

什么是ClickHouse？

ClickHouse是一个用于联机分析(OLAP)的列式数据库管理系统(DBMS)。

在传统的行式数据库系统中，数据按如下顺序存储：

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

处于同一行中的数据总是被物理的存储在一起。

常见的行式数据库系统有：[MySQL](#)、[Postgres](#)和[MS SQL Server](#)。

在列式数据库系统中，数据按如下的顺序存储：

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	...

这些示例只显示了数据的排列顺序。来自不同列的值被单独存储，来自同一列的数据被存储在一起。

常见的列式数据库有：[Vertica](#)、[Paraccel](#) ([Actian Matrix](#)，[Amazon Redshift](#))、[Sybase IQ](#)、[Exasol](#)、[Infobright](#)、[InfiniDB](#)、[MonetDB](#) ([VectorWise](#)，[Actian Vector](#))、[LucidDB](#)、[SAP HANA](#)、[Google Dremel](#)、[Google PowerDrill](#)、[Druid](#)、[kdb+](#)。

不同的数据存储方式适用不同的业务场景，数据访问的场景包括：进行了何种查询、多久查询一次以及各类查询的比例；每种类型的查询(行、列和字节)读取多少数据；读取数据和更新之间的关系；使用的数据集大小以及如何使用本地的数据集；是否使用事务，以及它们是如何进行隔离的；数据的复制机制与数据的完整性要求；每种类型的查询要求的延迟与吞吐量等。

系统负载越高，依据使用场景进行定制化就越重要，并且定制将会变的越精细。没有一个系统能够同时适用所有不同的业务场景。如果系统适用于广泛的场景，在负载高的情况下，要兼顾所有的场景，那么将不得不做出选择。是要平衡还是要效率？

OLAP场景的关键特征

- 绝大多数是读请求
- 数据以相当大的批次(> 1000行)更新，而不是单行更新;或者根本没有更新。

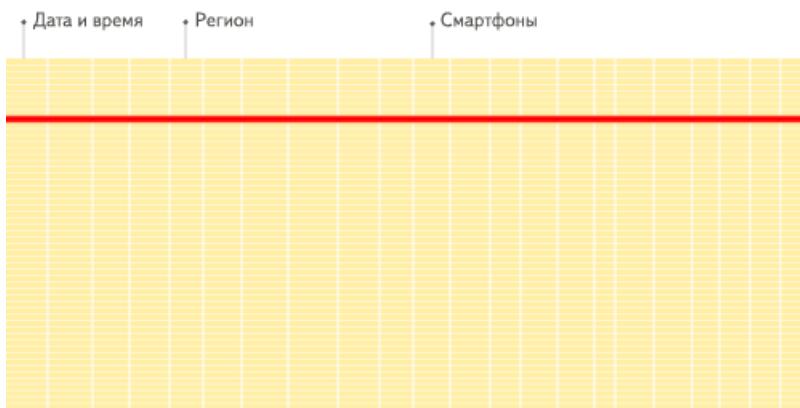
- 已添加到数据库的数据不能修改。
- 对于读取，从数据库中提取相当多的行，但只提取列的一小部分。
- 宽表，即每个表包含着大量的列
- 查询相对较少(通常每台服务器每秒查询数百次或更少)
- 对于简单查询，允许延迟大约50毫秒
- 列中的数据相对较小：数字和短字符串(例如，每个URL 60个字节)
- 处理单个查询时需要高吞吐量(每台服务器每秒可达数十亿行)
- 事务不是必须的
- 对数据一致性要求低
- 每个查询有一个大表。除了他以外，其他的都很小。
- 查询结果明显小于源数据。换句话说，数据经过过滤或聚合，因此结果适合于单个服务器的RAM中

很容易可以看出，OLAP场景与其他通常业务场景(例如,OLTP或K/V)有很大的不同，因此想要使用OLTP或Key-Value数据库去高效的处理分析查询场景，并不是非常完美的适用方案。例如，使用OLAP数据库去处理分析请求通常要优于使用MongoDB或Redis去处理分析请求。

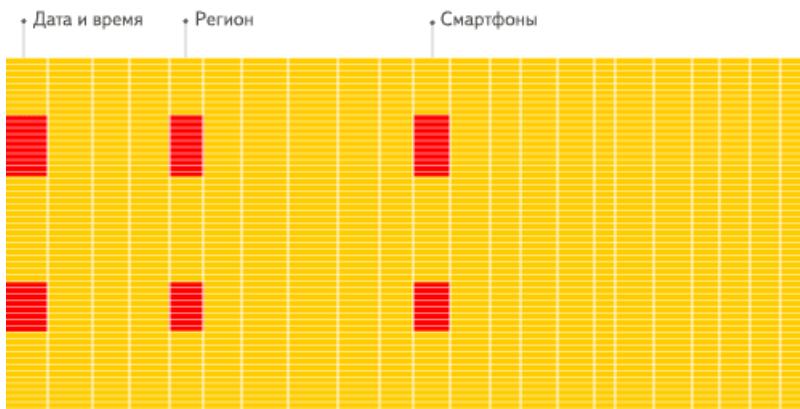
列式数据库更适合OLAP场景的原因

列式数据库更适合于OLAP场景(对于大多数查询而言，处理速度至少提高了100倍)，下面详细解释了原因(通过图片更有利干直观理解)：

行式



列式



看到差别了么？下面将详细介绍为什么会发生这种情况。

输入/输出

1. 针对分析类查询，通常只需要读取表的一小部分列。在列式数据库中你可以只读取你需要的数据。例如，如果只需要读取100列中的5列，这将帮助你最少减少20倍的I/O消耗。
2. 由于数据总是打包成批量读取的，所以压缩是非常容易的。同时数据按列分别存储这也更容易压缩。这进一步降低了I/O的体积。
3. 由于I/O的降低，这将帮助更多的数据被系统缓存。

例如，查询《统计每个广告平台的记录数量》需要读取《广告平台ID》这一列，它在未压缩的情况下需要1个字节进行存储。如果大部分流量不是来自广告平台，那么这一列至少可以以十倍的压缩率被压缩。当采用快速压缩算法，它的解压速度最少在十亿字节(未压缩数据)每秒。换句话说，这个查询可以在单个服务器上以每秒大约几十亿行的速度进行处理。这实际上是当前实现的速度。

CPU

由于执行一个查询需要处理大量的行，因此在整个向量上执行所有操作将比在每一行上执行所有操作更加高效。同时这将有助于实现一个几乎没有调用成本的查询引擎。如果你不这样做，使用任何一个机械硬盘，查询引擎都不可避免的停止CPU进行等待。所以，在数据按列存储并且按列执行是很有意义的。

有两种方法可以做到这一点：

1. 向量引擎：所有的操作都是为向量而不是为单个值编写的。这意味着多个操作之间的不再需要频繁的调用，并且调用的成本基本可以忽略不计。操作代码包含一个优化的内部循环。
2. 代码生成：生成一段代码，包含查询中的所有操作。

这是不应该在一个通用数据库中实现的，因为这在运行简单查询时是没有意义的。但是也有例外，例如，MemSQL使用代码生成来减少处理SQL查询的延迟(只是为了比较，分析型数据库通常需要优化的是吞吐而不是延迟)。

请注意，为了提高CPU效率，查询语言必须是声明型的(SQL或MDX)，或者至少一个向量(J, K)。查询应该只包含隐式循环，允许进行优化。

ClickHouse的特性

真正的列式数据库管理系统

在一个真正的列式数据库管理系统中，除了数据本身外不应该存在其他额外的数据。这意味着为了避免在值旁边存储它们的长度《number》，你必须支持固定长度数值类型。例如，10亿个UInt8类型的数据在未压缩的情况下大约消耗1GB左右的空间，如果不是这样的话，这将对CPU的使用产生强烈影响。即使是在未压缩的情况下，紧凑的存储数据也是非常重要的，因为解压缩的速度主要取决于未压缩数据的大小。

这是非常值得注意的，因为在一些其他系统中也可以将不同的列分别进行存储，但由于对其他场景进行的优化，使其无法有效的处理分析查询。例如：HBase，BigTable，Cassandra，HyperTable。在这些系统中，你可以得到每秒数十万的吞吐能力，但是无法得到每秒几亿行的吞吐能力。

需要说明的是，ClickHouse不单单是一个数据库，它是一个数据库管理系统。因为它允许在运行时创建表和数据库、加载数据和运行查询，而无需重新配置或重启服务。

数据压缩

在一些列式数据库管理系统中(例如：InfiniDB CE 和 MonetDB)并没有使用数据压缩。但是，若想达到比较优异的性能，数据压缩确实起到了至关重要的作用。

除了在磁盘空间和CPU消耗之间进行不同权衡的高效通用压缩编解码器之外，ClickHouse还提供针对特定类型数据的专用编解码器，这使得ClickHouse能够与更小的数据库(如时间序列数据库)竞争并超越它们。

数据的磁盘存储

许多的列式数据库(如 SAP HANA, Google PowerDrill)只能在内存中工作，这种方式会造成比实际更多的设备预算。

ClickHouse被设计用于工作在传统磁盘上的系统，它提供每GB更低的存储成本，但如果可以使用SSD和内存，它也会合理的利用这些资源。

多核心并行处理

ClickHouse会使用服务器上一切可用的资源，从而以最自然的方式并行处理大型查询。

多服务器分布式处理

上面提到的列式数据库管理系统中，几乎没有一个支持分布式的查询处理。

在ClickHouse中，数据可以保存在不同的shard上，每一个shard都由一组用于容错的replica组成，查询可以并行地在所有shard上进行处理。这些对用户来说是透明的

支持SQL

ClickHouse支持一种基于SQL的声明式查询语言，它在许多情况下与ANSI SQL标准相同。

支持的查询GROUP BY, ORDER BY, FROM, JOIN, IN以及非相关子查询。

相关(依赖性)子查询和窗口函数暂不受支持，但将来会被实现。

向量引擎

为了高效的使用CPU，数据不仅仅按列存储，同时还按向量(列的一部分)进行处理，这样可以更加高效地使用CPU。

实时的数据更新

ClickHouse支持在表中定义主键。为了使查询能够快速在主键中进行范围查找，数据总是以增量的方式有序的存储在MergeTree中。因此，数据可以持续不断地高效的写入到表中，并且写入的过程中不会存在任何加锁的行为。

索引

按照主键对数据进行排序，这将帮助ClickHouse在几十毫秒以内完成对数据特定值或范围的查找。

适合在线查询

在线查询意味着在没有对数据做任何预处理的情况下以极低的延迟处理查询并将结果加载到用户的页面中。

支持近似计算

ClickHouse提供各种各样的在允许牺牲数据精度的情况下对查询进行加速的方法：

1. 用于近似计算的各类聚合函数，如：distinct values, medians, quantiles
2. 基于数据的部分样本进行近似查询。这时，仅会从磁盘检索少部分比例的数据。
3. 不使用全部的聚合条件，通过随机选择有限个数据聚合条件进行聚合。这在数据聚合条件满足某些分布条件下，在提供相当准确的聚合结果的同时降低了计算资源的使用。

Adaptive Join Algorithm

ClickHouse支持自定义JOIN多个表，它更倾向于散列连接算法，如果有多个大表，则使用合并-连接算法

支持数据复制和数据完整性

ClickHouse使用异步的多主复制技术。当数据被写入任何一个可用副本后，系统会在后台将数据分发给其他副本，以保证系统在不同副本上保持相同的数据。在大多数情况下ClickHouse能在故障后自动恢复，在一些少数的复杂情况下需要手动恢复。

更多信息，参见 [数据复制](#)。

角色的访问控制

ClickHouse使用SQL查询实现用户帐户管理，并允许[角色的访问控制](#)，类似于ANSI SQL标准和流行的关系数据库管理系统。

限制

1. 没有完整的事务支持。
2. 缺少高频率，低延迟的修改或删除已存在数据的能力。仅能用于批量删除或修改数据，但这符合 [GDPR](#)。
3. 稀疏索引使得ClickHouse不适合通过其键检索单行的点查询。

性能

根据Yandex的内部测试结果，ClickHouse表现出了比同类可比较产品更优的性能。你可以在 [这里](#) 查看具体的测试结果。

许多其他的测试也证实这一点。你可以使用互联网搜索到它们，或者你也可以从 [我们收集的部分相关连接](#) 中查看。

单个大查询的吞吐量

吞吐量可以使用每秒处理的行数或每秒处理的字节数来衡量。如果数据被放置在page cache中，则一个不太复杂的查询在单个服务器上大约能够以2-10GB/s（未压缩）的速度进行处理（对于简单的查询，速度可以达到30GB/s）。如果数据没有在page cache中的话，那么速度将取决于你的磁盘系统和数据的压缩率。例如，如果一个磁盘允许以400MB/s的速度读取数据，并且数据压缩率是3，则数据的处理速度为1.2GB/s。这意味着，如果你是在提取一个10字节的列，那么它的处理速度大约是1-2亿行每秒。

对于分布式处理，处理速度几乎是线性扩展的，但这受限于聚合或排序的结果不是那么大的情况下。

处理短查询的延迟时间

如果一个查询使用主键并且没有太多行(几十万)进行处理，并且没有查询太多的列，那么在数据被page cache缓存的情况下，它的延迟应该小于50毫秒(在最佳的情况下应该小于10毫秒)。否则，延迟取决于数据的查找次数。如果你当前使用的是HDD，在数据没有加载的情况下，查询所需要的延迟可以通过以下公式计算得知：查找时间 (10 ms) * 查询的列的数量 * 查询的数据块的数量。

处理大量短查询的吞吐量

在相同的情况下，ClickHouse可以在单个服务器上每秒处理数百个查询（在最佳的情况下最多可以处理数千个）。但是由于这不适用于分析型场景。因此我们建议每秒最多查询100次。

数据的写入性能

我们建议每次写入不少于1000行的批量写入，或每秒不超过一个写入请求。当使用tab-separated格式将一份数据写入到MergeTree表中时，写入速度大约为50到200MB/s。如果您写入的数据每行为1Kb，那么写入的速度为50,000到200,000行每秒。如果您的行更小，那么写入速度将更高。为了提高写入性能，您可以使用多个INSERT进行并行写入，这将带来线性的性能提升。

ClickHouse历史

ClickHouse最初是为 [YandexMetrica 世界第二大Web分析平台](#) 而开发的。多年来一直作为该系统的核心组件被该系统持续使用着。目前为止，该系统在ClickHouse中有超过13万亿条记录，并且每天超过200多亿个事件被处理。它允许直接从原始数据中动态查询并生成报告。本文简要介绍了ClickHouse在其早期发展阶段的目标。

Yandex.Metrica基于用户定义的字段，对实时访问、连接会话，生成实时的统计报表。这种需求往往需要复杂聚合方式，比如对访问用户进行去重。构建报表的数据，是实时接收存储的新数据。

截至2014年4月，Yandex.Metrica每天跟踪大约120亿个事件（用户的点击和浏览）。为了可以创建自定义的报表，我们必须存储全部这些事件。同时，这些查询可能需要在几百毫秒内扫描数百万行的数据，或在几秒内扫描数亿行的数据。

Yandex.Metrica以及其他Yandex服务的使用案例

在Yandex.Metrica中，ClickHouse被用于多个场景中。

它的主要任务是使用原始数据在线的提供各种数据报告。它使用374台服务器的集群，存储了20.3万亿行的数据。在去除重复与副本数据的情况下，压缩后的数据达到了2PB。未压缩前（TSV格式）它大概有17PB。

ClickHouse还被使用在：

- 存储来自Yandex.Metrica的会话重放数据。
- 处理中间数据
- 与Analytics一起构建全球报表。
- 为调试Yandex.Metrica引擎运行查询
- 分析来自API和用户界面的日志数据

ClickHouse在其他Yandex服务中至少有12个安装：search verticals, Market, Direct, business analytics, mobile development, AdFox, personal services等。

聚合与非聚合数据

有一种流行的观点认为，想要有效的计算统计数据，必须要聚合数据，因为聚合将降低数据量。

但是数据聚合是一个有诸多限制的解决方案，例如：

- 你必须提前知道用户定义的报表的字段列表
- 用户无法自定义报表
- 当聚合条件过多时，可能不会减少数据，聚合是无用的。
- 存在大量报表时，有太多的聚合变化（组合爆炸）
- 当聚合条件有非常大的基数时（如：url），数据量没有太大减少（少于两倍）
- 聚合的数据量可能会增长而不是收缩
- 用户不会查看我们为他生成的所有报告，大部分计算将是无用的
- 各种聚合可能违背了数据的逻辑完整性

如果我们直接使用非聚合数据而不进行任何聚合时，我们的计算量可能是减少的。

然而，相对于聚合中很大一部分工作被离线完成，在线计算需要尽快的完成计算，因为用户在等待结果。

Yandex.Metrica有一个专门用于聚合数据的系统，称为Metrage，它可以用作大部分报表。

从2009年开始，Yandex.Metrica还为非聚合数据使用专门的OLAP数据库，称为OLAPServer，它以前用于报表构建系统。

OLAPServer可以很好的工作在非聚合数据上，但是它有诸多限制，导致无法根据需要将其用于所有报表中。如，缺少对数据类型的支持（只支持数据），无法实时增量的更新数据（只能通过每天重写数据完成）。OLAPServer不是一个数据库管理系统，它只是一个数据库。

为了消除OLAPServer的这些局限性，解决所有报表使用非聚合数据的问题，我们开发了ClickHouse数据库管理系统。

免责声明

如下使用ClickHouse的公司和他们的成功案例来源于公开资源，因此和实际情况可能有所出入。如果您分享您公司使用ClickHouse的故事，我们将不胜感激 **将其添加到列表**，但请确保你这样做不会有任何保密协议的问题。也欢迎提供来自其他公司的出版物的更新。

公司简介	行业	用例	群集大小	(Un)压缩数据大小* (of single replica)	参考资料
2gis	地图	监测	—	—	俄文，2019年7月
Aloha 浏览器	移动应用程序	浏览器后端	—	—	俄文幻灯片，2019年5月
阿玛迪斯	旅行	分析	—	—	新闻稿,四月2018
Appsflyer	移动分析	主要产品	—	—	俄文，2019年7月
ArenaData	数据平台	主要产品	—	—	俄文幻灯片，十二月2019
Badoo	约会	时间序列	—	—	俄文幻灯片，十二月2019
Benocs	网络遥测和分析	主要产品	—	—	英文幻灯片，2017年10月
彭博社	金融、媒体	监测	102个服务器	—	幻灯片，2018年5月
Bloxy	区块链	分析	—	—	俄文幻灯片，八月2018
Dataliance/UltraPower	电信	分析	—	—	中文幻灯片，2018年1月
CARTO	商业智能	地理分析	—	—	地理空间处理与ClickHouse
CERN	研究	实验	—	—	新闻稿,四月2012
思科	网络	流量分析	—	—	闪电对话，十月2019
城堡证券	金融	—	—	—	贡献，2019年3月

公司简介	行业	用例	群集大小	(Un)压缩数据大小 <u>(of single replica)</u>	参考资料
Citymobil	出租车	分析	—	—	俄文博客文章，三月 2020
内容广场	网站分析	主要产品	—	—	法文博客文章，十一月 2018
Cloudflare	CDN	流量分析	36服务器	—	博客文章,五月 2017, 博客文章,三月 2018
Corunet	分析	主要产品	—	—	英文幻灯片，2019年4月
CraeditX 氚信	金融AI	分析	—	—	英文幻灯片，2019年11月
Criteo/Storetail	零售	主要产品	—	—	英文幻灯片，十月 2018
德意志银行	金融	商业智能分析	—	—	英文幻灯片，十月 2019
Diva-e	数字咨询	主要产品	—	—	英文幻灯片，2019年9月
Exness	交易	指标，日志记录	—	—	俄语交谈，2019年5月
精灵	广告网络	主要产品	—	—	日文博客，2017年7月
虎牙	视频流	分析	—	—	中文幻灯片，2018年10月
Idealista	房地产	分析	—	—	英文博客文章,四月 2019
Infovista	网络	分析	—	—	英文幻灯片，十月 2019
InnoGames	游戏	指标，日志记录	—	—	俄文幻灯片，2019年9月

公司简介	行业	用例	群集大小	(Un)压缩数据大小 <u>(of single replica)</u>	参考资料
Integros	视频服务平台	分析	—	—	俄文幻灯片，2019年5月
科迪亚克数据	云	主要产品	—	—	虏茅驴麓卤戮碌禄路戮鲁拢
Kontur	软件开发	指标	—	—	俄语交谈，2018年11月
LifeStreet	广告网络	主要产品	75台服务器（3个副本）	5.27PiB	俄文博客文章，2017年2月
Mail.ru 云解决方案	云服务	主要产品	—	—	运行ClickHouse实例，俄语
MessageBird	电信	统计	—	—	英文幻灯片，2018年11月
MGID	广告网络	网络分析	—	—	我们在实施分析DBMS ClickHouse的经验，俄文
OneAPM	监测和数据分析	主要产品	—	—	中文幻灯片，2018年10月
Pragma Innovation	遥测和大数据分析	主要产品	—	—	英文幻灯片，十月2018
青云	云服务	主要产品	—	—	中文幻灯片，2018年10月
Qrator	DDoS保护	主要产品	—	—	博客文章,三月2019
百分点	分析	主要产品	—	—	中文幻灯片，2019年6月
漫步者	互联网服务	分析	—	—	俄语讲座，2018年4月
腾讯	通讯软件	日志记录	—	—	中文讲座，2019年11月
流量之星	广告网络	—	—	—	俄文幻灯片，2018年5月

公司简介	行业	用例	群集大小	(Un)压缩数据大小 <u>(of single replica)</u>	参考资料
S7航空公司	航空公司	指标，日志记录	—	—	俄文，2019年3月
SEMrush	营销	主要产品	—	—	俄文幻灯片，八月2018
scireum GmbH	电子商务	主要产品	—	—	德语讲座，2020年2月
Sentry	软件开发	产品后端	—	—	英文博客文章,五月2019
SGK	政府社会保障	分析	—	—	英文幻灯片，2019年11月
seo.do	分析	主要产品	—	—	英文幻灯片，2019年11月
新浪	新闻	—	—	—	中文幻灯片，2018年10月
SMI2	新闻	分析	—	—	俄文博客文章，2017年11月
Splunk	业务分析	主要产品	—	—	英文幻灯片，2018年1月
Spotify	音乐	实验	—	—	幻灯片，七月2018
腾讯	大数据	数据处理	—	—	中文幻灯片，2018年10月
腾讯QQ音乐(TME)	大数据	数据处理	—	—	博客文章，2020年6月
优步	出租车	日志记录	—	—	幻灯片，二月2020
VKontakte	社交网络	统计，日志记录	—	—	俄文幻灯片，八月2018
Wisebits	IT解决方案	分析	—	—	俄文幻灯片，2019年5月

公司简介	行业	用例	群集大小	(Un)压缩数据大小 <u>(of single replica)</u>	参考资料
晓信科技	教育	共同目的	—	—	英文幻灯片, 2019年11月
喜马拉雅	音频共享	OLAP	—	—	英文幻灯片, 2019年11月
Yandex云	公有云	主要产品	—	—	俄文, 2019年12月
Yandex DataLens	商业智能	主要产品	—	—	俄文幻灯片, 十二月2019
Yandex市场	电子商务	指标, 日志记录	—	—	俄文, 2019年1月
Yandex Metrica	网站分析	主要产品	一个集群中的360台服务器, 一个部门中的1862台服务器	66.41PiB/5.68PiB	幻灯片, 二月2020
ЦВТ	软件开发	指标, 日志记录	—	—	博客文章,三月2019, 俄文
МКБ	银行	网络系统监控	—	—	俄文幻灯片, 2019年9月
金数据	商业智能分析	主要产品	—	—	中文幻灯片, 2019年10月
Instana	APM平台	主要产品	—	—	推特消息
Wargaming	游戏		—	—	采访
Crazypanda	游戏		—	—	ClickHouse 社区会议
FunCorp	游戏		—	—	文章

入门

如果您是ClickHouse的新手，并希望亲身体验它的性能。

首先需要完成 [安装与部署](#).

之后，您可以通过教程与示例数据完成自己的入门第一步：

- [QuickStart教程](#) 快速了解Clickhouse的操作流程
- [示例数据集-航班飞行数据](#) 示例数据，提供了常用的SQL查询场景

安装

系统要求

ClickHouse可以在任何具有x86_64，AArch64或PowerPC64LE CPU架构的Linux，FreeBSD或Mac OS X上运行。

官方预构建的二进制文件通常针对x86_64进行编译，并利用SSE 4.2指令集，因此，除非另有说明，支持它的CPU使用将成为额外的系统需求。下面是检查当前CPU是否支持SSE 4.2的命令：

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

要在不支持SSE 4.2或AArch64，PowerPC64LE架构的处理器上运行ClickHouse，您应该通过适当的配置调整从[源代码构建ClickHouse](#)。

可用安装包

DEB安装包

建议使用Debian或Ubuntu的官方预编译deb软件包。运行以下命令来安装包：

```
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
```

如果您想使用最新的版本，请用testing替代stable(我们只推荐您用于测试环境)。

你也可以从这里手动下载安装包：[下载](#)。

安装包列表：

- `clickhouse-common-static` — ClickHouse编译的二进制文件。
- `clickhouse-server` — 创建clickhouse-server软连接，并安装默认配置服务
- `clickhouse-client` — 创建clickhouse-client客户端工具软连接，并安装客户端配置文件。
- `clickhouse-common-static-dbg` — 带有调试信息的ClickHouse二进制文件。

RPM安装包

推荐使用CentOS、RedHat和所有其他基于rpm的Linux发行版的官方预编译rpm包。

首先，您需要添加官方存储库：

```
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
```

如果您想使用最新的版本，请用 `testing` 替代 `stable`(我们只推荐您用于测试环境)。`prestable`有时也可用。

然后运行命令安装：

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

你也可以从这里手动下载安装包：[下载](#)。

Tgz安装包

如果您的操作系统不支持安装 `deb` 或 `rpm` 包，建议使用官方预编译的 `tgz` 软件包。

所需的版本可以通过 `curl` 或 `wget` 从存储库 <https://repo.clickhouse.com/tgz/> 下载。

下载后解压缩下载资源文件并使用安装脚本进行安装。以下是一个最新版本的安装示例：

```
export LATEST_VERSION=`curl https://api.github.com/repos/ClickHouse/ClickHouse/tags 2>/dev/null | grep -E '[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
```

对于生产环境，建议使用最新的 `stable` 版本。你可以在 GitHub 页面 <https://github.com/ClickHouse/ClickHouse/tags> 找到它，它以后缀 `-stable` 标志。

Docker安装包

要在 Docker 中运行 ClickHouse，请遵循 [Docker Hub](#) 上的指南。它是官方的 `deb` 安装包。

其他环境安装包

对于非linux操作系统和Arch64 CPU架构，ClickHouse将会以 `master` 分支的最新提交的进行编译提供(它将会有几小时的延迟)。

- **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`

下载后，您可以使用 `clickhouse client` 连接服务，或者使用 `clickhouse local` 模式处理数据，不过您必须要额外在 GitHub 下载 `server` 和 `users` 配置文件。

不建议在生产环境中使用这些构建版本，因为它们没有经过充分的测试，但是您可以自行承担这样做的风险。它们只是ClickHouse功能的一个部分。

使用源码安装

要手动编译ClickHouse，请遵循[Linux](#)或[Mac OS X](#)说明。

您可以编译并安装它们，也可以使用不安装包的程序。通过手动构建，您可以禁用[SSE 4.2](#)或[AArch64 cpu](#)。

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

您需要创建一个数据和元数据文件夹，并为所需的用户[chown](#)授权。它们的路径可以在服务器配置(`src/programs/server/config.xml`)中改变，默认情况下它们是：

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

在Gentoo上，你可以使用[emerge clickhouse](#)从源代码安装ClickHouse。

启动

如果没有[service](#)，可以运行如下命令在后台启动服务：

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

日志文件将输出在[/var/log/clickhouse-server/](#)文件夹。

如果服务器没有启动，检查[/etc/clickhouse-server/config.xml](#)中的配置。

您也可以手动从控制台启动服务器：

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

在这种情况下，日志将被打印到控制台，这在开发过程中很方便。

如果配置文件在当前目录中，则不需要指定`--config-file`参数。默认情况下，它的路径为[./config.xml](#)。

ClickHouse支持访问限制设置。它们位于[users.xml](#)文件(与[config.xml](#)同级目录)。

默认情况下，允许[default](#)用户从任何地方访问，不需要密码。可查看[user/default/networks](#)。

更多信息，请参见[Configuration Files](#)。

启动服务后，您可以使用命令行客户端连接到它：

```
$ clickhouse-client
```

默认情况下，使用[default](#)用户并不携带密码连接到[localhost:9000](#)。还可以使用`--host`参数连接到指定服务器。

终端必须使用[UTF-8](#)编码。

更多信息，请参阅[Command-line client](#)。

示例：

```
$ ./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.

:)
```

恭喜，系统已经工作了！

为了继续进行实验，你可以尝试下载测试数据集或查看[教程](#)。

ClickHouse教程

从本教程中可以获得什么？

通过学习本教程，您将了解如何设置一个简单的ClickHouse集群。它会很小，但是可以容错和扩展。然后，我们将使用其中一个示例数据集来填充数据并执行一些演示查询。

单节点设置

为了延迟演示分布式环境的复杂性，我们将首先在单个服务器或虚拟机上部署ClickHouse。ClickHouse通常是从[deb](#)或[rpm](#)包安装，但对于不支持它们的操作系统也有[其他方法](#)。

例如，您选择[deb](#)安装包，执行：

```
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
```

在我们安装的软件中包含这些包：

- `clickhouse-client` 包，包含[clickhouse-client](#)客户端，它是交互式ClickHouse控制台客户端。
- `clickhouse-common` 包，包含一个ClickHouse可执行文件。
- `clickhouse-server` 包，包含要作为服务端运行的ClickHouse配置文件。

服务器配置文件位于`/etc/clickhouse-server/`。在继续之前，请注意`config.xml`中的`<path>`元素。它决定了数据存储的位置，因此它应该位于磁盘容量的卷上；默认值是`/var/lib/clickhouse/`。如果你想调整配置，直接编辑`config`是不方便的。考虑到它可能会在将来的包更新中被重写。建议重写配置元素的方法是在配置中创建[config.d文件夹](#)，作为`config.xml`的重写方式。

你可能已经注意到了，`clickhouse-server`安装后不会自动启动。它也不会在更新后自动重新启动。您启动服务端的方式取决于您的初始系统，通常情况下是这样：

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

或

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

服务端日志的默认位置是`/var/log/clickhouse-server/`。当服务端在日志中记录Ready for connections消息，即表示服务端已准备好处理客户端连接。

一旦clickhouse-server启动并运行，我们可以利用clickhouse-client连接到服务端，并运行一些测试查询，如`SELECT "Hello, world!"`；

▶ Clickhouse-client的快速提示

导入示例数据集

现在是时候用一些示例数据填充我们的ClickHouse服务端。在本教程中，我们将使用Yandex.Metrica的匿名数据，它是在ClickHouse成为开源之前作为生产环境运行的第一个服务（关于这一点的更多内容请参阅[ClickHouse历史](#)）。[多种导入Yandex.Metrica数据集方法](#)，为了本教程，我们将使用最现实的一个。

下载并提取表数据

```
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
```

提取的文件大小约为10GB。

创建表

与大多数数据库管理系统一样，ClickHouse在逻辑上将表分组为数据库。包含一个default数据库，但我们将创建一个新的数据库tutorial：

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

与创建数据库相比，创建表的语法要复杂得多（请参阅[参考资料](#)。一般CREATE TABLE声明必须指定三个关键的事情：

1. 要创建的表的名称。
2. 表结构，例如：列名和对应的[数据类型](#)。
3. 表引擎及其设置，这决定了对此表的查询操作是如何在物理层面执行的所有细节。

Yandex.Metrica是一个网络分析服务，样本数据集不包括其全部功能，因此只有两个表可以创建：

- hits 表包含所有用户在服务所涵盖的所有网站上完成的每个操作。
- visits 表包含预先构建的会话，而不是单个操作。

让我们看看并执行这些表的实际创建表查询：

```
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,
`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)

```

您可以使用clickhouse-client的交互模式执行这些查询（只需在终端中启动它，而不需要提前指定查询）。或者如果你愿意，可以尝试一些[替代接口](#)。

正如我们所看到的，hits_v1使用 [MergeTree引擎](#)，而visits_v1使用 [Collapsing](#) 引擎。

导入数据

数据导入到ClickHouse是通过[INSERT INTO](#)方式完成的，查询类似许多SQL数据库。然而，数据通常是在一个提供[支持序列化格式](#)而不是[VALUES子句](#)（也支持）。

我们之前下载的文件是以制表符分隔的格式，所以这里是如何通过控制台客户端导入它们：

```

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有很多[要调整的设置](#)在控制台客户端中指定它们的一种方法是通过参数，就像我们看到上面语句中的--max_insert_block_size。找出可用的设置、含义及其默认值的最简单方法是查询system.settings表：

```

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."

```

您也可以[OPTIMIZE](#)导入后的表。使用MergeTree-family引擎配置的表总是在后台合并数据部分以优化数据存储（或至少检查是否有意义）。这些查询强制表引擎立即进行存储优化，而不是稍后一段时间执行：

```

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

```

这些查询开始I/O和CPU密集型操作，所以如果表一直接收到新数据，最好不要管它，让合并在后台运行。

现在我们可以检查表导入是否成功：

```

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

```

查询示例

```
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')
```

集群部署

ClickHouse集群是一个同质集群。设置步骤：

1. 在群集的所有机器上安装ClickHouse服务端
2. 在配置文件中设置集群配置
3. 在每个实例上创建本地表
4. 创建一个**分布式表**

分布式表实际上是一种view，映射到ClickHouse集群的本地表。从分布式表中执行**SELECT**查询会使用集群所有分片的资源。您可以为多个集群指定**configs**，并创建多个分布式表，为不同的集群提供视图。

具有三个分片，每个分片一个副本的集群的示例配置：

```
<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>
```

为了进一步演示，让我们使用和创建**hits_v1**表相同的**CREATE TABLE**语句创建一个新的本地表，但表名不同：

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

创建提供集群本地表视图的分布式表：

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

常见的做法是在集群的所有计算机上创建类似的分布式表。它允许在群集的任何计算机上运行分布式查询。还有一个替代选项可以使用以下方法为给定的SELECT查询创建临时分布式表[远程表功能](#)。

让我们运行INSERT SELECT将该表传播到多个服务器。

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

注意:

这种方法不适合大型表的分片。有一个单独的工具 [clickhouse-copier](#) 这可以重新分片任意大表。

正如您所期望的那样，如果计算量大的查询使用3台服务器而不是一个，则运行速度快N倍。

在这种情况下，我们使用了具有3个分片的集群，每个分片都包含一个副本。

为了在生产环境中提供弹性，我们建议每个分片应包含分布在多个可用区或数据中心（或至少机架）之间的2-3个副本。请注意，ClickHouse支持无限数量的副本。

包含三个副本的一个分片集群的示例配置：

```
<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>
```

启用本机复制[Zookeeper](#)是必需的。ClickHouse负责所有副本的数据一致性，并在失败后自动运行恢复过程。建议将ZooKeeper集群部署在单独的服务器上（其中没有其他进程，包括运行的ClickHouse）。

注意

ZooKeeper不是一个严格的要求：在某些简单的情况下，您可以通过将数据写入应用程序代码中的所有副本来自行复制数据。这种方法是不建议的，在这种情况下，ClickHouse将无法保证所有副本上的数据一致性。因此需要由您的应用来保证这一点。

ZooKeeper位置在配置文件中指定：

```
<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>
```

此外，我们需要设置宏来识别每个用于创建表的分片和副本：

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

如果在创建复制表时没有副本，则会实例化新的第一个副本。如果已有实时副本，则新副本将克隆现有副本中的数据。您可以选择首先创建所有复制的表，然后向其中插入数据。另一种选择是创建一些副本，并在数据插入之后或期间添加其他副本。

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

在这里，我们使用ReplicatedMergeTree表引擎。在参数中，我们指定包含分片和副本标识符的ZooKeeper路径。

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

复制在多主机模式下运行。数据可以加载到任何副本中，然后系统自动将其与其他实例同步。复制是异步的，因此在给定时刻，并非所有副本都可能包含最近插入的数据。至少应该有一个副本允许数据摄入。另一些则会在重新激活后同步数据并修复一致性。请注意，这种方法允许最近插入的数据丢失的可能性很低。

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](#).

示例数据集

本节介绍如何获取示例数据集并将其导入ClickHouse。对于某些数据集，还可以使用示例查询。

对于某些数据集示例查询也可用。

- [Anonymized Yandex.Metrica Dataset](#)
- [Star Schema Benchmark](#)

- [WikiStat](#)
 - [Terabyte of Click Logs from Criteo](#)
 - [AMPLab Big Data Benchmark](#)
 - [New York Taxi Data](#)
 - [OnTime](#)
-

Anonymized Yandex.Metrica Data

数据集由两个表组成，包含关于Yandex.Metrica的hits(hits_v1)和visit(visits_v1)的匿名数据。你可以阅读更多关于Yandex的信息。在[ClickHouse历史](#)的Metrica部分。

数据集由两个表组成，他们中的任何一个都可以下载作为一个压缩tsv.xz的文件或准备的分区。除此之外，一个扩展版的hits表包含1亿行TSV在https://datasets.clickhouse.com/hits/tsv/hits_100m_obfuscated_v1.tsv.xz，准备分区在https://datasets.clickhouse.com/hits/partitions/hits_100m_obfuscated_v1.tar.xz。

从准备好的分区获取表

下载和导入hits表：

```
curl -O https://datasets.clickhouse.com/hits/partitions/hits_v1.tar
tar xvf hits_v1.tar -C /var/lib/clickhouse # path to ClickHouse data directory
## check permissions on unpacked data, fix if required
sudo service clickhouse-server restart
clickhouse-client --query "SELECT COUNT(*) FROM datasets.hits_v1"
```

下载和导入visits表：

```
curl -O https://datasets.clickhouse.com/visits/partitions/visits_v1.tar
tar xvf visits_v1.tar -C /var/lib/clickhouse # path to ClickHouse data directory
## check permissions on unpacked data, fix if required
sudo service clickhouse-server restart
clickhouse-client --query "SELECT COUNT(*) FROM datasets.visits_v1"
```

从TSV压缩文件获取表

从TSV压缩文件下载并导入hits：

```
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"
```

从压缩tsv文件下载和导入visits:

```

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"

```

查询示例

[使用教程](#)是以Yandex.Metrica数据集开始教程。

可以在ClickHouse的[stateful tests](#) 中找到对这些表的查询的其他示例(它们被命名为`test.hists`和`test.visits`)。

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](#).

Star Schema Benchmark

编译 dbgen:

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

开始生成数据：

注意

使用-s 100dbgen 将生成 6 亿行数据(67GB), 如果使用-s 1000 它会生成 60 亿行数据(这需要很多时间))

```
$ ./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
```

在 ClickHouse 中创建数据表：

```

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;

```

写入数据：

```

$ 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

```

将star schema转换为flat schema：

```

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;

```

运行查询：

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;
```

WikiStat

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

创建表结构：

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

加载数据：

```
$ 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
```

Terabyte of Click Logs from Criteo

可以从 <http://labs.criteo.com/downloads/download-terabyte-click-logs/> 上下载数据

创建原始数据对应的表结构：

```
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
```

下载数据：

```
$ 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-perfet01j --query="INSERT INTO criteo_log FORMAT TabSeparated"; done
```

创建转换后的数据对应的表结构：

```

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)

```

将第一张表中的原始数据转化写入到第二张表中去：

```

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;

```

AMPLab Big Data Benchmark

参考 <https://amplab.cs.berkeley.edu/benchmark/>

需要您在Amazon注册一个免费的账号。注册时需要您提供信用卡、邮箱、电话等信息。之后可以在Amazon AWS Console获取新的访问密钥

在控制台运行以下命令：

```
$ 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 ..
```

在ClickHouse运行如下查询：

```
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);
```

回到控制台运行如下命令：

```
$ 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
```

简单的查询示例：

```
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
```

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

```

```

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
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,
  ...

```

```

at,
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,
  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;

```

-- 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 (
    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](#).

纽约出租车数据

纽约市出租车数据有以下两个方式获取：

- 从原始数据导入
- 下载处理好的数据

怎样导入原始数据

可以参考 <https://github.com/toddwschneider/nyc-taxi-data> 和 <http://tech.marksblogg.com/billion-nyc-taxi-rides-redshift.html> 中的关于数据集结构描述与数据下载指令说明。

数据集包含227GB的CSV文件。在1Gbig的带宽下，下载大约需要一个小时这大约需要一个小时的下载时间(从s3.amazonaws.com并行下载时间至少可以缩减一半)。

下载时注意损坏的文件。可以检查文件大小并重新下载损坏的文件。

有些文件中包含一些无效的行，您可以使用如下语句修复他们：

```
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
```

然后必须在PostgreSQL中对数据进行预处理。这将创建多边形中选择的点(将地图上的点与纽约市的行政区相匹配)，并使用连接将所有数据合并到一个非规范化的平面表中。为此，您需要安装支持PostGIS的PostgreSQL。

运行initialize_database.sh时要小心，并手动重新检查是否正确创建了所有表。

在PostgreSQL中处理每个月的数据大约需要20-30分钟，总共大约需要48小时。

您可以按如下方式检查下载的行数：

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

(根据Mark Litwintschik的系列博客报道数据略多余11亿行)

PostgreSQL处理这些数据大概需要370GB的磁盘空间。

从PostgreSQL中导出数据：

```

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';

```

数据快照的创建速度约为每秒50MB。在创建快照时，PostgreSQL以每秒约28MB的速度从磁盘读取数据。这大约需要5个小时。最终生成的TSV文件为590612904969 bytes。

在ClickHouse中创建临时表：

