data type `Set` This fixes #7799, #8674 (alexey-milovidov)
- Now we will try finalize mutations more frequently. #9427 (alesapin)
- Fix `intDiv` by minus one constant #9351 (hcza)
- Fix possible race condition in `BlockIO`. #9356 (Nikolai Kochetov)
- Fix bug leading to server termination when trying to use / drop `Kafka` table created with wrong parameters. #9513 (filimonov)
- Added workaround if OS returns wrong result for `timer_create` function. #8837 (alexey-milovidov)
- Fixed error in usage of `min_marks_for_seek` parameter. Fixed the error message when there is no sharding key in Distributed table and we try to skip unused shards. #8908 (Azat Khuzhin)

## Improvement

- Implement ALTER MODIFY/DROP queries on top of mutations for ReplicatedMergeTree\* engines family. Now ALTERS blocks only at the metadata update stage, and don't block after that. [#8701](#) ([alesapin](#))
- Add ability to rewrite CROSS to INNER JOINs with WHERE section containing unqualified names. [#9512](#) ([Artem Zuikov](#))
- Make SHOW TABLES and SHOW DATABASES queries support the WHERE expressions and FROM/IN [#9076](#) ([sundyli](#))
- Added a setting deduplicate\_blocks\_in\_dependent\_materialized\_views. [#9070](#) ([urykhy](#))
- After recent changes MySQL client started to print binary strings in hex thereby making them not readable ([#9032](#)). The workaround in ClickHouse is to mark string columns as UTF-8, which is not always, but usually the case. [#9079](#) ([Yuriy Baranov](#))
- Add support of String and FixedString keys for sumMap [#8903](#) ([Baudouin Giard](#))
- Support string keys in SummingMergeTree maps [#8933](#) ([Baudouin Giard](#))
- Signal termination of thread to the thread pool even if the thread has thrown exception [#8736](#) ([Ding Xiang Fei](#))
- Allow to set query\_id in clickhouse-benchmark [#9416](#) ([Anton Popov](#))
- Don't allow strange expressions in ALTER TABLE ... PARTITION partition query. This addresses [#7192](#) [#8835](#) ([alexey-milovidov](#))
- The table system.table\_engines now provides information about feature support (like supports\_ttl or supports\_sort\_order). [#8830](#) ([Max Akhmedov](#))
- Enable system.metric\_log by default. It will contain rows with values of ProfileEvents, CurrentMetrics collected with "collect\_interval\_milliseconds" interval (one second by default). The table is very small (usually in order of megabytes) and collecting this data by default is reasonable. [#9225](#) ([alexey-milovidov](#))
- Initialize query profiler for all threads in a group, e.g. it allows to fully profile insert-queries. Fixes [#6964](#) [#8874](#) ([Ivan](#))
- Now temporary LIVE VIEW is created by CREATE LIVE VIEW name WITH TIMEOUT [42] ... instead of CREATE TEMPORARY LIVE VIEW ..., because the previous syntax was not consistent with CREATE TEMPORARY TABLE ... [#9131](#) ([tavplubix](#))
- Add text\_log.level configuration parameter to limit entries that goes to system.text\_log table [#8809](#) ([Azat Khuzhin](#))
- Allow to put downloaded part to a disks/volumes according to TTL rules [#8598](#) ([Vladimir Chebotarev](#))
- For external MySQL dictionaries, allow to mutualize MySQL connection pool to "share" them among dictionaries. This option significantly reduces the number of connections to MySQL servers. [#9409](#) ([Clément Rodriguez](#))
- Show nearest query execution time for quantiles in clickhouse-benchmark output instead of interpolated values. It's better to show values that correspond to the execution time of some queries. [#8712](#) ([alexey-milovidov](#))
- Possibility to add key & timestamp for the message when inserting data to Kafka. Fixes [#7198](#) [#8969](#) ([filimonov](#))

- If server is run from terminal, highlight thread number, query id and log priority by colors. This is for improved readability of correlated log messages for developers. [#8961](#) ([alexey-milovidov](#))
- Better exception message while loading tables for `Ordinary` database. [#9527](#) ([alexey-milovidov](#))
- Implement `arraySlice` for arrays with aggregate function states. This fixes [#9388](#) [#9391](#) ([alexey-milovidov](#))
- Allow constant functions and constant arrays to be used on the right side of IN operator. [#8813](#) ([Anton Popov](#))
- If zookeeper exception has happened while fetching data for `system.replicas`, display it in a separate column. This implements [#9137](#) [#9138](#) ([alexey-milovidov](#))
- Atomically remove MergeTree data parts on destroy. [#8402](#) ([Vladimir Chebotarev](#))
- Support row-level security for Distributed tables. [#8926](#) ([Ivan](#))
- Now we recognize suffix (like KB, KiB...) in settings values. [#8072](#) ([Mikhail Korotov](#))
- Prevent out of memory while constructing result of a large JOIN. [#8637](#) ([Artem Zuikov](#))
- Added names of clusters to suggestions in interactive mode in `clickhouse-client`. [#8709](#) ([alexey-milovidov](#))
- Initialize query profiler for all threads in a group, e.g. it allows to fully profile insert-queries [#8820](#) ([Ivan](#))
- Added column `exception_code` in `system.query_log` table. [#8770](#) ([Mikhail Korotov](#))
- Enabled MySQL compatibility server on port `9004` in the default server configuration file. Fixed password generation command in the example in configuration. [#8771](#) ([Yuriy Baranov](#))
- Prevent abort on shutdown if the filesystem is readonly. This fixes [#9094](#) [#9100](#) ([alexey-milovidov](#))
- Better exception message when length is required in HTTP POST query. [#9453](#) ([alexey-milovidov](#))
- Add `_path` and `_file` virtual columns to HDFS and File engines and `hdfs` and `file` table functions [#8489](#) ([Olga Khvostikova](#))
- Fix error `Cannot find column` while inserting into `MATERIALIZED VIEW` in case if new column was added to view's internal table. [#8766](#) [#8788](#) ([vzakaznikov](#)) [#8788](#) [#8806](#) ([Nikolai Kochetov](#)) [#8803](#) ([Nikolai Kochetov](#))
- Fix progress over native client-server protocol, by send progress after final update (like logs). This may be relevant only to some third-party tools that are using native protocol. [#9495](#) ([Azat Khuzhin](#))
- Add a system metric tracking the number of client connections using MySQL protocol ([#9013](#)). [#9015](#) ([Eugene Klimov](#))
- From now on, HTTP responses will have `X-ClickHouse-Timezone` header set to the same timezone value that `SELECT timezone()` would report. [#9493](#) ([Denis Glazachev](#))

## Performance Improvement

- Improve performance of analysing index with IN [#9261](#) ([Anton Popov](#))
- Simpler and more efficient code in Logical Functions + code cleanups. A followup to [#8718](#) [#8728](#) ([Alexander Kazakov](#))
- Overall performance improvement (in range of 5%..200% for affected queries) by ensuring even more strict aliasing with C++20 features. [#9304](#) ([Amos Bird](#))
- More strict aliasing for inner loops of comparison functions. [#9327](#) ([alexey-milovidov](#))

- More strict aliasing for inner loops of arithmetic functions. #9325 (alexey-milovidov)
- A ~3 times faster implementation for `ColumnVector::replicate()`, via which `ColumnConst::convertToFullColumn()` is implemented. Also will be useful in tests when materializing constants. #9293 (Alexander Kazakov)
- Another minor performance improvement to `ColumnVector::replicate()` (this speeds up the `materialize` function and higher order functions) an even further improvement to #9293 #9442 (Alexander Kazakov)
- Improved performance of `stochasticLinearRegression` aggregate function. This patch is contributed by Intel. #8652 (alexey-milovidov)
- Improve performance of `reinterpretAsFixedString` function. #9342 (alexey-milovidov)
- Do not send blocks to client for `Null` format in processors pipeline. #8797 (Nikolai Kochetov) #8767 (Alexander Kuzmenkov)

## Build/Testing/Packaging Improvement

- Exception handling now works correctly on Windows Subsystem for Linux. See <https://github.com/ClickHouse-Extras/libunwind/pull/3> This fixes #6480 #9564 (sobolevsv)
- Replace `readline` with `replxx` for interactive line editing in `clickhouse-client` #8416 (Ivan)
- Better build time and less template instantiations in `FunctionsComparison`. #9324 (alexey-milovidov)
- Added integration with `clang-tidy` in CI. See also #6044 #9566 (alexey-milovidov)
- Now we link ClickHouse in CI using `lld` even for `gcc`. #9049 (alesapin)
- Allow to randomize thread scheduling and insert glitches when `THREAD_FUZZER_*` environment variables are set. This helps testing. #9459 (alexey-milovidov)
- Enable secure sockets in stateless tests #9288 (tavplubix)
- Make `SPLIT_SHARED_LIBRARIES=OFF` more robust #9156 (Azat Khuzhin)
- Make "performance\_introspection\_and\_logging" test reliable to random server stuck. This may happen in CI environment. See also #9515 #9528 (alexey-milovidov)
- Validate XML in style check. #9550 (alexey-milovidov)
- Fixed race condition in test `00738_lock_for_inner_table`. This test relied on sleep. #9555 (alexey-milovidov)
- Remove performance tests of type `once`. This is needed to run all performance tests in statistical comparison mode (more reliable). #9557 (alexey-milovidov)
- Added performance test for arithmetic functions. #9326 (alexey-milovidov)
- Added performance test for `sumMap` and `sumMapWithOverflow` aggregate functions. Follow-up for #8933 #8947 (alexey-milovidov)
- Ensure style of `ErrorCodes` by style check. #9370 (alexey-milovidov)
- Add script for tests history. #8796 (alesapin)
- Add GCC warning `-Wsuggest-override` to locate and fix all places where `override` keyword must be used. #8760 (kreuzerkrieg)
- Ignore weak symbol under Mac OS X because it must be defined #9538 (Deleted user)
- Normalize running time of some queries in performance tests. This is done in preparation to run all the performance tests in comparison mode. #9565 (alexey-milovidov)

- Fix some tests to support pytest with query tests [#9062](#) ([Ivan](#))
- Enable SSL in build with MSan, so server will not fail at startup when running stateless tests [#9531](#) ([tavplubix](#))
- Fix database substitution in test results [#9384](#) ([Ilya Yatsishin](#))
- Build fixes for miscellaneous platforms [#9381](#) ([proller](#)) [#8755](#) ([proller](#)) [#8631](#) ([proller](#))
- Added disks section to stateless-with-coverage test docker image [#9213](#) ([Pavel Kovalenko](#))
- Get rid of in-source-tree files when building with GRPC [#9588](#) ([Amos Bird](#))
- Slightly faster build time by removing SessionCleaner from Context. Make the code of SessionCleaner more simple. [#9232](#) ([alexey-milovidov](#))
- Updated checking for hung queries in clickhouse-test script [#8858](#) ([Alexander Kazakov](#))
- Removed some useless files from repository. [#8843](#) ([alexey-milovidov](#))
- Changed type of math perftests from `once` to `loop`. [#8783](#) ([Nikolai Kochetov](#))
- Add docker image which allows to build interactive code browser HTML report for our codebase. [#8781](#) ([alesapin](#)) See [Woboq Code Browser](#)
- Suppress some test failures under MSan. [#8780](#) ([Alexander Kuzmenkov](#))
- Speedup "exception while insert" test. This test often times out in debug-with-coverage build. [#8711](#) ([alexey-milovidov](#))
- Updated `libcxx` and `libcxxabi` to master. In preparation to [#9304](#) [#9308](#) ([alexey-milovidov](#))
- Fix flaky test `00910_zookeeper_test_alter_compression_codecs`. [#9525](#) ([alexey-milovidov](#))
- Clean up duplicated linker flags. Make sure the linker won't look up an unexpected symbol. [#9433](#) ([Amos Bird](#))
- Add `clickhouse-odbc` driver into test images. This allows to test interaction of ClickHouse with ClickHouse via its own ODBC driver. [#9348](#) ([filimonov](#))
- Fix several bugs in unit tests. [#9047](#) ([alesapin](#))
- Enable `-Wmissing/include-dirs` GCC warning to eliminate all non-existing includes - mostly as a result of CMake scripting errors [#8704](#) ([kreuzerkrieg](#))
- Describe reasons if query profiler cannot work. This is intended for [#9049](#) [#9144](#) ([alexey-milovidov](#))
- Update OpenSSL to upstream master. Fixed the issue when TLS connections may fail with the message `OpenSSL SSL_read: error:14094438:SSL routines:ssl3_read_bytes:tlsv1 alert internal error` and `SSL Exception: error:2400006E:random number generator::error retrieving entropy`. The issue was present in version 20.1. [#8956](#) ([alexey-milovidov](#))
- Update Dockerfile for server [#8893](#) ([Ilya Mazaev](#))
- Minor fixes in build-gcc-from-sources script [#8774](#) ([Michael Nacharov](#))
- Replace `numbers` to zeros in perftests where `number` column is not used. This will lead to more clean test results. [#9600](#) ([Nikolai Kochetov](#))
- Fix stack overflow issue when using `initializer_list` in Column constructors. [#9367](#) ([Deleted user](#))

- Upgrade librdkafka to v1.3.0. Enable bundled `rdkafka` and `gsasl` libraries on Mac OS X. #9000 (Andrew Onyshchuk)
- build fix on GCC 9.2.0 #9306 (vxider)

## ClickHouse release v20.1

### ClickHouse release v20.1.16.120-stable 2020-60-26

#### Bug Fix

- Fix rare crash caused by using `Nullable` column in prewhere condition. Continuation of #11608. #11869 (Nikolai Kochetov).
- Don't allow arrayJoin inside higher order functions. It was leading to broken protocol synchronization. This closes #3933. #11846 (alexey-milovidov).
- Fix unexpected behaviour of queries like `SELECT *, xyz.*` which were success while an error expected. #11753 (hexiaoting).
- Fixed `LOGICAL_ERROR` caused by wrong type deduction of complex literals in Values input format. #11732 (tavplubix).
- Fix `ORDER BY ... WITH FILL` over const columns. #11697 (Anton Popov).
- Pass proper timeouts when communicating with XDBC bridge. Recently timeouts were not respected when checking bridge liveness and receiving meta info. #11690 (alexey-milovidov).
- Add support for regular expressions with case-insensitive flags. This fixes #11101 and fixes #11506. #11649 (alexey-milovidov).
- Fix bloom filters for String (data skipping indices). #11638 (Azat Khuzhin).
- Fix rare crash caused by using `Nullable` column in prewhere condition. (Probably it is connected with #11572 somehow). #11608 (Nikolai Kochetov).
- Fix wrong exit code of the clickhouse-client, when `exception.code() % 256 = 0`. #11601 (filimonov).
- Fix trivial error in log message about "Mark cache size was lowered" at server startup. This closes #11399. #11589 (alexey-milovidov).
- Now clickhouse-server docker container will prefer IPv6 checking server aliveness. #11550 (Ivan Starkov).
- Fix memory leak when exception is thrown in the middle of aggregation with -State functions. This fixes #8995. #11496 (alexey-milovidov).
- Fix usage of primary key wrapped into a function with 'FINAL' modifier and 'ORDER BY' optimization. #10715 (Anton Popov).

### ClickHouse release v20.1.15.109-stable 2020-06-19

#### Bug Fix

- Fix excess lock for structure during alter. #11790 (alesapin).

### ClickHouse release v20.1.14.107-stable 2020-06-11

#### Bug Fix

- Fix error Size of offsets does not match size of column for queries with `PREWHERE` column in (subquery) and `ARRAY JOIN`. #11580 (Nikolai Kochetov).

# ClickHouse release v20.1.13.105-stable 2020-06-10

## Bug Fix

- Fix the error Data compressed with different methods that can happen if `min_bytes_to_use_direct_io` is enabled and PREWHERE is active and using SAMPLE or high number of threads. This fixes #11539. #11540 (alexey-milovidov).
- Fix return compressed size for codecs. #11448 (Nikolai Kochetov).
- Fix server crash when a column has compression codec with non-literal arguments. Fixes #11365. #11431 (alesapin).
- Fix pointInPolygon with nan as point. Fixes #11375. #11421 (Alexey Ilyukhov).
- Fixed geohashesInBox with arguments outside of latitude/longitude range. #11403 (Vasily Nemkov).
- Fix possible Pipeline stuck error for queries with external sort and limit. Fixes #11359. #11366 (Nikolai Kochetov).
- Fix crash in `quantilesExactWeightedArray`. #11337 (Nikolai Kochetov).
- Make writing to MATERIALIZED VIEW with setting `parallel_view_processing = 1` parallel again. Fixes #10241. #11330 (Nikolai Kochetov).
- Fix visitParamExtractRaw when extracted JSON has strings with unbalanced { or [. #11318 (Ewout).
- Fix very rare race condition in ThreadPool. #11314 (alexey-milovidov).
- Fix potential uninitialized memory in conversion. Example: `SELECT toIntervalSecond(now64())`. #11311 (alexey-milovidov).
- Fix the issue when index analysis cannot work if a table has Array column in primary key and if a query is filtering by this column with `empty` or `notEmpty` functions. This fixes #11286. #11303 (alexey-milovidov).
- Fix bug when query speed estimation can be incorrect and the limit of `min_execution_speed` may not work or work incorrectly if the query is throttled by `max_network_bandwidth`, `max_execution_speed` or `priority` settings. Change the default value of `timeout_before_checking_execution_speed` to non-zero, because otherwise the settings `min_execution_speed` and `max_execution_speed` have no effect. This fixes #11297. This fixes #5732. This fixes #6228. Usability improvement: avoid concatenation of exception message with progress bar in `clickhouse-client`. #11296 (alexey-milovidov).
- Fix crash while reading malformed data in Protobuf format. This fixes #5957, fixes #11203. #11258 (Vitaly Baranov).
- Fix possible error `Cannot capture column` for higher-order functions with `Array(Array(LowCardinality))` captured argument. #11185 (Nikolai Kochetov).
- If data skipping index is dependent on columns that are going to be modified during background merge (for SummingMergeTree, AggregatingMergeTree as well as for TTL GROUP BY), it was calculated incorrectly. This issue is fixed by moving index calculation after merge so the index is calculated on merged data. #11162 (Azat Khuzhin).
- Remove logging from mutation finalization task if nothing was finalized. #11109 (alesapin).
- Fixed `parseDateTime64BestEffort` argument resolution bugs. #10925. #11038 (Vasily Nemkov).
- Fix incorrect raw data size in method `getRawData()`. #10964 (Igr).
- Fix backward compatibility with tuples in Distributed tables. #10889 (Anton Popov).

- Fix SIGSEGV in StringHashTable (if such key does not exist). #10870 (Azat Khuzhin).
- Fixed bug in ReplicatedMergeTree which might cause some ALTER on OPTIMIZE query to hang waiting for some replica after it become inactive. #10849 (tavplubix).
- Fix columns order after Block::sortColumns() (also add a test that shows that it affects some real use case - Buffer engine). #10826 (Azat Khuzhin).
- Fix the issue with ODBC bridge when no quoting of identifiers is requested. This fixes #7984. #10821 (alexey-milovidov).
- Fix UBSan and MSan report in DateLUT. #10798 (alexey-milovidov).
- ▪ Make use of `src_type` for correct type conversion in key conditions. Fixes #6287. #10791 (Andrew Onyshchuk).
- Fix parallel\_view\_processing behavior. Now all insertions into MATERIALIZED VIEW without exception should be finished if exception happened. Fixes #10241. #10757 (Nikolai Kochetov).
- Fix combinator -OrNull and -OrDefault when combined with -State. #10741 (hcz).
- Fix disappearing totals. Totals could have been filtered if query had had join or subquery with external where condition. Fixes #10674. #10698 (Nikolai Kochetov).
- Fix multiple usages of IN operator with the identical set in one query. #10686 (Anton Popov).
- Fix order of parameters in AggregateTransform constructor. #10667 (palasonic1).
- Fix the lack of parallel execution of remote queries with `distributed_aggregation_memory_efficient` enabled. Fixes #10655. #10664 (Nikolai Kochetov).
- Fix predicates optimization for distributed queries (`enable_optimize_predicate_expression=1`) for queries with HAVING section (i.e. when filtering on the server initiator is required), by preserving the order of expressions (and this is enough to fix), and also force aggregator use column names over indexes. Fixes: #10613, #11413. #10621 (Azat Khuzhin).
- Fix error the BloomFilter false positive must be a double number between 0 and 1 #10551. #10569 (Winter Zhang).
- Fix SELECT of column ALIAS which default expression type different from column type. #10563 (Azat Khuzhin).
- ▪ Implemented comparison between DateTime64 and String values (just like for DateTime). #10560 (Vasily Nemkov).

## ClickHouse release v20.1.12.86, 2020-05-26

### Bug Fix

- Fixed incompatibility of two-level aggregation between versions 20.1 and earlier. This incompatibility happens when different versions of ClickHouse are used on initiator node and remote nodes and the size of GROUP BY result is large and aggregation is performed by a single String field. It leads to several unmerged rows for a single key in result. #10952 (alexey-milovidov).
- Fixed data corruption for LowCardinality(FixedString) key column in SummingMergeTree which could have happened after merge. Fixes #10489. #10721 (Nikolai Kochetov).
- Fixed bug, which causes http requests stuck on client close when `readonly=2` and `cancel_http_READONLY_queries_on_client_close=1`. Fixes #7939, #7019, #7736, #7091. #10684 (tavplubix).

- Fixed a bug when on `SYSTEM DROP DNS CACHE` query also drop caches, which are used to check if user is allowed to connect from some IP addresses. [#10608 \(tavplubix\)](#).
- Fixed incorrect scalar results inside inner query of `MATERIALIZED VIEW` in case if this query contained dependent table. [#10603 \(Nikolai Kochetov\)](#).
- Fixed the situation when mutation finished all parts, but hung up in `is_done=0`. [#10526 \(alesapin\)](#).
- Fixed overflow at beginning of unix epoch for timezones with fractional offset from UTC. This fixes [#9335](#). [#10513 \(alexey-milovidov\)](#).
- Fixed improper shutdown of Distributed storage. [#10491 \(Azat Khuzhin\)](#).
- Fixed numeric overflow in `simpleLinearRegression` over large integers. [#10474 \(hc2z\)](#).
- Fixed removing metadata directory when attach database fails. [#10442 \(Winter Zhang\)](#).
- Added a check of number and type of arguments when creating `BloomFilter` index [#9623](#). [#10431 \(Winter Zhang\)](#).
- Fixed the issue when a query with `ARRAY JOIN`, `ORDER BY` and `LIMIT` may return incomplete result. This fixes [#10226](#). [#10427 \(alexey-milovidov\)](#).
- Prefer `fallback_to_stale_replicas` over `skip_unavailable_shards`. [#10422 \(Azat Khuzhin\)](#).
- Fixed wrong flattening of `Array(Tuple(...))` data types. This fixes [#10259](#). [#10390 \(alexey-milovidov\)](#).
- Fixed wrong behavior in `HashTable` that caused compilation error when trying to read `HashMap` from buffer. [#10386 \(palasonic1\)](#).
- Fixed possible Pipeline stuck error in `ConcatProcessor` which could have happened in remote query. [#10381 \(Nikolai Kochetov\)](#).
- Fixed error Pipeline stuck with `max_rows_to_group_by` and `group_by_overflow_mode = 'break'`. [#10279 \(Nikolai Kochetov\)](#).
- Fixed several bugs when some data was inserted with quorum, then deleted somehow (`DROP PARTITION, TTL`) and this leaded to the stuck of `INSERTS` or false-positive exceptions in `SELECTs`. This fixes [#9946](#). [#10188 \(Nikita Mikhaylov\)](#).
- Fixed incompatibility when versions prior to 18.12.17 are used on remote servers and newer is used on initiating server, and GROUP BY both fixed and non-fixed keys, and when two-level group by method is activated. [#3254 \(alexey-milovidov\)](#).

## Build/Testing/Packaging Improvement

- Added CA certificates to `clickhouse-server` docker image. [#10476 \(filimonov\)](#).

## ClickHouse release v20.1.10.70, 2020-04-17

### Bug Fix

- Fix rare possible exception `Cannot drain connections: cancel first`. [#10239 \(Nikolai Kochetov\)](#).
- Fixed bug where ClickHouse would throw 'Unknown function lambda.' error message when user tries to run `ALTER UPDATE/DELETE` on tables with `ENGINE = Replicated*`. Check for nondeterministic functions now handles lambda expressions correctly. [#10237 \(Alexander Kazakov\)](#).
- Fix `parseDateTimeBestEffort` for strings in RFC-2822 when day of week is Tuesday or Thursday. This fixes [#10082](#). [#10214 \(alexey-milovidov\)](#).

- Fix column names of constants inside `JOIN` that may clash with names of constants outside of `JOIN`. [#10207 \(alexey-milovidov\)](#).
- Fix possible infinite query execution when the query actually should stop on `LIMIT`, while reading from infinite source like `system.numbers` or `system.zeros`. [#10206 \(Nikolai Kochetov\)](#).
- Fix move-to-prewhere optimization in presence of `arrayJoin` functions (in certain cases). This fixes [#10092](#). [#10195 \(alexey-milovidov\)](#).
- Add the ability to relax the restriction on non-deterministic functions usage in mutations with `allow_nondeterministic_mutations` setting. [#10186 \(filimonov\)](#).
- Convert blocks if structure does not match on `INSERT` into table with `Distributed` engine. [#10135 \(Azat Khuzhin\)](#).
- Fix `SIGSEGV` on `INSERT` into `Distributed` table when its structure differs from the underlying tables. [#10105 \(Azat Khuzhin\)](#).
- Fix possible rows loss for queries with `JOIN` and `UNION ALL`. Fixes [#9826](#), [#10113](#), [#10099 \(Nikolai Kochetov\)](#).
- Add arguments check and support identifier arguments for MySQL Database Engine. [#10077 \(Winter Zhang\)](#).
- Fix bug in clickhouse dictionary source from localhost clickhouse server. The bug may lead to memory corruption if types in dictionary and source are not compatible. [#10071 \(alesapin\)](#).
- Fix error `Cannot clone block with columns because block has 0 columns ... While executing GroupingAggregatedTransform`. It happened when setting `distributed_aggregation_memory_efficient` was enabled, and distributed query read aggregating data with different level from different shards (mixed single and two level aggregation). [#10063 \(Nikolai Kochetov\)](#).
- Fix a segmentation fault that could occur in `GROUP BY` over string keys containing trailing zero bytes ([#8636](#), [#8925](#)). [#10025 \(Alexander Kuzmenkov\)](#).
- Fix bug in which the necessary tables weren't retrieved at one of the processing stages of queries to some databases. Fixes [#9699](#), [#9949 \(achulkov2\)](#).
- Fix 'Not found column in block' error when `JOIN` appears with `TOTALS`. Fixes [#9839](#), [#9939 \(Artem Zuikov\)](#).
- Fix a bug with `ON CLUSTER` DDL queries freezing on server startup. [#9927 \(Gagan Arneja\)](#).
- Fix `TRUNCATE` for Join table engine ([#9917](#)). [#9920 \(Amos Bird\)](#).
- Fix 'scalar does not exist' error in `ALTER` queries ([#9878](#)). [#9904 \(Amos Bird\)](#).
- Fix race condition between drop and optimize in `ReplicatedMergeTree`. [#9901 \(alesapin\)](#).
- Fixed `DeleteOnDestroy` logic in `ATTACH PART` which could lead to automatic removal of attached part and added few tests. [#9410 \(Vladimir Chebotarev\)](#).

## Build/Testing/Packaging Improvement

- Fix unit test `collapsing_sorted_stream`. [#9367 \(Deleted user\)](#).

## ClickHouse release v20.1.9.54, 2020-03-28

### Bug Fix

- Fix 'Different expressions with the same alias' error when query has `PREWHERE` and `WHERE` on distributed table and `SET distributed_product_mode = 'local'`. [#9871 \(Artem Zuikov\)](#).

- Fix mutations excessive memory consumption for tables with a composite primary key. This fixes #9850. #9860 (alesapin).
- For INSERT queries shard now clamps the settings got from the initiator to the shard's constraints instead of throwing an exception. This fix allows to send INSERT queries to a shard with another constraints. This change improves fix #9447. #9852 (Vitaly Baranov).
- Fix possible exception Got 0 in totals chunk, expected 1 on client. It happened for queries with JOIN in case if right joined table had zero rows. Example: select \* from system.one t1 join system.one t2 on t1.dummy = t2.dummy limit 0 FORMAT TabSeparated;. Fixes #9777. #9823 (Nikolai Kochetov).
- Fix SIGSEGV with optimize\_skip\_unused\_shards when type cannot be converted. #9804 (Azat Khuzhin).
- Fixed a few cases when timezone of the function argument wasn't used properly. #9574 (Vasily Nemkov).

## Improvement

- Remove ORDER BY stage from mutations because we read from a single ordered part in a single thread. Also add check that the order of rows in mutation is ordered in sorting key order and this order is not violated. #9886 (alesapin).

## Build/Testing/Packaging Improvement

- Clean up duplicated linker flags. Make sure the linker won't look up an unexpected symbol. #9433 (Amos Bird).

# ClickHouse release v20.1.8.41, 2020-03-20

## Bug Fix

- Fix possible permanent Cannot schedule a task error (due to unhandled exception in ParallelAggregatingBlockInputStream::Handler::onFinish/onFinishThread). This fixes #6833. #9154 (Azat Khuzhin)
- Fix excessive memory consumption in ALTER queries (mutations). This fixes #9533 and #9670. #9754 (alesapin)
- Fix bug in backquoting in external dictionaries DDL. This fixes #9619. #9734 (alesapin)

# ClickHouse release v20.1.7.38, 2020-03-18

## Bug Fix

- Fixed incorrect internal function names for sumKahan and sumWithOverflow. It lead to exception while using this functions in remote queries. #9636 (Azat Khuzhin). This issue was in all ClickHouse releases.
- Allow ALTER ON CLUSTER of Distributed tables with internal replication. This fixes #3268. #9617 (shinoi2). This issue was in all ClickHouse releases.
- Fix possible exceptions Size of filter does not match size of column and Invalid number of rows in Chunk in MergeTreeRangeReader. They could appear while executing PREWHERE in some cases. Fixes #9132. #9612 (Anton Popov)
- Fixed the issue: timezone was not preserved if you write a simple arithmetic expression like time + 1 (in contrast to an expression like time + INTERVAL 1 SECOND). This fixes #5743. #9323 (alexey-milovidov). This issue was in all ClickHouse releases.
- Now it's not possible to create or add columns with simple cyclic aliases like a DEFAULT b, b DEFAULT a. #9603 (alesapin)

- Fixed the issue when padding at the end of base64 encoded value can be malformed. Update base64 library. This fixes #9491, closes #9492 #9500 (alexey-milovidov)
- Fix data race at destruction of `Poco::HTTPServer`. It could happen when server is started and immediately shut down. #9468 (Anton Popov)
- Fix possible crash/wrong number of rows in `LIMIT n WITH TIES` when there are a lot of rows equal to n'th row. #9464 (tavplubix)
- Fix possible mismatched checksums with column TTLs. #9451 (Anton Popov)
- Fix crash when a user tries to `ALTER MODIFY SETTING` for old-formatted `MergeTree` table engines family. #9435 (alesapin)
- Now we will try finalize mutations more frequently. #9427 (alesapin)
- Fix replication protocol incompatibility introduced in #8598. #9412 (alesapin)
- Fix `not(has())` for the `bloom_filter` index of array types. #9407 (achimbab)
- Fixed the behaviour of `match` and `extract` functions when haystack has zero bytes. The behaviour was wrong when haystack was constant. This fixes #9160 #9163 (alexey-milovidov) #9345 (alexey-milovidov)

## Build/Testing/Packaging Improvement

- Exception handling now works correctly on Windows Subsystem for Linux. See <https://github.com/ClickHouse-Extras/libunwind/pull/3> This fixes #6480 #9564 (sobolevsv)

## ClickHouse release v20.1.6.30, 2020-03-05

### Bug Fix

- Fix data incompatibility when compressed with `T64` codec. #9039 (abyss7)
- Fix order of ranges while reading from `MergeTree` table in one thread. Fixes #8964. #9050 (Curtizj)
- Fix possible segfault in `MergeTreeRangeReader`, while executing `PREWHERE`. Fixes #9064. #9106 (Curtizj)
- Fix `reinterpretAsFixedString` to return `FixedString` instead of `String`. #9052 (oandrew)
- Fix `joinGet` with nullable return types. Fixes #8919 #9014 (amosbird)
- Fix fuzz test and incorrect behaviour of `bitTestAll`/`bitTestAny` functions. #9143 (alexey-milovidov)
- Fix the behaviour of `match` and `extract` functions when haystack has zero bytes. The behaviour was wrong when haystack was constant. Fixes #9160 #9163 (alexey-milovidov)
- Fixed execution of inversed predicates when non-strictly monotonic functional index is used. Fixes #9034 #9223 (Akazz)
- Allow to rewrite `CROSS` to `INNER JOIN` if there's [NOT] `LIKE` operator in `WHERE` section. Fixes #9191 #9229 (4ertus2)

- Allow first column(s) in a table with Log engine be an alias.  
[#9231 \(abyss7\)](#)
- Allow comma join with `IN()` inside. Fixes [#7314](#).  
[#9251 \(4ertus2\)](#)
- Improve `ALTER MODIFY/ADD` queries logic. Now you cannot `ADD` column without type, `MODIFY` default expression does not change type of column and `MODIFY` type does not loose default expression value. Fixes [#8669](#).  
[#9227 \(alesapin\)](#)
- Fix mutations finalization, when already done mutation can have status `is_done=0`.  
[#9217 \(alesapin\)](#)
- Support "Processors" pipeline for `system.numbers` and `system.numbers_mt`. This also fixes the bug when `max_execution_time` is not respected.  
[#7796 \(KochetovNicolai\)](#)
- Fix wrong counting of `DictCacheKeysRequestedFound` metric.  
[#9411 \(nikitamikhaylov\)](#)
- Added a check for storage policy in `ATTACH PARTITION FROM`, `REPLACE PARTITION`, `MOVE TO TABLE` which otherwise could make data of part inaccessible after restart and prevent ClickHouse to start.  
[#9383 \(excitoon\)](#)
- Fixed UBSan report in `MergeTreeIndexSet`. This fixes [#9250](#)  
[#9365 \(alexey-milovidov\)](#)
- Fix possible datarace in `BlockIO`.  
[#9356 \(KochetovNicolai\)](#)
- Support for `UInt64` numbers that don't fit in `Int64` in JSON-related functions. Update `SIMDJSON` to master. This fixes [#9209](#)  
[#9344 \(alexey-milovidov\)](#)
- Fix the issue when the amount of free space is not calculated correctly if the data directory is mounted to a separate device. For default disk calculate the free space from data subdirectory. This fixes [#7441](#) [#9257 \(millb\)](#)
- Fix the issue when TLS connections may fail with the message `OpenSSL SSL_read: error:14094438:SSL routines:ssl3_read_bytes:tlsv1 alert internal error` and `SSL Exception: error:2400006E:random number generator::error retrieving entropy`. Update OpenSSL to upstream master.  
[#8956 \(alexey-milovidov\)](#)
- When executing `CREATE` query, fold constant expressions in storage engine arguments. Replace empty database name with current database. Fixes [#6508](#), [#3492](#). Also fix check for local address in `ClickHouseDictionarySource`.  
[#9262 \(tabplubix\)](#)
- Fix segfault in `StorageMerge`, which can happen when reading from `StorageFile`.  
[#9387 \(tabplubix\)](#)
- Prevent losing data in `Kafka` in rare cases when exception happens after reading suffix but before commit. Fixes [#9378](#). Related: [#7175](#)  
[#9507 \(filimonov\)](#)
- Fix bug leading to server termination when trying to use / drop `Kafka` table created with wrong parameters. Fixes [#9494](#). Incorporates [#9507](#).  
[#9513 \(filimonov\)](#)

## New Feature

- Add `deduplicate_blocks_in_dependent_materialized_views` option to control the behaviour of idempotent inserts into tables with materialized views. This new feature was added to the bugfix release by a special request from Altinity.  
[#9070 \(urykhy\)](#)

## ClickHouse release v20.1.2.4, 2020-01-22

### Backward Incompatible Change

- Make the setting `merge_tree_uniform_read_distribution` obsolete. The server still recognizes this setting but it has no effect. [#8308 \(alexey-milovidov\)](#)
- Changed return type of the function `greatCircleDistance` to `Float32` because now the result of calculation is `Float32`. [#7993 \(alexey-milovidov\)](#)
- Now it's expected that query parameters are represented in "escaped" format. For example, to pass string `a<tab>b` you have to write `a\tb` or `a\<tab>b` and respectively, `a%5Ctb` or `a%5C%09b` in URL. This is needed to add the possibility to pass NULL as `\N`. This fixes [#7488](#). [#8517 \(alexey-milovidov\)](#)
- Enable `use_minimalistic_part_header_in_zookeeper` setting for `ReplicatedMergeTree` by default. This will significantly reduce amount of data stored in ZooKeeper. This setting is supported since version 19.1 and we already use it in production in multiple services without any issues for more than half a year. Disable this setting if you have a chance to downgrade to versions older than 19.1. [#6850 \(alexey-milovidov\)](#)
- Data skipping indices are production ready and enabled by default. The settings `allow_experimental_data_skipping_indices`, `allow_experimental_cross_to_join_conversion` and `allow_experimental_multiple_joins_emulation` are now obsolete and do nothing. [#7974 \(alexey-milovidov\)](#)
- Add new ANY JOIN logic for StorageJoin consistent with JOIN operation. To upgrade without changes in behaviour you need add SETTINGS `any_join_distinct_right_table_keys = 1` to Engine Join tables metadata or recreate these tables after upgrade. [#8400 \(Artem Zuikov\)](#)
- Require server to be restarted to apply the changes in logging configuration. This is a temporary workaround to avoid the bug where the server logs to a deleted log file (see [#8696](#)). [#8707 \(Alexander Kuzmenkov\)](#)

## New Feature

- Added information about part paths to `system.merges`. [#8043 \(Vladimir Chebotarev\)](#)
- Add ability to execute `SYSTEM RELOAD DICTIONARY` query in ON CLUSTER mode. [#8288 \(Guillaume Tassery\)](#)
- Add ability to execute `CREATE DICTIONARY` queries in ON CLUSTER mode. [#8163 \(alesapin\)](#)
- Now user's profile in `users.xml` can inherit multiple profiles. [#8343 \(Mikhail f. Shiryaev\)](#)
- Added `system.stack_trace` table that allows to look at stack traces of all server threads. This is useful for developers to introspect server state. This fixes [#7576](#). [#8344 \(alexey-milovidov\)](#)
- Add `DateTime64` datatype with configurable sub-second precision. [#7170 \(Vasily Nemkov\)](#)
- Add table function `clusterAllReplicas` which allows to query all the nodes in the cluster. [#8493 \(kiran sunkari\)](#)
- Add aggregate function `categoricalInformationValue` which calculates the information value of a discrete feature. [#8117 \(hczi\)](#)

- Speed up parsing of data files in CSV, TSV and JSONEachRow format by doing it in parallel. #7780 (Alexander Kuzmenkov)
- Add function `bankerRound` which performs banker's rounding. #8112 (hcz)
- Support more languages in embedded dictionary for region names: 'ru', 'en', 'ua', 'uk', 'by', 'kz', 'tr', 'de', 'uz', 'lv', 'lt', 'et', 'pt', 'he', 'vi'. #8189 (alexey-milovidov)
- Improvements in consistency of ANY JOIN logic. Now `t1 ANY LEFT JOIN t2` equals `t2 ANY RIGHT JOIN t1`. #7665 (Artem Zuikov)
- Add setting `any_join_distinct_right_table_keys` which enables old behaviour for ANY INNER JOIN. #7665 (Artem Zuikov)
- Add new SEMI and ANTI JOIN. Old ANY INNER JOIN behaviour now available as SEMI LEFT JOIN. #7665 (Artem Zuikov)
- Added Distributed format for File engine and file table function which allows to read from .bin files generated by asynchronous inserts into Distributed table. #8535 (Nikolai Kochetov)
- Add optional reset column argument for `runningAccumulate` which allows to reset aggregation results for each new key value. #8326 (Sergey Kononenko)
- Add ability to use ClickHouse as Prometheus endpoint. #7900 (vdimir)
- Add section `<remote_url_allow_hosts>` in config.xml which restricts allowed hosts for remote table engines and table functions URL, S3, HDFS. #7154 (Mikhail Korotov)
- Added function `greatCircleAngle` which calculates the distance on a sphere in degrees. #8105 (alexey-milovidov)
- Changed Earth radius to be consistent with H3 library. #8105 (alexey-milovidov)
- Added `JSONCompactEachRow` and `JSONCompactEachRowWithNamesAndTypes` formats for input and output. #7841 (Mikhail Korotov)
- Added feature for file-related table engines and table functions (File, S3, URL, HDFS) which allows to read and write gzip files based on additional engine parameter or file extension. #7840 (Andrey Bodrov)
- Added the `randomASCII(length)` function, generating a string with a random set of ASCII printable characters. #8401 (BayoNet)
- Added function `JSONExtractArrayRaw` which returns an array on unparsed json array elements from JSON string. #8081 (Oleg Matrokhin)
- Add `arrayZip` function which allows to combine multiple arrays of equal lengths into one array of tuples. #8149 (Winter Zhang)
- Add ability to move data between disks according to configured TTL-expressions for \*MergeTree table engines family. #8140 (Vladimir Chebotarev)
- Added new aggregate function `avgWeighted` which allows to calculate weighted average. #7898 (Andrey Bodrov)
- Now parallel parsing is enabled by default for TSV, TSKV, CSV and JSONEachRow formats. #7894 (Nikita Mikhaylov)
- Add several geo functions from H3 library: `h3GetResolution`, `h3EdgeAngle`, `h3EdgeLength`, `h3IsValid` and `h3kRing`. #8034 (Konstantin Malanchev)

- Added support for brotli (`br`) compression in file-related storages and table functions. This fixes [#8156](#). [#8526 \(alexey-milovidov\)](#)

- Add `groupBit*` functions for the `SimpleAggregationFunction` type. [#8485 \(Guillaume Tassery\)](#)

## Bug Fix

- Fix rename of tables with `Distributed` engine. Fixes issue [#7868](#). [#8306 \(tavplubix\)](#)
- Now dictionaries support `EXPRESSION` for attributes in arbitrary string in non-ClickHouse SQL dialect. [#8098 \(alesapin\)](#)
- Fix broken `INSERT SELECT FROM mysql(...)` query. This fixes [#8070](#) and [#7960](#). [#8234 \(tavplubix\)](#)
- Fix error "Mismatch column sizes" when inserting default `Tuple` from `JSONEachRow`. This fixes [#5653](#). [#8606 \(tavplubix\)](#)
- Now an exception will be thrown in case of using `WITH TIES` alongside `LIMIT BY`. Also add ability to use `TOP` with `LIMIT BY`. This fixes [#7472](#). [#7637 \(Nikita Mikhaylov\)](#)
- Fix unintended dependency from fresh glibc version in `clickhouse-odbc-bridge` binary. [#8046 \(Amos Bird\)](#)
- Fix bug in check function of `*MergeTree` engines family. Now it does not fail in case when we have equal amount of rows in last granule and last mark (non-final). [#8047 \(alesapin\)](#)
- Fix insert into `Enum*` columns after `ALTER` query, when underlying numeric type is equal to table specified type. This fixes [#7836](#). [#7908 \(Anton Popov\)](#)
- Allowed non-constant negative "size" argument for function `substring`. It was not allowed by mistake. This fixes [#4832](#). [#7703 \(alexey-milovidov\)](#)
- Fix parsing bug when wrong number of arguments passed to `(OJ)DBC` table engine. [#7709 \(alesapin\)](#)
- Using command name of the running clickhouse process when sending logs to syslog. In previous versions, empty string was used instead of command name. [#8460 \(Michael Nacharov\)](#)
- Fix check of allowed hosts for `localhost`. This PR fixes the solution provided in [#8241](#). [#8342 \(Vitaly Baranov\)](#)
- Fix rare crash in `argMin` and `argMax` functions for long string arguments, when result is used in `runningAccumulate` function. This fixes [#8325](#) [#8341 \(dinosaur\)](#)
- Fix memory overcommit for tables with `Buffer` engine. [#8345 \(Azat Khuzhin\)](#)
- Fixed potential bug in functions that can take `NULL` as one of the arguments and return non-`NULL`. [#8196 \(alexey-milovidov\)](#)
- Better metrics calculations in thread pool for background processes for `MergeTree` table engines. [#8194 \(Vladimir Chebotarev\)](#)
- Fix function `IN` inside `WHERE` statement when row-level table filter is present. Fixes [#6687](#) [#8357 \(Ivan\)](#)
- Now an exception is thrown if the integral value is not parsed completely for settings values. [#7678 \(Mikhail Korotov\)](#)
- Fix exception when aggregate function is used in query to distributed table with more than two local shards. [#8164 \(小路\)](#)
- Now bloom filter can handle zero length arrays and does not perform redundant calculations. [#8242 \(achimbab\)](#)

- Fixed checking if a client host is allowed by matching the client host to `host_regex` specified in `users.xml`. #8241 (Vitaly Baranov)
- Relax ambiguous column check that leads to false positives in multiple `JOIN ON` section. #8385 (Artem Zuikov)
- Fixed possible server crash (`std::terminate`) when the server cannot send or write data in `JSON` or `XML` format with values of `String` data type (that require `UTF-8` validation) or when compressing result data with Brotli algorithm or in some other rare cases. This fixes #7603 #8384 (alexey-milovidov)
- Fix race condition in `StorageDistributedDirectoryMonitor` found by CI. This fixes #8364. #8383 (Nikolai Kochetov)
- Now background merges in `*MergeTree` table engines family preserve storage policy volume order more accurately. #8549 (Vladimir Chebotarev)
- Now table engine `Kafka` works properly with `Native` format. This fixes #6731 #7337 #8003. #8016 (filimonov)
- Fixed formats with headers (like `CSVWithNames`) which were throwing exception about EOF for table engine `Kafka`. #8016 (filimonov)
- Fixed a bug with making set from subquery in right part of `IN` section. This fixes #5767 and #2542. #7755 (Nikita Mikhaylov)
- Fix possible crash while reading from storage File. #7756 (Nikolai Kochetov)
- Fixed reading of the files in `Parquet` format containing columns of type `list`. #8334 (maxulan)
- Fix error `Not found column` for distributed queries with `PREWHERE` condition dependent on sampling key if `max_parallel_replicas > 1`. #7913 (Nikolai Kochetov)
- Fix error `Not found column` if query used `PREWHERE` dependent on table's alias and the result set was empty because of primary key condition. #7911 (Nikolai Kochetov)
- Fixed return type for functions `rand` and `randConstant` in case of `Nullable` argument. Now functions always return `UInt32` and never `Nullable(UInt32)`. #8204 (Nikolai Kochetov)
- Disabled predicate push-down for `WITH FILL` expression. This fixes #7784. #7789 (Winter Zhang)
- Fixed incorrect `count()` result for `SummingMergeTree` when `FINAL` section is used. #3280 #7786 (Nikita Mikhaylov)
- Fix possible incorrect result for constant functions from remote servers. It happened for queries with functions like `version()`, `uptime()`, etc. which returns different constant values for different servers. This fixes #7666. #7689 (Nikolai Kochetov)
- Fix complicated bug in push-down predicate optimization which leads to wrong results. This fixes a lot of issues on push-down predicate optimization. #8503 (Winter Zhang)
- Fix crash in `CREATE TABLE .. AS` dictionary query. #8508 (Azat Khuzhin)
- Several improvements ClickHouse grammar in `.g4` file. #8294 (taiyang-li)
- Fix bug that leads to crashes in `JOINS` with tables with engine `Join`. This fixes #7556 #8254 #7915 #8100. #8298 (Artem Zuikov)
- Fix redundant dictionaries reload on `CREATE DATABASE`. #7916 (Azat Khuzhin)
- Limit maximum number of streams for read from `StorageFile` and `StorageHDFS`. Fixes #7650. #7981 (alesapin)