```

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_boroct2010 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_boroct2010 Nullable(String),
    dropoff_cdeligibil Nullable(String),
    dropoff_ntacode   Nullable(String),
    dropoff_ntaname   Nullable(String),
    dropoff_puma      Nullable(String)
) ENGINE = Log;

```

接下来，需要将字段转换为更正确的数据类型，并且在可能的情况下，消除NULL。

```

$ time clickhouse-client --query="INSERT INTO trips FORMAT TabSeparated" < trips.tsv
real 75m56.214s

```

数据的读取速度为112-140 Mb/秒。

通过这种方式将数据加载到Log表中需要76分钟。

这个表中的数据需要使用142GB的磁盘空间。

(也可以直接使用COPY ... TO PROGRAM从Postgres中导入数据)

数据中所有与天气相关的字段(precipitation.....average_wind_speed)都填充了NULL。所以，我们将从最终数据集中删除它们

首先，我们使用单台服务器创建表，后面我们将在多台节点上创建这些表。

创建表结构并写入数据：

```
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, 'Pleasant Plains' = 111, 'Ridgewood' = 112, 'Rockaway Beach' = 113, 'South Bronx' = 114, 'South Brooklyn' = 115, 'South Bronx' = 116, 'South Brooklyn' = 117, 'South Bronx' = 118, 'South Brooklyn' = 119, 'South Bronx' = 120, 'South Brooklyn' = 121, 'South Bronx' = 122, 'South Brooklyn' = 123, 'South Bronx' = 124, 'South Brooklyn' = 125, 'South Bronx' = 126, 'South Brooklyn' = 127, 'South Bronx' = 128, 'South Brooklyn' = 129, 'South Bronx' = 130, 'South Brooklyn' = 131, 'South Bronx' = 132, 'South Brooklyn' = 133, 'South Bronx' = 134, 'South Brooklyn' = 135, 'South Bronx' = 136, 'South Brooklyn' = 137, 'South Bronx' = 138, 'South Brooklyn' = 139, 'South Bronx' = 140, 'South Brooklyn' = 141, 'South Bronx' = 142, 'South Brooklyn' = 143, 'South Bronx' = 144, 'South Brooklyn' = 145, 'South Bronx' = 146, 'South Brooklyn' = 147, 'South Bronx' = 148, 'South Brooklyn' = 149, 'South Bronx' = 150, 'South Brooklyn' = 151, 'South Bronx' = 152, 'South Brooklyn' = 153, 'South Bronx' = 154, 'South Brooklyn' = 155, 'South Bronx' = 156, 'South Brooklyn' = 157, 'South Bronx' = 158, 'South Brooklyn' = 159, 'South Bronx' = 160, 'South Brooklyn' = 161, 'South Bronx' = 162, 'South Brooklyn' = 163, 'South Bronx' = 164, 'South Brooklyn' = 165, 'South Bronx' = 166, 'South Brooklyn' = 167, 'South Bronx' = 168, 'South Brooklyn' = 169, 'South Bronx' = 170, 'South Brooklyn' = 171, 'South Bronx' = 172, 'South Brooklyn' = 173, 'South Bronx' = 174, 'South Brooklyn' = 175, 'South Bronx' = 176, 'South Brooklyn' = 177, 'South Bronx' = 178, 'South Brooklyn' = 179, 'South Bronx' = 180, 'South Brooklyn' = 181, 'South Bronx' = 182, 'South Brooklyn' = 183, 'South Bronx' = 184, 'South Brooklyn' = 185, 'South Bronx' = 186, 'South Brooklyn' = 187, 'South Bronx' = 188, 'South Brooklyn' = 189, 'South Bronx' = 190, 'South Brooklyn' = 191, 'South Bronx' = 192, 'South Brooklyn' = 193, 'South Bronx' = 194, 'South Brooklyn' = 195, 'South Bronx' = 196, 'South Brooklyn' = 197, 'South Bronx' = 198, 'South Brooklyn' = 199, 'South Bronx' = 200, 'South Brooklyn' = 201, 'South Bronx' = 202, 'South Brooklyn' = 203, 'South Bronx' = 204, 'South Brooklyn' = 205, 'South Bronx' = 206, 'South Brooklyn' = 207, 'South Bronx' = 208, 'South Brooklyn' = 209, 'South Bronx' = 210, 'South Brooklyn' = 211, 'South Bronx' = 212, 'South Brooklyn' = 213, 'South Bronx' = 214, 'South Brooklyn' = 215, 'South Bronx' = 216, 'South Brooklyn' = 217, 'South Bronx' = 218, 'South Brooklyn' = 219, 'South Bronx' = 220, 'South Brooklyn' = 221, 'South Bronx' = 222, 'South Brooklyn' = 223, 'South Bronx' = 224, 'South Brooklyn' = 225, 'South Bronx' = 226, 'South Brooklyn' = 227, 'South Bronx' = 228, 'South Brooklyn' = 229, 'South Bronx' = 230, 'South Brooklyn' = 231, 'South Bronx' = 232, 'South Brooklyn' = 233, 'South Bronx' = 234, 'South Brooklyn' = 235, 'South Bronx' = 236, 'South Brooklyn' = 237, 'South Bronx' = 238, 'South Brooklyn' = 239, 'South Bronx' = 240, 'South Brooklyn' = 241, 'South Bronx' = 242, 'South Brooklyn' = 243, 'South Bronx' = 244, 'South Brooklyn' = 245, 'South Bronx' = 246, 'South Brooklyn' = 247, 'South Bronx' = 248, 'South Brooklyn' = 249, 'South Bronx' = 250, 'South Brooklyn' = 251, 'South Bronx' = 252, 'South Brooklyn' = 253, 'South Bronx' = 254, 'South Brooklyn' = 255, 'South Bronx' = 256, 'South Brooklyn' = 257, 'South Bronx' = 258, 'South Brooklyn' = 259, 'South Bronx' = 260, 'South Brooklyn' = 261, 'South Bronx' = 262, 'South Brooklyn' = 263, 'South Bronx' = 264, 'South Brooklyn' = 265, 'South Bronx' = 266, 'South Brooklyn' = 267, 'South Bronx' = 268, 'South Brooklyn' = 269, 'South Bronx' = 270, 'South Brooklyn' = 271, 'South Bronx' = 272, 'South Brooklyn' = 273, 'South Bronx' = 274, 'South Brooklyn' = 275, 'South Bronx' = 276, 'South Brooklyn' = 277, 'South Bronx' = 278, 'South Brooklyn' = 279, 'South Bronx' = 280, 'South Brooklyn' = 281, 'South Bronx' = 282, 'South Brooklyn' = 283, 'South Bronx' = 284, 'South Brooklyn' = 285, 'South Bronx' = 286, 'South Brooklyn' = 287, 'South Bronx' = 288, 'South Brooklyn' = 289, 'South Bronx' = 290, 'South Brooklyn' = 291, 'South Bronx' = 292, 'South Brooklyn' = 293, 'South Bronx' = 294, 'South Brooklyn' = 295, 'South Bronx' = 296, 'South Brooklyn' = 297, 'South Bronx' = 298, 'South Brooklyn' = 299, 'South Bronx' = 300, 'South Brooklyn' = 301, 'South Bronx' = 302, 'South Brooklyn' = 303, 'South Bronx' = 304, 'South Brooklyn' = 305, 'South Bronx' = 306, 'South Brooklyn' = 307, 'South Bronx' = 308, 'South Brooklyn' = 309, 'South Bronx' = 310, 'South Brooklyn' = 311, 'South Bronx' = 312, 'South Brooklyn' = 313, 'South Bronx' = 314, 'South Brooklyn' = 315, 'South Bronx' = 316, 'South Brooklyn' = 317, 'South Bronx' = 318, 'South Brooklyn' = 319, 'South Bronx' = 320, 'South Brooklyn' = 321, 'South Bronx' = 322, 'South Brooklyn' = 323, 'South Bronx' = 324, 'South Brooklyn' = 325, 'South Bronx' = 326, 'South Brooklyn' = 327, 'South Bronx' = 328, 'South Brooklyn' = 329, 'South Bronx' = 330, 'South Brooklyn' = 331, 'South Bronx' = 332, 'South Brooklyn' = 333, 'South Bronx' = 334, 'South Brooklyn' = 335, 'South Bronx' = 336, 'South Brooklyn' = 337, 'South Bronx' = 338, 'South Brooklyn' = 339, 'South Bronx' = 340, 'South Brooklyn' = 341, 'South Bronx' = 342, 'South Brooklyn' = 343, 'South Bronx' = 344, 'South Brooklyn' = 345, 'South Bronx' = 346, 'South Brooklyn' = 347, 'South Bronx' = 348, 'South Brooklyn' = 349, 'South Bronx' = 350, 'South Brooklyn' = 351, 'South Bronx' = 352, 'South Brooklyn' = 353, 'South Bronx' = 354, 'South Brooklyn' = 355, 'South Bronx' = 356, 'South Brooklyn' = 357, 'South Bronx' = 358, 'South Brooklyn' = 359, 'South Bronx' = 360, 'South Brooklyn' = 361, 'South Bronx' = 362, 'South Brooklyn' = 363, 'South Bronx' = 364, 'South Brooklyn' = 365, 'South Bronx' = 366, 'South Brooklyn' = 367, 'South Bronx' = 368, 'South Brooklyn' = 369, 'South Bronx' = 370, 'South Brooklyn' = 371, 'South Bronx' = 372, 'South Brooklyn' = 373, 'South Bronx' = 374, 'South Brooklyn' = 375, 'South Bronx' = 376, 'South Brooklyn' = 377, 'South Bronx' = 378, 'South Brooklyn' = 379, 'South Bronx' = 380, 'South Brooklyn' = 381, 'South Bronx' = 382, 'South Brooklyn' = 383, 'South Bronx' = 384, 'South Brooklyn' = 385, 'South Bronx' = 386, 'South Brooklyn' = 387, 'South Bronx' = 388, 'South Brooklyn' = 389, 'South Bronx' = 390, 'South Brooklyn' = 391, 'South Bronx' = 392, 'South Brooklyn' = 393, 'South Bronx' = 394, 'South Brooklyn' = 395, 'South Bronx' = 396, 'South Brooklyn' = 397, 'South Bronx' = 398, 'South Brooklyn' = 399, 'South Bronx' = 400, 'South Brooklyn' = 401, 'South Bronx' = 402, 'South Brooklyn' = 403, 'South Bronx' = 404, 'South Brooklyn' = 405, 'South Bronx' = 406, 'South Brooklyn' = 407, 'South Bronx' = 408, 'South Brooklyn' = 409, 'South Bronx' = 410, 'South Brooklyn' = 411, 'South Bronx' = 412, 'South Brooklyn' = 413, 'South Bronx' = 414, 'South Brooklyn' = 415, 'South Bronx' = 416, 'South Brooklyn' = 417, 'South Bronx' = 418, 'South Brooklyn' = 419, 'South Bronx' = 420, 'South Brooklyn' = 421, 'South Bronx' = 422, 'South Brooklyn' = 423, 'South Bronx' = 424, 'South Brooklyn' = 425, 'South Bronx' = 426, 'South Brooklyn' = 427, 'South Bronx' = 428, 'South Brooklyn' = 429, 'South Bronx' = 430, 'South Brooklyn' = 431, 'South Bronx' = 432, 'South Brooklyn' = 433, 'South Bronx' = 434, 'South Brooklyn' = 435, 'South Bronx' = 436, 'South Brooklyn' = 437, 'South Bronx' = 438, 'South Brooklyn' = 439, 'South Bronx' = 440, 'South Brooklyn' = 441, 'South Bronx' = 442, 'South Brooklyn' = 443, 'South Bronx' = 444, 'South Brooklyn' = 445, 'South Bronx' = 446, 'South Brooklyn' = 447, 'South Bronx' = 448, 'South Brooklyn' = 449, 'South Bronx' = 450, 'South Brooklyn' = 451, 'South Bronx' = 452, 'South Brooklyn' = 453, 'South Bronx' = 454, 'South Brooklyn' = 455, 'South Bronx' = 456, 'South Brooklyn' = 457, 'South Bronx' = 458, 'South Brooklyn' = 459, 'South Bronx' = 460, 'South Brooklyn' = 461, 'South Bronx' = 462, 'South Brooklyn' = 463, 'South Bronx' = 464, 'South Brooklyn' = 465, 'South Bronx' = 466, 'South Brooklyn' = 467, 'South Bronx' = 468, 'South Brooklyn' = 469, 'South Bronx' = 470, 'South Brooklyn' = 471, 'South Bronx' = 472, 'South Brooklyn' = 473, 'South Bronx' = 474, 'South Brooklyn' = 475, 'South Bronx' = 476, 'South Brooklyn' = 477, 'South Bronx' = 478, 'South Brooklyn' = 479, 'South Bronx' = 480, 'South Brooklyn' = 481, 'South Bronx' = 482, 'South Brooklyn' = 483, 'South Bronx' = 484, 'South Brooklyn' = 485, 'South Bronx' = 486, 'South Brooklyn' = 487, 'South Bronx' = 488, 'South Brooklyn' = 489, 'South Bronx' = 490, 'South Brooklyn' = 491, 'South Bronx' = 492, 'South Brooklyn' = 493, 'South Bronx' = 494, 'South Brooklyn' = 495, 'South Bronx' = 496, 'South Brooklyn' = 497, 'South Bronx' = 498, 'South Brooklyn' = 499, 'South Bronx' = 500, 'South Brooklyn' = 501, 'South Bronx' = 502, 'South Brooklyn' = 503, 'South Bronx' = 504, 'South Brooklyn' = 505, 'South Bronx' = 506, 'South Brooklyn' = 507, 'South Bronx' = 508, 'South Brooklyn' = 509, 'South Bronx' = 510, 'South Brooklyn' = 511, 'South Bronx' = 512, 'South Brooklyn' = 513, 'South Bronx' = 514, 'South Brooklyn' = 515, 'South Bronx' = 516, 'South Brooklyn' = 517, 'South Bronx' = 518, 'South Brooklyn' = 519, 'South Bronx' = 520, 'South Brooklyn' = 521, 'South Bronx' = 522, 'South Brooklyn' = 523, 'South Bronx' = 524, 'South Brooklyn' = 525, 'South Bronx' = 526, 'South Brooklyn' = 527, 'South Bronx' = 528, 'South Brooklyn' = 529, 'South Bronx' = 530, 'South Brooklyn' = 531, 'South Bronx' = 532, 'South Brooklyn' = 533, 'South Bronx' = 534, 'South Brooklyn' = 535, 'South Bronx' = 536, 'South Brooklyn' = 537, 'South Bronx' = 538, 'South Brooklyn' = 539, 'South Bronx' = 540, 'South Brooklyn' = 541, 'South Bronx' = 542, 'South Brooklyn' = 543, 'South Bronx' = 544, 'South Brooklyn' = 545, 'South Bronx' = 546, 'South Brooklyn' = 547, 'South Bronx' = 548, 'South Brooklyn' = 549, 'South Bronx' = 550, 'South Brooklyn' = 551, 'South Bronx' = 552, 'South Brooklyn' = 553, 'South Bronx' = 554, 'South Brooklyn' = 555, 'South Bronx' = 556, 'South Brooklyn' = 557, 'South Bronx' = 558, 'South Brooklyn' = 559, 'South Bronx' = 560, 'South Brooklyn' = 561, 'South Bronx' = 562, 'South Brooklyn' = 563, 'South Bronx' = 564, 'South Brooklyn' = 565, 'South Bronx' = 566, 'South Brooklyn' = 567, 'South Bronx' = 568, 'South Brooklyn' = 569, 'South Bronx' = 570, 'South Brooklyn' = 571, 'South Bronx' = 572, 'South Brooklyn' = 573, 'South Bronx' = 574, 'South Brooklyn' = 575, 'South Bronx' = 576, 'South Brooklyn' = 577, 'South Bronx' = 578, 'South Brooklyn' = 579, 'South Bronx' = 580, 'South Brooklyn' = 581, 'South Bronx' = 582, 'South Brooklyn' = 583, 'South Bronx' = 584, 'South Brooklyn' = 585, 'South Bronx' = 586, 'South Brooklyn' = 587, 'South Bronx' = 588, 'South Brooklyn' = 589, 'South Bronx' = 590, 'South Brooklyn' = 591, 'South Bronx' = 592, 'South Brooklyn' = 593, 'South Bronx' = 594, 'South Brooklyn' = 595, 'South Bronx' = 596, 'South Brooklyn' = 597, 'South Bronx' = 598, 'South Brooklyn' = 599, 'South Bronx' = 600, 'South Brooklyn' = 601, 'South Bronx' = 602, 'South Brooklyn' = 603, 'South Bronx' = 604, 'South Brooklyn' = 605, 'South Bronx' = 606, 'South Brooklyn' = 607, 'South Bronx' = 608, 'South Brooklyn' = 609, 'South Bronx' = 610, 'South Brooklyn' = 611, 'South Bronx' = 612, 'South Brooklyn' = 613, 'South Bronx' = 614, 'South Brooklyn' = 615, 'South Bronx' = 616, 'South Brooklyn' = 617, 'South Bronx' = 618, 'South Brooklyn' = 619, 'South Bronx' = 620, 'South Brooklyn' = 621, 'South Bronx' = 622, 'South Brooklyn' = 623, 'South Bronx' = 624, 'South Brooklyn' = 625, 'South Bronx' = 626, 'South Brooklyn' = 627, 'South Bronx' = 628, 'South Brooklyn' = 629, 'South Bronx' = 630, 'South Brooklyn' = 631, 'South Bronx' = 632, 'South Brooklyn' = 633, 'South Bronx' = 634, 'South Brooklyn' = 635, 'South Bronx' = 636, 'South Brooklyn' = 637, 'South Bronx' = 638, 'South Brooklyn' = 639, 'South Bronx' = 640, 'South Brooklyn' = 641, 'South Bronx' = 642, 'South Brooklyn' = 643, 'South Bronx' = 644, 'South Brooklyn' = 645, 'South Bronx' = 646, 'South Brooklyn' = 647, 'South Bronx' = 648, 'South Brooklyn' = 649, 'South Bronx' = 650, 'South Brooklyn' = 651, 'South Bronx' = 652, 'South Brooklyn' = 653, 'South Bronx' = 654, 'South Brooklyn' = 655, 'South Bronx' = 656, 'South Brooklyn' = 657, 'South Bronx' = 658, 'South Brooklyn' = 659, 'South Bronx' = 660, 'South Brooklyn' = 661, 'South Bronx' = 662, 'South Brooklyn' = 663, 'South Bronx' = 664, 'South Brooklyn' = 665, 'South Bronx' = 666, 'South Brooklyn' = 667, 'South Bronx' = 668, 'South Brooklyn' = 669, 'South Bronx' = 670, 'South Brooklyn' = 671, 'South Bronx' = 672, 'South Brooklyn' = 673, 'South Bronx' = 674, 'South Brooklyn' = 675, 'South Bronx' = 676, 'South Brooklyn' = 677, 'South Bronx' = 678, 'South Brooklyn' = 679, 'South Bronx' = 680, 'South Brooklyn' = 681, 'South Bronx' = 682, 'South Brooklyn' = 683, 'South Bronx' = 684, 'South Brooklyn' = 685, 'South Bronx' = 686, 'South Brooklyn' = 687, 'South Bronx' = 688, 'South Brooklyn' = 689, 'South Bronx' = 690, 'South Brooklyn' = 691, 'South Bronx' = 692, 'South Brooklyn' = 693, 'South Bronx' = 694, 'South Brooklyn' = 695, 'South Bronx' = 696, 'South Brooklyn' = 697, 'South Bronx' = 698, 'South Brooklyn' = 699, 'South Bronx' = 700, 'South Brooklyn' = 701, 'South Bronx' = 702, 'South Brooklyn' = 703, 'South Bronx' = 704, 'South Brooklyn' = 705, 'South Bronx' = 706, 'South Brooklyn' = 707, 'South Bronx' = 708, 'South Brooklyn' = 709, 'South Bronx' = 710, 'South Brooklyn' = 711, 'South Bronx' = 712, 'South Brooklyn' = 713, 'South Bronx' = 714, 'South Brooklyn' = 715, 'South Bronx' = 716, 'South Brooklyn' = 717, 'South Bronx' = 718, 'South Brooklyn' = 719, 'South Bronx' = 720, 'South Brooklyn' = 721, 'South Bronx' = 722, 'South Brooklyn' = 723, 'South Bronx' = 724, 'South Brooklyn' = 725, 'South Bronx' = 726, 'South Brooklyn' = 727, 'South Bronx' = 728, 'South Brooklyn' = 729, 'South Bronx' = 730, 'South Brooklyn' = 731, 'South Bronx' = 732, 'South Brooklyn' = 733, 'South Bronx' = 734, 'South Brooklyn' = 735, 'South Bronx' = 736, 'South Brooklyn' = 737, 'South Bronx' = 738, 'South Brooklyn' = 739, 'South Bronx' = 740, 'South Brooklyn' = 741, 'South Bronx' = 742, 'South Brooklyn' = 743, 'South Bronx' = 744, 'South Brooklyn' = 745, 'South Bronx' = 746, 'South Brooklyn' = 747, 'South Bronx' = 748, 'South Brooklyn' = 749, 'South Bronx' = 750, 'South Brooklyn' = 751, 'South Bronx' = 752, 'South Brooklyn' = 753, 'South Bronx' = 754, 'South Brooklyn' = 755, 'South Bronx' = 756, 'South Brooklyn' = 757, 'South Bronx' = 758, 'South Brooklyn' = 759, 'South Bronx' = 760, 'South Brooklyn' = 761, 'South Bronx' = 762, 'South Brooklyn' = 763, 'South Bronx' = 764, 'South Brooklyn' = 765, 'South Bronx' = 766, 'South Brooklyn' = 767, 'South Bronx' = 768, 'South Brooklyn' = 769, 'South Bronx' = 770, 'South Brooklyn' = 771, 'South Bronx' = 772, 'South Brooklyn' = 773, 'South Bronx' = 774, 'South Brooklyn' = 775, 'South Bronx' = 776, 'South Brooklyn' = 777, 'South Bronx' = 778, 'South Brooklyn' = 779, 'South Bronx' = 780, 'South Brooklyn' = 781, 'South Bronx' = 782, 'South Brooklyn' = 783, 'South Bronx' = 784, 'South Brooklyn' = 785, 'South Bronx' = 786, 'South Brooklyn' = 787, 'South Bronx' = 788, 'South Brooklyn' = 789, 'South Bronx' = 790, 'South Brooklyn' = 791, 'South Bronx' = 792, 'South Brooklyn' = 793, 'South Bronx' = 794, 'South Brooklyn' = 795, 'South Bronx' = 796, 'South Brooklyn' = 797, 'South Bronx' = 798, 'South Brooklyn' = 799, 'South Bronx' = 800, 'South Brooklyn' = 801, 'South Bronx' = 802, 'South Brooklyn' = 803, 'South Bronx' = 804, 'South Brooklyn' = 805, 'South Bronx' = 806, 'South Brooklyn' = 807, 'South Bronx' = 808, 'South Brooklyn' = 809, 'South Bronx' = 810, 'South Brooklyn' = 811, 'South Bronx' = 812, 'South Brooklyn' = 813, 'South Bronx' = 814, 'South Brooklyn' = 815, 'South Bronx' = 816, 'South Brooklyn' = 817, 'South Bronx' = 818, 'South Brooklyn' = 819, 'South Bronx' = 820, 'South Brooklyn' = 821, 'South Bronx' = 822, 'South Brooklyn' = 823, 'South Bronx' = 824, 'South Brooklyn' = 825, 'South Bronx' = 826, 'South Brooklyn' = 827, 'South Bronx' = 828, 'South Brooklyn' = 829, 'South Bronx' = 830, 'South Brooklyn' = 831, 'South Bronx' = 832, 'South Brooklyn' = 833, 'South Bronx' = 834, 'South Brooklyn' = 835, 'South Bronx' = 836, 'South Brooklyn' = 837, 'South Bronx' = 838, 'South Brooklyn' = 839, 'South Bronx' = 840, 'South Brooklyn' = 841, 'South Bronx' = 842, 'South Brooklyn' = 843, 'South Bronx' = 844, 'South Brooklyn' = 845, 'South Bronx' = 846, 'South Brooklyn' = 847, 'South Bronx' = 848, 'South Brooklyn' = 849, 'South Bronx' = 850, 'South Brooklyn' = 851, 'South Bronx' = 852, 'South Brooklyn' = 853, 'South Bronx' = 854, 'South Brooklyn' = 855, 'South Bronx' = 856, 'South Brooklyn' = 857, 'South Bronx' = 858, 'South Brooklyn' = 859, 'South Bronx' = 860, 'South Brooklyn' = 861, 'South Bronx' = 862, 'South Brooklyn' = 863, 'South Bronx' = 864, 'South Brooklyn' = 865, 'South Bronx' = 866, 'South Brooklyn' = 867, 'South Bronx' = 868, 'South Brooklyn' = 869, 'South Bronx' = 870, 'South Brooklyn' = 871, 'South Bronx' = 872, 'South Brooklyn' = 873, 'South Bronx' = 874, 'South Brooklyn' = 875, 'South Bronx' = 876, 'South Brooklyn' = 877, 'South Bronx' = 878, 'South Brooklyn' = 879, 'South Bronx' = 880, 'South Brooklyn' = 881, 'South Bronx' = 882, 'South Brooklyn' = 883, 'South Bronx' = 884, 'South Brooklyn' = 885, 'South Bronx' = 886, 'South Brooklyn' = 887, 'South Bronx' = 888, 'South Brooklyn' = 889, 'South Bronx' = 890, 'South Brooklyn' = 891, 'South Bronx' = 892, 'South Brooklyn' = 893, 'South Bronx' = 894, 'South Brooklyn' = 895, 'South Bronx' = 896, 'South Brooklyn' = 897, 'South Bronx' = 898, 'South Brooklyn' = 899, 'South Bronx' = 900, 'South Brooklyn' = 901, 'South Bronx' = 902, 'South Brooklyn' = 903, 'South Bronx' = 904, 'South Brooklyn' = 905, 'South Bronx' = 906, 'South Brooklyn' = 907, 'South Bronx' = 908, 'South Brooklyn' = 909, 'South Bronx' = 910, 'South Brooklyn' = 911, 'South Bronx' = 912, 'South Brooklyn' = 913, 'South Bronx' = 914, 'South Brooklyn' = 915, 'South Bronx' = 916, 'South Brooklyn' = 917, 'South Bronx' = 918, 'South Brooklyn' = 919, 'South Bronx' = 920, 'South Brooklyn' = 921, 'South Bronx' = 922, 'South Brooklyn' = 923, 'South Bronx' = 924, 'South Brooklyn' = 925, 'South Bronx' = 926, 'South Brooklyn' = 927, 'South Bronx' = 928, 'South Brooklyn' = 929, 'South Bronx' = 930, 'South Brooklyn' = 931, 'South Bronx' = 932, 'South Brooklyn' = 933, 'South Bronx' = 934, 'South Brooklyn' = 935, 'South Bronx' = 936, 'South Brooklyn' = 937, 'South Bronx' = 938, 'South Brooklyn' = 939, 'South Bronx' = 940, 'South Brooklyn' = 941, 'South Bronx' = 942, 'South Brooklyn' = 943, 'South Bronx' = 944, 'South Brooklyn' = 945, 'South Bronx' = 946, 'South Brooklyn' = 947, 'South Bronx' = 948, 'South Brooklyn' = 949, 'South Bronx' = 950, 'South Brooklyn' = 951, 'South Bronx' = 952, 'South Brooklyn' = 953, 'South Bronx' = 954, 'South Brooklyn' = 955, 'South Bronx' = 956, 'South Brooklyn' = 957, 'South Bronx' = 958, 'South Brooklyn' = 959, 'South Bronx' = 960, 'South Brooklyn' = 961, 'South Bronx' = 962, 'South Brooklyn' = 963, 'South Bronx' = 964, 'South Brooklyn' = 965, 'South Bronx' = 966, 'South Brooklyn' = 967, 'South Bronx' = 968, 'South Brooklyn' = 969, 'South Bronx' = 970, 'South Brooklyn' = 971, 'South Bronx' = 972, 'South Brooklyn' = 973, 'South Bronx' = 974, 'South Brooklyn' = 975, 'South Bronx' = 976, 'South Brooklyn' = 977, 'South Bronx' = 978, 'South Brooklyn' = 979, 'South Bronx' = 980, 'South Brooklyn' = 981, 'South Bronx' = 982, 'South Brooklyn' = 983, 'South Bronx' = 984, 'South Brooklyn' = 985, 'South Bronx' = 986, 'South Brooklyn' = 987, 'South Bronx' = 988, 'South Brooklyn' = 989, 'South Bronx' = 990, 'South Brooklyn' = 991, 'South Bronx' = 992, 'South Brooklyn' = 993, 'South Bronx' = 994, '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
```

这需要3030秒，速度约为每秒428,000行。

要加快速度，可以使用Log引擎替换MergeTree引擎来创建表。在这种情况下，下载速度超过200秒。

这个表需要使用126GB的磁盘空间。

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

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

除此之外，你还可以在MergeTree上运行OPTIMIZE查询来进行优化。但这不是必须的，因为即使在没有进行优化的情况下它的表现依然是很好的。

下载预处理好的分区数据

```
$ 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"
```

信息

如果要运行下面的SQL查询，必须使用完整的表名，`datasets.trips_mergetree`。

单台服务器运行结果

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

我们使用的是如下配置的服务器：

两个 Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz，总共有16个物理内核，128GiB RAM，8X6TB HD，RAID-5

执行时间是取三次运行中最好的值，但是从第二次查询开始，查询就将从文件系统的缓存中读取数据。同时在每次读取和处理后不在进行缓存。

在三台服务器中创建表结构：

在每台服务器中运行：

```
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_boroc2010 FixedString(7),  
pickup_cdeligibil Enum8('0' = 0, 'E' = 1, 'I' = 2), pickup_ntacode FixedString(4), pickup_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, 'Bellrose' = 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-Doulaston-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), 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, '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-Doulaston-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), dropoff_puma UInt16) ENGINE = MergeTree(pickup_date, pickup_datetime, 8192)

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

运行如下查询重新分布数据：

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

这个查询需要运行2454秒。

在三台服务器集群中运行的结果：

Q1: 0.212秒.

Q2 : 0.438秒。

Q3 : 0.733秒。

Q4: 1.241秒.

这并不奇怪，因为查询是线性扩展的。

我们同时在140台服务器的集群中运行的结果：

Q1 : 0.028秒。

Q2 : 0.043秒。

Q3 : 0.051秒。

Q4 : 0.072秒。

在这种情况下，查询处理时间首先由网络延迟确定。

我们使用位于芬兰Yandex数据中心的客户机在俄罗斯的一个集群上运行查询，这增加了大约20毫秒的延迟。

总结

服务器	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

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

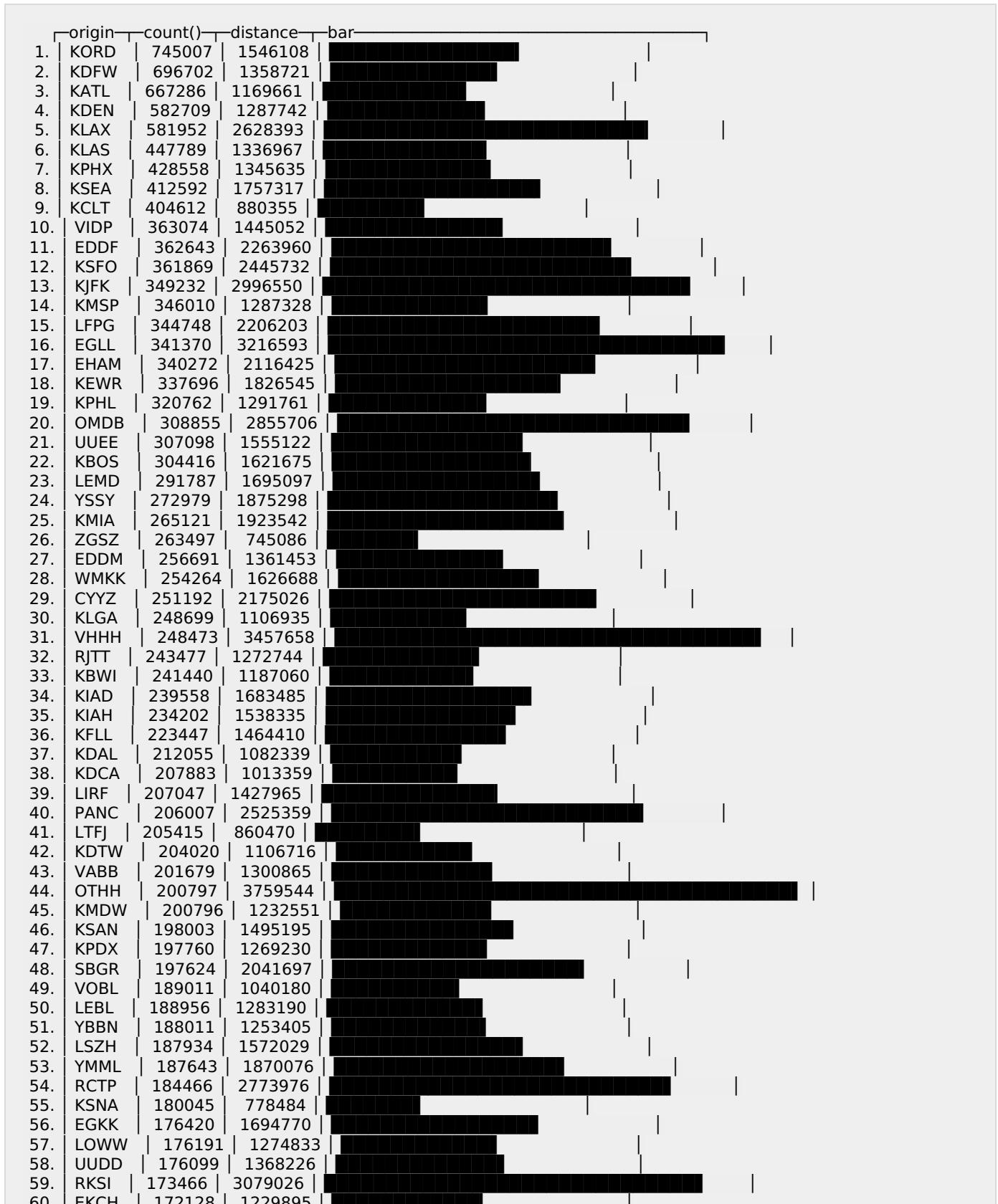
Query:

```

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:





Number of flights from three major Moscow airports, weekly

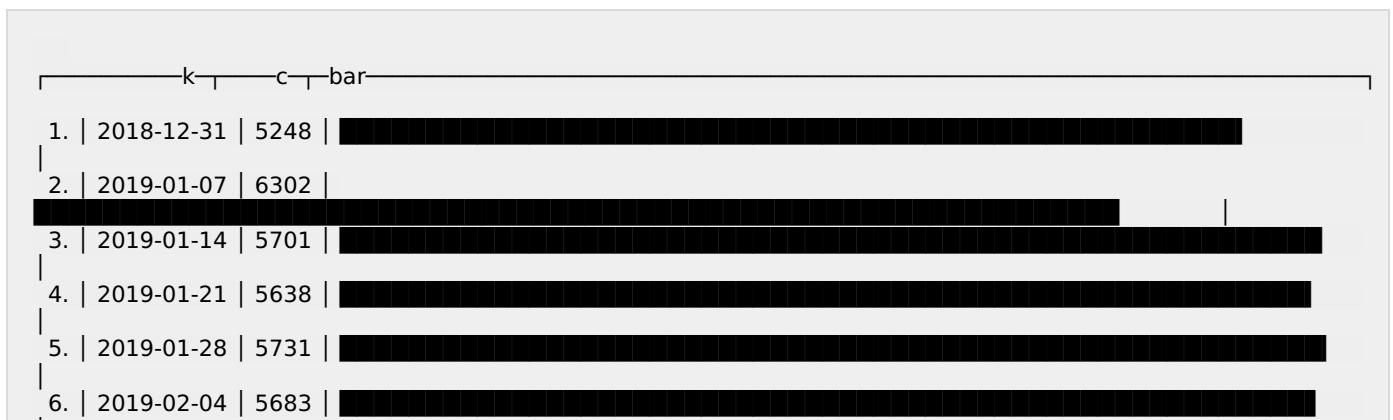
Query:

```

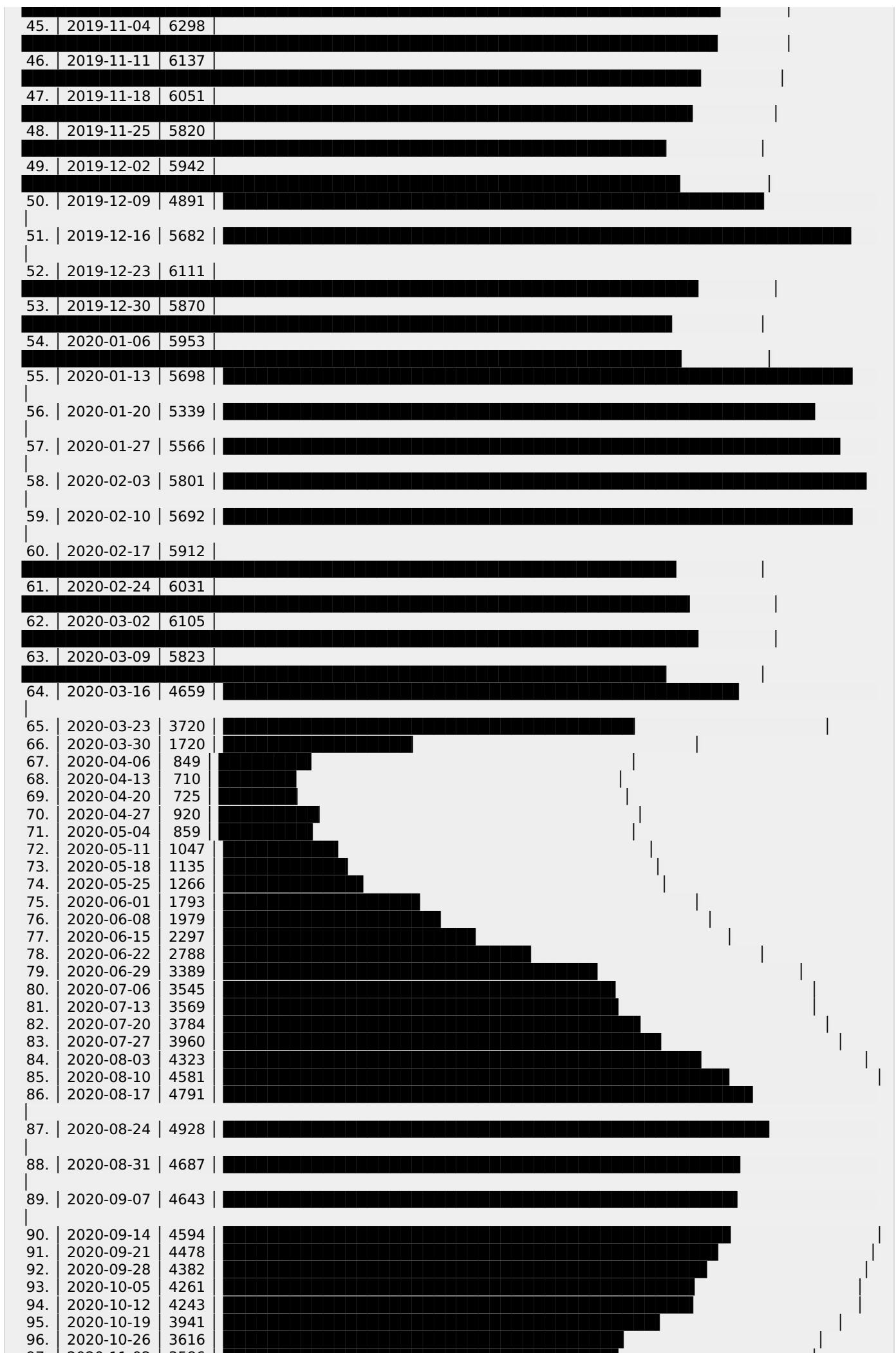
SELECT
    toMonday(day) AS k,
    count() AS c,
    bar(c, 0, 10000, 100) AS bar
FROM opensky
WHERE origin IN ('UUEE', 'UUDD', 'UUWW')
GROUP BY k
ORDER BY k ASC;

```

Result:



7.	2019-02-11	5759	[REDACTED]
8.	2019-02-18	5736	[REDACTED]
9.	2019-02-25	5873	[REDACTED]
10.	2019-03-04	5965	[REDACTED]
11.	2019-03-11	5900	[REDACTED]
12.	2019-03-18	5823	[REDACTED]
13.	2019-03-25	5899	[REDACTED]
14.	2019-04-01	6043	[REDACTED]
15.	2019-04-08	6098	[REDACTED]
16.	2019-04-15	6196	[REDACTED]
17.	2019-04-22	6486	[REDACTED]
18.	2019-04-29	6682	[REDACTED]
19.	2019-05-06	6739	[REDACTED]
20.	2019-05-13	6600	[REDACTED]
21.	2019-05-20	6575	[REDACTED]
22.	2019-05-27	6786	[REDACTED]
23.	2019-06-03	6872	[REDACTED]
24.	2019-06-10	7045	[REDACTED]
25.	2019-06-17	7045	[REDACTED]
26.	2019-06-24	6852	[REDACTED]
27.	2019-07-01	7248	[REDACTED]
28.	2019-07-08	7284	[REDACTED]
29.	2019-07-15	7142	[REDACTED]
30.	2019-07-22	7108	[REDACTED]
31.	2019-07-29	7251	[REDACTED]
32.	2019-08-05	7403	[REDACTED]
33.	2019-08-12	7457	[REDACTED]
34.	2019-08-19	7502	[REDACTED]
35.	2019-08-26	7540	[REDACTED]
36.	2019-09-02	7237	[REDACTED]
37.	2019-09-09	7328	[REDACTED]
38.	2019-09-16	5566	[REDACTED]
39.	2019-09-23	7049	[REDACTED]
40.	2019-09-30	6880	[REDACTED]
41.	2019-10-07	6518	[REDACTED]
42.	2019-10-14	6688	[REDACTED]
43.	2019-10-21	6667	[REDACTED]
44.	2019-10-28	6303	[REDACTED]



97.	2020-11-02	3580
98.	2020-11-09	3403
99.	2020-11-16	3336
100.	2020-11-23	3230
101.	2020-11-30	3183
102.	2020-12-07	3285
103.	2020-12-14	3367
104.	2020-12-21	3748
105.	2020-12-28	3986
106.	2021-01-04	3906
107.	2021-01-11	3425
108.	2021-01-18	3144
109.	2021-01-25	3115
110.	2021-02-01	3285
111.	2021-02-08	3321
112.	2021-02-15	3475
113.	2021-02-22	3549
114.	2021-03-01	3755
115.	2021-03-08	3080
116.	2021-03-15	3789
117.	2021-03-22	3804
118.	2021-03-29	4238
119.	2021-04-05	4307
120.	2021-04-12	4225
121.	2021-04-19	4391
122.	2021-04-26	4868
123.	2021-05-03	4977
124.	2021-05-10	5164
125.	2021-05-17	4986
126.	2021-05-24	5024
127.	2021-05-31	4824
128.	2021-06-07	5652
129.	2021-06-14	5613
130.	2021-06-21	6061
131.	2021-06-28	2554

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

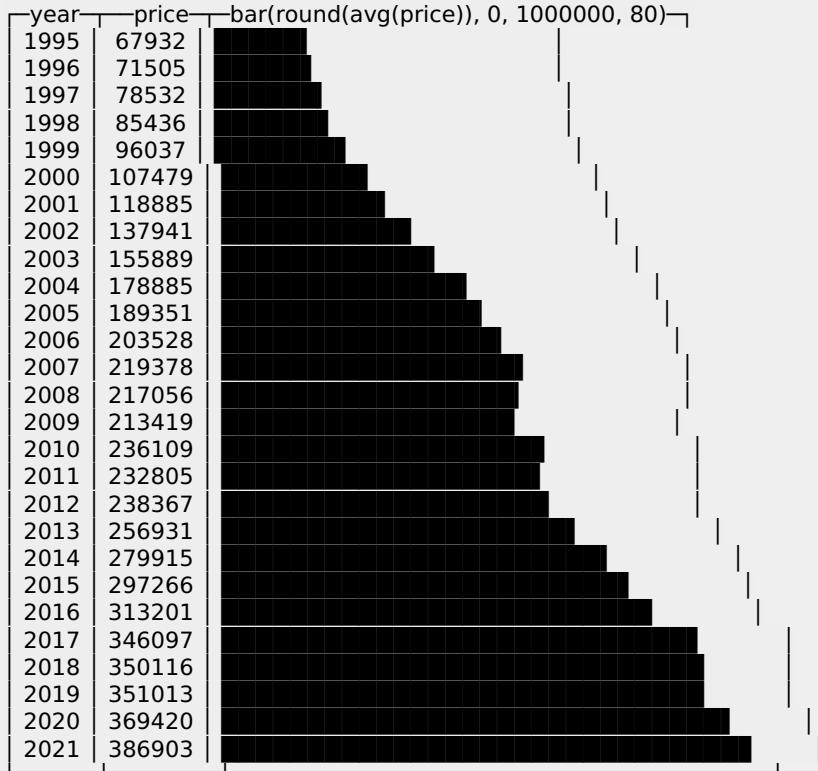
Run Some Queries

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:

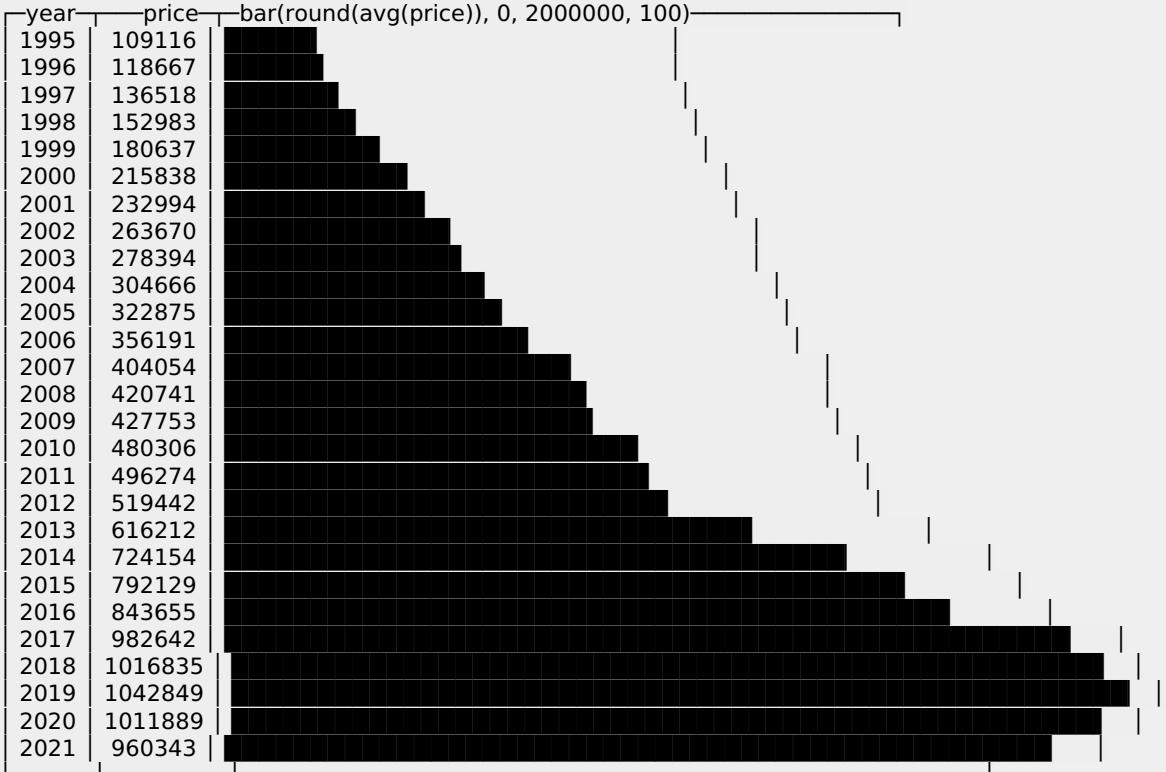


Query 2. Average Price per Year in London

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

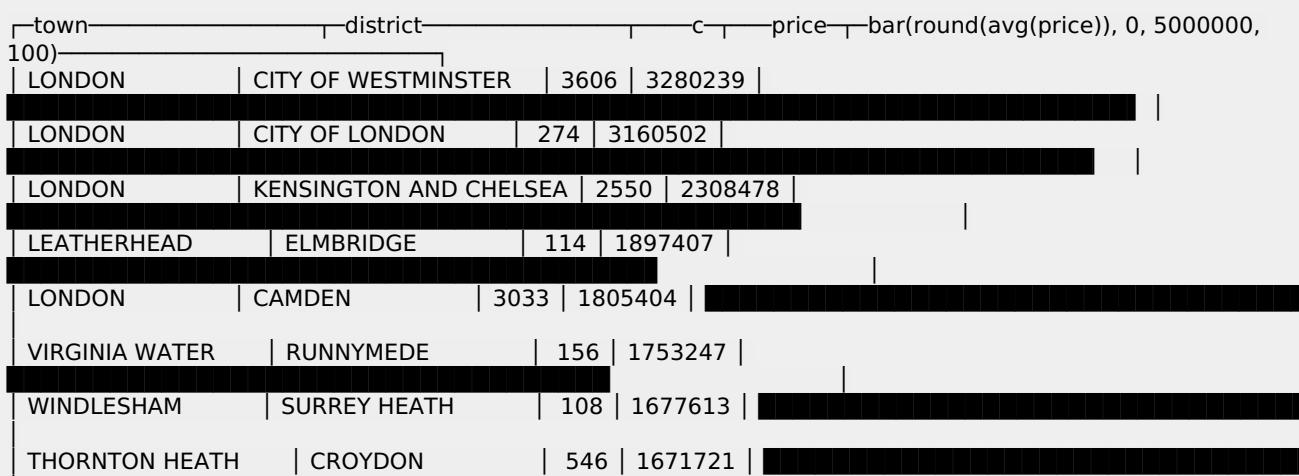
Query:

```

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;

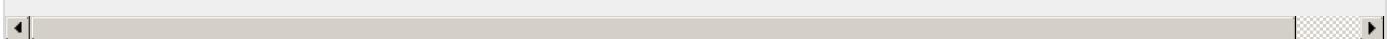
```

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:

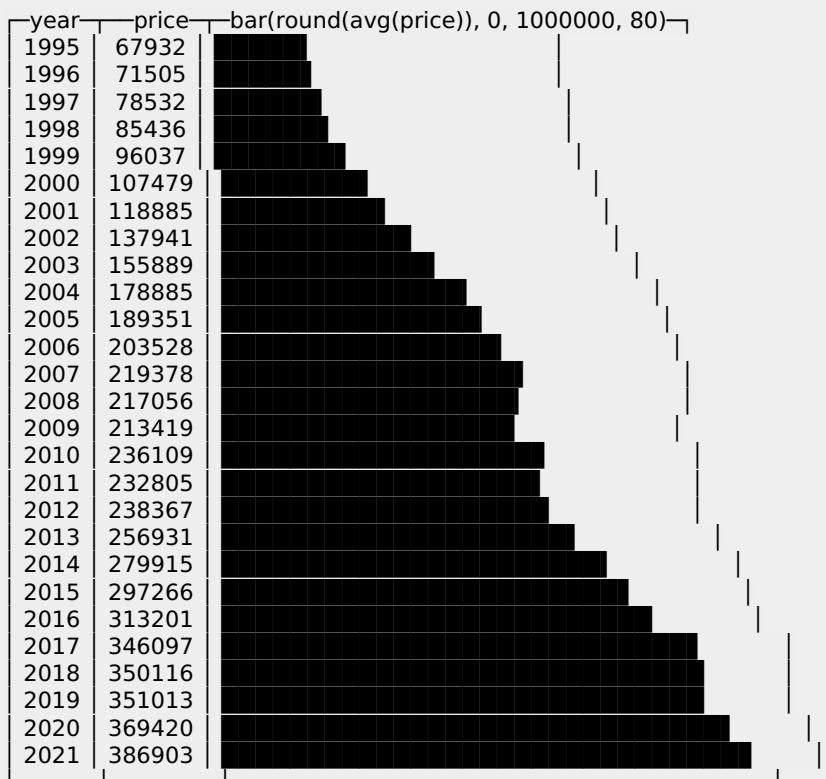
```
SET allow_experimental_projection_optimization = 1;
```

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:

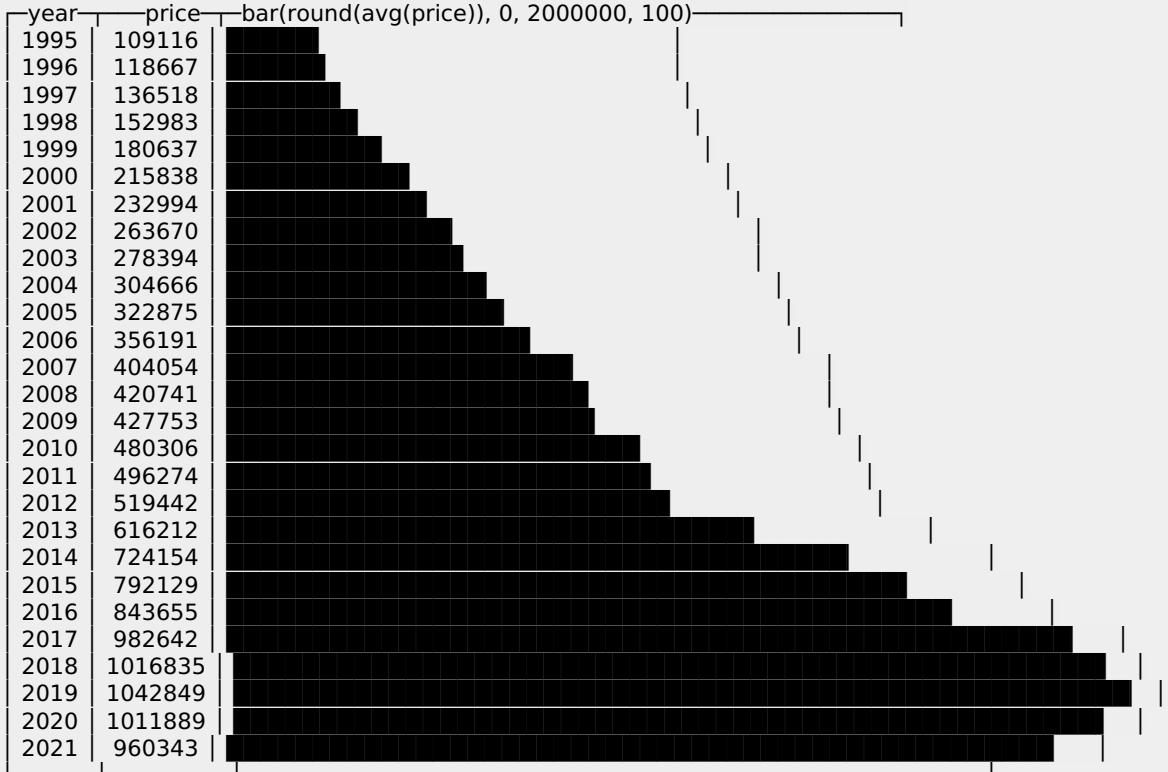


Query 2. Average Price Per Year in London

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

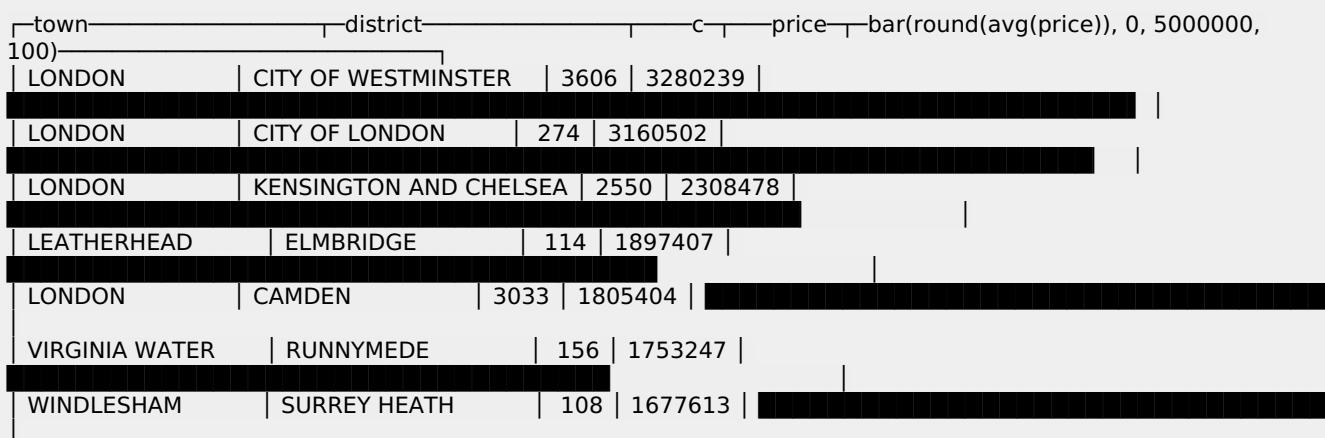
The condition (date >= '2020-01-01') needs to be modified to match projection dimension (toYear(date) >= 2020).

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 LEATHERHEAD	ELMBRIDGE GUILDFORD	485 986581 [REDACTED] 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 CRANBROOK	EALING TUNBRIDGE WELLS	2658 871755 [REDACTED] 431 862348 [REDACTED]
LONDON BELVEDERE	MERTON BEXLEY	2099 859118 [REDACTED] 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 STOCKBRIDGE	GUILDFORD TEST VALLEY	176 793494 [REDACTED] 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](#).

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" = 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), (-37.838636380279524, 55.72859509486539), (-37.8395161005249, 55.727705075632784), (-37.83897964285276, 55.722727886185154), (-37.83862557539366, 55.72034817326636), (-37.83559735744853, 55.71944437307499), (-37.835370708803126, 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, 55.57505691895805), (-37.60443099999999, 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, 55.68235377885389), (-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), (-37.37244070571134, 55.76432079697595), (-37.3724259757175, 55.76636979670426)
```

```
(3/.369922155/5/884, 55./6/3541/953104), (3/.369892695//02/5, 55./68234193165/5), (3/.3/0214/301635/5,
55.782312184391266), (37.370493611114505, 55.78436801120489), (37.37120164550783, 55.78596427165359),
(37.37284851456452, 55.7874378183096), (37.37608325135799, 55.7886695054807), (37.3764587460632,
55.78947647305964), (37.3753000265506, 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,
55.87055369615854), (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,
55.89239745507079), (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), (37.55796999999999, 55.90901557907218), (37.572711542327866,
55.91059395704873), (37.57942799999999, 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), (37.8334143888553,
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

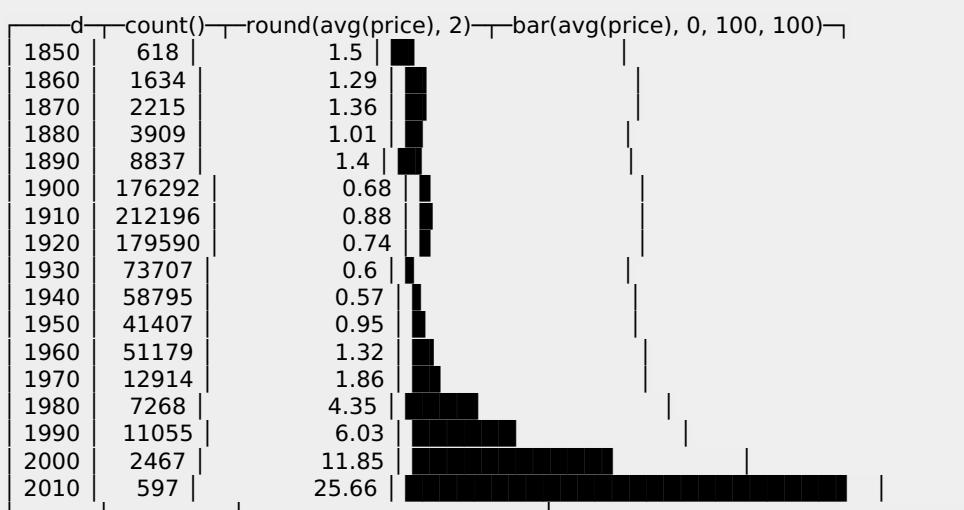
Query:

```

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

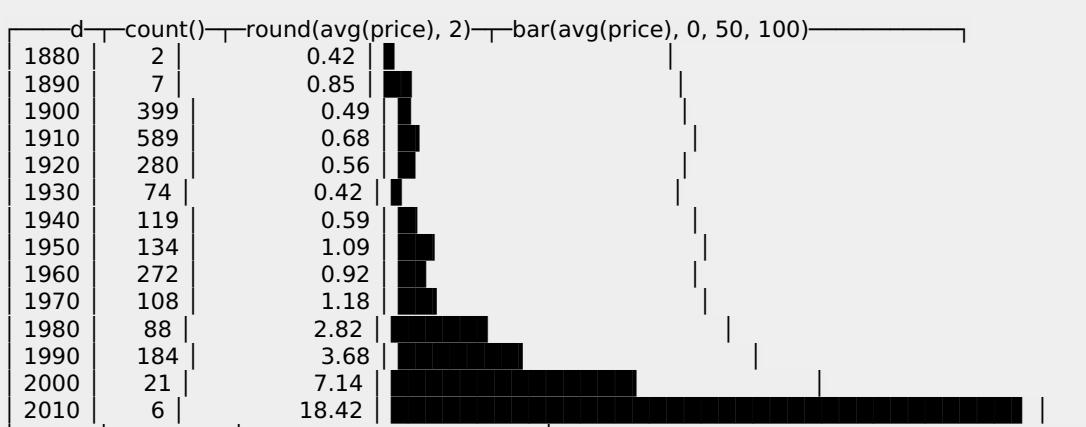
Query:

```

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:

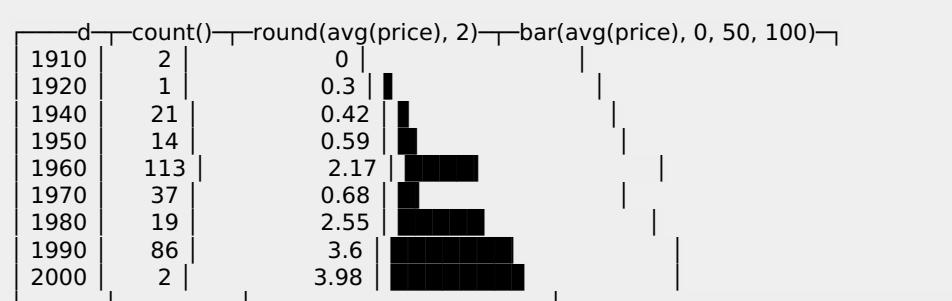


Vodka

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.

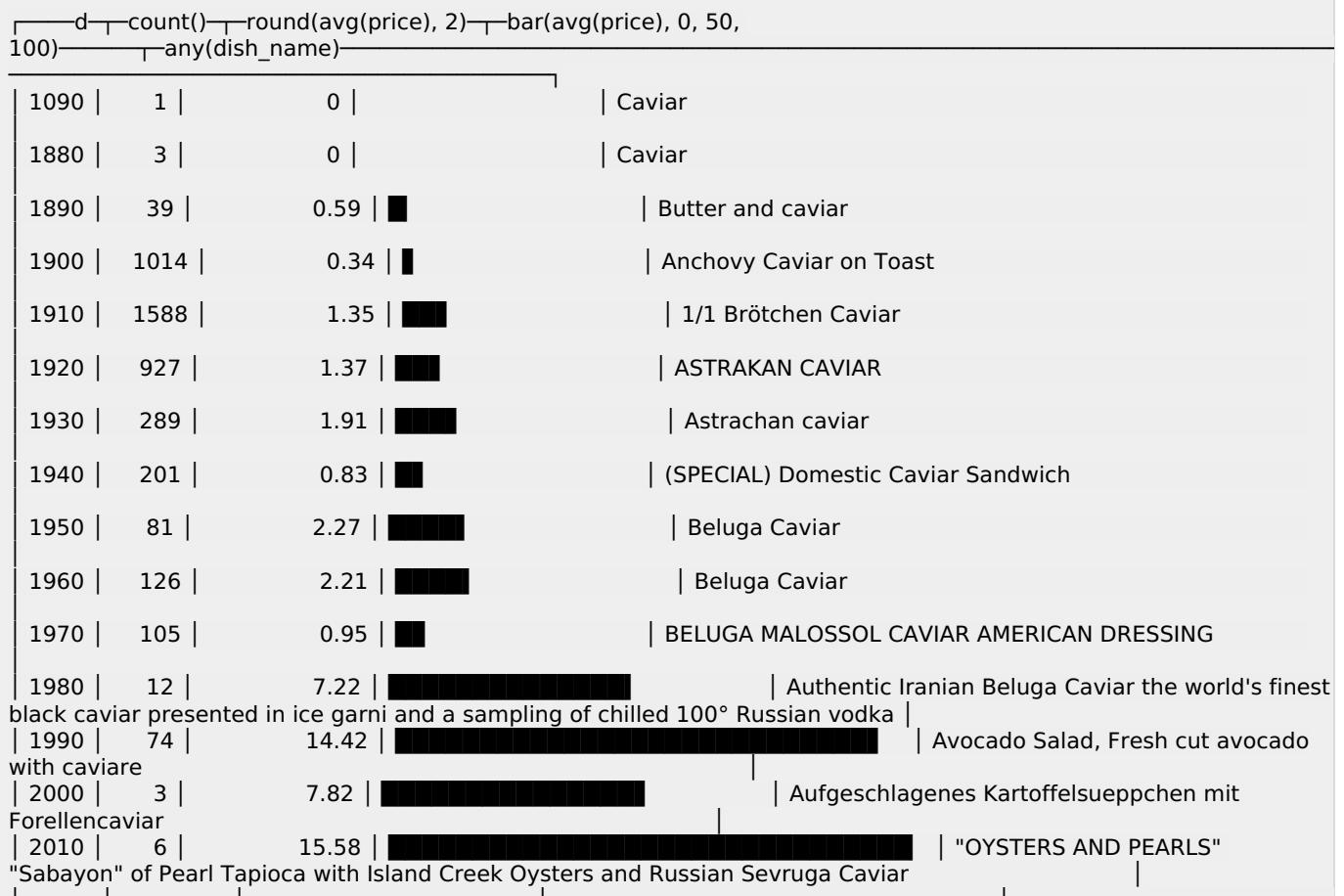
Caviar

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](#).

OnTime

航班飞行数据有以下两个方式获取：

- 从原始数据导入
- 下载预处理好的数据

从原始数据导入

下载数据：

```
for s in `seq 1987 2018`  
do  
for m in `seq 1 12`  
do  
wget  
https://transtats.bts.gov/PREZIP/On\_Time\_Reportng\_Carrier\_On\_Time\_Performance\_1987\_present\_\${s}\_\${m}.zip  
done  
done
```

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

创建表结构：

```
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
```

```

`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;

```

加载数据：

```
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 ontimedata FORMAT CSVWithNames'"
```

下载预处理好的分区数据

```
$ curl -O https://datasets.clickhouse.com/ontimedata/partitions/ontimedata.tar
$ tar xvf ontimedata.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.ontimedata"
```

信息

如果要运行下面的SQL查询，必须使用完整的表名，`datasets.ontimedata`。

查询：

Q0.

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

Q1. 查询从2000年到2008年每天的航班数

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

Q2. 查询从2000年到2008年每周延误超过10分钟的航班数。

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

Q3. 查询2000年到2008年每个机场延误超过10分钟以上的次数

```
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. 查询2007年各航空公司延误超过10分钟以上的次数

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

Q5. 查询2007年各航空公司延误超过10分钟以上的百分比

```

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;

```

更好的查询版本：

```

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

```

Q6. 同上一个查询一致,只是查询范围扩大到2000年到2008年

```

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;

```

更好的查询版本：

```

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. 每年航班延误超过10分钟的百分比

```

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;

```

更好的查询版本：

```

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

```

Q8. 每年更受人们喜爱的目的地

```

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;

```

Bonus:

```

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;

```

这个性能测试由Vadim Tkachenko提供。参考：

- <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>

ClickHouse体验平台

[ClickHouse体验平台](#) 允许人们通过即时运行查询来尝试ClickHouse，而无需设置他们的服务器或集群。

体验平台中提供几个示例数据集以及显示ClickHouse特性的示例查询。还有一些ClickHouse LTS版本可供尝试。

ClickHouse体验平台提供了小型集群[Managed Service for ClickHouse](#)实例配置(4 vCPU, 32 GB RAM)它们托管在[Yandex.Cloud](#). 更多信息查询[cloud providers](#).

您可以使用任何HTTP客户端对ClickHouse体验平台进行查询，例如[curl](#)或者[wget](#),或使用[JDBC](#)或者[ODBC](#)驱动连接。关于支持ClickHouse的软件产品的更多信息详见[here](#).

Credentials

参数	值
HTTPS端点	https://play-api.clickhouse.com:8443
TCP端点	play-api.clickhouse.com:9440
用户	playground
密码	clickhouse

还有一些带有特定ClickHouse版本的附加信息来试验它们之间的差异(端口和用户/密码与上面相同):

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

注意

所有这些端点都需要安全的TLS连接。

查询限制

查询以只读用户身份执行。这意味着一些局限性:

- 不允许DDL查询
- 不允许插入查询

还强制执行以下设置:

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

ClickHouse体验还有如下：

[ClickHouse管理服务](#)

实例托管 [Yandex云](#)。

更多信息 [云提供商](#)。

示例

使用curl连接Https服务：

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

TCP连接示例[CLI](#):

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

Implementation Details

ClickHouse体验平台界面实际上是通过ClickHouse [HTTP API](#)接口实现的。

ClickHouse体验平台是一个ClickHouse集群，没有任何附加的服务器端应用程序。如上所述，ClickHouse的HTTPS和TCP/TLS端点也可以作为体验平台的一部分公开使用，代理通过[Cloudflare Spectrum](#)增加一层额外的保护和改善连接。

注意

强烈不推荐在任何其他情况下将ClickHouse服务器暴露给公共互联网。确保它只在私有网络上侦听，并由正确配置的防火墙监控。

客户端

ClickHouse提供了两个网络接口(两个都可以选择包装在TLS中以增加安全性):

- [HTTP](#), 包含文档，易于使用。
- [Native TCP](#)，简单，方便使用。

在大多数情况下，建议使用适当的工具或库，而不是直接与它们交互。Yandex官方支持的项目有:

- [命令行客户端](#)
- [JDBC驱动](#)
- [ODBC驱动](#)
- [C++客户端](#)

还有一些广泛的第三方库可供ClickHouse使用:

- [客户端库](#)
- [第三方集成库](#)
- [可视化UI](#)

命令行客户端

ClickHouse提供了一个原生命令行客户端clickhouse-client客户端支持命令行支持的更多信息详见[Configuring](#)。

安装部署后，系统默认会安装clickhouse-client(同时它属于clickhouse-client安装包中)。

```
$ 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.

:)
```

不同的客户端和服务器版本彼此兼容，但是一些特性可能在旧客户机中不可用。我们建议使用与服务器应用相同版本的客户端。当你尝试使用旧版本的客户端时，服务器上的clickhouse-client会显示如下信息:

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

使用方式

客户端可以在交互和非交互(批处理)模式下使用。要使用批处理模式，请指定`query`参数，或将数据发送到`stdin`(它会验证`stdin`是否是终端)，或两者同时进行。与HTTP接口类似，当使用`query`参数并向`stdin`发送数据时，客户端请求就是一行的`stdin`输入作为`query`的参数。这种方式在大规模的插入请求中非常方便。

使用客户端插入数据的示例：

```
$ 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";
```

在批量模式中，默认的数据格式是TabSeparated分隔的。您可以根据查询来灵活设置FORMAT格式。

默认情况下，在批量模式中只能执行单个查询。为了从一个Script中执行多个查询，可以使用`--multิquery`参数。除了`INSERT`请求外，这种方式在任何地方都有用。查询的结果会连续且不含分隔符地输出。

同样的，为了执行大规模的查询，您可以为每个查询执行一次`clickhouse-client`。但注意到每次启动`clickhouse-client`程序都需要消耗几十毫秒时间。

在交互模式下，每条查询过后，你可以直接输入下一条查询命令。

如果`multiline`没有指定（默认没指定）：为了执行查询，按下Enter即可。查询语句不是必须使用分号结尾。如果需要写一个或多个查询语句，可以在换行之前输入一个反斜杠\，然后在您按下Enter键后，您就可以输入当前语句的下一行查询了。

如果指定了`multiline`：为了执行查询，需要以分号结尾并且按下Enter键。如果行末没有分号，将认为当前语句并没有输入完而要求继续输入下一行。

若只运行单个查询，分号后面的所有内容都会被忽略。

您可以指定\G来替代分号或者在分号后面，这表示使用Vertical的格式。在这种格式下，每一个值都会打印在不同的行中，这种方式对于宽表来说很方便。这个不常见的特性是为了兼容MySQL命令而加的。

命令行客户端是基于replxx(类似于readline)。换句话说，它可以使用我们熟悉的快捷键方式来操作以及保留历史命令。历史命令会写入在`~/.clickhouse-client-history`中。

默认情况下，输出的格式是PrettyCompact。您可以通过FORMAT设置根据不同查询来修改格式，或者通过在查询末尾指定\G字符，或通过在命令行中使用`--format`或`--vertical`参数，或使用客户端的配置文件。

若要退出客户端，使用Ctrl+D（或Ctrl+C），或者输入以下其中一个命令：`exit`, `quit`, `logout`, `учше`, `йгше`, `дшпшге`, `exit;`, `quit;`, `logout;`, `q`, `Q`, `:q`

当执行一个查询的时候，客户端会显示：

1. 进度，进度会每秒更新十次（默认情况下）。对于很快的查询，进度可能没有时间显示。
2. 为了调试会显示解析且格式化后的查询语句。
3. 指定格式的输出结果。
4. 输出结果的行数的行数，经过的时间，以及查询处理的速度。

您可以通过Ctrl+C来取消一个长时间的查询。然而，您依然需要等待服务端来中止请求。在某个阶段去取消查询是不可能的。如果您不等待并再次按下Ctrl + C，客户端将会退出。

命令行客户端允许通过外部数据（外部临时表）来查询。更多相关信息，请参考《[外部数据查询处理](#)》。

查询参数

您可以创建带有参数的查询，并将值从客户端传递给服务器。这允许避免在客户端使用特定的动态值格式化查询。例如：

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

查询语法

像平常一样格式化一个查询，然后把你想要从app参数传递到查询的值用大括号格式化，格式如下：

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

- `name` — 占位符标识符。在控制台客户端，使用`--param_<name> = value`来指定
- `data type` — **数据类型**参数值。例如，一个数据结构(`integer`, ('`string`', `integer`))拥有`Tuple(UInt8, Tuple(String, UInt8))`数据类型(你也可以用另一个`integer`类型)。

示例

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

配置

您可以通过以下方式传入参数到clickhouse-client中（所有的参数都有默认值）：

- 通过命令行

命令行参数会覆盖默认值和配置文件的配置。

- 配置文件

配置文件的配置会覆盖默认值

命令行参数

- `--host, -h` — 服务端的host名称，默认是`localhost`。您可以选择使用host名称或者IPv4或IPv6地址。
- `--port` — 连接的端口，默认值：9000。注意HTTP接口以及TCP原生接口使用的是不同端口。
- `--user, -u` — 用户名。默认值：`default`。
- `--password` — 密码。默认值：空字符串。
- `--query, -q` — 使用非交互模式查询。
- `--database, -d` — 默认当前操作的数据库。默认值：服务端默认的配置（默认是`default`）。
- `--multiline, -m` — 如果指定，允许多行语句查询（Enter仅代表换行，不代表查询语句完结）。
- `--multiquery, -n` — 如果指定，允许处理用;`;`分隔的多个查询，只在非交互模式下生效。
- `--format, -f` — 使用指定的默认格式输出结果。
- `--vertical, -E` — 如果指定，默认情况下使用垂直格式输出结果。这与`-format=Vertical`相同。在这种格式中，每个值都在单独的行上打印，这种方式对显示宽表很有帮助。
- `--time, -t` — 如果指定，非交互模式下会打印查询执行的时间到`stderr`中。
- `--stacktrace` — 如果指定，如果出现异常，会打印堆栈跟踪信息。
- `--config-file` — 配置文件的名称。
- `--secure` — 如果指定，将通过安全连接连接到服务器。

- `--history_file` — 存放命令历史的文件的路径。
- `--param_<name>` — 查询参数配置[查询参数](#)。

配置文件

`clickhouse-client`使用以下第一个配置文件：

- 通过`--config-file`参数指定。
- `./clickhouse-client.xml`
- `~/.clickhouse-client/config.xml`
- `/etc/clickhouse-client/config.xml`

配置文件示例：

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

原生接口 (TCP) {#native-interface-tcp}

原生接口协议用于[命令行客户端](#)，用于分布式查询处理期间的服务器间通信，以及其他C++程序。不幸的是，原生ClickHouse协议还没有正式的规范，但它可以从ClickHouse源代码[从这里开始](#)或通过拦截和分析TCP流量进行逆向工程。

HTTP客户端

HTTP接口允许您在任何编程语言的任何平台上使用ClickHouse。我们使用它在Java和Perl以及shell脚本中工作。在其他部门中，HTTP接口用于Perl、Python和Go。HTTP接口比原生接口受到更多的限制，但它具有更好的兼容性。

默认情况下，`clickhouse-server`会在8123端口上监控HTTP请求（这可以在配置中修改）。

如果你发送了一个未携带任何参数的GET /请求，它会返回一个字符串 «Ok.» (结尾有换行)。可以将它用在健康检查脚本中。

如果你发送了一个未携带任何参数的GET /请求，它返回响应码200和OK字符串定义，可在[Http服务响应配置](#)定义(在末尾添加换行)

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

通过URL中的`query`参数来发送请求，或者发送POST请求，或者将查询的开头部分放在URL的`query`参数中，其他部分放在POST中（我们会在后面解释为什么这样做是有必要的）。URL的大小会限制在16KB，所以发送大型查询时要时刻记住这点。

如果请求成功，将会收到200的响应状态码和响应主体中的结果。

如果发生了某个异常，将会收到500的响应状态码和响应主体中的异常描述信息。

当使用GET方法请求时，`readonly`会被设置。换句话说，若要作修改数据的查询，只能发送POST方法的请求。可以将查询通过POST主体发送，也可以通过URL参数发送。

示例：

```
$ 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
```

可以看到，curl 命令由于空格需要 URL 转义，所以不是很方便。尽管 wget 命令对url做了 URL 转义，但我们并不推荐使用他，因为在 HTTP 1.1 协议下使用 keep-alive 和 Transfer-Encoding: chunked 头部设置它并不能很好的工作。

```
$ 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
```

如您所见，curl有些不方便，因为空格必须进行URL转义。

尽管wget本身会对所有内容进行转义，但我们不推荐使用它，因为在使用keepalive和传输编码chunked时，它在HTTP 1.1上不能很好地工作。

```
$ 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
```

如果部分查询是在参数中发送的，部分是在POST中发送的，则在这两个数据部分之间插入换行。

错误示例：

```
$ 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
```

默认情况下，返回的数据是TabSeparated格式的，更多信息，见[Formats](#)部分。

您可以使用查询的FORMAT子句来设置其他格式。

另外，还可以使用default_formatURL参数或X-ClickHouse-Format头来指定TabSeparated之外的默认格式。

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

```
1  
1
```

INSERT必须通过POST方法来插入数据。在这种情况下，您可以在URL参数中编写查询的开始部分，并使用POST传递要插入的数据。例如，要插入的数据可以是来自MySQL的一个以tab分隔的存储。通过这种方式，INSERT查询替换了从MySQL查询的LOAD DATA LOCAL INFILE。

示例：创建一个表：

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

使用类似INSERT的查询来插入数据：

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

数据可以从查询中单独发送：

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

您可以指定任何数据格式。Values格式与将INSERT写入t值时使用的格式相同：

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

若要插入tab分割的数据，需要指定对应的格式：

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

从表中读取内容。由于查询处理是并行的，数据以随机顺序输出。

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

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

删除表：

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

成功请求后并不会返回数据，返回一个空的响应体。

在传输数据时，可以使用ClickHouse内部压缩格式。压缩的数据具有非标准格式，您需要使用特殊的clickhouse-compressor程序来处理它(它是与clickhouse-client包一起安装的)。为了提高数据插入的效率，您可以通过使用[http_native_compression_disable_checksumming_on_decompress](#)设置禁用服务器端校验。

如果在URL中指定了compress=1，服务会返回压缩的数据。

如果在URL中指定了decompress=1，服务会解压通过POST方法发送的数据。

您也可以选择使用[HTTP compression](#)。发送一个压缩的POST请求，附加请求头Content-Encoding: compression_method。为了使ClickHouse响应，您必须附加Accept-Encoding: compression_method。ClickHouse支持gzip，br和deflate [compression methods](#)。要启用HTTP压缩，必须使用ClickHouse[启用Http压缩](#)配置。您可以在[Http zlib压缩级别](#)设置中为所有压缩方法配置数据压缩级别。

您可以使用它在传输大量数据时减少网络流量，或者创建立即压缩的转储。

通过压缩发送数据的例子：

```
## 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/'
```

警告

一些HTTP客户端可能会在默认情况下从服务器解压数据(使用gzip和deflate)，即使您未正确地使用了压缩设置，您也可能会得到解压数据。

您可以使用databaseURL参数或X-ClickHouse-Database头来指定默认数据库。

```
$ 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
```

默认情况下，在服务器设置中注册的数据库被用作默认数据库。默认情况下，它是名为default的数据库。或者，您可以始终在表名之前使用点来指定数据库。

用户名和密码可以通过以下三种方式指定：

1. 通过HTTP Basic Authentication。示例：

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

1. 通过URL参数中的user和password。示例：

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

1. 使用X-ClickHouse-User或X-ClickHouse-Key头指定，示例：

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

如果未指定用户名，则使用`default`。如果未指定密码，则使用空密码。

您还可以使用URL参数来指定处理单个查询或整个设置配置文件的任何设置。例子:`http://localhost:8123/?profile=web&max_rows_to_read=10000000000&query=SELECT+1`

更多信息，详见[设置](#)部分。

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

有关其他参数的信息，请参考[SET](#)一节。

类似地，您可以在HTTP协议中使用ClickHouse会话。为此，需要向请求添加`session_id`GET参数。您可以使用任何字符串作为会话ID。默认情况下，会话在60秒不活动后终止。要更改此超时配置，请修改服务器配置中的`default_session_timeout`设置，或向请求添加`session_timeout`GET参数。要检查会话状态，使用`session_check=1`参数。一次只能在单个会话中执行一个查询。

您可以在`X-ClickHouse-Progress`响应头中收到查询进度的信息。为此，启用[Http Header携带进度](#)。示例：

```
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"}
```

显示字段信息：

- `read_rows` — 读取的行数。
- `read_bytes` — 读取的数据字节数。
- `total_rows_to_read` — 读取的数据总行数。
- `written_rows` — 写入数据行数。
- `written_bytes` — 写入数据字节数。

如果HTTP连接丢失，运行的请求不会自动停止。解析和数据格式化是在服务器端执行的，使用Http连接可能无效。

可选的`query_id`参数可能当做query ID传入（或者任何字符串）。更多信息，详见[replace_running_query](#)部分。

可选的`quota_key`参数可能当做quota key传入（或者任何字符串）。更多信息，详见[Quotas](#)部分。

HTTP接口允许传入额外的数据（外部临时表）来查询。更多信息，详见[外部数据查询处理](#)部分。

响应缓冲

可以在服务器端启用响应缓冲。提供了`buffer_size`和`wait_end_of_query`两个URL参数来达此目的。

`buffer_size`决定了查询结果要在服务内存中缓冲多少个字节数据。如果响应体比这个阈值大，缓冲区会写入到HTTP管道，剩下的数据也直接发到HTTP管道中。

为了确保整个响应体被缓冲，可以设置`wait_end_of_query=1`。这种情况下，存入内存的数据会被缓冲到服务端的一个临时文件中。

示例：