- Fix bug in `ALTER ... MODIFY ... CODEC` query, when user specify both default expression and codec. Fixes #8593. #8614 (alesapin)
- Fix error in background merge of columns with `SimpleAggregateFunction(LowCardinality)` type. #8613 (Nikolai Kochetov)
- Fixed type check in function `toDateTime64`. #8375 (Vasily Nemkov)
- Now server do not crash on `LEFT` or `FULL JOIN` with and Join engine and unsupported `join_use_nulls` settings. #8479 (Artem Zuikov)
- Now `DROP DICTIONARY IF EXISTS db.dict` query does not throw exception if `db` does not exist. #8185 (Vitaly Baranov)
- Fix possible crashes in table functions (`file`, `mysql`, `remote`) caused by usage of reference to removed `IStorage` object. Fix incorrect parsing of columns specified at insertion into table function. #7762 (tavplubix)
- Ensure network be up before starting `clickhouse-server`. This fixes #7507. #8570 (Zhichang Yu)
- Fix timeouts handling for secure connections, so queries does not hang indefinitely. This fixes #8126. #8128 (alexey-milovidov)
- Fix `clickhouse-copier`'s redundant contention between concurrent workers. #7816 (Ding Xiang Fei)
- Now mutations does not skip attached parts, even if their mutation version were larger than current mutation version. #7812 (Zhichang Yu) #8250 (alesapin)
- Ignore redundant copies of `*MergeTree` data parts after move to another disk and server restart. #7810 (Vladimir Chebotarev)
- Fix crash in `FULL JOIN` with `LowCardinality` in `JOIN` key. #8252 (Artem Zuikov)
- Forbidden to use column name more than once in insert query like `INSERT INTO tbl (x, y, x)`. This fixes #5465, #7681. #7685 (alesapin)
- Added fallback for detection the number of physical CPU cores for unknown CPUs (using the number of logical CPU cores). This fixes #5239. #7726 (alexey-milovidov)
- Fix `There's no column` error for materialized and alias columns. #8210 (Artem Zuikov)
- Fixed sever crash when `EXISTS` query was used without `TABLE` or `DICTIONARY` qualifier. Just like `EXISTS t`. This fixes #8172. This bug was introduced in version 19.17. #8213 (alexey-milovidov)
- Fix rare bug with error "Sizes of columns does not match" that might appear when using `SimpleAggregateFunction` column. #7790 (Boris Granveaud)
- Fix bug where user with empty `allow_databases` got access to all databases (and same for `allow_dictionaries`). #7793 (DeifyTheGod)
- Fix client crash when server already disconnected from client. #8071 (Azat Khuzhin)
- Fix `ORDER BY` behaviour in case of sorting by primary key prefix and non primary key suffix. #7759 (Anton Popov)
- Check if qualified column present in the table. This fixes #6836. #7758 (Artem Zuikov)
- Fixed behavior with `ALTER MOVE` ran immediately after merge finish moves superpart of specified. Fixes #8103. #8104 (Vladimir Chebotarev)

- Fix possible server crash while using `UNION` with different number of columns. Fixes #7279. #7929 ([Nikolai Kochetov](#))
- Fix size of result substring for function `substr` with negative size. #8589 ([Nikolai Kochetov](#))
- Now server does not execute part mutation in `MergeTree` if there are not enough free threads in background pool. #8588 ([tavplubix](#))
- Fix a minor typo on formatting `UNION ALL AST`. #7999 ([lita091](#))
- Fixed incorrect bloom filter results for negative numbers. This fixes #8317. #8566 ([Winter Zhang](#))
- Fixed potential buffer overflow in decompress. Malicious user can pass fabricated compressed data that will cause read after buffer. This issue was found by Eldar Zaitov from Yandex information security team. #8404 ([alexey-milovidov](#))
- Fix incorrect result because of integers overflow in `arrayIntersect`. #7777 ([Nikolai Kochetov](#))
- Now `OPTIMIZE TABLE` query will not wait for offline replicas to perform the operation. #8314 ([javi santana](#))
- Fixed `ALTER TTL` parser for `Replicated*MergeTree` tables. #8318 ([Vladimir Chebotarev](#))
- Fix communication between server and client, so server read temporary tables info after query failure. #8084 ([Azat Khuzhin](#))
- Fix `bitmapAnd` function error when intersecting an aggregated bitmap and a scalar bitmap. #8082 ([Yue Huang](#))
- Refine the definition of `ZXid` according to the ZooKeeper Programmer's Guide which fixes bug in `clickhouse-cluster-copier`. #8088 ([Ding Xiang Fei](#))
- `odbc` table function now respects `external_table_functions_use_nulls` setting. #7506 ([Vasily Nemkov](#))
- Fixed bug that lead to a rare data race. #8143 ([Alexander Kazakov](#))
- Now `SYSTEM RELOAD DICTIONARY` reloads a dictionary completely, ignoring `update_field`. This fixes #7440. #8037 ([Vitaly Baranov](#))
- Add ability to check if dictionary exists in create query. #8032 ([alesapin](#))
- Fix `Float*` parsing in `Values` format. This fixes #7817. #7870 ([tavplubix](#))
- Fix crash when we cannot reserve space in some background operations of `*MergeTree` table engines family. #7873 ([Vladimir Chebotarev](#))
- Fix crash of merge operation when table contains `SimpleAggregateFunction(LowCardinality)` column. This fixes #8515. #8522 ([Azat Khuzhin](#))
- Restore support of all ICU locales and add the ability to apply collations for constant expressions. Also add language name to `system.collations` table. #8051 ([alesapin](#))
- Fix bug when external dictionaries with zero minimal lifetime (`LIFETIME(MIN 0 MAX N)`, `LIFETIME(N)`) don't update in background. #7983 ([alesapin](#))
- Fix crash when external dictionary with ClickHouse source has subquery in query. #8351 ([Nikolai Kochetov](#))
- Fix incorrect parsing of file extension in table with engine `URL`. This fixes #8157. #8419 ([Andrey Bodrov](#))
- Fix `CHECK TABLE` query for `*MergeTree` tables without key. Fixes #7543. #7979 ([alesapin](#))
- Fixed conversion of `Float64` to MySQL type. #8079 ([Yuriy Baranov](#))

- Now if table was not completely dropped because of server crash, server will try to restore and load it. #8176 (tavplubix)
- Fixed crash in table function `file` while inserting into file that does not exist. Now in this case file would be created and then insert would be processed. #8177 (Olga Khvostikova)
- Fix rare deadlock which can happen when `trace_log` is in enabled. #7838 (filimonov)
- Add ability to work with different types besides `Date` in `RangeHashed` external dictionary created from DDL query. Fixes 7899. #8275 (alesapin)
- Fixes crash when `now64()` is called with result of another function. #8270 (Vasily Nemkov)
- Fixed bug with detecting client IP for connections through mysql wire protocol. #7743 (Dmitry Muzyka)
- Fix empty array handling in `arraySplit` function. This fixes #7708. #7747 (hcza)
- Fixed the issue when `pid`-file of another running `clickhouse-server` may be deleted. #8487 (Weiqing Xu)
- Fix dictionary reload if it has `invalidate_query`, which stopped updates and some exception on previous update tries. #8029 (alesapin)
- Fixed error in function `arrayReduce` that may lead to "double free" and error in aggregate function combinator `Resample` that may lead to memory leak. Added aggregate function `aggThrow`. This function can be used for testing purposes. #8446 (alexey-milovidov)

## Improvement

- Improved logging when working with S3 table engine. #8251 (Grigory Pervakov)
- Printed help message when no arguments are passed when calling `clickhouse-local`. This fixes #5335. #8230 (Andrey Nagorny)
- Add setting `mutations_sync` which allows to wait `ALTER UPDATE/DELETE` queries synchronously. #8237 (alesapin)
- Allow to set up relative `user_files_path` in `config.xml` (in the way similar to `format_schema_path`). #7632 (hcza)
- Add exception for illegal types for conversion functions with `-OrZero` postfix. #7880 (Andrey Konyaev)
- Simplify format of the header of data sending to a shard in a distributed query. #8044 (Vitaly Baranov)
- Live View table engine refactoring. #8519 (vzakaznikov)
- Add additional checks for external dictionaries created from DDL-queries. #8127 (alesapin)
- Fix error `Column ... already exists` while using `FINAL` and `SAMPLE` together, e.g. `select count() from table final sample 1/2`. Fixes #5186. #7907 (Nikolai Kochetov)
- Now table the first argument of `joinGet` function can be table identifier. #7707 (Amos Bird)
- Allow using `MaterializedView` with subqueries above Kafka tables. #8197 (filimonov)
- Now background moves between disks run it the seprate thread pool. #7670 (Vladimir Chebotarev)
- SYSTEM RELOAD DICTIONARY now executes synchronously. #8240 (Vitaly Baranov)
- Stack traces now display physical addresses (offsets in object file) instead of virtual memory addresses (where the object file was loaded). That allows the use of `addr2line` when binary is position independent and ASLR is active. This fixes #8360. #8387 (alexey-milovidov)
- Support new syntax for row-level security filters: `<table name='table_name'>...</table>`. Fixes #5779. #8381 (Ivan)

- Now `cityHash` function can work with `Decimal` and `UUID` types. Fixes #5184. #7693 (Mikhail Korotov)
- Removed fixed index granularity (it was 1024) from system logs because it's obsolete after implementation of adaptive granularity. #7698 (alexey-milovidov)
- Enabled MySQL compatibility server when ClickHouse is compiled without SSL. #7852 (Yuriy Baranov)
- Now server checksums distributed batches, which gives more verbose errors in case of corrupted data in batch. #7914 (Azat Khuzhin)
- Support `DROP DATABASE`, `DETACH TABLE`, `DROP TABLE` and `ATTACH TABLE` for MySQL database engine. #8202 (Winter Zhang)
- Add authentication in S3 table function and table engine. #7623 (Vladimir Chebotarev)
- Added check for extra parts of `MergeTree` at different disks, in order to not allow to miss data parts at undefined disks. #8118 (Vladimir Chebotarev)
- Enable SSL support for Mac client and server. #8297 (Ivan)
- Now ClickHouse can work as MySQL federated server (see <https://dev.mysql.com/doc/refman/5.7/en/federated-create-server.html>). #7717 (Maxim Fedotov)
- `clickhouse-client` now only enable bracketed-paste when multiquery is on and multiline is off. This fixes #7757. #7761 (Amos Bird)
- Support `Array(Decimal)` in `if` function. #7721 (Artem Zuikov)
- Support Decimals in `arrayDifference`, `arrayCumSum` and `arrayCumSumNegative` functions. #7724 (Artem Zuikov)
- Added lifetime column to `system.dictionaries` table. #6820 #7727 (kekekekule)
- Improved check for existing parts on different disks for \*`MergeTree` table engines. Addresses #7660. #8440 (Vladimir Chebotarev)
- Integration with AWS SDK for S3 interactions which allows to use all S3 features out of the box. #8011 (Pavel Kovalenko)
- Added support for subqueries in Live View tables. #7792 (vzakaznikov)
- Check for using `Date` or `DateTime` column from `TTL` expressions was removed. #7920 (Vladimir Chebotarev)
- Information about disk was added to `system.detached_parts` table. #7833 (Vladimir Chebotarev)
- Now settings `max_(table|partition)_size_to_drop` can be changed without a restart. #7779 (Grigory Pervakov)
- Slightly better usability of error messages. Ask user not to remove the lines below Stack trace:. #7897 (alexey-milovidov)
- Better reading messages from `Kafka` engine in various formats after #7935. #8035 (Ivan)
- Better compatibility with MySQL clients which don't support `sha2_password` auth plugin. #8036 (Yuriy Baranov)
- Support more column types in MySQL compatibility server. #7975 (Yuriy Baranov)
- Implement `ORDER BY` optimization for `Merge`, `Buffer` and `Materilized View` storages with underlying `MergeTree` tables. #8130 (Anton Popov)

- Now we always use POSIX implementation of `getrandom` to have better compatibility with old kernels (< 3.17). [#7940 \(Amos Bird\)](#)
- Better check for valid destination in a move TTL rule. [#8410 \(Vladimir Chebotarev\)](#)
- Better checks for broken insert batches for `Distributed` table engine. [#7933 \(Azat Khuzhin\)](#)
- Add column with array of parts name which mutations must process in future to `system.mutations` table. [#8179 \(alesapin\)](#)
- Parallel merge sort optimization for processors. [#8552 \(Nikolai Kochetov\)](#)
- The settings `mark_cache_min_lifetime` is now obsolete and does nothing. In previous versions, mark cache can grow in memory larger than `mark_cache_size` to accomodate data within `mark_cache_min_lifetime` seconds. That was leading to confusion and higher memory usage than expected, that is especially bad on memory constrained systems. If you will see performance degradation after installing this release, you should increase the `mark_cache_size`. [#8484 \(alexey-milovidov\)](#)
- Preparation to use `tid` everywhere. This is needed for [#7477](#). [#8276 \(alexey-milovidov\)](#)

## Performance Improvement

- Performance optimizations in processors pipeline. [#7988 \(Nikolai Kochetov\)](#)
- Non-blocking updates of expired keys in cache dictionaries (with permission to read old ones). [#8303 \(Nikita Mikhaylov\)](#)
- Compile ClickHouse without `-fno-omit-frame-pointer` globally to spare one more register. [#8097 \(Amos Bird\)](#)
- Speedup `greatCircleDistance` function and add performance tests for it. [#7307 \(Olga Khvostikova\)](#)
- Improved performance of function `roundDown`. [#8465 \(alexey-milovidov\)](#)
- Improved performance of `max`, `min`, `argMin`, `argMax` for `DateTime64` data type. [#8199 \(Vasily Nemkov\)](#)
- Improved performance of sorting without a limit or with big limit and external sorting. [#8545 \(alexey-milovidov\)](#)
- Improved performance of formatting floating point numbers up to 6 times. [#8542 \(alexey-milovidov\)](#)
- Improved performance of `modulo` function. [#7750 \(Amos Bird\)](#)
- Optimized `ORDER BY` and merging with single column key. [#8335 \(alexey-milovidov\)](#)
- Better implementation for `arrayReduce`, `-Array` and `-State` combinators. [#7710 \(Amos Bird\)](#)
- Now `PREWHERE` should be optimized to be at least as efficient as `WHERE`. [#7769 \(Amos Bird\)](#)
- Improve the way `round` and `roundBankers` handling negative numbers. [#8229 \(hcz\)](#)
- Improved decoding performance of `DoubleDelta` and `Gorilla` codecs by roughly 30-40%. This fixes [#7082](#). [#8019 \(Vasily Nemkov\)](#)
- Improved performance of `base64` related functions. [#8444 \(alexey-milovidov\)](#)
- Added a function `geoDistance`. It is similar to `greatCircleDistance` but uses approximation to WGS-84 ellipsoid model. The performance of both functions are near the same. [#8086 \(alexey-milovidov\)](#)
- Faster `min` and `max` aggregation functions for `Decimal` data type. [#8144 \(Artem Zuikov\)](#)
- Vectorize processing `arrayReduce`. [#7608 \(Amos Bird\)](#)

- if chains are now optimized as multilf. #8355 (kamalov-ruslan)
- Fix performance regression of Kafka table engine introduced in 19.15. This fixes #7261. #7935 (filimonov)
- Removed "pie" code generation that gcc from Debian packages occasionally brings by default. #8483 (alexey-milovidov)
- Parallel parsing data formats #6553 (Nikita Mikhaylov)
- Enable optimized parser of Values with expressions by default (input\_format\_values\_deduce\_templates\_of\_expressions=1). #8231 (tavplubix)

## Build/Testing/Packaging Improvement

- Build fixes for ARM and in minimal mode. #8304 (proller)
- Add coverage file flush for clickhouse-server when std::atexit is not called. Also slightly improved logging in stateless tests with coverage. #8267 (alesapin)
- Update LLVM library in contrib. Avoid using LLVM from OS packages. #8258 (alexey-milovidov)
- Make bundled curl build fully quiet. #8232 #8203 (Pavel Kovalenko)
- Fix some MemorySanitizer warnings. #8235 (Alexander Kuzmenkov)
- Use add\_warning and no\_warning macros in CMakeLists.txt. #8604 (Ivan)
- Add support of Minio S3 Compatible object (<https://min.io/>) for better integration tests. #7863 #7875 (Pavel Kovalenko)
- Imported libc headers to contrib. It allows to make builds more consistent across various systems (only for x86\_64-linux-gnu). #5773 (alexey-milovidov)
- Remove -fPIC from some libraries. #8464 (alexey-milovidov)
- Clean CMakeLists.txt for curl. See <https://github.com/ClickHouse/ClickHouse/pull/8011#issuecomment-569478910> #8459 (alexey-milovidov)
- Silent warnings in CapNProto library. #8220 (alexey-milovidov)
- Add performance tests for short string optimized hash tables. #7679 (Amos Bird)
- Now ClickHouse will build on AArch64 even if MADV\_FREE is not available. This fixes #8027. #8243 (Amos Bird)
- Update zlib-ng to fix memory sanitizer problems. #7182 #8206 (Alexander Kuzmenkov)
- Enable internal MySQL library on non-Linux system, because usage of OS packages is very fragile and usually does not work at all. This fixes #5765. #8426 (alexey-milovidov)
- Fixed build on some systems after enabling libc++. This supersedes #8374. #8380 (alexey-milovidov)
- Make Field methods more type-safe to find more errors. #7386 #8209 (Alexander Kuzmenkov)
- Added missing files to the libc-headers submodule. #8507 (alexey-milovidov)
- Fix wrong JSON quoting in performance test output. #8497 (Nikolai Kochetov)
- Now stack trace is displayed for std::exception and Poco::Exception. In previous versions it was available only for DB::Exception. This improves diagnostics. #8501 (alexey-milovidov)
- Porting clock\_gettime and clock\_nanosleep for fresh glibc versions. #8054 (Amos Bird)

- Enable `part_log` in example config for developers. #8609 (alexey-milovidov)
- Fix async nature of reload in `01036_no_superfluous_dict_reload_on_create_database*`. #8111 (Azat Khuzhin)
- Fixed codec performance tests. #8615 (Vasily Nemkov)
- Add install scripts for `.tgz` build and documentation for them. #8612 #8591 (alesapin)
- Removed old `ZSTD` test (it was created in year 2016 to reproduce the bug that pre 1.0 version of `ZSTD` has had). This fixes #8618. #8619 (alexey-milovidov)
- Fixed build on Mac OS Catalina. #8600 (meo)
- Increased number of rows in codec performance tests to make results noticeable. #8574 (Vasily Nemkov)
- In debug builds, treat `LOGICAL_ERROR` exceptions as assertion failures, so that they are easier to notice. #8475 (Alexander Kuzmenkov)
- Make formats-related performance test more deterministic. #8477 (alexey-milovidov)
- Update `Iz4` to fix a MemorySanitizer failure. #8181 (Alexander Kuzmenkov)
- Suppress a known MemorySanitizer false positive in exception handling. #8182 (Alexander Kuzmenkov)
- Update `gcc` and `g++` to version 9 in `build/docker/build.sh` #7766 (TLightSky)
- Add performance test case to test that `PREWHERE` is worse than `WHERE`. #7768 (Amos Bird)
- Progress towards fixing one flaky test. #8621 (alexey-milovidov)
- Avoid MemorySanitizer report for data from `libunwind`. #8539 (alexey-milovidov)
- Updated `libc++` to the latest version. #8324 (alexey-milovidov)
- Build ICU library from sources. This fixes #6460. #8219 (alexey-milovidov)
- Switched from `libressl` to `openssl`. ClickHouse should support TLS 1.3 and SNI after this change. This fixes #8171. #8218 (alexey-milovidov)
- Fixed UBSan report when using `chacha20_poly1305` from SSL (happens on connect to <https://yandex.ru/>). #8214 (alexey-milovidov)
- Fix mode of default password file for `.deb` linux distros. #8075 (proller)
- Improved expression for getting `clickhouse-server` PID in `clickhouse-test`. #8063 (Alexander Kazakov)
- Updated contrib/gtest to v1.10.0. #8587 (Alexander Burmak)
- Fixed ThreadSanitizer report in `base64` library. Also updated this library to the latest version, but it does not matter. This fixes #8397. #8403 (alexey-milovidov)
- Fix `00600_replace_running_query` for processors. #8272 (Nikolai Kochetov)
- Remove support for `tcmalloc` to make `CMakeLists.txt` simpler. #8310 (alexey-milovidov)
- Release `gcc` builds now use `libc++` instead of `libstdc++`. Recently `libc++` was used only with clang. This will improve consistency of build configurations and portability. #8311 (alexey-milovidov)
- Enable ICU library for build with MemorySanitizer. #8222 (alexey-milovidov)
- Suppress warnings from `CapNProto` library. #8224 (alexey-milovidov)

- Removed special cases of code for `tcmalloc`, because it's no longer supported. [#8225 \(alexey-milovidov\)](#)
- In CI coverage task, kill the server gracefully to allow it to save the coverage report. This fixes incomplete coverage reports we've been seeing lately. [#8142 \(alesapin\)](#)
- Performance tests for all codecs against `Float64` and `UInt64` values. [#8349 \(Vasily Nemkov\)](#)
- `termcap` is very much deprecated and lead to various problems (f.g. missing "up" cap and echoing ^ instead of multi line) . Favor `terminfo` or bundled `ncurses`. [#7737 \(Amos Bird\)](#)
- Fix `test_storage_s3` integration test. [#7734 \(Nikolai Kochetov\)](#)
- Support `StorageFile(<format>, null)` to insert block into given format file without actually write to disk. This is required for performance tests. [#8455 \(Amos Bird\)](#)
- Added argument `--print-time` to functional tests which prints execution time per test. [#8001 \(Nikolai Kochetov\)](#)
- Added asserts to `KeyCondition` while evaluating RPN. This will fix warning from gcc-9. [#8279 \(alexey-milovidov\)](#)
- Dump cmake options in CI builds. [#8273 \(Alexander Kuzmenkov\)](#)
- Don't generate debug info for some fat libraries. [#8271 \(alexey-milovidov\)](#)
- Make `log_to_console.xml` always log to stderr, regardless of is it interactive or not. [#8395 \(Alexander Kuzmenkov\)](#)
- Removed some unused features from `clickhouse-performance-test` tool. [#8555 \(alexey-milovidov\)](#)
- Now we will also search for `lld-X` with corresponding `clang-X` version. [#8092 \(alesapin\)](#)
- Parquet build improvement. [#8421 \(maxulan\)](#)
- More GCC warnings [#8221 \(kreuzerkrieg\)](#)
- Package for Arch Linux now allows to run ClickHouse server, and not only client. [#8534 \(Vladimir Chebotarev\)](#)
- Fix test with processors. Tiny performance fixes. [#7672 \(Nikolai Kochetov\)](#)
- Update contrib/protobuf. [#8256 \(Matwey V. Kornilov\)](#)
- In preparation of switching to c++20 as a new year celebration. "May the C++ force be with ClickHouse." [#8447 \(Amos Bird\)](#)

## Experimental Feature

- Added experimental setting `min_bytes_to_use_mmap_io`. It allows to read big files without copying data from kernel to userspace. The setting is disabled by default. Recommended threshold is about 64 MB, because mmap/munmap is slow. [#8520 \(alexey-milovidov\)](#)
- Reworked quotas as a part of access control system. Added new table `system.quotas`, new functions `currentQuota`, `currentQuotaKey`, new SQL syntax `CREATE QUOTA`, `ALTER QUOTA`, `DROP QUOTA`, `SHOW QUOTA`. [#7257 \(Vitaly Baranov\)](#)
- Allow skipping unknown settings with warnings instead of throwing exceptions. [#7653 \(Vitaly Baranov\)](#)
- Reworked row policies as a part of access control system. Added new table `system.row_policies`, new function `currentRowPolicies()`, new SQL syntax `CREATE POLICY`, `ALTER POLICY`, `DROP POLICY`, `SHOW CREATE POLICY`, `SHOW POLICIES`. [#7808 \(Vitaly Baranov\)](#)

## Security Fix

- Fixed the possibility of reading directories structure in tables with File table engine. This fixes #8536. #8537 (alexey-milovidov)

## Changelog for 2019

---

### ClickHouse Release 19.17

#### ClickHouse Release 19.17.6.36, 2019-12-27

##### Bug Fix

- Fixed potential buffer overflow in decompress. Malicious user can pass fabricated compressed data that could cause read after buffer. This issue was found by Eldar Zaitov from Yandex information security team. #8404 (alexey-milovidov)
- Fixed possible server crash (`std::terminate`) when the server cannot send or write data in JSON or XML format with values of String data type (that require UTF-8 validation) or when compressing result data with Brotli algorithm or in some other rare cases. #8384 (alexey-milovidov)
- Fixed dictionaries with source from a clickhouse `VIEW`, now reading such dictionaries does not cause the error `There is no query.` #8351 (Nikolai Kochetov)
- Fixed checking if a client host is allowed by `host_regex` specified in `users.xml`. #8241, #8342 (Vitaly Baranov)
- `RENAME TABLE` for a distributed table now renames the folder containing inserted data before sending to shards. This fixes an issue with successive renames `tableA->tableB`, `tableC->tableA`. #8306 (tavplubix)
- `range_hashed` external dictionaries created by DDL queries now allow ranges of arbitrary numeric types. #8275 (alesapin)
- Fixed `INSERT INTO table SELECT ... FROM mysql(...)` table function. #8234 (tavplubix)
- Fixed segfault in `INSERT INTO TABLE FUNCTION file()` while inserting into a file which does not exist. Now in this case file would be created and then insert would be processed. #8177 (Olga Khvostikova)
- Fixed bitmapAnd error when intersecting an aggregated bitmap and a scalar bitmap. #8082 (Yue Huang)
- Fixed segfault when `EXISTS` query was used without `TABLE` or `DICTIONARY` qualifier, just like `EXISTS t.` #8213 (alexey-milovidov)
- Fixed return type for functions `rand` and `randConstant` in case of nullable argument. Now functions always return `UInt32` and never `Nullable(UInt32)`. #8204 (Nikolai Kochetov)
- Fixed `DROP DICTIONARY IF EXISTS db.dict`, now it does not throw exception if `db` does not exist. #8185 (Vitaly Baranov)
- If a table wasn't completely dropped because of server crash, the server will try to restore and load it #8176 (tavplubix)
- Fixed a trivial count query for a distributed table if there are more than two shard local table. #8164 (小路)
- Fixed bug that lead to a data race in `DB::BlockStreamProfileInfo::calculateRowsBeforeLimit()` #8143 (Alexander Kazakov)

- Fixed ALTER table MOVE part executed immediately after merging the specified part, which could cause moving a part which the specified part merged into. Now it correctly moves the specified part. #8104 (Vladimir Chebotarev)
- Expressions for dictionaries can be specified as strings now. This is useful for calculation of attributes while extracting data from non-ClickHouse sources because it allows to use non-ClickHouse syntax for those expressions. #8098 (alesapin)
- Fixed a very rare race in clickhouse-copier because of an overflow in ZXid. #8088 (Ding Xiang Fei)
- Fixed the bug when after the query failed (due to “Too many simultaneous queries” for example) it would not read external tables info, and the next request would interpret this info as the beginning of the next query causing an error like Unknown packet from client. #8084 (Azat Khuzhin)
- Avoid null dereference after “Unknown packet X from server” #8071 (Azat Khuzhin)
- Restore support of all ICU locales, add the ability to apply collations for constant expressions and add language name to system.collations table. #8051 (alesapin)
- Number of streams for read from StorageFile and StorageHDFS is now limited, to avoid exceeding the memory limit. #7981 (alesapin)
- Fixed CHECK TABLE query for \*MergeTree tables without key. #7979 (alesapin)
- Removed the mutation number from a part name in case there were no mutations. This removing improved the compatibility with older versions. #8250 (alesapin)
- Fixed the bug that mutations are skipped for some attached parts due to their data\_version are larger than the table mutation version. #7812 (Zhichang Yu)
- Allow starting the server with redundant copies of parts after moving them to another device. #7810 (Vladimir Chebotarev)
- Fixed the error “Sizes of columns does not match” that might appear when using aggregate function columns. #7790 (Boris Granveaud)
- Now an exception will be thrown in case of using WITH TIES alongside LIMIT BY. And now it’s possible to use TOP with LIMIT BY. #7637 (Nikita Mikhaylov)
- Fix dictionary reload if it has invalidate\_query, which stopped updates and some exception on previous update tries. #8029 (alesapin)

## ClickHouse Release 19.17.4.11, 2019-11-22

### Backward Incompatible Change

- Using column instead of AST to store scalar subquery results for better performance. Setting enable\_scalar\_subquery\_optimization was added in 19.17 and it was enabled by default. It leads to errors like this during upgrade to 19.17.2 or 19.17.3 from previous versions. This setting was disabled by default in 19.17.4, to make possible upgrading from 19.16 and older versions without errors. #7392 (Amos Bird)

### New Feature

- Add the ability to create dictionaries with DDL queries. #7360 (alesapin)
- Make bloom\_filter type of index supporting LowCardinality and Nullable #7363 #7561 (Nikolai Kochetov)
- Add function isValidJSON to check that passed string is a valid json. #5910 #7293 (Vdimir)
- Implement arrayCompact function #7328 (Memo)

- Created function `hex` for Decimal numbers. It works like `hex(reinterpretAsString())`, but does not delete last zero bytes. [#7355 \(Mikhail Korotov\)](#)
- Add `arrayFill` and `arrayReverseFill` functions, which replace elements by other elements in front/back of them in the array. [#7380 \(hcz\)](#)
- Add `CRC32IEEE()/CRC64()` support [#7480 \(Azat Khuzhin\)](#)
- Implement `char` function similar to one in mysql [#7486 \(sundyli\)](#)
- Add `bitmapTransform` function. It transforms an array of values in a bitmap to another array of values, the result is a new bitmap [#7598 \(Zhichang Yu\)](#)
- Implemented `javaHashUTF16LE()` function [#7651 \(achimbab\)](#)
- Add `_shard_num` virtual column for the Distributed engine [#7624 \(Azat Khuzhin\)](#)

## Experimental Feature

- Support for processors (new query execution pipeline) in `MergeTree`. [#7181 \(Nikolai Kochetov\)](#)

## Bug Fix

- Fix incorrect float parsing in `Values` [#7817 #7870 \(tavplubix\)](#)
- Fix rare deadlock which can happen when `trace_log` is enabled. [#7838 \(filimonov\)](#)
- Prevent message duplication when producing Kafka table has any MVs selecting from it [#7265 \(Ivan\)](#)
- Support for `Array(LowCardinality(Nullable(String)))` in `IN`. Resolves [#7364 #7366 \(achimbab\)](#)
- Add handling of `SQL_TINYINT` and `SQL_BIGINT`, and fix handling of `SQL_FLOAT` data source types in ODBC Bridge. [#7491 \(Denis Glazachev\)](#)
- Fix aggregation (`avg` and `quantiles`) over empty decimal columns [#7431 \(Andrey Konyaev\)](#)
- Fix `INSERT` into `Distributed` with `MATERIALIZED` columns [#7377 \(Azat Khuzhin\)](#)
- Make `MOVE PARTITION` work if some parts of partition are already on destination disk or volume [#7434 \(Vladimir Chebotarev\)](#)
- Fixed bug with hardlinks failing to be created during mutations in `ReplicatedMergeTree` in multi-disk configurations. [#7558 \(Vladimir Chebotarev\)](#)
- Fixed a bug with a mutation on a `MergeTree` when whole part remains unchanged and best space is being found on another disk [#7602 \(Vladimir Chebotarev\)](#)
- Fixed bug with `keep_free_space_ratio` not being read from disks configuration [#7645 \(Vladimir Chebotarev\)](#)
- Fix bug with table contains only `Tuple` columns or columns with complex paths. Fixes [7541](#). [#7545 \(alesapin\)](#)
- Do not account memory for `Buffer` engine in `max_memory_usage` limit [#7552 \(Azat Khuzhin\)](#)
- Fix final mark usage in `MergeTree` tables ordered by `tuple()`. In rare cases it could lead to `Can't adjust last granule` error while select. [#7639 \(Anton Popov\)](#)
- Fix bug in mutations that have predicate with actions that require context (for example functions for `json`), which may lead to crashes or strange exceptions. [#7664 \(alesapin\)](#)
- Fix mismatch of database and table names escaping in `data/` and `shadow/` directories [#7575 \(Alexander Burmak\)](#)

- Support duplicated keys in RIGHT|FULL JOINS, e.g. ON t.x = u.x AND t.x = u.y. Fix crash in this case. #7586 (Artem Zuikov)
- Fix Not found column <expression> in block when joining on expression with RIGHT or FULL JOIN. #7641 (Artem Zuikov)
- One more attempt to fix infinite loop in PrettySpace format #7591 (Olga Khvostikova)
- Fix bug in concat function when all arguments were FixedString of the same size. #7635 (alesapin)
- Fixed exception in case of using 1 argument while defining S3, URL and HDFS storages. #7618 (Vladimir Chebotarev)
- Fix scope of the InterpreterSelectQuery for views with query #7601 (Azat Khuzhin)

## Improvement

- Nullable columns recognized and NULL-values handled correctly by ODBC-bridge #7402 (Vasily Nemkov)
- Write current batch for distributed send atomically #7600 (Azat Khuzhin)
- Throw an exception if we cannot detect table for column name in query. #7358 (Artem Zuikov)
- Add merge\_max\_block\_size setting to MergeTreeSettings #7412 (Artem Zuikov)
- Queries with HAVING and without GROUP BY assume group by constant. So, SELECT 1 HAVING 1 now returns a result. #7496 (Amos Bird)
- Support parsing (X,) as tuple similar to python. #7501, #7562 (Amos Bird)
- Make range function behaviors almost like pythonic one. #7518 (sundyli)
- Add constraints columns to table system.settings #7553 (Vitaly Baranov)
- Better Null format for tcp handler, so that it's possible to use select ignore(<expression>) from table format Null for perf measure via clickhouse-client #7606 (Amos Bird)
- Queries like CREATE TABLE ... AS (SELECT (1, 2)) are parsed correctly #7542 (hcz)

## Performance Improvement

- The performance of aggregation over short string keys is improved. #6243 (Alexander Kuzmenkov, Amos Bird)
- Run another pass of syntax/expression analysis to get potential optimizations after constant predicates are folded. #7497 (Amos Bird)
- Use storage meta info to evaluate trivial SELECT count() FROM table; #7510 (Amos Bird, alexey-milovidov)
- Vectorize processing arrayReduce similar to Aggregator addBatch. #7608 (Amos Bird)
- Minor improvements in performance of Kafka consumption #7475 (Ivan)

## Build/Testing/Packaging Improvement

- Add support for cross-compiling to the CPU architecture AARCH64. Refactor packager script. #7370 #7539 (Ivan)
- Unpack darwin-x86\_64 and linux-aarch64 toolchains into mounted Docker volume when building packages #7534 (Ivan)
- Update Docker Image for Binary Packager #7474 (Ivan)
- Fixed compile errors on MacOS Catalina #7585 (Ernest Poletaev)

- Some refactoring in query analysis logic: split complex class into several simple ones. [#7454](#) ([Artem Zuikov](#))
- Fix build without submodules [#7295](#) ([proller](#))
- Better `add_globs` in CMake files [#7418](#) ([Amos Bird](#))
- Remove hardcoded paths in `unwind` target [#7460](#) ([Konstantin Podshumok](#))
- Allow to use mysql format without ssl [#7524](#) ([proller](#))

## Other

- Added ANTLR4 grammar for ClickHouse SQL dialect [#7595](#) [#7596](#) ([alexey-milovidov](#))

# ClickHouse Release 19.16

ClickHouse Release 19.16.14.65, 2020-03-25

- Fixed up a bug in batched calculations of ternary logical OPs on multiple arguments (more than 10). [#8718](#) ([Alexander Kazakov](#)) This bugfix was backported to version 19.16 by a special request from Altinity.

ClickHouse Release 19.16.14.65, 2020-03-05

- Fix distributed subqueries incompatibility with older CH versions. Fixes [#7851](#) ([tabplubix](#))
- When executing `CREATE` query, fold constant expressions in storage engine arguments. Replace empty database name with current database. Fixes [#6508](#), [#3492](#). Also fix check for local address in `ClickHouseDictionarySource`.  
[#9262](#) ([tabplubix](#))
- Now background merges in `*MergeTree` table engines family preserve storage policy volume order more accurately.  
[#8549](#) ([Vladimir Chebotarev](#))
- Prevent losing data in `Kafka` in rare cases when exception happens after reading suffix but before commit. Fixes [#9378](#). Related: [#7175](#)  
[#9507](#) ([filimonov](#))
- Fix bug leading to server termination when trying to use / drop `Kafka` table created with wrong parameters. Fixes [#9494](#). Incorporates [#9507](#).  
[#9513](#) ([filimonov](#))
- Allow using `MaterializedView` with subqueries above `Kafka` tables.  
[#8197](#) ([filimonov](#))

## New Feature

- Add `deduplicate_blocks_in_dependent_materialized_views` option to control the behaviour of idempotent inserts into tables with materialized views. This new feature was added to the bugfix release by a special request from Altinity.  
[#9070](#) ([urykhy](#))

# ClickHouse Release 19.16.2.2, 2019-10-30

Backward Incompatible Change

- Add missing arity validation for count/countIf.  
[#7095](#)  
[#7298 \(Vdimir\)](#)
- Remove legacy `asterisk_left_columns_only` setting (it was disabled by default).  
[#7335 \(Artem Zuikov\)](#)
- Format strings for Template data format are now specified in files.  
[#7118 \(tavplubix\)](#)

## New Feature

- Introduce `uniqCombined64()` to calculate cardinality greater than `UINT_MAX`.  
[#7213](#),  
[#7222 \(Azat Khuzhin\)](#)
- Support Bloom filter indexes on Array columns.  
[#6984 \(achimbab\)](#)
- Add a function `getMacro(name)` that returns String with the value of corresponding `<macros>` from server configuration. [#7240 \(alexey-milovidov\)](#)
- Set two configuration options for a dictionary based on an HTTP source: `credentials` and `http-headers`. [#7092 \(Guillaume Tassery\)](#)
- Add a new `ProfileEvent Merge` that counts the number of launched background merges.  
[#7093 \(Mikhail Korotov\)](#)
- Add fullHostName function that returns a fully qualified domain name.  
[#7263](#)  
[#7291 \(sundyli\)](#)
- Add function `arraySplit` and `arrayReverseSplit` which split an array by “cut off” conditions. They are useful in time sequence handling.  
[#7294 \(hczi\)](#)
- Add new functions that return the Array of all matched indices in multiMatch family of functions.  
[#7299 \(Danila Kutenin\)](#)
- Add a new database engine `Lazy` that is optimized for storing a large number of small -Log tables. [#7171 \(Nikita Vasilev\)](#)
- Add aggregate functions `groupBitmapAnd`, `-Or`, `-Xor` for bitmap columns. [#7109 \(Zhichang Yu\)](#)
- Add aggregate function combinators `-OrNull` and `-OrDefault`, which return null or default values when there is nothing to aggregate.  
[#7331 \(hczi\)](#)

- Introduce CustomSeparated data format that supports custom escaping and delimiter rules. #7118 ([tavplubix](#))
- Support Redis as source of external dictionary. #4361 #6962 ([comunodi](#), [Anton Popov](#))

## Bug Fix

- Fix wrong query result if it has WHERE IN (SELECT ...) section and `optimize_read_in_order` is used. #7371 ([Anton Popov](#))
- Disabled MariaDB authentication plugin, which depends on files outside of project. #7140 ([Yuriy Baranov](#))
- Fix exception `Cannot convert column ... because it is constant but values of constants are different in source and result` which could rarely happen when functions `now()`, `today()`, `yesterday()`, `randConstant()` are used. #7156 ([Nikolai Kochetov](#))
- Fixed issue of using HTTP keep alive timeout instead of TCP keep alive timeout. #7351 ([Vasily Nemkov](#))
- Fixed a segmentation fault in `groupBitmapOr` (issue #7109). #7289 ([Zhichang Yu](#))
- For materialized views the commit for Kafka is called after all data were written. #7175 ([Ivan](#))
- Fixed wrong `duration_ms` value in `system.part_log` table. It was ten times off. #7172 ([Vladimir Chebotarev](#))
- A quick fix to resolve crash in LIVE VIEW table and re-enabling all LIVE VIEW tests. #7201 ([vzakaznikov](#))
- Serialize NULL values correctly in min/max indexes of MergeTree parts. #7234 ([Alexander Kuzmenkov](#))
- Don't put virtual columns to .sql metadata when table is created as CREATE TABLE AS. #7183 ([Ivan](#))
- Fix segmentation fault in ATTACH PART query. #7185 ([alesapin](#))
- Fix wrong result for some queries given by the optimization of empty IN subqueries and empty INNER/RIGHT JOIN. #7284 ([Nikolai Kochetov](#))

- Fixing AddressSanitizer error in the LIVE VIEW getHeader() method.

#7271

(vzakaznikov)

## Improvement

- Add a message in case of queue\_wait\_max\_ms wait takes place.

#7390 (Azat

Khuzhin)

- Made setting s3\_min\_upload\_part\_size table-level.

#7059 (Vladimir

Chebotarev)

- Check TTL in StorageFactory. #7304

(sundyli)

- Squash left-hand blocks in partial merge join (optimization).

#7122 (Artem

Zuikov)

- Do not allow non-deterministic functions in mutations of Replicated table engines, because this can introduce inconsistencies between replicas.

#7247 (Alexander

Kazakov)

- Disable memory tracker while converting exception stack trace to string. It can prevent the loss of error messages of type Memory limit exceeded on server, which caused the Attempt to read after eof exception on client. #7264

(Nikolai Kochetov)

- Miscellaneous format improvements. Resolves

#6033,

#2633,

#6611,

#6742

#7215

(tavplubix)

- ClickHouse ignores values on the right side of IN operator that are not convertible to the left side type. Make it work properly for compound types – Array and Tuple.

#7283 (Alexander

Kuzmenkov)

- Support missing inequalities for ASOF JOIN. It's possible to join less-or-equal variant and strict greater and less variants for ASOF column in ON syntax.

#7282 (Artem

Zuikov)

- Optimize partial merge join. #7070

(Artem Zuikov)

- Do not use more than 98K of memory in uniqCombined functions.