```
$ 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'
```

查询请求响应状态码和HTTP头被发送到客户端后，若发生查询处理出错，使用缓冲区可以避免这种情况的发生。在这种情况下，响应主体的结尾会写入一条错误消息，而在客户端，只能在解析阶段检测到该错误。

查询参数

您可以使用参数创建查询，并通过相应的HTTP请求参数为它们传递值。有关更多信息，请参见[CLI查询参数](#)。

示例

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

特定的HTTP接口

ClickHouse通过HTTP接口支持特定的查询。例如，您可以如下所示向表写入数据：

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

ClickHouse还支持预定义的HTTP接口，可以帮助您更容易与第三方工具集成，如[Prometheus Exporter](#)。

示例：

- 首先，将此部分添加到服务器配置文件中：

```
<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>
```

- 请求Prometheus格式的URL以获取数据：

```

$ 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

```

正如您从示例中看到的，如果在 config.xml 文件中配置了 `http_handlers`，并且 `http_handlers` 可以包含许多规则。ClickHouse 将把接收到的 HTTP 请求与 rule 中的预定义类型进行匹配，第一个匹配的将运行处理程序。如果匹配成功，ClickHouse 将执行相应的预定义查询。

现在 rule 可以配置 `method`, `header`, `url`, `handler`:

- `method` 负责匹配 HTTP 请求的方法部分。`method` 完全符合 HTTP 协议中 `method` 的定义。这是一个可选的配置。如果它没有在配置文件中定义，那么它与 HTTP 请求的方法部分不匹配。

- `url` 负责匹配 HTTP 请求的 URL 部分。它匹配 RE2 正则表达式。这是一个可选的配置。如果配置文件中没有定义它，则它与 HTTP 请求的 URL 部分不匹配。
- `headers` 负责匹配 HTTP 请求的头部分。它与 RE2 的正则表达式兼容。这是一个可选的配置。如果它没有在配置文件中定义，那么它与 HTTP 请求的头部分不匹配。

- `handler` 包含主要的处理部分。现在 `handler` 可以配置 `type`, `status`, `content_type`, `response_content`, `query`, `query_param_name`。

`type` 目前支持三种类型: **特定查询**, **动态查询**, **static**.

- `query` — 使用 `predefined_query_handler` 类型，在调用处理程序时执行查询。
- `query_param_name` — 与 `dynamic_query_handler` 类型一起使用，提取并执行 HTTP 请求参数中与 `query_param_name` 值对应的值。
- `status` — 与 `static` 类型一起使用，响应状态代码。
- `content_type` — 与 `static` 类型一起使用，响应信息 **Content-Type**。
- `response_content` — 与 `static` 类型一起使用，响应发送给客户端的内容，当使用前缀 `file://` 或 `config://` 时，从发送给客户端的文件或配置中查找内容。

接下来是不同 `type` 的配置方法。

特定查询

`predefined_query_handler` 支持设置 `Settings` 和 `query_params` 参数。您可以将 `query` 配置为 `predefined_query_handler` 类型。

`query` 是一个预定义的 `predefined_query_handler` 查询，它由 ClickHouse 在匹配 HTTP 请求并返回查询结果时执行。这是一个必须的配置。

以下是定义的 `max_threads` 和 `max_alter_threads` 设置，然后查询系统表以检查这些设置是否设置成功。

示例：

```
<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
```

警告

在一个 `predefined_query_handler` 中，只支持 `insert` 类型的一个查询。

动态查询

`dynamic_query_handler` 时，查询以 HTTP 请求参数的形式编写。区别在于，在 `predefined_query_handler` 中，查询是在配置文件中编写的。您可以在 `dynamic_query_handler` 中配置 `query_param_name`。

ClickHouse提取并执行与HTTP请求URL中的`query_param_name`值对应的值。`query_param_name`的默认值是`/query`。这是一个可选的配置。如果配置文件中没有定义，则不会传入参数。

为了试验这个功能，示例定义了`max_threads`和`max_alter_threads`，`queries`设置是否成功的值。

示例：

```
<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
```

static

`static`可以返回`content_type`, `status`和`response_content`。`response_content`可以返回指定的内容。

示例：

返回信息。

```
<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!%
```

从配置中查找发送到客户端的内容。

```
<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>%
```

从发送到客户端的文件中查找内容。

```

<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 接口

ClickHouse 支持 MySQL wire 通讯协议。可以通过在配置文件中设置 `mysql_port` 来启用它：

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

使用命令行工具 `mysql` 进行连接的示例：

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

如果连接成功，则输出：

```
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>
```

为了与所有MySQL客户端兼容，建议在配置文件中使用 **double SHA1** 来指定用户密码。

如果使用 **SHA256** 指定用户密码，一些客户端将无法进行身份验证（比如`mysqljs`和旧版本的命令行工具`mysql`）。

限制：

- 不支持prepared queries
- 某些数据类型以字符串形式发送

输入/输出格式

ClickHouse可以接受和返回各种格式的数据。受支持的输入格式可用于提交给`INSERT`语句、从文件表(File,URL,HDFS或者外部目录)执行`SELECT`语句，受支持的输出格式可用于格式化`SELECT`语句的返回结果，或者通过`INSERT`写入到文件表。

以下是支持的格式：

格式	输入	输出
TabSeparated	✓	✓
TabSeparatedRaw	✓	✓
TabSeparatedWithNames	✓	✓
TabSeparatedWithNamesAndTypes	✓	✓
Template	✓	✓
TemplateIgnoreSpaces	✓	✗
CSV	✓	✓
CSVWithNames	✓	✓

格式	输入	输出
CustomSeparated	✓	✓
Values	✓	✓
Vertical	✗	✓
JSON	✗	✓
JSONAsString	✓	✗
JSONStrings	✗	✓
JSONCompact	✗	✓
JSONCompactStrings	✗	✓
JSONEachRow	✓	✓
JSONEachRowWithProgress	✗	✓
JSONStringsEachRow	✓	✓
JSONStringsEachRowWithProgress	✗	✓
JSONCompactEachRow	✓	✓
JSONCompactEachRowWithNamesAndTypes	✓	✓
JSONCompactStringsEachRow	✓	✓
JSONCompactStringsEachRowWithNamesAndTypes	✓	✓
TSKV	✓	✓
Pretty	✗	✓
PrettyCompact	✗	✓
PrettyCompactMonoBlock	✗	✓
PrettyNoEscapes	✗	✓
PrettySpace	✗	✓
Protobuf	✓	✓
ProtobufSingle	✓	✓
Avro	✓	✓

格式	输入	输出
AvroConfluent	✓	✗
Parquet	✓	✓
Arrow	✓	✓
ArrowStream	✓	✓
ORC	✓	✓
RowBinary	✓	✓
RowBinaryWithNamesAndTypes	✓	✓
Native	✓	✓
Null	✗	✓
XML	✗	✓
CapnProto	✓	✗
LineAsString	✓	✗
Regexp	✓	✗
RawBLOB	✓	✓

您可以使用ClickHouse设置一些格式化参数。更多详情设置请参考[设置](#)

TabSeparated

在**TabSeparated**分隔格式中，数据按行写入。每行包含由制表符分隔的值，每个值后跟一个制表符，除了行中最后一个值，最后的值后面是一个换行符。在任何地方都采用严格的Unix换行(\n)。最后一行结束后必须再插入一个换行符。值以文本格式编写，不包含引号，并使用转义的特殊字符。

这种格式也被称为**TSV**。

TabSeparated格式便于其他的程序和脚本处理数据。默认情况下，HTTP接口和命令行客户端的批处理模式中会使用这个格式。这种格式还允许在不同dbms之间传输数据。例如，您可以从MySQL获取转储并将其上传到ClickHouse，反之亦然。

TabSeparated格式支持输出总计的结果(当SQL语句包含**WITH TOTALS**)和极值(当extremes被设置为1时)。在这种情况下，总计值和极值会在主数据后输出。主要结果、总值和极值之间用空行分隔。示例:

```
set extremes=1;
SELECT EventDate, count() AS c FROM test.hits_v1 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

0000-00-00 8873898

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

数据格式化

整数是用十进制形式写的。数字可以在开头包含一个额外的+字符(解析时忽略该符号，格式化时不记录该符号)。非负数不能包含负号。在读取时，允许将空字符串解析为零，或者(对于有符号类型)将'-'(仅有减号的字符串)解析为零。不符合相应数据类型的数字可能被解析为数值，而不会出现错误消息。

浮点数以十进制形式书写。用.作为小数点的符号。支持指数符号，如inf、+inf、-inf和nan。小数点前或后可以不出现数字(如123.或.123)。

在格式化期间，浮点数精度可能会丢失。

在解析期间，并不严格要求读取与机器可以表示的最接近的数值。

日期以YYYY-MM-DD格式编写，并以相同的格式解析，但允许使用任何字符作为分隔符。

日期和时间以YYYY-MM-DD hh:mm:ss的格式书写，并以相同的格式解析，但允许使用任何字符作为分隔符。

时区采用客户端或服务器端时区(取决于谁对数据进行格式化)。对于带有时间的日期，没有是哦用夏时制时间。因此，如果导出的数据采用了夏时制，则实际入库的时间不一定与预期的时间对应，解析将根据解析动作发起方选择时间。

在读取操作期间，不正确的日期和具有时间的日期可以自然溢出(如2021-01-32)或设置成空日期和时间，而不会出现错误消息。

有个例外情况，时间解析也支持Unix时间戳(如果它恰好由10个十进制数字组成)。其结果与时区无关。格式YYYY-MM-DD hh:mm:ss和NNNNNNNNNN这两种格式会自动转换。

字符串输出时，特殊字符会自动转义。以下转义序列用于输出:\b, \f, \r, \n, \t, \0, \', \\。解析还支持\a、\v和\xHH(HH代表十六进制编码)和\c，其中c是任何字符(这些序列被转换为c)。因此，读取数据时，换行符可以写成\n或\\。例如，如果要表示字符串Hello world中'Hello'与'world'中间的空格实际上是个换行符，可以写成下面的形式:

```
Hello\nworld
```

等同于

```
Hello\
world
```

第二种形式也受支持，因为MySQL导出tab-separated格式的数据时使用这种格式。

使用TabSeparated格式传递数据时至少需要转义以下特殊字符:制表符(\t)、换行符(\n)和反斜杠(\)。

只有一小部分符号被转义。您可以很容易地找到一个能够破坏命令行终端输出的特殊字符。

数组用方括号包裹、逗号分隔的形式表示(例如[11,22,33])。数组中的数字项按上述规则进行格式化。日期和日期时间类型用单引号包裹。字符串用单引号包裹，遵循上述转义规则。

NULL将输出为\N。

Nested结构的每个元素都表示为数组。

示例：

```
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

与 **TabSeparated** 格式的不同之处在于，写入的数据不会进行转义处理。

使用这种格式解析时，每个字段中不允许出现制表符或换行符。

这种格式也被称为 **TSVRaw**。

TabSeparatedWithNames

不同于 **TabSeparated**，列名会写在第一行。

在解析过程中，第一行被完全忽略。您不能依赖列名来确定它们的位置或检查它们的正确性。

(将来可能会添加对头行解析的支持。)

这种格式也被称为 **TSVWithNames**。

TabSeparatedWithNamesAndTypes

与 **TabSeparated** 格式不同的是列名写在第一行，而列类型写在第二行。

在解析过程中，将完全忽略第一行和第二行。

这种格式也被称为 **TSVWithNamesAndTypes**。

Template

此格式允许指定带有占位符的自定义格式字符串，这些占位符用于指定转义规则。

它使用 `format_schema`, `format_schema_rows`, `format_schema_rows_between_delimiter` 以及其他格式的一些设置(例如转义 JSON 时使用 `output_format_json_quote_64bit_integers`，具体请向下阅读)

设置 `format_template_row` 用于指定行格式文件的路径，该格式文件包含行格式字符串，语法如下:

```
delimiter_i${column_i:serializeAs_i}delimiter_i${column_i:serializeAs_i} ... delimiter_i
```

其中，`delimiter_i` 是各值之间的分隔符(\$符号可以转义为 \$\$)，

`column_i` 是选择或插入值的列的名称或索引(如果为空，则跳过该列)，

`serializeAs_i` 是列值的转义规则。支持以下转义规则:

- `CSV`, `JSON`, `XML` (类似于相同名称的格式)
- `Escaped` (类似于 `TSV`)
- `Quoted` (类似于 `Values`)

- **Raw** (不转义，类似于TSVRaw)

- **None** (不转义，具体请向下阅读)

如果省略了转义规则，那么将使用**None**。**XML**和**Raw**只适用于输出。

对于下面的格式字符串：

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

SearchPhrase、**c**和**price**列的值遵循**Quoted**、**Escaped**和**JSON**转义规则，将分别在**Search phrase:**、**count:**、**ad price:**和**;**分隔符之间打印(用于**SELECT**)或输入期望的值(用于**INSERT**)。例如：

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

format_template_rows_between_delimiter设置指定行之间的分隔符，它将打印(或输入期望值)在每一行之后，最后一行除外(该设置默认值为\n)。

设置**format_template_resultset**指定结果集格式文件路径，该文件包含结果集的格式字符串。结果集的格式字符串与上述的行格式字符串具有相同的语法，并允许指定前缀、后缀，还提供打印一些附加信息的方法。该文件使用如下占位符，用于取代行格式字符串的列名的位置(即**column_i**)：

- **data** 代表遵循**format_template_row**格式的数据行，由**format_template_rows_between_delimiter**设置制定的字符分隔。此占位符必须是格式字符串中的第一个占位符。
- **totals** 代表遵循**format_template_row**格式的数据行，该行用于代表结果的总计值(当SQL语句包含了**WITH TOTALS**)
- **min** 代表遵循**format_template_row**格式的数据行，该行用于代表结果的最小值(当**extremes**设置为1时)
- **max** 代表遵循**format_template_row**格式的数据行，该行用于代表结果的最大值(当**extremes**设置为1时)
- **rows** 代表输出行的总数
- **rows_before_limit** 代表没有**LIMIT**限制的结果最小行数。仅当查询包含**LIMIT**时才输出此值。如果查询包含**GROUP BY**，那么**rows_before_limit_at_least**就是没有**LIMIT**的确切行数。
- **time** 代表请求执行时间(秒)
- **rows_read** 代表已读取的行数
- **bytes_read** 代表已读取(未压缩)的字节数

占位符**data**、**totals**、**min**和**max**不允许指定转义规则(允许显式指定**None**)。其余占位符可以指定任何转义规则。

如果**format_template_resultset**设置为空，则使用 **\${data}** 作为默认值。

对于**insert**查询，格式允许跳过某些列或某些字段的前缀或后缀(参见示例)。

Select示例：

```
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>
```

结果：

```

<!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>

```

Insert示例：

```

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和Sign 占位符是表中列的名称。将忽略行中Useless field后面和后缀中\nTotal rows:之后的值。

输入数据中的所有分隔符必须严格等于指定格式字符串中的分隔符。

TemplateIgnoreSpaces

这种格式只适用于输入。

类似于Template，但跳过输入流中分隔符和值之间的空白字符。但是，如果格式字符串包含空格字符，这些字符将会出现在输入流中。也允许指定空占位符(\${}或\${:None})来将一些分隔符分割为单独的部分，以忽略它们之间的空格。这种占位符仅用于跳过空白字符。

如果列的值在所有行的顺序相同，那么可以使用这种格式读取JSON。可以使用以下请求从格式为JSON的输出示例中插入数据：

```
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

类似于TabSeparated，但是输出的值是name=value格式。名称的转义方式与TabSeparated格式相同，=符号也会被转义。

```
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转化为\n。

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

```
x=1 y=\n
```

当有大量的小列时，这种格式效率十分低下，并且通常没有理由使用它。不过，就效率而言，它并不比JSONEachRow差。这种格式支持数据输出和解析。用于解析时，可以任意指定列的顺序，也可以省略某些列，那些列的值为该列的默认值，一般情况下为0或空白。不支持将可在表中可指定的复杂值设为默认值。

解析时允许出现后没有=的字段tskv。此字段会被忽略。

CSV

按,分隔的数据格式([RFC](#))。

格式化时，每一行的值会用双引号括起，日期和时间也会以双引号包括。数字不用双引号括起，字符串中的双引号会以两个双引号输出，除此之外没有其他规则来做字符转义了。值由分隔符隔开，这个分隔符默认是`,`。每一行使用Unix换行符(`LF,\n`)分隔。

数组序列化成CSV规则如下：首先将数组序列化为TabSeparated格式的字符串，然后将结果字符串用双引号括起后输出到CSV。CSV格式的元组被序列化为单独的列(即它们在元组中的嵌套关系会丢失)。

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

* 默认情况下分隔符是`,`，在[format_csv_delimiter](#)中可以了解更多分隔符配置。

解析的时候，值可以使用或不使用双引号或者单引号括起来。在这种情况下，每行通过分隔符或换行符(CR或LF)区分。违反RFC规则的是，在解析未用引号括起的行时，会忽略前缀和结尾的空格和制表符。对于换行符，Unix(`LF,\n`)，Windows(`CR LF\r\n`)和Mac OS Classic(`CR LF\t\n`)都受支持。

如果启用[input_format_defaults_for_omitted_fields](#)，对应列如果存在未输入的空白，且没有用双引号括起，将用默认值替换。

`NULL`被格式化为`\N`或`NULL`或一个不是引号的其他字符串(详见配置[input_format_csv_unquoted_null_literal_as_null](#)或[input_format_defaults_for_omitted_fields](#))。

CSV格式支持输出总数和极值的方式与TabSeparated相同。

CSVWithNames

会输出带头部的信息(字段列表)，和TabSeparatedWithNames一样。

CustomSeparated

类似于[Template](#)，但它打印或读取所有列，并使用设置[format_custom_escaping_rule](#)和分隔符设置[format_custom_field_delimiter](#),[format_custom_row_before_delimiter](#),[format_custom_row_after_delimiter](#),[format_custom_row_between_delimiter](#),[format_custom_result_before_delimiter](#),[format_custom_result_after_delimiter](#)的转义规则，而不是从格式字符串。

也有CustomSeparatedIgnoreSpaces格式，这个类似于TemplateIgnoreSpaces。

JSON

以JSON格式输出数据。除了数据表之外，它还输出列名和类型，以及一些附加信息：输出行的总数，以及如果没有LIMIT的话可输出的行数。示例：

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

```
{
    "meta": [
        {
            "name": "'hello'",
            "type": "String"
        },
        {
            "name": "multiply(42, number)",
            "type": "UInt64"
        },
        {
            "name": "range(5)",
            "type": "Array(UInt8)"
        }
    ],
    "data": [
        {
            "'hello)": "hello",
            "multiply(42, number)": "0",
            "range(5)": [0,1,2,3,4]
        },
        {
            "'hello)": "hello",
            "multiply(42, number)": "42",
            "range(5)": [0,1,2,3,4]
        },
        {
            "'hello)": "hello",
            "multiply(42, number)": "84",
            "range(5)": [0,1,2,3,4]
        }
    ],
    "rows": 3,
    "rows_before_limit_at_least": 3
}
```

JSON与JavaScript兼容。为了确保这一点，一些字符被另外转义：斜线/被转义为\/；换行符U+2028和U+2029会打断一些浏览器的解析，它们会被转义为\uXXXX。ASCII控制字符被转义：退格，换页，换行，回车和制表符被转义为\b，\f，\n，\r，\t。剩下的0x00-0x1F被转义成相应的\uXXXX序列。无效的UTF-8序列替换为字符◆，使得输出文本包含有效的UTF-8序列。为了与JavaScript兼容，默认情况下，Int64和UInt64整数用双引号引起来。要除去引号，可以将配置参数output_format_json_quote_64bit_integers设置为0。

`rows`代表结果输出的行数。

`rows_before_limit_at_least`代表去掉LIMIT过滤后的最小行总数。只会在查询包含LIMIT条件时输出。若查询包含 GROUP BY，`rows_before_limit_at_least`就是去掉LIMIT后过滤后的准确行数。

`totals` – 总值 (当指定WITH TOTALS时)。

`extremes` – 极值(当extremes设置为1时)。

该格式仅适用于输出查询结果，但不适用于解析输入(将数据插入到表中)。

ClickHouse支持NULL，在JSON输出中显示为null。若要在输出中启用+nan、-nan、+inf、-inf值，请设置output_format_json_quote_denormals为1。

参考

- [JSONEachRow](#)格式
- [output_format_json_array_of_rows](#)设置

JSONStrings

与JSON的不同之处在于数据字段以字符串输出，而不是以类型化JSON值输出。

示例：

```
{  
    "meta":  
    [  
        {  
            "name": "'hello'",  
            "type": "String"  
        },  
        {  
            "name": "multiply(42, number)",  
            "type": "UInt64"  
        },  
        {  
            "name": "range(5)",  
            "type": "Array(UInt8)"  
        }  
    ],  
    "data":  
    [  
        {"  
            "'hello)": "hello",  
            "multiply(42, number)": "0",  
            "range(5)": "[0,1,2,3,4]"  
        },  
        {"  
            "'hello)": "hello",  
            "multiply(42, number)": "42",  
            "range(5)": "[0,1,2,3,4]"  
        },  
        {"  
            "'hello)": "hello",  
            "multiply(42, number)": "84",  
            "range(5)": "[0,1,2,3,4]"  
        }  
    ],  
    "rows": 3,  
    "rows_before_limit_at_least": 3  
}
```

注意range(5)的值。

JSONAsString

在这种格式中，一个JSON对象被解释为一个值。如果输入有几个JSON对象(逗号分隔)，它们将被解释为独立的行。

这种格式只能针对有一列类型为 **String** 的表。表中其余的列必须设置为 **DEFAULT** 或 **MATERIALIZED**，或者忽略。一旦将整个JSON对象收集为字符串，就可以使用**JSON函数**运行它。

示例

查询：

```
DROP TABLE IF EXISTS json_as_string;  
CREATE TABLE json_as_string (json String) ENGINE = Memory;  
INSERT INTO json_as_string FORMAT JSONAsString {"foo":{"bar":{"x":"y"}, "baz":1}}, {}, {"any json structure":1}  
SELECT * FROM json_as_string;
```

结果：

```
json
{"foo": {"bar": {"x": "y"}, "baz": 1} } |
{} |
{"any json stucture": 1} |
```

JSONCompact JSONCompactStrings

与JSON格式不同的是它以数组的方式输出结果，而不是以结构体。

示例：

```
// JSONCompact
{
  "meta":
  [
    {
      "name": "hello",
      "type": "String"
    },
    {
      "name": "multiply(42, number)",
      "type": "UInt64"
    },
    {
      "name": "range(5)",
      "type": "Array(UInt8)"
    }
  ],
  "data":
  [
    ["hello", "0", [0,1,2,3,4]],
    ["hello", "42", [0,1,2,3,4]],
    ["hello", "84", [0,1,2,3,4]]
  ],
  "rows": 3,
  "rows_before_limit_at_least": 3
}
```

```
//JSONCompactStrings
{
    "meta":
    [
        {
            "name": "'hello'",
            "type": "String"
        },
        {
            "name": "multiply(42, number)",
            "type": "UInt64"
        },
        {
            "name": "range(5)",
            "type": "Array(UInt8)"
        }
    ],
    "data":
    [
        ["hello", "0", "[0,1,2,3,4]"],
        ["hello", "42", "[0,1,2,3,4]"],
        ["hello", "84", "[0,1,2,3,4]"]
    ],
    "rows": 3,
    "rows_before_limit_at_least": 3
}
```

JSONEachRow

JSONStringsEachRow

JSONCompactEachRow

JSONCompactStringsEachRow

使用这些格式时，ClickHouse会将行输出为用换行符分隔的JSON值，这些输出数据作为一个整体时，由于没有分隔符(,)因而不是有效的JSON文档。

```
{"some_int":42,"some_str":"hello","some_tuple":[1,"a"]} // JSONEachRow
[42,"hello",[1,"a"]] // JSONCompactEachRow
["42","hello","(2,'a')"] // JSONCompactStringsEachRow
```

在插入数据时，应该为每一行提供一个单独的JSON值。

JSONEachRowWithProgress

JSONStringsEachRowWithProgress

与JSONEachRow/JSONStringsEachRow不同的是，ClickHouse还将生成作为JSON值的进度信息。

```
{"row": {"'hello': 'hello", "multiply(42, number)": "0", "range(5)": "[0,1,2,3,4]"} }
{"row": {"'hello': 'hello", "multiply(42, number)": "42", "range(5)": "[0,1,2,3,4]"} }
{"row": {"'hello': 'hello", "multiply(42, number)": "84", "range(5)": "[0,1,2,3,4]"} }
{"progress": {"read_rows": "3", "read_bytes": "24", "written_rows": "0", "written_bytes": "0", "total_rows_to_read": "3"}}
```

JSONCompactEachRowWithNamesAndTypes

JSONCompactStringsEachRowWithNamesAndTypes

与JSONCompactEachRow/JSONCompactStringsEachRow不同的是，列名和类型被写入前两行。

```
[["hello", "multiply(42, number)", "range(5)"]
["String", "UInt64", "Array(UInt8)"]
["hello", "0", [0,1,2,3,4]]
["hello", "42", [0,1,2,3,4]]
["hello", "84", [0,1,2,3,4]]]
```

Inserting Data

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

ClickHouse允许：

- 以任意顺序排列列名，后跟对应的值。
- 省略一些值。

ClickHouse忽略元素之间的空格和对象后面的逗号。您可以在一行中传递所有对象，不需要用换行符把它们分开。

省略值处理

ClickHouse将省略的值替换为对应的[数据类型](#)默认值。

如果指定了[DEFAULT expr](#)，则ClickHouse根据属性使用不同的替换规则，详看[input_format_defaults_for_omitted_fields](#)设置。

参考下面的例子：

```
CREATE TABLE IF NOT EXISTS example_table
(
    x UInt32,
    a DEFAULT x * 2
) ENGINE = Memory;
```

- 如果[input_format_defaults_for_omitted_fields = 0](#)，那么x和a的默认值等于0(作为UInt32数据类型的默认值)。
- 如果[input_format_defaults_for_omitted_fields = 1](#)，那么x的默认值为0，但a的默认值为x * 2。

注意

当使用[insert_sample_with_metadata = 1](#)插入数据时，与使用[insert_sample_with_metadata = 0](#)相比，ClickHouse消耗更多的计算资源。

Selecting Data

以UserActivity表为例：

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	-1
4324182021466249494	6	185	1

当查询[SELECT * FROM UserActivity FORMAT JSONEachRow](#)返回：

```
{"UserID":"4324182021466249494","PageViews":5,"Duration":146,"Sign":-1}
{"UserID":"4324182021466249494","PageViews":6,"Duration":185,"Sign":1}
```

与JSON格式不同，没有替换无效的UTF-8序列。值以与JSON相同的方式转义。

提示

字符串中可以输出任意一组字节。如果您确信表中的数据可以被格式化为JSON而不会丢失任何信息，那么就使用`JSONEachRow`格式。

Nested Structures

如果您有一个包含Nested数据类型列的表，您可以插入具有相同结构的JSON数据。使用`input_format_import_nested_json`设置启用该特性。

例如，请参考下表：

```
CREATE TABLE json_each_row_nested (n Nested (s String, i Int32) ) ENGINE = Memory
```

正如您在Nested数据类型描述中看到的，ClickHouse将嵌套结构的每个部分作为一个单独的列(`n.s`和`n.i`)。您可以通过以下方式插入数据：

```
INSERT INTO json_each_row_nested FORMAT JSONEachRow {"n.s": ["abc", "def"], "n.i": [1, 23]}
```

将数据作为分层JSON对象集插入，需要设置`input_format_import_nested_json=1`。

```
{  
  "n": {  
    "s": ["abc", "def"],  
    "i": [1, 23]  
  }  
}
```

如果没有此设置，ClickHouse将引发异常。

```
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]

Native

最高性能的格式。通过二进制格式的块进行写入和读取。对于每个块，该中的行数，列数，列名称和类型以及列的部分将被相继记录。换句话说，这种格式是columnar的 - 它不会将列转换为行。这种格式用于服务器间交互、命令行客户端和C++客户端与服务器交互。

您可以使用此格式快速生成只能由ClickHouse DBMS读取的格式。但自己处理这种格式是没有意义的。

Null

没有输出。但是，查询已处理完毕，并且在使用命令行客户端时，数据将传输到客户端。这仅用于测试，包括性能测试。显然，这种格式只适用于输出，不适用于解析。

Pretty

将数据以表格形式输出，也可以使用ANSI转义字符在终端中设置颜色。

它会绘制一个完整的表格，每行数据在终端中占用两行。

每个结果块作为一个单独的表输出。这是必要的，以便在输出块时不缓冲结果(为了预先计算所有值的可见宽度，缓冲是必要的)。

NULL输出为 `NULL`。

示例

```
SELECT * FROM system.numbers limit 2 format Pretty;
```

number
0
1

Pretty的所有格式不进行字符转义。示例显示了PrettyCompact格式:

```
SELECT 'String with \'quotes\' and \t character' AS Escaping_test
```

```
Escaping_test
String with 'quotes' and      character |
```

为避免将太多数据传输到终端，只打印前10,000行。如果行数大于或等于10,000，则会显示消息Showed first 10 000。该格式仅适用于输出查询结果，但不适用于解析输入(将数据插入到表中)。

Pretty格式支持输出合计值(当使用WITH TOTALS时)和极值(当extremes设置为1时)。在这些情况下，合计值和极值将输出在主要数据之后，在单独的表中。示例(显示为PrettyCompact格式):

```
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

PrettyCompact

与 Pretty 格式不一样的是 PrettyCompact 去掉了行之间的表格分割线，这样使得结果更加紧凑。
这种格式会在交互命令行客户端下默认使用。

```
select * from system.numbers limit 2 format PrettyCompact;
```

number
0
1

PrettyCompactMonoBlock

与 PrettyCompact 格式不一样的是，它支持 10,000 行数据缓冲，然后输出在一个表格中，而不分块。

PrettyNoEscapes

与 Pretty 格式不一样的是，它不使用 ANSI 字符转义，这在浏览器显示数据以及在使用 watch 命令行工具是有必要的。

示例：

```
watch -n1 "clickhouse-client --query='SELECT event, value FROM system.events FORMAT PrettyCompactNoEscapes'"
```

您可以使用 HTTP 接口来获取数据，显示在浏览器中。

PrettyCompactNoEscapes

用法类似上述。

PrettySpaceNoEscapes

用法类似上述。

PrettySpace

与 PrettyCompact 格式不一样的是，它使用空格来代替网格来显示数据。

```
select * from system.numbers limit 2 format PrettySpace;
```

number

```
0  
1
```

RowBinary

以二进制格式逐行格式化和解析数据。行和值连续列出，没有分隔符。

这种格式比 Native 格式效率低，因为它是基于行的。

整数使用固定长度的小端表示法。例如，UInt64 使用 8 个字节。

DateTime 被表示为 UInt32 类型的 Unix 时间戳值。

Date 被表示为 UInt16 对象，它的值为自 1970-01-01 以来经过的天数。

字符串表示为 varint 长度(无符号 LEB128)，后跟字符串的字节数。

FixedString 被简单地表示为字节序列。

数组表示为 varint 长度(无符号 LEB128)，后跟有序的数组元素。

对于 NULL 的支持，一个为 1 或 0 的字节会加在每个 可为空 值前面。如果为 1，那么该值就是 NULL。如果为 0，则不为 NULL。

RowBinaryWithNamesAndTypes

类似于 RowBinary，但添加了头部信息：

- LEB128-编码列数(N)
- N Strings 指定列名
- N Strings 指定列类型

Values

在括号中打印每一行。行由逗号分隔。最后一行之后没有逗号。括号内的值也用逗号分隔。数字以十进制格式输出，不含引号。数组以方括号输出。字符串、日期、日期时间用引号包围输出。转义字符的解析规则与 TabSeparated 格式类似。在格式化过程中，不插入额外的空格，但在解析过程中，空格是被允许并跳过的(除了数组值之外的空格，这是不允许的)。NULL 为 NULL。

以 Values 格式传递数据时需要转义的最小字符集是：单引号和反斜线。

这是 INSERT INTO t VALUES ... 中可以使用的格式，但您也可以将其用于查询结果。

另

见：[input_format_values_interpret_expressions](#) 和 [input_format_values_deduce_templates_of_expressions](#)。

Vertical

根据指定的列名，打印出每一行的值。这种格式适用于具有大量的列时，显示几个列。NULL 输出为 NULL。

示例：

```
SELECT * FROM t_null FORMAT Vertical
```

Row 1:

x: 1
y: NULL

该格式仅适用于输出查询结果，但不适用于解析输入(将数据插入到表中)。

XML

该格式仅适用于输出查询结果，但不适用于解析输入，示例：

```
<?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>
```

如果列名称没有可接受的格式，则仅使用 `field` 作为元素名称。通常，XML 结构遵循 JSON 结构。
就像JSON一样，将无效的 UTF-8 字符都替换成字符◆，以便输出文本将包含有效的 UTF-8 字符序列。

在字符串值中，字符 `<` 和 `&` 被转义为 `<` 和 `&`。

数组输出类似于 `<array> <elem> Hello </ elem> <elem> World </ elem> ... </ array>` 元组输出类似于 `<tuple> <elem> Hello </ elem> <elem> World </ ELEM> ... </tuple>`。

CapnProto

Cap'n Proto 是一种二进制消息格式，类似Protobuf和Thriftis，但与 JSON 或 MessagePack 格式不一样。

Cap'n Proto 消息格式是严格类型的，而不是自我描述，这意味着它们需要架构描述。架构描述可以实时地应用，并针对每个查询进行缓存。

```
SELECT SearchPhrase, count() AS c FROM test.hits  
GROUP BY SearchPhrase FORMAT CapnProto SETTINGS schema = 'schema:Message'
```

其中 `schema.capnp` 描述如下：6y2

```
struct Message {  
    SearchPhrase @0 :Text;  
    c @1 :Uint64;  
}
```

格式文件存储的目录可以在服务配置中的 `format_schema_path` 指定。

Cap'n Proto 反序列化是很高效的，通常不会增加系统的负载。

Protobuf

Protobuf-是一个 **Protocol Buffers** 格式。

此格式需要外部格式描述文件(proto文件)。该描述文件会进行缓存，以备后续查询。

ClickHouse支持 `proto2` 和 `proto3` 语法的proto文件，支持重复/可选/必填字段。

使用示例：

```
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'"
```

proto文件 `schemafile.proto` 看起来像这样：

```
syntax = "proto3";  
  
message MessageType {  
    string name = 1;  
    string surname = 2;  
    uint32 birthDate = 3;  
    repeated string phoneNumbers = 4;  
};
```

Clickhouse通过字段名称来对应列名称。字段名称不区分大小写，`_`与`.`视为相同符号。如果Proto文件指定的字段类型与列类型不相符，会进行转换。

支持Protobuf嵌套消息。例如，对于下面Proto文件中的z字段：

```

message MessageType {
    message XType {
        message YType {
            int32 z;
        };
        repeated YType y;
    };
    XType x;
}

```

ClickHouse会试图找到一个名为 `x.y.z` (或 `x_y_z` 或 `X.y_Z` 等)的列。

嵌套消息适用于输入或输出一个 [嵌套数据结构](#).

在protobuf模式中定义的默认值，如下：

```

syntax = "proto2";

message MessageType {
    optional int32 result_per_page = 3 [default = 10];
}

```

该默认值会被忽略，Clickhouse会使用 [表默认值](#)作为默认值。

ClickHouse在输入和输出protobuf消息采用length-delimited 格式。

这意味着每个消息之前，应该写它的长度作为一个 `varint`.

另请参阅 [如何在流行语言中读取/写入长度分隔的protobuf消息](#).

Avro

[Apache Avro](#) 是在Apache Hadoop项目中开发的面向行的数据序列化框架。

ClickHouse Avro格式支持读取和写入 [Avro数据文件](#).

数据类型匹配{#sql_reference/data_types-matching}

下表显示了支持的数据类型以及它们如何匹配ClickHouse [数据类型](#) 在 `INSERT` 和 `SELECT` 查询。

Avro数据类型 <code>INSERT</code>	ClickHouse数据类型	Avro数据类型 <code>SELECT</code>
<code>boolean, int, long, float, double</code>	<code>Int(8 16 32), UInt(8 16 32)</code>	<code>int</code>
<code>boolean, int, long, float, double</code>	<code>Int64, UInt64</code>	<code>long</code>
<code>boolean, int, long, float, double</code>	<code>Float32</code>	<code>float</code>
<code>boolean, int, long, float, double</code>	<code>Float64</code>	<code>double</code>
<code>bytes, string, fixed, enum</code>	<code>String</code>	<code>bytes</code>
<code>bytes, string, fixed</code>	<code>FixedSize(N)</code>	<code>fixed(N)</code>
<code>enum</code>	<code>Enum(8 16)</code>	<code>enum</code>
<code>array(T)</code>	<code>Array(T)</code>	<code>array(T)</code>
<code>union(null, T), union(T, null)</code>	<code>Nullable(T)</code>	<code>union(null, T)</code>

Avro数据类型 INSERT	ClickHouse数据类型	Avro数据类型 SELECT
null	Nullable(Nothing)	null
int (date) *	Date	int (date) *
long (timestamp-millis) *	DateTime64(3)	long (timestamp-millis) *
long (timestamp-micros) *	DateTime64(6)	long (timestamp-micros) *

* Avro逻辑类型

不支持的Avro数据类型: record (非根架构), map

不支持的Avro逻辑数据类型: time-millis, time-micros, duration

插入数据

将Avro文件中的数据插入ClickHouse表:

```
$ cat file.avro | clickhouse-client --query="INSERT INTO {some_table} FORMAT Avro"
```

输入Avro文件的根架构必须是 record 类型。

Clickhouse通过字段名称来对应架构的列名称。字段名称区分大小写。未使用的字段会被跳过。

ClickHouse表列的数据类型可能与插入的Avro数据的相应字段不同。插入数据时，ClickHouse根据上表解释数据类型，然后通过 **Cast** 将数据转换为相应的列类型。

选择数据

从ClickHouse表中选择数据到Avro文件:

```
$ clickhouse-client --query="SELECT * FROM {some_table} FORMAT Avro" > file.avro
```

列名必须:

- 以 [A-Za-z_] 开始
- 随后只包含 [A-Za-z0-9_]

输出Avro文件压缩和同步间隔可以经由 **output_format_avro_codec** 和 **output_format_avro_sync_interval** 设置。

AvroConfluent

AvroConfluent支持解码单个对象的Avro消息，这常用于 **Kafka** 和 **Confluent Schema Registry**。

每个Avro消息都嵌入了一个架构id，该架构id可以在架构注册表的帮助下解析为实际架构。

模式解析后会进行缓存。

架构注册表URL配置为 **format_avro_schema_registry_url**

数据类型匹配 **{#sql_reference/data_types-matching-1}**

和 **Avro** 相同。

用途

要快速验证架构解析，您可以使用 **kafkacat** 与 **clickhouse-local**:

```
$ 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 **AvroConfluent** 与 **Kafka**:

```
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

设置 `format_avro_schema_registry_url` 需要写入配置文件 `users.xml` 以在 ClickHouse 重启后，该设置仍为您的设定值。您也可以在使用 Kafka 引擎的时候指定该设置。

Parquet

Apache Parquet 是 Hadoop 生态系统中普遍使用的列式存储格式。ClickHouse 支持此格式的读写操作。

数据类型匹配 {#sql_reference/data_types-matching-2}

下表显示了 ClickHouse 支持的数据类型以及它们在 `INSERT` 和 `SELECT` 查询如何对应 ClickHouse 的 **data types**。

Parquet 数据类型 (<code>INSERT</code>)	ClickHouse 数据类型	Parquet 数据类型 (<code>SELECT</code>)
UINT8, BOOL	UInt8	UINT8
INT8	Int8	INT8
UINT16	UInt16	UINT16
INT16	Int16	INT16
UINT32	UInt32	UINT32
INT32	Int32	INT32
UINT64	UInt64	UINT64
INT64	Int64	INT64

Parquet数据类型 (<code>INSERT</code>)	ClickHouse数据类型	Parquet数据类型 (<code>SELECT</code>)
FLOAT, HALF_FLOAT	Float32	FLOAT
DOUBLE	Float64	DOUBLE
DATE32	Date	UINT16
DATE64, TIMESTAMP	DateTime	UINT32
STRING, BINARY	String	STRING
—	FixedString	STRING
DECIMAL	Decimal	DECIMAL

ClickHouse支持对 `Decimal` 类型设置精度。 `INSERT` 查询将 Parquet `DECIMAL` 类型视为 ClickHouse `Decimal128` 类型。

不支持的Parquet数据类型: DATE32, TIME32, FIXED_SIZE_BINARY, JSON, UUID, ENUM.

ClickHouse表列的数据类型可能与插入的Parquet数据的相应字段不同。 插入数据时，ClickHouse根据上表解释数据类型，然后 `Cast` 为ClickHouse表列设置的数据类型的数据。

插入和选择数据

您可以通过以下命令将Parquet数据从文件插入到ClickHouse表中:

```
$ cat {filename} | clickhouse-client --query="INSERT INTO {some_table} FORMAT Parquet"
```

您可以从ClickHouse表中选择数据，并通过以下命令将它们保存到Parquet格式的文件中:

```
$ clickhouse-client --query="SELECT * FROM {some_table} FORMAT Parquet" > {some_file.pq}
```

要与Hadoop交换数据，您可以使用 `HDFS table engine`.

Arrow

`Apache Arrow`是一种用于内存数据库的格式，共有两种模式，文件与流模式。Clickhouse支持对这两种格式进行读写。

`Arrow`对应的是文件模式，这种格式适用于内存的随机访问。

ArrowStream

`ArrowStream`对应的是Arrow的流模式，这种格式适用于内存的流式处理。

ORC

`Apache ORC` 是Hadoop生态系统中普遍存在的列式存储格式。

数据类型匹配{#sql_reference/data_types-matching-3}

下表显示了支持的数据类型以及它们如何在 `SELECT` 与 `INSERT` 查询中匹配ClickHouse的 `数据类型`。

ORC 数据类型 (<code>INSERT</code>)	ClickHouse 数据类型	ORC 数据类型 (<code>SELECT</code>)
UINT8, BOOL	UInt8	UINT8
INT8	Int8	INT8
UINT16	UInt16	UINT16
INT16	Int16	INT16
UINT32	UInt32	UINT32
INT32	Int32	INT32
UINT64	UInt64	UINT64
INT64	Int64	INT64
FLOAT, HALF_FLOAT	Float32	FLOAT
DOUBLE	Float64	DOUBLE
DATE32	Date	DATE32
DATE64, TIMESTAMP	DateTime	TIMESTAMP
STRING, BINARY	String	BINARY
DECIMAL	Decimal	DECIMAL
-	Array	LIST

ClickHouse 支持的可配置精度的 `Decimal` 类型。`INSERT` 查询将 ORC 格式的 `DECIMAL` 类型视为 ClickHouse 的 `Decimal128` 类型。

不支持的 ORC 数据类型: `TIME32`, `FIXED_SIZE_BINARY`, `JSON`, `UUID`, `ENUM`.

ClickHouse 表列的数据类型不必匹配相应的 ORC 数据字段。插入数据时, ClickHouse 根据上表解释数据类型, 然后 `Cast` 将数据转换为 ClickHouse 表列的数据类型集。

插入数据

您可以通过以下命令将文件中的 ORC 数据插入到 ClickHouse 表中:

```
$ cat filename.orc | clickhouse-client --query="INSERT INTO some_table FORMAT ORC"
```

选择数据

您可以通过以下命令将 ClickHouse 表中某些数据导出到 ORC 文件:

```
$ clickhouse-client --query="SELECT * FROM {some_table} FORMAT ORC" > {filename.orc}
```

要与 Hadoop 交换数据, 您可以使用 [HDFS 表引擎](#).

LineAsString

这种格式下，每行输入数据都会当做一个字符串。这种格式仅适用于仅有一列String类型的列的表。其余列必须设置为DEFAULT、MATERIALIZED或者被忽略。

示例：

查询如下：

```
DROP TABLE IF EXISTS line_as_string;
CREATE TABLE line_as_string (field String) ENGINE = Memory;
INSERT INTO line_as_string FORMAT LineAsString "I love apple", "I love banana", "I love orange";
SELECT * FROM line_as_string;
```

结果如下：

```
field
"I love apple", "I love banana", "I love orange"; |
```

Regexp

每一列输入数据根据正则表达式解析。使用Regexp格式时，可以使用如下设置：

- `format_regex`, String类型。包含re2格式的正则表达式。
- `format_regex_escaping_rule`, String类型。支持如下转义规则：
 - CSV(规则相同于CSV)
 - JSON(相同于JSONEachRow)
 - Escaped(相同于TSV)
 - Quoted(相同于Values)
 - Raw(将整个子匹配项进行提取，不转义)
- `format_regex_skip_unmatched`, UInt8类型。当`format_regex`表达式没有匹配到结果时是否抛出异常。可为0或1。

用法

`format_regex`设置会应用于每一行输入数据。正则表达式的子匹配项数必须等于输入数据期望得到的列数。

每一行输入数据通过换行符\n或者\r\n分隔。

匹配到的子匹配项会根据每一列的数据格式进行解析，转义规则根据`format_regex_escaping_rule`进行。

当正则表达式对某行没有匹配到结果，`format_regex_skip_unmatched`设为1时，该行会被跳过。`format_regex_skip_unmatched`设为0时，会抛出异常。

示例

设有如下data.tsv:

```
id: 1 array: [1,2,3] string: str1 date: 2020-01-01
id: 2 array: [1,2,3] string: str2 date: 2020-01-02
id: 3 array: [1,2,3] string: str3 date: 2020-01-03
```

与表：

```
CREATE TABLE imp_regex_table (id UInt32, array Array(UInt32), string String, date Date) ENGINE = Memory;
```

导入命令：

```
$ cat data.tsv | clickhouse-client --query "INSERT INTO imp_regex_table FORMAT Regexp SETTINGS  
format_regex='id: (.+?) array: (.+?) string: (.+?) date: (.+?)', format_regex_escaping_rule='Escaped',  
format_regex_skip_unmatched=0;"
```

查询：

```
SELECT * FROM imp_regex_table;
```

结果：

id	array	string	date
1	[1,2,3]	str1	2020-01-01
2	[1,2,3]	str2	2020-01-02
3	[1,2,3]	str3	2020-01-03

RawBLOB

这种格式下，所有输入数据视为一个值。该格式仅适用于仅有String类型的列的表。输出时，使用二进制格式输出。当输出结果不唯一时，输出是有歧义的，并且不能通过该输出还原原数据。

下面是RawBLOB与TabSeparatedRaw的对比：

RawBLOB:

- 二进制格式输出，无转义。
- 值之间没有分隔符。
- 每行最后的值后面没有换行符。

TabSeparatedRaw:

- 数据无转义输出。
- 每行的值通过制表符分隔。
- 每行最后的值得后面有换行符。

下面是RawBLOB与RowBinary的对比：

RawBLOB:

- 字符串前面没有表示长度的标志

RowBinary:

- 字符串前面有变长标志(LEB128格式表示)，用于表示字符串长度，后接字符串内容。

当传入空数据，Clickhouse会抛出异常：

```
Code: 108. DB::Exception: No data to insert
```

示例

```
$ clickhouse-client --query "CREATE TABLE {some_table} (a String) ENGINE = Memory;"  
$ cat {filename} | clickhouse-client --query="INSERT INTO {some_table} FORMAT RawBLOB"  
$ clickhouse-client --query "SELECT * FROM {some_table} FORMAT RawBLOB" | md5sum
```

结果：

```
f9725a22f9191e064120d718e26862a9 -
```

格式架构

包含格式架构的文件名由设置 `format_schema` 指定。当使用 `CapnProto` 或 `Protobuf` 其中一种格式时，需要设置该项。

格式架构为架构文件名和此文件中消息类型的组合，用冒号分隔，例如 `schemafile.proto:MessageType`。

如果文件具有格式的标准扩展名（例如，`Protobuf` 格式的架构文件标准扩展名为 `.proto`），它可以被省略，在这种情况下，格式模式如下所示 `schemafile:MessageType`。

如果您通过 `Client` 在 `交互模式` 下输入或输出数据，格式架构中指定的文件名可以使用绝对路径或客户端当前目录的相对路径。

如果在 `批处理模式` 下使用客户端，则由于安全原因，架构的路径必须使用相对路径。

如果您通过 `HTTP` 接口 `[../interfaces/http.md]` 输入或输出数据，格式架构中指定的文件名应该位于服务器设置的 `format_schema_path` 指定的目录中。

跳过错误

一些格式，如 `CSV`, `TabSeparated`, `TSKV`, `JSONEachRow`, `Template`, `CustomSeparated` 和 `Protobuf` 如果发生解析错误，可以跳过引发错误的行，并从下一行开始继续解析。详情请见设置 `input_format_allow_errors_num` 和 `input_format_allow_errors_ratio`。

限制：

- 在解析错误的情况下 `JSONEachRow` 跳过该行的所有数据，直到遇到新行（或 `EOF`），所以行必须由换行符分隔以正确统计错误行的数量。
- `Template` 和 `CustomSeparated` 在最后一列之后和行之间使用分隔符来查找下一行的开头，所以跳过错误只有在行分隔符和列分隔符其中至少有一个不为空时才有效。

JDBC 驱动

- [官方驱动](#)
- 第三方驱动：
 - [ClickHouse-Native-JDBC](#)
 - [clickhouse4j](#)

ODBC 驱动

- [官方驱动](#)。

C++ 客户端库

请参考仓库的描述文件 [clickhouse-cpp](#)。

第三方工具

这是第三方工具的链接集合，它们提供了一些ClickHouse的接口。它可以是可视化界面、命令行界面或API:

- Client libraries
- Integrations
- GUI
- Proxies

注意

支持通用API的通用工具[ODBC](#)或[JDBC](#)，通常也适用于ClickHouse，但这里没有列出，因为它们实在太多了。

第三方开发库

声明

Yandex没有维护下面列出的库，也没有做过任何广泛的测试来确保它们的质量。

- Python
 - [infi.clickhouse_orm](#)
 - [clickhouse-driver](#)
 - [clickhouse-client](#)
 - [aiochclient](#)
 - [asynch](#)
- PHP
 - [smi2/phpclickhouse](#)
 - [8bitov/clickhouse-php-client](#)
 - [bozerkins/clickhouse-client](#)
 - [simpod/clickhouse-client](#)
 - [seva-code/php-click-house-client](#)
 - [SeasClick C++ client](#)
 - [one-ck](#)
 - [glushkovds/phpclickhouse-laravel](#)
- Go
 - [clickhouse](#)
 - [go-clickhouse](#)
 - [mailrugo-clickhouse](#)
 - [golang-clickhouse](#)

- Swift
 - ClickHouseNIO
 - ClickHouseVapor ORM
- NodeJs
 - clickhouse (NodeJs)
 - node-clickhouse
- Perl
 - perl-DBD-ClickHouse
 - HTTP-ClickHouse
 - AnyEvent-ClickHouse
- Ruby
 - ClickHouse (Ruby)
 - clickhouse-activerecord
- R
 - clickhouse-r
 - RClickHouse
- Java
 - clickhouse-client-java
 - clickhouse-client
- Scala
 - clickhouse-scala-client
- Kotlin
 - AORM
- C#
 - Octonica.ClickHouseClient
 - ClickHouse.Ado
 - ClickHouse.Client
 - ClickHouse.Net
- Elixir
 - clickhousex
 - pillar
- Nim
 - nim-clickhouse
- Haskell
 - hdbc-clickhouse

声明

Yandex没有维护下面列出的库，也没有做过任何广泛的测试来确保它们的质量。

基础设施

- 关系数据库
 - MySQL
 - mysql2ch
 - ProxySQL
 - clickhouse-mysql-data-reader
 - horgh-replicator
 - PostgreSQL
 - clickhousedb_fdw
 - infi.clickhouse_fdw (uses `infi.clickhouse_orm`)
 - pg2ch
 - clickhouse_fdw
 - MSSQL
 - ClickHouseMigrator
- 消息队列
 - Kafka
 - clickhouse_sinker (uses Go client)
 - stream-loader-clickhouse
- 流处理
 - Flink
 - flink-clickhouse-sink
- 对象存储
 - S3
 - clickhouse-backup
- 容器编排
 - Kubernetes
 - clickhouse-operator
- 配置管理
 - puppet
 - innogames/clickhouse
 - mfedorov/clickhouse

- Monitoring
 - Graphite
 - graphouse
 - carbon-clickhouse
 - graphite-clickhouse
 - graphite-ch-optimizer - optimizes stale partitions in *GraphiteMergeTree if rules from rollup configuration could be applied
 - Grafana
 - clickhouse-grafana
 - Prometheus
 - clickhouse_exporter
 - PromHouse
 - clickhouse_exporter (uses Go client)
 - Nagios
 - check_clickhouse
 - check_clickhouse.py
 - Zabbix
 - clickhouse-zabbix-template
 - Sematext
 - clickhouse integration
- Logging
 - rsyslog
 - omclickhouse
 - fluentd
 - loghouse (for Kubernetes)
 - logagent
 - logagent output-plugin-clickhouse
- Geo
 - MaxMind
 - clickhouse-maxmind-geoip

编程语言

- Python
 - SQLAlchemy
 - sqlalchemy-clickhouse (uses infi.clickhouse_orm)
 - pandas
 - pandahouse
- PHP
 - Doctrine
 - dbal-clickhouse

- R
 - [dplyr](#)
 - [RClickHouse](#) (uses [clickhouse-cpp](#))
- Java
 - [Hadoop](#)
 - [clickhouse-hdfs-loader](#) (uses [JDBC](#))
- Scala
 - [Akka](#)
 - [clickhouse-scala-client](#)
- C#
 - [ADO.NET](#)
 - [ClickHouse.Ado](#)
 - [ClickHouse.Client](#)
 - [ClickHouse.Net](#)
 - [ClickHouse.Net.Migrations](#)
- Elixir
 - [Ecto](#)
 - [clickhouse_ecto](#)
- Ruby
 - [Ruby on Rails](#)
 - [activecube](#)
 - [ActiveRecord](#)
 - [GraphQL](#)
 - [activecube-graphql](#)

第三方代理

chproxy

[chproxy](#), 是一个用于ClickHouse数据库的HTTP代理和负载均衡器。

特性:

- 用户路由和响应缓存。
- 灵活的限制。
- 自动SSL证书续订。

使用go语言实现。

KittenHouse

[KittenHouse](#)被设计为ClickHouse和应用服务器之间的本地代理，以防不可能或不方便在应用程序端缓冲插入数据。

特性:

- 内存和磁盘上的数据缓冲。
- 表路由。

- 负载平衡和运行状况检查。

使用 go 语言实现。

ClickHouse-Bulk

[ClickHouse-Bulk](#) 是一个简单的 ClickHouse 收集器。

特性：

- 按阈值或间隔对请求进行分组并发送。
- 多个远程服务器。
- 基本身份验证。

使用 go 语言实现。

第三方开发的可视化界面

开源

Tabix

ClickHouse Web 界面 [Tabix](#).

主要功能：

- 浏览器直接连接 ClickHouse，不需要安装其他软件。
- 高亮语法的编辑器。
- 自动命令补全。
- 查询命令执行的图形分析工具。
- 配色方案选项。

[Tabix 文档](#).

HouseOps

[HouseOps](#) 是一个交互式 UI/IDE 工具，可以运行在 OSX, Linux and Windows 平台中。

主要功能：

- 查询高亮语法提示，可以以表格或 JSON 格式查看数据。
- 支持导出 CSV 或 JSON 格式数据。
- 支持查看查询执行的详情，支持 KILL 查询。
- 图形化显示，支持显示数据库中所有的表和列的详细信息。
- 快速查看列占用的空间。
- 服务配置。

以下功能正在计划开发：

- 数据库管理
- 用户管理
- 实时数据分析
- 集群监控
- 集群管理
- 监控副本情况以及 Kafka 引擎表

灯塔

灯塔 是ClickHouse的轻量级Web界面。

特征：

- 包含过滤和元数据的表列表。
- 带有过滤和排序的表格预览。
- 只读查询执行。

DBeaver

DBeaver 具有ClickHouse支持的通用桌面数据库客户端。

特征：

- 使用语法高亮显示查询开发。
- 表格预览。
- 自动完成。

clickhouse-cli

clickhouse-cli 是ClickHouse的替代命令行客户端，用Python 3编写。

特征：

- 自动完成。
- 查询和数据输出的语法高亮显示。
- 寻呼机支持数据输出。
- 自定义PostgreSQL类命令。

clickhouse-flamegraph

[clickhouse-flamegraph](<https://github.com/Slach/clickhouse-flamegraph>) 是一个可视化的专业工具`system.trace_log`如[flamegraph](<http://www.brendangregg.com/flamegraphs.html>).

DBM

DBM DBM是一款ClickHouse可视化管理工具！

特征：

- 支持查询历史（分页、全部清除等）
- 支持选中的sql子句查询(多窗口等)
- 支持终止查询

- 支持表管理
- 支持数据库管理
- 支持自定义查询
- 支持多数据源管理（连接测试、监控）
- 支持监控（处理进程、连接、查询）
- 支持迁移数据

商业

Holistics

Holistics 在2019年被Gartner FrontRunners列为可用性最高排名第二的商业智能工具之一。 Holistics是一个基于SQL的全栈数据平台和商业智能工具，用于设置您的分析流程。

特征：

- 自动化的电子邮件，Slack和Google表格报告时间表。
- 强大的SQL编辑器，具有版本控制，自动完成，可重用的查询组件和动态过滤器。
- 通过iframe在自己的网站或页面中嵌入仪表板。
- 数据准备和ETL功能。
- SQL数据建模支持数据的关系映射。

DataGrip

DataGrip 是JetBrains的数据库IDE，专门支持ClickHouse。 它还嵌入到其他基于IntelliJ的工具中：PyCharm，IntelliJ IDEA，GoLand，PhpStorm等。

特征：

- 非常快速的代码完成。
- ClickHouse语法高亮显示。
- 支持ClickHouse特有的功能，例如嵌套列，表引擎。
- 数据编辑器。
- 重构。
- 搜索和导航。

数据库引擎

数据库引擎允许您处理数据表。

默认情况下，ClickHouse使用**Atomic**数据库引擎。它提供了可配置的table engines和SQL dialect。

您还可以使用以下数据库引擎：

- MySQL
- MaterializeMySQL
- Lazy

- [Atomic](#)
- [PostgreSQL](#)
- [MaterializedPostgreSQL](#)
- [Replicated](#)

[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	<code>Int8</code>
SHORT	<code>Int16</code>
INT24	<code>Int32</code>
LONG	<code>UInt32</code>
LONGLONG	<code>UInt64</code>
FLOAT	<code>Float32</code>
DOUBLE	<code>Float64</code>
DECIMAL, NEWDECIMAL	<code>Decimal</code>

MySQL	ClickHouse
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
2	222	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] MaterializedMySQL

这是一个实验性的特性，不应该在生产中使用。

创建ClickHouse数据库，包含MySQL中所有的表，以及这些表中的所有数据。

ClickHouse服务器作为MySQL副本工作。它读取binlog并执行DDL和DML查询。

这个功能是实验性的。

创建数据库

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

引擎参数

- host:port — MySQL服务地址
- database — MySQL数据库名称
- user — MySQL用户名
- password — MySQL用户密码

引擎配置

- max_rows_in_buffer — 允许数据缓存到内存中的最大行数(对于单个表和无法查询的缓存数据)。当超过行数时，数据将被物化。默认值: 65505。
- max_bytes_in_buffer — 允许在内存中缓存数据的最大字节数(对于单个表和无法查询的缓存数据)。当超过行数时，数据将被物化。默认值: 1048576.
- max_rows_in_buffers — 允许数据缓存到内存中的最大行数(对于数据库和无法查询的缓存数据)。当超过行数时，数据将被物化。默认值: 65505.
- max_bytes_in_buffers — 允许在内存中缓存数据的最大字节数(对于数据库和无法查询的缓存数据)。当超过行数时，数据将被物化。默认值: 1048576.
- max_flush_data_time — 允许数据在内存中缓存的最大毫秒数(对于数据库和无法查询的缓存数据)。当超过这个时间时，数据将被物化。默认值: 1000.
- max_wait_time_when_mysql_unavailable — 当MySQL不可用时重试间隔(毫秒)。负值禁止重试。默认值: 1000.

- `allows_query_when_mysql_lost` — 当mysql丢失时，允许查询物化表。默认值: 0 (false).

```
CREATE DATABASE mysql ENGINE = MaterializeMySQL('localhost:3306', 'db', 'user', '***')
SETTINGS
    allows_query_when_mysql_lost=true,
    max_wait_time_when_mysql_unavailable=10000;
```

MySQL服务器端配置

为了MaterializeMySQL正确的工作，有一些强制性的MySQL侧配置设置应该设置：

- `default_authentication_plugin = mysql_native_password`，因为MaterializeMySQL只能使用此方法授权。
- `gtid_mode = on`，因为要提供正确的MaterializeMySQL复制，基于GTID的日志记录是必须的。注意，在打开这个模式On时，你还应该指定`enforce_gtid_consistency = on`。

虚拟列

当使用MaterializeMySQL数据库引擎时，ReplacingMergeTree表与虚拟的`_sign`和`_version`列一起使用。

- `_version` — 同步版本。类型`UInt64`.
- `_sign` — 删除标记。类型`Int8`. Possible values:
 - 1 — 行不会删除，
 - -1 — 行被删除。

支持的数据类型

MySQL	ClickHouse
TINY	<code>Int8</code>
SHORT	<code>Int16</code>
INT24	<code>Int32</code>
LONG	<code>UInt32</code>
LONGLONG	<code>UInt64</code>
FLOAT	<code>Float32</code>
DOUBLE	<code>Float64</code>
DECIMAL, NEWDECIMAL	<code>Decimal</code>
DATE, NEWDATE	<code>Date</code>
DATETIME, TIMESTAMP	<code>DateTime</code>
DATETIME2, TIMESTAMP2	<code>DateTime64</code>
ENUM	<code>Enum</code>
STRING	<code>String</code>

MySQL	ClickHouse
VARCHAR, VAR_STRING	String
BLOB	String
BINARY	FixedString

不支持其他类型。如果MySQL表包含此类类型的列，ClickHouse抛出异常"Unhandled data type"并停止复制。

Nullable已经支持

使用方式

兼容性限制

除了数据类型的限制外，与MySQL数据库相比，还存在一些限制，在实现复制之前应先解决这些限制：

- MySQL中的每个表都应该包含PRIMARY KEY
- 对于包含ENUM字段值超出范围（在ENUM签名中指定）的行的表，复制将不起作用。

DDL查询

MySQL DDL查询转换为相应的ClickHouse DDL查询(ALTER, CREATE, DROP, RENAME)。如果ClickHouse无法解析某个DDL查询，则该查询将被忽略。

Data Replication

MaterializeMySQL不支持直接INSERT, DELETE和UPDATE查询。但是，它们是在数据复制方面支持的：

- MySQL的INSERT查询转换为INSERT并携带_sign=1.
- MySQL的DELETE查询转换为INSERT并携带_sign=-1.
- MySQL的UPDATE查询转换为INSERT并携带_sign=-1, INSERT和_sign=1.

查询MaterializeMySQL表

SELECT查询MaterializeMySQL表有一些细节：

- 如果_version在SELECT中没有指定，则使用FINAL修饰符。所以只有带有MAX(_version)的行才会被选中。
- 如果_sign在SELECT中没有指定，则默认使用WHERE _sign=1。因此，删除的行不会包含在结果集中。
- 结果包括列中的列注释，因为它们存在于SQL数据库表中。

Index Conversion

MySQL的PRIMARY KEY和INDEX子句在ClickHouse表中转换为ORDER BY元组。

ClickHouse只有一个物理顺序，由ORDER BY子句决定。要创建一个新的物理顺序，使用materialized views。

Notes

- 带有_sign=-1的行不会从表中物理删除。
- MaterializeMySQL引擎不支持级联UPDATE/DELETE查询。
- 复制很容易被破坏。

- 禁止对数据库和表进行手工操作。
- MaterializeMySQL受`optimize_on_insert`设置的影响。当MySQL服务器中的表发生变化时，数据会合并到MaterializeMySQL数据库中相应的表中。

使用示例

MySQL操作:

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

a	b	c
2	222	Wow!

ClickHouse中的数据库，与MySQL服务器交换数据:

创建的数据库和表:

```
CREATE DATABASE mysql ENGINE = MaterializeMySQL('localhost:3306', 'db', 'user', '***');
SHOW TABLES FROM mysql;
```

name
test

然后插入数据:

```
SELECT * FROM mysql.test;
```

a	b
1	11
2	22

删除数据后，添加列并更新:

```
SELECT * FROM mysql.test;
```

a	b	c
2	222	Wow!

[experimental] MaterializedPostgreSQL

使用PostgreSQL数据库表的初始数据转储创建ClickHouse数据库，并启动复制过程，即执行后台作业，以便在远程PostgreSQL数据库中的PostgreSQL数据库表上发生新更改时应用这些更改。

ClickHouse服务器作为PostgreSQL副本工作。它读取WAL并执行DML查询。DDL不是复制的，但可以处理（如下所述）。

创建数据库

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

Engine参数

- host:port — PostgreSQL服务地址
- database — PostgreSQL数据库名
- user — PostgreSQL用户名
- password — 用户密码

设置

- materialized_postgresql_max_block_size
- materialized_postgresql_tables_list
- materialized_postgresql_allow_automatic_update

```
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;
```

必备条件

- 在postgresql配置文件中将wal_level设置为logical，将max_replication_slots设置为2。
- 每个复制表必须具有以下一个replica identity:
 1. default (主键)
 2. 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;
```

总是先检查主键。如果不存在，则检查索引（定义为副本标识索引）。

如果使用index作为副本标识，则表中必须只有一个这样的索引。

你可以用下面的命令来检查一个特定的表使用了什么类型：

```
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;
```

注意

1. **TOAST**不支持值转换。将使用数据类型的默认值。

使用示例

```
CREATE DATABASE postgresql_db
ENGINE = MaterializedPostgreSQL('postgres1:5432', 'postgres_database', 'postgres_user', 'postgres_password');

SELECT * FROM postgresql_db.postgres_table;
```

MySQL

MySQL引擎用于将远程的MySQL服务器中的表映射到ClickHouse中，并允许您对表进行INSERT和SELECT查询，以方便您在ClickHouse与MySQL之间进行数据交换

MySQL数据库引擎会将对其的查询转换为MySQL语法并发送到MySQL服务器中，因此您可以执行诸如SHOW TABLES或SHOW CREATE TABLE之类的操作。

但您无法对其执行以下操作：

- RENAME
- CREATE TABLE
- ALTER

创建数据库

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

引擎参数

- host:port — MySQL服务地址
- database — MySQL数据库名称
- user — MySQL用户名
- password — MySQL用户密码

支持的数据类型

MySQL

ClickHouse

UNSIGNED TINYINT

UInt8

MySQL	ClickHouse
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	Date
DATETIME, TIMESTAMP	DateTime
BINARY	FixedString

其他的MySQL数据类型将全部都转换为 [String](#).

[Nullable](#) 已经支持

全局变量支持

为了更好地兼容，您可以在SQL样式中设置全局变量，如 `@@identifier`.

支持这些变量：

- `version`
- `max_allowed_packet`

警告

到目前为止，这些变量是存根，并且不对应任何内容。

示例：

```
SELECT @@version;
```

使用示例

MySQL操作:

```
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中的数据库，与MySQL服务器交换数据：

```
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

Lazy

在最后一次访问之后，只在RAM中保存`expiration_time_in_seconds`秒。只能用于*Log表。

它是为存储许多小的*Log表而优化的，对于这些表，访问之间有很长的时间间隔。

创建数据库

```
CREATE DATABASE testlazy ENGINE = Lazy(expiration_time_in_seconds);
```

Atomic

它支持非阻塞的DROP TABLE和RENAME TABLE查询和原子的EXCHANGE TABLES t1 AND t2查询。默认情况下使用Atomic数据库引擎。

创建数据库

```
CREATE DATABASE test[ ENGINE = Atomic];
```

使用方式

Table UUID

数据库Atomic中的所有表都有唯一的UUID，并将数据存储在目录`/clickhouse_path/store/xxx/xxxxyyyy-yyyy-yyyy-yyyy-yyyyyyyyyy/`，其中`xxxxyyyy-yyyy-yyyy-yyyyyyyyyy`是该表的UUID。

通常，UUID是自动生成的，但用户也可以在创建表时以相同的方式显式指定UUID(不建议这样做)。可以使用`show_table_uuid_in_table_create_query_if_not_nil`设置。显示UUID的使用SHOW CREATE查询。例如：

```
CREATE TABLE name UUID '28f1c61c-2970-457a-bffe-454156ddcfef' (n UInt64) ENGINE = ...;
```

RENAME TABLES

RENAME查询是在不更改UUID和移动表数据的情况下执行的。这些查询不会等待使用表的查询完成，而是会立即执行。

DROP/DETACH TABLES

在DROP TABLE上，不删除任何数据，数据库Atomic只是通过将元数据移动到`/clickhouse_path/metadata_dropped/`将表标记为已删除，并通知后台线程。最终表数据删除前的延迟由`database_atomic_delay_before_drop_table_sec`设置指定。

可以使用SYNC修饰符指定同步模式。使用`database_atomic_wait_for_drop_and_detach_synchronously`设置执行此操作。在本例中，DROP等待运行`SELECT, INSERT`和其他使用表完成的查询。表在不使用时将被实际删除。

EXCHANGE TABLES

EXCHANGE以原子方式交换表。因此，不是这种非原子操作：

```
RENAME TABLE new_table TO tmp, old_table TO new_table, tmp TO old_table;
```

可以使用一个原子查询：

```
EXCHANGE TABLES new_table AND old_table;
```

ReplicatedMergeTree in Atomic Database

对于ReplicatedMergeTree表，建议不要在ZooKeeper和副本名称中指定engine-path的参数。在这种情况下，将使用配置的参数default_replica_path和default_replica_name。如果要显式指定引擎的参数，建议使用{uuid}宏。这是非常有用的，以便为ZooKeeper中的每个表自动生成唯一的路径。

另请参考

- [system.databases](#) 系统表

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;
```

name
table1
table2

Shows the tables:

```
SELECT * FROM sqlite_db.table1;
```

col1	col2
line1	1
line2	2
line3	3

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

允许连接到远程PostgreSQL服务。支持读写操作(SELECT和INSERT查询)，以在ClickHouse和PostgreSQL之间交换数据。

在SHOW TABLES和DESCRIBE TABLE查询的帮助下，从远程PostgreSQL实时访问表列表和表结构。

支持表结构修改(ALTER TABLE ... ADD|DROP COLUMN)。如果use_table_cache参数(参见下面的引擎参数)设置为1，则会缓存表结构，不会检查是否被修改，但可以用DETACH和ATTACH查询进行更新。

创建数据库

```
CREATE DATABASE test_database
ENGINE = PostgreSQL('host:port', 'database', 'user', 'password'[, `use_table_cache`]);
```

引擎参数

- host:port — PostgreSQL服务地址
- database — 远程数据库名次
- user — PostgreSQL用户名
- password — PostgreSQL用户密码
- use_table_cache — 定义数据库表结构是否已缓存或不进行。可选的。默认值：0.

支持的数据类型

PostgreSQL	ClickHouse
DATE	Date
TIMESTAMP	DateTime
REAL	Float32
DOUBLE	Float64
DECIMAL, NUMERIC	Decimal
SMALLINT	Int16
INTEGER	Int32
BIGINT	Int64
SERIAL	UInt32
BIGSERIAL	UInt64
TEXT, CHAR	String
INTEGER	Nullable(Int32)
ARRAY	Array

使用示例

ClickHouse 中的数据库，与 PostgreSQL 服务器交换数据：

```
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 |
```

从PostgreSQL表中读取数据：

```
SELECT * FROM test_database.test_table;
```

id	value
1	2

将数据写入PostgreSQL表：

```
INSERT INTO test_database.test_table VALUES (3,4);
SELECT * FROM test_database.test_table;
```

int	id	value
1	2	
3	4	

在PostgreSQL中修改了表结构：

```
postgres> ALTER TABLE test_table ADD COLUMN data Text
```

当创建数据库时，参数use_table_cache被设置为1，ClickHouse中的表结构被缓存，因此没有被修改：

```
DESCRIBE TABLE test_database.test_table;
```

name	type
id	Nullable(Integer)
value	Nullable(Integer)

分离表并再次附加它之后，结构被更新了：

```
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

该引擎基于Atomic引擎。它支持通过将DDL日志写入ZooKeeper并在给定数据库的所有副本上执行的元数据复制。

一个ClickHouse服务器可以同时运行和更新多个复制的数据库。但是同一个复制的数据库不能有多个副本。

创建数据库

```
CREATE DATABASE testdb ENGINE = Replicated('zoo_path', 'shard_name', 'replica_name') [SETTINGS ...]
```

引擎参数

- `zoo_path` — ZooKeeper地址，同一个ZooKeeper路径对应同一个数据库。
- `shard_name` — 分片的名字。数据库副本按`shard_name`分组到分片中。
- `replica_name` — 副本的名字。同一分片的所有副本的副本名称必须不同。

警告

对于ReplicatedMergeTree表，如果没有提供参数，则使用默认参数`:clickhouse{uuid}/{shard}`和`{replica}`。这些可以在服务器设置`default_replica_path`和`default_replica_name`中更改。宏`{uuid}`被展开到表的`uuid`，`{shard}`和`{replica}`被展开到服务器配置的值，而不是数据库引擎参数。但是在将来，可以使用Replicated数据库的`shard_name`和`replica_name`。

使用方式

使用Replicated数据库的DDL查询的工作方式类似于ON CLUSTER查询，但有细微差异。

首先，DDL请求尝试在启动器(最初从用户接收请求的主机)上执行。如果请求没有完成，那么用户立即收到一个错误，其他主机不会尝试完成它。如果在启动器上成功地完成了请求，那么所有其他主机将自动重试，直到完成请求。启动器将尝试在其他主机上等待查询完成(不超过distributed_ddl_task_timeout)，并返回一个包含每个主机上查询执行状态的表。

错误情况下的行为是由distributed_ddl_output_mode设置调节的，对于Replicated数据库，最好将其设置为`null_status_on_timeout` - 例如，如果一些主机没有时间执行distributed_ddl_task_timeout的请求，那么不要抛出异常，但在表中显示它们的NULL状态。

`system.clusters`系统表包含一个名为复制数据库的集群，它包含数据库的所有副本。当创建/删除副本时，这个集群会自动更新，它可以用于Distributed表。

当创建数据库的新副本时，该副本会自己创建表。如果副本已经不可用很长一段时间，并且已经滞后于复制日志-它用ZooKeeper中的当前元数据检查它的本地元数据，将带有数据的额外表移动到一个单独的非复制数据库(以免意外地删除任何多余的东西)，创建缺失的表，如果表名已经被重命名，则更新表名。数据在ReplicatedMergeTree级别被复制，也就是说，如果表没有被复制，数据将不会被复制(数据库只负责元数据)。

使用示例

创建三台主机的集群：

```
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}');
```

运行DDL：

```
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	

显示系统表:

```
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

创建分布式表并插入数据:

```
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]

向一台主机添加副本:

```
node4 :) CREATE DATABASE r ENGINE=Replicated('some/path/r','other_shard','r2');
```

集群配置如下所示:

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

分布式表也将从新主机获取数据:

```
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]

VersionedCollapsingMergeTree

这个引擎:

- 允许快速写入不断变化的对象状态。
- 删除后台中的旧对象状态。这显着降低了存储体积。

请参阅部分 [崩溃](#) 有关详细信息。

引擎继承自 **MergeTree** 并将折叠行的逻辑添加到合并数据部分的算法中。**VersionedCollapsingMergeTree** 用于相同的目录的 **折叠树** 但使用不同的折叠算法，允许以多个线程的任何顺序插入数据。特别是，**Version** 列有助于正确折叠行，即使它们以错误的顺序插入。相比之下，**CollapsingMergeTree** 只允许严格连续插入。

创建表

```
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, ...]
```

有关查询参数的说明，请参阅 [查询说明](#)。

引擎参数

VersionedCollapsingMergeTree(sign, version)

- **sign** — 指定行类型的列名：`1` 是一个“state”行，`-1` 是一个“cancel”行
列数据类型应为 `Int8`。

- **version** — 指定对象状态版本的列名。

列数据类型应为 `UInt*`。

查询 Clauses

当创建一个 **VersionedCollapsingMergeTree** 表时，跟创建一个 **MergeTree** 表的时候需要相同 **Clause**

► 不推荐使用的创建表的方法

折叠

数据

考虑一种情况，您需要为某个对象保存不断变化的数据。对于一个对象有一行，并在发生更改时更新该行是合理的。但是，对于数据库管理系统来说，更新操作非常昂贵且速度很慢，因为它需要重写存储中的数据。如果需要快速写入数据，则不能接受更新，但可以按如下顺序将更改写入对象。

使用 **Sign** 列写入行时。如果 **Sign = 1** 这意味着该行是一个对象的状态（让我们把它称为“state”行）。如果 **Sign = -1** 它指示具有相同属性的对象的状态的取消（让我们称之为“cancel”行）。还可以使用 **Version** 列，它应该用单独的数字标识对象的每个状态。

例如，我们要计算用户在某个网站上访问了多少页面以及他们在那里的时间。在某个时间点，我们用用户活动的状态写下面的行：

UserID	PageViews	Duration	Sign	Version
4324182021466249494	5	146	1	1

在稍后的某个时候，我们注册用户活动的变化，并用以下两行写入它。

UserID	PageViews	Duration	Sign	Version
4324182021466249494	5	146	-1	1
4324182021466249494	6	185	1	2

第一行取消对象（用户）的先前状态。它应该复制已取消状态的所有字段，除了 `Sign`.

第二行包含当前状态。

因为我们只需要用户活动的最后一个状态，行

UserID	PageViews	Duration	Sign	Version
4324182021466249494	5	146	1	1
4324182021466249494	5	146	-1	1

可以删除，折叠对象的无效（回）状态。`VersionedCollapsingMergeTree` 在合并数据部分时执行此操作。

要了解为什么每次更改都需要两行，请参阅 [算法](#).

使用注意事项

- 写入数据的程序应该记住对象的状态以取消它。该“cancel”字符串应该是“state”与相反的字符串 `Sign`. 这增加了存储的初始大小，但允许快速写入数据。
- 列中长时间增长的数组由于写入负载而降低了引擎的效率。数据越简单，效率就越高。
- `SELECT` 结果很大程度上取决于对象变化历史的一致性。准备插入数据时要准确。不一致的数据将导致不可预测的结果，例如会话深度等非负指标的负值。

算法

当 `ClickHouse` 合并数据部分时，它会删除具有相同主键和版本但 `Sign` 值不同的一对行. 行的顺序并不重要。

当 `ClickHouse` 插入数据时，它会按主键对行进行排序。如果 `Version` 列不在主键中，`ClickHouse` 将其隐式添加到主键作为最后一个字段并使用它进行排序。

选择数据

`ClickHouse` 不保证具有相同主键的所有行都将位于相同的结果数据部分中，甚至位于相同的物理服务器上。对于写入数据和随后合并数据部分都是如此。此外，`ClickHouse` 流程 `SELECT` 具有多个线程的查询，并且无法预测结果中的行顺序。这意味着，如果有必要从 `VersionedCollapsingMergeTree` 表中得到完全“collapsed”的数据，聚合是必需的。

要完成折叠，请使用 `GROUP BY` 考虑符号的子句和聚合函数。例如，要计算数量，请使用 `sum(Sign)` 而不是 `count()`. 要计算的东西的总和，使用 `sum(Sign * x)` 而不是 `sum(x)`，并添加 `HAVING sum(Sign) > 0`.