#7236,

#7270 (Azat

Khuzhin)

- Flush parts of right-hand joining table on disk in PartialMergeJoin (if there is not enough memory). Load data back when needed. [#7186](#)  
[\(Artem Zuikov\)](#)

## Performance Improvement

- Speed up joinGet with const arguments by avoiding data duplication.  
[#7359](#) ([Amos Bird](#))
- Return early if the subquery is empty.  
[#7007](#) ([小路](#))
- Optimize parsing of SQL expression in Values.  
[#6781](#)  
[\(tavplubix\)](#)

## Build/Testing/Packaging Improvement

- Disable some contribs for cross-compilation to Mac OS.  
[#7101](#) ([Ivan](#))
- Add missing linking with PocoXML for clickhouse\_common\_io.  
[#7200](#) ([Azat Khuzhin](#))
- Accept multiple test filter arguments in clickhouse-test.  
[#7226](#) ([Alexander Kuzmenkov](#))
- Enable musl and jemalloc for ARM. [#7300](#)  
([Amos Bird](#))
- Added --client-option parameter to `clickhouse-test` to pass additional parameters to client.  
[#7277](#) ([Nikolai Kochetov](#))
- Preserve existing configs on rpm package upgrade.  
[#7103](#)  
([filimonov](#))
- Fix errors detected by PVS. [#7153](#) ([Artem Zuikov](#))
- Fix build for Darwin. [#7149](#)  
([Ivan](#))
- glibc 2.29 compatibility. [#7142](#) ([Amos Bird](#))
- Make sure dh\_clean does not touch potential source files.  
[#7205](#) ([Amos Bird](#))
- Attempt to avoid conflict when updating from altinity rpm - it has config file packaged separately in clickhouse-server-common. [#7073](#)  
([filimonov](#))

- Optimize some header files for faster rebuilds.  
[#7212](#),  
[#7231 \(Alexander Kuzmenkov\)](#)
- Add performance tests for Date and DateTime. [#7332 \(Vasily Nemkov\)](#)
- Fix some tests that contained non-deterministic mutations.  
[#7132 \(Alexander Kazakov\)](#)
- Add build with MemorySanitizer to CI. [#7066 \(Alexander Kuzmenkov\)](#)
- Avoid use of uninitialized values in MetricsTransmitter.  
[#7158 \(Azat Khuzhin\)](#)
- Fix some issues in Fields found by MemorySanitizer.  
[#7135](#),  
[#7179 \(Alexander Kuzmenkov\), #7376 \(Amos Bird\)](#)
- Fix undefined behavior in murmurhash32. [#7388 \(Amos Bird\)](#)
- Fix undefined behavior in StoragesInfoStream. [#7384 \(tavplubix\)](#)
- Fixed constant expressions folding for external database engines (MySQL, ODBC, JDBC). In previous versions it wasn't working for multiple constant expressions and was not working at all for Date, DateTime and UUID. This fixes [#7245 #7252 \(alexey-milovidov\)](#)
- Fixing ThreadSanitizer data race error in the LIVE VIEW when accessing no\_users\_thread variable.  
[#7353 \(vzakaznikov\)](#)
- Get rid of malloc symbols in libcommon  
[#7134](#),  
[#7065 \(Amos Bird\)](#)
- Add global flag ENABLE\_LIBRARIES for disabling all libraries.  
[#7063 \(proller\)](#)

## Code Cleanup

- Generalize configuration repository to prepare for DDL for Dictionaries. [#7155 \(alesapin\)](#)
- Parser for dictionaries DDL without any semantic.  
[#7209 \(alesapin\)](#)

- Split ParserCreateQuery into different smaller parsers.  
[#7253](#)  
(alesapin)
- Small refactoring and renaming near external dictionaries.  
[#7111](#)  
(alesapin)
- Refactor some code to prepare for role-based access control. [#7235](#) ([Vitaly Baranov](#))
- Some improvements in DatabaseOrdinary code.  
[#7086](#) ([Nikita Vasilev](#))
- Do not use iterators in find() and emplace() methods of hash tables.  
[#7026](#) ([Alexander Kuzmenkov](#))
- Fix getMultipleValuesFromConfig in case when parameter root is not empty. [#7374](#) ([Mikhail Korotov](#))
- Remove some copy-paste (TemporaryFile and TemporaryFileStream)  
[#7166](#) ([Artem Zuikov](#))
- Improved code readability a little bit (MergeTreeData::getActiveContainingPart).  
[#7361](#) ([Vladimir Chebotarev](#))
- Wait for all scheduled jobs, which are using local objects, if ThreadPool::schedule(...) throws an exception. Rename ThreadPool::schedule(...) to ThreadPool::scheduleOrThrowOnError(...) and fix comments to make obvious that it may throw.  
[#7350](#)  
(tavplubix)

## ClickHouse Release 19.15

### ClickHouse Release 19.15.4.10, 2019-10-31

#### Bug Fix

- Added handling of SQL\_TINYINT and SQL\_BIGINT, and fix handling of SQL\_FLOAT data source types in ODBC Bridge.  
[#7491](#) ([Denis Glazachev](#))
- Allowed to have some parts on destination disk or volume in MOVE PARTITION.  
[#7434](#) ([Vladimir Chebotarev](#))
- Fixed NULL-values in nullable columns through ODBC-bridge.  
[#7402](#) ([Vasily Nemkov](#))
- Fixed INSERT into Distributed non local node with MATERIALIZED columns.  
[#7377](#) ([Azat Khuzhin](#))
- Fixed function getMultipleValuesFromConfig.  
[#7374](#) ([Mikhail Korotov](#))

- Fixed issue of using HTTP keep alive timeout instead of TCP keep alive timeout.  
[#7351 \(Vasily Nemkov\)](#)
- Wait for all jobs to finish on exception (fixes rare segfaults).  
[#7350 \(tavplubix\)](#)
- Don't push to MVs when inserting into Kafka table.  
[#7265 \(Ivan\)](#)
- Disable memory tracker for exception stack.  
[#7264 \(Nikolai Kochetov\)](#)
- Fixed bad code in transforming query for external database.  
[#7252 \(alexey-milovidov\)](#)
- Avoid use of uninitialized values in MetricsTransmitter.  
[#7158 \(Azat Khuzhin\)](#)
- Added example config with macros for tests ([alexey-milovidov](#))

## ClickHouse Release 19.15.3.6, 2019-10-09

### Bug Fix

- Fixed bad\_variant in hashed dictionary.  
[\(alesapin\)](#)
- Fixed up bug with segmentation fault in ATTACH PART query.  
[\(alesapin\)](#)
- Fixed time calculation in MergeTreeData.  
[\(Vladimir Chebotarev\)](#)
- Commit to Kafka explicitly after the writing is finalized.  
[#7175 \(Ivan\)](#)
- Serialize NULL values correctly in min/max indexes of MergeTree parts.  
[#7234 \(Alexander Kuzmenkov\)](#)

## ClickHouse Release 19.15.2.2, 2019-10-01

### New Feature

- Tiered storage: support to use multiple storage volumes for tables with MergeTree engine. It's possible to store fresh data on SSD and automatically move old data to HDD. ([example](#)). [#4918 \(Igr\)](#) [#6489 \(alesapin\)](#)
- Add table function `input` for reading incoming data in `INSERT SELECT` query. [#5450 \(palasonic1\)](#) [#6832 \(Anton Popov\)](#)
- Add a `sparse_hashed` dictionary layout, that is functionally equivalent to the `hashed` layout, but is more memory efficient. It uses about twice as less memory at the cost of slower value retrieval. [#6894 \(Azat Khuzhin\)](#)
- Implement ability to define list of users for access to dictionaries. Only current connected database using. [#6907 \(Guillaume Tassery\)](#)
- Add `LIMIT` option to `SHOW` query. [#6944 \(Philipp Malkovsky\)](#)
- Add `bitmapSubsetLimit(bitmap, range_start, limit)` function, that returns subset of the smallest `limit` values in set that is no smaller than `range_start`. [#6957 \(Zhichang Yu\)](#)

- Add `bitmapMin` and `bitmapMax` functions. #6970 (Zhichang Yu)
- Add function `repeat` related to issue-6648 #6999 (flynn)

## Experimental Feature

- Implement (in memory) Merge Join variant that does not change current pipeline. Result is partially sorted by merge key. Set `partial_merge_join = 1` to use this feature. The Merge Join is still in development. #6940 (Artem Zuikov)
- Add S3 engine and table function. It is still in development (no authentication support yet). #5596 (Vladimir Chebotarev)

## Improvement

- Every message read from Kafka is inserted atomically. This resolves almost all known issues with Kafka engine. #6950 (Ivan)
- Improvements for failover of Distributed queries. Shorten recovery time, also it is now configurable and can be seen in `system.clusters`. #6399 (Vasily Nemkov)
- Support numeric values for Enums directly in `IN` section. #6766 #6941 (dimarub2000)
- Support (optional, disabled by default) redirects on URL storage. #6914 (maqroll)
- Add information message when client with an older version connects to a server. #6893 (Philipp Malkovsky)
- Remove maximum backoff sleep time limit for sending data in Distributed tables #6895 (Azat Khuzhin)
- Add ability to send profile events (counters) with cumulative values to graphite. It can be enabled under `<events_cumulative>` in server `config.xml`. #6969 (Azat Khuzhin)
- Add automatically cast type `T` to `LowCardinality(T)` while inserting data in column of type `LowCardinality(T)` in Native format via HTTP. #6891 (Nikolai Kochetov)
- Add ability to use function `hex` without using `reinterpretAsString` for `Float32`, `Float64`. #7024 (Mikhail Korotov)

## Build/Testing/Packaging Improvement

- Add gdb-index to clickhouse binary with debug info. It will speed up startup time of gdb. #6947 (alesapin)
- Speed up deb packaging with patched dpkg-deb which uses pigz. #6960 (alesapin)
- Set `enable_fuzzing = 1` to enable libfuzzer instrumentation of all the project code. #7042 (kyprizel)
- Add split build smoke test in CI. #7061 (alesapin)
- Add build with MemorySanitizer to CI. #7066 (Alexander Kuzmenkov)
- Replace `libsparsehash` with `sparsehash-c11` #6965 (Azat Khuzhin)

## Bug Fix

- Fixed performance degradation of index analysis on complex keys on large tables. This fixes #6924. #7075 (alexey-milovidov)
- Fix logical error causing segfaults when selecting from Kafka empty topic. #6909 (Ivan)
- Fix too early MySQL connection close in `MySQLBlockInputStream.cpp`. #6882 (Clément Rodriguez)
- Returned support for very old Linux kernels (fix #6841) #6853 (alexey-milovidov)

- Fix possible data loss in `insert select` query in case of empty block in input stream. #6834 #6862 #6911 ([Nikolai Kochetov](#))
- Fix for function `ArrayEnumerateUniqRanked` with empty arrays in params #6928 ([proller](#))
- Fix complex queries with array joins and global subqueries. #6934 ([Ivan](#))
- Fix `Unknown identifier` error in `ORDER BY` and `GROUP BY` with multiple `JOINS` #7022 ([Artem Zuikov](#))
- Fixed `MSan` warning while executing function with `LowCardinality` argument. #7062 ([Nikolai Kochetov](#))

## Backward Incompatible Change

- Changed serialization format of `bitmap*` aggregate function states to improve performance. Serialized states of `bitmap*` from previous versions cannot be read. #6908 ([Zhichang Yu](#))

# ClickHouse Release 19.14

## ClickHouse Release 19.14.7.15, 2019-10-02

### Bug Fix

- This release also contains all bug fixes from 19.11.12.69.
- Fixed compatibility for distributed queries between 19.14 and earlier versions. This fixes #7068. #7069 ([alexey-milovidov](#))

## ClickHouse Release 19.14.6.12, 2019-09-19

### Bug Fix

- Fix for function `ArrayEnumerateUniqRanked` with empty arrays in params. #6928 ([proller](#))
- Fixed subquery name in queries with `ARRAY JOIN` and `GLOBAL IN subquery` with alias. Use subquery alias for external table name if it is specified. #6934 ([Ivan](#))

### Build/Testing/Packaging Improvement

- Fix `flapping` test `00715_fetch_merged_or_mutated_part_zookeeper` by rewriting it to a shell scripts because it needs to wait for mutations to apply. #6977 ([Alexander Kazakov](#))
- Fixed UBSan and MemSan failure in function `groupUniqArray` with empty array argument. It was caused by placing of empty `PaddedPODArray` into hash table zero cell because constructor for zero cell value was not called. #6937 ([Amos Bird](#))

## ClickHouse Release 19.14.3.3, 2019-09-10

### New Feature

- `WITH FILL` modifier for `ORDER BY`. (continuation of #5069) #6610 ([Anton Popov](#))
- `WITH TIES` modifier for `LIMIT`. (continuation of #5069) #6610 ([Anton Popov](#))
- Parse unquoted `NULL` literal as `NULL` (if setting `format_csv_unquoted_null_literal_as_null=1`). Initialize null fields with default values if data type of this field is not nullable (if setting `input_format_null_as_default=1`). #5990 #6055 ([tavplubix](#))
- Support for wildcards in paths of table functions `file` and `hdfs`. If the path contains wildcards, the table will be readonly. Example of usage: `select * from hdfs('hdfs://hdfs1:9000/some_dir/another_dir/*/*{0..9}{0..9}')` and `select * from file('some_dir/{some_file,another_file,yet_another}.tsv', 'TSV', 'value UInt32')`. #6092 ([Olga Khvostikova](#))

- New `system.metric_log` table which stores values of `system.events` and `system.metrics` with specified time interval. #6363 #6467 (Nikita Mikhaylov) #6530 (alexey-milovidov)
- Allow to write ClickHouse text logs to `system.text_log` table. #6037 #6103 (Nikita Mikhaylov) #6164 (alexey-milovidov)
- Show private symbols in stack traces (this is done via parsing symbol tables of ELF files). Added information about file and line number in stack traces if debug info is present. Speedup symbol name lookup with indexing symbols present in program. Added new SQL functions for introspection: `demangle` and `addressToLine`. Renamed function `symbolizeAddress` to `addressToSymbol` for consistency. Function `addressToSymbol` will return mangled name for performance reasons and you have to apply `demangle`. Added setting `allow_introspection_functions` which is turned off by default. #6201 (alexey-milovidov)
- Table function `values` (the name is case-insensitive). It allows to read from `VALUES` list proposed in #5984. Example: `SELECT * FROM VALUES('a UInt64, s String', (1, 'one'), (2, 'two'), (3, 'three'))` #6217. #6209 (dimarub2000)
- Added an ability to alter storage settings. Syntax: `ALTER TABLE <table> MODIFY SETTING <setting> = <value>`. #6366 #6669 #6685 (alesapin)
- Support for removing of detached parts. Syntax: `ALTER TABLE <table_name> DROP DETACHED PART '<part_id>'`. #6158 (tavplubix)
- Table constraints. Allows to add constraint to table definition which will be checked at insert. #5273 (Gleb Novikov) #6652 (alexey-milovidov)
- Support for cascaded materialized views. #6324 (Amos Bird)
- Turn on query profiler by default to sample every query execution thread once a second. #6283 (alexey-milovidov)
- Input format `ORC`. #6454 #6703 (akonyaev90)
- Added two new functions: `sigmoid` and `tanh` (that are useful for machine learning applications). #6254 (alexey-milovidov)
- Function `hasToken(haystack, token)`, `hasTokenCaseInsensitive(haystack, token)` to check if given token is in haystack. Token is a maximal length substring between two non alphanumeric ASCII characters (or boundaries of haystack). Token must be a constant string. Supported by `tokenbf_v1` index specialization. #6596, #6662 (Vasily Nemkov)
- New function `neighbor(value, offset[, default_value])`. Allows to reach prev/next value within column in a block of data. #5925 (Alex Krash) 6685365ab8c5b74f9650492c88a012596eb1b0c6 341e2e4587a18065c2da1ca888c73389f48ce36c Alexey Milovidov
- Created a function `currentUser()`, returning login of authorized user. Added alias `user()` for compatibility with MySQL. #6470 (Alex Krash)
- New aggregate functions `quantilesExactInclusive` and `quantilesExactExclusive` which were proposed in #5885. #6477 (dimarub2000)
- Function `bitmapRange(bitmap, range_begin, range_end)` which returns new set with specified range (not include the `range_end`). #6314 (Zhichang Yu)
- Function `geohashesInBox(longitude_min, latitude_min, longitude_max, latitude_max, precision)` which creates array of precision-long strings of geohash-boxes covering provided area. #6127 (Vasily Nemkov)
- Implement support for `INSERT` query with `Kafka` tables. #6012 (Ivan)
- Added support for `_partition` and `_timestamp` virtual columns to Kafka engine. #6400 (Ivan)

- Possibility to remove sensitive data from `query_log`, server logs, process list with regexp-based rules. #5710 (filimonov)

## Experimental Feature

- Input and output data format `Template`. It allows to specify custom format string for input and output. #4354 #6727 (tavplubix)
- Implementation of `LIVE VIEW` tables that were originally proposed in #2898, prepared in #3925, and then updated in #5541. See #5541 for detailed description. #5541 (vzakaznikov) #6425 (Nikolai Kochetov) #6656 (vzakaznikov) Note that `LIVE VIEW` feature may be removed in next versions.

## Bug Fix

- This release also contains all bug fixes from 19.13 and 19.11.
- Fix segmentation fault when the table has skip indices and vertical merge happens. #6723 (alesapin)
- Fix per-column TTL with non-trivial column defaults. Previously in case of force TTL merge with `OPTIMIZE ... FINAL` query, expired values was replaced by type defaults instead of user-specified column defaults. #6796 (Anton Popov)
- Fix Kafka messages duplication problem on normal server restart. #6597 (Ivan)
- Fixed infinite loop when reading Kafka messages. Do not pause/resume consumer on subscription at all - otherwise it may get paused indefinitely in some scenarios. #6354 (Ivan)
- Fix `Key expression contains comparison between inconvertible types` exception in `bitmapContains` function. #6136 #6146 #6156 (dimarub2000)
- Fix segfault with enabled `optimize_skip_unused_shards` and missing sharding key. #6384 (Anton Popov)
- Fixed wrong code in mutations that may lead to memory corruption. Fixed segfault with read of address `0x14c0` that may happed due to concurrent `DROP TABLE` and `SELECT` from `system.parts` or `system.parts_columns`. Fixed race condition in preparation of mutation queries. Fixed deadlock caused by `OPTIMIZE` of Replicated tables and concurrent modification operations like `ALTERs`. #6514 (alexey-milovidov)
- Removed extra verbose logging in MySQL interface #6389 (alexey-milovidov)
- Return the ability to parse boolean settings from 'true' and 'false' in the configuration file. #6278 (alesapin)
- Fix crash in `quantile` and `median` function over `Nullable(Decimal128)`. #6378 (Artem Zuikov)
- Fixed possible incomplete result returned by `SELECT` query with `WHERE` condition on primary key contained conversion to `Float` type. It was caused by incorrect checking of monotonicity in `toFloat` function. #6248 #6374 (dimarub2000)
- Check `max_expanded_ast_elements` setting for mutations. Clear mutations after `TRUNCATE TABLE`. #6205 (Winter Zhang)
- Fix JOIN results for key columns when used with `join_use_nulls`. Attach Nulls instead of columns defaults. #6249 (Artem Zuikov)
- Fix for skip indices with vertical merge and alter. Fix for `Bad size of marks file` exception. #6594 #6713 (alesapin)
- Fix rare crash in `ALTER MODIFY COLUMN` and vertical merge when one of merged/altered parts is empty (0 rows) #6746 #6780 (alesapin)

- Fixed bug in conversion of `LowCardinality` types in `AggregateFunctionFactory`. This fixes #6257. #6281 ([Nikolai Kochetov](#))
- Fix wrong behavior and possible segfaults in `topK` and `topKWeighted` aggregated functions. #6404 ([Anton Popov](#))
- Fixed unsafe code around `getIdentifier` function. #6401 #6409 ([alexey-milovidov](#))
- Fixed bug in MySQL wire protocol (is used while connecting to ClickHouse from MySQL client). Caused by heap buffer overflow in `PacketPayloadWriteBuffer`. #6212 ([Yuriy Baranov](#))
- Fixed memory leak in `bitmapSubsetInRange` function. #6819 ([Zhichang Yu](#))
- Fix rare bug when mutation executed after granularity change. #6816 ([alesapin](#))
- Allow protobuf message with all fields by default. #6132 ([Vitaly Baranov](#))
- Resolve a bug with `nullIf` function when we send a `NULL` argument on the second argument. #6446 ([Guillaume Tassery](#))
- Fix rare bug with wrong memory allocation/deallocation in complex key cache dictionaries with string fields which leads to infinite memory consumption (looks like memory leak). Bug reproduces when string size was a power of two starting from eight (8, 16, 32, etc). #6447 ([alesapin](#))
- Fixed Gorilla encoding on small sequences which caused exception `Cannot write after end of buffer`. #6398 #6444 ([Vasily Nemkov](#))
- Allow to use not nullable types in JOINS with `join_use_nulls` enabled. #6705 ([Artem Zuikov](#))
- Disable `Poco::AbstractConfiguration` substitutions in query in `clickhouse-client`. #6706 ([alexey-milovidov](#))
- Avoid deadlock in `REPLACE PARTITION`. #6677 ([alexey-milovidov](#))
- Using `arrayReduce` for constant arguments may lead to segfault. #6242 #6326 ([alexey-milovidov](#))
- Fix inconsistent parts which can appear if replica was restored after `DROP PARTITION`. #6522 #6523 ([tavplubix](#))
- Fixed hang in `JSONExtractRaw` function. #6195 #6198 ([alexey-milovidov](#))
- Fix bug with incorrect skip indices serialization and aggregation with adaptive granularity. #6594. #6748 ([alesapin](#))
- Fix WITH ROLLUP and WITH CUBE modifiers of GROUP BY with two-level aggregation. #6225 ([Anton Popov](#))
- Fix bug with writing secondary indices marks with adaptive granularity. #6126 ([alesapin](#))
- Fix initialization order while server startup. Since `StorageMergeTree::background_task_handle` is initialized in `startup()` the `MergeTreeBlockOutputStream::write()` may try to use it before initialization. Just check if it is initialized. #6080 ([Ivan](#))
- Clearing the data buffer from the previous read operation that was completed with an error. #6026 ([Nikolay](#))
- Fix bug with enabling adaptive granularity when creating a new replica for Replicated\*MergeTree table. #6394 #6452 ([alesapin](#))
- Fixed possible crash during server startup in case of exception happened in `libunwind` during exception at access to uninitialized `ThreadStatus` structure. #6456 ([Nikita Mikhaylov](#))
- Fix crash in `yandexConsistentHash` function. Found by fuzz test. #6304 #6305 ([alexey-milovidov](#))

- Fixed the possibility of hanging queries when server is overloaded and global thread pool becomes near full. This have higher chance to happen on clusters with large number of shards (hundreds), because distributed queries allocate a thread per connection to each shard. For example, this issue may reproduce if a cluster of 330 shards is processing 30 concurrent distributed queries. This issue affects all versions starting from 19.2. [#6301 \(alexey-milovidov\)](#)
- Fixed logic of `arrayEnumerateUniqRanked` function. [#6423 \(alexey-milovidov\)](#)
- Fix segfault when decoding symbol table. [#6603 \(Amos Bird\)](#)
- Fixed irrelevant exception in cast of `LowCardinalityNullable` to not-`Nullable` column in case if it does not contain Nulls (e.g. in query like `SELECT CAST(CAST('Hello' AS LowCardinalityNullable(String))) AS String`). [#6094 #6119 \(Nikolai Kochetov\)](#)
- Removed extra quoting of description in `system.settings` table. [#6696 #6699 \(alexey-milovidov\)](#)
- Avoid possible deadlock in `TRUNCATE` of Replicated table. [#6695 \(alexey-milovidov\)](#)
- Fix reading in order of sorting key. [#6189 \(Anton Popov\)](#)
- Fix `ALTER TABLE ... UPDATE` query for tables with `enable_mixed_granularity_parts=1`. [#6543 \(alesapin\)](#)
- Fix bug opened by [#4405](#) (since 19.4.0). Reproduces in queries to Distributed tables over MergeTree tables when we does not query any columns (`SELECT 1`). [#6236 \(alesapin\)](#)
- Fixed overflow in integer division of signed type to unsigned type. The behaviour was exactly as in C or C++ language (integer promotion rules) that may be surprising. Please note that the overflow is still possible when dividing large signed number to large unsigned number or vice-versa (but that case is less usual). The issue existed in all server versions. [#6214 #6233 \(alexey-milovidov\)](#)
- Limit maximum sleep time for throttling when `max_execution_speed` or `max_execution_speed_bytes` is set. Fixed false errors like `Estimated query execution time (inf seconds) is too long.` [#5547 #6232 \(alexey-milovidov\)](#)
- Fixed issues about using `MATERIALIZED` columns and aliases in `MaterializedView`. [#448 #3484 #3450 #2878 #2285 #3796 \(Amos Bird\)](#) [#6316 \(alexey-milovidov\)](#)
- Fix `FormatFactory` behaviour for input streams which are not implemented as processor. [#6495 \(Nikolai Kochetov\)](#)
- Fixed typo. [#6631 \(Alex Ryndin\)](#)
- Typo in the error message ( `is` -> `are` ). [#6839 \(Denis Zhuravlev\)](#)
- Fixed error while parsing of columns list from string if type contained a comma (this issue was relevant for `File`, `URL`, `HDFS` storages) [#6217. #6209 \(dimarub2000\)](#)

## Security Fix

- This release also contains all bug security fixes from 19.13 and 19.11.
- Fixed the possibility of a fabricated query to cause server crash due to stack overflow in SQL parser. Fixed the possibility of stack overflow in Merge and Distributed tables, materialized views and conditions for row-level security that involve subqueries. [#6433 \(alexey-milovidov\)](#)

## Improvement

- Correct implementation of ternary logic for AND/OR. [#6048 \(Alexander Kazakov\)](#)

- Now values and rows with expired TTL will be removed after `OPTIMIZE ... FINAL` query from old parts without TTL infos or with outdated TTL infos, e.g. after `ALTER ... MODIFY TTL` query. Added queries `SYSTEM STOP/START TTL MERGES` to disallow/allow assign merges with TTL and filter expired values in all merges. [#6274 \(Anton Popov\)](#)
- Possibility to change the location of ClickHouse history file for client using `CLOCKHOUSE_HISTORY_FILE` env. [#6840 \(filimonov\)](#)
- Remove `dry_run` flag from `InterpreterSelectQuery`. ... [#6375 \(Nikolai Kochetov\)](#)
- Support `ASOF JOIN` with `ON` section. [#6211 \(Artem Zuikov\)](#)
- Better support of skip indexes for mutations and replication. Support for `MATERIALIZE/CLEAR INDEX ... IN PARTITION` query. `UPDATE x = x` recalculates all indices that use column `x`. [#5053 \(Nikita Vasilev\)](#)
- Allow to `ATTACH` live views (for example, at the server startup) regardless to `allow_experimental_live_view` setting. [#6754 \(alexey-milovidov\)](#)
- For stack traces gathered by query profiler, do not include stack frames generated by the query profiler itself. [#6250 \(alexey-milovidov\)](#)
- Now table functions `values`, `file`, `url`, `hdfs` have support for ALIAS columns. [#6255 \(alexey-milovidov\)](#)
- Throw an exception if `config.d` file does not have the corresponding root element as the config file. [#6123 \(dimarub2000\)](#)
- Print extra info in exception message for no space left on device [#6182](#), [#6252](#) [#6352 \(tavplubix\)](#)
- When determining shards of a `Distributed` table to be covered by a read query (for `optimize_skip_unused_shards = 1`) ClickHouse now checks conditions from both `prewhere` and `where` clauses of select statement. [#6521 \(Alexander Kazakov\)](#)
- Enabled `SIMDJSON` for machines without AVX2 but with SSE 4.2 and PCLMUL instruction set. [#6285](#) [#6320 \(alexey-milovidov\)](#)
- ClickHouse can work on filesystems without `O_DIRECT` support (such as ZFS and Btrfs) without additional tuning. [#4449](#) [#6730 \(alexey-milovidov\)](#)
- Support push down predicate for final subquery. [#6120 \(TCeason\)](#) [#6162 \(alexey-milovidov\)](#)
- Better `JOIN ON` keys extraction [#6131 \(Artem Zuikov\)](#)
- Updated `SIMDJSON`. [#6285](#). [#6306 \(alexey-milovidov\)](#)
- Optimize selecting of smallest column for `SELECT count()` query. [#6344 \(Amos Bird\)](#)
- Added `strict` parameter in `windowFunnel()`. When the `strict` is set, the `windowFunnel()` applies conditions only for the unique values. [#6548 \(achimbab\)](#)
- Safer interface of `mysqlxx::Pool`. [#6150 \(avasiliev\)](#)
- Options line size when executing with `--help` option now corresponds with terminal size. [#6590 \(dimarub2000\)](#)
- Disable “read in order” optimization for aggregation without keys. [#6599 \(Anton Popov\)](#)
- HTTP status code for `INCORRECT_DATA` and `TYPE_MISMATCH` error codes was changed from default 500 Internal Server Error to 400 Bad Request. [#6271 \(Alexander Rodin\)](#)
- Move `Join` object from `ExpressionAction` into `AnalyzedJoin`. `ExpressionAnalyzer` and `ExpressionAction` do not know about `Join` class anymore. Its logic is hidden by `AnalyzedJoin` iface. [#6801 \(Artem Zuikov\)](#)

- Fixed possible deadlock of distributed queries when one of shards is localhost but the query is sent via network connection. [#6759](#) ([alexey-milovidov](#))
- Changed semantic of multiple tables `RENAME` to avoid possible deadlocks. [#6757](#). [#6756](#) ([alexey-milovidov](#))
- Rewritten MySQL compatibility server to prevent loading full packet payload in memory. Decreased memory consumption for each connection to approximately `2 * DBMS_DEFAULT_BUFFER_SIZE` (read/write buffers). [#5811](#) ([Yuriy Baranov](#))
- Move AST alias interpreting logic out of parser that does not have to know anything about query semantics. [#6108](#) ([Artem Zuikov](#))
- Slightly more safe parsing of `NamesAndTypesList`. [#6408](#). [#6410](#) ([alexey-milovidov](#))
- `clickhouse-copier`: Allow use `where_condition` from config with `partition_key` alias in query for checking partition existence (Earlier it was used only in reading data queries). [#6577](#) ([proller](#))
- Added optional message argument in `throwIf`. ([#5772](#)) [#6329](#) ([Vdimir](#))
- Server exception got while sending insertion data is now being processed in client as well. [#5891](#) [#6711](#) ([dimarub2000](#))
- Added a metric `DistributedFilesToInsert` that shows the total number of files in filesystem that are selected to send to remote servers by Distributed tables. The number is summed across all shards. [#6600](#) ([alexey-milovidov](#))
- Move most of JOINS prepare logic from `ExpressionAction/ExpressionAnalyzer` to `AnalyzedJoin`. [#6785](#) ([Artem Zuikov](#))
- Fix TSan warning ‘lock-order-inversion’. [#6740](#) ([Vasily Nemkov](#))
- Better information messages about lack of Linux capabilities. Logging fatal errors with “fatal” level, that will make it easier to find in `system.text_log`. [#6441](#) ([alexey-milovidov](#))
- When enable dumping temporary data to the disk to restrict memory usage during `GROUP BY`, `ORDER BY`, it didn’t check the free disk space. The fix add a new setting `min_free_disk_space`, when the free disk space is smaller then the threshold, the query will stop and throw `ErrorCodes::NOT_ENOUGH_SPACE`. [#6678](#) ([Weiqing Xu](#)) [#6691](#) ([alexey-milovidov](#))
- Removed recursive rwlock by thread. It makes no sense, because threads are reused between queries. `SELECT` query may acquire a lock in one thread, hold a lock from another thread and exit from first thread. In the same time, first thread can be reused by `DROP` query. This will lead to false “Attempt to acquire exclusive lock recursively” messages. [#6771](#) ([alexey-milovidov](#))
- Split `ExpressionAnalyzer.appendJoin()`. Prepare a place in `ExpressionAnalyzer` for `MergeJoin`. [#6524](#) ([Artem Zuikov](#))
- Added `mysql_native_password` authentication plugin to MySQL compatibility server. [#6194](#) ([Yuriy Baranov](#))
- Less number of `clock_gettime` calls; fixed ABI compatibility between debug/release in `Allocator` (insignificant issue). [#6197](#) ([alexey-milovidov](#))
- Move `collectUsedColumns` from `ExpressionAnalyzer` to `SyntaxAnalyzer`. `SyntaxAnalyzer` makes `required_source_columns` itself now. [#6416](#) ([Artem Zuikov](#))
- Add setting `joined_subquery_requires_alias` to require aliases for subselects and table functions in `FROM` that more than one table is present (i.e. queries with JOINS). [#6733](#) ([Artem Zuikov](#))
- Extract `GetAggregatesVisitor` class from `ExpressionAnalyzer`. [#6458](#) ([Artem Zuikov](#))

- system.query\_log: change data type of type column to Enum. #6265 (Nikita Mikhaylov)
- Static linking of sha256\_password authentication plugin. #6512 (Yuriy Baranov)
- Avoid extra dependency for the setting compile to work. In previous versions, the user may get error like cannot open crtI.o, unable to find library -lc etc. #6309 (alexey-milovidov)
- More validation of the input that may come from malicious replica. #6303 (alexey-milovidov)
- Now clickhouse-obfuscator file is available in clickhouse-client package. In previous versions it was available as clickhouse obfuscator (with whitespace). #5816 #6609 (dimarub2000)
- Fixed deadlock when we have at least two queries that read at least two tables in different order and another query that performs DDL operation on one of tables. Fixed another very rare deadlock. #6764 (alexey-milovidov)
- Added os\_thread\_ids column to system.processes and system.query\_log for better debugging possibilities. #6763 (alexey-milovidov)
- A workaround for PHP mysqlnd extension bugs which occur when sha256\_password is used as a default authentication plugin (described in #6031). #6113 (Yuriy Baranov)
- Remove unneeded place with changed nullability columns. #6693 (Artem Zuikov)
- Set default value of queue\_max\_wait\_ms to zero, because current value (five seconds) makes no sense. There are rare circumstances when this settings has any use. Added settings replace\_running\_query\_max\_wait\_ms, kafka\_max\_wait\_ms and connection\_pool\_max\_wait\_ms for disambiguation. #6692 (alexey-milovidov)
- Extract SelectQueryExpressionAnalyzer from ExpressionAnalyzer. Keep the last one for non-select queries. #6499 (Artem Zuikov)
- Removed duplicating input and output formats. #6239 (Nikolai Kochetov)
- Allow user to override poll\_interval and idle\_connection\_timeout settings on connection. #6230 (alexey-milovidov)
- MergeTree now has an additional option ttl\_only\_drop\_parts (disabled by default) to avoid partial pruning of parts, so that they dropped completely when all the rows in a part are expired. #6191 (Sergi Vladyskin)
- Type checks for set index functions. Throw exception if function got a wrong type. This fixes fuzz test with UBSan. #6511 (Nikita Vasilev)

## Performance Improvement

- Optimize queries with ORDER BY expressions clause, where expressions have coinciding prefix with sorting key in MergeTree tables. This optimization is controlled by optimize\_read\_in\_order setting. #6054 #6629 (Anton Popov)
- Allow to use multiple threads during parts loading and removal. #6372 #6074 #6438 (alexey-milovidov)
- Implemented batch variant of updating aggregate function states. It may lead to performance benefits. #6435 (alexey-milovidov)
- Using FastOps library for functions exp, log, sigmoid, tanh. FastOps is a fast vector math library from Michael Parakhin (Yandex CTO). Improved performance of exp and log functions more than 6 times. The functions exp and log from Float32 argument will return Float32 (in previous versions they always return Float64). Now exp(nan) may return inf. The result of exp and log functions may be not the nearest machine representable number to the true answer. #6254 (alexey-milovidov) Using Danila Kutenin variant to make fastops working #6317 (alexey-milovidov)

- Disable consecutive key optimization for UInt8/16. #6298 #6701 (akuzm)
- Improved performance of `simdjson` library by getting rid of dynamic allocation in `ParsedJson::Iterator`. #6479 (Vitaly Baranov)
- Pre-fault pages when allocating memory with `mmap()`. #6667 (akuzm)
- Fix performance bug in `Decimal` comparison. #6380 (Artem Zuikov)

## Build/Testing/Packaging Improvement

- Remove Compiler (runtime template instantiation) because we've win over it's performance. #6646 (alexey-milovidov)
- Added performance test to show degradation of performance in gcc-9 in more isolated way. #6302 (alexey-milovidov)
- Added table function `numbers_mt`, which is multithreaded version of `numbers`. Updated performance tests with hash functions. #6554 (Nikolai Kochetov)
- Comparison mode in `clickhouse-benchmark` #6220 #6343 (dimarub2000)
- Best effort for printing stack traces. Also added `SIGPROF` as a debugging signal to print stack trace of a running thread. #6529 (alexey-milovidov)
- Every function in its own file, part 10. #6321 (alexey-milovidov)
- Remove doubled const `TABLE_IS_READ_ONLY`. #6566 (filimonov)
- Formatting changes for `StringHashMap` PR #5417. #6700 (akuzm)
- Better subquery for join creation in `ExpressionAnalyzer`. #6824 (Artem Zuikov)
- Remove a redundant condition (found by PVS Studio). #6775 (akuzm)
- Separate the hash table interface for `ReverseIndex`. #6672 (akuzm)
- Refactoring of settings. #6689 (alesapin)
- Add comments for `set` index functions. #6319 (Nikita Vasilev)
- Increase OOM score in debug version on Linux. #6152 (akuzm)
- HDFS HA now work in debug build. #6650 (Weiqing Xu)
- Added a test to `transform_query_for_external_database`. #6388 (alexey-milovidov)
- Add test for multiple materialized views for Kafka table. #6509 (Ivan)
- Make a better build scheme. #6500 (Ivan)
- Fixed `test_external_dictionaries` integration in case it was executed under non root user. #6507 (Nikolai Kochetov)
- The bug reproduces when total size of written packets exceeds `DBMS_DEFAULT_BUFFER_SIZE`. #6204 (Yuriy Baranov)
- Added a test for `RENAME` table race condition #6752 (alexey-milovidov)
- Avoid data race on Settings in `KILL QUERY`. #6753 (alexey-milovidov)
- Add integration test for handling errors by a cache dictionary. #6755 (Vitaly Baranov)
- Disable parsing of ELF object files on Mac OS, because it makes no sense. #6578 (alexey-milovidov)

- Attempt to make changelog generator better. #6327 (alexey-milovidov)
- Adding `-Wshadow` switch to the GCC. #6325 (kreuzerkrieg)
- Removed obsolete code for `mimalloc` support. #6715 (alexey-milovidov)
- `zlib-ng` determines x86 capabilities and saves this info to global variables. This is done in `defalteInit` call, which may be made by different threads simultaneously. To avoid multithreaded writes, do it on library startup. #6141 (akuzm)
- Regression test for a bug which in `join` which was fixed in #5192. #6147 (Bakhtiyor Ruziev)
- Fixed MSan report. #6144 (alexey-milovidov)
- Fix flapping TTL test. #6782 (Anton Popov)
- Fixed false data race in `MergeTreeDataPart::is_frozen` field. #6583 (alexey-milovidov)
- Fixed timeouts in fuzz test. In previous version, it managed to find false hangup in query `SELECT * FROM numbers_mt(gccMurmurHash(""))`. #6582 (alexey-milovidov)
- Added debug checks to `static_cast` of columns. #6581 (alexey-milovidov)
- Support for Oracle Linux in official RPM packages. #6356 #6585 (alexey-milovidov)
- Changed json perftests from `once` to `loop` type. #6536 (Nikolai Kochetov)
- `odbc-bridge.cpp` defines `main()` so it should not be included in `clickhouse-lib`. #6538 (Orivej Desh)
- Test for crash in `FULL|RIGHT JOIN` with nulls in right table's keys. #6362 (Artem Zuikov)
- Added a test for the limit on expansion of aliases just in case. #6442 (alexey-milovidov)
- Switched from `boost::filesystem` to `std::filesystem` where appropriate. #6253 #6385 (alexey-milovidov)
- Added RPM packages to website. #6251 (alexey-milovidov)
- Add a test for fixed `Unknown identifier` exception in `IN` section. #6708 (Artem Zuikov)
- Simplify `shared_ptr_helper` because people facing difficulties understanding it. #6675 (alexey-milovidov)
- Added performance tests for fixed Gorilla and DoubleDelta codec. #6179 (Vasily Nemkov)
- Split the integration test `test_dictionaries` into 4 separate tests. #6776 (Vitaly Baranov)
- Fix PVS-Studio warning in `PipelineExecutor`. #6777 (Nikolai Kochetov)
- Allow to use `library` dictionary source with ASan. #6482 (alexey-milovidov)
- Added option to generate changelog from a list of PRs. #6350 (alexey-milovidov)
- Lock the `TinyLog` storage when reading. #6226 (akuzm)
- Check for broken symlinks in CI. #6634 (alexey-milovidov)
- Increase timeout for “stack overflow” test because it may take a long time in debug build. #6637 (alexey-milovidov)
- Added a check for double whitespaces. #6643 (alexey-milovidov)
- Fix `new/delete` memory tracking when build with sanitizers. Tracking is not clear. It only prevents memory limit exceptions in tests. #6450 (Artem Zuikov)
- Enable back the check of undefined symbols while linking. #6453 (Ivan)

- Avoid rebuilding `hyperscan` every day. [#6307 \(alexey-milovidov\)](#)
- Fixed UBSan report in `ProtobufWriter`. [#6163 \(alexey-milovidov\)](#)
- Don't allow to use query profiler with sanitizers because it is not compatible. [#6769 \(alexey-milovidov\)](#)
- Add test for reloading a dictionary after fail by timer. [#6114 \(Vitaly Baranov\)](#)
- Fix inconsistency in `PipelineExecutor::prepareProcessor` argument type. [#6494 \(Nikolai Kochetov\)](#)
- Added a test for bad URLs. [#6493 \(alexey-milovidov\)](#)
- Added more checks to `CAST` function. This should get more information about segmentation fault in fuzzy test. [#6346 \(Nikolai Kochetov\)](#)
- Added `gcc-9` support to `docker/builder` container that builds image locally. [#6333 \(Gleb Novikov\)](#)
- Test for primary key with `LowCardinality(String)`. [#5044 #6219 \(dimarub2000\)](#)
- Fixed tests affected by slow stack traces printing. [#6315 \(alexey-milovidov\)](#)
- Add a test case for crash in `groupUniqArray` fixed in [#6029. #4402 #6129 \(akuzm\)](#)
- Fixed indices mutations tests. [#6645 \(Nikita Vasilev\)](#)
- In performance test, do not read query log for queries we didn't run. [#6427 \(akuzm\)](#)
- Materialized view now could be created with any low cardinality types regardless to the setting about suspicious low cardinality types. [#6428 \(Olga Khvostikova\)](#)
- Updated tests for `send_logs_level` setting. [#6207 \(Nikolai Kochetov\)](#)
- Fix build under gcc-8.2. [#6196 \(Max Akhmedov\)](#)
- Fix build with internal libc++. [#6724 \(Ivan\)](#)
- Fix shared build with `rdkafka` library [#6101 \(Ivan\)](#)
- Fixes for Mac OS build (incomplete). [#6390 \(alexey-milovidov\) #6429 \(alex-zaitsev\)](#)
- Fix "splitted" build. [#6618 \(alexey-milovidov\)](#)
- Other build fixes: [#6186 \(Amos Bird\) #6486 #6348 \(vxider\) #6744 \(Ivan\) #6016 #6421 #6491 \(proller\)](#)

## Backward Incompatible Change

- Removed rarely used table function `catBoostPool` and storage `CatBoostPool`. If you have used this table function, please write email to `clickhouse-feedback@yandex-team.com`. Note that CatBoost integration remains and will be supported. [#6279 \(alexey-milovidov\)](#)
- Disable `ANY RIGHT JOIN` and `ANY FULL JOIN` by default. Set `any_join_distinct_right_table_keys` setting to enable them. [#5126 #6351 \(Artem Zuikov\)](#)

## ClickHouse Release 19.13

### ClickHouse Release 19.13.6.51, 2019-10-02

#### Bug Fix

- This release also contains all bug fixes from 19.11.12.69.

### ClickHouse Release 19.13.5.44, 2019-09-20

#### Bug Fix

- This release also contains all bug fixes from 19.14.6.12.
- Fixed possible inconsistent state of table while executing `DROP` query for replicated table while zookeeper is not accessible. #6045 #6413 (Nikita Mikhaylov)
- Fix for data race in StorageMerge #6717 (alexey-milovidov)
- Fix bug introduced in query profiler which leads to endless recv from socket. #6386 (alesapin)
- Fix excessive CPU usage while executing `JSONExtractRaw` function over a boolean value. #6208 (Vitaly Baranov)
- Fixes the regression while pushing to materialized view. #6415 (Ivan)
- Table function `url` had the vulnerability allowed the attacker to inject arbitrary HTTP headers in the request. This issue was found by Nikita Tikhomirov. #6466 (alexey-milovidov)
- Fix useless `AST` check in Set index. #6510 #6651 (Nikita Vasilev)
- Fixed parsing of `AggregateFunction` values embedded in query. #6575 #6773 (Zhichang Yu)
- Fixed wrong behaviour of `trim` functions family. #6647 (alexey-milovidov)

## ClickHouse Release 19.13.4.32, 2019-09-10

### Bug Fix

- This release also contains all bug security fixes from 19.11.9.52 and 19.11.10.54.
- Fixed data race in `system.parts` table and `ALTER` query. #6245 #6513 (alexey-milovidov)
- Fixed mismatched header in streams happened in case of reading from empty distributed table with sample and prewhere. #6167 (Lixiang Qian) #6823 (Nikolai Kochetov)
- Fixed crash when using `IN` clause with a subquery with a tuple. #6125 #6550 (tavplubix)
- Fix case with same column names in `GLOBAL JOIN ON` section. #6181 (Artem Zuikov)
- Fix crash when casting types to `Decimal` that do not support it. Throw exception instead. #6297 (Artem Zuikov)
- Fixed crash in `extractAll()` function. #6644 (Artem Zuikov)
- Query transformation for MySQL, ODBC, JDBC table functions now works properly for `SELECT WHERE` queries with multiple `AND` expressions. #6381 #6676 (dimarub2000)
- Added previous declaration checks for MySQL 8 integration. #6569 (Rafael David Tinoco)

### Security Fix

- Fix two vulnerabilities in codecs in decompression phase (malicious user can fabricate compressed data that will lead to buffer overflow in decompression). #6670 (Artem Zuikov)

## ClickHouse Release 19.13.3.26, 2019-08-22

### Bug Fix

- Fix `ALTER TABLE ... UPDATE` query for tables with `enable_mixed_granularity_parts=1`. #6543 (alesapin)
- Fix NPE when using `IN` clause with a subquery with a tuple. #6125 #6550 (tavplubix)
- Fixed an issue that if a stale replica becomes alive, it may still have data parts that were removed by `DROP PARTITION`. #6522 #6523 (tavplubix)

- Fixed issue with parsing CSV [#6426](#) [#6559](#) ([tavplubix](#))
- Fixed data race in system.parts table and ALTER query. This fixes [#6245](#), [#6513](#) ([alexey-milovidov](#))
- Fixed wrong code in mutations that may lead to memory corruption. Fixed segfault with read of address `0x14c0` that may happen due to concurrent `DROP TABLE` and `SELECT` from `system.parts` or `system.parts_columns`. Fixed race condition in preparation of mutation queries. Fixed deadlock caused by `OPTIMIZE` of Replicated tables and concurrent modification operations like ALTERs. [#6514](#) ([alexey-milovidov](#))
- Fixed possible data loss after `ALTER DELETE` query on table with skipping index. [#6224](#) [#6282](#) ([Nikita Vasilev](#))

## Security Fix

- If the attacker has write access to ZooKeeper and is able to run custom server available from the network where ClickHouse runs, it can create custom-built malicious server that will act as ClickHouse replica and register it in ZooKeeper. When another replica will fetch data part from malicious replica, it can force clickhouse-server to write to arbitrary path on filesystem. Found by Eldar Zaitov, information security team at Yandex. [#6247](#) ([alexey-milovidov](#))

## ClickHouse Release 19.13.2.19, 2019-08-14

### New Feature

- Sampling profiler on query level. Example. [#4247](#) ([laplab](#)) [#6124](#) ([alexey-milovidov](#)) [#6250](#) [#6283](#) [#6386](#)
- Allow to specify a list of columns with `COLUMNS('regexp')` expression that works like a more sophisticated variant of `*` asterisk. [#5951](#) ([mfridental](#)), ([alexey-milovidov](#))
- `CREATE TABLE AS table_function()` is now possible [#6057](#) ([dimarub2000](#))
- Adam optimizer for stochastic gradient descent is used by default in `stochasticLinearRegression()` and `stochasticLogisticRegression()` aggregate functions, because it shows good quality without almost any tuning. [#6000](#) ([Quid37](#))
- Added functions for working with the custom week number [#5212](#) ([Andy Yang](#))
- `RENAME` queries now work with all storages. [#5953](#) ([Ivan](#))
- Now client receive logs from server with any desired level by setting `send_logs_level` regardless to the log level specified in server settings. [#5964](#) ([Nikita Mikhaylov](#))

### Backward Incompatible Change

- The setting `input_format_defaults_for_omitted_fields` is enabled by default. Inserts in Distributed tables need this setting to be the same on cluster (you need to set it before rolling update). It enables calculation of complex default expressions for omitted fields in `JSONEachRow` and `CSV*` formats. It should be the expected behavior but may lead to negligible performance difference. [#6043](#) ([Artem Zuikov](#)), [#5625](#) ([akuzm](#))

### Experimental Features

- New query processing pipeline. Use `experimental_use_processors=1` option to enable it. Use for your own trouble. [#4914](#) ([Nikolai Kochetov](#))

### Bug Fix

- Kafka integration has been fixed in this version.

- Fixed DoubleDelta encoding of Int64 for large DoubleDelta values, improved DoubleDelta encoding for random data for Int32. [#5998](#) ([Vasily Nemkov](#))
- Fixed overestimation of max\_rows\_to\_read if the setting merge\_tree\_uniform\_read\_distribution is set to 0. [#6019](#) ([alexey-milovidov](#))

## Improvement

- Throws an exception if config.d file does not have the corresponding root element as the config file [#6123](#) ([dimarub2000](#))

## Performance Improvement

- Optimize count(). Now it uses the smallest column (if possible). [#6028](#) ([Amos Bird](#))

## Build/Testing/Packaging Improvement

- Report memory usage in performance tests. [#5899](#) ([akuzm](#))
- Fix build with external libcxx [#6010](#) ([Ivan](#))
- Fix shared build with rdkafka library [#6101](#) ([Ivan](#))

# ClickHouse Release 19.11

## ClickHouse Release 19.11.13.74, 2019-11-01

### Bug Fix

- Fixed rare crash in ALTER MODIFY COLUMN and vertical merge when one of merged/altered parts is empty (0 rows). [#6780](#) ([alesapin](#))
- Manual update of SIMDJSON. This fixes possible flooding of stderr files with bogus json diagnostic messages. [#7548](#) ([Alexander Kazakov](#))
- Fixed bug with mrk file extension for mutations ([alesapin](#))

## ClickHouse Release 19.11.12.69, 2019-10-02

### Bug Fix

- Fixed performance degradation of index analysis on complex keys on large tables. This fixes [#6924](#). [#7075](#) ([alexey-milovidov](#))
- Avoid rare SIGSEGV while sending data in tables with Distributed engine (Failed to send batch: file with index XXXXX is absent). [#7032](#) ([Azat Khuzhin](#))
- Fix Unknown identifier with multiple joins. This fixes [#5254](#). [#7022](#) ([Artem Zuikov](#))

## ClickHouse Release 19.11.11.57, 2019-09-13

- Fix logical error causing segfaults when selecting from Kafka empty topic. [#6902](#) [#6909](#) ([Ivan](#))
- Fix for function ArrayEnumerateUniqRanked with empty arrays in params. [#6928](#) ([proller](#))

## ClickHouse Release 19.11.10.54, 2019-09-10

### Bug Fix

- Do store offsets for Kafka messages manually to be able to commit them all at once for all partitions. Fixes potential duplication in “one consumer - many partitions” scenario. [#6872](#) ([Ivan](#))

## ClickHouse Release 19.11.9.52, 2019-09-06

- Improve error handling in cache dictionaries. #6737 (Vitaly Baranov)
- Fixed bug in function `arrayEnumerateUniqRanked`. #6779 (proller)
- Fix `JSONExtract` function while extracting a `Tuple` from JSON. #6718 (Vitaly Baranov)
- Fixed possible data loss after `ALTER DELETE` query on table with skipping index. #6224 #6282 (Nikita Vasilev)
- Fixed performance test. #6392 (alexey-milovidov)
- Parquet: Fix reading boolean columns. #6579 (alexey-milovidov)
- Fixed wrong behaviour of `nullIf` function for constant arguments. #6518 (Guillaume Tassery) #6580 (alexey-milovidov)
- Fix Kafka messages duplication problem on normal server restart. #6597 (Ivan)
- Fixed an issue when long `ALTER UPDATE` or `ALTER DELETE` may prevent regular merges to run. Prevent mutations from executing if there is no enough free threads available. #6502 #6617 (tavplubix)
- Fixed error with processing “timezone” in server configuration file. #6709 (alexey-milovidov)
- Fix kafka tests. #6805 (Ivan)

## Security Fix

- If the attacker has write access to ZooKeeper and is able to run custom server available from the network where ClickHouse runs, it can create custom-built malicious server that will act as ClickHouse replica and register it in ZooKeeper. When another replica will fetch data part from malicious replica, it can force clickhouse-server to write to arbitrary path on filesystem. Found by Eldar Zaitov, information security team at Yandex. #6247 (alexey-milovidov)

## ClickHouse Release 19.11.8.46, 2019-08-22

### Bug Fix

- Fix `ALTER TABLE ... UPDATE` query for tables with `enable_mixed_granularity_parts=1`. #6543 (alesapin)
- Fix NPE when using `IN` clause with a subquery with a tuple. #6125 #6550 (tavplubix)
- Fixed an issue that if a stale replica becomes alive, it may still have data parts that were removed by `DROP PARTITION`. #6522 #6523 (tavplubix)
- Fixed issue with parsing CSV #6426 #6559 (tavplubix)
- Fixed data race in `system.parts` table and `ALTER` query. This fixes #6245. #6513 (alexey-milovidov)
- Fixed wrong code in mutations that may lead to memory corruption. Fixed segfault with read of address `0x14c0` that may happen due to concurrent `DROP TABLE` and `SELECT` from `system.parts` or `system.parts_columns`. Fixed race condition in preparation of mutation queries. Fixed deadlock caused by `OPTIMIZE` of Replicated tables and concurrent modification operations like `ALTERs`. #6514 (alexey-milovidov)

## ClickHouse Release 19.11.7.40, 2019-08-14

### Bug Fix

- Kafka integration has been fixed in this version.
- Fix segfault when using `arrayReduce` for constant arguments. #6326 (alexey-milovidov)
- Fixed `toFloat()` monotonicity. #6374 (dimarub2000)

- Fix segfault with enabled `optimize_skip_unused_shards` and missing sharding key. #6384 (Curtiz)
- Fixed logic of `arrayEnumerateUniqRanked` function. #6423 (alexey-milovidov)
- Removed extra verbose logging from MySQL handler. #6389 (alexey-milovidov)
- Fix wrong behavior and possible segfaults in `topK` and `topKWeighted` aggregated functions. #6404 (Curtiz)
- Do not expose virtual columns in `system.columns` table. This is required for backward compatibility. #6406 (alexey-milovidov)
- Fix bug with memory allocation for string fields in complex key cache dictionary. #6447 (alesapin)
- Fix bug with enabling adaptive granularity when creating new replica for `Replicated*MergeTree` table. #6452 (alesapin)
- Fix infinite loop when reading Kafka messages. #6354 (abyss7)
- Fixed the possibility of a fabricated query to cause server crash due to stack overflow in SQL parser and possibility of stack overflow in `Merge` and `Distributed` tables #6433 (alexey-milovidov)
- Fixed Gorilla encoding error on small sequences. #6444 (Enmk)

## Improvement

- Allow user to override `poll_interval` and `idle_connection_timeout` settings on connection. #6230 (alexey-milovidov)

# ClickHouse Release 19.11.5.28, 2019-08-05

## Bug Fix

- Fixed the possibility of hanging queries when server is overloaded. #6301 (alexey-milovidov)
- Fix FPE in `yandexConsistentHash` function. This fixes #6304. #6126 (alexey-milovidov)
- Fixed bug in conversion of `LowCardinality` types in `AggregateFunctionFactory`. This fixes #6257. #6281 (Nikolai Kochetov)
- Fix parsing of `bool` settings from `true` and `false` strings in configuration files. #6278 (alesapin)
- Fix rare bug with incompatible stream headers in queries to `Distributed` table over `MergeTree` table when part of `WHERE` moves to `PREWHERE`. #6236 (alesapin)
- Fixed overflow in integer division of signed type to unsigned type. This fixes #6214. #6233 (alexey-milovidov)

## Backward Incompatible Change

- Kafka still broken.

# ClickHouse Release 19.11.4.24, 2019-08-01

## Bug Fix

- Fix bug with writing secondary indices marks with adaptive granularity. #6126 (alesapin)
- Fix `WITH ROLLUP` and `WITH CUBE` modifiers of `GROUP BY` with two-level aggregation. #6225 (Anton Popov)
- Fixed hang in `JSONExtractRaw` function. Fixed #6195 #6198 (alexey-milovidov)
- Fix segfault in `ExternalLoader::reloadOutdated()`. #6082 (Vitaly Baranov)

- Fixed the case when server may close listening sockets but not shutdown and continue serving remaining queries. You may end up with two running clickhouse-server processes. Sometimes, the server may return an error `bad_function_call` for remaining queries. [#6231 \(alexey-milovidov\)](#)
- Fixed useless and incorrect condition on update field for initial loading of external dictionaries via ODBC, MySQL, ClickHouse and HTTP. This fixes [#6069 #6083 \(alexey-milovidov\)](#)
- Fixed irrelevant exception in cast of `LowCardinality(Nullable)` to not-`Nullable` column in case if it does not contain Nulls (e.g. in query like `SELECT CAST(CAST('Hello' AS LowCardinality(Nullable(String))) AS String)`). [#6094 #6119 \(Nikolai Kochetov\)](#)
- Fix non-deterministic result of “`uniq`” aggregate function in extreme rare cases. The bug was present in all ClickHouse versions. [#6058 \(alexey-milovidov\)](#)
- Segfault when we set a little bit too high CIDR on the function `IPv6CIDRToRange`. [#6068 \(Guillaume Tassery\)](#)
- Fixed small memory leak when server throw many exceptions from many different contexts. [#6144 \(alexey-milovidov\)](#)
- Fix the situation when consumer got paused before subscription and not resumed afterwards. [#6075 \(Ivan\)](#) Note that Kafka is broken in this version.
- Clearing the Kafka data buffer from the previous read operation that was completed with an error [#6026 \(Nikolay\)](#) Note that Kafka is broken in this version.
- Since `StorageMergeTree::background_task_handle` is initialized in `startup()` the `MergeTreeBlockOutputStream::write()` may try to use it before initialization. Just check if it is initialized. [#6080 \(Ivan\)](#)

## Build/Testing/Packaging Improvement

- Added official `rpm` packages. [#5740 \(proller\) \(alesapin\)](#)
- Add an ability to build `.rpm` and `.tgz` packages with `packager` script. [#5769 \(alesapin\)](#)
- Fixes for “Arcadia” build system. [#6223 \(proller\)](#)

## Backward Incompatible Change

- Kafka is broken in this version.

# ClickHouse Release 19.11.3.11, 2019-07-18

## New Feature

- Added support for prepared statements. [#5331 \(Alexander\) #5630 \(alexey-milovidov\)](#)
- `DoubleDelta` and `Gorilla` column codecs [#5600 \(Vasily Nemkov\)](#)
- Added `os_thread_priority` setting that allows to control the “nice” value of query processing threads that is used by OS to adjust dynamic scheduling priority. It requires `CAP_SYS_NICE` capabilities to work. This implements [#5858 #5909 \(alexey-milovidov\)](#)
- Implement `_topic`, `_offset`, `_key` columns for Kafka engine [#5382 \(Ivan\)](#) Note that Kafka is broken in this version.
- Add aggregate function combinator `-Resample` [#5590 \(hc2\)](#)
- Aggregate functions `groupArrayMovingSum(win_size)(x)` and `groupArrayMovingAvg(win_size)(x)`, which calculate moving sum/avg with or without window-size limitation. [#5595 \(inv2004\)](#)

- Add synonym `arrayFlatten \<-> flatten` #5764 (hcz)
- Integrate H3 function `geoToH3` from Uber. #4724 (Remen Ivan) #5805 (alexey-milovidov)

## Bug Fix

- Implement DNS cache with asynchronous update. Separate thread resolves all hosts and updates DNS cache with period (setting `dns_cache_update_period`). It should help, when ip of hosts changes frequently. #5857 (Anton Popov)
- Fix segfault in `Delta` codec which affects columns with values less than 32 bits size. The bug led to random memory corruption. #5786 (alesapin)
- Fix segfault in TTL merge with non-physical columns in block. #5819 (Anton Popov)
- Fix rare bug in checking of part with `LowCardinality` column. Previously `checkDataPart` always fails for part with `LowCardinality` column. #5832 (alesapin)
- Avoid hanging connections when server thread pool is full. It is important for connections from `remote` table function or connections to a shard without replicas when there is long connection timeout. This fixes #5878 #5881 (alexey-milovidov)
- Support for constant arguments to `evalMLModel` function. This fixes #5817 #5820 (alexey-milovidov)
- Fixed the issue when ClickHouse determines default time zone as `UCT` instead of `UTC`. This fixes #5804. #5828 (alexey-milovidov)
- Fixed buffer underflow in `visitParamExtractRaw`. This fixes #5901 #5902 (alexey-milovidov)
- Now distributed `DROP/ALTER/TRUNCATE/OPTIMIZE ON CLUSTER` queries will be executed directly on leader replica. #5757 (alesapin)
- Fix `coalesce` for `ColumnConst` with `ColumnNullable` + related changes. #5755 (Artem Zuikov)
- Fix the `ReadBufferFromKafkaConsumer` so that it keeps reading new messages after `commit()` even if it was stalled before #5852 (Ivan)
- Fix `FULL` and `RIGHT JOIN` results when joining on `Nullable` keys in right table. #5859 (Artem Zuikov)
- Possible fix of infinite sleeping of low-priority queries. #5842 (alexey-milovidov)
- Fix race condition, which cause that some queries may not appear in `query_log` after `SYSTEM FLUSH LOGS` query. #5456 #5685 (Anton Popov)
- Fixed `heap-use-after-free` ASan warning in `ClusterCopier` caused by watch which try to use already removed copier object. #5871 (Nikolai Kochetov)
- Fixed wrong `StringRef` pointer returned by some implementations of `IColumn::deserializeAndInsertFromArena`. This bug affected only unit-tests. #5973 (Nikolai Kochetov)
- Prevent source and intermediate array join columns of masking same name columns. #5941 (Artem Zuikov)
- Fix insert and select query to MySQL engine with MySQL style identifier quoting. #5704 (Winter Zhang)
- Now `CHECK TABLE` query can work with MergeTree engine family. It returns check status and message if any for each part (or file in case of simpler engines). Also, fix bug in fetch of a broken part. #5865 (alesapin)
- Fix `SPLIT_SHARED_LIBRARIES` runtime #5793 (Danila Kutenin)

- Fixed time zone initialization when `/etc/localtime` is a relative symlink like `../usr/share/zoneinfo/Europe/Moscow` [#5922 \(alexey-milovidov\)](#)
- clickhouse-copier: Fix use-after free on shutdown [#5752 \(proller\)](#)
- Updated `simdjson`. Fixed the issue that some invalid JSONs with zero bytes successfully parse. [#5938 \(alexey-milovidov\)](#)
- Fix shutdown of SystemLogs [#5802 \(Anton Popov\)](#)
- Fix hanging when condition in `invalidate_query` depends on a dictionary. [#6011 \(Vitaly Baranov\)](#)

## Improvement

- Allow unresolvable addresses in cluster configuration. They will be considered unavailable and tried to resolve at every connection attempt. This is especially useful for Kubernetes. This fixes [#5714](#) [#5924 \(alexey-milovidov\)](#)
- Close idle TCP connections (with one hour timeout by default). This is especially important for large clusters with multiple distributed tables on every server, because every server can possibly keep a connection pool to every other server, and after peak query concurrency, connections will stall. This fixes [#5879](#) [#5880 \(alexey-milovidov\)](#)
- Better quality of `topK` function. Changed the `SavingSpace` set behavior to remove the last element if the new element have a bigger weight. [#5833](#) [#5850 \(Guillaume Tassery\)](#)
- URL functions to work with domains now can work for incomplete URLs without scheme [#5725 \(alesapin\)](#)
- Checksums added to the `system.parts_columns` table. [#5874 \(Nikita Mikhaylov\)](#)
- Added `Enum` data type as a synonym for `Enum8` or `Enum16`. [#5886 \(dimarub2000\)](#)
- Full bit transpose variant for `T64` codec. Could lead to better compression with `zstd`. [#5742 \(Artem Zuikov\)](#)
- Condition on `startsWith` function now can uses primary key. This fixes [#5310](#) and [#5882](#) [#5919 \(dimarub2000\)](#)
- Allow to use `clickhouse-copier` with cross-replication cluster topology by permitting empty database name. [#5745 \(nvartolomei\)](#)
- Use `UTC` as default timezone on a system without `tzdata` (e.g. bare Docker container). Before this patch, error message `Could not determine local time zone` was printed and server or client refused to start. [#5827 \(alexey-milovidov\)](#)
- Returned back support for floating point argument in function `quantileTiming` for backward compatibility. [#5911 \(alexey-milovidov\)](#)
- Show which table is missing column in error messages. [#5768 \(Ivan\)](#)
- Disallow run query with same `query_id` by various users [#5430 \(proller\)](#)
- More robust code for sending metrics to Graphite. It will work even during long multiple `RENAME TABLE` operation. [#5875 \(alexey-milovidov\)](#)
- More informative error messages will be displayed when ThreadPool cannot schedule a task for execution. This fixes [#5305](#) [#5801 \(alexey-milovidov\)](#)
- Inverting `ngramSearch` to be more intuitive [#5807 \(Danila Kutenin\)](#)
- Add user parsing in HDFS engine builder [#5946 \(akonyaev90\)](#)

- Update default value of `max_ast_elements` parameter [#5933](#) ([Artem Konovalov](#))
- Added a notion of obsolete settings. The obsolete setting `allow_experimental_low_cardinality_type` can be used with no effect. [0f15c01c6802f7ce1a1494c12c846be8c98944cd](#) [Alexey Milovidov](#)

## Performance Improvement

- Increase number of streams to SELECT from Merge table for more uniform distribution of threads. Added setting `max_streams_multiplier_for_merge_tables`. This fixes [#5797](#) [#5915](#) ([alexey-milovidov](#))

## Build/Testing/Packaging Improvement

- Add a backward compatibility test for client-server interaction with different versions of clickhouse. [#5868](#) ([alesapin](#))
- Test coverage information in every commit and pull request. [#5896](#) ([alesapin](#))
- Cooperate with address sanitizer to support our custom allocators (`Arena` and `ArenaWithFreeLists`) for better debugging of “use-after-free” errors. [#5728](#) ([akuzm](#))
- Switch to [LLVM libunwind implementation](#) for C++ exception handling and for stack traces printing [#4828](#) ([Nikita Lapkov](#))
- Add two more warnings from -Weverything [#5923](#) ([alexey-milovidov](#))
- Allow to build ClickHouse with Memory Sanitizer. [#3949](#) ([alexey-milovidov](#))
- Fixed ubsan report about `bitTest` function in fuzz test. [#5943](#) ([alexey-milovidov](#))
- Docker: added possibility to init a ClickHouse instance which requires authentication. [#5727](#) ([Korviakov Andrey](#))
- Update librdkafka to version 1.1.0 [#5872](#) ([Ivan](#))
- Add global timeout for integration tests and disable some of them in tests code. [#5741](#) ([alesapin](#))
- Fix some ThreadSanitizer failures. [#5854](#) ([akuzm](#))
- The `--no-undefined` option forces the linker to check all external names for existence while linking. It's very useful to track real dependencies between libraries in the split build mode. [#5855](#) ([Ivan](#))
- Added performance test for [#5797](#) [#5914](#) ([alexey-milovidov](#))
- Fixed compatibility with gcc-7. [#5840](#) ([alexey-milovidov](#))
- Added support for gcc-9. This fixes [#5717](#) [#5774](#) ([alexey-milovidov](#))
- Fixed error when libunwind can be linked incorrectly. [#5948](#) ([alexey-milovidov](#))
- Fixed a few warnings found by PVS-Studio. [#5921](#) ([alexey-milovidov](#))
- Added initial support for `clang-tidy` static analyzer. [#5806](#) ([alexey-milovidov](#))
- Convert BSD/Linux endian macros( ‘be64toh’ and ‘htobe64’) to the Mac OS X equivalents [#5785](#) ([Fu Chen](#))
- Improved integration tests guide. [#5796](#) ([Vladimir Chebotarev](#))
- Fixing build at macosx + gcc9 [#5822](#) ([filimonov](#))
- Fix a hard-to-spot typo: `aggreAGte` -> aggregate. [#5753](#) ([akuzm](#))
- Fix freebsd build [#5760](#) ([proller](#))

- Add link to experimental YouTube channel to website #5845 (Ivan Blinkov)
- CMake: add option for coverage flags: WITH\_COVERAGE #5776 (proller)
- Fix initial size of some inline PODArray's. #5787 (akuzm)
- clickhouse-server.postinst: fix os detection for centos 6 #5788 (proller)
- Added Arch linux package generation. #5719 (Vladimir Chebotarev)
- Split Common/config.h by libs (dbms) #5715 (proller)
- Fixes for “Arcadia” build platform #5795 (proller)
- Fixes for unconventional build (gcc9, no submodules) #5792 (proller)
- Require explicit type in unalignedStore because it was proven to be bug-prone #5791 (akuzm)
- Fixes MacOS build #5830 (filimonov)
- Performance test concerning the new JIT feature with bigger dataset, as requested here #5263 #5887 (Guillaume Tassery)
- Run stateful tests in stress test 12693e568722f11e19859742f56428455501fd2a (alesapin)

## Backward Incompatible Change

- Kafka is broken in this version.
- Enable adaptive\_index\_granularity = 10MB by default for new MergeTree tables. If you created new MergeTree tables on version 19.11+, downgrade to versions prior to 19.6 will be impossible. #5628 (alesapin)
- Removed obsolete undocumented embedded dictionaries that were used by Yandex.Metrica. The functions OSIn, SEIn, OSToRoot, SEToRoot, OSHierarchy, SEHierarchy are no longer available. If you are using these functions, write email to [clickhouse-feedback@yandex-team.com](mailto:clickhouse-feedback@yandex-team.com). Note: at the last moment we decided to keep these functions for a while. #5780 (alexey-milovidov)

# ClickHouse Release 19.10

## ClickHouse Release 19.10.1.5, 2019-07-12

### New Feature

- Add new column codec: T64. Made for (U)IntX/EnumX/Data(Time)/DecimalX columns. It should be good for columns with constant or small range values. Codec itself allows enlarge or shrink data type without re-compression. #5557 (Artem Zuikov)
- Add database engine MySQL that allow to view all the tables in remote MySQL server #5599 (Winter Zhang)
- bitmapContains implementation. It's 2x faster than bitmapHasAny if the second bitmap contains one element. #5535 (Zhichang Yu)
- Support for crc32 function (with behaviour exactly as in MySQL or PHP). Do not use it if you need a hash function. #5661 (Remen Ivan)
- Implemented SYSTEM START/STOP DISTRIBUTED SENDS queries to control asynchronous inserts into Distributed tables. #4935 (Winter Zhang)

### Bug Fix

- Ignore query execution limits and max parts size for merge limits while executing mutations. #5659 (Anton Popov)
- Fix bug which may lead to deduplication of normal blocks (extremely rare) and insertion of duplicate blocks (more often). #5549 (alesapin)
- Fix of function `arrayEnumerateUniqRanked` for arguments with empty arrays #5559 (proller)
- Don't subscribe to Kafka topics without intent to poll any messages. #5698 (Ivan)
- Make setting `join_use_nulls` get no effect for types that cannot be inside Nullable #5700 (Olga Khvostikova)
- Fixed `Incorrect size of index granularity` errors #5720 (coraxster)
- Fix Float to Decimal convert overflow #5607 (coraxster)
- Flush buffer when `WriteBufferFromHDFS`'s destructor is called. This fixes writing into HDFS. #5684 (Xindong Peng)

## Improvement

- Treat empty cells in `csv` as default values when the setting `input_format_defaults_for_omitted_fields` is enabled. #5625 (akuzm)
- Non-blocking loading of external dictionaries. #5567 (Vitaly Baranov)
- Network timeouts can be dynamically changed for already established connections according to the settings. #4558 (Konstantin Podshumok)
- Using “`public_suffix_list`” for functions `firstSignificantSubdomain`, `cutToFirstSignificantSubdomain`. It's using a perfect hash table generated by `gperf` with a list generated from the file: [https://publicsuffix.org/list/public\\_suffix\\_list.dat](https://publicsuffix.org/list/public_suffix_list.dat). (for example, now we recognize the domain ac.uk as non-significant). #5030 (Guillaume Tassery)
- Adopted `IPv6` data type in system tables; unified client info columns in `system.processes` and `system.query_log` #5640 (alexey-milovidov)
- Using sessions for connections with MySQL compatibility protocol. #5476 #5646 (Yuriy Baranov)
- Support more `ALTER` queries `ON CLUSTER`. #5593 #5613 (sundyli)
- Support `<logger>` section in `clickhouse-local` config file. #5540 (proller)
- Allow run query with `remote` table function in `clickhouse-local` #5627 (proller)

## Performance Improvement

- Add the possibility to write the final mark at the end of MergeTree columns. It allows to avoid useless reads for keys that are out of table data range. It is enabled only if adaptive index granularity is in use. #5624 (alesapin)
- Improved performance of MergeTree tables on very slow filesystems by reducing number of `stat` syscalls. #5648 (alexey-milovidov)
- Fixed performance degradation in reading from MergeTree tables that was introduced in version 19.6. Fixes #5631. #5633 (alexey-milovidov)

## Build/Testing/Packaging Improvement

- Implemented `TestKeeper` as an implementation of ZooKeeper interface used for testing #5643 (alexey-milovidov) (levushkin aleksej)

- From now on `.sql` tests can be run isolated by server, in parallel, with random database. It allows to run them faster, add new tests with custom server configurations, and be sure that different tests does not affect each other. #5554 (Ivan)
- Remove `<name>` and `<metrics>` from performance tests #5672 (Olga Khvostikova)
- Fixed “select\_format” performance test for `Pretty` formats #5642 (alexey-milovidov)

## ClickHouse Release 19.9

### ClickHouse Release 19.9.3.31, 2019-07-05

#### Bug Fix

- Fix segfault in Delta codec which affects columns with values less than 32 bits size. The bug led to random memory corruption. #5786 (alesapin)
- Fix rare bug in checking of part with LowCardinality column. #5832 (alesapin)
- Fix segfault in TTL merge with non-physical columns in block. #5819 (Anton Popov)
- Fix potential infinite sleeping of low-priority queries. #5842 (alexey-milovidov)
- Fix how ClickHouse determines default time zone as UCT instead of UTC. #5828 (alexey-milovidov)
- Fix bug about executing distributed DROP/ALTER/TRUNCATE/OPTIMIZE ON CLUSTER queries on follower replica before leader replica. Now they will be executed directly on leader replica. #5757 (alesapin)
- Fix race condition, which cause that some queries may not appear in `query_log` instantly after SYSTEM FLUSH LOGS query. #5685 (Anton Popov)
- Added missing support for constant arguments to `evalMLModel` function. #5820 (alexey-milovidov)

### ClickHouse Release 19.9.2.4, 2019-06-24

#### New Feature

- Print information about frozen parts in `system.parts` table. #5471 (proller)
- Ask client password on `clickhouse-client` start on tty if not set in arguments #5092 (proller)
- Implement `dictGet` and `dictGetOrDefault` functions for Decimal types. #5394 (Artem Zuikov)

#### Improvement

- Debian init: Add service stop timeout #5522 (proller)
- Add setting `forbidden` by default to create table with suspicious types for LowCardinality #5448 (Olga Khvostikova)
- Regression functions return model weights when not used as State in function `evalMLMethod`. #5411 (Quid37)
- Rename and improve regression methods. #5492 (Quid37)
- Clearer interfaces of string searchers. #5586 (Danila Kutenin)

#### Bug Fix

- Fix potential data loss in Kafka #5445 (Ivan)
- Fix potential infinite loop in `PrettySpace` format when called with zero columns #5560 (Olga Khvostikova)

- Fixed UInt32 overflow bug in linear models. Allow eval ML model for non-const model argument. [#5516](#) ([Nikolai Kochetov](#))
- `ALTER TABLE ... DROP INDEX IF EXISTS ...` should not raise an exception if provided index does not exist [#5524](#) ([Gleb Novikov](#))
- Fix segfault with `bitmapHasAny` in scalar subquery [#5528](#) ([Zhichang Yu](#))
- Fixed error when replication connection pool does not retry to resolve host, even when DNS cache was dropped. [#5534](#) ([alesapin](#))
- Fixed `ALTER ... MODIFY TTL` on ReplicatedMergeTree. [#5539](#) ([Anton Popov](#))
- Fix INSERT into Distributed table with MATERIALIZED column [#5429](#) ([Azat Khuzhin](#))
- Fix bad alloc when truncate Join storage [#5437](#) ([TCeason](#))
- In recent versions of package tzdata some of files are symlinks now. The current mechanism for detecting default timezone gets broken and gives wrong names for some timezones. Now at least we force the timezone name to the contents of TZ if provided. [#5443](#) ([Ivan](#))
- Fix some extremely rare cases with MultiVolnitsky searcher when the constant needles in sum are at least 16KB long. The algorithm missed or overwrote the previous results which can lead to the incorrect result of `multiSearchAny`. [#5588](#) ([Danila Kutenin](#))
- Fix the issue when settings for ExternalData requests couldn't use ClickHouse settings. Also, for now, settings `date_time_input_format` and `low_cardinality_allow_in_native_format` cannot be used because of the ambiguity of names (in external data it can be interpreted as table format and in the query it can be a setting). [#5455](#) ([Danila Kutenin](#))
- Fix bug when parts were removed only from FS without dropping them from Zookeeper. [#5520](#) ([alesapin](#))
- Remove debug logging from MySQL protocol [#5478](#) ([alexey-milovidov](#))
- Skip ZNONODE during DDL query processing [#5489](#) ([Azat Khuzhin](#))
- Fix mix `UNION ALL` result column type. There were cases with inconsistent data and column types of resulting columns. [#5503](#) ([Artem Zuikov](#))
- Throw an exception on wrong integers in `dictGetT` functions instead of crash. [#5446](#) ([Artem Zuikov](#))
- Fix wrong `element_count` and `load_factor` for hashed dictionary in `system.dictionaries` table. [#5440](#) ([Azat Khuzhin](#))

## Build/Testing/Packaging Improvement

- Fixed build without Brotli HTTP compression support (`ENABLE_BROTLI=OFF` cmake variable). [#5521](#) ([Anton Yuzhaninov](#))
- Include roaring.h as roaring/roaring.h [#5523](#) ([Orivej Desh](#))
- Fix gcc9 warnings in hyperscan (#line directive is evil!) [#5546](#) ([Danila Kutenin](#))
- Fix all warnings when compiling with gcc-9. Fix some contrib issues. Fix gcc9 ICE and submit it to bugzilla. [#5498](#) ([Danila Kutenin](#))
- Fixed linking with lld [#5477](#) ([alexey-milovidov](#))
- Remove unused specializations in dictionaries [#5452](#) ([Artem Zuikov](#))

- Improvement performance tests for formatting and parsing tables for different types of files [#5497](#) ([Olga Khvostikova](#))
- Fixes for parallel test run [#5506](#) ([proller](#))
- Docker: use configs from clickhouse-test [#5531](#) ([proller](#))
- Fix compile for FreeBSD [#5447](#) ([proller](#))
- Upgrade boost to 1.70 [#5570](#) ([proller](#))
- Fix build clickhouse as submodule [#5574](#) ([proller](#))
- Improve JSONExtract performance tests [#5444](#) ([Vitaly Baranov](#))

## ClickHouse Release 19.8

### ClickHouse Release 19.8.3.8, 2019-06-11

#### New Features

- Added functions to work with JSON [#4686](#) ([hczi](#)) [#5124](#). ([Vitaly Baranov](#))
- Add a function basename, with a similar behaviour to a basename function, which exists in a lot of languages (`os.path.basename` in python, `basename` in PHP, etc...). Work with both an UNIX-like path or a Windows path. [#5136](#) ([Guillaume Tassery](#))
- Added `LIMIT n, m BY` or `LIMIT m OFFSET n BY` syntax to set offset of n for LIMIT BY clause. [#5138](#) ([Anton Popov](#))
- Added new data type `SimpleAggregateFunction`, which allows to have columns with light aggregation in an `AggregatingMergeTree`. This can only be used with simple functions like `any`, `anyLast`, `sum`, `min`, `max`. [#4629](#) ([Boris Granveaud](#))
- Added support for non-constant arguments in function `ngramDistance` [#5198](#) ([Danila Kutenin](#))
- Added functions `skewPop`, `skewSamp`, `kurtPop` and `kurtSamp` to compute for sequence skewness, sample skewness, kurtosis and sample kurtosis respectively. [#5200](#) ([hczi](#))
- Support rename operation for `MaterializeView` storage. [#5209](#) ([Guillaume Tassery](#))
- Added server which allows connecting to ClickHouse using MySQL client. [#4715](#) ([Yuriy Baranov](#))
- Add `toDecimal*OrZero` and `toDecimal*OrNull` functions. [#5291](#) ([Artem Zuikov](#))
- Support Decimal types in functions: `quantile`, `quantiles`, `median`, `quantileExactWeighted`, `quantilesExactWeighted`, `medianExactWeighted`. [#5304](#) ([Artem Zuikov](#))
- Added `toValidUTF8` function, which replaces all invalid UTF-8 characters by replacement character ♦ (U+FFFD). [#5322](#) ([Danila Kutenin](#))
- Added `format` function. Formatting constant pattern (simplified Python format pattern) with the strings listed in the arguments. [#5330](#) ([Danila Kutenin](#))
- Added `system.detached_parts` table containing information about detached parts of `MergeTree` tables. [#5353](#) ([akuzm](#))
- Added `ngramSearch` function to calculate the non-symmetric difference between needle and haystack. [#5418](#)[#5422](#) ([Danila Kutenin](#))

- Implementation of basic machine learning methods (stochastic linear regression and logistic regression) using aggregate functions interface. Has different strategies for updating model weights (simple gradient descent, momentum method, Nesterov method). Also supports mini-batches of custom size. [#4943 \(Quid37\)](#)
- Implementation of `geohashEncode` and `geohashDecode` functions. [#5003 \(Vasily Nemkov\)](#)
- Added aggregate function `timeSeriesGroupSum`, which can aggregate different time series that sample timestamp not alignment. It will use linear interpolation between two sample timestamp and then sum time-series together. Added aggregate function `timeSeriesGroupRateSum`, which calculates the rate of time-series and then sum rates together. [#4542 \(Yangkuan Liu\)](#)
- Added functions `IPv4CIDRtoIPv4Range` and `IPv6CIDRtoIPv6Range` to calculate the lower and higher bounds for an IP in the subnet using a CIDR. [#5095 \(Guillaume Tassery\)](#)
- Add a X-ClickHouse-Summary header when we send a query using HTTP with enabled setting `send_progress_in_http_headers`. Return the usual information of X-ClickHouse-Progress, with additional information like how many rows and bytes were inserted in the query. [#5116 \(Guillaume Tassery\)](#)

## Improvements

- Added `max_parts_in_total` setting for MergeTree family of tables (default: 100 000) that prevents unsafe specification of partition key [#5166. #5171 \(alexey-milovidov\)](#)
- `clickhouse-obfuscator`: derive seed for individual columns by combining initial seed with column name, not column position. This is intended to transform datasets with multiple related tables, so that tables will remain JOINable after transformation. [#5178 \(alexey-milovidov\)](#)
- Added functions `JSONExtractRaw`, `JSONExtractKeyAndValues`. Renamed functions `jsonExtract<type>` to `JSONExtract<type>`. When something goes wrong these functions return the correspondent values, not `NULL`. Modified function `JSONExtract`, now it gets the return type from its last parameter and does not inject nullables. Implemented fallback to RapidJSON in case AVX2 instructions are not available. Simdjson library updated to a new version. [#5235 \(Vitaly Baranov\)](#)
- Now `if` and `multilf` functions do not rely on the condition's `Nullable`, but rely on the branches for sql compatibility. [#5238 \(Jian Wu\)](#)
- `In` predicate now generates `Null` result from `Null` input like the `Equal` function. [#5152 \(Jian Wu\)](#)
- Check the time limit every (`flush_interval` / `poll_timeout`) number of rows from Kafka. This allows to break the reading from Kafka consumer more frequently and to check the time limits for the top-level streams [#5249 \(Ivan\)](#)
- Link rdkafka with bundled SASL. It should allow to use SASL SCRAM authentication [#5253 \(Ivan\)](#)
- Batched version of RowRefList for ALL JOINS. [#5267 \(Artem Zuikov\)](#)
- `clickhouse-server`: more informative listen error messages. [#5268 \(proller\)](#)
- Support dictionaries in `clickhouse-copier` for functions in `<sharding_key>` [#5270 \(proller\)](#)
- Add new setting `kafka_commit_every_batch` to regulate Kafka committing policy. It allows to set commit mode: after every batch of messages is handled, or after the whole block is written to the storage. It's a trade-off between losing some messages or reading them twice in some extreme situations. [#5308 \(Ivan\)](#)
- Make `windowFunnel` support other Unsigned Integer Types. [#5320 \(sundyli\)](#)
- Allow to shadow virtual column `_table` in Merge engine. [#5325 \(Ivan\)](#)
- Make `sequenceMatch` aggregate functions support other unsigned Integer types [#5339 \(sundyli\)](#)

- Better error messages if checksum mismatch is most likely caused by hardware failures. #5355 (alexey-milovidov)
- Check that underlying tables support sampling for StorageMerge #5366 (Ivan)
- Close MySQL connections after their usage in external dictionaries. It is related to issue #893. #5395 (Clément Rodriguez)
- Improvements of MySQL Wire Protocol. Changed name of format to MySQLWire. Using RAI<sup>I</sup> for calling RSA\_free. Disabling SSL if context cannot be created. #5419 (Yuriy Baranov)
- clickhouse-client: allow to run with unaccessible history file (read-only, no disk space, file is directory, ...). #5431 (proller)
- Respect query settings in asynchronous INSERTs into Distributed tables. #4936 (TCeason)
- Renamed functions leastSqr to simpleLinearRegression, LinearRegression to linearRegression, LogisticRegression to logisticRegression. #5391 (Nikolai Kochetov)

## Performance Improvements

- Parallelize processing of parts of non-replicated MergeTree tables in ALTER MODIFY query. #4639 (Ivan Kush)
- Optimizations in regular expressions extraction. #5193 #5191 (Danila Kutenin)
- Do not add right join key column to join result if it's used only in join on section. #5260 (Artem Zuikov)
- Freeze the Kafka buffer after first empty response. It avoids multiple invocations of ReadBuffer::next() for empty result in some row-parsing streams. #5283 (Ivan)
- concat function optimization for multiple arguments. #5357 (Danila Kutenin)
- Query optimisation. Allow push down IN statement while rewriting comma/cross join into inner one. #5396 (Artem Zuikov)
- Upgrade our LZ4 implementation with reference one to have faster decompression. #5070 (Danila Kutenin)
- Implemented MSD radix sort (based on kxsort), and partial sorting. #5129 (Evgenii Pravda)

## Bug Fixes

- Fix push require columns with join #5192 (Winter Zhang)
- Fixed bug, when ClickHouse is run by systemd, the command sudo service clickhouse-server forcerestart was not working as expected. #5204 (proller)
- Fix http error codes in DataPartsExchange (interserver http server on 9009 port always returned code 200, even on errors). #5216 (proller)
- Fix SimpleAggregateFunction for String longer than MAX\_SMALL\_STRING\_SIZE #5311 (Azat Khuzhin)
- Fix error for Decimal to Nullable(Decimal) conversion in IN. Support other Decimal to Decimal conversions (including different scales). #5350 (Artem Zuikov)
- Fixed FPU clobbering in simdjson library that lead to wrong calculation of uniqHLL and uniqCombined aggregate function and math functions such as log. #5354 (alexey-milovidov)
- Fixed handling mixed const/nonconst cases in JSON functions. #5435 (Vitaly Baranov)

- Fix retention function. Now all conditions that satisfy in a row of data are added to the data state. #5119 (小路)
- Fix result type for quantileExact with Decimals. #5304 (Artem Zuikov)

## Documentation

- Translate documentation for CollapsingMergeTree to chinese. #5168 (张风啸)
- Translate some documentation about table engines to chinese.  
#5134  
#5328  
(never lee)

## Build/Testing/Packaging Improvements

- Fix some sanitizer reports that show probable use-after-free. #5139 #5143 #5393 (Ivan)
- Move performance tests out of separate directories for convenience. #5158 (alexey-milovidov)
- Fix incorrect performance tests. #5255 (alesapin)
- Added a tool to calculate checksums caused by bit flips to debug hardware issues. #5334 (alexey-milovidov)
- Make runner script more usable. #5340#5360 (filimonov)
- Add small instruction how to write performance tests. #5408 (alesapin)
- Add ability to make substitutions in create, fill and drop query in performance tests #5367 (Olga Khvostikova)

# ClickHouse Release 19.7

## ClickHouse Release 19.7.5.29, 2019-07-05

### Bug Fix

- Fix performance regression in some queries with JOIN. #5192 (Winter Zhang)

## ClickHouse Release 19.7.5.27, 2019-06-09

### New Features

- Added bitmap related functions bitmapHasAny and bitmapHasAll analogous to hasAny and hasAll functions for arrays. #5279 (Sergi Vladynkin)

### Bug Fixes

- Fix segfault on minmax INDEX with Null value. #5246 (Nikita Vasilev)
- Mark all input columns in LIMIT BY as required output. It fixes ‘Not found column’ error in some distributed queries. #5407 (Constantin S. Pan)
- Fix “Column ‘0’ already exists” error in SELECT .. PREWHERE on column with DEFAULT #5397 (proller)
- Fix ALTER MODIFY TTL query on ReplicatedMergeTree. #5539 (Anton Popov)
- Don’t crash the server when Kafka consumers have failed to start. #5285 (Ivan)
- Fixed bitmap functions produce wrong result. #5359 (Andy Yang)
- Fix element\_count for hashed dictionary (do not include duplicates) #5440 (Azat Khuzhin)

- Use contents of environment variable TZ as the name for timezone. It helps to correctly detect default timezone in some cases. [#5443](#) ([Ivan](#))
- Do not try to convert integers in `dictGetT` functions, because it does not work correctly. Throw an exception instead. [#5446](#) ([Artem Zuikov](#))
- Fix settings in ExternalData HTTP request. [#5455](#) ([Danila Kutenin](#))
- Fix bug when parts were removed only from FS without dropping them from Zookeeper. [#5520](#) ([alesapin](#))
- Fix segmentation fault in `bitmapHasAny` function. [#5528](#) ([Zhichang Yu](#))
- Fixed error when replication connection pool does not retry to resolve host, even when DNS cache was dropped. [#5534](#) ([alesapin](#))
- Fixed `DROP INDEX IF EXISTS` query. Now `ALTER TABLE ... DROP INDEX IF EXISTS ...` query does not raise an exception if provided index does not exist. [#5524](#) ([Gleb Novikov](#))
- Fix union all supertype column. There were cases with inconsistent data and column types of resulting columns. [#5503](#) ([Artem Zuikov](#))
- Skip ZNONODE during DDL query processing. Before if another node removes the znode in task queue, the one that did not process it, but already get list of children, will terminate the DDLWorker thread. [#5489](#) ([Azat Khuzhin](#))
- Fix `INSERT` into `Distributed()` table with `MATERIALIZED` column. [#5429](#) ([Azat Khuzhin](#))

## ClickHouse Release 19.7.3.9, 2019-05-30

### New Features

- Allow to limit the range of a setting that can be specified by user. These constraints can be set up in user settings profile. [#4931](#) ([Vitaly Baranov](#))
- Add a second version of the function `groupUniqArray` with an optional `max_size` parameter that limits the size of the resulting array. This behavior is similar to `groupArray(max_size)(x)` function. [#5026](#) ([Guillaume Tassery](#))
- For `TSVWithNames`/`CSVWithNames` input file formats, column order can now be determined from file header. This is controlled by `input_format_with_names_use_header` parameter. [#5081](#) ([Alexander](#))

### Bug Fixes

- Crash with `uncompressed_cache + JOIN` during merge ([#5197](#))  
[#5133](#) ([Danila Kutenin](#))
- Segmentation fault on a `clickhouse-client` query to system tables. [#5066](#)  
[#5127](#) ([Ivan](#))

- Data loss on heavy load via KafkaEngine (#4736)  
[#5080](#)  
([Ivan](#))
- Fixed very rare data race condition that could happen when executing a query with UNION ALL involving at least two SELECTs from system.columns, system.tables, system.parts, system.parts\_tables or tables of Merge family and performing ALTER of columns of the related tables concurrently. [#5189](#) ([alexey-milovidov](#))

## Performance Improvements

- Use radix sort for sorting by single numeric column in ORDER BY without LIMIT. [#5106](#),  
[#4439](#)  
([Evgenii Pravda](#),  
[alexey-milovidov](#))

## Documentation

- Translate documentation for some table engines to Chinese.  
[#5107](#),  
[#5094](#),  
[#5087](#)  
([张风啸](#)),  
[#5068](#) ([never](#)  
[lee](#))

## Build/Testing/Packaging Improvements

- Print UTF-8 characters properly in `clickhouse-test`.  
[#5084](#)  
([alexey-milovidov](#))
- Add command line parameter for `clickhouse-client` to always load suggestion data. [#5102](#)  
([alexey-milovidov](#))
- Resolve some of PVS-Studio warnings.  
[#5082](#)  
([alexey-milovidov](#))
- Update LZ4 [#5040](#) ([Danila Kutenin](#))
- Add gperf to build requirements for upcoming pull request #5030.  
[#5110](#)  
([proller](#))

# ClickHouse Release 19.6

## ClickHouse Release 19.6.3.18, 2019-06-13

### Bug Fixes

- Fixed IN condition pushdown for queries from table functions `mysql` and `odbc` and corresponding table engines. This fixes #3540 and #2384. [#5313](#) ([alexey-milovidov](#))
- Fix deadlock in Zookeeper. [#5297](#) ([github1youlc](#))
- Allow quoted decimals in CSV. [#5284](#) ([Artem Zuikov](#))

- Disallow conversion from float Inf/NaN into Decimals (throw exception). [#5282](#) ([Artem Zuikov](#))
- Fix data race in rename query. [#5247](#) ([Winter Zhang](#))
- Temporarily disable LFAlloc. Usage of LFAlloc might lead to a lot of MAP\_FAILED in allocating UncompressedCache and in a result to crashes of queries at high loaded servers. [cfdba93](#)([Danila Kutenin](#))

## ClickHouse Release 19.6.2.11, 2019-05-13

### New Features

- TTL expressions for columns and tables. [#4212](#) ([Anton Popov](#))
- Added support for `brotli` compression for HTTP responses (Accept-Encoding: br) [#4388](#) ([Mikhail](#))
- Added new function `isValidUTF8` for checking whether a set of bytes is correctly utf-8 encoded. [#4934](#) ([Danila Kutenin](#))
- Add new load balancing policy `first_or_random` which sends queries to the first specified host and if it's inaccessible send queries to random hosts of shard. Useful for cross-replication topology setups. [#5012](#) ([nvartolomei](#))

### Experimental Features

- Add setting `index_granularity_bytes` (adaptive index granularity) for MergeTree\* tables family. [#4826](#) ([alesapin](#))

### Improvements

- Added support for non-constant and negative size and length arguments for function `substringUTF8`. [#4989](#) ([alexey-milovidov](#))
- Disable push-down to right table in left join, left table in right join, and both tables in full join. This fixes wrong JOIN results in some cases. [#4846](#) ([Ivan](#))
- `clickhouse-copier`: auto upload task configuration from `--task-file` option [#4876](#) ([proller](#))
- Added typos handler for storage factory and table functions factory. [#4891](#) ([Danila Kutenin](#))
- Support asterisks and qualified asterisks for multiple joins without subqueries [#4898](#) ([Artem Zuikov](#))
- Make missing column error message more user friendly. [#4915](#) ([Artem Zuikov](#))

### Performance Improvements

- Significant speedup of ASOF JOIN [#4924](#) ([Martijn Bakker](#))

### Backward Incompatible Changes

- HTTP header `Query-Id` was renamed to `X-ClickHouse-Query-Id` for consistency. [#4972](#) ([Mikhail](#))

### Bug Fixes

- Fixed potential null pointer dereference in `clickhouse-copier`. [#4900](#) ([proller](#))
- Fixed error on query with JOIN + ARRAY JOIN [#4938](#) ([Artem Zuikov](#))
- Fixed hanging on start of the server when a dictionary depends on another dictionary via a database with engine=Dictionary. [#4962](#) ([Vitaly Baranov](#))
- Partially fix distributed\_product\_mode = local. It's possible to allow columns of local tables in where/having/order by/... via table aliases. Throw exception if table does not have alias. There's not possible to access to the columns without table aliases yet. [#4986](#) ([Artem Zuikov](#))