聚合 `count`, `sum` 和 `avg` 可以这样计算。聚合 `uniq` 如果对象至少具有一个非折叠状态，则可以计算。聚合 `min` 和 `max` 无法计算是因为 `VersionedCollapsingMergeTree` 不保存折叠状态值的历史记录。

如果您需要提取数据“collapsing”但是，如果没有聚合（例如，要检查是否存在其最新值与某些条件匹配的行），则可以使用 `FINAL` 修饰 `FROM` 条件这种方法效率低下，不应与大型表一起使用。

使用示例

示例数据：

UserID	PageViews	Duration	Sign	Version
4324182021466249494	5	146	1	1
4324182021466249494	5	146	-1	1
4324182021466249494	6	185	1	2

创建表:

```
CREATE TABLE UAct
(
    UserID UInt64,
    PageViews UInt8,
    Duration UInt8,
    Sign Int8,
    Version UInt8
)
ENGINE = VersionedCollapsingMergeTree(Sign, Version)
ORDER BY UserID
```

插入数据:

```
INSERT INTO UAct VALUES (4324182021466249494, 5, 146, 1, 1)
```

```
INSERT INTO UAct VALUES (4324182021466249494, 5, 146, -1, 1),(4324182021466249494, 6, 185, 1, 2)
```

我们用两个 `INSERT` 查询以创建两个不同的数据部分。如果我们使用单个查询插入数据，ClickHouse将创建一个数据部分，并且永远不会执行任何合并。

获取数据:

```
SELECT * FROM UAct
```

UserID	PageViews	Duration	Sign	Version
4324182021466249494	5	146	1	1
4324182021466249494	5	146	-1	1
4324182021466249494	6	185	1	2

我们在这里看到了什么，折叠的部分在哪里？

我们使用两个创建了两个数据部分 `INSERT` 查询。该 `SELECT` 查询是在两个线程中执行的，结果是行的随机顺序。由于数据部分尚未合并，因此未发生折叠。ClickHouse在我们无法预测的未知时间点合并数据部分。

这就是为什么我们需要聚合:

```
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

如果我们不需要聚合，并希望强制折叠，我们可以使用 `FINAL` 修饰符 `FROM` 条款

```
SELECT * FROM UAct FINAL
```

UserID	PageViews	Duration	Sign	Version
4324182021466249494	6	185	1	2

这是一个非常低效的方式来选择数据。不要把它用于数据量大的表。

GraphiteMergeTree

该引擎用来对 **Graphite** 数据进行瘦身及汇总。对于想使用CH来存储Graphite数据的开发者来说可能有用。

如果不需要对Graphite数据做汇总，那么可以使用任意的CH表引擎；但若需要，那就采用 **GraphiteMergeTree** 引擎。它能减少存储空间，同时能提高Graphite数据的查询效率。

该引擎继承自 **MergeTree**.

创建表

```
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, ...]
```

建表语句的详细说明请参见 [创建表](#)

含有Graphite数据集的表应该包含以下的数据列：

- 指标名称(Graphite sensor)，数据类型：String
- 指标的时间度量，数据类型：DateTime
- 指标的值，数据类型：任意数值类型
- 指标的版本号，数据类型：任意数值类型

CH以最大的版本号保存行记录，若版本号相同，保留最后写入的数据。

以上列必须设置在汇总参数配置中。

GraphiteMergeTree 参数

- config_section - 配置文件中标识汇总规则的节点名称

建表语句

在创建 GraphiteMergeTree 表时，需要采用和 [clauses](#) 相同的语句，就像创建 MergeTree 一样。

► 已废弃的建表语句

汇总配置的参数

汇总的配置参数由服务器配置的 [graphite_rollup](#) 参数定义。参数名称可以是任意的。允许为多个不同表创建多组配置并使用。

汇总配置的结构如下：

所需的列

模式Patterns

所需的列

- `path_column_name` — 保存指标名称的列名 (Graphite sensor). 默认值: `Path`.
- `time_column_name` — 保存指标时间度量的列名. 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`.默认值: `Value`.
- `version_column_name` - 保存指标的版本号列. 默认值: `Timestamp`.

模式Patterns

`patterns` 的结构：

```
pattern
  regexp
  function
pattern
  regexp
  age + precision
...
pattern
  regexp
  function
  age + precision
...
pattern
...
default
  function
  age + precision
...
```

Attention

模式必须严格按顺序配置：

1. 不含`function` or `retention`的Patterns
1. 同时含有`function` and `retention`的Patterns
1. `default`的Patterns.

CH在处理行记录时，会检查 `pattern` 节点的规则。每个 `pattern` (含`default`) 节点可以包含 `function` 用于聚合操作，或`retention`参数，或者两者都有。如果指标名称和 `regexp`相匹配，相应 `pattern`的规则会生效；否则，使用 `default` 节点的规则。

`pattern` 和 `default` 节点的字段设置：

- `regexp`- 指标名的pattern.
- `age` - 数据的最小存活时间(按秒算).
- `precision`- 按秒来衡量数据存活时间时的精确程度. 必须能被86400整除 (一天的秒数).
- `function` - 对于存活时间在 `[age, age + precision]`之内的数据，需要使用的聚合函数

配置示例

```
<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>
```

AggregatingMergeTree

该引擎继承自 [MergeTree](#)，并改变了数据片段的合并逻辑。ClickHouse 会将一个数据片段内所有具有相同主键（准确的说是 [排序键](#)）的行替换成一行，这一行会存储一系列聚合函数的状态。

可以使用 [AggregatingMergeTree](#) 表来做增量数据的聚合统计，包括物化视图的数据聚合。

引擎使用以下类型来处理所有列：

- [AggregateFunction](#)
- [SimpleAggregateFunction](#)

[AggregatingMergeTree](#) 适用于能够按照一定的规则缩减行数的情况。

建表

```
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, ...]
```

语句参数的说明，请参阅 [建表语句描述](#)。

子句

创建 [AggregatingMergeTree](#) 表时，需用跟创建 [MergeTree](#) 表一样的 [子句](#)。

SELECT 和 INSERT

要插入数据，需使用带有 -State- 聚合函数的 **INSERT SELECT** 语句。

从 `AggregatingMergeTree` 表中查询数据时，需使用 `GROUP BY` 子句并且要使用与插入时相同的聚合函数，但后缀要改为 -`Merge`。

对于 `SELECT` 查询的结果，`AggregateFunction` 类型的值对 ClickHouse 的所有输出格式都实现了特定的二进制表示法。在进行数据转储时，例如使用 `TabSeparated` 格式进行 `SELECT` 查询，那么这些转储数据也能直接用 `INSERT` 语句导回。

聚合物化视图的示例

创建一个跟踪 `test.visits` 表的 `AggregatingMergeTree` 物化视图：

```
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;
```

向 `test.visits` 表中插入数据。

```
INSERT INTO test.visits ...
```

数据会同时插入到表和视图中，并且视图 `test.basic` 会将里面的数据聚合。

要获取聚合数据，我们需要在 `test.basic` 视图上执行类似 `SELECT ... GROUP BY ...` 这样的查询：

```
SELECT
    StartDate,
    sumMerge(Visits) AS Visits,
    uniqMerge(Users) AS Users
FROM test.basic
GROUP BY StartDate
ORDER BY StartDate;
```

CollapsingMergeTree

该引擎继承于 `MergeTree`，并在数据块合并算法中添加了折叠行的逻辑。

`CollapsingMergeTree` 会异步的删除（折叠）这些除了特定列 `Sign` 有 `1` 和 `-1` 的值以外，其余所有字段的值都相等的成对的行。没有成对的行会被保留。更多的细节请看本文的 [折叠](#) 部分。

因此，该引擎可以显著的降低存储量并提高 `SELECT` 查询效率。

建表

```

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, ...]

```

请求参数的描述，参考[请求参数](#)。

CollapsingMergeTree 参数

- sign — 类型列的名称：1是«状态»行，-1是«取消»行。

列数据类型 — Int8。

子句

创建 CollapsingMergeTree 表时，需要与创建 MergeTree 表时相同的[子句](#)。

▶ 已弃用的建表方法

折叠

数据

考虑你需要为某个对象保存不断变化的数据的情景。似乎为一个对象保存一行记录并在其发生任何变化时更新记录是合乎逻辑的，但是更新操作对 DBMS 来说是昂贵且缓慢的，因为它需要重写存储中的数据。如果你需要快速的写入数据，则更新操作是不可接受的，但是你可以按下面的描述顺序地更新一个对象的变化。

在写入行的时候使用特定的列 Sign。如果 Sign = 1 则表示这一行是对象的状态，我们称之为«状态»行。如果 Sign = -1 则表示是对具有相同属性的状态行的取消，我们称之为«取消»行。

例如，我们想要计算用户在某个站点访问的页面页面数以及他们在那停留的时间。在某个时候，我们将用户的活动状态写入下面这样的行。

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	1

一段时间后，我们写入下面的两行来记录用户活动的变化。

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	-1
4324182021466249494	6	185	1

第一行取消了这个对象（用户）的状态。它需要复制被取消的状态行的所有除了 Sign 的属性。

第二行包含了当前的状态。

因为我们只需要用户活动的最后状态，这些行

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	1
4324182021466249494	5	146	-1

可以在折叠对象的失效（老的）状态的时候被删除。`CollapsingMergeTree` 会在合并数据片段的时候做这件事。

为什么我们每次改变需要 2 行可以阅读 **算法** 段。

这种方法的特殊属性

- 写入的程序应该记住对象的状态从而可以取消它。«取消»字符串应该是«状态»字符串的复制，除了相反的 `Sign`。它增加了存储的初始数据的大小，但使得写入数据更快速。
- 由于写入的负载，列中长的增长阵列会降低引擎的效率。数据越简单，效率越高。
- `SELECT` 的结果很大程度取决于对象变更历史的一致性。在准备插入数据时要准确。在不一致的数据中会得到不可预料的结果，例如，像会话深度这种非负指标的负值。

算法

当 `ClickHouse` 合并数据片段时，每组具有相同主键的连续行被减少到不超过两行，一行 `Sign = 1`（«状态»行），另一行 `Sign = -1`（«取消»行），换句话说，数据项被折叠了。

对每个结果的数据部分 `ClickHouse` 保存：

- 第一个«取消»和最后一个«状态»行，如果«状态»和«取消»行的数量匹配且最后一个行是«状态»行
- 最后一个«状态»行，如果«状态»行比«取消»行多一个或一个以上。
- 第一个«取消»行，如果«取消»行比«状态»行多一个或一个以上。
- 没有行，在其他所有情况下。

合并会继续，但是 `ClickHouse` 会把此情况视为逻辑错误并将其记录在服务日志中。这个错误会在相同的数据被插入超过一次时出现。

因此，折叠不应该改变统计数据的结果。

变化逐渐地被折叠，因此最终几乎每个对象都只剩下了最后的状态。

`Sign` 是必须的因为合并算法不保证所有有相同主键的行都会在同一个结果数据片段中，甚至是在同一台物理服务器上。`ClickHouse` 用多线程来处理 `SELECT` 请求，所以它不能预测结果中行的顺序。如果要从 `CollapsingMergeTree` 表中获取完全«折叠»后的数据，则需要聚合。

要完成折叠，请使用 `GROUP BY` 子句和用于处理符号的聚合函数编写请求。例如，要计算数量，使用 `sum(Sign)` 而不是 `count()`。要计算某物的总和，使用 `sum(Sign * x)` 而不是 `sum(x)`，并添加 `HAVING sum(Sign) > 0` 子句。

聚合体 `count,sum` 和 `avg` 可以用这种方式计算。如果一个对象至少有一个未被折叠的状态，则可以计算 `uniq` 聚合。`min` 和 `max` 聚合无法计算，因为 `CollapsingMergeTree` 不会保存折叠状态的值的历史记录。

如果你需要在不进行聚合的情况下获取数据（例如，要检查是否存在最新值与特定条件匹配的行），你可以在 `FROM` 从句中使用 `FINAL` 修饰符。这种方法显然是更低效的。

示例

示例数据：

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	1
4324182021466249494	5	146	-1
4324182021466249494	6	185	1

建表：

```
CREATE TABLE UAct
(
    UserID UInt64,
    PageViews UInt8,
    Duration UInt8,
    Sign Int8
)
ENGINE = CollapsingMergeTree(Sign)
ORDER BY UserID
```

插入数据：

```
INSERT INTO UAct VALUES (4324182021466249494, 5, 146, 1)
```

```
INSERT INTO UAct VALUES (4324182021466249494, 5, 146, -1), (4324182021466249494, 6, 185, 1)
```

我们使用两次 `INSERT` 请求来创建两个不同的数据片段。如果我们使用一个请求插入数据，ClickHouse 只会创建一个数据片段且不会执行任何合并操作。

获取数据：

```
SELECT * FROM UAct
```

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	-1
4324182021466249494	6	185	1

UserID	PageViews	Duration	Sign
4324182021466249494	5	146	1

我们看到了什么，哪里有折叠？

通过两个 `INSERT` 请求，我们创建了两个数据片段。`SELECT` 请求在两个线程中被执行，我们得到了随机顺序的行。没有发生折叠是因为还没有合并数据片段。ClickHouse 在一个我们无法预料的未知时刻合并数据片段。

因此我们需要聚合：

```
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

如果我们不需要聚合并想要强制进行折叠，我们可以在 `FROM` 从句中使用 `FINAL` 修饰语。

```
SELECT * FROM UAct FINAL
```

UserID	PageViews	Duration	Sign
4324182021466249494	6	185	1

这种查询数据的方法是非常低效的。不要在大表中使用它。

MergeTree

Clickhouse 中最强大的表引擎当属 **MergeTree** (合并树) 引擎及该系列 (***MergeTree**) 中的其他引擎。

MergeTree 系列的引擎被设计用于插入极大量的数据到一张表当中。数据可以以数据片段的形式一个接着一个的快速写入，数据片段在后台按照一定的规则进行合并。相比在插入时不断修改 (重写) 已存储的数据，这种策略会高效很多。

主要特点：

- 存储的数据按主键排序。

这使得您能够创建一个小型的稀疏索引来加快数据检索。

- 如果指定了 **分区键** 的话，可以使用分区。

在相同数据集和相同结果集的情况下 ClickHouse 中某些带分区的操作会比普通操作更快。查询中指定了分区键时 ClickHouse 会自动截取分区数据。这也有效增加了查询性能。

- 支持数据副本。

ReplicatedMergeTree 系列的表提供了数据副本功能。更多信息，请参阅 [数据副本](#) 一节。

- 支持数据采样。

需要的话，您可以给表设置一个采样方法。

注意

合并 引擎并不属于 ***MergeTree** 系列。

建表

```
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()
ORDER BY expr
[PARTITION BY expr]
[PRIMARY KEY expr]
[SAMPLE BY expr]
[TTL expr [DELETE|TO DISK 'xxx'|TO VOLUME 'xxx'], ...]
[SETTINGS name=value, ...]
```

对于以上参数的描述，可参考 [CREATE 语句 的描述](#)。

子句

- **ENGINE** - 引擎名和参数。`ENGINE = MergeTree()`. **MergeTree** 引擎没有参数。

- **ORDER BY** — 排序键。

可以是一组列的元组或任意的表达式。例如: `ORDER BY (CounterID, EventDate)`。

如果没有使用 `PRIMARY KEY` 显式指定的主键, ClickHouse 会使用排序键作为主键。

如果不需要排序, 可以使用 `ORDER BY tuple()`. 参考 [选择主键](#)

- **PARTITION BY** — 分区键 , 可选项。

要按月分区, 可以使用表达式 `toYYYYMM(date_column)` , 这里的 `date_column` 是一个 `Date` 类型的列。分区名的格式会是 "YYYYMM"。

- **PRIMARY KEY** - 如果要 [选择与排序键不同的主键](#), 在这里指定, 可选项。

默认情况下主键跟排序键 (由 `ORDER BY` 子句指定) 相同。

因此, 大部分情况下不需要再专门指定一个 `PRIMARY KEY` 子句。

- **SAMPLE BY** - 用于抽样的表达式, 可选项。

如果要用抽样表达式, 主键中必须包含这个表达式。例如:

`SAMPLE BY intHash32(UserID) ORDER BY (CounterID, EventDate, intHash32(UserID))`。

- **TTL** - 指定行存储的持续时间并定义数据片段在硬盘和卷上的移动逻辑的规则列表, 可选项。

表达式中必须存在至少一个 `Date` 或 `DateTime` 类型的列, 比如:

`TTL date + INTERVAL 1 DAY`

规则的类型 `DELETE|TO DISK 'xxx'|TO VOLUME 'xxx'` 指定了当满足条件 (到达指定时间) 时所要执行的动作: 移除过期的行, 还是将数据片段 (如果数据片段中的所有行都满足表达式的话) 移动到指定的磁盘 (`TO DISK 'xxx'`) 或卷 (`TO VOLUME 'xxx'`)。默认的规则是移除 (`DELETE`)。可以在列表中指定多个规则, 但最多只能有一个`DELETE`的规则。

更多细节, 请查看 [表和列的 TTL](#)

- **SETTINGS** — 控制 MergeTree 行为的额外参数, 可选项:

- `index_granularity` — 索引粒度。索引中相邻的『标记』间的数据行数。默认值8192。参考[数据存储](#)。

- `index_granularity_bytes` — 索引粒度, 以字节为单位, 默认值: 10Mb。如果想要仅按数据行数限制索引粒度, 请设置为 0(不建议)。

- `min_index_granularity_bytes` - 允许的最小数据粒度, 默认值: 1024b。该选项用于防止误操作, 添加了一个非常低索引粒度的表。参考[数据存储](#)

- `enable_mixed_granularity_parts` — 是否启用通过 `index_granularity_bytes` 控制索引粒度的大小。在19.11版本之前, 只有 `index_granularity` 配置能够用于限制索引粒度的大小。当从具有很大的行 (几十上百兆字节) 的表中查询数据时候, `index_granularity_bytes` 配置能够提升ClickHouse的性能。如果您的表里有很大的行, 可以开启这项配置来提升SELECT 查询的性能。

- `use_minimalistic_part_header_in_zookeeper` — ZooKeeper中数据片段存储方式。如果`use_minimalistic_part_header_in_zookeeper=1` , ZooKeeper 会存储更少的数据。更多信息参考[服务配置参数](#)() 这章中的 [设置描述](#)。

- `min_merge_bytes_to_use_direct_io` — 使用直接 I/O 来操作磁盘的合并操作时要求的最小数据量。合并数据片段时, ClickHouse 会计算要被合并的所有数据的总存储空间。如果大小超过了 `min_merge_bytes_to_use_direct_io` 设置的字节数, 则 ClickHouse 将使用直接 I/O 接口 (`O_DIRECT` 选项) 对磁盘读写。如果设置 `min_merge_bytes_to_use_direct_io = 0` , 则会禁用直接 I/O。默认值: `10 * 1024 * 1024 * 1024` 字节。

- `merge_with_ttl_timeout` — TTL合并频率的最小间隔时间, 单位: 秒。默认值: 86400 (1 天)。

- `write_final_mark` — 是否启用在数据片段尾部写入最终索引标记。默认值: 1 (不要关闭)。

- `merge_max_block_size` — 在块中进行合并操作时的最大行数限制。默认值：8192
- `storage_policy` — 存储策略。参见 [使用具有多个块的设备进行数据存储](#)。
- `min_bytes_for_wide_part,min_rows_for_wide_part` 在数据片段中可以使用 Wide 格式进行存储的最小字节数/行数。您可以不设置、只设置一个，或全都设置。参考：[数据存储](#)
- `max_parts_in_total` - 所有分区中最大块的数量(意义不明)
- `max_compress_block_size` - 在数据压缩写入表前，未压缩数据块的最大大小。您可以在全局设置中设置该值(参见[max_compress_block_size](#))。建表时指定该值会覆盖全局设置。
- `min_compress_block_size` - 在数据压缩写入表前，未压缩数据块的最小大小。您可以在全局设置中设置该值(参见[min_compress_block_size](#))。建表时指定该值会覆盖全局设置。
- `max_partitions_to_read` - 一次查询中可访问的分区最大数。您可以在全局设置中设置该值(参见[max_partitions_to_read](#))。

示例配置

```
ENGINE MergeTree() PARTITION BY toYYYYMM(EventDate) ORDER BY (CounterID, EventDate, intHash32(UserID))
SAMPLE BY intHash32(UserID) SETTINGS index_granularity=8192
```

在这个例子中，我们设置了按月进行分区。

同时我们设置了一个按用户 ID 哈希的抽样表达式。这使得您可以对该表中每个 `CounterID` 和 `EventDate` 的数据伪随机分布。如果您在查询时指定了 `SAMPLE` 子句。ClickHouse 会返回对于用户子集的一个均匀的伪随机数据采样。

`index_granularity` 可省略因为 8192 是默认设置。

▶ 已弃用的建表方法

数据存储

表由按主键排序的数据片段 (DATA PART) 组成。

当数据被插入到表中时，会创建多个数据片段并按主键的字典序排序。例如，主键是 (`CounterID, Date`) 时，片段中数据首先按 `CounterID` 排序，具有相同 `CounterID` 的部分按 `Date` 排序。

不同分区的数据会被分成不同的片段，ClickHouse 在后台合并数据片段以便更高效存储。不同分区的数据片段不会进行合并。合并机制并不保证具有相同主键的行全都合并到同一个数据片段中。

数据片段可以以 `Wide` 或 `Compact` 格式存储。在 `Wide` 格式下，每一列都会在文件系统中存储为单独的文件，在 `Compact` 格式下所有列都存储在一个文件中。`Compact` 格式可以提高插入量少插入频率频繁时的性能。

数据存储格式由 `min_bytes_for_wide_part` 和 `min_rows_for_wide_part` 表引擎参数控制。如果数据片段中的字节数或行数少于相应的设置值，数据片段会以 `Compact` 格式存储，否则会以 `Wide` 格式存储。

每个数据片段被逻辑的分割成颗粒 (granules)。颗粒是 ClickHouse 中进行数据查询时的最小不可分割数据集。ClickHouse 不会对行或值进行拆分，所以每个颗粒总是包含整数个行。每个颗粒的第一行通过该行的主键值进行标记，ClickHouse 会为每个数据片段创建一个索引文件来存储这些标记。对于每列，无论它是否包含在主键当中，ClickHouse 都会存储类似标记。这些标记让您可以在列文件中直接找到数据。

颗粒的大小通过表引擎参数 `index_granularity` 和 `index_granularity_bytes` 控制。颗粒的行数的在 `[1, index_granularity]` 范围中，这取决于行的大小。如果单行的大小超过了 `index_granularity_bytes` 设置的值，那么一个颗粒的大小会超过 `index_granularity_bytes`。在这种情况下，颗粒的大小等于该行的大小。

主键和索引在查询中的表现

我们以 `(CounterID, Date)` 以主键。排序好的索引的图示会是下面这样：

```
全部数据 : [-----]
CounterID: [aaaaaaaaaaaaaaaaaaaaabbbbcdfffffffggggggghhhhhhhiiiiiiik|||||||]
Date:      [111111122222233312332111122222333211111121222223111122231112233]
标记:      |   |   |   |   |   |   |   |   |   |
标记号:    a,1  a,2  a,3  b,3  e,2  e,3  g,1  h,2  i,1  i,3  l,3
           0     1     2     3     4     5     6     7     8     9     10
```

如果指定查询如下：

- `CounterID in ('a', 'h')`，服务器会读取标记号在 `[0, 3]` 和 `[6, 8)` 区间中的数据。
- `CounterID IN ('a', 'h') AND Date = 3`，服务器会读取标记号在 `[1, 3)` 和 `[7, 8)` 区间中的数据。
- `Date = 3`，服务器会读取标记号在 `[1, 10]` 区间中的数据。

上面例子可以看出使用索引通常会比全表描述要高效。

稀疏索引会引起额外的数据读取。当读取主键单个区间范围的数据时，每个数据块中最多会多读 `index_granularity * 2` 行额外的数据。

稀疏索引使得您可以处理极大量的行，因为大多数情况下，这些索引常驻于内存。

`ClickHouse` 不要求主键唯一，所以您可以插入多条具有相同主键的行。

您可以在 `PRIMARY KEY` 与 `ORDER BY` 条件中使用可为空的类型的表达式，但强烈建议不要这么做。为了启用这项功能，请打开 `allow_nullable_key`，`NULLS_LAST` 规则也适用于 `ORDER BY` 条件中有 `NULL` 值的情况下。

主键的选择

主键中列的数量并没有明确的限制。依据数据结构，您可以在主键包含多些或少些列。这样可以：

- 改善索引的性能。

如果当前主键是 `(a, b)`，在下列情况下添加另一个 `c` 列会提升性能：

- 查询会使用 `c` 列作为条件
- 很长的数据范围（`index_granularity` 的数倍）里 `(a, b)` 都是相同的值，并且这样的情况很普遍。换言之，就是加入另一列后，可以让您的查询略过很长的数据范围。
- 改善数据压缩。

`ClickHouse` 以主键排序片段数据，所以，数据的一致性越高，压缩越好。

- 在 `CollapsingMergeTree` 和 `SummingMergeTree` 引擎里进行数据合并时会提供额外的处理逻辑。

在这种情况下，指定与主键不同的 `排序键` 也是有意义的。

长的主键会对插入性能和内存消耗有负面影响，但主键中额外的列并不影响 `SELECT` 查询的性能。

可以使用 `ORDER BY tuple()` 语法创建没有主键的表。在这种情况下 `ClickHouse` 根据数据插入的顺序存储。如果在使用 `INSERT ... SELECT` 时希望保持数据的排序，请设置 `max_insert_threads = 1`。

想要根据初始顺序进行数据查询，使用 `单线程查询`

选择与排序键不同的主键

`ClickHouse` 可以做到指定一个跟排序键不一样的主键，此时排序键用于在数据片段中进行排序，主键用于在索引文件中进行标记的写入。这种情况下，主键表达式元组必须是排序键表达式元组的前缀（即主键为 `(a, b)`，排序列必须为 `(a, b, **)`）。

当使用 **SummingMergeTree** 和 **AggregatingMergeTree** 引擎时，这个特性非常有用。通常在使用这类引擎时，表里的列分两种：维度和度量。典型的查询会通过任意的 **GROUP BY** 对度量列进行聚合并通过维度列进行过滤。由于 **SummingMergeTree** 和 **AggregatingMergeTree** 会对排序键相同的行进行聚合，所以把所有的维度放进排序键是很自然的做法。但这将导致排序键中包含大量的列，并且排序键会伴随着新添加的维度不断的更新。

在这种情况下合理的做法是，只保留少量的列在主键当中用于提升扫描效率，将维度列添加到排序键中。

对排序键进行 **ALTER** 是轻量级的操作，因为当一个新列同时被加入到表里和排序键里时，已存在的数据片段并不需要修改。由于旧的排序键是新排序键的前缀，并且新添加的列中没有数据，因此在表修改时的数据对于新旧的排序键来说都是有序的。

索引和分区在查询中的应用

对于 **SELECT** 查询，**ClickHouse** 分析是否可以使用索引。如果 **WHERE/PREWHERE** 子句具有下面这些表达式（作为完整 **WHERE** 条件的一部分或全部）则可以使用索引：进行相等/不相等的比较；对主键列或分区列进行 **IN** 运算、有固定前缀的 **LIKE** 运算（如 **name like 'test%'**）、函数运算（部分函数适用），还有对上述表达式进行逻辑运算。

因此，在索引键的一个或多个区间上快速地执行查询是可能的。下面例子中，指定标签；指定标签和日期范围；指定标签和日期；指定多个标签和日期范围等执行查询，都会非常快。

当引擎配置如下时：

```
ENGINE MergeTree() PARTITION BY toYYYYMM(EventDate) ORDER BY (CounterID, EventDate) SETTINGS  
index_granularity=8192
```

这种情况下，这些查询：

```
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 会依据主键索引剪掉不符合的数据，依据按月分区的分区键剪掉那些不包含符合数据的分区。

上文的查询显示，即使索引用于复杂表达式，因为读表操作经过优化，所以使用索引不会比完整扫描慢。

下面这个例子中，不会使用索引。

```
SELECT count() FROM table WHERE CounterID = 34 OR URL LIKE '%upyachka%'
```

要检查 **ClickHouse** 执行一个查询时能否使用索引，可设置 **force_index_by_date** 和 **force_primary_key**。

使用按月分区的分区列允许只读取包含适当日期区间的数据块，这种情况下，数据块会包含很多天（最多整月）的数据。在块中，数据按主键排序，主键第一列可能不包含日期。因此，仅使用日期而没有用主键字段作为条件的查询将会导致需要读取超过这个指定日期以外的数据。

部分单调主键的使用

考虑这样的场景，比如一个月中的天数。它们在一个月的范围内形成一个 **单调序列**，但如果扩展到更大的时间范围它们就不再单调了。这就是一个部分单调序列。如果用户使用部分单调的主键创建表，**ClickHouse** 同样会创建一个稀疏索引。当用户从这类表中查询数据时，**ClickHouse** 会对查询条件进行分析。如果用户希望获取两个索引标记之间的数据并且这两个标记在一个月以内，**ClickHouse** 可以在这种特殊情况下使用到索引，因为它可以计算出查询参数与索引标记之间的距离。

如果查询参数范围内的主键不是单调序列，那么 **ClickHouse** 无法使用索引。在这种情况下，**ClickHouse** 会进行全表扫描。

ClickHouse 在任何主键代表一个部分单调序列的情况下都会使用这个逻辑。

跳数索引

此索引在 `CREATE` 语句的列部分里定义。

```
INDEX index_name expr TYPE type(...) GRANULARITY granularity_value
```

*MergeTree 系列的表可以指定跳数索引。

跳数索引是指数据片段按照粒度(建表时指定的`index_granularity`)分割成小块后，将上述SQL的`granularity_value`数量的小块组合成一个大的块，对这些大块写入索引信息，这样有助于使用`where`筛选时跳过大量不必要的数据，减少`SELECT`需要读取的数据量。

示例

```
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()
...
```

上例中的索引能让 ClickHouse 执行下面这些查询时减少读取数据量。

```
SELECT count() FROM table WHERE s < 'z'
SELECT count() FROM table WHERE u64 * i32 == 10 AND u64 * length(s) >= 1234
```

可用的索引类型

■ `minmax`

存储指定表达式的极值（如果表达式是 `tuple`，则存储 `tuple` 中每个元素的极值），这些信息用于跳过数据块，类似主键。

■ `set(max_rows)`

存储指定表达式的不重复值（不超过 `max_rows` 个，`max_rows=0` 则表示『无限制』）。这些信息可用于检查数据块是否满足 `WHERE` 条件。

■ `ngrambf_v1(n, size_of_bloom_filter_in_bytes, number_of_hash_functions, random_seed)`

存储一个包含数据块中所有 n 元短语 (ngram) 的 布隆过滤器。只可用在字符串上。

可用于优化 `equals`，`like` 和 `in` 表达式的性能。

■ `n` - 短语长度。

■ `size_of_bloom_filter_in_bytes` - 布隆过滤器大小，字节为单位。（因为压缩得好，可以指定比较大的值，如 256 或 512）。

■ `number_of_hash_functions` - 布隆过滤器中使用的哈希函数的个数。

■ `random_seed` - 哈希函数的随机种子。

■ `tokenbf_v1(size_of_bloom_filter_in_bytes, number_of_hash_functions, random_seed)`

跟 `ngrambf_v1` 类似，但是存储的是 token 而不是 ngrams。Token 是由非字母数字的符号分割的序列。

- `bloom_filter(bloom_filter([false_positive]))` – 为指定的列存储布隆过滤器

可选参数`false_positive`用来指定从布隆过滤器收到错误响应的几率。取值范围是(0,1)，默认值：0.025

支持的数据类型：`Int*`, `UInt*`, `Float*`, `Enum`, `Date`, `DateTime`, `String`, `FixedString`, `Array`, `LowCardinality`, `Nullable`。

以下函数会用到这个索引：`equals`, `notEquals`, `in`, `notin`, `has`

```
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
```

函数支持

`WHERE`子句中的条件可以包含对某列数据进行运算的函数表达式，如果列是索引的一部分，ClickHouse会在执行函数时尝试使用索引。不同的函数对索引的支持是不同的。

`set`索引会对所有函数生效，其他索引对函数的生效情况见下表

函数(操作符)/索引	primary key	minmax	ngrambf_v1	tokenbf_v1	bloom_filter
<code>equals (=, ==)</code>	✓	✓	✓	✓	✓
<code>notEquals(!=, \<>)</code>	✓	✓	✓	✓	✓
<code>like</code>	✓	✓	✓	✓	✓
<code>notLike</code>	✓	✓	✗	✗	✗
<code>startsWith</code>	✓	✓	✓	✓	✗
<code>endsWith</code>	✗	✗	✓	✓	✗
<code>multiSearchAny</code>	✗	✗	✓	✗	✗
<code>in</code>	✓	✓	✓	✓	✓
<code>notin</code>	✓	✓	✓	✓	✓
<code>less (\<)</code>	✓	✓	✗	✗	✗
<code>greater (>)</code>	✓	✓	✗	✗	✗
<code>lessOrEquals (\<=)</code>	✓	✓	✗	✗	✗
<code>greaterOrEquals (>=)</code>	✓	✓	✗	✗	✗
<code>empty</code>	✓	✓	✗	✗	✗
<code>notEmpty</code>	✓	✓	✗	✗	✗
<code>hasToken</code>	✗	✗	✗	✓	✗

常量参数小于ngram大小的函数不能使用`ngrambf_v1`进行查询优化。

注意

布隆过滤器可能会包含不符合条件的匹配，所以 `ngrambf_v1`, `tokenbf_v1` 和 `bloom_filter` 索引不能用于结果返回为假的函数，例如：

- 可以用来优化的场景
- `s LIKE '%test%'`
- `NOT s NOT LIKE '%test%'`
- `s = 1`
- `NOT s != 1`
- `startsWith(s, 'test')`
- 不能用来优化的场景
- `NOT s LIKE '%test%'`
- `s NOT LIKE '%test%'`
- `NOT s = 1`
- `s != 1`
- `NOT startsWith(s, 'test')`

并发数据访问

对于表的并发访问，我们使用多版本机制。换言之，当一张表同时被读和更新时，数据从当前查询到的一组片段中读取。没有冗长的锁。插入不会阻碍读取。

对表的读操作是自动并行的。

列和表的 TTL

TTL 用于设置值的生命周期，它既可以为整张表设置，也可以为每个列字段单独设置。表级别的 TTL 还会指定数据在磁盘和卷上自动转移的逻辑。

TTL 表达式的计算结果必须是 **日期** 或 **日期时间** 类型的字段。

示例：

```
TTL time_column  
TTL time_column + interval
```

要定义 `interval`，需要使用 **时间间隔** 操作符。

```
TTL date_time + INTERVAL 1 MONTH  
TTL date_time + INTERVAL 15 HOUR
```

列 TTL

当列中的值过期时，ClickHouse 会将它们替换成该列数据类型的默认值。如果数据片段中列的所有值均已过期，则 ClickHouse 会从文件系统中的数据片段中删除此列。

TTL子句不能被用于主键字段。

示例：

创建表时指定 **TTL**

```
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;
```

为表中已存在的列字段添加 **TTL**

```
ALTER TABLE example_table
MODIFY COLUMN
c String TTL d + INTERVAL 1 DAY;
```

修改列字段的 **TTL**

```
ALTER TABLE example_table
MODIFY COLUMN
c String TTL d + INTERVAL 1 MONTH;
```

表 **TTL**

表可以设置一个用于移除过期行的表达式，以及多个用于在磁盘或卷上自动转移数据片段的表达式。当表中的行过期时，ClickHouse 会删除所有对应的行。对于数据片段的转移特性，必须所有的行都满足转移条件。

```
TTL expr
[DELETE|TO DISK 'xxx'|TO VOLUME 'xxx'][, DELETE|TO DISK 'aaa'|TO VOLUME 'bbb'] ...
[WHERE conditions]
[GROUP BY key_expr [SET v1 = aggr_func(v1) [, v2 = aggr_func(v2) ...]] ]
```

TTL 规则的类型紧跟在每个 **TTL** 表达式后面，它会影响满足表达式时（到达指定时间时）应当执行的操作：

- **DELETE** - 删除过期的行（默认操作）；
- **TO DISK 'aaa'** - 将数据片段移动到磁盘 aaa；
- **TO VOLUME 'bbb'** - 将数据片段移动到卷 bbb.
- **GROUP BY** - 聚合过期的行

使用 **WHERE** 从句，您可以指定哪些过期的行会被删除或聚合(不适用于移动)。**GROUP BY** 表达式必须是表主键的前缀。如果某列不是**GROUP BY** 表达式的一部分，也没有在**SET** 从句显示引用，结果行中相应列的值是随机的(就好像使用了**any** 函数)。

示例：

创建时指定 **TTL**

```
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';
```

修改表的 TTL

```
ALTER TABLE example_table
    MODIFY TTL d + INTERVAL 1 DAY;
```

创建一张表，设置一个月后数据过期，这些过期的行中日期为星期一的删除：

```
CREATE TABLE table_with_where
(
    d DateTime,
    a Int
)
ENGINE = MergeTree
PARTITION BY toYYYYMM(d)
ORDER BY d
TTL d + INTERVAL 1 MONTH DELETE WHERE toDayOfWeek(d) = 1;
```

创建一张表，设置过期的列会被聚合。列x包含每组行中的最大值，y为最小值，d为可能任意值。

```
CREATE TABLE table_for_aggregation
(
    d DateTime,
    k1 Int,
    k2 Int,
    x Int,
    y Int
)
ENGINE = MergeTree
ORDER BY (k1, k2)
TTL d + INTERVAL 1 MONTH GROUP BY k1, k2 SET x = max(x), y = min(y);
```

删除数据

ClickHouse 在数据片段合并时会删除掉过期的数据。

当 ClickHouse 发现数据过期时，它将会执行一个计划外的合并。要控制这类合并的频率，您可以设置 `merge_with_ttl_timeout`。如果该值被设置的太低，它将引发大量计划外的合并，这可能会消耗大量资源。

如果在合并的过程中执行 `SELECT` 查询，则可能会得到过期的数据。为了避免这种情况，可以在 `SELECT` 之前使用 `OPTIMIZE`。

使用多个块设备进行数据存储 介绍

MergeTree 系列表引擎可以将数据存储在多个块设备上。这对某些可以潜在被划分为“冷”“热”的表来说是很有用的。最新数据被定期的查询但只需要很小的空间。相反，详尽的历史数据很少被用到。如果有多个块可用，那么“热”的数据可以放置在快速的磁盘上（比如 NVMe 固态硬盘或内存），“冷”的数据可以放在相对较慢的磁盘上（比如机械硬盘）。

数据片段是 MergeTree 引擎表的最小可移动单元。属于同一个数据片段的数据被存储在同一块磁盘上。数据片段会在后台自动的在磁盘间移动，也可以通过 **ALTER** 查询来移动。

术语

- 磁盘 — 挂载到文件系统的块设备
- 默认磁盘 — 在服务器设置中通过 **path** 参数指定的数据存储
- 卷 — 相同磁盘的顺序列表（类似于 **JBOD**）
- 存储策略 — 卷的集合及他们之间的数据移动规则

以上名称的信息在Clickhouse中系统表**system.storage_policies**和**system.disks**体现。为了应用存储策略，可以在建表时使用**storage_policy**设置。

配置

磁盘、卷和存储策略应当在主配置文件 `config.xml` 或 `config.d` 目录中的独立文件中的 `<storage_configuration>` 标签内定义。

配置结构：

```
<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>
```

标签：

- `<disk_name_N>` — 磁盘名，名称必须与其他磁盘不同。
- `path` — 服务器将用来存储数据 (`data` 和 `shadow` 目录) 的路径，应当以 ‘/’ 结尾。
- `keep_free_space_bytes` — 需要保留的剩余磁盘空间。

磁盘定义的顺序无关紧要。

存储策略配置：

```

<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>

```

标签：

- `policy_name_N` — 策略名称，不能重复。
- `volume_name_N` — 卷名称，不能重复。
- `disk` — 卷中的磁盘。
- `max_data_part_size_bytes` — 卷中的磁盘可以存储的数据片段的最大大小。
- `move_factor` — 当可用空间少于这个因子时，数据将自动的向下一个卷（如果有的话）移动（默认值为 `0.1`）。
- `prefer_not_to_merge` - 禁止在这个卷中进行数据合并。该选项启用时，对该卷的数据不能进行合并。这个选项主要用于慢速磁盘。

配置示例：

```

<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>

    <small_jbod_with_external_no_merges>
        <volumes>
            <main>
                <disk>jbod1</disk>
            </main>
            <external>
                <disk>external</disk>
                <prefer_not_to_merge>true</prefer_not_to_merge>
            </external>
        </volumes>
    </small_jbod_with_external_no_merges>
</policies>
...
</storage_configuration>

```

在给出的例子中，`hdd_in_order` 策略实现了 **循环制** 方法。因此这个策略只定义了一个卷 (`single`)，数据片段会以循环的顺序全部存储到它的磁盘上。当有多个类似的磁盘挂载到系统上，但没有配置 RAID 时，这种策略非常有用。请注意一个每个独立的磁盘驱动都并不可靠，您可能需要用3份或更多的复制份数来补偿它。

如果在系统中有不同类型的磁盘可用，可以使用 `moving_from_ss_to_hdd`。`hot` 卷由 SSD 磁盘 (`fast_ss`) 组成，这个卷上可以存储的数据片段的最大大小为 1GB。所有大于 1GB 的数据片段都会被直接存储到 `cold` 卷上，`cold` 卷包含一个名为 `disk1` 的 HDD 磁盘。

同样，一旦 `fast_ss` 被填充超过 80%，数据会通过后台进程向 `disk1` 进行转移。

存储策略中卷的枚举顺序是很重要的。因为当一个卷被充满时，数据会向下一个卷转移。磁盘的枚举顺序同样重要，因为数据是依次存储在磁盘上的。

在创建表时，可以应用存储策略：

```

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_ss_to_hdd'

```

`default` 存储策略意味着只使用一个卷，这个卷只包含一个在 `<path>` 中定义的磁盘。您可以使用 [ALTER TABLE ... MODIFY SETTING] 来修改存储策略，新的存储策略应该包含所有以前的磁盘和卷，并使用相同的名称。

可以通过 `background_move_pool_size` 设置调整执行后台任务的线程数。

详细说明

对于 MergeTree 表，数据通过以下不同的方式写入到磁盘当中：

- 插入（`INSERT`查询）
- 后台合并和数据变异
- 从另一个副本下载
- `ALTER TABLE ... FREEZE PARTITION` 冻结分区

除了数据变异和冻结分区以外的情况下，数据按照以下逻辑存储到卷或磁盘上：

1. 首个卷（按定义顺序）拥有足够的磁盘空间存储数据片段（`unreserved_space > current_part_size`）并且允许存储给定数据片段的大小（`max_data_part_size_bytes > current_part_size`）
2. 在这个数据卷内，紧挨着先前存储数据的那块磁盘之后的磁盘，拥有比数据片段大的剩余空间。（`unreserved_space - keep_free_space_bytes > current_part_size`）

更进一步，数据变异和分区冻结使用的是 **硬链接**。不同磁盘之间的硬链接是不支持的，所以在这种情况下数据片段都会被存储到原来的那一块磁盘上。

在后台，数据片段基于剩余空间（`move_factor`参数）根据卷在配置文件中定义的顺序进行转移。数据永远不会从最后一个移出也不会从第一个移入。可以通过系统表 `system.part_log`（字段 `type = MOVE_PART`）和 `system.parts`（字段 `path` 和 `disk`）来监控后台的移动情况。具体细节可以通过服务器日志查看。

用户可以通过 `ALTER TABLE ... MOVE PART|PARTITION ... TO VOLUME|DISK ...` 强制移动一个数据片段或分区到另外一个卷，所有后台移动的限制都会被考虑在内。这个查询会自行启动，无需等待后台操作完成。如果没有足够的可用空间或任何必须条件没有被满足，用户会收到报错信息。

数据移动不会妨碍到数据复制。也就是说，同一张表的不同副本可以指定不同的存储策略。

在后台合并和数据变异之后，旧的数据片段会在一定时间后被移除（`old_parts_lifetime`）。在这期间，他们不能被移动到其他的卷或磁盘。也就是说，直到数据片段被完全移除，它们仍然会被磁盘占用空间计算在内。

使用S3进行数据存储

MergeTree系列表引擎允许使用S3存储数据，需要修改磁盘类型为S3。

示例配置：

```

<storage_configuration>
...
<disks>
  <s3>
    <type>s3</type>
    <endpoint>https://storage.yandexcloud.net/my-bucket/root-path/</endpoint>
    <access_key_id>your_access_key_id</access_key_id>
    <secret_access_key>your_secret_access_key</secret_access_key>
    <region></region>