- Fix potentially wrong result for `SELECT DISTINCT` with `JOIN` #5001 (Artem Zuikov)
- Fixed very rare data race condition that could happen when executing a query with `UNION ALL` involving at least two `SELECTs` from `system.columns`, `system.tables`, `system.parts`, `system.parts_tables` or tables of Merge family and performing `ALTER` of columns of the related tables concurrently. #5189 (alexey-milovidov)

## Build/Testing/Packaging Improvements

- Fixed test failures when running `clickhouse-server` on different host #4713 (Vasily Nemkov)
- `clickhouse-test`: Disable color control sequences in non tty environment. #4937 (alesapin)
- `clickhouse-test`: Allow use any test database (remove `test.` qualification where it possible) #5008 (proller)
- Fix ubsan errors #5037 (Vitaly Baranov)
- Yandex LFAlloc was added to ClickHouse to allocate `MarkCache` and `UncompressedCache` data in different ways to catch segfaults more reliable #4995 (Danila Kutenin)
- Python util to help with backports and changelogs. #4949 (Ivan)

# ClickHouse Release 19.5

## ClickHouse Release 19.5.4.22, 2019-05-13

### Bug Fixes

- Fixed possible crash in `bitmap*` functions #5220 #5228 (Andy Yang)
- Fixed very rare data race condition that could happen when executing a query with `UNION ALL` involving at least two `SELECTs` from `system.columns`, `system.tables`, `system.parts`, `system.parts_tables` or tables of Merge family and performing `ALTER` of columns of the related tables concurrently. #5189 (alexey-milovidov)
- Fixed error `Set for IN is not created yet` in case of using single `LowCardinality` column in the left part of `IN`. This error happened if `LowCardinality` column was the part of primary key. #5031 #5154 (Nikolai Kochetov)
- Modification of retention function: If a row satisfies both the first and NTH condition, only the first satisfied condition is added to the data state. Now all conditions that satisfy in a row of data are added to the data state. #5119 (小路)

## ClickHouse Release 19.5.3.8, 2019-04-18

### Bug Fixes

- Fixed type of setting `max_partitions_per_insert_block` from boolean to `UInt64`. #5028 (Mohammad Hossein Sekhavat)

## ClickHouse Release 19.5.2.6, 2019-04-15

### New Features

- `Hyperscan` multiple regular expression matching was added (functions `multiMatchAny`, `multiMatchAnyIndex`, `multiFuzzyMatchAny`, `multiFuzzyMatchAnyIndex`). #4780, #4841 (Danila Kutenin)
- `multiSearchFirstPosition` function was added. #4780 (Danila Kutenin)
- Implement the predefined expression filter per row for tables. #4792 (Ivan)
- A new type of data skipping indices based on bloom filters (can be used for `equal`, `in` and `like` functions). #4499 (Nikita Vasilev)

- Added ASOF JOIN which allows to run queries that join to the most recent value known. #4774 #4867 #4863 #4875 (Martijn Bakker, Artem Zuikov)
- Rewrite multiple COMMA JOIN to CROSS JOIN. Then rewrite them to INNER JOIN if possible. #4661 (Artem Zuikov)

## Improvement

- `topK` and `topKWeighted` now supports custom `loadFactor` (fixes issue #4252). #4634 (Kirill Danshin)
- Allow to use `parallel_replicas_count > 1` even for tables without sampling (the setting is simply ignored for them). In previous versions it was lead to exception. #4637 (Alexey Elymanov)
- Support for `CREATE OR REPLACE VIEW`. Allow to create a view or set a new definition in a single statement. #4654 (Boris Granveaud)
- Buffer table engine now supports `PREWHERE`. #4671 (Yangkuan Liu)
- Add ability to start replicated table without metadata in zookeeper in `readonly` mode. #4691 (alesapin)
- Fixed flicker of progress bar in clickhouse-client. The issue was most noticeable when using `FORMAT Null` with streaming queries. #4811 (alexey-milovidov)
- Allow to disable functions with `hyperscan` library on per user basis to limit potentially excessive and uncontrolled resource usage. #4816 (alexey-milovidov)
- Add version number logging in all errors. #4824 (proller)
- Added restriction to the `multiMatch` functions which requires string size to fit into `unsigned int`. Also added the number of arguments limit to the `multiSearch` functions. #4834 (Danila Kutenin)
- Improved usage of scratch space and error handling in Hyperscan. #4866 (Danila Kutenin)
- Fill `system.graphite_detentions` from a table config of \*GraphiteMergeTree engine tables. #4584 (Mikhail f. Shiryaev)
- Rename `trigramDistance` function to `ngramDistance` and add more functions with `CaseInsensitive` and `UTF`. #4602 (Danila Kutenin)
- Improved data skipping indices calculation. #4640 (Nikita Vasilev)
- Keep ordinary, DEFAULT, MATERIALIZED and ALIAS columns in a single list (fixes issue #2867). #4707 (Alex Zatelepin)

## Bug Fix

- Avoid `std::terminate` in case of memory allocation failure. Now `std::bad_alloc` exception is thrown as expected. #4665 (alexey-milovidov)
- Fixes capnproto reading from buffer. Sometimes files wasn't loaded successfully by HTTP. #4674 (Vladislav)
- Fix error Unknown log entry type: 0 after `OPTIMIZE TABLE FINAL` query. #4683 (Amos Bird)
- Wrong arguments to `hasAny` or `hasAll` functions may lead to segfault. #4698 (alexey-milovidov)
- Deadlock may happen while executing `DROP DATABASE` dictionary query. #4701 (alexey-milovidov)
- Fix undefined behavior in `median` and `quantile` functions. #4702 (hcjz)
- Fix compression level detection when `network_compression_method` in lowercase. Broken in v19.1. #4706 (proller)

- Fixed ignorance of <timezone>UTC</timezone> setting (fixes issue #4658). #4718 (proller)
- Fix histogram function behaviour with Distributed tables. #4741 (olegkv)
- Fixed tsan report destroy of a locked mutex. #4742 (alexey-milovidov)
- Fixed TSan report on shutdown due to race condition in system logs usage. Fixed potential use-after-free on shutdown when part\_log is enabled. #4758 (alexey-milovidov)
- Fix recheck parts in ReplicatedMergeTreeAlterThread in case of error. #4772 (Nikolai Kochetov)
- Arithmetic operations on intermediate aggregate function states were not working for constant arguments (such as subquery results). #4776 (alexey-milovidov)
- Always backquote column names in metadata. Otherwise it's impossible to create a table with column named index (server won't restart due to malformed ATTACH query in metadata). #4782 (alexey-milovidov)
- Fix crash in ALTER ... MODIFY ORDER BY on Distributed table. #4790 (TCeason)
- Fix segfault in JOIN ON with enabled enable\_optimize\_predicate\_expression. #4794 (Winter Zhang)
- Fix bug with adding an extraneous row after consuming a protobuf message from Kafka. #4808 (Vitaly Baranov)
- Fix crash of JOIN on not-nullable vs nullable column. Fix NULLs in right keys in ANY JOIN + join\_use\_nulls. #4815 (Artem Zuikov)
- Fix segmentation fault in clickhouse-copier. #4835 (proller)
- Fixed race condition in SELECT from system.tables if the table is renamed or altered concurrently. #4836 (alexey-milovidov)
- Fixed data race when fetching data part that is already obsolete. #4839 (alexey-milovidov)
- Fixed rare data race that can happen during RENAME table of MergeTree family. #4844 (alexey-milovidov)
- Fixed segmentation fault in function arrayIntersect. Segmentation fault could happen if function was called with mixed constant and ordinary arguments. #4847 (Lixiang Qian)
- Fixed reading from Array(LowCardinality) column in rare case when column contained a long sequence of empty arrays. #4850 (Nikolai Kochetov)
- Fix crash in FULL/RIGHT JOIN when we joining on nullable vs not nullable. #4855 (Artem Zuikov)
- Fix No message received exception while fetching parts between replicas. #4856 (alesapin)
- Fixed arrayIntersect function wrong result in case of several repeated values in single array. #4871 (Nikolai Kochetov)
- Fix a race condition during concurrent ALTER COLUMN queries that could lead to a server crash (fixes issue #3421). #4592 (Alex Zatelepin)
- Fix incorrect result in FULL/RIGHT JOIN with const column. #4723 (Artem Zuikov)
- Fix duplicates in GLOBAL JOIN with asterisk. #4705 (Artem Zuikov)
- Fix parameter deduction in ALTER MODIFY of column CODEC when column type is not specified. #4883 (alesapin)

- Functions `cutQueryStringAndFragment()` and `queryStringAndFragment()` now works correctly when `URL` contains a fragment and no query. [#4894 \(Vitaly Baranov\)](#)
- Fix rare bug when setting `min_bytes_to_use_direct_io` is greater than zero, which occurs when thread have to seek backward in column file. [#4897 \(alesapin\)](#)
- Fix wrong argument types for aggregate functions with `LowCardinality` arguments (fixes issue [#4919](#)). [#4922 \(Nikolai Kochetov\)](#)
- Fix wrong name qualification in `GLOBAL JOIN`. [#4969 \(Artem Zuikov\)](#)
- Fix function `toISOWeek` result for year 1970. [#4988 \(alexey-milovidov\)](#)
- Fix `DROP`, `TRUNCATE` and `OPTIMIZE` queries duplication, when executed on `ON CLUSTER` for `ReplicatedMergeTree*` tables family. [#4991 \(alesapin\)](#)

## Backward Incompatible Change

- Rename setting `insert_sample_with_metadata` to setting `input_format_defaults_for_omitted_fields`. [#4771 \(Artem Zuikov\)](#)
- Added setting `max_partitions_per_insert_block` (with value 100 by default). If inserted block contains larger number of partitions, an exception is thrown. Set it to 0 if you want to remove the limit (not recommended). [#4845 \(alexey-milovidov\)](#)
- Multi-search functions were renamed (`multiPosition` to `multiSearchAllPositions`, `multiSearch` to `multiSearchAny`, `firstMatch` to `multiSearchFirstIndex`). [#4780 \(Danila Kutenin\)](#)

## Performance Improvement

- Optimize Volnitsky searcher by inlining, giving about 5-10% search improvement for queries with many needles or many similar bigrams. [#4862 \(Danila Kutenin\)](#)
- Fix performance issue when setting `use_uncompressed_cache` is greater than zero, which appeared when all read data contained in cache. [#4913 \(alesapin\)](#)

## Build/Testing/Packaging Improvement

- Hardening debug build: more granular memory mappings and ASLR; add memory protection for mark cache and index. This allows to find more memory stomping bugs in case when ASan and MSan cannot do it. [#4632 \(alexey-milovidov\)](#)
- Add support for cmake variables `ENABLE_PROTOBUF`, `ENABLE_PARQUET` and `ENABLE_BROTLI` which allows to enable/disable the above features (same as we can do for librdkafka, mysql, etc). [#4669 \(Silviu Caragea\)](#)
- Add ability to print process list and stacktraces of all threads if some queries are hung after test run. [#4675 \(alesapin\)](#)
- Add retries on `Connection loss` error in `clickhouse-test`. [#4682 \(alesapin\)](#)
- Add freebsd build with vagrant and build with thread sanitizer to packager script. [#4712 #4748 \(alesapin\)](#)
- Now user asked for password for user '`default`' during installation. [#4725 \(proller\)](#)
- Suppress warning in `rdkafka` library. [#4740 \(alexey-milovidov\)](#)
- Allow ability to build without ssl. [#4750 \(proller\)](#)
- Add a way to launch `clickhouse-server` image from a custom user. [#4753 \(Mikhail f. Shiryaev\)](#)

- Upgrade contrib boost to 1.69. [#4793](#) ([proller](#))
- Disable usage of `mremap` when compiled with Thread Sanitizer. Surprisingly enough, TSan does not intercept `mremap` (though it does intercept `mmap`, `munmap`) that leads to false positives. Fixed TSan report in stateful tests. [#4859](#) ([alexey-milovidov](#))
- Add test checking using format schema via HTTP interface. [#4864](#) ([Vitaly Baranov](#))

## ClickHouse Release 19.4

### ClickHouse Release 19.4.4.33, 2019-04-17

#### Bug Fixes

- Avoid `std::terminate` in case of memory allocation failure. Now `std::bad_alloc` exception is thrown as expected. [#4665](#) ([alexey-milovidov](#))
- Fixes capnproto reading from buffer. Sometimes files wasn't loaded successfully by HTTP. [#4674](#) ([Vladislav](#))
- Fix error Unknown log entry type: 0 after `OPTIMIZE TABLE FINAL` query. [#4683](#) ([Amos Bird](#))
- Wrong arguments to `hasAny` or `hasAll` functions may lead to segfault. [#4698](#) ([alexey-milovidov](#))
- Deadlock may happen while executing `DROP DATABASE` dictionary query. [#4701](#) ([alexey-milovidov](#))
- Fix undefined behavior in `median` and `quantile` functions. [#4702](#) ([hcz](#))
- Fix compression level detection when `network_compression_method` in lowercase. Broken in v19.1. [#4706](#) ([proller](#))
- Fixed ignorance of `<timezone>UTC</timezone>` setting (fixes issue [#4658](#)). [#4718](#) ([proller](#))
- Fix `histogram` function behaviour with `Distributed` tables. [#4741](#) ([olegkv](#))
- Fixed tsan report `destroy of a locked mutex`. [#4742](#) ([alexey-milovidov](#))
- Fixed TSan report on shutdown due to race condition in system logs usage. Fixed potential use-after-free on shutdown when `part_log` is enabled. [#4758](#) ([alexey-milovidov](#))
- Fix recheck parts in `ReplicatedMergeTreeAlterThread` in case of error. [#4772](#) ([Nikolai Kochetov](#))
- Arithmetic operations on intermediate aggregate function states were not working for constant arguments (such as subquery results). [#4776](#) ([alexey-milovidov](#))
- Always backquote column names in metadata. Otherwise it's impossible to create a table with column named `index` (server won't restart due to malformed `ATTACH` query in metadata). [#4782](#) ([alexey-milovidov](#))
- Fix crash in `ALTER ... MODIFY ORDER BY` on `Distributed` table. [#4790](#) ([TCeason](#))
- Fix segfault in `JOIN ON` with enabled `enable_optimize_predicate_expression`. [#4794](#) ([Winter Zhang](#))
- Fix bug with adding an extraneous row after consuming a protobuf message from Kafka. [#4808](#) ([Vitaly Baranov](#))
- Fix segmentation fault in `clickhouse-copier`. [#4835](#) ([proller](#))
- Fixed race condition in `SELECT` from `system.tables` if the table is renamed or altered concurrently. [#4836](#) ([alexey-milovidov](#))
- Fixed data race when fetching data part that is already obsolete. [#4839](#) ([alexey-milovidov](#))

- Fixed rare data race that can happen during `RENAME` table of MergeTree family. #4844 (alexey-milovidov)
- Fixed segmentation fault in function `arrayIntersect`. Segmentation fault could happen if function was called with mixed constant and ordinary arguments. #4847 (Lixiang Qian)
- Fixed reading from `Array(LowCardinality)` column in rare case when column contained a long sequence of empty arrays. #4850 (Nikolai Kochetov)
- Fix `No message received` exception while fetching parts between replicas. #4856 (alesapin)
- Fixed `arrayIntersect` function wrong result in case of several repeated values in single array. #4871 (Nikolai Kochetov)
- Fix a race condition during concurrent `ALTER COLUMN` queries that could lead to a server crash (fixes issue #3421). #4592 (Alex Zatelepin)
- Fix parameter deduction in `ALTER MODIFY` of column `CODEC` when column type is not specified. #4883 (alesapin)
- Functions `cutQueryStringAndFragment()` and `queryStringAndFragment()` now works correctly when `URL` contains a fragment and no query. #4894 (Vitaly Baranov)
- Fix rare bug when setting `min_bytes_to_use_direct_io` is greater than zero, which occurs when thread have to seek backward in column file. #4897 (alesapin)
- Fix wrong argument types for aggregate functions with `LowCardinality` arguments (fixes issue #4919). #4922 (Nikolai Kochetov)
- Fix function `toISOWeek` result for year 1970. #4988 (alexey-milovidov)
- Fix `DROP`, `TRUNCATE` and `OPTIMIZE` queries duplication, when executed on `ON CLUSTER` for `ReplicatedMergeTree*` tables family. #4991 (alesapin)

## Improvements

- Keep ordinary, `DEFAULT`, `MATERIALIZED` and `ALIAS` columns in a single list (fixes issue #2867). #4707 (Alex Zatelepin)

## ClickHouse Release 19.4.3.11, 2019-04-02

### Bug Fixes

- Fix crash in `FULL/RIGHT JOIN` when we joining on nullable vs not nullable. #4855 (Artem Zuikov)
- Fix segmentation fault in `clickhouse-copier`. #4835 (proller)

### Build/Testing/Packaging Improvement

- Add a way to launch `clickhouse-server` image from a custom user. #4753 (Mikhail f. Shiryaev)

## ClickHouse Release 19.4.2.7, 2019-03-30

### Bug Fixes

- Fixed reading from `Array(LowCardinality)` column in rare case when column contained a long sequence of empty arrays. #4850 (Nikolai Kochetov)

## ClickHouse Release 19.4.1.3, 2019-03-19

### Bug Fixes

- Fixed remote queries which contain both `LIMIT BY` and `LIMIT`. Previously, if `LIMIT BY` and `LIMIT` were used for remote query, `LIMIT` could happen before `LIMIT BY`, which led to too filtered result. [#4708](#) ([Constantin S. Pan](#))

## ClickHouse Release 19.4.0.49, 2019-03-09

### New Features

- Added full support for `Protobuf` format (input and output, nested data structures). [#4174](#) [#4493](#) ([Vitaly Baranov](#))
- Added bitmap functions with Roaring Bitmaps. [#4207](#) ([Andy Yang](#)) [#4568](#) ([Vitaly Baranov](#))
- Parquet format support. [#4448](#) ([proller](#))
- N-gram distance was added for fuzzy string comparison. It is similar to q-gram metrics in R language. [#4466](#) ([Danila Kutenin](#))
- Combine rules for graphite rollup from dedicated aggregation and retention patterns. [#4426](#) ([Mikhail f. Shiryaev](#))
- Added `max_execution_speed` and `max_execution_speed_bytes` to limit resource usage. Added `min_execution_speed_bytes` setting to complement the `min_execution_speed`. [#4430](#) ([Winter Zhang](#))
- Implemented function `flatten`. [#4555](#) [#4409](#) ([alexey-milovidov](#), [kzon](#))
- Added functions `arrayEnumerateDenseRanked` and `arrayEnumerateUniqRanked` (it's like `arrayEnumerateUniq` but allows to fine tune array depth to look inside multidimensional arrays). [#4475](#) ([proller](#)) [#4601](#) ([alexey-milovidov](#))
- Multiple JOINS with some restrictions: no asterisks, no complex aliases in ON/WHERE/GROUP BY/... [#4462](#) ([Artem Zuikov](#))

### Bug Fixes

- This release also contains all bug fixes from 19.3 and 19.1.
- Fixed bug in data skipping indices: order of granules after `INSERT` was incorrect. [#4407](#) ([Nikita Vasilev](#))
- Fixed `set` index for `Nullable` and `LowCardinality` columns. Before it, `set` index with `Nullable` or `LowCardinality` column led to error `Data type must be deserialized with multiple streams while selecting`. [#4594](#) ([Nikolai Kochetov](#))
- Correctly set `update_time` on full `executable` dictionary update. [#4551](#) ([Tema Novikov](#))
- Fix broken progress bar in 19.3. [#4627](#) ([filimonov](#))
- Fixed inconsistent values of `MemoryTracker` when memory region was shrunked, in certain cases. [#4619](#) ([alexey-milovidov](#))
- Fixed undefined behaviour in `ThreadPool`. [#4612](#) ([alexey-milovidov](#))
- Fixed a very rare crash with the message `mutex lock failed: Invalid argument` that could happen when a `MergeTree` table was dropped concurrently with a `SELECT`. [#4608](#) ([Alex Zatelepin](#))
- ODBC driver compatibility with `LowCardinality` data type. [#4381](#) ([proller](#))
- FreeBSD: Fixup for `AIOContextPool`: Found `io_event` with unknown id 0 error. [#4438](#) ([urgordeadbeef](#))
- `system.part_log` table was created regardless to configuration. [#4483](#) ([alexey-milovidov](#))
- Fix undefined behaviour in `dictIsIn` function for cache dictionaries. [#4515](#) ([alesapin](#))

- Fixed a deadlock when a SELECT query locks the same table multiple times (e.g. from different threads or when executing multiple subqueries) and there is a concurrent DDL query. [#4535 \(Alex Zatelepin\)](#)
- Disable compile\_expressions by default until we get own llvm contrib and can test it with clang and asan. [#4579 \(alesapin\)](#)
- Prevent `std::terminate` when `invalidate_query` for `clickhouse` external dictionary source has returned wrong resultset (empty or more than one row or more than one column). Fixed issue when the `invalidate_query` was performed every five seconds regardless to the lifetime. [#4583 \(alexey-milovidov\)](#)
- Avoid deadlock when the `invalidate_query` for a dictionary with `clickhouse` source was involving `system.dictionaries` table or `Dictionaries` database (rare case). [#4599 \(alexey-milovidov\)](#)
- Fixes for CROSS JOIN with empty WHERE. [#4598 \(Artem Zuikov\)](#)
- Fixed segfault in function “replicate” when constant argument is passed. [#4603 \(alexey-milovidov\)](#)
- Fix lambda function with predicate optimizer. [#4408 \(Winter Zhang\)](#)
- Multiple JOINs multiple fixes. [#4595 \(Artem Zuikov\)](#)

## Improvements

- Support aliases in JOIN ON section for right table columns. [#4412 \(Artem Zuikov\)](#)
- Result of multiple JOINs need correct result names to be used in subselects. Replace flat aliases with source names in result. [#4474 \(Artem Zuikov\)](#)
- Improve push-down logic for joined statements. [#4387 \(Ivan\)](#)

## Performance Improvements

- Improved heuristics of “move to PREWHERE” optimization. [#4405 \(alexey-milovidov\)](#)
- Use proper lookup tables that uses HashTable’s API for 8-bit and 16-bit keys. [#4536 \(Amos Bird\)](#)
- Improved performance of string comparison. [#4564 \(alexey-milovidov\)](#)
- Cleanup distributed DDL queue in a separate thread so that it does not slow down the main loop that processes distributed DDL tasks. [#4502 \(Alex Zatelepin\)](#)
- When `min_bytes_to_use_direct_io` is set to 1, not every file was opened with O\_DIRECT mode because the data size to read was sometimes underestimated by the size of one compressed block. [#4526 \(alexey-milovidov\)](#)

## Build/Testing/Packaging Improvement

- Added support for clang-9 [#4604 \(alexey-milovidov\)](#)
- Fix wrong `_asm_` instructions (again) [#4621 \(Konstantin Podshumok\)](#)
- Add ability to specify settings for `clickhouse-performance-test` from command line. [#4437 \(alesapin\)](#)
- Add dictionaries tests to integration tests. [#4477 \(alesapin\)](#)
- Added queries from the benchmark on the website to automated performance tests. [#4496 \(alexey-milovidov\)](#)
- `xxhash.h` does not exist in external lz4 because it is an implementation detail and its symbols are namespaced with `XXH_NAMESPACE` macro. When lz4 is external, xxHash has to be external too, and the dependents have to link to it. [#4495 \(Orivej Desh\)](#)

- Fixed a case when `quantileTiming` aggregate function can be called with negative or floating point argument (this fixes fuzz test with undefined behaviour sanitizer). [#4506 \(alexey-milovidov\)](#)
- Spelling error correction. [#4531 \(sdk2\)](#)
- Fix compilation on Mac. [#4371 \(Vitaly Baranov\)](#)
- Build fixes for FreeBSD and various unusual build configurations. [#4444 \(proller\)](#)

## ClickHouse Release 19.3

### ClickHouse Release 19.3.9.1, 2019-04-02

#### Bug Fixes

- Fix crash in `FULL/RIGHT JOIN` when we joining on nullable vs not nullable. [#4855 \(Artem Zuikov\)](#)
- Fix segmentation fault in `clickhouse-copier`. [#4835 \(proller\)](#)
- Fixed reading from `Array(LowCardinality)` column in rare case when column contained a long sequence of empty arrays. [#4850 \(Nikolai Kochetov\)](#)

#### Build/Testing/Packaging Improvement

- Add a way to launch `clickhouse-server` image from a custom user [#4753 \(Mikhail f. Shiryaev\)](#)

### ClickHouse Release 19.3.7, 2019-03-12

#### Bug Fixes

- Fixed error in #3920. This error manifests itself as random cache corruption (messages `Unknown codec family code, Cannot seek through file`) and segfaults. This bug first appeared in version 19.1 and is present in versions up to 19.1.10 and 19.3.6. [#4623 \(alexey-milovidov\)](#)

### ClickHouse Release 19.3.6, 2019-03-02

#### Bug Fixes

- When there are more than 1000 threads in a thread pool, `std::terminate` may happen on thread exit. [Azat Khuzhin #4485 #4505 \(alexey-milovidov\)](#)
- Now it's possible to create `ReplicatedMergeTree*` tables with comments on columns without defaults and tables with columns codecs without comments and defaults. Also fix comparison of codecs. [#4523 \(alesapin\)](#)
- Fixed crash on JOIN with array or tuple. [#4552 \(Artem Zuikov\)](#)
- Fixed crash in `clickhouse-copier` with the message `ThreadStatus not created`. [#4540 \(Artem Zuikov\)](#)
- Fixed hangup on server shutdown if distributed DDLs were used. [#4472 \(Alex Zatelepin\)](#)
- Incorrect column numbers were printed in error message about text format parsing for columns with number greater than 10. [#4484 \(alexey-milovidov\)](#)

#### Build/Testing/Packaging Improvements

- Fixed build with AVX enabled. [#4527 \(alexey-milovidov\)](#)
- Enable extended accounting and IO accounting based on good known version instead of kernel under which it is compiled. [#4541 \(nvartolomei\)](#)
- Allow to skip setting of `core_dump.size_limit`, warning instead of throw if limit set fail. [#4473 \(proller\)](#)

- Removed the `inline` tags of `void readBinary(...)` in `Field.cpp`. Also merged redundant `namespace DB` blocks. [#4530 \(hczi\)](#)

## ClickHouse Release 19.3.5, 2019-02-21

### Bug Fixes

- Fixed bug with large http insert queries processing. [#4454 \(alesapin\)](#)
- Fixed backward incompatibility with old versions due to wrong implementation of `send_logs_level` setting. [#4445 \(alexey-milovidov\)](#)
- Fixed backward incompatibility of table function `remote` introduced with column comments. [#4446 \(alexey-milovidov\)](#)

## ClickHouse Release 19.3.4, 2019-02-16

### Improvements

- Table index size is not accounted for memory limits when doing `ATTACH TABLE` query. Avoided the possibility that a table cannot be attached after being detached. [#4396 \(alexey-milovidov\)](#)
- Slightly raised up the limit on max string and array size received from ZooKeeper. It allows to continue to work with increased size of `CLIENT_JVMFLAGS=-Djute.maxbuffer=...` on ZooKeeper. [#4398 \(alexey-milovidov\)](#)
- Allow to repair abandoned replica even if it already has huge number of nodes in its queue. [#4399 \(alexey-milovidov\)](#)
- Add one required argument to `SET` index (max stored rows number). [#4386 \(Nikita Vasilev\)](#)

### Bug Fixes

- Fixed `WITH ROLLUP` result for group by single `LowCardinality` key. [#4384 \(Nikolai Kochetov\)](#)
- Fixed bug in the set index (dropping a granule if it contains more than `max_rows` rows). [#4386 \(Nikita Vasilev\)](#)
- A lot of FreeBSD build fixes. [#4397 \(proller\)](#)
- Fixed aliases substitution in queries with subquery containing same alias (issue [#4110](#)). [#4351 \(Artem Zuikov\)](#)

### Build/Testing/Packaging Improvements

- Add ability to run `clickhouse-server` for stateless tests in docker image. [#4347 \(Vasily Nemkov\)](#)

## ClickHouse Release 19.3.3, 2019-02-13

### New Features

- Added the `KILL MUTATION` statement that allows removing mutations that are for some reasons stuck. Added `latest_failed_part`, `latest_fail_time`, `latest_fail_reason` fields to the `system.mutations` table for easier troubleshooting. [#4287 \(Alex Zatelepin\)](#)
- Added aggregate function `entropy` which computes Shannon entropy. [#4238 \(Quid37\)](#)
- Added ability to send queries `INSERT INTO tbl VALUES (....)` to server without splitting on `query` and `data` parts. [#4301 \(alesapin\)](#)
- Generic implementation of `arrayWithConstant` function was added. [#4322 \(alexey-milovidov\)](#)
- Implemented `NOT BETWEEN` comparison operator. [#4228 \(Dmitry Naumov\)](#)

- Implement `sumMapFiltered` in order to be able to limit the number of keys for which values will be summed by `sumMap`. #4129 (Léo Ercolanelli)
- Added support of `Nullable` types in `mysql` table function. #4198 (Emmanuel Donin de Rosière)
- Support for arbitrary constant expressions in `LIMIT` clause. #4246 (k3box)
- Added `topKWeighted` aggregate function that takes additional argument with (unsigned integer) weight. #4245 (Andrew Golman)
- `StorageJoin` now supports `join_any_take_last_row` setting that allows overwriting existing values of the same key. #3973 (Amos Bird)
- Added function `toStartOfInterval`. #4304 (Vitaly Baranov)
- Added `RowBinaryWithNamesAndTypes` format. #4200 (Oleg V. Kozlyuk)
- Added IPv4 and IPv6 data types. More effective implementations of `IPv*` functions. #3669 (Vasily Nemkov)
- Added function `toStartOfTenMinutes()`. #4298 (Vitaly Baranov)
- Added `Protobuf` output format. #4005 #4158 (Vitaly Baranov)
- Added brotli support for HTTP interface for data import (INSERTs). #4235 (Mikhail)
- Added hints while user make typo in function name or type in command line client. #4239 (Danila Kutenin)
- Added `Query-Id` to Server's HTTP Response header. #4231 (Mikhail)

## Experimental Features

- Added `minmax` and `set` data skipping indices for MergeTree table engines family. #4143 (Nikita Vasilev)
- Added conversion of `CROSS JOIN` to `INNER JOIN` if possible. #4221 #4266 (Artem Zuikov)

## Bug Fixes

- Fixed `Not found column` for duplicate columns in `JOIN ON` section. #4279 (Artem Zuikov)
- Make `START REPLICATED SENDS` command start replicated sends. #4229 (nvartolomei)
- Fixed aggregate functions execution with `Array(LowCardinality)` arguments. #4055 (KochetovNicolai)
- Fixed wrong behaviour when doing `INSERT ... SELECT ... FROM file(...)` query and file has `CSVWithNames` or `TSVWithNames` format and the first data row is missing. #4297 (alexey-milovidov)
- Fixed crash on dictionary reload if dictionary not available. This bug was appeared in 19.1.6. #4188 (proller)
- Fixed `ALL JOIN` with duplicates in right table. #4184 (Artem Zuikov)
- Fixed segmentation fault with `use_uncompressed_cache=1` and exception with wrong uncompressed size. This bug was appeared in 19.1.6. #4186 (alesapin)
- Fixed `compile_expressions` bug with comparison of big (more than int16) dates. #4341 (alesapin)
- Fixed infinite loop when selecting from table function `numbers(0)`. #4280 (alexey-milovidov)
- Temporarily disable predicate optimization for `ORDER BY`. #3890 (Winter Zhang)
- Fixed `Illegal instruction` error when using base64 functions on old CPUs. This error has been reproduced only when ClickHouse was compiled with gcc-8. #4275 (alexey-milovidov)

- Fixed No message received error when interacting with PostgreSQL ODBC Driver through TLS connection. Also fixes segfault when using MySQL ODBC Driver. [#4170 \(alexey-milovidov\)](#)
- Fixed incorrect result when `Date` and `DateTime` arguments are used in branches of conditional operator (function `if`). Added generic case for function `if`. [#4243 \(alexey-milovidov\)](#)
- ClickHouse dictionaries now load within `clickhouse` process. [#4166 \(alexey-milovidov\)](#)
- Fixed deadlock when `SELECT` from a table with `File` engine was retried after `No such file or directory` error. [#4161 \(alexey-milovidov\)](#)
- Fixed race condition when selecting from `system.tables` may give `table does not exist` error. [#4313 \(alexey-milovidov\)](#)
- `clickhouse-client` can segfault on exit while loading data for command line suggestions if it was run in interactive mode. [#4317 \(alexey-milovidov\)](#)
- Fixed a bug when the execution of mutations containing `IN` operators was producing incorrect results. [#4099 \(Alex Zatelepin\)](#)
- Fixed error: if there is a database with `Dictionary` engine, all dictionaries forced to load at server startup, and if there is a dictionary with ClickHouse source from localhost, the dictionary cannot load. [#4255 \(alexey-milovidov\)](#)
- Fixed error when system logs are tried to create again at server shutdown. [#4254 \(alexey-milovidov\)](#)
- Correctly return the right type and properly handle locks in `joinGet` function. [#4153 \(Amos Bird\)](#)
- Added `sumMapWithOverflow` function. [#4151 \(Léo Ercolanelli\)](#)
- Fixed segfault with `allow_experimental_multiple_joins_emulation`. [52de2c \(Artem Zuikov\)](#)
- Fixed bug with incorrect `Date` and `DateTime` comparison. [#4237 \(valexey\)](#)
- Fixed fuzz test under undefined behavior sanitizer: added parameter type check for `quantile*Weighted` family of functions. [#4145 \(alexey-milovidov\)](#)
- Fixed rare race condition when removing of old data parts can fail with `File not found` error. [#4378 \(alexey-milovidov\)](#)
- Fix install package with missing `/etc/clickhouse-server/config.xml`. [#4343 \(proller\)](#)

## Build/Testing/Packaging Improvements

- Debian package: correct `/etc/clickhouse-server/preprocessed` link according to config. [#4205 \(proller\)](#)
- Various build fixes for FreeBSD. [#4225 \(proller\)](#)
- Added ability to create, fill and drop tables in `perftest`. [#4220 \(alesapin\)](#)
- Added a script to check for duplicate includes. [#4326 \(alexey-milovidov\)](#)
- Added ability to run queries by index in performance test. [#4264 \(alesapin\)](#)
- Package with debug symbols is suggested to be installed. [#4274 \(alexey-milovidov\)](#)
- Refactoring of `performance-test`. Better logging and signals handling. [#4171 \(alesapin\)](#)
- Added docs to anonymized Yandex.Metrika datasets. [#4164 \(alesapin\)](#)
- Added tool for converting an old month-partitioned part to the custom-partitioned format. [#4195 \(Alex Zatelepin\)](#)

- Added docs about two datasets in s3. #4144 (alesapin)
- Added script which creates changelog from pull requests description. #4169 #4173 (KochetovNicolai) (KochetovNicolai)
- Added puppet module for ClickHouse. #4182 (Maxim Fedotov)
- Added docs for a group of undocumented functions. #4168 (Winter Zhang)
- ARM build fixes. #4210#4306 #4291 (proller) (proller)
- Dictionary tests now able to run from `ctest`. #4189 (proller)
- Now `/etc/ssl` is used as default directory with SSL certificates. #4167 (alexey-milovidov)
- Added checking SSE and AVX instruction at start. #4234 (lgr)
- Init script will wait server until start. #4281 (proller)

## Backward Incompatible Changes

- Removed `allow_experimental_low_cardinality_type` setting. `LowCardinality` data types are production ready. #4323 (alexey-milovidov)
- Reduce mark cache size and uncompressed cache size accordingly to available memory amount. #4240 (Lopatin Konstantin)
- Added keyword `INDEX` in `CREATE TABLE` query. A column with name `index` must be quoted with backticks or double quotes: ``index``. #4143 (Nikita Vasilev)
- `sumMap` now promote result type instead of overflow. The old `sumMap` behavior can be obtained by using `sumMapWithOverflow` function. #4151 (Léo Ercolanelli)

## Performance Improvements

- `std::sort` replaced by `pdqsort` for queries without `LIMIT`. #4236 (Evgenii Pravda)
- Now server reuse threads from global thread pool. This affects performance in some corner cases. #4150 (alexey-milovidov)

## Improvements

- Implemented AIO support for FreeBSD. #4305 (urgordeadbeef)
- `SELECT * FROM a JOIN b USING a, b` now return `a` and `b` columns only from the left table. #4141 (Artem Zuikov)
- Allow `-C` option of client to work as `-c` option. #4232 (syominsergey)
- Now option `--password` used without value requires password from stdin. #4230 (BSD\_Conqueror)
- Added highlighting of unescaped metacharacters in string literals that contain `LIKE` expressions or regexps. #4327 (alexey-milovidov)
- Added cancelling of HTTP read only queries if client socket goes away. #4213 (nvartolomei)
- Now server reports progress to keep client connections alive. #4215 (Ivan)
- Slightly better message with reason for `OPTIMIZE` query with `optimize_throw_if_noop` setting enabled. #4294 (alexey-milovidov)
- Added support of `--version` option for clickhouse server. #4251 (Lopatin Konstantin)
- Added `--help/-h` option to `clickhouse-server`. #4233 (Yuriy Baranov)

- Added support for scalar subqueries with aggregate function state result. [#4348](#) ([Nikolai Kochetov](#))
- Improved server shutdown time and ALTERs waiting time. [#4372](#) ([alexey-milovidov](#))
- Added info about the replicated\_can\_become\_leader setting to system.replicas and add logging if the replica won't try to become leader. [#4379](#) ([Alex Zatelepin](#))

## ClickHouse Release 19.1

### ClickHouse Release 19.1.14, 2019-03-14

- Fixed error `Column ... queried more than once` that may happen if the setting `asterisk_left_columns_only` is set to 1 in case of using `GLOBAL JOIN` with `SELECT *` (rare case). The issue does not exist in 19.3 and newer. [#6bac7d8d](#) ([Artem Zuikov](#))

### ClickHouse Release 19.1.13, 2019-03-12

This release contains exactly the same set of patches as 19.3.7.

### ClickHouse Release 19.1.10, 2019-03-03

This release contains exactly the same set of patches as 19.3.6.

## ClickHouse Release 19.1

### ClickHouse Release 19.1.9, 2019-02-21

#### Bug Fixes

- Fixed backward incompatibility with old versions due to wrong implementation of `send_logs_level` setting. [#4445](#) ([alexey-milovidov](#))
- Fixed backward incompatibility of table function `remote` introduced with column comments. [#4446](#) ([alexey-milovidov](#))

### ClickHouse Release 19.1.8, 2019-02-16

#### Bug Fixes

- Fix install package with missing `/etc/clickhouse-server/config.xml`. [#4343](#) ([proller](#))

## ClickHouse Release 19.1

### ClickHouse Release 19.1.7, 2019-02-15

#### Bug Fixes

- Correctly return the right type and properly handle locks in `joinGet` function. [#4153](#) ([Amos Bird](#))
- Fixed error when system logs are tried to create again at server shutdown. [#4254](#) ([alexey-milovidov](#))
- Fixed error: if there is a database with `Dictionary` engine, all dictionaries forced to load at server startup, and if there is a dictionary with ClickHouse source from localhost, the dictionary cannot load. [#4255](#) ([alexey-milovidov](#))
- Fixed a bug when the execution of mutations containing `IN` operators was producing incorrect results. [#4099](#) ([Alex Zatelepin](#))
- `clickhouse-client` can segfault on exit while loading data for command line suggestions if it was run in interactive mode. [#4317](#) ([alexey-milovidov](#))

- Fixed race condition when selecting from `system.tables` may give `table does not exist` error. [#4313](#) ([alexey-milovidov](#))
- Fixed deadlock when `SELECT` from a table with `File` engine was retried after `No such file or directory` error. [#4161](#) ([alexey-milovidov](#))
- Fixed an issue: local ClickHouse dictionaries are loaded via TCP, but should load within process. [#4166](#) ([alexey-milovidov](#))
- Fixed `No message received` error when interacting with PostgreSQL ODBC Driver through TLS connection. Also fixes segfault when using MySQL ODBC Driver. [#4170](#) ([alexey-milovidov](#))
- Temporarily disable predicate optimization for `ORDER BY`. [#3890](#) ([Winter Zhang](#))
- Fixed infinite loop when selecting from table function `numbers(0)`. [#4280](#) ([alexey-milovidov](#))
- Fixed `compile_expressions` bug with comparison of big (more than `int16`) dates. [#4341](#) ([alesapin](#))
- Fixed segmentation fault with `uncompressed_cache=1` and exception with wrong uncompressed size. [#4186](#) ([alesapin](#))
- Fixed ALL JOIN with duplicates in right table. [#4184](#) ([Artem Zuikov](#))
- Fixed wrong behaviour when doing `INSERT ... SELECT ... FROM file(...)` query and file has `CSVWithNames` or `TSVWithNames` format and the first data row is missing. [#4297](#) ([alexey-milovidov](#))
- Fixed aggregate functions execution with `Array(LowCardinality)` arguments. [#4055](#) ([KochetovNicolai](#))
- Debian package: correct `/etc/clickhouse-server/preprocessed` link according to config. [#4205](#) ([proller](#))
- Fixed fuzz test under undefined behavior sanitizer: added parameter type check for `quantile*Weighted` family of functions. [#4145](#) ([alexey-milovidov](#))
- Make `START REPLICATED SENDS` command start replicated sends. [#4229](#) ([nvartolomei](#))
- Fixed `Not found column` for duplicate columns in `JOIN ON` section. [#4279](#) ([Artem Zuikov](#))
- Now `/etc/ssl` is used as default directory with SSL certificates. [#4167](#) ([alexey-milovidov](#))
- Fixed crash on dictionary reload if dictionary not available. [#4188](#) ([proller](#))
- Fixed bug with incorrect `Date` and `DateTime` comparison. [#4237](#) ([valexey](#))
- Fixed incorrect result when `Date` and `DateTime` arguments are used in branches of conditional operator (function `if`). Added generic case for function `if`. [#4243](#) ([alexey-milovidov](#))

## ClickHouse Release 19.1.6, 2019-01-24

### New Features

- Custom per column compression codecs for tables. [#3899](#) [#4111](#) ([alesapin](#), [Winter Zhang](#), [Anatoly](#))
- Added compression codec Delta. [#4052](#) ([alesapin](#))
- Allow to `ALTER` compression codecs. [#4054](#) ([alesapin](#))
- Added functions `left`, `right`, `trim`, `ltrim`, `rtrim`, `timestampadd`, `timestampsub` for SQL standard compatibility. [#3826](#) ([Ivan Blinkov](#))
- Support for write in `HDFS` tables and `hdfs` table function. [#4084](#) ([alesapin](#))
- Added functions to search for multiple constant strings from big haystack: `multiPosition`, `multiSearch`, `firstMatch` also with `-UTF8`, `-CaseInsensitive`, and `-CaseInsensitiveUTF8` variants. [#4053](#) ([Danila Kutenin](#))

- Pruning of unused shards if `SELECT` query filters by sharding key (setting `optimize_skip_unused_shards`).  
[#3851 \(Gleb Kanterov, Ivan\)](#)
- Allow `Kafka` engine to ignore some number of parsing errors per block. [#4094 \(Ivan\)](#)
- Added support for `CatBoost` multiclass models evaluation. Function `modelEvaluate` returns tuple with per-class raw predictions for multiclass models. `libcatboostmodel.so` should be built with [#607](#). [#3959 \(KochetovNicola\)](#)
- Added functions `filesystemAvailable`, `filesystemFree`, `filesystemCapacity`. [#4097 \(Boris Granveaud\)](#)
- Added hashing functions `xxHash64` and `xxHash32`. [#3905 \(filimonov\)](#)
- Added `gccMurmurHash` hashing function (GCC flavoured Murmur hash) which uses the same hash seed as `gcc` [#4000 \(sundyli\)](#)
- Added hashing functions `javaHash`, `hiveHash`. [#3811 \(shangshujie365\)](#)
- Added table function `remoteSecure`. Function works as `remote`, but uses secure connection. [#4088 \(proller\)](#)

## Experimental Features

- Added multiple JOINs emulation (`allow_experimental_multiple_joins_emulation` setting). [#3946 \(Artem Zuikov\)](#)

## Bug Fixes

- Make `compiled_expression_cache_size` setting limited by default to lower memory consumption. [#4041 \(alesapin\)](#)
- Fix a bug that led to hangups in threads that perform ALTERs of Replicated tables and in the thread that updates configuration from ZooKeeper. [#2947](#) [#3891](#) [#3934 \(Alex Zatelepin\)](#)
- Fixed a race condition when executing a distributed ALTER task. The race condition led to more than one replica trying to execute the task and all replicas except one failing with a ZooKeeper error. [#3904 \(Alex Zatelepin\)](#)
- Fix a bug when `from_zk` config elements weren't refreshed after a request to ZooKeeper timed out. [#2947](#) [#3947 \(Alex Zatelepin\)](#)
- Fix bug with wrong prefix for IPv4 subnet masks. [#3945 \(alesapin\)](#)
- Fixed crash (`std::terminate`) in rare cases when a new thread cannot be created due to exhausted resources. [#3956 \(alexey-milovidov\)](#)
- Fix bug when in `remote` table function execution when wrong restrictions were used for in `getStructureOfRemoteTable`. [#4009 \(alesapin\)](#)
- Fix a leak of netlink sockets. They were placed in a pool where they were never deleted and new sockets were created at the start of a new thread when all current sockets were in use. [#4017 \(Alex Zatelepin\)](#)
- Fix bug with closing `/proc/self/fd` directory earlier than all fds were read from `/proc` after forking odbc-bridge subprocess. [#4120 \(alesapin\)](#)
- Fixed String to UInt monotonic conversion in case of usage String in primary key. [#3870 \(Winter Zhang\)](#)
- Fixed error in calculation of integer conversion function monotonicity. [#3921 \(alexey-milovidov\)](#)
- Fixed segfault in `arrayEnumerateUniq`, `arrayEnumerateDense` functions in case of some invalid arguments. [#3909 \(alexey-milovidov\)](#)
- Fix UB in StorageMerge. [#3910 \(Amos Bird\)](#)

- Fixed segfault in functions `addDays`, `subtractDays`. #3913 (alexey-milovidov)
- Fixed error: functions `round`, `floor`, `trunc`, `ceil` may return bogus result when executed on integer argument and large negative scale. #3914 (alexey-milovidov)
- Fixed a bug induced by ‘kill query sync’ which leads to a core dump. #3916 (muVulDeePecker)
- Fix bug with long delay after empty replication queue. #3928 #3932 (alesapin)
- Fixed excessive memory usage in case of inserting into table with `LowCardinality` primary key. #3955 (KochetovNicolai)
- Fixed `LowCardinality` serialization for `Native` format in case of empty arrays. #3907 #4011 (KochetovNicolai)
- Fixed incorrect result while using distinct by single `LowCardinality` numeric column. #3895 #4012 (KochetovNicolai)
- Fixed specialized aggregation with `LowCardinality` key (in case when `compile` setting is enabled). #3886 (KochetovNicolai)
- Fix user and password forwarding for replicated tables queries. #3957 (alesapin) (小路)
- Fixed very rare race condition that can happen when listing tables in Dictionary database while reloading dictionaries. #3970 (alexey-milovidov)
- Fixed incorrect result when HAVING was used with ROLLUP or CUBE. #3756 #3837 (Sam Chou)
- Fixed column aliases for query with `JOIN ON` syntax and distributed tables. #3980 (Winter Zhang)
- Fixed error in internal implementation of `quantileTDigest` (found by Artem Vakhrushev). This error never happens in ClickHouse and was relevant only for those who use ClickHouse codebase as a library directly. #3935 (alexey-milovidov)

## Improvements

- Support for `IF NOT EXISTS` in `ALTER TABLE ADD COLUMN` statements along with `IF EXISTS` in `DROP/MODIFY/CLEAR/COMMENT COLUMN`. #3900 (Boris Granveaud)
- Function `parseDateTimeBestEffort`: support for formats `DD.MM.YYYY`, `DD.MM.YY`, `DD-MM-YYYY`, `DD-Mon-YYYY`, `DD/Month/YYYY` and similar. #3922 (alexey-milovidov)
- `CapnProtoInputStream` now support jagged structures. #4063 (Odin Hultgren Van Der Horst)
- Usability improvement: added a check that server process is started from the data directory’s owner. Do not allow to start server from root if the data belongs to non-root user. #3785 (sergey-v-galtsev)
- Better logic of checking required columns during analysis of queries with JOINs. #3930 (Artem Zuikov)
- Decreased the number of connections in case of large number of Distributed tables in a single server. #3726 (Winter Zhang)
- Supported totals row for `WITH TOTALS` query for ODBC driver. #3836 (Maksim Koritckiy)
- Allowed to use `Enums` as integers inside if function. #3875 (Ivan)
- Added `low_cardinality_allow_in_native_format` setting. If disabled, do not use `LowCardinality` type in `Native` format. #3879 (KochetovNicolai)
- Removed some redundant objects from compiled expressions cache to lower memory usage. #4042 (alesapin)

- Add check that `SET send_logs_level = 'value'` query accept appropriate value. #3873 (Sabyanin Maxim)
- Fixed data type check in type conversion functions. #3896 (Winter Zhang)

## Performance Improvements

- Add a MergeTree setting `use_minimalistic_part_header_in_zookeeper`. If enabled, Replicated tables will store compact part metadata in a single part znode. This can dramatically reduce ZooKeeper snapshot size (especially if the tables have a lot of columns). Note that after enabling this setting you will not be able to downgrade to a version that does not support it. #3960 (Alex Zatelepin)
- Add an DFA-based implementation for functions `sequenceMatch` and `sequenceCount` in case pattern does not contain time. #4004 (Léo Ercolanelli)
- Performance improvement for integer numbers serialization. #3968 (Amos Bird)
- Zero left padding `PODArray` so that -1 element is always valid and zeroed. It's used for branchless calculation of offsets. #3920 (Amos Bird)
- Reverted `jemalloc` version which lead to performance degradation. #4018 (alexey-milovidov)

## Backward Incompatible Changes

- Removed undocumented feature `ALTER MODIFY PRIMARY KEY` because it was superseded by the `ALTER MODIFY ORDER BY` command. #3887 (Alex Zatelepin)
- Removed function `shardByHash`. #3833 (alexey-milovidov)
- Forbid using scalar subqueries with result of type `AggregateFunction`. #3865 (Ivan)

## Build/Testing/Packaging Improvements

- Added support for PowerPC (`ppc64le`) build. #4132 (Danila Kutenin)
- Stateful functional tests are run on public available dataset. #3969 (alexey-milovidov)
- Fixed error when the server cannot start with the `bash: /usr/bin/clickhouse-extract-from-config: Operation not permitted` message within Docker or `systemd-nspawn`. #4136 (alexey-milovidov)
- Updated `rdkafka` library to v1.0.0-RC5. Used `cppkafka` instead of raw C interface. #4025 (Ivan)
- Updated `mariadb-client` library. Fixed one of issues found by UBSan. #3924 (alexey-milovidov)
- Some fixes for UBSan builds. #3926 #3021 #3948 (alexey-milovidov)
- Added per-commit runs of tests with UBSan build.
- Added per-commit runs of PVS-Studio static analyzer.
- Fixed bugs found by PVS-Studio. #4013 (alexey-milovidov)
- Fixed glibc compatibility issues. #4100 (alexey-milovidov)
- Move Docker images to 18.10 and add compatibility file for glibc >= 2.28 #3965 (alesapin)
- Add env variable if user do not want to chown directories in server Docker image. #3967 (alesapin)
- Enabled most of the warnings from `-Weverything` in clang. Enabled `-Wpedantic`. #3986 (alexey-milovidov)
- Added a few more warnings that are available only in clang 8. #3993 (alexey-milovidov)
- Link to `libLLVM` rather than to individual LLVM libs when using shared linking. #3989 (Orivej Desh)
- Added sanitizer variables for test images. #4072 (alesapin)

- clickhouse-server debian package will recommend `libcap2-bin` package to use `setcap` tool for setting capabilities. This is optional. [#4093 \(alexey-milovidov\)](#)
- Improved compilation time, fixed includes. [#3898 \(proller\)](#)
- Added performance tests for hash functions. [#3918 \(filimonov\)](#)
- Fixed cyclic library dependences. [#3958 \(proller\)](#)
- Improved compilation with low available memory. [#4030 \(proller\)](#)
- Added test script to reproduce performance degradation in jemalloc. [#4036 \(alexey-milovidov\)](#)
- Fixed misspells in comments and string literals under dbms. [#4122 \(maiha\)](#)
- Fixed typos in comments. [#4089 \(Evgenii Pravda\)](#)

## Changelog for 2018

---

### ClickHouse Release 18.16

#### ClickHouse Release 18.16.1, 2018-12-21

##### Bug Fixes:

- Fixed an error that led to problems with updating dictionaries with the ODBC source. [#3825, #3829](#)
- JIT compilation of aggregate functions now works with LowCardinality columns. [#3838](#)

##### Improvements:

- Added the `low_cardinality_allow_in_native_format` setting (enabled by default). When disabled, LowCardinality columns will be converted to ordinary columns for SELECT queries and ordinary columns will be expected for INSERT queries. [#3879](#)

##### Build Improvements:

- Fixes for builds on macOS and ARM.

#### ClickHouse Release 18.16.0, 2018-12-14

##### New Features:

- DEFAULT expressions are evaluated for missing fields when loading data in semi-structured input formats (`JSONEachRow`, `TSKV`). The feature is enabled with the `insert_sample_with_metadata` setting. [#3555](#)
- The `ALTER TABLE` query now has the `MODIFY ORDER BY` action for changing the sorting key when adding or removing a table column. This is useful for tables in the `MergeTree` family that perform additional tasks when merging based on this sorting key, such as `SummingMergeTree`, `AggregatingMergeTree`, and so on. [#3581 #3755](#)
- For tables in the `MergeTree` family, now you can specify a different sorting key (`ORDER BY`) and index (`PRIMARY KEY`). The sorting key can be longer than the index. [#3581](#)
- Added the `hdfs` table function and the `HDFS` table engine for importing and exporting data to HDFS. [chenxing-xc](#)
- Added functions for working with base64: `base64Encode`, `base64Decode`, `tryBase64Decode`. [Alexander Krasheninnikov](#)

- Now you can use a parameter to configure the precision of the `uniqCombined` aggregate function (select the number of HyperLogLog cells). [#3406](#)
- Added the `system.contributors` table that contains the names of everyone who made commits in ClickHouse. [#3452](#)
- Added the ability to omit the partition for the `ALTER TABLE ... FREEZE` query in order to back up all partitions at once. [#3514](#)
- Added `dictGet` and `dictGetOrDefault` functions that do not require specifying the type of return value. The type is determined automatically from the dictionary description. [Amos Bird](#)
- Now you can specify comments for a column in the table description and change it using `ALTER`. [#3377](#)
- Reading is supported for `Join` type tables with simple keys. [Amos Bird](#)
- Now you can specify the options `join_use_nulls`, `max_rows_in_join`, `max_bytes_in_join`, and `join_overflow_mode` when creating a `Join` type table. [Amos Bird](#)
- Added the `joinGet` function that allows you to use a `Join` type table like a dictionary. [Amos Bird](#)
- Added the `partition_key`, `sorting_key`, `primary_key`, and `sampling_key` columns to the `system.tables` table in order to provide information about table keys. [#3609](#)
- Added the `is_in_partition_key`, `is_in_sorting_key`, `is_in_primary_key`, and `is_in_sampling_key` columns to the `system.columns` table. [#3609](#)
- Added the `min_time` and `max_time` columns to the `system.parts` table. These columns are populated when the partitioning key is an expression consisting of `DateTime` columns. [Emmanuel Donin de Rosière](#)

## Bug Fixes:

- Fixes and performance improvements for the `LowCardinality` data type. `GROUP BY` using `LowCardinality(Nullable(...))`. Getting the values of extremes. Processing high-order functions. `LEFT ARRAY JOIN`. Distributed `GROUP BY`. Functions that return `Array`. Execution of `ORDER BY`. Writing to `Distributed` tables (nicelulu). Backward compatibility for `INSERT` queries from old clients that implement the `Native` protocol. Support for `LowCardinality` for `JOIN`. Improved performance when working in a single stream. [#3823](#) [#3803](#) [#3799](#) [#3769](#) [#3744](#) [#3681](#) [#3651](#) [#3649](#) [#3641](#) [#3632](#) [#3568](#) [#3523](#) [#3518](#)
- Fixed how the `select_sequential_consistency` option works. Previously, when this setting was enabled, an incomplete result was sometimes returned after beginning to write to a new partition. [#2863](#)
- Databases are correctly specified when executing DDL `ON CLUSTER` queries and `ALTER UPDATE/DELETE`. [#3772](#) [#3460](#)
- Databases are correctly specified for subqueries inside a `VIEW`. [#3521](#)
- Fixed a bug in `PREWHERE` with `FINAL` for `VersionedCollapsingMergeTree`. [7167bfd7](#)
- Now you can use `KILL QUERY` to cancel queries that have not started yet because they are waiting for the table to be locked. [#3517](#)
- Corrected date and time calculations if the clocks were moved back at midnight (this happens in Iran, and happened in Moscow from 1981 to 1983). Previously, this led to the time being reset a day earlier than necessary, and also caused incorrect formatting of the date and time in text format. [#3819](#)
- Fixed bugs in some cases of `VIEW` and subqueries that omit the database. [Winter Zhang](#)
- Fixed a race condition when simultaneously reading from a `MATERIALIZED VIEW` and deleting a `MATERIALIZED VIEW` due to not locking the internal `MATERIALIZED VIEW`. [#3404](#) [#3694](#)

- Fixed the error `Lock handler cannot be nullptr`. [#3689](#)
- Fixed query processing when the `compile_expressions` option is enabled (it's enabled by default). Nondeterministic constant expressions like the `now` function are no longer unfolded. [#3457](#)
- Fixed a crash when specifying a non-constant scale argument in `toDecimal32/64/128` functions.
- Fixed an error when trying to insert an array with `NULL` elements in the `Values` format into a column of type `Array` without `Nullable` (if `input_format_values_interpret_expressions = 1`). [#3487](#) [#3503](#)
- Fixed continuous error logging in `DDLWorker` if ZooKeeper is not available. [8f50c620](#)
- Fixed the return type for `quantile*` functions from `Date` and `DateTime` types of arguments. [#3580](#)
- Fixed the `WITH` clause if it specifies a simple alias without expressions. [#3570](#)
- Fixed processing of queries with named sub-queries and qualified column names when `enable_optimize_predicate_expression` is enabled. [Winter Zhang](#)
- Fixed the error `Attempt to attach to nullptr thread group` when working with materialized views. [Marek Vavruša](#)
- Fixed a crash when passing certain incorrect arguments to the `arrayReverse` function. [73e3a7b6](#)
- Fixed the buffer overflow in the `extractURLParameter` function. Improved performance. Added correct processing of strings containing zero bytes. [141e9799](#)
- Fixed buffer overflow in the `lowerUTF8` and `upperUTF8` functions. Removed the ability to execute these functions over `FixedString` type arguments. [#3662](#)
- Fixed a rare race condition when deleting `MergeTree` tables. [#3680](#)
- Fixed a race condition when reading from `Buffer` tables and simultaneously performing `ALTER` or `DROP` on the target tables. [#3719](#)
- Fixed a segfault if the `max_temporary_non_const_columns` limit was exceeded. [#3788](#)

## Improvements:

- The server does not write the processed configuration files to the `/etc/clickhouse-server/` directory. Instead, it saves them in the `preprocessed_configs` directory inside `path`. This means that the `/etc/clickhouse-server/` directory does not have write access for the `clickhouse` user, which improves security. [#2443](#)
- The `min_merge_bytes_to_use_direct_io` option is set to 10 GiB by default. A merge that forms large parts of tables from the `MergeTree` family will be performed in `O_DIRECT` mode, which prevents excessive page cache eviction. [#3504](#)
- Accelerated server start when there is a very large number of tables. [#3398](#)
- Added a connection pool and HTTP Keep-Alive for connections between replicas. [#3594](#)
- If the query syntax is invalid, the 400 Bad Request code is returned in the `HTTP` interface (500 was returned previously). [31bc680a](#)
- The `join_default_strictness` option is set to `ALL` by default for compatibility. [120e2cbe](#)
- Removed logging to `stderr` from the `re2` library for invalid or complex regular expressions. [#3723](#)
- Added for the `Kafka` table engine: checks for subscriptions before beginning to read from Kafka; the `kafka_max_block_size` setting for the table. [Marek Vavruša](#)

- The `cityHash64`, `farmHash64`, `metroHash64`, `sipHash64`, `halfMD5`, `murmurHash2_32`, `murmurHash2_64`, `murmurHash3_32`, and `murmurHash3_64` functions now work for any number of arguments and for arguments in the form of tuples. [#3451](#) [#3519](#)
- The `arrayReverse` function now works with any types of arrays. [#73e3a7b6](#)
- Added an optional parameter: the slot size for the `timeSlots` function. [Kirill Shvakov](#)
- For `FULL` and `RIGHT JOIN`, the `max_block_size` setting is used for a stream of non-joined data from the right table. [Amos Bird](#)
- Added the `--secure` command line parameter in `clickhouse-benchmark` and `clickhouse-performance-test` to enable TLS. [#3688](#) [#3690](#)
- Type conversion when the structure of a `Buffer` type table does not match the structure of the destination table. [Vitaly Baranov](#)
- Added the `tcp_keep_alive_timeout` option to enable keep-alive packets after inactivity for the specified time interval. [#3441](#)
- Removed unnecessary quoting of values for the partition key in the `system.parts` table if it consists of a single column. [#3652](#)
- The modulo function works for `Date` and `DateTime` data types. [#3385](#)
- Added synonyms for the `POWER`, `LN`, `LCASE`, `UCASE`, `REPLACE`, `LOCATE`, `SUBSTR`, and `MID` functions. [#3774](#) [#3763](#) Some function names are case-insensitive for compatibility with the SQL standard. Added syntactic sugar `SUBSTRING(expr FROM start FOR length)` for compatibility with SQL. [#3804](#)
- Added the ability to mlock memory pages corresponding to `clickhouse-server` executable code to prevent it from being forced out of memory. This feature is disabled by default. [#3553](#)
- Improved performance when reading from `O_DIRECT` (with the `min_bytes_to_use_direct_io` option enabled). [#3405](#)
- Improved performance of the `dictGet...OrDefault` function for a constant key argument and a non-constant default argument. [Amos Bird](#)
- The `firstSignificantSubdomain` function now processes the domains `gov`, `mil`, and `edu`. [Igor Hatarist](#) Improved performance. [#3628](#)
- Ability to specify custom environment variables for starting `clickhouse-server` using the `SYS-V init.d` script by defining `CLICKHOUSE_PROGRAM_ENV` in `/etc/default/clickhouse`.  
[Pavlo Bashynskyi](#)
- Correct return code for the `clickhouse-server` init script. [#3516](#)
- The `system.metrics` table now has the `VersionInteger` metric, and `system.build_options` has the added line `VERSION_INTEGER`, which contains the numeric form of the ClickHouse version, such as `18016000`. [#3644](#)
- Removed the ability to compare the `Date` type with a number to avoid potential errors like `date = 2018-12-17`, where quotes around the date are omitted by mistake. [#3687](#)
- Fixed the behavior of stateful functions like `rowNumberInAllBlocks`. They previously output a result that was one number larger due to starting during query analysis. [Amos Bird](#)
- If the `force_restore_data` file can't be deleted, an error message is displayed. [Amos Bird](#)

## Build Improvements:

- Updated the `jemalloc` library, which fixes a potential memory leak. [Amos Bird](#)

- Profiling with `jemalloc` is enabled by default in order to debug builds. [#2cc82f5c](#)
- Added the ability to run integration tests when only `Docker` is installed on the system. [#3650](#)
- Added the fuzz expression test in `SELECT` queries. [#3442](#)
- Added a stress test for commits, which performs functional tests in parallel and in random order to detect more race conditions. [#3438](#)
- Improved the method for starting `clickhouse-server` in a Docker image. [Elghazal Ahmed](#)
- For a Docker image, added support for initializing databases using files in the `/docker-entrypoint-initdb.d` directory. [Konstantin Lebedev](#)
- Fixes for builds on ARM. [#3709](#)

#### Backward Incompatible Changes:

- Removed the ability to compare the `Date` type with a number. Instead of `toDate('2018-12-18') = 17883`, you must use explicit type conversion `= toDate(17883)` [#3687](#)

## ClickHouse Release 18.14

### ClickHouse Release 18.14.19, 2018-12-19

#### Bug Fixes:

- Fixed an error that led to problems with updating dictionaries with the ODBC source. [#3825](#), [#3829](#)
- Databases are correctly specified when executing DDL `ON CLUSTER` queries. [#3460](#)
- Fixed a segfault if the `max_temporary_non_const_columns` limit was exceeded. [#3788](#)

#### Build Improvements:

- Fixes for builds on ARM.

### ClickHouse Release 18.14.18, 2018-12-04

#### Bug Fixes:

- Fixed error in `dictGet...` function for dictionaries of type `range`, if one of the arguments is constant and other is not. [#3751](#)
- Fixed error that caused messages `netlink: ...: attribute type 1 has an invalid length` to be printed in Linux kernel log, that was happening only on fresh enough versions of Linux kernel. [#3749](#)
- Fixed segfault in function `empty` for argument of `FixedString` type. [Daniel, Dao Quang Minh](#)
- Fixed excessive memory allocation when using large value of `max_query_size` setting (a memory chunk of `max_query_size` bytes was preallocated at once). [#3720](#)

#### Build Changes:

- Fixed build with LLVM/Clang libraries of version 7 from the OS packages (these libraries are used for runtime query compilation). [#3582](#)

### ClickHouse Release 18.14.17, 2018-11-30

#### Bug Fixes:

- Fixed cases when the ODBC bridge process did not terminate with the main server process. [#3642](#)

- Fixed synchronous insertion into the `Distributed` table with a columns list that differs from the column list of the remote table. [#3673](#)
- Fixed a rare race condition that can lead to a crash when dropping a MergeTree table. [#3643](#)
- Fixed a query deadlock in case when query thread creation fails with the `Resource temporarily unavailable` error. [#3643](#)
- Fixed parsing of the `ENGINE` clause when the `CREATE AS table` syntax was used and the `ENGINE` clause was specified before the `AS table` (the error resulted in ignoring the specified engine). [#3692](#)

## ClickHouse Release 18.14.15, 2018-11-21

### Bug Fixes:

- The size of memory chunk was overestimated while deserializing the column of type `Array(String)` that leads to “Memory limit exceeded” errors. The issue appeared in version 18.12.13. [#3589](#)

## ClickHouse Release 18.14.14, 2018-11-20

### Bug Fixes:

- Fixed `ON CLUSTER` queries when cluster configured as secure (flag `<secure>`). [#3599](#)

### Build Changes:

- Fixed problems (Ilvm-7 from system, macos) [#3582](#)

## ClickHouse Release 18.14.13, 2018-11-08

### Bug Fixes:

- Fixed the `Block` structure mismatch in `MergingSorted stream` error. [#3162](#)
- Fixed `ON CLUSTER` queries in case when secure connections were turned on in the cluster config (the `<secure>` flag). [#3465](#)
- Fixed an error in queries that used `SAMPLE`, `PREWHERE` and alias columns. [#3543](#)
- Fixed a rare `unknown compression method` error when the `min_bytes_to_use_direct_io` setting was enabled. [#3544](#)

### Performance Improvements:

- Fixed performance regression of queries with `GROUP BY` of columns of `UInt16` or `Date` type when executing on AMD EPYC processors. [Igor Lapko](#)
- Fixed performance regression of queries that process long strings. [#3530](#)

### Build Improvements:

- Improvements for simplifying the Arcadia build. [#3475](#), [#3535](#)

## ClickHouse Release 18.14.12, 2018-11-02

### Bug Fixes:

- Fixed a crash on joining two unnamed subqueries. [#3505](#)
- Fixed generating incorrect queries (with an empty `WHERE` clause) when querying external databases. [hotid](#)
- Fixed using an incorrect timeout value in ODBC dictionaries. [Marek Vavruša](#)

# ClickHouse Release 18.14.11, 2018-10-29

## Bug Fixes:

- Fixed the error Block structure mismatch in UNION stream: different number of columns in LIMIT queries. [#2156](#)
- Fixed errors when merging data in tables containing arrays inside Nested structures. [#3397](#)
- Fixed incorrect query results if the merge\_tree\_uniform\_read\_distribution setting is disabled (it is enabled by default). [#3429](#)
- Fixed an error on inserts to a Distributed table in Native format. [#3411](#)

# ClickHouse Release 18.14.10, 2018-10-23

- The `compile_expressions` setting (JIT compilation of expressions) is disabled by default. [#3410](#)
- The `enable_optimize_predicate_expression` setting is disabled by default.

# ClickHouse Release 18.14.9, 2018-10-16

## New Features:

- The `WITH CUBE` modifier for `GROUP BY` (the alternative syntax `GROUP BY CUBE(...)` is also available). [#3172](#)
- Added the `formatDateTime` function. [Alexandr Krasheninnikov](#)
- Added the `JDBC` table engine and `jdbc` table function (requires installing `clickhouse-jdbc-bridge`). [Alexandr Krasheninnikov](#)
- Added functions for working with the ISO week number: `toISOWeek`, `toISOYear`, `toStartOfISOYear`, and `toDayOfYear`. [#3146](#)
- Now you can use `Nullable` columns for `MySQL` and `ODBC` tables. [#3362](#)
- Nested data structures can be read as nested objects in `JSONEachRow` format. Added the `input_format_import_nested_json` setting. [Veloman Yunkan](#)
- Parallel processing is available for many `MATERIALIZED VIEWS` when inserting data. See the `parallel_view_processing` setting. [Marek Vavruša](#)
- Added the `SYSTEM FLUSH LOGS` query (forced log flushes to system tables such as `query_log`) [#3321](#)
- Now you can use pre-defined `database` and `table` macros when declaring `Replicated` tables. [#3251](#)
- Added the ability to read `Decimal` type values in engineering notation (indicating powers of ten). [#3153](#)

## Experimental Features:

- Optimization of the `GROUP BY` clause for `LowCardinality` data types. [#3138](#)
- Optimized calculation of expressions for `LowCardinality` data types. [#3200](#)

## Improvements:

- Significantly reduced memory consumption for queries with `ORDER BY` and `LIMIT`. See the `max_bytes_before_remerge_sort` setting. [#3205](#)
- In the absence of `JOIN` (`LEFT`, `INNER`, ...), `INNER JOIN` is assumed. [#3147](#)
- Qualified asterisks work correctly in queries with `JOIN`. [Winter Zhang](#)
- The `ODBC` table engine correctly chooses the method for quoting identifiers in the SQL dialect of a remote database. [Alexandr Krasheninnikov](#)

- The `compile_expressions` setting (JIT compilation of expressions) is enabled by default.
- Fixed behavior for simultaneous `DROP DATABASE/TABLE IF EXISTS` and `CREATE DATABASE/TABLE IF NOT EXISTS`. Previously, a `CREATE DATABASE ... IF NOT EXISTS` query could return the error message “File ... already exists”, and the `CREATE TABLE ... IF NOT EXISTS` and `DROP TABLE IF EXISTS` queries could return `Table ... is creating or attaching right now.` [#3101](#)
- LIKE and IN expressions with a constant right half are passed to the remote server when querying from MySQL or ODBC tables. [#3182](#)
- Comparisons with constant expressions in a WHERE clause are passed to the remote server when querying from MySQL and ODBC tables. Previously, only comparisons with constants were passed. [#3182](#)
- Correct calculation of row width in the terminal for `Pretty` formats, including strings with hieroglyphs. [Amos Bird](#).
- `ON CLUSTER` can be specified for `ALTER UPDATE` queries.
- Improved performance for reading data in `JSONEachRow` format. [#3332](#)
- Added synonyms for the `LENGTH` and `CHARACTER_LENGTH` functions for compatibility. The `CONCAT` function is no longer case-sensitive. [#3306](#)
- Added the `TIMESTAMP` synonym for the `DateTime` type. [#3390](#)
- There is always space reserved for `query_id` in the server logs, even if the log line is not related to a query. This makes it easier to parse server text logs with third-party tools.
- Memory consumption by a query is logged when it exceeds the next level of an integer number of gigabytes. [#3205](#)
- Added compatibility mode for the case when the client library that uses the Native protocol sends fewer columns by mistake than the server expects for the `INSERT` query. This scenario was possible when using the `clickhouse-cpp` library. Previously, this scenario caused the server to crash. [#3171](#)
- In a user-defined WHERE expression in `clickhouse-copier`, you can now use a `partition_key` alias (for additional filtering by source table partition). This is useful if the partitioning scheme changes during copying, but only changes slightly. [#3166](#)
- The workflow of the `Kafka` engine has been moved to a background thread pool in order to automatically reduce the speed of data reading at high loads. [Marek Vavruša](#).
- Support for reading `Tuple` and `Nested` values of structures like `struct` in the `Cap'n'Proto` format. [Marek Vavruša](#)
- The list of top-level domains for the `firstSignificantSubdomain` function now includes the domain `biz`. [decaseal](#)
- In the configuration of external dictionaries, `null_value` is interpreted as the value of the default data type. [#3330](#)
- Support for the `intDiv` and `intDivOrZero` functions for `Decimal`. [b48402e8](#)
- Support for the `Date`, `DateTime`, `UUID`, and `Decimal` types as a key for the `sumMap` aggregate function. [#3281](#)
- Support for the `Decimal` data type in external dictionaries. [#3324](#)
- Support for the `Decimal` data type in `SummingMergeTree` tables. [#3348](#)

- Added specializations for `UUID` in `if`. [#3366](#)
- Reduced the number of `open` and `close` system calls when reading from a `MergeTree` table. [#3283](#)
- A `TRUNCATE TABLE` query can be executed on any replica (the query is passed to the leader replica). [Kirill Shvakov](#)

## Bug Fixes:

- Fixed an issue with `Dictionary` tables for `range_hashed` dictionaries. This error occurred in version 18.12.17. [#1702](#)
- Fixed an error when loading `range_hashed` dictionaries (the message `Unsupported type Nullable (...)`). This error occurred in version 18.12.17. [#3362](#)
- Fixed errors in the `pointInPolygon` function due to the accumulation of inaccurate calculations for polygons with a large number of vertices located close to each other. [#3331](#) [#3341](#)
- If after merging data parts, the checksum for the resulting part differs from the result of the same merge in another replica, the result of the merge is deleted and the data part is downloaded from the other replica (this is the correct behavior). But after downloading the data part, it couldn't be added to the working set because of an error that the part already exists (because the data part was deleted with some delay after the merge). This led to cyclical attempts to download the same data. [#3194](#)
- Fixed incorrect calculation of total memory consumption by queries (because of incorrect calculation, the `max_memory_usage_for_all_queries` setting worked incorrectly and the `MemoryTracking` metric had an incorrect value). This error occurred in version 18.12.13. [Marek Vavruša](#)
- Fixed the functionality of `CREATE TABLE ... ON CLUSTER ... AS SELECT ...`. This error occurred in version 18.12.13. [#3247](#)
- Fixed unnecessary preparation of data structures for `JOINS` on the server that initiates the query if the `JOIN` is only performed on remote servers. [#3340](#)
- Fixed bugs in the `Kafka` engine: deadlocks after exceptions when starting to read data, and locks upon completion [Marek Vavruša](#).
- For `Kafka` tables, the optional `schema` parameter was not passed (the schema of the `Cap'n'Proto` format). [Vojtech Splichal](#)
- If the ensemble of ZooKeeper servers has servers that accept the connection but then immediately close it instead of responding to the handshake, ClickHouse chooses to connect another server. Previously, this produced the error `Cannot read all data. Bytes read: 0. Bytes expected: 4.` and the server couldn't start. [8218cf3a](#)
- If the ensemble of ZooKeeper servers contains servers for which the DNS query returns an error, these servers are ignored. [17b8e209](#)
- Fixed type conversion between `Date` and `DateTime` when inserting data in the `VALUES` format (if `input_format_values_interpret_expressions = 1`). Previously, the conversion was performed between the numerical value of the number of days in Unix Epoch time and the Unix timestamp, which led to unexpected results. [#3229](#)
- Corrected type conversion between `Decimal` and integer numbers. [#3211](#)
- Fixed errors in the `enable_optimize_predicate_expression` setting. [Winter Zhang](#)
- Fixed a parsing error in CSV format with floating-point numbers if a non-default CSV separator is used, such as ; [#3155](#)

- Fixed the `arrayCumSumNonNegative` function (it does not accumulate negative values if the accumulator is less than zero). [Aleksey Studnev](#)
- Fixed how `Merge` tables work on top of `Distributed` tables when using `PREWHERE`. [#3165](#)
- Bug fixes in the `ALTER UPDATE` query.
- Fixed bugs in the `odbc` table function that appeared in version 18.12. [#3197](#)
- Fixed the operation of aggregate functions with `StateArray` combinators. [#3188](#)
- Fixed a crash when dividing a `Decimal` value by zero. [69dd6609](#)
- Fixed output of types for operations using `Decimal` and integer arguments. [#3224](#)
- Fixed the segfault during `GROUP BY` on `Decimal128`. [3359ba06](#)
- The `log_query_threads` setting (logging information about each thread of query execution) now takes effect only if the `log_queries` option (logging information about queries) is set to 1. Since the `log_query_threads` option is enabled by default, information about threads was previously logged even if query logging was disabled. [#3241](#)
- Fixed an error in the distributed operation of the quantiles aggregate function (the error message `Not found column quantile...`). [292a8855](#)
- Fixed the compatibility problem when working on a cluster of version 18.12.17 servers and older servers at the same time. For distributed queries with `GROUP BY` keys of both fixed and non-fixed length, if there was a large amount of data to aggregate, the returned data was not always fully aggregated (two different rows contained the same aggregation keys). [#3254](#)
- Fixed handling of substitutions in `clickhouse-performance-test`, if the query contains only part of the substitutions declared in the test. [#3263](#)
- Fixed an error when using `FINAL` with `PREWHERE`. [#3298](#)
- Fixed an error when using `PREWHERE` over columns that were added during `ALTER`. [#3298](#)
- Added a check for the absence of `arrayJoin` for `DEFAULT` and `MATERIALIZED` expressions. Previously, `arrayJoin` led to an error when inserting data. [#3337](#)
- Added a check for the absence of `arrayJoin` in a `PREWHERE` clause. Previously, this led to messages like `Size ... does not match` or `Unknown compression method` when executing queries. [#3357](#)
- Fixed segfault that could occur in rare cases after optimization that replaced AND chains from equality evaluations with the corresponding IN expression. [liuyimin-bytedance](#)
- Minor corrections to `clickhouse-benchmark`: previously, client information was not sent to the server; now the number of queries executed is calculated more accurately when shutting down and for limiting the number of iterations. [#3351](#) [#3352](#)

#### Backward Incompatible Changes:

- Removed the `allow_experimental_decimal_type` option. The `Decimal` data type is available for default use. [#3329](#)

## ClickHouse Release 18.12

ClickHouse Release 18.12.17, 2018-09-16

New Features:

- `invalidate_query` (the ability to specify a query to check whether an external dictionary needs to be updated) is implemented for the `clickhouse` source. [#3126](#)
- Added the ability to use `UInt*`, `Int*`, and `DateTime` data types (along with the `Date` type) as a `range_hashed` external dictionary key that defines the boundaries of ranges. Now `NULL` can be used to designate an open range. [Vasily Nemkov](#)
- The `Decimal` type now supports `var*` and `stddev*` aggregate functions. [#3129](#)
- The `Decimal` type now supports mathematical functions (`exp`, `sin` and so on.) [#3129](#)
- The `system.part_log` table now has the `partition_id` column. [#3089](#)

#### Bug Fixes:

- Merge now works correctly on `Distributed` tables. [Winter Zhang](#)
- Fixed incompatibility (unnecessary dependency on the `glibc` version) that made it impossible to run ClickHouse on `Ubuntu Precise` and older versions. The incompatibility arose in version 18.12.13. [#3130](#)
- Fixed errors in the `enable_optimize_predicate_expression` setting. [Winter Zhang](#)
- Fixed a minor issue with backwards compatibility that appeared when working with a cluster of replicas on versions earlier than 18.12.13 and simultaneously creating a new replica of a table on a server with a newer version (shown in the message `Can not clone replica, because the ...` updated to new ClickHouse version which is logical, but shouldn't happen). [#3122](#)

#### Backward Incompatible Changes:

- The `enable_optimize_predicate_expression` option is enabled by default (which is rather optimistic). If query analysis errors occur that are related to searching for the column names, set `enable_optimize_predicate_expression` to 0. [Winter Zhang](#)

## ClickHouse Release 18.12.14, 2018-09-13

#### New Features:

- Added support for `ALTER UPDATE` queries. [#3035](#)
- Added the `allow_ddl` option, which restricts the user's access to DDL queries. [#3104](#)
- Added the `min_merge_bytes_to_use_direct_io` option for `MergeTree` engines, which allows you to set a threshold for the total size of the merge (when above the threshold, data part files will be handled using `O_DIRECT`). [#3117](#)
- The `system.merges` system table now contains the `partition_id` column. [#3099](#)

#### Improvements

- If a data part remains unchanged during mutation, it isn't downloaded by replicas. [#3103](#)
- Autocomplete is available for names of settings when working with `clickhouse-client`. [#3106](#)

#### Bug Fixes:

- Added a check for the sizes of arrays that are elements of `Nested` type fields when inserting. [#3118](#)
- Fixed an error updating external dictionaries with the `ODBC` source and `hashed` storage. This error occurred in version 18.12.13.
- Fixed a crash when creating a temporary table from a query with an `IN` condition. [Winter Zhang](#)
- Fixed an error in aggregate functions for arrays that can have `NULL` elements. [Winter Zhang](#)

# ClickHouse Release 18.12.13, 2018-09-10

## New Features:

- Added the `DECIMAL(digits, scale)` data type (`Decimal32(scale)`, `Decimal64(scale)`, `Decimal128(scale)`). To enable it, use the setting `allow_experimental_decimal_type`. [#2846](#) [#2970](#) [#3008](#) [#3047](#)
- New `WITH ROLLUP` modifier for `GROUP BY` (alternative syntax: `GROUP BY ROLLUP(...)`). [#2948](#)
- In queries with `JOIN`, the star character expands to a list of columns in all tables, in compliance with the SQL standard. You can restore the old behavior by setting `asterisk_left_columns_only` to 1 on the user configuration level. [Winter Zhang](#)
- Added support for `JOIN` with table functions. [Winter Zhang](#)
- Autocomplete by pressing Tab in clickhouse-client. [Sergey Shcherbin](#)
- Ctrl+C in clickhouse-client clears a query that was entered. [#2877](#)
- Added the `join_default_strictness` setting (values: "", "any", "all"). This allows you to not specify `ANY` or `ALL` for `JOIN`. [#2982](#)
- Each line of the server log related to query processing shows the query ID. [#2482](#)
- Now you can get query execution logs in clickhouse-client (use the `send_logs_level` setting). With distributed query processing, logs are cascaded from all the servers. [#2482](#)
- The `system.query_log` and `system.processes` (`SHOW PROCESSLIST`) tables now have information about all changed settings when you run a query (the nested structure of the `Settings` data). Added the `log_query_settings` setting. [#2482](#)
- The `system.query_log` and `system.processes` tables now show information about the number of threads that are participating in query execution (see the `thread_numbers` column). [#2482](#)
- Added `ProfileEvents` counters that measure the time spent on reading and writing over the network and reading and writing to disk, the number of network errors, and the time spent waiting when network bandwidth is limited. [#2482](#)
- Added `ProfileEvents` counters that contain the system metrics from `rusage` (you can use them to get information about CPU usage in userspace and the kernel, page faults, and context switches), as well as `taskstats` metrics (use these to obtain information about I/O wait time, CPU wait time, and the amount of data read and recorded, both with and without page cache). [#2482](#)
- The `ProfileEvents` counters are applied globally and for each query, as well as for each query execution thread, which allows you to profile resource consumption by query in detail. [#2482](#)
- Added the `system.query_thread_log` table, which contains information about each query execution thread. Added the `log_query_threads` setting. [#2482](#)
- The `system.metrics` and `system.events` tables now have built-in documentation. [#3016](#)
- Added the `arrayEnumerateDense` function. [Amos Bird](#)
- Added the `arrayCumSumNonNegative` and `arrayDifference` functions. [Aleksey Studnev](#)
- Added the `retention` aggregate function. [Sundy Li](#)
- Now you can add (merge) states of aggregate functions by using the plus operator, and multiply the states of aggregate functions by a nonnegative constant. [#3062](#) [#3034](#)
- Tables in the MergeTree family now have the virtual column `_partition_id`. [#3089](#)

## Experimental Features:

- Added the `LowCardinality(T)` data type. This data type automatically creates a local dictionary of values and allows data processing without unpacking the dictionary. [#2830](#)
- Added a cache of JIT-compiled functions and a counter for the number of uses before compiling. To JIT compile expressions, enable the `compile_expressions` setting. [#2990](#) [#3077](#)

## Improvements:

- Fixed the problem with unlimited accumulation of the replication log when there are abandoned replicas. Added an effective recovery mode for replicas with a long lag.
- Improved performance of `GROUP BY` with multiple aggregation fields when one of them is string and the others are fixed length.
- Improved performance when using `PREWHERE` and with implicit transfer of expressions in `PREWHERE`.
- Improved parsing performance for text formats (CSV, TSV). [Amos Bird](#) [#2980](#)
- Improved performance of reading strings and arrays in binary formats. [Amos Bird](#)
- Increased performance and reduced memory consumption for queries to `system.tables` and `system.columns` when there is a very large number of tables on a single server. [#2953](#)
- Fixed a performance problem in the case of a large stream of queries that result in an error (the `_dl_addr` function is visible in `perf top`, but the server isn't using much CPU). [#2938](#)
- Conditions are cast into the View (when `enable_optimize_predicate_expression` is enabled). [Winter Zhang](#)
- Improvements to the functionality for the `UUID` data type. [#3074](#) [#2985](#)
- The `UUID` data type is supported in The-Alchemist dictionaries. [#2822](#)
- The `visitParamExtractRaw` function works correctly with nested structures. [Winter Zhang](#)
- When the `input_format_skip_unknown_fields` setting is enabled, object fields in `JSONEachRow` format are skipped correctly. [BlahGeek](#)
- For a `CASE` expression with conditions, you can now omit `ELSE`, which is equivalent to `ELSE NULL`. [#2920](#)
- The operation timeout can now be configured when working with ZooKeeper. [urykhy](#)
- You can specify an offset for `LIMIT n, m` as `LIMIT n OFFSET m`. [#2840](#)
- You can use the `SELECT TOP n` syntax as an alternative for `LIMIT`. [#2840](#)
- Increased the size of the queue to write to system tables, so the `SystemLog` parameter `queue` is full error does not happen as often.
- The `windowFunnel` aggregate function now supports events that meet multiple conditions. [Amos Bird](#)
- Duplicate columns can be used in a `USING` clause for `JOIN`. [#3006](#)
- Pretty formats now have a limit on column alignment by width. Use the `output_format_pretty_max_column_pad_width` setting. If a value is wider, it will still be displayed in its entirety, but the other cells in the table will not be too wide. [#3003](#)
- The `odbc` table function now allows you to specify the database/schema name. [Amos Bird](#)
- Added the ability to use a username specified in the `clickhouse-client` config file. [Vladimir Kozbin](#)

- The `ZooKeeperExceptions` counter has been split into three counters: `ZooKeeperUserExceptions`, `ZooKeeperHardwareExceptions`, and `ZooKeeperOtherExceptions`.
- `ALTER DELETE` queries work for materialized views.
- Added randomization when running the cleanup thread periodically for `ReplicatedMergeTree` tables in order to avoid periodic load spikes when there are a very large number of `ReplicatedMergeTree` tables.
- Support for `ATTACH TABLE ... ON CLUSTER` queries. [#3025](#)

## Bug Fixes:

- Fixed an issue with `Dictionary` tables (throws the `Size of offsets does not match size of column or Unknown compression method` exception). This bug appeared in version 18.10.3. [#2913](#)
- Fixed a bug when merging `CollapsingMergeTree` tables if one of the data parts is empty (these parts are formed during merge or `ALTER DELETE` if all data was deleted), and the `vertical` algorithm was used for the merge. [#3049](#)
- Fixed a race condition during `DROP` or `TRUNCATE` for `Memory` tables with a simultaneous `SELECT`, which could lead to server crashes. This bug appeared in version 1.1.54388. [#3038](#)
- Fixed the possibility of data loss when inserting in `Replicated` tables if the `Session is expired` error is returned (data loss can be detected by the `ReplicatedDataLoss` metric). This error occurred in version 1.1.54378. [#2939](#) [#2949](#) [#2964](#)
- Fixed a segfault during `JOIN ... ON`. [#3000](#)
- Fixed the error searching column names when the `WHERE` expression consists entirely of a qualified column name, such as `WHERE table.column`. [#2994](#)
- Fixed the “Not found column” error that occurred when executing distributed queries if a single column consisting of an `IN` expression with a subquery is requested from a remote server. [#3087](#)
- Fixed the `Block structure mismatch in UNION stream: different number of columns` error that occurred for distributed queries if one of the shards is local and the other is not, and optimization of the move to `PREWHERE` is triggered. [#2226](#) [#3037](#) [#3055](#) [#3065](#) [#3073](#) [#3090](#) [#3093](#)
- Fixed the `pointInPolygon` function for certain cases of non-convex polygons. [#2910](#)
- Fixed the incorrect result when comparing `nan` with integers. [#3024](#)
- Fixed an error in the `zlib-ng` library that could lead to segfault in rare cases. [#2854](#)
- Fixed a memory leak when inserting into a table with `AggregateFunction` columns, if the state of the aggregate function is not simple (allocates memory separately), and if a single insertion request results in multiple small blocks. [#3084](#)
- Fixed a race condition when creating and deleting the same `Buffer` or `MergeTree` table simultaneously.
- Fixed the possibility of a segfault when comparing tuples made up of certain non-trivial types, such as tuples. [#2989](#)
- Fixed the possibility of a segfault when running certain `ON CLUSTER` queries. [Winter Zhang](#)
- Fixed an error in the `arrayDistinct` function for `Nullable` array elements. [#2845](#) [#2937](#)
- The `enable_optimize_predicate_expression` option now correctly supports cases with `SELECT *`. [Winter Zhang](#)
- Fixed the segfault when re-initializing the ZooKeeper session. [#2917](#)
- Fixed potential blocking when working with ZooKeeper.

- Fixed incorrect code for adding nested data structures in a `SummingMergeTree`.
- When allocating memory for states of aggregate functions, alignment is correctly taken into account, which makes it possible to use operations that require alignment when implementing states of aggregate functions. [chenxing-xc](#)

#### Security Fix:

- Safe use of ODBC data sources. Interaction with ODBC drivers uses a separate `clickhouse-odbc-bridge` process. Errors in third-party ODBC drivers no longer cause problems with server stability or vulnerabilities. [#2828](#) [#2879](#) [#2886](#) [#2893](#) [#2921](#)
- Fixed incorrect validation of the file path in the `catBoostPool` table function. [#2894](#)
- The contents of system tables (`tables`, `databases`, `parts`, `columns`, `parts_columns`, `merges`, `mutations`, `replicas`, and `replication_queue`) are filtered according to the user's configured access to databases (`allow_databases`). [Winter Zhang](#)

#### Backward Incompatible Changes:

- In queries with `JOIN`, the star character expands to a list of columns in all tables, in compliance with the SQL standard. You can restore the old behavior by setting `asterisk_left_columns_only` to 1 on the user configuration level.

#### Build Changes:

- Most integration tests can now be run by commit.
- Code style checks can also be run by commit.
- The `memcpy` implementation is chosen correctly when building on CentOS7/Fedora. [Etienne Champetier](#)
- When using clang to build, some warnings from `-Weverything` have been added, in addition to the regular `-Wall-Wextra -Werror`. [#2957](#)
- Debugging the build uses the `jemalloc` debug option.
- The interface of the library for interacting with ZooKeeper is declared abstract. [#2950](#)

## ClickHouse Release 18.10

### ClickHouse Release 18.10.3, 2018-08-13

#### New Features:

- HTTPS can be used for replication. [#2760](#)
- Added the functions `murmurHash2_64`, `murmurHash3_32`, `murmurHash3_64`, and `murmurHash3_128` in addition to the existing `murmurHash2_32`. [#2791](#)
- Support for Nullable types in the ClickHouse ODBC driver (ODBCDriver2 output format). [#2834](#)
- Support for `UUID` in the key columns.