    <server_side_encryption_customer_key_base64>your_base64_encoded_customer_key</server_side_encryption_customer_key_base64>
      <proxy>
        <uri>http://proxy1</uri>
        <uri>http://proxy2</uri>
      </proxy>
      <connect_timeout_ms>10000</connect_timeout_ms>
      <request_timeout_ms>5000</request_timeout_ms>
      <retry_attempts>10</retry_attempts>
      <single_read_retries>4</single_read_retries>
      <min_bytes_for_seek>1000</min_bytes_for_seek>
      <metadata_path>/var/lib/clickhouse/disks/s3/</metadata_path>
      <cache_enabled>true</cache_enabled>
      <cache_path>/var/lib/clickhouse/disks/s3/cache/</cache_path>
      <skip_access_check>false</skip_access_check>
    </s3>
  </disks>
...
</storage_configuration>

```

必须的参数：

- `endpoint` - S3的结点URL，以path或virtual hosted格式书写。
- `access_key_id` - S3的Access Key ID。
- `secret_access_key` - S3的Secret Access Key。

可选参数：

- `region` - S3的区域名称
- `use_environment_credentials` - 从环境变量`AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY`和`AWS_SESSION_TOKEN`中读取认证参数。默认值为`false`。
- `use_insecure_imds_request` - 如果设置为`true`，S3客户端在认证时会使用不安全的IMDS请求。默认值为`false`。
- `proxy` - 访问S3结点URL时代理设置。每一个uri项的值都应该是合法的代理URL。
- `connect_timeout_ms` - Socket连接超时时间，默认值为`10000`，即10秒。
- `request_timeout_ms` - 请求超时时间，默认值为`5000`，即5秒。
- `retry_attempts` - 请求失败后的重试次数，默认值为`10`。
- `single_read_retries` - 读过程中连接丢失后重试次数，默认值为`4`。
- `min_bytes_for_seek` - 使用查找操作，而不是顺序读操作的最小字节数，默认值为`1000`。
- `metadata_path` - 本地存放S3元数据文件的路径，默认值为`/var/lib/clickhouse/disks/<disk_name>/`
- `cache_enabled` - 是否允许缓存标记和索引文件。默认值为`true`。
- `cache_path` - 本地缓存标记和索引文件的路径。默认值为`/var/lib/clickhouse/disks/<disk_name>/cache/`。
- `skip_access_check` - 如果为`true`，Clickhouse启动时不检查磁盘是否可用。默认为`false`。

- `server_side_encryption_customer_key_base64` - 如果指定该项的值，请求时会加上为了访问SSE-C加密数据而必须的头信息。

S3磁盘也可以设置冷热存储：

```
<storage_configuration>
...
<disks>
  <s3>
    <type>s3</type>
    <endpoint>https://storage.yandexcloud.net/my-bucket/root-path/</endpoint>
    <access_key_id>your_access_key_id</access_key_id>
    <secret_access_key>your_secret_access_key</secret_access_key>
  </s3>
</disks>
<policies>
  <s3_main>
    <volumes>
      <main>
        <disk>s3</disk>
      </main>
    </volumes>
  </s3_main>
  <s3_cold>
    <volumes>
      <main>
        <disk>default</disk>
      </main>
      <external>
        <disk>s3</disk>
      </external>
    </volumes>
    <move_factor>0.2</move_factor>
  </s3_cold>
</policies>
...
</storage_configuration>
```

指定了`cold`选项后，本地磁盘剩余空间如果小于`move_factor * disk_size`，或有TTL设置时，数据就会定时迁移至S3了。

ReplacingMergeTree

该引擎和 [MergeTree](#) 的不同之处在于它会删除排序键值相同的重复项。

数据的去重只会在数据合并期间进行。合并在后台一个不确定的时间进行，因此你无法预先作出计划。有一些数据可能仍未被处理。尽管你可以调用 `OPTIMIZE` 语句发起计划外的合并，但请不要依靠它，因为 `OPTIMIZE` 语句会引发对数据的大量读写。

因此，`ReplacingMergeTree` 适用于在后台清除重复的数据以节省空间，但是它不保证没有重复的数据出现。

建表

```
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]
[SAMPLE BY expr]
[SETTINGS name=value, ...]
```

有关建表参数的描述，可参考 [创建表](#)。

ReplacingMergeTree 的参数

- `ver` — 版本列。类型为 `UInt*`, `Date` 或 `DateTime`。可选参数。

在数据合并的时候，ReplacingMergeTree 从所有具有相同排序键的行中选择一行留下：

- 如果 `ver` 列未指定，保留最后一条。
- 如果 `ver` 列已指定，保留 `ver` 值最大的版本。

子句

创建 ReplacingMergeTree 表时，需要使用与创建 MergeTree 表时相同的 子句。

▶ 已弃用的建表方法

SummingMergeTree

该引擎继承自 MergeTree。区别在于，当合并 SummingMergeTree 表的数据片段时，ClickHouse 会把所有具有相同主键的行合并为一行，该行包含了被合并的行中具有数值数据类型的列的汇总值。如果主键的组合方式使得单个键值对应于大量的行，则可以显著的减少存储空间并加快数据查询的速度。

我们推荐将该引擎和 MergeTree 一起使用。例如，在准备做报告的时候，将完整的数据存储在 MergeTree 表中，并且使用 SummingMergeTree 来存储聚合数据。这种方法可以使你避免因为使用不正确的主键组合方式而丢失有价值的数据。

建表

```
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, ...]
```

请求参数的描述，参考 [请求描述](#)。

SummingMergeTree 的参数

- `columns` - 包含了将要被汇总的列的列名的元组。可选参数。

所选的列必须是数值类型，并且不可位于主键中。

如果没有指定 `columns`，ClickHouse 会把所有不在主键中的数值类型的列都进行汇总。

子句

创建 SummingMergeTree 表时，需要与创建 MergeTree 表时相同的 子句。

▶ 已弃用的建表方法

用法示例

考虑如下的表：

```
CREATE TABLE summtt
(
    key UInt32,
    value UInt32
)
ENGINE = SummingMergeTree()
ORDER BY key
```

向其中插入数据：

```
:) INSERT INTO summtt Values(1,1),(1,2),(2,1)
```

ClickHouse 可能不会完整的汇总所有行（[见下文](#)），因此我们在查询中使用了聚合函数 `sum` 和 `GROUP BY` 子句。

```
SELECT key, sum(value) FROM summtt GROUP BY key
```

key	sum(value)
2	1
1	3

数据处理

当数据被插入到表中时，他们将被原样保存。ClickHouse 定期合并插入的数据片段，并在这个时候对所有具有相同主键的行中的列进行汇总，将这些行替换为包含汇总数据的一行记录。

ClickHouse 会按片段合并数据，以至于不同的数据片段中会包含具有相同主键的行，即单个汇总片段将会是不完整的。因此，聚合函数 `sum()` 和 `GROUP BY` 子句应该在 `(SELECT)` 查询语句中被使用，如上文中的例子所述。

汇总的通用规则

列中数值类型的值会被汇总。这些列的集合在参数 `columns` 中被定义。

如果用于汇总的所有列中的值均为 0，则该行会被删除。

如果列不在主键中且无法被汇总，则会在现有的值中任选一个。

主键所在的列中的值不会被汇总。

AggregateFunction 列中的汇总

对于 `AggregateFunction` 类型的列，ClickHouse 根据对应函数表现为 `AggregatingMergeTree` 引擎的聚合。

嵌套结构

表中可以具有以特殊方式处理的嵌套数据结构。

如果嵌套表的名称以 `Map` 结尾，并且包含至少两个符合以下条件的列：

- 第一列是数值类型 `(*Int*, Date, DateTime)`，我们称之为 `key`,
- 其他的列是可计算的 `(*Int*, Float32/64)`，我们称之为 `(values...)`,

然后这个嵌套表会被解释为一个 `key => (values...)` 的映射，当合并它们的行时，两个数据集中的元素会被根据 `key` 合并为相应的 `(values...)` 的汇总值。

示例：

```
[(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)]
```

请求数据时，使用 [sumMap\(key,value\)](#) 函数来对 Map 进行聚合。

对于嵌套数据结构，你无需在列的元组中指定列以进行汇总。

数据副本

只有 MergeTree 系列里的表可支持副本：

- ReplicatedMergeTree
- ReplicatedSummingMergeTree
- ReplicatedReplacingMergeTree
- ReplicatedAggregatingMergeTree
- ReplicatedCollapsingMergeTree
- ReplicatedVersionedCollapsingMergetree
- ReplicatedGraphiteMergeTree

副本是表级别的，不是整个服务器级的。所以，服务器里可以同时有复制表和非复制表。

副本不依赖分片。每个分片有它自己的独立副本。

对于 `INSERT` 和 `ALTER` 语句操作数据的会在压缩的情况下被复制（更多信息，看 [ALTER](#)）。

而 `CREATE`，`DROP`，`ATTACH`，`DETACH` 和 `RENAME` 语句只会在单个服务器上执行，不会被复制。

- `The CREATE TABLE` 在运行此语句的服务器上创建一个新的可复制表。如果此表已存在其他服务器上，则给该表添加新副本。
- `The DROP TABLE` 删除运行此查询的服务器上的副本。
- `The RENAME` 重命名一个副本。换句话说，可复制表不同的副本可以有不同的名称。

要使用副本，需在配置文件中设置 ZooKeeper 集群的地址。例如：

```
<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>
```

需要 ZooKeeper 3.4.5 或更高版本。

你可以配置任何现有的 ZooKeeper 集群，系统会使用里面的目录来存取元数据（该目录在创建可复制表时指定）。

如果配置文件中没有设置 ZooKeeper，则无法创建复制表，并且任何现有的复制表都将变为只读。

`SELECT` 查询并不需要借助 ZooKeeper，副本并不影响 `SELECT` 的性能，查询复制表与非复制表速度是一样的。查询分布式表时，ClickHouse的处理方式可通过设置 `max_replica_delay_for_distributed_queries` 和 `fallback_to_stale_replicas_for_distributed_queries` 修改。

对于每个 `INSERT` 语句，会通过几个事务将十来个记录添加到 ZooKeeper。（确切地说，这是针对每个插入的数据块；每个 `INSERT` 语句的每 `max_insert_block_size = 1048576` 行和最后剩余的都各算作一个块。）相比非复制表，写 zk 会导致 `INSERT` 的延迟略长一些。但只要你按照建议每秒不超过一个 `INSERT` 地批量插入数据，不会有任何问题。一个 ZooKeeper 集群能给整个 ClickHouse 集群支撑协调每秒几百个 `INSERT`。数据插入的吞吐量（每秒的行数）可以跟不用复制的数据一样高。

对于非常大的集群，你可以把不同的 ZooKeeper 集群用于不同的分片。然而，即使 Yandex.Metrica 集群（大约 300 台服务器）也证明还不需要这么做。

复制是多主异步。`INSERT` 语句（以及 `ALTER`）可以发给任意可用的服务器。数据会先插入到执行该语句的服务器上，然后被复制到其他服务器。由于它是异步的，在其他副本上最近插入的数据会有一些延迟。如果部分副本不可用，则数据在其可用时再写入。副本可用的情况下，则延迟时长是通过网络传输压缩数据块所需的时间。

默认情况下，`INSERT` 语句仅等待一个副本写入成功后返回。如果数据只成功写入一个副本后该副本所在的服务器不再存在，则存储的数据会丢失。要启用数据写入多个副本才确认返回，使用 `insert_quorum` 选项。

单个数据块写入是原子的。`INSERT` 的数据按每块最多 `max_insert_block_size = 1048576` 行进行分块，换句话说，如果 `INSERT` 插入的行少于 `1048576`，则该 `INSERT` 是原子的。

数据块会去重。对于被多次写的相同数据块（大小相同且具有相同顺序的相同行的数据块），该块仅会写入一次。这样设计的原因是万一在网络故障时客户端应用程序不知道数据是否成功写入DB，此时可以简单地重复 `INSERT`。把相同的数据发送给多个副本 `INSERT` 并不会有问题是。因为这些 `INSERT` 是完全相同的（会被去重）。去重参数参看服务器设置 `merge_tree`。（注意：Replicated*MergeTree 才会去重，不需要 zookeeper 的不带 MergeTree 不会去重）

在复制期间，只有要插入的源数据通过网络传输。进一步的数据转换（合并）会在所有副本上以相同的方式进行处理执行。这样可以最大限度地减少网络使用，这意味着即使副本在不同的数据中心，数据同步也能工作良好。（能在不同数据中心中的同步数据是副本机制的主要目标。）

你可以给数据做任意多的副本。Yandex.Metrica 在生产中使用双副本。某些情况下，给每台服务器都使用 RAID-5 或 RAID-6 和 RAID-10。是一种相对可靠和方便的解决方案。

系统会监视副本数据同步情况，并能在发生故障后恢复。故障转移是自动的（对于小的数据差异）或半自动的（当数据差异很大时，这可能意味着有配置错误）。

创建复制表

在表引擎名称上加上 `Replicated` 前缀。例如：`ReplicatedMergeTree`。

Replicated*MergeTree 参数

- `zoo_path` — ZooKeeper 中该表的路径。
- `replica_name` — ZooKeeper 中的该表的副本名称。

示例：

```
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)
```

已弃用的建表语法示例：

```
CREATE TABLE table_name
(
    EventDate DateTime,
    CounterID UInt32,
    UserID UInt32
) ENGINE = ReplicatedMergeTree('/clickhouse/tables/{layer}-{shard}/table_name', '{replica}', EventDate,
intHash32(UserID), (CounterID, EventDate, intHash32(UserID), EventTime), 8192)
```

如上例所示，这些参数可以包含宏替换的占位符，即大括号的部分。它们会被替换为配置文件里‘macros’那部分配置的值。示例：

```
<macros>
<layer>05</layer>
<shard>02</shard>
<replica>example05-02-1.yandex.ru</replica>
</macros>
```

«ZooKeeper 中该表的路径»对每个可复制表都要是唯一的。不同分片上的表要有不同的路径。
这种情况下，路径包含下面这些部分：

/clickhouse/tables/ 是公共前缀，我们推荐使用这个。

{layer}-{shard} 是分片标识部分。在此示例中，由于 Yandex.Metrica 集群使用了两级分片，所以它是由两部分组成的。
但对于大多数情况来说，你只需保留 {shard} 占位符即可，它会替换展开为分片标识。

table_name 是该表在 ZooKeeper 中的名称。使其与 ClickHouse 中的表名相同比较好。这里它被明确定义，跟 ClickHouse 表名不一样，它并不会被 RENAME 语句修改。

HINT：你可以在前面添加一个数据库名称 table_name 也是例如。 db_name.table_name

副本名称用于标识同一个表分片的不同副本。你可以使用服务器名称，如上例所示。同个分片中不同副本的副本名称要唯一。

你也可以显式指定这些参数，而不是使用宏替换。对于测试和配置小型集群这可能会很方便。但是，这种情况下，则不能使用分布式 DDL 语句（ON CLUSTER）。

使用大型集群时，我们建议使用宏替换，因为它可以降低出错的可能性。

在每个副本服务器上运行 CREATE TABLE 查询。将创建新的复制表，或给现有表添加新副本。

如果其他副本上已包含了某些数据，在表上添加新副本，则在运行语句后，数据会从其他副本复制到新副本。换句话说，新副本会与其他副本同步。

要删除副本，使用 DROP TABLE。但它只删除那个 - 位于运行该语句的服务器上的副本。

故障恢复

如果服务器启动时 ZooKeeper 不可用，则复制表会切换为只读模式。系统会定期尝试去连接 ZooKeeper。

如果在 INSERT 期间 ZooKeeper 不可用，或者在与 ZooKeeper 交互时发生错误，则抛出异常。

连接到 ZooKeeper 后，系统会检查本地文件系统中的数据集是否与预期的数据集（ZooKeeper 存储此信息）一致。如果存在轻微的不一致，系统会通过与副本同步数据来解决。

如果系统检测到损坏的数据片段（文件大小错误）或无法识别的片段（写入文件系统但未记录在 ZooKeeper 中的部分），则会把它们移动到‘detached’子目录（不会删除）。而副本中其他任何缺少的但正常数据片段都会被复制同步。

注意，ClickHouse 不会执行任何破坏性操作，例如自动删除大量数据。

当服务器启动（或与 ZooKeeper 建立新会话）时，它只检查所有文件的数量和大小。如果文件大小一致但中间某处已有字节被修改过，不会立即被检测到，只有在尝试读取 `SELECT` 查询的数据时才会检测到。该查询会引发校验和不匹配或压缩块大小不一致的异常。这种情况下，数据片段会添加到验证队列中，并在必要时从其他副本中复制。

如果本地数据集与预期数据的差异太大，则会触发安全机制。服务器在日志中记录此内容并拒绝启动。这种情况很可能是配置错误，例如，一个分片上的副本意外配置为别的分片上的副本。然而，此机制的阈值设置得相当低，在正常故障恢复期间可能会出现这种情况。在这种情况下，数据恢复则是半自动模式，通过用户主动操作触发。

要触发启动恢复，可在 ZooKeeper 中创建节点 `/path_to_table/replica_name/flags/force_restore_data`，节点值可以是任何内容，或运行命令来恢复所有的可复制表：

```
sudo -u clickhouse touch /var/lib/clickhouse/flags/force_restore_data
```

然后重启服务器。启动时，服务器会删除这些标志并开始恢复。

在数据完全丢失后的恢复

如果其中一个服务器的所有数据和元数据都消失了，请按照以下步骤进行恢复：

1. 在服务器上安装 ClickHouse。在包含分片标识符和副本的配置文件中正确定义宏配置，如果有用到的话。
2. 如果服务器上有非复制表则必须手动复制，可以从副本服务器上（在 `/var/lib/clickhouse/data/db_name/table_name/` 目录中）复制它们的数据。
3. 从副本服务器上中复制位于 `/var/lib/clickhouse/metadata/` 中的表定义信息。如果在表定义信息中显式指定了分片或副本标识符，请更正它以使其对应于该副本。（另外，启动服务器，然后会在 `/var/lib/clickhouse/metadata/` 中的 `.sql` 文件中生成所有的 `ATTACH TABLE` 语句。）
4. 要开始恢复，ZooKeeper 中创建节点 `/path_to_table/replica_name/flags/force_restore_data`，节点内容不限，或运行命令来恢复所有复制的表：`sudo -u clickhouse touch /var/lib/clickhouse/flags/force_restore_data`

然后启动服务器（如果它已运行则重启）。数据会从副本中下载。

另一种恢复方式是从 ZooKeeper (`/path_to_table/replica_name`) 中删除有数据丢的副本的所有元信息，然后再按照«[创建可复制表](#)»中的描述重新创建副本。

恢复期间的网络带宽没有限制。特别注意这一点，尤其是要一次恢复很多副本。

MergeTree 转换为 ReplicatedMergeTree

我们使用 MergeTree 来表示 MergeTree 系列中的所有表引擎，ReplicatedMergeTree 同理。

如果你有一个手动同步的 MergeTree 表，您可以将其转换为可复制表。如果你已经在 MergeTree 表中收集了大量数据，并且现在要启用复制，则可以执行这些操作。

如果各个副本上的数据不一致，则首先对其进行同步，或者除保留的一个副本外，删除其他所有副本上的数据。

重命名现有的 MergeTree 表，然后使用旧名称创建 ReplicatedMergeTree 表。

将数据从旧表移动到新表 (`/var/lib/clickhouse/data/db_name/table_name/`) 目录内的 ‘detached’ 目录中。

然后在其中一个副本上运行 `ALTER TABLE ATTACH PARTITION`，将这些数据片段添加到工作集中。

ReplicatedMergeTree 转换为 MergeTree

使用其他名称创建 MergeTree 表。将具有 ReplicatedMergeTree 表数据的目录中的所有数据移动到新表的数据目录中。然后删除 ReplicatedMergeTree 表并重新启动服务器。

如果你想在不启动服务器的情况下清除 ReplicatedMergeTree 表：

- 删除元数据目录中的相应 `.sql` 文件 (`/var/lib/clickhouse/metadata/`)。
- 删除 ZooKeeper 中的相应路径 (`/path_to_table/replica_name`)。

之后，你可以启动服务器，创建一个 MergeTree 表，将数据移动到其目录，然后重新启动服务器。

当 ZooKeeper 集群中的元数据丢失或损坏时恢复方法

如果 ZooKeeper 中的数据丢失或损坏，如上所述，你可以通过将数据转移到非复制表来保存数据。

自定义分区键

MergeTree 系列的表（包括 可复制表）可以使用分区。基于 MergeTree 表的 物化视图 也支持分区。

分区是在一个表中通过指定的规则划分而成的逻辑数据集。可以按任意标准进行分区，如按月，按日或按事件类型。为了减少需要操作的数据，每个分区都是分开存储的。访问数据时，ClickHouse 尽量使用这些分区的最小子集。

分区是在 建表 时通过 PARTITION BY expr 子句指定的。分区键可以是表中列的任意表达式。例如，指定按月分区，表达式为 toYYYYMM(date_column)：

```
CREATE TABLE visits
(
    VisitDate Date,
    Hour UInt8,
    ClientID UUID
)
ENGINE = MergeTree()
PARTITION BY toYYYYMM(VisitDate)
ORDER BY Hour;
```

分区键也可以是表达式元组（类似 主键）。例如：

```
ENGINE = ReplicatedCollapsingMergeTree('/clickhouse/tables/name', 'replica1', Sign)
PARTITION BY (toMonday(StartDate), EventType)
ORDER BY (CounterID, StartDate, intHash32(UserID));
```

上例中，我们设置按一周内的事件类型分区。

新数据插入到表中时，这些数据会存储为按主键排序的新片段（块）。插入后 10-15 分钟，同一分区的各个片段会合并为一个整个片段。

注意

那些有相同分区表达式值的数据片段才会合并。这意味着 你不应该用太精细的分区方案（超过一千个分区）。否则，会因为文件系统中的文件数量过多和需要打开的文件描述符过多，导致 SELECT 查询效率不佳。

可以通过 system.parts 表查看表片段和分区信息。例如，假设我们有一个 visits 表，按月分区。对 system.parts 表执行 SELECT：

```
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

`partition` 列存储分区的名称。此示例中有两个分区：201901 和 201902。在 `ALTER ... PARTITION` 语句中你可以使用该列值来指定分区名称。

`name` 列为分区中数据片段的名称。在 `ALTER ATTACH PART` 语句中你可以使用此列值中来指定片段名称。

这里我们拆解下第一个数据片段的名称：`201901_1_3_1`：

- `201901` 是分区名称。
- `1` 是数据块的最小编号。
- `3` 是数据块的最大编号。
- `1` 是块级别（即在由块组成的合并树中，该块在树中的深度）。

注意

旧类型表的片段名称为：`20190117_20190123_2_2_0`（最小日期 - 最大日期 - 最小块编号 - 最大块编号 - 块级别）。

`active` 列为片段状态。`1` 代表激活状态；`0` 代表非激活状态。非激活片段是那些在合并到较大片段之后剩余的源数据片段。损坏的数据片段也表示为非活动状态。

正如在示例中所看到的，同一分区中有几个独立的片段（例如，`201901_1_3_1`和`201901_1_9_2`）。这意味着这些片段尚未合并。ClickHouse 会定期的对插入的数据片段进行合并，大约是在插入后15分钟左右。此外，你也可以使用 `OPTIMIZE` 语句发起一个计划外的合并。例如：

```
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

非激活片段会在合并后的10分钟左右被删除。

查看片段和分区信息的另一种方法是进入表的目录：`/var/lib/clickhouse/data/<database>/<table>/`。例如：

```
/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
```

'201901_1_1_0'，'201901_1_7_1' 等文件夹是数据片段的目录。每个片段都与一个对应的分区相关，并且只包含这个月的数据（本例中的表按月分区）。

detached 目录存放着使用 **DETACH** 语句从表中卸载的片段。损坏的片段不会被删除而是也会移到该目录下。服务器不会去使用 **detached** 目录中的数据片段。因此你可以随时添加，删除或修改此目录中的数据 – 在运行 **ATTACH** 语句前，服务器不会感知到。

注意，在操作服务器时，你不能手动更改文件系统上的片段集或其数据，因为服务器不会感知到这些修改。对于非复制表，可以在服务器停止时执行这些操作，但不建议这样做。对于复制表，在任何情况下都不要更改片段文件。

ClickHouse 支持对分区执行这些操作：删除分区，将分区从一个表复制到另一个表，或创建备份。了解分区的所有操作，请参阅 [分区和片段的操作](#) 一节。

日志引擎系列

这些引擎是为了需要写入许多小数据量（少于一百万行）的表的场景而开发的。

这系列的引擎有：

- [StripeLog](#)
- [日志](#)
- [TinyLog](#)

共同属性

引擎：

- 数据存储在磁盘上。
- 写入时将数据追加在文件末尾。
- 不支持 **突变** 操作。
- 不支持索引。

这意味着 `SELECT` 在范围查询时效率不高。

- 非原子地写入数据。

如果某些事情破坏了写操作，例如服务器的异常关闭，你将会得到一张包含了损坏数据的表。

差异

[Log](#) 和 [StripeLog](#) 引擎支持：

- 并发访问数据的锁。

`INSERT` 请求执行过程中表会被锁定，并且其他的读写数据的请求都会等待直到锁定被解除。如果没有写数据的请求，任意数量的读请求都可以并发执行。

- 并行读取数据。

在读取数据时，ClickHouse 使用多线程。每个线程处理不同的数据块。

Log 引擎为表中的每一列使用不同的文件。**StripeLog** 将所有的数据存储在一个文件中。因此 **StripeLog** 引擎在操作系统中使用更少的描述符，但是 **Log** 引擎提供更高的读性能。

TinyLog 引擎是该系列中最简单的引擎并且提供了最少的功能和最低的性能。**TinyLog** 引擎不支持并行读取和并发数据访问，并将每一列存储在不同的文件中。它比其余两种支持并行读取的引擎的读取速度更慢，并且使用了和 **Log** 引擎同样多的描述符。你可以在简单的低负载的情景下使用它。

Log

Log 与 **TinyLog** 的不同之处在于，«标记»的小文件与列文件存在一起。这些标记写在每个数据块上，并且包含偏移量，这些偏移量指示从哪里开始读取文件以便跳过指定的行数。这使得可以在多个线程中读取表数据。对于并发数据访问，可以同时执行读取操作，而写入操作则阻塞读取和其它写入。**Log** 引擎不支持索引。同样，如果写入表失败，则该表将被破坏，并且从该表读取将返回错误。**Log** 引擎适用于临时数据，**write-once** 表以及测试或演示目的。

StripeLog

该引擎属于日志引擎系列。请在[日志引擎系列](#)文章中查看引擎的共同属性和差异。

在你需要写入许多小数据量（小于一百万行）的表的场景下使用这个引擎。

建表

```
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
```

查看[建表](#)请求的详细说明。

写数据

StripeLog 引擎将所有列存储在一个文件中。对每一次 **Insert** 请求，ClickHouse 将数据块追加在表文件的末尾，逐列写入。

ClickHouse 为每张表写入以下文件：

- **data.bin** — 数据文件。
- **index.mrk** — 带标记的文件。标记包含了已插入的每个数据块中每列的偏移量。

StripeLog 引擎不支持 **ALTER UPDATE** 和 **ALTER DELETE** 操作。

读数据

带标记的文件使得 ClickHouse 可以并行的读取数据。这意味着 `SELECT` 请求返回行的顺序是不可预测的。使用 `ORDER BY` 子句对行进行排序。

使用示例

建表：

```
CREATE TABLE stripe_log_table
(
    timestamp DateTime,
    message_type String,
    message String
)
ENGINE = StripeLog
```

插入数据：

```
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')
```

我们使用两次 `INSERT` 请求从而在 `data.bin` 文件中创建两个数据块。

ClickHouse 在查询数据时使用多线程。每个线程读取单独的数据块并在完成后独立的返回结果行。这样的结果是，大多数情况下，输出中块的顺序和输入时相应块的顺序是不同的。例如：

```
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

对结果排序（默认增序）：

```
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

TinyLog

最简单的表引擎，用于将数据存储在磁盘上。每列都存储在单独的压缩文件中。写入时，数据将附加到文件末尾。

并发数据访问不受任何限制：

- 如果同时从表中读取并在不同的查询中写入，则读取操作将抛出异常
- 如果同时写入多个查询中的表，则数据将被破坏。

这种表引擎的典型用法是 `write-once`：首先只写入一次数据，然后根据需要多次读取。查询在单个流中执行。换句话说，此引擎适用于相对较小的表（建议最多 1,000,000 行）。如果您有许多小表，则使用此表引擎是适合的，因为它比 Log 引擎更简单（需要打开的文件更少）。当您拥有大量小表时，可能会导致性能低下，但在可能已经在其它 DBMS 时使用过，则您可能会发现切换使用 TinyLog 类型的表更容易。不支持索引。

在 Yandex.Metrica 中，TinyLog 表用于小批量处理的中间数据。

MongoDB

MongoDB 引擎是只读表引擎，允许从远程 MongoDB 集合中读取数据(`SELECT`查询)。引擎只支持非嵌套的数据类型。不支持 `INSERT` 查询。

创建一张表

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name
(
    name1 [type1],
    name2 [type2],
    ...
) ENGINE = MongoDB(host:port, database, collection, user, password);
```

引擎参数

- `host:port` — MongoDB 服务器地址。
- `database` — 数据库名称。
- `collection` — 集合名称。
- `user` — MongoDB 用户。
- `password` — 用户密码。

用法示例

ClickHouse 中的表，从 MongoDB 集合中读取数据：

```
CREATE TABLE mongo_table
(
    key UInt64,
    data String
) ENGINE = MongoDB('mongo1:27017', 'test', 'simple_table', 'testuser', 'clickhouse');
```

查询：

```
SELECT COUNT() FROM mongo_table;
```

```
count()
4 |
```

S3 表引擎

这个引擎提供与 Amazon S3 生态系统的集成。这个引擎类似于 HDFS 引擎，但提供了 S3 特有的功能。

创建表

```
CREATE TABLE s3_engine_table (name String, value UInt32)
ENGINE = S3(path, [aws_access_key_id, aws_secret_access_key,] format, [compression])
```

引擎参数

- **path** — 带有文件路径的 Bucket url。在只读模式下支持以下通配符: *, ?, {abc,def} 和 {N..M} 其中 N, M 是数字, 'abc', 'def' 是字符串。更多信息见[下文](#)。
- **format** — 文件的格式。
- **aws_access_key_id, aws_secret_access_key** - AWS 账号的长期凭证。你可以使用凭证来对你的请求进行认证。参数是可选的。如果没有指定凭据, 将从配置文件中读取凭据。更多信息参见[使用 S3 来存储数据](#)。
- **compression** — 压缩类型。支持的值: none, gzip/gz, brotli/br, xz/LZMA, zstd/zst。参数是可选的。默认情况下, 通过文件扩展名自动检测压缩类型。

示例

1. 创建 s3_engine_table 表:

```
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. 填充文件:

```
INSERT INTO s3_engine_table VALUES ('one', 1), ('two', 2), ('three', 3);
```

3. 查询数据:

```
SELECT * FROM s3_engine_table LIMIT 2;
```

name	value
one	1
two	2

虚拟列

- **_path** — 文件路径。
- **_file** — 文件名。

有关虚拟列的更多信息, 见[这里](#)。

实施细则

- 读取和写入可以是并行的
- 以下是不支持的:
 - ALTER 和 SELECT..SAMPLE 操作。
 - 索引。
 - 复制。

路径中的通配符

`path` 参数可以使用类 bash 的通配符来指定多个文件。对于正在处理的文件应该存在并匹配到整个路径模式。文件列表的确定是在 `SELECT` 的时候进行（而不是在 `CREATE` 的时候）。

- `*` — 替代任何数量的任何字符，除了 / 以及空字符串。
- `?` — 替代任何单个字符。
- `{some_string,another_string,yet_another_one}` — 替代 `'some_string', 'another_string', 'yet_another_one'` 字符串。
- `{N..M}` — 替换 N 到 M 范围内的任何数字，包括两个边界的值。N 和 M 可以以 0 开头，比如 `000..078`

带 `{}` 的结构类似于 [远程表函数](#)。

示例

1. 假设我们在 S3 上有几个 CSV 格式的文件，URI如下：

- `'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'`

有几种方法来创建由所有六个文件组成的数据表：

第一种方式：

```
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');
```

另一种方式：

```
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');
```

表由两个目录中的所有文件组成（所有文件应满足查询中描述的格式和模式）。

```
CREATE TABLE table_with_asterisk (name String, value UInt32) ENGINE = S3('https://storage.yandexcloud.net/my-test-bucket-768/{some,another}_prefix/*', 'CSV');
```

如果文件列表中包含有从零开头的数字范围，请对每个数字分别使用带括号的结构，或者使用`?`。

示例

使用文件 `file-000.csv`, `file-001.csv`, ..., `file-999.csv` 来创建表：

```
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');
```

虚拟列

- `_path` — 文件路径。

- `_file` — 文件名。

另请参阅

- 虚拟列

S3 相关的设置

以下设置可以在查询执行前设置，也可以放在配置文件中。

- `s3_max_single_part_upload_size` - 使用单文件上传至 S3 的对象的最大文件大小。默认值是64Mb。
- `s3_min_upload_part_size` - 使用S3多文件块上传时，文件块的最小文件大小。默认值是512Mb。
- `s3_max_redirects` - 允许的最大S3重定向跳数。默认值是10。
- `s3_single_read_retries` - 单次读取时的最大尝试次数。默认值是4。

安全考虑：如果恶意用户可以指定任意的 S3 网址，`s3_max_redirects`参数必须设置为零，以避免SSRF攻击；或者，必须在服务器配置中指定`remote_host_filter`。

基于 Endpoint 的设置

在配置文件中可以为给定的端点指定以下设置（将通过URL的准确前缀来匹配）。

- `endpoint` - 指定一个端点的前缀。必要参数。
- `access_key_id`和`secret_access_key` - 用于指定端点的登陆凭据。可选参数。
- `use_environment_credentials` - 如果设置为true，S3客户端将尝试从环境变量和Amazon EC2元数据中为指定的端点获取证书。可选参数，默认值是false。
- `region` - 指定S3的区域名称。可选参数。
- `use_insecure_imds_request` - 如果设置为true，S3客户端将使用不安全的 IMDS 请求，同时从Amazon EC2 元数据获取证书。可选参数，默认值是false。
- `header` - 添加指定的HTTP头到给定端点的请求中。可选参数，可以使用多次此参数来添加多个值。
- `server_side_encryption_customer_key_base64` - 如果指定，需要指定访问 SSE-C 加密的 S3 对象所需的头信息。可选参数。
- `max_single_read_retries` - 单次读取时的最大尝试次数。默认值是4。可选参数。

示例：

```
<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>
```

用法

假设我们在 S3 上有几个 CSV 格式的文件，URI 如下：

- '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'

1. 有几种方式来制作由所有六个文件组成的表格，其中一种方式如下：

```
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. 另一种方式：

```
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. 表由两个目录中的所有文件组成（所有文件应满足查询中描述的格式和模式）：

```
CREATE TABLE table_with_asterisk (name String, value UInt32)
ENGINE = S3('https://storage.yandexcloud.net/my-test-bucket-768/{some,another}_prefix/*', 'CSV');
```

Warning

如果文件列表中包含有从0开头的数字范围，请对每个数字分别使用带括号的结构，或者使用?.

4. 从文件file-000.csv, file-001.csv, ..., file-999.csv创建表：

```
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');
```

另请参阅

- [S3 表函数](#)

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 引擎

这个引擎允许 ClickHouse 与 [rocksdb](#) 进行集成。

创建一张表

```
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)
```

必要参数:

- `primary_key_name` – any column name in the column list.
- 必须指定 `primary key`, 仅支持主键中的一个列. 主键将被序列化为二进制的[rocksdb key](#).
- 主键以外的列将以相应的顺序在二进制中序列化为[rocksdb](#)值.
- 带有键 `equals` 或 `in` 过滤的查询将被优化为从 `rocksdb` 进行多键查询.

示例：

```
CREATE TABLE test
(
    `key` String,
    `v1` UInt32,
    `v2` String,
    `v3` Float32,
)
ENGINE = EmbeddedRocksDB
PRIMARY KEY key
```

RabbitMQ 引擎

该引擎允许 ClickHouse 与 [RabbitMQ](#) 进行集成。

[RabbitMQ](#) 可以让你：

- 发布或订阅数据流。
- 在数据流可用时进行处理。

创建一张表

```
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',
    rabbitmq_exchange_name = 'exchange_name',
    rabbitmq_format = 'data_format'[,]
    [rabbitmq_exchange_type = 'exchange_type',]
    [rabbitmq_routing_key_list = 'key1,key2,...',]
    [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]
```

必要参数：

- `rabbitmq_host_port` – 主机名:端口号 (比如, `localhost:5672`).
- `rabbitmq_exchange_name` – RabbitMQ exchange 名称.
- `rabbitmq_format` – 消息格式. 使用与 `SQLFORMAT` 函数相同的标记, 如 `JSONEachRow`. 更多信息, 请参阅 [Formats](#) 部分.

可选参数：

- `rabbitmq_exchange_type` – RabbitMQ exchange 的类型: `direct`, `fanout`, `topic`, `headers`, `consistent_hash`. 默认是: `fanout`.
- `rabbitmq_routing_key_list` – 一个以逗号分隔的路由键列表.
- `rabbitmq_rowDelimiter` – 用于消息结束的分隔符.

- `rabbitmq_schema` – 如果格式需要模式定义，必须使用该参数。比如, **Cap'n Proto** 需要模式文件的路径以及根 `schema.capnp:Message` 对象的名称。
- `rabbitmq_num_consumers` – 每个表的消费者数量。默认：`1`。如果一个消费者的吞吐量不够，可以指定更多的消费者。
- `rabbitmq_num_queues` – 队列的总数。默认值：`1`. 增加这个数字可以显著提高性能.
- `rabbitmq_queue_base` - 指定一个队列名称的提示。这个设置的使用情况如下。
- `rabbitmq_deadletter_exchange` - 为**dead letter exchange**指定名称。你可以用这个 `exchange` 的名称创建另一个表，并在消息被重新发布到 `dead letter exchange` 的情况下收集它们。默认情况下，没有指定 `dead letter exchange`。Specify name for a **dead letter exchange**.
- `rabbitmq_persistent` - 如果设置为 `1 (true)`, 在插入查询中交付模式将被设置为 `2` (将消息标记为 '`persistent`'). 默认是: `0`.
- `rabbitmq_skip_broken_messages` – RabbitMQ 消息解析器对每块模式不兼容消息的容忍度。默认值：`0`. 如果 `rabbitmq_skip_broken_messages = N`，那么引擎将跳过 `N` 个无法解析的 RabbitMQ 消息（一条消息等于一行数据）。
- `rabbitmq_max_block_size`
- `rabbitmq_flush_interval_ms`

同时，格式的设置也可以与 `rabbitmq` 相关的设置一起添加。

示例：

```
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';
```

RabbitMQ 服务器配置应使用 ClickHouse 配置文件添加。

必要配置：

```
<rabbitmq>
<username>root</username>
<password>clickhouse</password>
</rabbitmq>
```

可选配置：

```
<rabbitmq>
<vhost>clickhouse</vhost>
</rabbitmq>
```

描述

`SELECT`对于读取消息不是特别有用（除了调试），因为每个消息只能读取一次。使用**物化视图**创建实时线程更为实用。要做到这一点：

1. 使用引擎创建一个 RabbitMQ 消费者，并将其视为一个数据流。
2. 创建一个具有所需结构的表。
3. 创建一个物化视图，转换来自引擎的数据并将其放入先前创建的表中。

当物化视图加入引擎时，它开始在后台收集数据。这允许您持续接收来自 RabbitMQ 的消息，并使用 SELECT 将它们转换为所需格式。

一个 RabbitMQ 表可以有多个你需要的物化视图。

数据可以根据rabbitmq_exchange_type和指定的rabbitmq_routing_key_list进行通道。

每个表不能多于一个 exchange。一个 exchange 可以在多个表之间共享 - 因为可以使用路由让数据同时进入多个表。

Exchange 类型的选项:

- `direct` - 路由是基于精确匹配的键。例如表的键列表: `key1,key2,key3,key4,key5`, 消息键可以是等同他们中的任意一个.
- `fanout` - 路由到所有的表 (exchange 名称相同的情况) 无论是什么键都是这样.
- `topic` - 路由是基于带有点分隔键的模式. 比如: `*.logs, records.*.*.2020, *.2018,*.2019,*.2020`.
- `headers` - 路由是基于`key=value`的匹配, 设置为`x-match=all`或`x-match=any`. 例如表的键列表: `x-match=all,format=logs,type=report,year=2020`.
- `consistent_hash` - 数据在所有绑定的表之间均匀分布 (exchange 名称相同的情况). 请注意, 这种 exchange 类型必须启用 RabbitMQ 插件: `rabbitmq-plugins enable rabbitmq_consistent_hash_exchange`.

设置`rabbitmq_queue_base`可用于以下情况:

- 来让不同的表共享队列, 这样就可以为同一个队列注册多个消费者, 这使得性能更好。如果使用`rabbitmq_num_consumers`和/或`rabbitmq_num_queues`设置, 在这些参数相同的情况下, 实现队列的精确匹配。
- 以便在不是所有消息都被成功消费时, 能够恢复从某些持久队列的阅读。要从一个特定的队列恢复消耗 - 在`rabbitmq_queue_base`设置中设置其名称, 不要指定`rabbitmq_num_consumers`和`rabbitmq_num_queues` (默认为 1)。要恢复所有队列的消费, 这些队列是为一个特定的表所声明的 - 只要指定相同的设置。`rabbitmq_queue_base, rabbitmq_num_consumers, rabbitmq_num_queues`。默认情况下, 队列名称对表来说是唯一的。
- 以重复使用队列, 因为它们被声明为持久的, 并且不会自动删除。可以通过任何 RabbitMQ CLI 工具删除)

为了提高性能, 收到的消息被分组为大小为 `max_insert_block_size` 的块。如果在`stream_flush_interval_ms`毫秒内没有形成数据块, 无论数据块是否完整, 数据都会被刷到表中。

如果`rabbitmq_num_consumers`和/或`rabbitmq_num_queues`设置与`rabbitmq_exchange_type`一起被指定, 那么:

- 必须启用`rabbitmq-consistent-hash-exchange` 插件.
- 必须指定已发布信息的 `message_id`属性 (对于每个信息/批次都是唯一的) 。

对于插入查询时有消息元数据, 消息元数据被添加到每个发布的消息中:`:messageID`和`republished`标志 (如果值为`true`, 则表示消息发布不止一次) - 可以通过消息头访问。

不要在插入和物化视图中使用同一个表。

示例:

```

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;

```

虚拟列

- `_exchange_name` - RabbitMQ exchange 名称.
- `_channel_id` - 接收消息的消费者所声明的频道ID.
- `_delivery_tag` - 收到消息的DeliveryTag. 以每个频道为范围.
- `_redelivered` - 消息的redelivered标志.
- `_message_id` - 收到的消息的ID；如果在消息发布时被设置，则为非空.
- `_timestamp` - 收到的消息的时间戳；如果在消息发布时被设置，则为非空.

PostgreSQL

PostgreSQL 引擎允许 ClickHouse 对存储在远程 PostgreSQL 服务器上的数据执行 `SELECT` 和 `INSERT` 查询.

创建一张表

```

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`]);

```

表结构可以与 PostgreSQL 源表结构不同：

- 列名应与 PostgreSQL 源表中的列名相同，但您可以按任何顺序使用其中的一些列。
- 列类型可能与源表中的列类型不同。 ClickHouse尝试将数值映射 到ClickHouse的数据类型。
- 设置 `external_table_functions_use_nulls` 来定义如何处理 Nullable 列. 默认值是 1, 当设置为 0 时 - 表函数将不会使用 nullable 列，而是插入默认值来代替 null. 这同样适用于数组数据类型中的 null 值.

引擎参数

- `host:port` — PostgreSQL 服务器地址.
- `database` — 数据库名称.
- `table` — 表名称.
- `user` — PostgreSQL 用户.

- `password` — 用户密码.
- `schema` — Non-default table schema. 可选.

实施细节

在 PostgreSQL 上的 `SELECT` 查询以 `COPY (SELECT ...)` TO STDOUT 的方式在只读 PostgreSQL 事务中运行，每次 `SELECT` 查询后提交。

简单的 `WHERE` 子句，如`=`，`!=`，`>`，`>=`，`<`，`<=`，和`IN`是在PostgreSQL 服务器上执行。

所有的连接、聚合、排序、`IN [array]`条件和`LIMIT`采样约束都是在 PostgreSQL 的查询结束后才在ClickHouse中执行的。

在 PostgreSQL 上的 `INSERT` 查询以 `COPY "table_name" (field1, field2, ... fieldN) FROM STDIN` 的方式在 PostgreSQL 事务中运行，每条 `INSERT` 语句后自动提交。

PostgreSQL 的 `Array` 类型会被转换为 ClickHouse 数组。

Note

要小心 - 一个在 PostgreSQL 中的数组数据，像`type_name[]`这样创建，可以在同一列的不同表行中包含不同维度的多维数组。但是在 ClickHouse 中，只允许在同一列的所有表行中包含相同维数的多维数组。

支持设置 PostgreSQL 字典源中 `Replicas` 的优先级。地图中的数字越大，优先级就越低。最高的优先级是 0。

在下面的例子中，副本example01-1有最高的优先级。

```
<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>
```

用法示例

PostgreSQL 中的表:

```

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    |      |    2 | test |
(1 row)

```

ClickHouse 中的表，从上面创建的 PostgreSQL 表中检索数据：

```

CREATE TABLE default.postgresql_table
(
  `float_nullable` Nullable(Float32),
  `str` String,
  `int_id` Int32
)
ENGINE = PostgreSQL('localhost:5432', 'public', 'test', 'postges_user', 'postgres_password');

```

```
SELECT * FROM postgresql_table WHERE str IN ('test');
```

float_nullable	str	int_id
NULL	test	1

使用非默认的模式：

```

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');

```

另请参阅

- 使用 PostgreSQL 作为外部字典的来源

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.

集成的表引擎

ClickHouse 提供了多种方式来与外部系统集成，包括表引擎。像所有其他的表引擎一样，使用CREATE TABLE或ALTER TABLE查询语句来完成配置。然后从用户的角度来看，配置的集成看起来像查询一个正常的表，但对它的查询是代理给外部系统的。这种透明的查询是这种方法相对于其他集成方法的主要优势之一，比如外部字典或表函数，它们需要在每次使用时使用自定义查询方法。

以下是支持的集成方式：

- [ODBC](#)
- [JDBC](#)
- [MySQL](#)
- [MongoDB](#)
- [HDFS](#)
- [S3](#)
- [Kafka](#)
- [EmbeddedRocksDB](#)
- [RabbitMQ](#)
- [PostgreSQL](#)

JDBC

允许CH通过 [JDBC](#) 连接到外部数据库。

要实现JDBC连接，CH需要使用以后台进程运行的程序 [clickhouse-jdbc-bridge](#)。

该引擎支持 [Nullable](#) 数据类型。

建表

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name
(
    columns list...
)
ENGINE = JDBC(datasource_uri, external_database, external_table)
```

引擎参数

- `datasource_uri` — 外部DBMS的URI或名字.