#### Improvements:

- Clusters can be removed without restarting the server when they are deleted from the config files. [#2777](#)
- External dictionaries can be removed without restarting the server when they are removed from config files. [#2779](#)
- Added `SETTINGS` support for the `Kafka` table engine. [Alexander Marshalov](#)

- Improvements for the `UUID` data type (not yet complete). [#2618](#)
- Support for empty parts after merges in the `SummingMergeTree`, `CollapsingMergeTree` and `VersionedCollapsingMergeTree` engines. [#2815](#)
- Old records of completed mutations are deleted (`ALTER DELETE`). [#2784](#)
- Added the `system.merge_tree_settings` table. [Kirill Shvakov](#)
- The `system.tables` table now has dependency columns: `dependencies_database` and `dependencies_table`. [Winter Zhang](#)
- Added the `max_partition_size_to_drop` config option. [#2782](#)
- Added the `output_format_json_escape_forward_slashes` option. [Alexander Bocharov](#)
- Added the `max_fetch_partition_retries_count` setting. [#2831](#)
- Added the `prefer_localhost_replica` setting for disabling the preference for a local replica and going to a local replica without inter-process interaction. [#2832](#)
- The `quantileExact` aggregate function returns `nan` in the case of aggregation on an empty `Float32` or `Float64` set. [Sundy Li](#)

## Bug Fixes:

- Removed unnecessary escaping of the connection string parameters for ODBC, which made it impossible to establish a connection. This error occurred in version 18.6.0.
- Fixed the logic for processing `REPLACE PARTITION` commands in the replication queue. If there are two `REPLACE` commands for the same partition, the incorrect logic could cause one of them to remain in the replication queue and not be executed. [#2814](#)
- Fixed a merge bug when all data parts were empty (parts that were formed from a merge or from `ALTER DELETE` if all data was deleted). This bug appeared in version 18.1.0. [#2930](#)
- Fixed an error for concurrent `Set` or `Join`. [Amos Bird](#)
- Fixed the `Block structure mismatch in UNION stream: different number of columns` error that occurred for `UNION ALL` queries inside a sub-query if one of the `SELECT` queries contains duplicate column names. [Winter Zhang](#)
- Fixed a memory leak if an exception occurred when connecting to a MySQL server.
- Fixed incorrect clickhouse-client response code in case of a query error.
- Fixed incorrect behavior of materialized views containing `DISTINCT`. [#2795](#)

## Backward Incompatible Changes

- Removed support for `CHECK TABLE` queries for Distributed tables.

## Build Changes:

- The allocator has been replaced: `jemalloc` is now used instead of `tcmalloc`. In some scenarios, this increases speed up to 20%. However, there are queries that have slowed by up to 20%. Memory consumption has been reduced by approximately 10% in some scenarios, with improved stability. With highly competitive loads, CPU usage in userspace and in system shows just a slight increase. [#2773](#)
- Use of `libressl` from a submodule. [#1983 #2807](#)
- Use of `unixodbc` from a submodule. [#2789](#)

- Use of mariadb-connector-c from a submodule. [#2785](#)
- Added functional test files to the repository that depend on the availability of test data (for the time being, without the test data itself).

## ClickHouse Release 18.6

### ClickHouse Release 18.6.0, 2018-08-02

#### New Features:

- Added support for ON expressions for the JOIN ON syntax:

```
JOIN ON Expr([table.]column ...) = Expr([table.]column, ...) [AND Expr([table.]column, ...) = Expr([table.]column, ...) ...]  
The expression must be a chain of equalities joined by the AND operator. Each side of the equality can  
be an arbitrary expression over the columns of one of the tables. The use of fully qualified column  
names is supported (table.name, database.table.name, table_alias.name, subquery_alias.name) for the right  
table. #2742
```

- HTTPS can be enabled for replication. [#2760](#)

#### Improvements:

- The server passes the patch component of its version to the client. Data about the patch version  
component is in `system.processes` and `query_log`. [#2646](#)

## ClickHouse Release 18.5

### ClickHouse Release 18.5.1, 2018-07-31

#### New Features:

- Added the hash function `murmurHash2_32` [#2756](#).

#### Improvements:

- Now you can use the `from_env` [#2741](#) attribute to set values in config files from environment variables.
- Added case-insensitive versions of the `coalesce`, `ifNull`, and `nullIf` functions [#2752](#).

#### Bug Fixes:

- Fixed a possible bug when starting a replica [#2759](#).

## ClickHouse Release 18.4

### ClickHouse Release 18.4.0, 2018-07-28

#### New Features:

- Added system tables: `formats`, `data_type_families`, `aggregate_function_combinators`, `table_functions`, `table_engines`, `collations` [#2721](#).
- Added the ability to use a table function instead of a table as an argument of a `remote` or `cluster` table  
function [#2708](#).
- Support for HTTP Basic authentication in the replication protocol [#2727](#).
- The `has` function now allows searching for a numeric value in an array of `Enum` values [Maxim Khrisanfov](#).
- Support for adding arbitrary message separators when reading from Kafka [Amos Bird](#).

#### Improvements:

- The `ALTER TABLE t DELETE WHERE` query does not rewrite data parts that were not affected by the `WHERE` condition [#2694](#).
- The `use_minimalistic_checksums_in_zookeeper` option for `ReplicatedMergeTree` tables is enabled by default. This setting was added in version 1.1.54378, 2018-04-16. Versions that are older than 1.1.54378 can no longer be installed.
- Support for running `KILL` and `OPTIMIZE` queries that specify `ON CLUSTER` [Winter Zhang](#).

#### Bug Fixes:

- Fixed the error `Column ... is not under an aggregate function and not in GROUP BY` for aggregation with an `IN` expression. This bug appeared in version 18.1.0. ([bbdd780b](#))
- Fixed a bug in the `windowFunnel` aggregate function [Winter Zhang](#).
- Fixed a bug in the `anyHeavy` aggregate function ([a2101df2](#))
- Fixed server crash when using the `countArray()` aggregate function.

#### Backward Incompatible Changes:

- Parameters for `Kafka` engine was changed from `Kafka(kafka_broker_list, kafka_topic_list, kafka_group_name, kafka_format[, kafka_schema, kafka_num_consumers])` to `Kafka(kafka_broker_list, kafka_topic_list, kafka_group_name, kafka_format[, kafka_row_delimiter, kafka_schema, kafka_num_consumers])`. If your tables use `kafka_schema` or `kafka_num_consumers` parameters, you have to manually edit the metadata files `path/metadata/database/table.sql` and add `kafka_row_delimiter` parameter with `" "` value.

## ClickHouse Release 18.1

### ClickHouse Release 18.1.0, 2018-07-23

#### New Features:

- Support for the `ALTER TABLE t DELETE WHERE` query for non-replicated `MergeTree` tables ([#2634](#)).
- Support for arbitrary types for the `uniq*` family of aggregate functions ([#2010](#)).
- Support for arbitrary types in comparison operators ([#2026](#)).
- The `users.xml` file allows setting a subnet mask in the format `10.0.0.1/255.255.255.0`. This is necessary for using masks for IPv6 networks with zeros in the middle ([#2637](#)).
- Added the `arrayDistinct` function ([#2670](#)).
- The `SummingMergeTree` engine can now work with `AggregateFunction` type columns ([Constantin S. Pan](#)).

#### Improvements:

- Changed the numbering scheme for release versions. Now the first part contains the year of release (A.D., Moscow timezone, minus 2000), the second part contains the number for major changes (increases for most releases), and the third part is the patch version. Releases are still backward compatible, unless otherwise stated in the changelog.
- Faster conversions of floating-point numbers to a string ([Amos Bird](#)).
- If some rows were skipped during an insert due to parsing errors (this is possible with the `input_allow_errors_num` and `input_allow_errors_ratio` settings enabled), the number of skipped rows is now written to the server log ([Leonardo Cecchi](#)).

#### Bug Fixes:

- Fixed the TRUNCATE command for temporary tables ([Amos Bird](#)).
- Fixed a rare deadlock in the ZooKeeper client library that occurred when there was a network error while reading the response ([c315200](#)).
- Fixed an error during a CAST to Nullable types ([#1322](#)).
- Fixed the incorrect result of the `maxIntersection()` function when the boundaries of intervals coincided ([Michael Furmur](#)).
- Fixed incorrect transformation of the OR expression chain in a function argument ([chenxing-xc](#)).
- Fixed performance degradation for queries containing `IN (subquery)` expressions inside another subquery ([#2571](#)).
- Fixed incompatibility between servers with different versions in distributed queries that use a `CAST` function that isn't in uppercase letters ([fe8c4d6](#)).
- Added missing quoting of identifiers for queries to an external DBMS ([#2635](#)).

#### Backward Incompatible Changes:

- Converting a string containing the number zero to `DateTime` does not work. Example: `SELECT toDateTime('0')`. This is also the reason that `DateTime DEFAULT '0'` does not work in tables, as well as `<null_value>0</null_value>` in dictionaries. Solution: replace 0 with `0000-00-00 00:00:00`.

## ClickHouse Release 1.1

### ClickHouse Release 1.1.54394, 2018-07-12

#### New Features:

- Added the `histogram` aggregate function ([Mikhail Surin](#)).
- Now `OPTIMIZE TABLE ... FINAL` can be used without specifying partitions for `ReplicatedMergeTree` ([Amos Bird](#)).

#### Bug Fixes:

- Fixed a problem with a very small timeout for sockets (one second) for reading and writing when sending and downloading replicated data, which made it impossible to download larger parts if there is a load on the network or disk (it resulted in cyclical attempts to download parts). This error occurred in version 1.1.54388.
- Fixed issues when using chroot in ZooKeeper if you inserted duplicate data blocks in the table.
- The `has` function now works correctly for an array with Nullable elements ([#2115](#)).
- The `system.tables` table now works correctly when used in distributed queries. The `metadata_modification_time` and `engine_full` columns are now non-virtual. Fixed an error that occurred if only these columns were queried from the table.
- Fixed how an empty `TinyLog` table works after inserting an empty data block ([#2563](#)).
- The `system.zookeeper` table works if the value of the node in ZooKeeper is `NUL`.

### ClickHouse Release 1.1.54390, 2018-07-06

#### New Features:

- Queries can be sent in `multipart/form-data` format (in the `query` field), which is useful if external data is also sent for query processing ([Olga Hvostikova](#)).

- Added the ability to enable or disable processing single or double quotes when reading data in CSV format. You can configure this in the `format_csv_allow_single_quotes` and `format_csv_allow_double_quotes` settings ([Amos Bird](#)).
- Now `OPTIMIZE TABLE ... FINAL` can be used without specifying the partition for non-replicated variants of `MergeTree` ([Amos Bird](#)).

## Improvements:

- Improved performance, reduced memory consumption, and correct memory consumption tracking with use of the IN operator when a table index could be used ([#2584](#)).
- Removed redundant checking of checksums when adding a data part. This is important when there are a large number of replicas, because in these cases the total number of checks was equal to  $N^2$ .
- Added support for `Array(Tuple(...))` arguments for the `arrayEnumerateUniq` function ([#2573](#)).
- Added `Nullable` support for the `runningDifference` function ([#2594](#)).
- Improved query analysis performance when there is a very large number of expressions ([#2572](#)).
- Faster selection of data parts for merging in `ReplicatedMergeTree` tables. Faster recovery of the ZooKeeper session ([#2597](#)).
- The `format_version.txt` file for `MergeTree` tables is re-created if it is missing, which makes sense if ClickHouse is launched after copying the directory structure without files ([Ciprian Hacman](#)).

## Bug Fixes:

- Fixed a bug when working with ZooKeeper that could make it impossible to recover the session and readonly states of tables before restarting the server.
- Fixed a bug when working with ZooKeeper that could result in old nodes not being deleted if the session is interrupted.
- Fixed an error in the `quantileTDigest` function for Float arguments (this bug was introduced in version 1.1.54388) ([Mikhail Surin](#)).
- Fixed a bug in the index for MergeTree tables if the primary key column is located inside the function for converting types between signed and unsigned integers of the same size ([#2603](#)).
- Fixed segfault if `macros` are used but they aren't in the config file ([#2570](#)).
- Fixed switching to the default database when reconnecting the client ([#2583](#)).
- Fixed a bug that occurred when the `use_index_for_in_with_subqueries` setting was disabled.

## Security Fix:

- Sending files is no longer possible when connected to MySQL (`LOAD DATA LOCAL INFILE`).

## ClickHouse Release 1.1.54388, 2018-06-28

### New Features:

- Support for the `ALTER TABLE t DELETE WHERE` query for replicated tables. Added the `system.mutations` table to track progress of this type of queries.
- Support for the `ALTER TABLE t [REPLACE|ATTACH] PARTITION` query for \*MergeTree tables.
- Support for the `TRUNCATE TABLE` query ([Winter Zhang](#))

- Several new `SYSTEM` queries for replicated tables (`RESTART REPLICAS`, `SYNC REPLICA`, `[STOP|START]` `[MERGES|FETCHES|SENDS REPLICATED|REPLICATION QUEUES]`).
- Added the ability to write to a table with the MySQL engine and the corresponding table function ([sundy-li](#)).
- Added the `url()` table function and the `URL` table engine ([Alexander Sapin](#)).
- Added the `windowFunnel` aggregate function ([sundy-li](#)).
- New `startsWith` and `endsWith` functions for strings ([Vadim Plakhtinsky](#)).
- The `numbers()` table function now allows you to specify the offset ([Winter Zhang](#)).
- The password to `clickhouse-client` can be entered interactively.
- Server logs can now be sent to syslog ([Alexander Krasheninnikov](#)).
- Support for logging in dictionaries with a shared library source ([Alexander Sapin](#)).
- Support for custom CSV delimiters ([Ivan Zhukov](#))
- Added the `date_time_input_format` setting. If you switch this setting to 'best\_effort', DateTime values will be read in a wide range of formats.
- Added the `clickhouse-obfuscator` utility for data obfuscation. Usage example: publishing data used in performance tests.

## Experimental Features:

- Added the ability to calculate `and` arguments only where they are needed ([Anastasia Tsarkova](#))
- JIT compilation to native code is now available for some expressions ([pyos](#)).

## Bug Fixes:

- Duplicates no longer appear for a query with `DISTINCT` and `ORDER BY`.
- Queries with `ARRAY JOIN` and `arrayFilter` no longer return an incorrect result.
- Fixed an error when reading an array column from a Nested structure ([#2066](#)).
- Fixed an error when analyzing queries with a `HAVING` clause like `HAVING tuple IN (...)`.
- Fixed an error when analyzing queries with recursive aliases.
- Fixed an error when reading from `ReplacingMergeTree` with a condition in `PREWHERE` that filters all rows ([#2525](#)).
- User profile settings were not applied when using sessions in the HTTP interface.
- Fixed how settings are applied from the command line parameters in `clickhouse-local`.
- The ZooKeeper client library now uses the session timeout received from the server.
- Fixed a bug in the ZooKeeper client library when the client waited for the server response longer than the timeout.
- Fixed pruning of parts for queries with conditions on partition key columns ([#2342](#)).
- Merges are now possible after `CLEAR COLUMN IN PARTITION` ([#2315](#)).
- Type mapping in the ODBC table function has been fixed ([sundy-li](#)).

- Type comparisons have been fixed for `DateTime` with and without the time zone ([Alexander Bocharov](#)).
- Fixed syntactic parsing and formatting of the `CAST` operator.
- Fixed insertion into a materialized view for the Distributed table engine ([Babacar Diassé](#)).
- Fixed a race condition when writing data from the `Kafka` engine to materialized views ([Yangkuan Liu](#)).
- Fixed SSRF in the `remote()` table function.
- Fixed exit behavior of `clickhouse-client` in multiline mode ([#2510](#)).

## Improvements:

- Background tasks in replicated tables are now performed in a thread pool instead of in separate threads ([Silviu Caragea](#)).
- Improved LZ4 compression performance.
- Faster analysis for queries with a large number of JOINs and sub-queries.
- The DNS cache is now updated automatically when there are too many network errors.
- Table inserts no longer occur if the insert into one of the materialized views is not possible because it has too many parts.
- Corrected the discrepancy in the event counters `Query`, `SelectQuery`, and `InsertQuery`.
- Expressions like `tuple IN (SELECT tuple)` are allowed if the tuple types match.
- A server with replicated tables can start even if you haven't configured ZooKeeper.
- When calculating the number of available CPU cores, limits on cgroups are now taken into account ([Atri Sharma](#)).
- Added chown for config directories in the systemd config file ([Mikhail Shiryaev](#)).

## Build Changes:

- The `gcc8` compiler can be used for builds.
- Added the ability to build `llvm` from submodule.
- The version of the `librdkafka` library has been updated to v0.11.4.
- Added the ability to use the system `libcpuid` library. The library version has been updated to 0.4.0.
- Fixed the build using the `vectorclass` library ([Babacar Diassé](#)).
- Cmake now generates files for `ninja` by default (like when using `-G Ninja`).
- Added the ability to use the `libtinfo` library instead of `libtermcap` ([Georgy Kondratiev](#)).
- Fixed a header file conflict in Fedora Rawhide ([#2520](#)).

## Backward Incompatible Changes:

- Removed escaping in `Vertical` and `Pretty*` formats and deleted the `VerticalRaw` format.
- If servers with version 1.1.54388 (or newer) and servers with an older version are used simultaneously in a distributed query and the query has the `cast(x, 'Type')` expression without the `AS` keyword and does not have the word `cast` in uppercase, an exception will be thrown with a message like `Not found column cast(0, 'UInt8') in block`. Solution: Update the server on the entire cluster.

## ClickHouse Release 1.1.54385, 2018-06-01

### Bug Fixes:

- Fixed an error that in some cases caused ZooKeeper operations to block.

## ClickHouse Release 1.1.54383, 2018-05-22

### Bug Fixes:

- Fixed a slowdown of replication queue if a table has many replicas.

## ClickHouse Release 1.1.54381, 2018-05-14

### Bug Fixes:

- Fixed a nodes leak in ZooKeeper when ClickHouse loses connection to ZooKeeper server.

## ClickHouse Release 1.1.54380, 2018-04-21

### New Features:

- Added the table function `file(path, format, structure)`. An example reading bytes from `/dev/urandom`:  
`In -s /dev/urandom /var/lib/clickhouse/user_files/random``clickhouse-client -q "SELECT * FROM file('random', 'RowBinary', 'd UInt8') LIMIT 10"`.

### Improvements:

- Subqueries can be wrapped in `()` brackets to enhance query readability. For example: `(SELECT 1) UNION ALL (SELECT 1)`.
- Simple `SELECT` queries from the `system.processes` table are not included in the `max_concurrent_queries` limit.

### Bug Fixes:

- Fixed incorrect behavior of the `IN` operator when select from `MATERIALIZED VIEW`.
- Fixed incorrect filtering by partition index in expressions like `partition_key_column IN (...)`.
- Fixed inability to execute `OPTIMIZE` query on non-leader replica if `RENAME` was performed on the table.
- Fixed the authorization error when executing `OPTIMIZE` or `ALTER` queries on a non-leader replica.
- Fixed freezing of `KILL QUERY`.
- Fixed an error in ZooKeeper client library which led to loss of watches, freezing of distributed DDL queue, and slowdowns in the replication queue if a non-empty `chroot` prefix is used in the ZooKeeper configuration.

### Backward Incompatible Changes:

- Removed support for expressions like `(a, b) IN (SELECT (a, b))` (you can use the equivalent expression `(a, b) IN (SELECT a, b)`). In previous releases, these expressions led to undetermined `WHERE` filtering or caused errors.

## ClickHouse Release 1.1.54378, 2018-04-16

### New Features:

- Logging level can be changed without restarting the server.
- Added the `SHOW CREATE DATABASE` query.
- The `query_id` can be passed to `clickhouse-client` (`elBroom`).

- New setting: `max_network_bandwidth_for_all_users`.
- Added support for `ALTER TABLE ... PARTITION ...` for `MATERIALIZED VIEW`.
- Added information about the size of data parts in uncompressed form in the system table.
- Server-to-server encryption support for distributed tables (`<secure>1</secure>` in the replica config in `<remote_servers>`).
- Configuration of the table level for the `ReplicatedMergeTree` family in order to minimize the amount of data stored in Zookeeper: : `use_minimalistic_checksums_in_zookeeper = 1`
- Configuration of the `clickhouse-client` prompt. By default, server names are now output to the prompt. The server's display name can be changed. It's also sent in the `X-ClickHouse-Display-Name` HTTP header (Kirill Shvakov).
- Multiple comma-separated `topics` can be specified for the `Kafka` engine (Tobias Adamson)
- When a query is stopped by `KILL QUERY` or `replace_running_query`, the client receives the `Query was canceled` exception instead of an incomplete result.

### Improvements:

- `ALTER TABLE ... DROP/DETACH PARTITION` queries are run at the front of the replication queue.
- `SELECT ... FINAL` and `OPTIMIZE ... FINAL` can be used even when the table has a single data part.
- A `query_log` table is recreated on the fly if it was deleted manually (Kirill Shvakov).
- The `lengthUTF8` function runs faster (zhang2014).
- Improved performance of synchronous inserts in `Distributed` tables (`insert_distributed_sync = 1`) when there is a very large number of shards.
- The server accepts the `send_timeout` and `receive_timeout` settings from the client and applies them when connecting to the client (they are applied in reverse order: the server socket's `send_timeout` is set to the `receive_timeout` value received from the client, and vice versa).
- More robust crash recovery for asynchronous insertion into `Distributed` tables.
- The return type of the `countEqual` function changed from `UInt32` to `UInt64` (谢磊).

### Bug Fixes:

- Fixed an error with `IN` when the left side of the expression is `Nullable`.
- Correct results are now returned when using tuples with `IN` when some of the tuple components are in the table index.
- The `max_execution_time` limit now works correctly with distributed queries.
- Fixed errors when calculating the size of composite columns in the `system.columns` table.
- Fixed an error when creating a temporary table `CREATE TEMPORARY TABLE IF NOT EXISTS`.
- Fixed errors in `StorageKafka` (#2075)
- Fixed server crashes from invalid arguments of certain aggregate functions.
- Fixed the error that prevented the `DETACH DATABASE` query from stopping background tasks for `ReplicatedMergeTree` tables.

- Too many parts state is less likely to happen when inserting into aggregated materialized views (#2084).
- Corrected recursive handling of substitutions in the config if a substitution must be followed by another substitution on the same level.
- Corrected the syntax in the metadata file when creating a `VIEW` that uses a query with `UNION ALL`.
- SummingMergeTree now works correctly for summation of nested data structures with a composite key.
- Fixed the possibility of a race condition when choosing the leader for ReplicatedMergeTree tables.

#### Build Changes:

- The build supports `ninja` instead of `make` and uses `ninja` by default for building releases.
- Renamed packages: `clickhouse-server-base` in `clickhouse-common-static`; `clickhouse-server-common` in `clickhouse-server`; `clickhouse-common-dbg` in `clickhouse-common-static-dbg`. To install, use `clickhouse-server` `clickhouse-client`. Packages with the old names will still load in the repositories for backward compatibility.

#### Backward Incompatible Changes:

- Removed the special interpretation of an IN expression if an array is specified on the left side. Previously, the expression `arr IN (set)` was interpreted as “at least one `arr` element belongs to the `set`”. To get the same behavior in the new version, write `arrayExists(x -> x IN (set), arr)`.
- Disabled the incorrect use of the socket option `SO_REUSEPORT`, which was incorrectly enabled by default in the Poco library. Note that on Linux there is no longer any reason to simultaneously specify the addresses `::` and `0.0.0.0` for listen – use just `::`, which allows listening to the connection both over IPv4 and IPv6 (with the default kernel config settings). You can also revert to the behavior from previous versions by specifying `<listen_reuse_port>1</listen_reuse_port>` in the config.

## ClickHouse Release 1.1.54370, 2018-03-16

#### New Features:

- Added the `system.macros` table and auto updating of macros when the config file is changed.
- Added the `SYSTEM RELOAD CONFIG` query.
- Added the `maxIntersections(left_col, right_col)` aggregate function, which returns the maximum number of simultaneously intersecting intervals `[left; right]`. The `maxIntersectionsPosition(left, right)` function returns the beginning of the “maximum” interval. ([Michael Furmur](#)).

#### Improvements:

- When inserting data in a Replicated table, fewer requests are made to ZooKeeper (and most of the user-level errors have disappeared from the ZooKeeper log).
- Added the ability to create aliases for data sets. Example: `WITH (1, 2, 3) AS set SELECT number IN set FROM system.numbers LIMIT 10`.

#### Bug Fixes:

- Fixed the `Illegal PREWHERE` error when reading from Merge tables for Distributedtables.
- Added fixes that allow you to start clickhouse-server in IPv4-only Docker containers.
- Fixed a race condition when reading from system `system.parts_columns` tables.
- Removed double buffering during a synchronous insert to a `Distributed` table, which could have caused the connection to timeout.

- Fixed a bug that caused excessively long waits for an unavailable replica before beginning a `SELECT` query.
- Fixed incorrect dates in the `system.parts` table.
- Fixed a bug that made it impossible to insert data in a `Replicated` table if `chroot` was non-empty in the configuration of the `ZooKeeper` cluster.
- Fixed the vertical merging algorithm for an empty `ORDER BY` table.
- Restored the ability to use dictionaries in queries to remote tables, even if these dictionaries are not present on the requestor server. This functionality was lost in release 1.1.54362.
- Restored the behavior for queries like `SELECT * FROM remote('server2', default.table) WHERE col IN (SELECT col2 FROM default.table)` when the right side of the `IN` should use a remote `default.table` instead of a local one. This behavior was broken in version 1.1.54358.
- Removed extraneous error-level logging of `Not found column ... in block`

## ClickHouse Release 1.1.54362, 2018-03-11

### New Features:

- Aggregation without `GROUP BY` for an empty set (such as `SELECT count(*) FROM table WHERE 0`) now returns a result with one row with null values for aggregate functions, in compliance with the SQL standard. To restore the old behavior (return an empty result), set `empty_result_for_aggregation_by_empty_set` to 1.
- Added type conversion for `UNION ALL`. Different alias names are allowed in `SELECT` positions in `UNION ALL`, in compliance with the SQL standard.
- Arbitrary expressions are supported in `LIMIT BY` clauses. Previously, it was only possible to use columns resulting from `SELECT`.
- An index of `MergeTree` tables is used when `IN` is applied to a tuple of expressions from the columns of the primary key. Example: `WHERE (UserID, EventDate) IN ((123, '2000-01-01'), ...)` (Anastasiya Tsarkova).
- Added the `clickhouse-copier` tool for copying between clusters and resharding data (beta).
- Added consistent hashing functions: `yandexConsistentHash`, `jumpConsistentHash`, `sumburConsistentHash`. They can be used as a sharding key in order to reduce the amount of network traffic during subsequent reshardings.
- Added functions: `arrayAny`, `arrayAll`, `hasAny`, `hasAll`, `arrayIntersect`, `arrayResize`.
- Added the `arrayCumSum` function (Javi Santana).
- Added the `parseDateTimeBestEffort`, `parseDateTimeBestEffortOrZero`, and `parseDateTimeBestEffortOrNull` functions to read the `DateTime` from a string containing text in a wide variety of possible formats.
- Data can be partially reloaded from external dictionaries during updating (load just the records in which the value of the specified field greater than in the previous download) (Arsen Hakobyan).
- Added the `cluster` table function. Example: `cluster(cluster_name, db, table)`. The `remote` table function can accept the cluster name as the first argument, if it is specified as an identifier.
- The `remote` and `cluster` table functions can be used in `INSERT` queries.
- Added the `create_table_query` and `engine_full` virtual columns to the `system.tablestable` . The `metadata_modification_time` column is virtual.

- Added the `data_path` and `metadata_path` columns to `system.tables` and `system.databases` tables, and added the `path` column to the `system.parts` and `system.parts_columns` tables.
- Added additional information about merges in the `system.part_log` table.
- An arbitrary partitioning key can be used for the `system.query_log` table (Kirill Shvakov).
- The `SHOW TABLES` query now also shows temporary tables. Added temporary tables and the `is_temporary` column to `system.tables` (zhang2014).
- Added `DROP TEMPORARY TABLE` and `EXISTS TEMPORARY TABLE` queries (zhang2014).
- Support for `SHOW CREATE TABLE` for temporary tables (zhang2014).
- Added the `system_profile` configuration parameter for the settings used by internal processes.
- Support for loading `object_id` as an attribute in `MongoDB` dictionaries (Pavel Litvinenko).
- Reading `null` as the default value when loading data for an external dictionary with the `MongoDB` source (Pavel Litvinenko).
- Reading `DateTime` values in the `Values` format from a Unix timestamp without single quotes.
- Failover is supported in `remote` table functions for cases when some of the replicas are missing the requested table.
- Configuration settings can be overridden in the command line when you run `clickhouse-server`. Example: `clickhouse-server -- --logger.level=information`.
- Implemented the `empty` function from a `FixedString` argument: the function returns 1 if the string consists entirely of null bytes (zhang2014).
- Added the `listen_try` configuration parameter for listening to at least one of the listen addresses without quitting, if some of the addresses can't be listened to (useful for systems with disabled support for IPv4 or IPv6).
- Added the `VersionedCollapsingMergeTree` table engine.
- Support for rows and arbitrary numeric types for the `library` dictionary source.
- `MergeTree` tables can be used without a primary key (you need to specify `ORDER BY tuple()`).
- A `Nullable` type can be `CAST` to a non-`Nullable` type if the argument is not `NULL`.
- `RENAME TABLE` can be performed for `VIEW`.
- Added the `throwIf` function.
- Added the `odbc_default_field_size` option, which allows you to extend the maximum size of the value loaded from an ODBC source (by default, it is 1024).
- The `system.processes` table and `SHOW PROCESSLIST` now have the `is_cancelled` and `peak_memory_usage` columns.

## Improvements:

- Limits and quotas on the result are no longer applied to intermediate data for `INSERT SELECT` queries or for `SELECT` subqueries.
- Fewer false triggers of `force_restore_data` when checking the status of Replicated tables when the server starts.
- Added the `allow_distributed_ddl` option.

- Nondeterministic functions are not allowed in expressions for `MergeTree` table keys.
- Files with substitutions from `config.d` directories are loaded in alphabetical order.
- Improved performance of the `arrayElement` function in the case of a constant multidimensional array with an empty array as one of the elements. Example: `[[1], []][x]`.
- The server starts faster now when using configuration files with very large substitutions (for instance, very large lists of IP networks).
- When running a query, table valued functions run once. Previously, `remote` and `mysql` table valued functions performed the same query twice to retrieve the table structure from a remote server.
- The `MkDocs` documentation generator is used.
- When you try to delete a table column that `DEFAULT/MATERIALIZED` expressions of other columns depend on, an exception is thrown (zhang2014).
- Added the ability to parse an empty line in text formats as the number 0 for `Float` data types. This feature was previously available but was lost in release 1.1.54342.
- Enum values can be used in `min`, `max`, `sum` and some other functions. In these cases, it uses the corresponding numeric values. This feature was previously available but was lost in the release 1.1.54337.
- Added `max_expanded_ast_elements` to restrict the size of the AST after recursively expanding aliases.

#### Bug Fixes:

- Fixed cases when unnecessary columns were removed from subqueries in error, or not removed from subqueries containing `UNION ALL`.
- Fixed a bug in merges for `ReplacingMergeTree` tables.
- Fixed synchronous insertions in `Distributed` tables (`insert_distributed_sync = 1`).
- Fixed segfault for certain uses of `FULL` and `RIGHT JOIN` with duplicate columns in subqueries.
- Fixed segfault for certain uses of `replace_running_query` and `KILL QUERY`.
- Fixed the order of the `source` and `last_exception` columns in the `system.dictionaries` table.
- Fixed a bug when the `DROP DATABASE` query did not delete the file with metadata.
- Fixed the `DROP DATABASE` query for `Dictionary` databases.
- Fixed the low precision of `uniqHLL12` and `uniqCombined` functions for cardinalities greater than 100 million items (Alex Bocharov).
- Fixed the calculation of implicit default values when necessary to simultaneously calculate default explicit expressions in `INSERT` queries (zhang2014).
- Fixed a rare case when a query to a `MergeTree` table couldn't finish (chenxing-xc).
- Fixed a crash that occurred when running a `CHECK` query for `Distributed` tables if all shards are local (chenxing.xc).
- Fixed a slight performance regression with functions that use regular expressions.
- Fixed a performance regression when creating multidimensional arrays from complex expressions.
- Fixed a bug that could cause an extra `FORMAT` section to appear in an `.sql` file with metadata.

- Fixed a bug that caused the `max_table_size_to_drop` limit to apply when trying to delete a `MATERIALIZED VIEW` looking at an explicitly specified table.
- Fixed incompatibility with old clients (old clients were sometimes sent data with the `DateTime('timezone')` type, which they do not understand).
- Fixed a bug when reading `Nested` column elements of structures that were added using `ALTER` but that are empty for the old partitions, when the conditions for these columns moved to `PREWHERE`.
- Fixed a bug when filtering tables by virtual `_table` columns in queries to `Merge` tables.
- Fixed a bug when using `ALIAS` columns in `Distributed` tables.
- Fixed a bug that made dynamic compilation impossible for queries with aggregate functions from the `quantile` family.
- Fixed a race condition in the query execution pipeline that occurred in very rare cases when using `Merge` tables with a large number of tables, and when using `GLOBAL` subqueries.
- Fixed a crash when passing arrays of different sizes to an `arrayReduce` function when using aggregate functions from multiple arguments.
- Prohibited the use of queries with `UNION ALL` in a `MATERIALIZED VIEW`.
- Fixed an error during initialization of the `part_log` system table when the server starts (by default, `part_log` is disabled).

#### Backward Incompatible Changes:

- Removed the `distributed_ddl_allow_replicated_alter` option. This behavior is enabled by default.
- Removed the `strict_insert_defaults` setting. If you were using this functionality, write to `clickhouse-feedback@yandex-team.com`.
- Removed the `UnsortedMergeTree` engine.

## ClickHouse Release 1.1.54343, 2018-02-05

- Added macros support for defining cluster names in distributed DDL queries and constructors of Distributed tables: `CREATE TABLE distr ON CLUSTER '{cluster}' (...) ENGINE = Distributed('{cluster}', 'db', 'table')`
- Now queries like `SELECT ... FROM table WHERE expr IN (subquery)` are processed using the `table` index.
- Improved processing of duplicates when inserting to Replicated tables, so they no longer slow down execution of the replication queue.

## ClickHouse Release 1.1.54342, 2018-01-22

This release contains bug fixes for the previous release 1.1.54337:

- Fixed a regression in 1.1.54337: if the default user has readonly access, then the server refuses to start up with the message `Cannot create database in readonly mode`.
- Fixed a regression in 1.1.54337: on systems with `systemd`, logs are always written to `syslog` regardless of the configuration; the `watchdog` script still uses `init.d`.
- Fixed a regression in 1.1.54337: wrong default configuration in the Docker image.
- Fixed nondeterministic behavior of `GraphiteMergeTree` (you can see it in log messages `Data after merge is not byte-identical to the data on another replicas`).

- Fixed a bug that may lead to inconsistent merges after OPTIMIZE query to Replicated tables (you may see it in log messages `Part ... intersects the previous part`).
- Buffer tables now work correctly when MATERIALIZED columns are present in the destination table (by zhang2014).
- Fixed a bug in implementation of NULL.

## ClickHouse Release 1.1.54337, 2018-01-18

### New Features:

- Added support for storage of multi-dimensional arrays and tuples (`Tuple` data type) in tables.
- Support for table functions for `DESCRIBE` and `INSERT` queries. Added support for subqueries in `DESCRIBE`. Examples: `DESC TABLE remote('host', default.hits); DESC TABLE (SELECT 1); INSERT INTO TABLE FUNCTION remote('host', default.hits)`. Support for `INSERT INTO TABLE` in addition to `INSERT INTO`.
- Improved support for time zones. The `DateTime` data type can be annotated with the timezone that is used for parsing and formatting in text formats. Example: `DateTime('Europe/Moscow')`. When timezones are specified in functions for `DateTime` arguments, the return type will track the timezone, and the value will be displayed as expected.
- Added the functions `toTimeZone`, `timeDiff`, `toQuarter`, `toRelativeQuarterNum`. The `toRelativeHour/Minute/Second` functions can take a value of type `Date` as an argument. The `now` function name is case-sensitive.
- Added the `toStartOfFifteenMinutes` function (Kirill Shvakov).
- Added the `clickhouse format` tool for formatting queries.
- Added the `format_schema_path` configuration parameter (Marek Vavruša). It is used for specifying a schema in `Cap'n Proto` format. Schema files can be located only in the specified directory.
- Added support for config substitutions (`incl` and `conf.d`) for configuration of external dictionaries and models (Pavel Yakunin).
- Added a column with documentation for the `system.settings` table (Kirill Shvakov).
- Added the `system.parts_columns` table with information about column sizes in each data part of `MergeTree` tables.
- Added the `system.models` table with information about loaded `CatBoost` machine learning models.
- Added the `mysql` and `odbc` table function and corresponding `MySQL` and `ODBC` table engines for accessing remote databases. This functionality is in the beta stage.
- Added the possibility to pass an argument of type `AggregateFunction` for the `groupArray` aggregate function (so you can create an array of states of some aggregate function).
- Removed restrictions on various combinations of aggregate function combinator. For example, you can use `avgForEachIf` as well as `avgIfForEach` aggregate functions, which have different behaviors.
- The `-ForEach` aggregate function combinator is extended for the case of aggregate functions of multiple arguments.
- Added support for aggregate functions of `Nullable` arguments even for cases when the function returns a non-`Nullable` result (added with the contribution of Silviu Caragea). Example: `groupArray`, `groupUniqArray`, `topK`.
- Added the `max_client_network_bandwidth` for `clickhouse-client` (Kirill Shvakov).

- Users with the `readonly = 2` setting are allowed to work with TEMPORARY tables (CREATE, DROP, INSERT...) (Kirill Shvakov).
- Added support for using multiple consumers with the Kafka engine. Extended configuration options for Kafka (Marek Vavruša).
- Added the `intExp3` and `intExp4` functions.
- Added the `sumKahan` aggregate function.
- Added the `to * Number*` OrNull functions, where `* Number*` is a numeric type.
- Added support for `WITH` clauses for an `INSERT SELECT` query (author: zhang2014).
- Added settings: `http_connection_timeout`, `http_send_timeout`, `http_receive_timeout`. In particular, these settings are used for downloading data parts for replication. Changing these settings allows for faster failover if the network is overloaded.
- Added support for `ALTER` for tables of type `Null` (Anastasiya Tsarkova).
- The `reinterpretAsString` function is extended for all data types that are stored contiguously in memory.
- Added the `--silent` option for the `clickhouse-local` tool. It suppresses printing query execution info in stderr.
- Added support for reading values of type `Date` from text in a format where the month and/or day of the month is specified using a single digit instead of two digits (Amos Bird).

## Performance Optimizations:

- Improved performance of aggregate functions `min`, `max`, `any`, `anyLast`, `anyHeavy`, `argMin`, `argMax` from string arguments.
- Improved performance of the functions `isInfinite`, `isFinite`, `isNaN`, `roundToExp2`.
- Improved performance of parsing and formatting `Date` and `DateTime` type values in text format.
- Improved performance and precision of parsing floating point numbers.
- Lowered memory usage for `JOIN` in the case when the left and right parts have columns with identical names that are not contained in `USING`.
- Improved performance of aggregate functions `varSamp`, `varPop`, `stddevSamp`, `stddevPop`, `covarSamp`, `covarPop`, `corr` by reducing computational stability. The old functions are available under the names `varSampStable`, `varPopStable`, `stddevSampStable`, `stddevPopStable`, `covarSampStable`, `covarPopStable`, `corrStable`.

## Bug Fixes:

- Fixed data deduplication after running a `DROP` or `DETACH PARTITION` query. In the previous version, dropping a partition and inserting the same data again was not working because inserted blocks were considered duplicates.
- Fixed a bug that could lead to incorrect interpretation of the `WHERE` clause for `CREATE MATERIALIZED VIEW` queries with `POPULATE`.
- Fixed a bug in using the `root_path` parameter in the `zookeeper_servers` configuration.
- Fixed unexpected results of passing the `Date` argument to `toStartOfDay`.
- Fixed the `addMonths` and `subtractMonths` functions and the arithmetic for `INTERVAL n MONTH` in cases when the result has the previous year.

- Added missing support for the `UUID` data type for `DISTINCT`, `JOIN`, and `uniq` aggregate functions and external dictionaries (Evgeniy Ivanov). Support for `UUID` is still incomplete.
- Fixed `SummingMergeTree` behavior in cases when the rows summed to zero.
- Various fixes for the `Kafka` engine (Marek Vavruša).
- Fixed incorrect behavior of the `Join` table engine (Amos Bird).
- Fixed incorrect allocator behavior under FreeBSD and OS X.
- The `extractAll` function now supports empty matches.
- Fixed an error that blocked usage of `libressl` instead of `openssl`.
- Fixed the `CREATE TABLE AS SELECT` query from temporary tables.
- Fixed non-atomicity of updating the replication queue. This could lead to replicas being out of sync until the server restarts.
- Fixed possible overflow in `gcd`, `lcm` and `modulo` (% operator) (Maks Skorokhod).
- `-preprocessed` files are now created after changing `umask` (`umask` can be changed in the config).
- Fixed a bug in the background check of parts (`MergeTreePartChecker`) when using a custom partition key.
- Fixed parsing of tuples (values of the `Tuple` data type) in text formats.
- Improved error messages about incompatible types passed to `multilif`, `array` and some other functions.
- Redesigned support for `Nullable` types. Fixed bugs that may lead to a server crash. Fixed almost all other bugs related to `NULL` support: incorrect type conversions in `INSERT SELECT`, insufficient support for `Nullable` in `HAVING` and `PREWHERE`, `join_use_nulls` mode, `Nullable` types as arguments of `OR` operator, etc.
- Fixed various bugs related to internal semantics of data types. Examples: unnecessary summing of `Enum` type fields in `SummingMergeTree`; alignment of `Enum` types in `Pretty` formats, etc.
- Stricter checks for allowed combinations of composite columns.
- Fixed the overflow when specifying a very large parameter for the `FixedString` data type.
- Fixed a bug in the `topK` aggregate function in a generic case.
- Added the missing check for equality of array sizes in arguments of n-ary variants of aggregate functions with an `-Array` combinator.
- Fixed a bug in `--pager` for `clickhouse-client` (author: ks1322).
- Fixed the precision of the `exp10` function.
- Fixed the behavior of the `visitParamExtract` function for better compliance with documentation.
- Fixed the crash when incorrect data types are specified.
- Fixed the behavior of `DISTINCT` in the case when all columns are constants.
- Fixed query formatting in the case of using the `tupleElement` function with a complex constant expression as the tuple element index.
- Fixed a bug in `Dictionary` tables for `range_hashed` dictionaries.
- Fixed a bug that leads to excessive rows in the result of `FULL` and `RIGHT JOIN` (Amos Bird).

- Fixed a server crash when creating and removing temporary files in `config.d` directories during config reload.
- Fixed the `SYSTEM DROP DNS CACHE` query: the cache was flushed but addresses of cluster nodes were not updated.
- Fixed the behavior of `MATERIALIZED VIEW` after executing `DETACH TABLE` for the table under the view (Marek Vavruša).

## Build Improvements:

- The `pbuilder` tool is used for builds. The build process is almost completely independent of the build host environment.
- A single build is used for different OS versions. Packages and binaries have been made compatible with a wide range of Linux systems.
- Added the `clickhouse-test` package. It can be used to run functional tests.
- The source tarball can now be published to the repository. It can be used to reproduce the build without using GitHub.
- Added limited integration with Travis CI. Due to limits on build time in Travis, only the debug build is tested and a limited subset of tests are run.
- Added support for `Cap'n'Proto` in the default build.
- Changed the format of documentation sources from `Restricted Text` to `Markdown`.
- Added support for `systemd` (Vladimir Smirnov). It is disabled by default due to incompatibility with some OS images and can be enabled manually.
- For dynamic code generation, `clang` and `lld` are embedded into the `clickhouse` binary. They can also be invoked as `clickhouse clang` and `clickhouse lld`.
- Removed usage of GNU extensions from the code. Enabled the `-Wextra` option. When building with `clang` the default is `libc++` instead of `libstdc++`.
- Extracted `clickhouse_parsers` and `clickhouse_common_io` libraries to speed up builds of various tools.

## Backward Incompatible Changes:

- The format for marks in `Log` type tables that contain `Nullable` columns was changed in a backward incompatible way. If you have these tables, you should convert them to the `TinyLog` type before starting up the new server version. To do this, replace `ENGINE = Log` with `ENGINE = TinyLog` in the corresponding `.sql` file in the `metadata` directory. If your table does not have `Nullable` columns or if the type of your table is not `Log`, then you do not need to do anything.
- Removed the `experimental_allow_extended_storage_definition_syntax` setting. Now this feature is enabled by default.
- The `runningIncome` function was renamed to `runningDifferenceStartingWithFirstValue` to avoid confusion.
- Removed the `FROM ARRAY JOIN arr` syntax when `ARRAY JOIN` is specified directly after `FROM` with no table (Amos Bird).
- Removed the `BlockTabSeparated` format that was used solely for demonstration purposes.

- Changed the state format for aggregate functions `varSamp`, `varPop`, `stddevSamp`, `stddevPop`, `covarSamp`, `covarPop`, `corr`. If you have stored states of these aggregate functions in tables (using the `AggregateFunction` data type or materialized views with corresponding states), please write to [clickhouse-feedback@yandex-team.com](mailto:clickhouse-feedback@yandex-team.com).
- In previous server versions there was an undocumented feature: if an aggregate function depends on parameters, you can still specify it without parameters in the `AggregateFunction` data type. Example: `AggregateFunction(quantiles, UInt64)` instead of `AggregateFunction(quantiles(0.5, 0.9), UInt64)`. This feature was lost. Although it was undocumented, we plan to support it again in future releases.
- Enum data types cannot be used in min/max aggregate functions. This ability will be returned in the next release.

#### Please Note When Upgrading:

- When doing a rolling update on a cluster, at the point when some of the replicas are running the old version of ClickHouse and some are running the new version, replication is temporarily stopped and the message `unknown parameter 'shard'` appears in the log. Replication will continue after all replicas of the cluster are updated.
- If different versions of ClickHouse are running on the cluster servers, it is possible that distributed queries using the following functions will have incorrect results: `varSamp`, `varPop`, `stddevSamp`, `stddevPop`, `covarSamp`, `covarPop`, `corr`. You should update all cluster nodes.

## Changelog for 2017

---

### ClickHouse Release 1.1.54327, 2017-12-21

This release contains bug fixes for the previous release 1.1.54318:

- Fixed bug with possible race condition in replication that could lead to data loss. This issue affects versions 1.1.54310 and 1.1.54318. If you use one of these versions with Replicated tables, the update is strongly recommended. This issue shows in logs in Warning messages like `Part ... from own log does not exist`. The issue is relevant even if you do not see these messages in logs.

### ClickHouse Release 1.1.54318, 2017-11-30

This release contains bug fixes for the previous release 1.1.54310:

- Fixed incorrect row deletions during merges in the SummingMergeTree engine
- Fixed a memory leak in unreplicated MergeTree engines
- Fixed performance degradation with frequent inserts in MergeTree engines
- Fixed an issue that was causing the replication queue to stop running
- Fixed rotation and archiving of server logs

### ClickHouse Release 1.1.54310, 2017-11-01

#### New Features:

- Custom partitioning key for the MergeTree family of table engines.
- **Kafka** table engine.
- Added support for loading **CatBoost** models and applying them to data stored in ClickHouse.
- Added support for time zones with non-integer offsets from UTC.

- Added support for arithmetic operations with time intervals.
- The range of values for the Date and DateTime types is extended to the year 2105.
- Added the CREATE MATERIALIZED VIEW x TO y query (specifies an existing table for storing the data of a materialized view).
- Added the `ATTACH TABLE` query without arguments.
- The processing logic for Nested columns with names ending in -Map in a SummingMergeTree table was extracted to the sumMap aggregate function. You can now specify such columns explicitly.
- Max size of the IP trie dictionary is increased to 128M entries.
- Added the `getSizeOfEnumType` function.
- Added the `sumWithOverflow` aggregate function.
- Added support for the Cap'n Proto input format.
- You can now customize compression level when using the zstd algorithm.

#### Backward Incompatible Changes:

- Creation of temporary tables with an engine other than Memory is not allowed.
- Explicit creation of tables with the View or MaterializedView engine is not allowed.
- During table creation, a new check verifies that the sampling key expression is included in the primary key.

#### Bug Fixes:

- Fixed hangups when synchronously inserting into a Distributed table.
- Fixed nonatomic adding and removing of parts in Replicated tables.
- Data inserted into a materialized view is not subjected to unnecessary deduplication.
- Executing a query to a Distributed table for which the local replica is lagging and remote replicas are unavailable does not result in an error anymore.
- Users do not need access permissions to the `default` database to create temporary tables anymore.
- Fixed crashing when specifying the Array type without arguments.
- Fixed hangups when the disk volume containing server logs is full.
- Fixed an overflow in the `toRelativeWeekNum` function for the first week of the Unix epoch.

#### Build Improvements:

- Several third-party libraries (notably Poco) were updated and converted to git submodules.

## ClickHouse Release 1.1.54304, 2017-10-19

#### New Features:

- TLS support in the native protocol (to enable, set `tcp_ssl_port` in `config.xml` ).

#### Bug Fixes:

- `ALTER` for replicated tables now tries to start running as soon as possible.
- Fixed crashing when reading data with the setting `preferred_block_size_bytes=0`.

- Fixed crashes of `clickhouse-client` when pressing `Page Down`
- Correct interpretation of certain complex queries with `GLOBAL IN` and `UNION ALL`
- `FREEZE PARTITION` always works atomically now.
- Empty POST requests now return a response with code 411.
- Fixed interpretation errors for expressions like `CAST(1 AS Nullable(UInt8))`.
- Fixed an error when reading `Array(Nullable(String))` columns from `MergeTree` tables.
- Fixed crashing when parsing queries like `SELECT dummy AS dummy, dummy AS b`
- Users are updated correctly with invalid `users.xml`
- Correct handling when an executable dictionary returns a non-zero response code.

## ClickHouse Release 1.1.54292, 2017-09-20

### New Features:

- Added the `pointInPolygon` function for working with coordinates on a coordinate plane.
- Added the `sumMap` aggregate function for calculating the sum of arrays, similar to `SummingMergeTree`.
- Added the `trunc` function. Improved performance of the rounding functions (`round`, `floor`, `ceil`, `roundToExp2`) and corrected the logic of how they work. Changed the logic of the `roundToExp2` function for fractions and negative numbers.
- The ClickHouse executable file is now less dependent on the `libc` version. The same ClickHouse executable file can run on a wide variety of Linux systems. There is still a dependency when using compiled queries (with the setting `compile = 1`, which is not used by default).
- Reduced the time needed for dynamic compilation of queries.

### Bug Fixes:

- Fixed an error that sometimes produced `part ... intersects previous part` messages and weakened replica consistency.
- Fixed an error that caused the server to lock up if ZooKeeper was unavailable during shutdown.
- Removed excessive logging when restoring replicas.
- Fixed an error in the `UNION ALL` implementation.
- Fixed an error in the `concat` function that occurred if the first column in a block has the `Array` type.
- Progress is now displayed correctly in the `system.merges` table.

## ClickHouse Release 1.1.54289, 2017-09-13

### New Features:

- `SYSTEM` queries for server administration: `SYSTEM RELOAD DICTIONARY`, `SYSTEM RELOAD DICTIONARIES`, `SYSTEM DROP DNS CACHE`, `SYSTEM SHUTDOWN`, `SYSTEM KILL`.
- Added functions for working with arrays: `concat`, `arraySlice`, `arrayPushBack`, `arrayPushFront`, `arrayPopBack`, `arrayPopFront`.
- Added `root` and `identity` parameters for the ZooKeeper configuration. This allows you to isolate individual users on the same ZooKeeper cluster.

- Added aggregate functions `groupBitAnd`, `groupBitOr`, and `groupBitXor` (for compatibility, they are also available under the names `BIT_AND`, `BIT_OR`, and `BIT_XOR`).
- External dictionaries can be loaded from MySQL by specifying a socket in the filesystem.
- External dictionaries can be loaded from MySQL over SSL (`ssl_cert`, `ssl_key`, `ssl_ca` parameters).
- Added the `max_network_bandwidth_for_user` setting to restrict the overall bandwidth use for queries per user.
- Support for `DROP TABLE` for temporary tables.
- Support for reading `DateTime` values in Unix timestamp format from the `CSV` and `JSONEachRow` formats.
- Lagging replicas in distributed queries are now excluded by default (the default threshold is 5 minutes).
- FIFO locking is used during ALTER: an ALTER query isn't blocked indefinitely for continuously running queries.
- Option to set `umask` in the config file.
- Improved performance for queries with `DISTINCT`.

#### Bug Fixes:

- Improved the process for deleting old nodes in ZooKeeper. Previously, old nodes sometimes didn't get deleted if there were very frequent inserts, which caused the server to be slow to shut down, among other things.
- Fixed randomization when choosing hosts for the connection to ZooKeeper.
- Fixed the exclusion of lagging replicas in distributed queries if the replica is localhost.
- Fixed an error where a data part in a `ReplicatedMergeTree` table could be broken after running `ALTER MODIFY` on an element in a `Nested` structure.
- Fixed an error that could cause `SELECT` queries to "hang".
- Improvements to distributed DDL queries.
- Fixed the query `CREATE TABLE ... AS <materialized view>`.
- Resolved the deadlock in the `ALTER ... CLEAR COLUMN IN PARTITION` query for `Buffer` tables.
- Fixed the invalid default value for `Enum` s (0 instead of the minimum) when using the `JSONEachRow` and `TSKV` formats.
- Resolved the appearance of zombie processes when using a dictionary with an `executable` source.
- Fixed segfault for the `HEAD` query.

#### Improved Workflow for Developing and Assembling ClickHouse:

- You can use `pbuilder` to build ClickHouse.
- You can use `libc++` instead of `libstdc++` for builds on Linux.
- Added instructions for using static code analysis tools: `Coverage`, `clang-tidy`, `cppcheck`.

#### Please Note When Upgrading:

- There is now a higher default value for the MergeTree setting `max_bytes_to_merge_at_max_space_in_pool` (the maximum total size of data parts to merge, in bytes): it has increased from 100 GiB to 150 GiB. This might result in large merges running after the server upgrade, which could cause an increased load on the disk subsystem. If the free space available on the server is less than twice the total amount of the merges that are running, this will cause all other merges to stop running, including merges of small data parts. As a result, INSERT queries will fail with the message “Merges are processing significantly slower than inserts.” Use the `SELECT * FROM system.merges` query to monitor the situation. You can also check the `DiskSpaceReservedForMerge` metric in the `system.metrics` table, or in Graphite. You do not need to do anything to fix this, since the issue will resolve itself once the large merges finish. If you find this unacceptable, you can restore the previous value for the `max_bytes_to_merge_at_max_space_in_pool` setting. To do this, go to the `<merge_tree>` section in `config.xml`, set `<merge_tree>``<max_bytes_to_merge_at_max_space_in_pool>107374182400</max_bytes_to_merge_at_max_space_in_pool>` and restart the server.

## ClickHouse Release 1.1.54284, 2017-08-29

- This is a bugfix release for the previous 1.1.54282 release. It fixes leaks in the parts directory in ZooKeeper.

## ClickHouse Release 1.1.54282, 2017-08-23

This release contains bug fixes for the previous release 1.1.54276:

- Fixed DB::Exception: Assertion violation: `!_path.empty()` when inserting into a Distributed table.
- Fixed parsing when inserting in RowBinary format if input data starts with';'.
- Errors during runtime compilation of certain aggregate functions (e.g. `groupArray()`).

## ClickHouse Release 1.1.54276, 2017-08-16

### New Features:

- Added an optional WITH section for a SELECT query. Example query: `WITH 1+1 AS a SELECT a, a*a`
- INSERT can be performed synchronously in a Distributed table: OK is returned only after all the data is saved on all the shards. This is activated by the setting `insert_distributed_sync=1`.
- Added the UUID data type for working with 16-byte identifiers.
- Added aliases of CHAR, FLOAT and other types for compatibility with the Tableau.
- Added the functions `toYYYYMM`, `toYYYYMMDD`, and `toYYYYMMDDhhmmss` for converting time into numbers.
- You can use IP addresses (together with the hostname) to identify servers for clustered DDL queries.
- Added support for non-constant arguments and negative offsets in the function `substring(str, pos, len)`.
- Added the `max_size` parameter for the `groupArray(max_size)(column)` aggregate function, and optimized its performance.

### Main Changes:

- Security improvements: all server files are created with 0640 permissions (can be changed via `<umask>` config parameter).
- Improved error messages for queries with invalid syntax.
- Significantly reduced memory consumption and improved performance when merging large sections of MergeTree data.

- Significantly increased the performance of data merges for the ReplacingMergeTree engine.
- Improved performance for asynchronous inserts from a Distributed table by combining multiple source inserts. To enable this functionality, use the setting `distributed_directory_monitor_batch_inserts=1`.

## Backward Incompatible Changes:

- Changed the binary format of aggregate states of `groupArray(array_column)` functions for arrays.

## Complete List of Changes:

- Added the `output_format_json_quote_denormals` setting, which enables outputting nan and inf values in JSON format.
- Optimized stream allocation when reading from a Distributed table.
- Settings can be configured in readonly mode if the value does not change.
- Added the ability to retrieve non-integer granules of the MergeTree engine in order to meet restrictions on the block size specified in the `preferred_block_size_bytes` setting. The purpose is to reduce the consumption of RAM and increase cache locality when processing queries from tables with large columns.
- Efficient use of indexes that contain expressions like `toStartOfHour(x)` for conditions like `toStartOfHour(x) op constexpr`.
- Added new settings for MergeTree engines (the `merge_tree` section in `config.xml`):
  - `replicated_deduplication_window_seconds` sets the number of seconds allowed for deduplicating inserts in Replicated tables.
  - `cleanup_delay_period` sets how often to start cleanup to remove outdated data.
  - `replicated_can_become_leader` can prevent a replica from becoming the leader (and assigning merges).
- Accelerated cleanup to remove outdated data from ZooKeeper.
- Multiple improvements and fixes for clustered DDL queries. Of particular interest is the new setting `distributed_ddl_task_timeout`, which limits the time to wait for a response from the servers in the cluster. If a ddl request has not been performed on all hosts, a response will contain a timeout error and a request will be executed in an async mode.
- Improved display of stack traces in the server logs.
- Added the “none” value for the compression method.
- You can use multiple `dictionaries_config` sections in `config.xml`.
- It is possible to connect to MySQL through a socket in the file system.
- The `system.parts` table has a new column with information about the size of marks, in bytes.

## Bug Fixes:

- Distributed tables using a Merge table now work correctly for a SELECT query with a condition on the `_table` field.
- Fixed a rare race condition in ReplicatedMergeTree when checking data parts.
- Fixed possible freezing on “leader election” when starting a server.
- The `max_replica_delay_for_distributed_queries` setting was ignored when using a local replica of the data source. This has been fixed.

- Fixed incorrect behavior of `ALTER TABLE CLEAR COLUMN IN PARTITION` when attempting to clean a non-existing column.
- Fixed an exception in the `multilf` function when using empty arrays or strings.
- Fixed excessive memory allocations when deserializing Native format.
- Fixed incorrect auto-update of Trie dictionaries.
- Fixed an exception when running queries with a GROUP BY clause from a Merge table when using SAMPLE.
- Fixed a crash of GROUP BY when using `distributed_aggregation_memory_efficient=1`.
- Now you can specify the `database.table` in the right side of IN and JOIN.
- Too many threads were used for parallel aggregation. This has been fixed.
- Fixed how the “if” function works with `FixedString` arguments.
- SELECT worked incorrectly from a Distributed table for shards with a weight of 0. This has been fixed.
- Running `CREATE VIEW IF EXISTS` no longer causes crashes.
- Fixed incorrect behavior when `input_format_skip_unknown_fields=1` is set and there are negative numbers.
- Fixed an infinite loop in the `dictGetHierarchy()` function if there is some invalid data in the dictionary.
- Fixed Syntax error: unexpected (...) errors when running distributed queries with subqueries in an IN or JOIN clause and Merge tables.
- Fixed an incorrect interpretation of a SELECT query from Dictionary tables.
- Fixed the “Cannot mremap” error when using arrays in IN and JOIN clauses with more than 2 billion elements.
- Fixed the failover for dictionaries with MySQL as the source.

#### Improved Workflow for Developing and Assembling ClickHouse:

- Builds can be assembled in Arcadia.
- You can use gcc 7 to compile ClickHouse.
- Parallel builds using `ccache+distcc` are faster now.

## ClickHouse Release 1.1.54245, 2017-07-04

#### New Features:

- Distributed DDL (for example, `CREATE TABLE ON CLUSTER`)
- The replicated query `ALTER TABLE CLEAR COLUMN IN PARTITION`.
- The engine for Dictionary tables (access to dictionary data in the form of a table).
- Dictionary database engine (this type of database automatically has Dictionary tables available for all the connected external dictionaries).
- You can check for updates to the dictionary by sending a request to the source.
- Qualified column names

- Quoting identifiers using double quotation marks.
- Sessions in the HTTP interface.
- The OPTIMIZE query for a Replicated table can run not only on the leader.

#### Backward Incompatible Changes:

- Removed SET GLOBAL.

#### Minor Changes:

- Now after an alert is triggered, the log prints the full stack trace.
- Relaxed the verification of the number of damaged/extraneous data parts at startup (there were too many false positives).

#### Bug Fixes:

- Fixed a bad connection “sticking” when inserting into a Distributed table.
- GLOBAL IN now works for a query from a Merge table that looks at a Distributed table.
- The incorrect number of cores was detected on a Google Compute Engine virtual machine. This has been fixed.
- Changes in how an executable source of cached external dictionaries works.
- Fixed the comparison of strings containing null characters.
- Fixed the comparison of Float32 primary key fields with constants.
- Previously, an incorrect estimate of the size of a field could lead to overly large allocations.
- Fixed a crash when querying a Nullable column added to a table using ALTER.
- Fixed a crash when sorting by a Nullable column, if the number of rows is less than LIMIT.
- Fixed an ORDER BY subquery consisting of only constant values.
- Previously, a Replicated table could remain in the invalid state after a failed DROP TABLE.
- Aliases for scalar subqueries with empty results are no longer lost.
- Now a query that used compilation does not fail with an error if the .so file gets damaged.

---

## ロード

### Q1 2020

- ロールベースのアクセス制御

### Q2 2020

- 外部認証サービスとの統合
- 資源のプールのためのより精密な分布のクラスター能力とユーザー

---

ClickHouseリリース19.14.3.3,2019-09-10で修正

CVE-2019-15024

An attacker that has write access to ZooKeeper and who can run a custom server available from the network where ClickHouse runs, can create a custom-built malicious server that will act as a ClickHouse replica and register it in ZooKeeper. When another replica will fetch data part from the malicious replica, it can force clickhouse-server to write to arbitrary path on filesystem.

クレジット : Yandexの情報セキュリティチームのEldar Zaitov

## CVE-2019-16535

An OOB read, OOB write and integer underflow in decompression algorithms can be used to achieve RCE or DoS via native protocol.

クレジット : Yandexの情報セキュリティチームのEldar Zaitov

## CVE-2019-16536

スタックオーバーフローへのDoSによっても実行させることができ、悪意のある認証クライアント

クレジット : Yandexの情報セキュリティチームのEldar Zaitov

## ClickHouseリリース 19.13.6.1, 2019-09-20で修正

## CVE-2019-18657

テーブル関数 url この脆弱性により、攻撃者は要求に任意のHTTPヘッダーを挿入することができました。

クレジット: ニキータ・チホミロフ

## ClickHouseリリース 18.12.13、2018-09-10で修正されました

## CVE-2018-14672

機能搭載CatBoostモデルの可路のフォーカストラバーサル読書任意のファイルをエラーメッセージが返されます。

クレジット : Yandexの情報セキュリティチームのアンドレイ Krasichkov

## ClickHouseリリース 18.10.3、2018-08-13で修正されました

## CVE-2018-14671

unixODBC許容荷重任意の共有オブジェクトからのファイルシステムにおけるリモートでコードが実行の脆弱性が存在します。

クレジット : Yandex情報セキュリティチームのAndrey KrasichkovとEvgeny Sidorov

## ClickHouseリリース 1.1.54388、2018-06-28で修正

## CVE-2018-14668

“remote” テーブル関数で任意のシンボルを許可 “user”, “password” と “default\_database” 分野を横断プロトコルの要求偽造攻撃であった。

クレジット : Yandexの情報セキュリティチームのアンドレイ Krasichkov

## ClickHouseリリース 1.1.54390、2018-07-06で修正

## CVE-2018-14669

ClickHouse MySQLクライアントがあった “LOAD DATA LOCAL INFILE” 機能が有効のとき、悪意のあるMySQLデータベース読む任意のファイルからの接続ClickHouseサーバーです。

## ClickHouseリリース1.1.54131、2017-01-10で修正

### CVE-2018-14670

Debパッケージ内の不適切な構成は、データベースの不正使用につながる可能性があります。

クレジット : 英国の国立サイバーセキュリティセンター (NCSC)

## General Questions About ClickHouse

Questions:

- [What is ClickHouse?](#)
- [Why ClickHouse is so fast?](#)
- [Who is using ClickHouse?](#)
- [What does “ClickHouse” mean?](#)
- [What does “He тормозит” mean?](#)
- [What is OLAP?](#)
- [What is a columnar database?](#)
- [Why not use something like MapReduce?](#)

### Don't see what you were looking for?

Check out [other F.A.Q. categories](#) or browse around main documentation articles found in the left sidebar.

## Why ClickHouse Is So Fast?

It was designed to be fast. Query execution performance has always been a top priority during the development process, but other important characteristics like user-friendliness, scalability, and security were also considered so ClickHouse could become a real production system.

ClickHouse was initially built as a prototype to do just a single task well: to filter and aggregate data as fast as possible. That's what needs to be done to build a typical analytical report and that's what a typical [GROUP BY](#) query does. ClickHouse team has made several high-level decisions that combined made achieving this task possible:

### Column-oriented storage

Source data often contain hundreds or even thousands of columns, while a report can use just a few of them. The system needs to avoid reading unnecessary columns, or most expensive disk read operations would be wasted.

### Indexes

ClickHouse keeps data structures in memory that allows reading not only used columns but only necessary row ranges of those columns.

### Data compression

Storing different values of the same column together often leads to better compression ratios

(compared to row-oriented systems) because in real data column often has the same or not so many different values for neighboring rows. In addition to general-purpose compression, ClickHouse supports [specialized codecs](#) that can make data even more compact.

## Vectorized query execution

ClickHouse not only stores data in columns but also processes data in columns. It leads to better CPU cache utilization and allows for [SIMD](#) CPU instructions usage.

## Scalability

ClickHouse can leverage all available CPU cores and disks to execute even a single query. Not only on a single server but all CPU cores and disks of a cluster as well.

But many other database management systems use similar techniques. What really makes ClickHouse stand out is **attention to low-level details**. Most programming languages provide implementations for most common algorithms and data structures, but they tend to be too generic to be effective. Every task can be considered as a landscape with various characteristics, instead of just throwing in random implementation. For example, if you need a hash table, here are some key questions to consider:

- Which hash function to choose?
- Collision resolution algorithm: [open addressing](#) vs [chaining](#)?
- Memory layout: one array for keys and values or separate arrays? Will it store small or large values?
- Fill factor: when and how to resize? How to move values around on resize?
- Will values be removed and which algorithm will work better if they will?
- Will we need fast probing with bitmaps, inline placement of string keys, support for non-movable values, prefetch, and batching?

Hash table is a key data structure for `GROUP BY` implementation and ClickHouse automatically chooses one of [30+ variations](#) for each specific query.

The same goes for algorithms, for example, in sorting you might consider:

- What will be sorted: an array of numbers, tuples, strings, or structures?
- Is all data available completely in RAM?
- Do we need a stable sort?
- Do we need a full sort? Maybe partial sort or n-th element will suffice?
- How to implement comparisons?
- Are we sorting data that has already been partially sorted?

Algorithms that they rely on characteristics of data they are working with can often do better than their generic counterparts. If it is not really known in advance, the system can try various implementations and choose the one that works best in runtime. For example, see an [article on how LZ4 decompression is implemented in ClickHouse](#).

Last but not least, the ClickHouse team always monitors the Internet on people claiming that they came up with the best implementation, algorithm, or data structure to do something and tries it out. Those claims mostly appear to be false, but from time to time you'll indeed find a gem.

## Tips for building your own high-performance software

- Keep in mind low-level details when designing your system.

- Design based on hardware capabilities.
- Choose data structures and abstractions based on the needs of the task.
- Provide specializations for special cases.
- Try new, “best” algorithms, that you read about yesterday.
- Choose an algorithm in runtime based on statistics.
- Benchmark on real datasets.
- Test for performance regressions in CI.
- Measure and observe everything.

## Who Is Using ClickHouse?

Being an open-source product makes this question not so straightforward to answer. You do not have to tell anyone if you want to start using ClickHouse, you just go grab source code or pre-compiled packages. There's no contract to sign and the [Apache 2.0 license](#) allows for unconstrained software distribution.

Also, the technology stack is often in a grey zone of what's covered by an NDA. Some companies consider technologies they use as a competitive advantage even if they are open-source and do not allow employees to share any details publicly. Some see some PR risks and allow employees to share implementation details only with their PR department approval.

So how to tell who is using ClickHouse?

One way is to **ask around**. If it's not in writing, people are much more willing to share what technologies are used in their companies, what the use cases are, what kind of hardware is used, data volumes, etc. We're talking with users regularly on [ClickHouse Meetups](#) all over the world and have heard stories about 1000+ companies that use ClickHouse. Unfortunately, that's not reproducible and we try to treat such stories as if they were told under NDA to avoid any potential troubles. But you can come to any of our future meetups and talk with other users on your own. There are multiple ways how meetups are announced, for example, you can subscribe to [our Twitter](#).

The second way is to look for companies **publicly saying** that they use ClickHouse. It's more substantial because there's usually some hard evidence like a blog post, talk video recording, slide deck, etc. We collect the collection of links to such evidence on our [Adopters](#) page. Feel free to contribute the story of your employer or just some links you've stumbled upon (but try not to violate your NDA in the process).

You can find names of very large companies in the adopters list, like Bloomberg, Cisco, China Telecom, Tencent, or Uber, but with the first approach, we found that there are many more. For example, if you take [the list of largest IT companies by Forbes \(2020\)](#) over half of them are using ClickHouse in some way. Also, it would be unfair not to mention [Yandex](#), the company which initially open-sourced ClickHouse in 2016 and happens to be one of the largest IT companies in Europe.

## What Does “ClickHouse” Mean?

It's a combination of “**Clickstream**” and “**Data wareHouse**”. It comes from the original use case at Yandex.Metrica, where ClickHouse was supposed to keep records of all clicks by people from all over the Internet, and it still does the job. You can read more about this use case on [ClickHouse history](#) page.

This two-part meaning has two consequences:

- The only correct way to write ClickHouse is with capital H.
- If you need to abbreviate it, use **CH**. For some historical reasons, abbreviating as CK is also popular in China, mostly because one of the first talks about ClickHouse in Chinese used this form.

## Fun fact

Many years after ClickHouse got its name, this approach of combining two words that are meaningful on their own has been highlighted as the best way to name a database in a [research by Andy Pavlo](#), an Associate Professor of Databases at Carnegie Mellon University. ClickHouse shared his “best database name of all time” award with Postgres.

## What Does “Не тормозит” Mean?

This question usually arises when people see official ClickHouse t-shirts. They have large words **“ClickHouse не тормозит”** on the front.

Before ClickHouse became open-source, it has been developed as an in-house storage system by the largest Russian IT company, [Yandex](#). That’s why it initially got its slogan in Russian, which is “не тормозит” (pronounced as “ne tormozit”). After the open-source release we first produced some of those t-shirts for events in Russia and it was a no-brainer to use the slogan as-is.

One of the following batches of those t-shirts was supposed to be given away on events outside of Russia and we tried to make the English version of the slogan. Unfortunately, the Russian language is kind of elegant in terms of expressing stuff and there was a restriction of limited space on a t-shirt, so we failed to come up with good enough translation (most options appeared to be either long or inaccurate) and decided to keep the slogan in Russian even on t-shirts produced for international events. It appeared to be a great decision because people all over the world get positively surprised and curious when they see it.

So, what does it mean? Here are some ways to translate “не тормозит”:

- If you translate it literally, it’d be something like “ClickHouse does not press the brake pedal”.
- If you’d want to express it as close to how it sounds to a Russian person with IT background, it’d be something like “If your larger system lags, it’s not because it uses ClickHouse”.
- Shorter, but not so precise versions could be “ClickHouse is not slow”, “ClickHouse does not lag” or just “ClickHouse is fast”.

If you haven’t seen one of those t-shirts in person, you can check them out online in many ClickHouse-related videos. For example, this one:

P.S. These t-shirts are not for sale, they are given away for free on most [ClickHouse Meetups](#), usually for best questions or other forms of active participation.

---

## What Is OLAP?

**OLAP** stands for Online Analytical Processing. It is a broad term that can be looked at from two perspectives: technical and business. But at the very high level, you can just read these words backward:

### **Processing**

Some source data is processed...

### **Analytical**

...to produce some analytical reports and insights...

### **Online**

...in real-time.

## OLAP from the Business Perspective

In recent years, business people started to realize the value of data. Companies who make their decisions blindly, more often than not fail to keep up with the competition. The data-driven approach of successful companies forces them to collect all data that might be remotely useful for making business decisions and need mechanisms to timely analyze them. Here's where OLAP database management systems (DBMS) come in.

In a business sense, OLAP allows companies to continuously plan, analyze, and report operational activities, thus maximizing efficiency, reducing expenses, and ultimately conquering the market share. It could be done either in an in-house system or outsourced to SaaS providers like web/mobile analytics services, CRM services, etc. OLAP is the technology behind many BI applications (Business Intelligence).

ClickHouse is an OLAP database management system that is pretty often used as a backend for those SaaS solutions for analyzing domain-specific data. However, some businesses are still reluctant to share their data with third-party providers and an in-house data warehouse scenario is also viable.

## OLAP from the Technical Perspective

All database management systems could be classified into two groups: OLAP (Online **Analytical** Processing) and OLTP (Online **Transactional** Processing). Former focuses on building reports, each based on large volumes of historical data, but doing it not so frequently. While the latter usually handle a continuous stream of transactions, constantly modifying the current state of data.

In practice OLAP and OLTP are not categories, it's more like a spectrum. Most real systems usually focus on one of them but provide some solutions or workarounds if the opposite kind of workload is also desired. This situation often forces businesses to operate multiple storage systems integrated, which might be not so big deal but having more systems make it more expensive to maintain. So the trend of recent years is HTAP (**Hybrid Transactional/Analytical Processing**) when both kinds of the workload are handled equally well by a single database management system.

Even if a DBMS started as a pure OLAP or pure OLTP, they are forced to move towards that HTAP direction to keep up with their competition. And ClickHouse is no exception, initially, it has been designed as **fast-as-possible OLAP system** and it still does not have full-fledged transaction support, but some features like consistent read/writes and mutations for updating/deleting data had to be added.

The fundamental trade-off between OLAP and OLTP systems remains:

- To build analytical reports efficiently it's crucial to be able to read columns separately, thus most OLAP databases are **columnar**,
- While storing columns separately increases costs of operations on rows, like append or in-place modification, proportionally to the number of columns (which can be huge if the systems try to collect all details of an event just in case). Thus, most OLTP systems store data arranged by rows.

## What Is a Columnar Database?

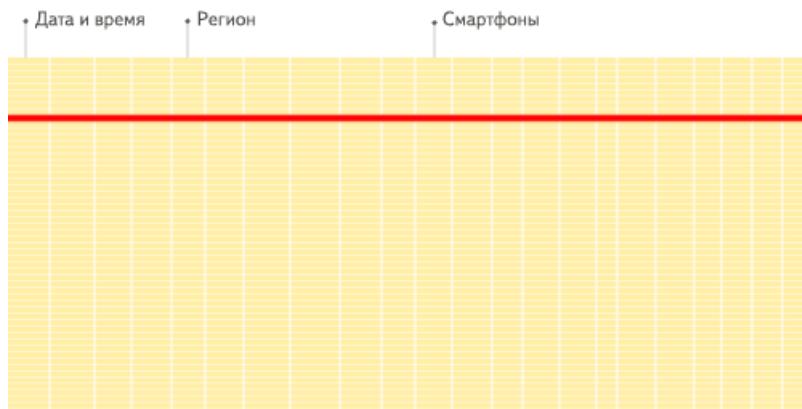
A columnar database stores data of each column independently. This allows to read data from disks only for those columns that are used in any given query. The cost is that operations that affect whole rows become proportionally more expensive. The synonym for a columnar database is a column-oriented database management system. ClickHouse is a typical example of such a system.

Key columnar database advantages are:

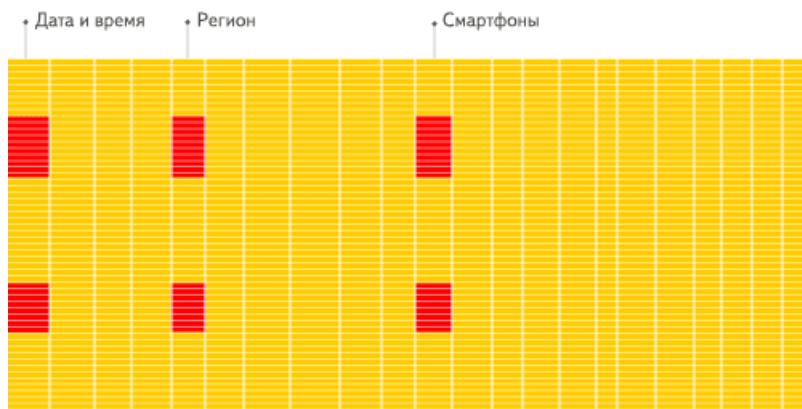
- Queries that use only a few columns out of many.
- Aggregating queries against large volumes of data.
- Column-wise data compression.

Here is the illustration of the difference between traditional row-oriented systems and columnar databases when building reports:

### Traditional row-oriented



### Columnar



A columnar database is a preferred choice for analytical applications because it allows to have many columns in a table just in case, but do not pay the cost for unused columns on read query execution time. Column-oriented databases are designed for big data processing because and data warehousing, they often natively scale using distributed clusters of low-cost hardware to increase throughput. ClickHouse does it with combination of [distributed](#) and [replicated](#) tables.

## Why Not Use Something Like MapReduce?

We can refer to systems like MapReduce as distributed computing systems in which the reduce operation is based on distributed sorting. The most common open-source solution in this class is [Apache Hadoop](#). Yandex uses its in-house solution, YT.

These systems aren't appropriate for online queries due to their high latency. In other words, they can't be used as the back-end for a web interface. These types of systems aren't useful for real-time data updates. Distributed sorting isn't the best way to perform reduce operations if the result of the operation and all the intermediate results (if there are any) are located in the RAM of a single server, which is usually the case for online queries. In such a case, a hash table is an optimal way to perform reduce operations. A common approach to optimizing map-reduce tasks is pre-aggregation (partial reduce) using a hash table in RAM. The user performs this optimization manually. Distributed sorting is one of the main causes of reduced performance when running simple map-reduce tasks.

Most MapReduce implementations allow you to execute arbitrary code on a cluster. But a declarative query language is better suited to OLAP to run experiments quickly. For example, Hadoop has Hive and Pig. Also consider Cloudera Impala or Shark (outdated) for Spark, as well as Spark SQL, Presto, and Apache Drill. Performance when running such tasks is highly sub-optimal compared to specialized systems, but relatively high latency makes it unrealistic to use these systems as the backend for a web interface.

## Questions About ClickHouse Use Cases

Questions:

- [Can I use ClickHouse as a time-series database?](#)
- [Can I use ClickHouse as a key-value storage?](#)

**Don't see what you were looking for?**

Check out [other F.A.Q. categories](#) or browse around main documentation articles found in the left sidebar.

## Can I Use ClickHouse As a Key-Value Storage?

The short answer is "**no**". The key-value workload is among top positions in the list of cases when **NOT** to use ClickHouse. It's an [OLAP](#) system after all, while there are many excellent key-value storage systems out there.

However, there might be situations where it still makes sense to use ClickHouse for key-value-like queries. Usually, it's some low-budget products where the main workload is analytical in nature and fits ClickHouse well, but there's also some secondary process that needs a key-value pattern with not so high request throughput and without strict latency requirements. If you had an unlimited budget, you would have

installed a secondary key-value database for thus secondary workload, but in reality, there's an additional cost of maintaining one more storage system (monitoring, backups, etc.) which might be desirable to avoid.

If you decide to go against recommendations and run some key-value-like queries against ClickHouse, here're some tips:

- The key reason why point queries are expensive in ClickHouse is its sparse primary index of main [MergeTree table engine family](#). This index can't point to each specific row of data, instead, it points to each N-th and the system has to scan from the neighboring N-th row to the desired one, reading excessive data along the way. In a key-value scenario, it might be useful to reduce the value of N with the `index_granularity` setting.
- ClickHouse keeps each column in a separate set of files, so to assemble one complete row it needs to go through each of those files. Their count increases linearly with the number of columns, so in the key-value scenario, it might be worth to avoid using many columns and put all your payload in a single `String` column encoded in some serialization format like JSON, Protobuf or whatever makes sense.
- There's an alternative approach that uses [Join](#) table engine instead of normal [MergeTree](#) tables and [joinGet](#) function to retrieve the data. It can provide better query performance but might have some usability and reliability issues. Here's an [usage example](#).

## Can I Use ClickHouse As a Time-Series Database?

ClickHouse is a generic data storage solution for [OLAP](#) workloads, while there are many specialized time-series database management systems. Nevertheless, ClickHouse's [focus on query execution speed](#) allows it to outperform specialized systems in many cases. There are many independent benchmarks on this topic out there, so we're not going to conduct one here. Instead, let's focus on ClickHouse features that are important to use if that's your use case.

First of all, there are [specialized codecs](#) which make typical time-series. Either common algorithms like `DoubleDelta` and `Gorilla` or specific to ClickHouse like `T64`.

Second, time-series queries often hit only recent data, like one day or one week old. It makes sense to use servers that have both fast nVME/SSD drives and high-capacity HDD drives. ClickHouse [TTL](#) feature allows to configure keeping fresh hot data on fast drives and gradually move it to slower drives as it ages. Rollup or removal of even older data is also possible if your requirements demand it.

Even though it's against ClickHouse philosophy of storing and processing raw data, you can use [materialized views](#) to fit into even tighter latency or costs requirements.

## Question About Operating ClickHouse Servers and Clusters

Questions:

- [Which ClickHouse version to use in production?](#)
- [Is it possible to delete old records from a ClickHouse table?](#)

**Don't see what you were looking for?**

Check out [other F.A.Q. categories](#) or browse around main documentation articles found in the left sidebar.

# Which ClickHouse Version to Use in Production?

First of all, let's discuss why people ask this question in the first place. There are two key reasons:

1. ClickHouse is developed with pretty high velocity and usually, there are 10+ stable releases per year. It makes a wide range of releases to choose from, which is not so trivial choice.
2. Some users want to avoid spending time figuring out which version works best for their use case and just follow someone else's advice.

The second reason is more fundamental, so we'll start with it and then get back to navigating through various ClickHouse releases.

## Which ClickHouse Version Do You Recommend?

It's tempting to hire consultants or trust some known experts to get rid of responsibility for your production environment. You install some specific ClickHouse version that someone else recommended, now if there's some issue with it - it's not your fault, it's someone else's. This line of reasoning is a big trap. No external person knows better what's going on in your company's production environment.

So how to properly choose which ClickHouse version to upgrade to? Or how to choose your first ClickHouse version? First of all, you need to invest in setting up a **realistic pre-production environment**. In an ideal world, it could be a completely identical shadow copy, but that's usually expensive.

Here're some key points to get reasonable fidelity in a pre-production environment with not so high costs:

- Pre-production environment needs to run an as close set of queries as you intend to run in production:
  - Don't make it read-only with some frozen data.
  - Don't make it write-only with just copying data without building some typical reports.
  - Don't wipe it clean instead of applying schema migrations.
- Use a sample of real production data and queries. Try to choose a sample that's still representative and makes `SELECT` queries return reasonable results. Use obfuscation if your data is sensitive and internal policies do not allow it to leave the production environment.
- Make sure that pre-production is covered by your monitoring and alerting software the same way as your production environment does.
- If your production spans across multiple datacenters or regions, make your pre-production does the same.
- If your production uses complex features like replication, distributed table, cascading materialize views, make sure they are configured similarly in pre-production.
- There's a trade-off on using the roughly same number of servers or VMs in pre-production as in production, but of smaller size, or much less of them, but of the same size. The first option might catch extra network-related issues, while the latter is easier to manage.

The second area to invest in is **automated testing infrastructure**. Don't assume that if some kind of query has executed successfully once, it'll continue to do so forever. It's ok to have some unit tests where ClickHouse is mocked but make sure your product has a reasonable set of automated tests that are run against real ClickHouse and check that all important use cases are still working as expected.

Extra step forward could be contributing those automated tests to [ClickHouse's open-source test infrastructure](#) that's continuously used in its day-to-day development. It definitely will take some additional time and effort to learn [how to run it](#) and then how to adapt your tests to this framework, but it'll pay off by ensuring that ClickHouse releases are already tested against them when they are announced stable, instead of repeatedly losing time on reporting the issue after the fact and then waiting for a bugfix to be implemented, backported and released. Some companies even have such test contributions to infrastructure by its use as an internal policy, most notably it's called [Beyonce's Rule](#) at Google.

When you have your pre-production environment and testing infrastructure in place, choosing the best version is straightforward:

1. Routinely run your automated tests against new ClickHouse releases. You can do it even for ClickHouse releases that are marked as [testing](#), but going forward to the next steps with them is not recommended.
2. Deploy the ClickHouse release that passed the tests to pre-production and check that all processes are running as expected.
3. Report any issues you discovered to [ClickHouse GitHub Issues](#).
4. If there were no major issues, it should be safe to start deploying ClickHouse release to your production environment. Investing in gradual release automation that implements an approach similar to [canary releases](#) or [green-blue deployments](#) might further reduce the risk of issues in production.

As you might have noticed, there's nothing specific to ClickHouse in the approach described above, people do that for any piece of infrastructure they rely on if they take their production environment seriously.

## How to Choose Between ClickHouse Releases?

If you look into contents of ClickHouse package repository, you'll see four kinds of packages:

1. [testing](#)
2. [prestable](#)
3. [stable](#)
4. [Its](#) (long-term support)

As was mentioned earlier, [testing](#) is good mostly to notice issues early, running them in production is not recommended because each of them is not tested as thoroughly as other kinds of packages.

[prestable](#) is a release candidate which generally looks promising and is likely to become announced as [stable](#) soon. You can try them out in pre-production and report issues if you see any.

For production use, there are two key options: [stable](#) and [Its](#). Here is some guidance on how to choose between them:

- [stable](#) is the kind of package we recommend by default. They are released roughly monthly (and thus provide new features with reasonable delay) and three latest stable releases are supported in terms of diagnostics and backporting of bugfixes.
- [Its](#) are released twice a year and are supported for a year after their initial release. You might prefer them over [stable](#) in the following cases:
  - Your company has some internal policies that do not allow for frequent upgrades or using non-LTS software.
  - You are using ClickHouse in some secondary products that either does not require any complex ClickHouse features and do not have enough resources to keep it updated.

Many teams who initially thought that [Its](#) is the way to go, often switch to [stable](#) anyway because of some recent feature that's important for their product.

## Important

One more thing to keep in mind when upgrading ClickHouse: we're always keeping eye on compatibility across releases, but sometimes it's not reasonable to keep and some minor details might change. So make sure you check the [changelog](#) before upgrading to see if there are any notes about backward-incompatible changes.

# Is It Possible to Delete Old Records from a ClickHouse Table?

The short answer is “yes”. ClickHouse has multiple mechanisms that allow freeing up disk space by removing old data. Each mechanism is aimed for different scenarios.

## TTL

ClickHouse allows to automatically drop values when some condition happens. This condition is configured as an expression based on any columns, usually just static offset for any timestamp column.

The key advantage of this approach is that it does not need any external system to trigger, once TTL is configured, data removal happens automatically in background.

## Note

TTL can also be used to move data not only to [/dev/null](#), but also between different storage systems, like from SSD to HDD.

More details on [configuring TTL](#).

## ALTER DELETE

ClickHouse does not have real-time point deletes like in [OLTP](#) databases. The closest thing to them are mutations. They are issued as `ALTER ... DELETE` or `ALTER ... UPDATE` queries to distinguish from normal `DELETE` or `UPDATE` as they are asynchronous batch operations, not immediate modifications. The rest of syntax after `ALTER TABLE` prefix is similar.

`ALTER DELETE` can be issued to flexibly remove old data. If you need to do it regularly, the main downside will be the need to have an external system to submit the query. There are also some performance considerations since mutation rewrite complete parts even there's only a single row to be deleted.

This is the most common approach to make your system based on ClickHouse [GDPR](#)-compliant.

More details on [mutations](#).

## DROP PARTITION

`ALTER TABLE ... DROP PARTITION` provides a cost-efficient way to drop a whole partition. It's not that flexible and needs proper partitioning scheme configured on table creation, but still covers most common cases. Like mutations need to be executed from an external system for regular use.

More details on [manipulating partitions](#).

## TRUNCATE

It's rather radical to drop all data from a table, but in some cases it might be exactly what you need.

More details on [table truncation](#).

## Questions About Integrating ClickHouse and Other Systems

Questions:

- [How do I export data from ClickHouse to a file?](#)
- [How to import JSON into ClickHouse?](#)
- [What if I have a problem with encodings when connecting to Oracle via ODBC?](#)

### Don't see what you were looking for?

Check out [other F.A.Q. categories](#) or browse around main documentation articles found in the left sidebar.

## How Do I Export Data from ClickHouse to a File?

### Using INTO OUTFILE Clause

Add an [INTO OUTFILE](#) clause to your query.

For example:

```
SELECT * FROM table INTO OUTFILE 'file'
```

By default, ClickHouse uses the [TabSeparated](#) format for output data. To select the [data format](#), use the [FORMAT clause](#).

For example:

```
SELECT * FROM table INTO OUTFILE 'file' FORMAT CSV
```

### Using a File-Engine Table

See [File](#) table engine.

### Using Command-Line Redirection

```
$ clickhouse-client --query "SELECT * from table" --format FormatName > result.txt
```

See [clickhouse-client](#).

## How to Import JSON Into ClickHouse?

ClickHouse supports a wide range of [data formats for input and output](#). There are multiple JSON variations among them, but the most commonly used for data ingestion is [JSONEachRow](#). It expects one JSON object per row, each object separated by a newline.

## Examples

Using [HTTP interface](#):

```
$ echo '{"foo":"bar"}' | curl 'http://localhost:8123/?query=INSERT%20INTO%20test%20FORMAT%20JSONEachRow' --data-binary @-
```

Using [CLI interface](#):

```
$ echo '{"foo":"bar"}' | clickhouse-client --query="INSERT INTO test FORMAT JSONEachRow"
```

Instead of inserting data manually, you might consider to use one of [client libraries](#) instead.

## Useful Settings

- `input_format_skip_unknown_fields` allows to insert JSON even if there were additional fields not present in table schema (by discarding them).
- `input_format_import_nested_json` allows to insert nested JSON objects into columns of [Nested](#) type.

### Note

Settings are specified as [GET parameters](#) for the HTTP interface or as additional command-line arguments prefixed with `--` for the [CLI interface](#).

## What If I Have a Problem with Encodings When Using Oracle Via ODBC?

If you use Oracle as a source of ClickHouse external dictionaries via Oracle ODBC driver, you need to set the correct value for the `NLS_LANG` environment variable in `/etc/default/clickhouse`. For more information, see the [Oracle NLS\\_LANG FAQ](#).

### Example

```
NLS_LANG=RUSSIAN_RUSSIA.UTF8
```

## 一般的な質問

### なぜMapReduceのようなものを使わないのですか？

MapReduceのようなシステムは、reduce操作が分散ソートに基づいている分散コンピューティングシステムと呼ぶことができます。このクラ [Apache Hadoop](#). Yandexは社内ソリューションYTを使用しています。

これらのシステムなどの適切なオンラインのクエリによる高い待ち時間をゼロにすることにつまり、webインターフェイスのバックエンドとして使用することはできません。これらのシステムは役立つリアルタイムデータの更新をした。分散ソートは、操作の結果とすべての中間結果（存在する場合）が単一のサーバーのRAMにある場合、reduce操作を実行する最良の方法ではありません。このような場合、ハッシュテーブルはreduce操作を実行する最適な方法です。Map-reduceタスクを最適化するための一般的なアプローチは、RAM内のハッシュテーブルを使用した事前集約（部分削減）です。ユーザーはこの最適化を手動で実行します。分散ソートは、単純なmap-reduceタスクを実行するときのパフォーマンスの低下の主な原因の一つです。

ほとんどのMapReduce実装では、クラスター上で任意のコードを実行できます。しかし、宣言型クエリ言語は、実験を迅速に実行するためにOLAPに適しています。たとえば、HadoopにはHiveとPigがあります。また、SparkのCloudera ImpalaやShark（旧式）、Spark SQL、Presto、Apache Drillについても検討してください。このようなタスクを実行するときのパフォーマンスは、特殊なシステムに比べて非常に最適ではありませんが、比較的遅延が高いため、これらのシステム

## になっているのでしょうか？しているのでエンコーディング利用の場合OracleをODBC？

外部ディクショナリのソースとしてODBCドライバを介してOracleを使用する場合は、外部ディクショナリの正しい値を設定する必要がある NLS\_LANG 環境変数 in /etc/default/clickhouse. 詳細については、を参照してください [Oracle NLS\\_LANG FAQ](#).

例

```
NLS_LANG=RUSSIAN_RUSSIA.UTF8
```

## ClickHouseからファイルにデータをエクスポートするには？

### INTO OUTFILE句の使用

追加 [INTO OUTFILE](#) クエリの句。

例えば：

```
SELECT * FROM table INTO OUTFILE 'file'
```

デフォルトでは、ClickHouseは [TabSeparated](#) 出力データの形式。選択するには [データ形式](#) は、[FORMAT句](#).

例えば：

```
SELECT * FROM table INTO OUTFILE 'file' FORMAT CSV
```

### ファイルエンジンテーブルの使用

見る [ファイル](#).

### コマンドラインリダイレクト

```
$ clickhouse-client --query "SELECT * from table" --format FormatName > result.txt
```

見る [clickhouse-クライアント](#).

Was this content helpful?  
RATING\_STARS

Rating: RATING\_VALUE - RATING\_COUNT  
votes

©2016–2021 ClickHouse, Inc.

Built from c936fa93c3