URI格式: `jdbc:<driver_name>://<host_name>:<port>/?user=<username>&password=<password>`.

MySQL示例: `jdbc:mysql://localhost:3306/?user=root&password=root`.

- `external_database` — 外部DBMS的数据库名.

- `external_table` — `external_database`中的外部表名或类似`select * from table1 where column1=1`的查询语句.

用法示例

通过mysql控制台客户端来创建表

Creating a table in MySQL server by connecting directly with its console client:

```
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)
```

在CH服务端创建表，并从中查询数据：

```
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
```

参见

- [JDBC表函数.](#)

ODBC

允许 ClickHouse 通过 [ODBC](#) 方式连接到外部数据库。

为了安全地实现 ODBC 连接，ClickHouse 使用了一个独立程序 `clickhouse-odbc-bridge`。如果 ODBC 驱动程序是直接从 `clickhouse-server` 中加载的，那么驱动问题可能会导致 ClickHouse 服务崩溃。当有需要时，ClickHouse 会自动启动 `clickhouse-odbc-bridge`。ODBC 桥梁程序与 `clickhouse-server` 来自相同的安装包。

该引擎支持 [Nullable](#) 数据类型。

创建表

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1],
    name2 [type2],
    ...
)
ENGINE = ODBC(connection_settings, external_database, external_table)
```

详情请见 [CREATE TABLE](#) 查询。

表结构可以与源表结构不同：

- 列名应与源表中的列名相同，但您可以按任何顺序使用其中的一些列。
- 列类型可能与源表中的列类型不同。 ClickHouse尝试将数值[映射](#)到ClickHouse的数据类型。
- 设置 `external_table_functions_use_nulls` 来定义如何处理 Nullable 列. 默认值是 `true`, 当设置为 `false` 时 - 表函数将不会使用 nullable 列，而是插入默认值来代替 `null`. 这同样适用于数组数据类型中的 `null` 值.

引擎参数

- `connection_settings` — 在 `odbc.ini` 配置文件中，连接配置的名称.
- `external_database` — 在外部 DBMS 中的数据库名.
- `external_table` — `external_database`中的表名.

用法示例

通过ODBC从本地安装的MySQL中检索数据

本示例已经在 Ubuntu Linux 18.04 和 MySQL server 5.7 上测试通过。

请确保已经安装了 unixODBC 和 MySQL Connector。

默认情况下（如果从软件包安装），ClickHouse以用户`clickhouse`的身份启动. 因此，您需要在MySQL服务器中创建并配置此用户。

```
$ sudo mysql
```

```
mysql> CREATE USER 'clickhouse'@'localhost' IDENTIFIED BY 'clickhouse';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'clickhouse'@'clickhouse' WITH GRANT OPTION;
```

然后在`/etc/odbc.ini`中配置连接。

```
$ cat /etc/odbc.ini
[mysqlconn]
DRIVER = /usr/local/lib/libmyodbc5w.so
SERVER = 127.0.0.1
PORT = 3306
DATABASE = test
USERNAME = clickhouse
PASSWORD = clickhouse
```

您可以从安装的 `unixodbc` 中使用 `isql` 实用程序来检查连接情况。

```
$ isql -v mysqlconn
+-----+
| Connected! |
|           |
... 
```

MySQL中的表:

```
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中的表，从MySQL表中检索数据:

```
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

另请参阅

- [ODBC 外部字典](#)
- [ODBC 表函数](#)

HDFS

这个引擎提供了与 [Apache Hadoop](#) 生态系统的集成，允许通过 ClickHouse 管理 [HDFS](#) 上的数据。这个引擎类似于 [文件](#) 和 [URL](#) 引擎，但提供了 Hadoop 的特定功能。

用法

```
ENGINE = HDFS(URI, format)
```

URI 参数是 HDFS 中整个文件的 URI。

format 参数指定一种可用的文件格式。执行

SELECT 查询时，格式必须支持输入，以及执行

INSERT 查询时，格式必须支持输出。你可以在 [格式](#) 章节查看可用的格式。

路径部分 **URI** 可能包含 **glob** 通配符。在这种情况下，表将是只读的。

示例：

1. 设置 `hdfs_engine_table` 表：

```
CREATE TABLE hdfs_engine_table (name String, value UInt32) ENGINE=HDFS('hdfs://hdfs1:9000/other_storage', 'TSV')
```

2. 填充文件：

```
INSERT INTO hdfs_engine_table VALUES ('one', 1), ('two', 2), ('three', 3)
```

3. 查询数据：

```
SELECT * FROM hdfs_engine_table LIMIT 2
```

name	value
one	1
two	2

实施细节

- 读取和写入可以并行
- 不支持：
 - `ALTER` 和 `SELECT...SAMPLE` 操作。
 - 索引。
 - 复制。

路径中的通配符

多个路径组件可以具有 **globs**。对于正在处理的文件应该存在并匹配到整个路径模式。文件列表的确定是在 **SELECT** 的时候进行（而不是在 **CREATE** 的时候）。

- `*` — 替代任何数量的任何字符，除了 `/` 以及空字符串。
- `?` — 代替任何单个字符。
- `{some_string,another_string,yet_another_one}` — 替代任何字符串 `'some_string'`, `'another_string'`, `'yet_another_one'`.
- `{N..M}` — 替换 N 到 M 范围内的任何数字，包括两个边界的值。

带 `{}` 的结构类似于 [远程](#) 表函数。

示例

1. 假设我们在 HDFS 上有几个 TSV 格式的文件，文件的 URI 如下：

- `'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. 有几种方法可以创建由所有六个文件组成的表：

```
CREATE TABLE table_with_range (name String, value UInt32) ENGINE =  
HDFS('hdfs://hdfs1:9000/{some,another}_dir/some_file_{1..3}', 'TSV')
```

另一种方式：

```
CREATE TABLE table_with_question_mark (name String, value UInt32) ENGINE =  
HDFS('hdfs://hdfs1:9000/{some,another}_dir/some_file ?', 'TSV')
```

表由两个目录中的所有文件组成（所有文件都应满足query中描述的格式和模式）：

```
CREATE TABLE table_with_asterisk (name String, value UInt32) ENGINE =  
HDFS('hdfs://hdfs1:9000/{some,another}_dir/*', 'TSV')
```

警告

如果文件列表包含带有前导零的数字范围，请单独使用带有大括号的构造或使用 `?`。

示例

创建具有名为文件的表 `file000, file001, ... , file999`：

```
CREARE TABLE big_table (name String, value UInt32) ENGINE = HDFS('hdfs://hdfs1:9000/big_dir/file{0..9}{0..9}  
{0..9}', 'CSV')
```

配置

与 GraphiteMergeTree 类似，HDFS 引擎支持使用 ClickHouse 配置文件进行扩展配置。有两个配置键可以使用：全局 (`hdfs`) 和用户级别 (`hdfs_*`)。首先全局配置生效，然后用户级别配置生效（如果用户级别配置存在）。

```
<!-- HDFS 引擎类型的全局配置选项 -->  
<hdfs>  
  <hadoop_kerberos_keytab>/tmp/keytab/clickhouse.keytab</hadoop_kerberos_keytab>  
  <hadoop_kerberos_principal>clickuser@TEST.CLICKHOUSE.TECH</hadoop_kerberos_principal>  
  <hadoop_security_authentication>kerberos</hadoop_security_authentication>  
</hdfs>  
  
<!-- 用户 "root" 的指定配置 -->  
<hdfs_root>  
  <hadoop_kerberos_principal>root@TEST.CLICKHOUSE.TECH</hadoop_kerberos_principal>  
</hdfs_root>
```

可选配置选项及其默认值的列表

libhdfs3 支持的

参数	默认值
rpc_client_connect_tcpnodelay	true
dfs_client_read_shortcircuit	true
output_replace-datanode-on-failure	true
input_notretry-another-node	false
input_localread_mappedfile	true
dfs_client_use_legacy_blockreader_local	false
rpc_client_ping_interval	10 * 1000
rpc_client_connect_timeout	600 * 1000
rpc_client_read_timeout	3600 * 1000
rpc_client_write_timeout	3600 * 1000
rpc_client_socekt_linger_timeout	-1
rpc_client_connect_retry	10
rpc_client_timeout	3600 * 1000
dfs_default_replica	3
input_connect_timeout	600 * 1000
input_read_timeout	3600 * 1000
input_write_timeout	3600 * 1000
input_localread_default_buffersize	1 * 1024 * 1024
dfs_prefetchsize	10
input_read_getblockinfo_retry	3
input_localread_blockinfo_cachesize	1000
input_read_max_retry	60
output_default_chunksize	512
output_default_packetsize	64 * 1024
output_default_write_retry	10
output_connect_timeout	600 * 1000
output_read_timeout	3600 * 1000
output_write_timeout	3600 * 1000
output_close_timeout	3600 * 1000
output_packetpool_size	1024
output_heeartbeat_interval	10 * 1000
dfs_client_failover_max_attempts	15
dfs_client_read_shortcircuit_streams_cache_size	256
dfs_client_socketcache_expiryMsec	3000
dfs_client_socketcache_capacity	16
dfs_default_blocksize	64 * 1024 * 1024
dfs_default_uri	"hdfs://localhost:9000"
hadoop_security_authentication	"simple"
hadoop_security_kerberos_ticket_cache_path	""
dfs_client_log_severity	"INFO"
dfs_domain_socket_path	""

[HDFS 配置参考](#) 也许会解释一些参数的含义。

ClickHouse 额外的配置

参数	默认值
hadoop_kerberos_keytab	""
hadoop_kerberos_principal	""
hadoop_kerberos_kinit_command	kinit

限制

- hadoop_security_kerberos_ticket_cache_path 只能在全局配置，不能指定用户

Kerberos 支持

如果 `hadoop_security_authentication` 参数的值为 'kerberos' , ClickHouse 将通过 Kerberos 设施进行认证。

这里的 [参数](#) 和 `hadoop_security_kerberos_ticket_cache_path` 也许会有帮助。

注意，由于 libhdfs3 的限制，只支持老式的方法。

数据节点的安全通信无法由 SASL 保证 (`HADOOP_SECURE_DN_USER` 是这种安全方法的一个可靠指标)

使用 `tests/integration/test_storage_kerberized_hdfs/hdfs_configs/bootstrap.sh` 脚本作为参考。

如果指定了 `hadoop_kerberos_keytab`, `hadoop_kerberos_principal` 或者 `hadoop_kerberos_kinit_command` , 将会调用 `kinit` 工具.在此情况下，`hadoop_kerberos_keytab` 和 `hadoop_kerberos_principal` 参数是必须配置的. `kinit` 工具和 `krb5` 配置文件是必要的.

虚拟列

- `_path` — 文件路径.

- `_file` — 文件名.

另请参阅

- [虚拟列](#)

Kafka

此引擎与 [Apache Kafka](#) 结合使用。

Kafka 特性：

- 发布或者订阅数据流。
- 容错存储机制。
- 处理流数据。

老版格式：

```
Kafka(kafka_broker_list, kafka_topic_list, kafka_group_name, kafka_format  
[, kafka_row_delimiter, kafka_schema, kafka_num_consumers])
```

新版格式：

```
Kafka SETTINGS  
kafka_broker_list = 'localhost:9092',  
kafka_topic_list = 'topic1,topic2',  
kafka_group_name = 'group1',  
kafka_format = 'JSONEachRow',  
kafka_row_delimiter = '\n',  
kafka_schema = "",  
kafka_num_consumers = 2
```

必要参数：

- `kafka_broker_list` – 以逗号分隔的 brokers 列表 (`localhost:9092`)。
- `kafka_topic_list` – topic 列表 (`my_topic`)。
- `kafka_group_name` – Kafka 消费组名称 (`group1`)。如果不希望消息在集群中重复，请在每个分片中使用相同的组名。
- `kafka_format` – 消息体格式。使用与 SQL 部分的 `FORMAT` 函数相同表示方法，例如 `JSONEachRow`。了解详细信息，请参考 [Formats](#) 部分。

可选参数：

- `kafka_row_delimiter` - 每个消息体（记录）之间的分隔符。
- `kafka_schema` - 如果解析格式需要一个 `schema` 时，此参数必填。例如，**普罗托船长** 需要 `schema` 文件路径以及根对象 `schema.capnp:Message` 的名字。
- `kafka_num_consumers` - 单个表的消费者数量。默认值是：1，如果一个消费者的吞吐量不足，则指定更多的消费者。消费者的总数不应该超过 `topic` 中分区的数量，因为每个分区只能分配一个消费者。

示例：

```
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;
```

消费的消息会被自动追踪，因此每个消息在不同的消费组里只会记录一次。如果希望获得两次数据，则使用另一个组名创建副本。

消费组可以灵活配置并且在集群之间同步。例如，如果群集中有10个主题和5个表副本，则每个副本将获得2个主题。如果副本数量发生变化，主题将自动在副本中重新分配。了解更多信息请访问 <http://kafka.apache.org/intro>。

`SELECT` 查询对于读取消息并不是很有用（调试除外），因为每条消息只能被读取一次。使用物化视图创建实时线程更实用。您可以这样做：

1. 使用引擎创建一个 Kafka 消费者并作为一条数据流。
2. 创建一个结构表。
3. 创建物化视图，该视图会在后台转换引擎中的数据并将其放入之前创建的表中。

当 `MATERIALIZED VIEW` 添加至引擎，它将会在后台收集数据。可以持续不断地从 Kafka 收集数据并通过 `SELECT` 将数据转换为所需要的格式。

示例：

```
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;
```

为了提高性能，接受的消息被分组为 `max_insert_block_size` 大小的块。如果未在 `stream_flush_interval_ms` 毫秒内形成块，则不关心块的完整性，都会将数据刷新到表中。

停止接收主题数据或更改转换逻辑，请 `detach` 物化视图：

```
DETACH TABLE consumer;
ATTACH TABLE consumer;
```

如果使用 `ALTER` 更改目标表，为了避免目标表与视图中的数据之间存在差异，推荐停止物化视图。

配置

与 `GraphiteMergeTree` 类似，`Kafka` 引擎支持使用 `ClickHouse` 配置文件进行扩展配置。可以使用两个配置键：全局 (`kafka`) 和 主题级别 (`kafka_*`)。首先应用全局配置，然后应用主题级配置（如果存在）。

```
<!-- 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>
```

有关详细配置选项列表，请参阅 [librdkafka 配置参考](#)。在 `ClickHouse` 配置中使用下划线 (`_`)，并不是使用点 (`.`)。例如，`check.crcs=true` 将是 `<check_crcs>true</check_crcs>`。

MySQL

`MySQL` 引擎可以对存储在远程 MySQL 服务器上的数据执行 `SELECT` 查询。

调用格式：

```
MySQL('host:port', 'database', 'table', 'user', 'password'[, replace_query, 'on_duplicate_clause']);
```

调用参数

- `host:port` — MySQL 服务器地址。
- `database` — 数据库的名称。

- `table` — 表名称。
- `user` — 数据库用户。
- `password` — 用户密码。
- `replace_query` — 将 `INSERT INTO` 查询是否替换为 `REPLACE INTO` 的标志。如果 `replace_query=1`，则替换查询。
- `'on_duplicate_clause'` — 将 `ON DUPLICATE KEY UPDATE 'on_duplicate_clause'` 表达式添加到 `INSERT` 查询语句中。例如：`impression = VALUES(impression) + impression`。如果需要指定 `'on_duplicate_clause'`，则需要设置 `replace_query=0`。如果同时设置 `replace_query = 1` 和 `'on_duplicate_clause'`，则会抛出异常。

此时，简单的 `WHERE` 子句（例如 `=, !=, >, >=, <, <=`）是在 MySQL 服务器上执行。

其余条件以及 `LIMIT` 采样约束语句仅在对 MySQL 的查询完成后才在 ClickHouse 中执行。

MySQL 引擎不支持 **可为空** 数据类型，因此，当从 MySQL 表中读取数据时，`NULL` 将转换为指定列类型的默认值（通常为 0 或空字符串）。

关联表引擎

使用 **JOIN** 操作的一种可选的数据结构。

Note

该文档和 **JOIN 语句** 无关。

建表语句

```
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, ...])
```

建表语句详情参见 [创建表](#)。

引擎参数

- `join_strictness` – **JOIN 限制**。
- `join_type` – **JOIN 类型**。
- `k1[, k2, ...]` – 进行 `JOIN` 操作时 `USING` 语句用到的 key 列

使用 `join_strictness` 和 `join_type` 参数时不需要用引号，例如，`Join(ANY, LEFT, col1)`。这些参数必须和进行 `join` 操作的表相匹配。否则，CH 不会报错，但是可能返回错误的数据。

表用法

示例

创建左关联表：

```
CREATE TABLE id_val(`id` UInt32, `val` UInt32) ENGINE = TinyLog
```

```
INSERT INTO id_val VALUES (1,11)(2,12)(3,13)
```

创建 Join 右边的表：

```
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)
```

表关联：

```
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

作为一种替换方式，可以从 Join 表获取数据，需要设置好 join 的 key 字段值。

```
SELECT joinGet('id_val_join', 'val', toUInt32(1))
```

```
joinGet('id_val_join', 'val', toUInt32(1))
```

```
21 |
```

数据查询及插入

可以使用 `INSERT` 语句向 Join 引擎表中添加数据。如果表是通过指定 `ANY` 限制参数来创建的，那么重复 key 的数据会被忽略。指定 `ALL` 限制参数时，所有行记录都会被添加进去。

不能通过 `SELECT` 语句直接从表中获取数据。请使用下面的方式：

- 将表放在 `JOIN` 的右边进行查询
- 调用 `joinGet` 函数，就像从字典中获取数据一样来查询表。

使用限制及参数设置

创建表时，会应用下列设置参数：

- `join_use_nulls`
- `max_rows_in_join`
- `max_bytes_in_join`
- `join_overflow_mode`
- `join_any_take_last_row`

Join 表不能在 `GLOBAL JOIN` 操作中使用

Join 表创建及 `查询` 时，允许使用 `join_use_nulls` 参数。如果使用不同的 `join_use_nulls` 设置，会导致表关联异常（取决于 join 的类型）。当使用函数 `joinGet` 时，请在建表和查询语句中使用相同的 `join_use_nulls` 参数设置。

数据存储

`Join`表的数据总是保存在内存中。当往表中插入行记录时，`CH`会将数据块保存在硬盘目录中，这样服务器重启时数据可以恢复。

如果服务器非正常重启，保存在硬盘上的数据块会丢失或被损坏。这种情况下，需要手动删除被损坏的数据文件。

随机数生成表引擎

随机数生成表引擎为指定的表模式生成随机数

使用示例：

- 测试时生成可复写的大表
- 为复杂测试生成随机输入

CH服务端的用法

```
ENGINE = GenerateRandom(random_seed, max_string_length, max_array_length)
```

生成数据时，通过`max_array_length` 设置array列的最大长度，`max_string_length`设置string数据的最大长度

该引擎仅支持 `SELECT` 查询语句。

该引擎支持能在表中存储的所有数据类型 [DataTypes](#)，除了 `LowCardinality` 和 `AggregateFunction`.

示例

1. 设置 `generate_engine_table` 引擎表：

```
CREATE TABLE generate_engine_table (name String, value UInt32) ENGINE = GenerateRandom(1, 5, 3)
```

2. 查询数据：

```
SELECT * FROM generate_engine_table LIMIT 3
```

name	value
c4xj	1412771199
r	1791099446
7#\$	124312908

实现细节

- 以下特性不支持：
 - `ALTER`
 - `SELECT ... SAMPLE`
 - `INSERT`
 - `Indices`
 - `Replication`

MaterializedView

物化视图的使用（更多信息请参阅 [CREATE TABLE](#)）。它需要使用一个不同的引擎来存储数据，这个引擎要在创建物化视图时指定。当从表中读取时，它就会使用该引擎。

Null

当写入 Null 类型的表时，将忽略数据。从 Null 类型的表中读取时，返回空。

但是，可以在 Null 类型的表上创建物化视图。写入表的数据将转发到视图中。

URL(URL, 格式)

用于管理远程 HTTP/HTTPS 服务器上的数据。该引擎类似文件引擎。

在 ClickHouse 服务器中使用引擎

Format 必须是 ClickHouse 可以用于

`SELECT` 查询的一种格式，若有必要，还要可用于 `INSERT`。有关支持格式的完整列表，请查看格式。

`URL` 必须符合统一资源定位符的结构。指定的 URL 必须指向一个 HTTP 或 HTTPS 服务器。对于服务端响应，不需要任何额外的 HTTP 头标记。

`INSERT` 和 `SELECT` 查询会分别转换为 `POST` 和 `GET` 请求。

对于 `POST` 请求的处理，远程服务器必须支持分块传输编码。

示例：

1. 在 Clickhouse 服务上创建一个 `url_engine_table` 表：

```
CREATE TABLE url_engine_table (word String, value UInt64)
ENGINE=URL('http://127.0.0.1:12345/', CSV)
```

2. 用标准的 Python 3 工具库创建一个基本的 HTTP 服务并启动它：

```
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. 查询请求：

```
SELECT * FROM url_engine_table
```

word	value
Hello	1
World	2

功能实现

- 读写操作都支持并发
- 不支持：
 - ALTER 和 SELECT...SAMPLE 操作。
 - 索引。
 - 副本。

内存表

Memory 引擎以未压缩的形式将数据存储在 RAM 中。数据完全以读取时获得的形式存储。换句话说，从这张表中读取是很轻松的。并发数据访问是同步的。锁范围小：读写操作不会相互阻塞。不支持索引。查询是并行化的。在简单查询上达到最大速率（超过 10 GB /秒），因为没有磁盘读取，不需要解压缩或反序列化数据。（值得注意的是，在许多情况下，与 MergeTree 引擎的性能几乎一样高）。重新启动服务器时，表中的数据消失，表将变为空。通常，使用此表引擎是不合理的。但是，它可用于测试，以及在相对较少的行（最多约 100,000,000）上需要最高性能的查询。

Memory 引擎是由系统用于临时表进行外部数据的查询（请参阅《外部数据用于请求处理》部分），以及用于实现 GLOBAL IN（请参见《IN 运算符》部分）。

分布

分布式引擎本身不存储数据，但可以在多个服务器上进行分布式查询。

读是自动并行的。读取时，远程服务器表的索引（如果有的话）会被使用。

分布式引擎参数：服务器配置文件中的集群名，远程数据库名，远程表名，数据分片键（可选）。

示例：

```
Distributed(logs, default, hits[, sharding_key])
```

将会从位于《logs》集群中 default.hits 表所有服务器上读取数据。

远程服务器不仅用于读取数据，还会对尽可能数据做部分处理。

例如，对于使用 GROUP BY 的查询，数据首先在远程服务器聚合，之后返回聚合函数的中间状态给查询请求的服务器。再在请求的服务器上进一步汇总数据。

数据库名参数除了用数据库名之外，也可用返回字符串的常量表达式。例如：currentDatabase()。

logs – 服务器配置文件中的集群名称。

集群示例配置如下：

```

<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>

```

这里定义了一个名为'logs'的集群，它由两个分片组成，每个分片包含两个副本。

分片是指包含数据不同部分的服务器（要读取所有数据，必须访问所有分片）。

副本是存储复制数据的服务器（要读取所有数据，访问任一副本上的数据即可）。

集群名称不能包含点号。

每个服务器需要指定 host，port，和可选的 user，password，secure，compression 的参数：

- **host** – 远程服务器地址。可以域名、IPv4或IPv6。如果指定域名，则服务在启动时发起一个 DNS 请求，并且请求结果会在服务器运行期间一直被记录。如果 DNS 请求失败，则服务不会启动。如果你修改了 DNS 记录，则需要重启服务。

- **port** – 消息传递的 TCP 端口（「tcp_port」配置通常设为 9000）。不要跟 http_port 混淆。

- **user** – 用于连接远程服务器的用户名。默认值：default。该用户必须有权限访问该远程服务器。访问权限配置在 users.xml 文件中。更多信息，请查看《访问权限》部分。

- **password** – 用于连接远程服务器的密码。默认值：空字符串。

- **secure** – 是否使用ssl进行连接，设为true时，通常也应该设置 port = 9440。服务器也要监听

<tcp_port_secure>9440</tcp_port_secure> 并有正确的证书。

- **compression** – 是否使用数据压缩。默认值：true。

配置了副本，读取操作会从每个分片里选择一个可用的副本。可配置负载平衡算法（挑选副本的方式） - 请参阅《load_balancing》设置。

如果跟服务器的连接不可用，则在尝试短超时的重连。如果重连失败，则选择下一个副本，依此类推。如果跟所有副本的连接尝试都失败，则尝试用相同的方式再重复几次。

该机制有利于系统可用性，但不保证完全容错：如有远程服务器能够接受连接，但无法正常工作或状况不佳。

你可以配置一个（这种情况下，查询操作更应该称为远程查询，而不是分布式查询）或任意多个分片。在每个分片中，可以配置一个或任意多个副本。不同分片可配置不同数量的副本。

可以在配置中配置任意数量的集群。

要查看集群，可使用《system.clusters》表。

通过分布式引擎可以像使用本地服务器一样使用集群。但是，集群不是自动扩展的：你必须编写集群配置到服务器配置文件中（最好，给所有集群的服务器写上完整配置）。

不支持用分布式表查询别的分布式表（除非该表只有一个分片）。或者说，要用分布表查查询《最终》的数据表。

分布式引擎需要将集群信息写入配置文件。配置文件中的集群信息会即时更新，无需重启服务器。如果你每次是要向不确定的一组分片和副本发送查询，则不适合创建分布式表 - 而应该使用《远程》表函数。请参阅《表函数》部分。

向集群写数据的方法有两种：

一，自己指定要将哪些数据写入哪些服务器，并直接在每个分片上执行写入。换句话说，在分布式表上《查询》，在数据表上 **INSERT**。

这是最灵活的解决方案 - 你可以使用任何分片方案，对于复杂业务特性的需求，这可能是非常重要的。

这也是最佳解决方案，因为数据可以完全独立地写入不同的分片。

二，在分布式表上执行 **INSERT**。在这种情况下，分布式表会跨服务器分发插入数据。

为了写入分布式表，必须要配置分片键（最后一个参数）。当然，如果只有一个分片，则写操作在没有分片键的情况下也能工作，因为这种情况下分片键没有意义。

每个分片都可以在配置文件中定义权重。默认情况下，权重等于1。数据依据分片权重按比例分发到分片上。例如，如果有两个分片，第一个分片的权重是9，而第二个分片的权重是10，则发送 9 / 19 的行到第一个分片，10 / 19 的行到第二个分片。

分片可在配置文件中定义 ‘internal_replication’ 参数。

此参数设置为《true》时，写操作只选一个正常的副本写入数据。如果分布式表的子表是复制表(*ReplicaMergeTree)，请使用此方案。换句话说，这其实是把数据的复制工作交给实际需要写入数据的表本身而不是分布式表。

若此参数设置为《false》（默认值），写操作会将数据写入所有副本。实质上，这意味着要分布式表本身来复制数据。这种方式不如使用复制表的好，因为不会检查副本的一致性，并且随着时间的推移，副本数据可能会有些不一样。

选择将一行数据发送到哪个分片的方法是，首先计算分片表达式，然后将这个计算结果除以所有分片的权重总和得到余数。该行会发送到那个包含该余数的从‘prev_weight’到‘prev_weights + weight’的半闭半开区间对应的分片上，其中‘prev_weights’ 是该分片前面的所有分片的权重和，‘weight’ 是该分片的权重。例如，如果有两个分片，第一个分片权重为9，而第二个分片权重为10，则余数在 [0,9) 中的行发给第一个分片，余数在 [9,19) 中的行发给第二个分片。

分片表达式可以是由常量和表列组成的任何返回整数表达式。例如，您可以使用表达式 ‘rand()’ 来随机分配数据，或者使用 ‘UserID’ 来按用户 ID 的余数分布（相同用户的数据将分配到单个分片上，这可降低带有用户信息的 IN 和 JOIN 的语句运行的复杂度）。如果该列数据分布不够均匀，可以将其包装在散列函数中：intHash64(UserID)。

这种简单的用余数来选择分片的方案是有局限的，并不总适用。它适用于中型和大型数据（数十台服务器）的场景，但不适用于巨量数据（数百台或更多服务器）的场景。后一种情况下，应根据业务特性需求考虑的分片方案，而不是直接用分布式表的多分片。

SELECT 查询会被发送到所有分片，并且无论数据在分片中如何分布（即使数据完全随机分布）都可正常工作。添加新分片时，不必将旧数据传输到该分片。你可以给新分片分配大权重然后写新数据 - 数据可能会稍分布不均，但查询会正确高效地运行。

下面的情况，你需要关注分片方案：

- 使用需要特定键连接数据（IN 或 JOIN）的查询。如果数据是用该键进行分片，则应使用本地 IN 或 JOIN 而不是 GLOBAL IN 或 GLOBAL JOIN，这样效率更高。
- 使用大量服务器（上百或更多），但有大量小查询（个别客户的查询 - 网站，广告商或合作伙伴）。为了使小查询不影响整个集群，让单个客户的数据处于单个分片上是有意义的。或者，正如我们在 Yandex.Metrica 中所做的那样，你可以配置两级分片：将整个集群划分为《层》，一个层可以包含多个分片。单个客户的数据位于单个层上，根据需要将分片添加到层中，层中的数据随机分布。然后给每层创建分布式表，再创建一个全局的分布式表用于全局的查询。

数据是异步写入的。对于分布式表的 **INSERT**，数据块只写本地文件系统。之后会尽快地在后台发送到远程服务器。你可以通过查看表目录中的文件列表（等待发送的数据）来检查数据是否成功发送：`/var/lib/clickhouse/data/database/table/`。

如果在 `INSERT` 到分布式表时服务器节点丢失或重启（如，设备故障），则插入的数据可能会丢失。如果在表目录中检测到损坏的数据分片，则会将其转移到«`broken`»子目录，并不再使用。

启用 `max_parallel_replicas` 选项后，会在分表的所有副本上并行查询处理。更多信息，请参阅«设置，`max_parallel_replicas`»部分。

合并

`Merge` 引擎（不要跟 `MergeTree` 引擎混淆）本身不存储数据，但可用于同时从任意多个其他的表中读取数据。

读是自动并行的，不支持写入。读取时，那些被真正读取到数据的表的索引（如果有的话）会被使用。

`Merge` 引擎的参数：一个数据库名和一个用于匹配表名的正则表达式。

示例：

```
Merge(hits, '^WatchLog')
```

数据会从 `hits` 数据库中表名匹配正则 '`^WatchLog`' 的表中读取。

除了数据库名，你也可以用一个返回字符串的常量表达式。例如，`currentDatabase()`。

正则表达式 — `re2`（支持 PCRE 一个子集的功能），大小写敏感。

了解关于正则表达式中转义字符的说明可参看 «`match`» 一节。

当选择需要读的表时，`Merge` 表本身会被排除，即使它匹配上了该正则。这样设计为了避免循环。

当然，是能够创建两个相互无限递归读取对方数据的 `Merge` 表的，但这并没有什么意义。

`Merge` 引擎的一个典型应用是可以像使用一张表一样使用大量的 `TinyLog` 表。

示例 2：

我们假定你有一个旧表（`WatchLog_old`），你想改变数据分区了，但又不想把旧数据转移到新表（`WatchLog_new`）里，并且你需要同时能看到这两个表的数据。

```
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	UserId	EventType	Cnt
2018-01-01	1	hit	3

date	UserId	EventType	Cnt
2018-01-02	2	hit	3

虚拟列

虚拟列是一种由表引擎提供而不是在表定义中的列。换种说法就是，这些列并没有在 `CREATE TABLE` 中指定，但可以在 `SELECT` 中使用。

下面列出虚拟列跟普通列的不同点：

- 虚拟列不在表结构定义里指定。

- 不能用 `INSERT` 向虚拟列写数据。
- 使用不指定列名的 `INSERT` 语句时，虚拟列要会被忽略掉。
- 使用星号通配符（`SELECT *`）时虚拟列不会包含在里面。
- 虚拟列不会出现在 `SHOW CREATE TABLE` 和 `DESC TABLE` 的查询结果里。

`Merge` 类型的表包括一个 `String` 类型的 `_table` 虚拟列。（如果该表本来已有了一个 `_table` 的列，那这个虚拟列会命名为 `_table1`；如果 `_table1` 也本就存在了，那这个虚拟列会被命名为 `_table2`，依此类推）该列包含被读数据的表名。

如果 `WHERE/PREWHERE` 子句包含了带 `_table` 的条件，并且没有依赖其他的列（如作为表达式谓词链接的一个子项或作为整个的表达式），这些条件的作用会像索引一样。这些条件会在那些可能被读数据的表的表名上执行，并且读操作只会在那些满足了该条件的表上去执行。

字典

`Dictionary` 引擎将字典数据展示为一个 ClickHouse 的表。

例如，考虑使用一个具有以下配置的 `products` 字典：

```
<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>
```

查询字典中的数据：

```
select name, type, key, attribute.names, attribute.types, bytes_allocated, element_count, source from
system.dictionaries where name = 'products';

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

你可以使用 [dictGet*](#) 函数来获取这种格式的字典数据。

当你需要获取原始数据，或者是想要使用 `JOIN` 操作的时候，这种视图并没有什么帮助。对于这些情况，你可以使用 `Dictionary` 引擎，它可以将字典数据展示在表中。

语法：

```
CREATE TABLE %table_name% (%fields%) engine = Dictionary(%dictionary_name%)`
```

示例：

```
create table products (product_id UInt64, title String) Engine = Dictionary(products);
CREATE TABLE products
(
    product_id UInt64,
    title String,
)
ENGINE = Dictionary(products)
```

Ok.

0 rows in set. Elapsed: 0.004 sec.

看一看表中的内容。

```
select * from products limit 1;
```

```
SELECT *
FROM products
LIMIT 1
```

product_id	title
152689	Some item

1 rows in set. Elapsed: 0.006 sec.

文件(输入格式)

数据源是以 Clickhouse 支持的一种输入格式（TabSeparated，Native等）存储数据的文件。

用法示例：

- 从 ClickHouse 导出数据到文件。
- 将数据从一种格式转换为另一种格式。
- 通过编辑磁盘上的文件来更新 ClickHouse 中的数据。

在 ClickHouse 服务器中的使用

File(Format)

选用的 `Format` 需要支持 `INSERT` 或 `SELECT`。有关支持格式的完整列表，请参阅 [格式](#)。

`ClickHouse` 不支持给 `File` 指定文件系统路径。它使用服务器配置中 `路径` 设定的文件夹。

使用 `File(Format)` 创建表时，它会在该文件夹中创建空的子目录。当数据写入该表时，它会写到该子目录中的 `data.Format` 文件中。

你也可以在服务器文件系统中手动创建这些子文件夹和文件，然后通过 `ATTACH` 将其创建为具有对应名称的表，这样你就可以从该文件中查询数据了。

注意

注意这个功能，因为 `ClickHouse` 不会跟踪这些文件在外部的更改。在 `ClickHouse` 中和 `ClickHouse` 外部同时写入会造成结果是不确定的。

示例：

1. 创建 `file_engine_table` 表：

```
CREATE TABLE file_engine_table (name String, value UInt32) ENGINE=File(TabSeparated)
```

默认情况下，`Clickhouse` 会创建目录 `/var/lib/clickhouse/data/default/file_engine_table`。

2. 手动创建 `/var/lib/clickhouse/data/default/file_engine_table/data.TabSeparated` 文件，并且包含内容：

```
$ cat data.TabSeparated
one 1
two 2
```

3. 查询这些数据：

```
SELECT * FROM file_engine_table
```

name	value
one	1
two	2

在 `Clickhouse-local` 中的使用

使用 `clickhouse-local` 时，`File` 引擎除了 `Format` 之外，还可以接收文件路径参数。可以使用数字或名称来指定标准输入/输出流，例如 `0` 或 `stdin`，`1` 或 `stdout`。

例如：

```
$ 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"
```

功能实现

- 读操作可支持并发，但写操作不支持

- 不支持：
 - ALTER
 - SELECT ... SAMPLE
 - 索引
 - 副本

用于查询处理的外部数据

ClickHouse 允许向服务器发送处理查询所需的数据以及 SELECT 查询。这些数据放在一个临时表中（请参阅《临时表》一节），可以在查询中使用（例如，在 IN 操作符中）。

例如，如果您有一个包含重要用户标识符的文本文件，则可以将其与使用此列表过滤的查询一起上传到服务器。

如果需要使用大量外部数据运行多个查询，请不要使用该特性。最好提前把数据上传到数据库。

可以使用命令行客户端（在非交互模式下）或使用 HTTP 接口上传外部数据。

在命令行客户端中，您可以指定格式的参数部分

```
--external --file=... [--name=...] [--format=...] [--types=...]|--structure=...]
```

对于传输的表的数量，可能有多个这样的部分。

-external – 标记子句的开始。

-file – 带有表存储的文件的路径，或者，它指的是STDIN。

只能从 stdin 中检索单个表。

以下的参数是可选的：**-name** – 表的名称，如果省略，则采用 _data。

-format – 文件中的数据格式。如果省略，则使用 TabSeparated。

以下的参数必选一个：**-types** – 逗号分隔列类型的列表。例如：UInt64,String。列将被命名为 _1, _2, ...

-structure – 表结构的格式 UserID UInt64, URL String。定义列的名字以及类型。

在 «file» 中指定的文件将由 «format» 中指定的格式解析，使用在 «types» 或 «structure» 中指定的数据类型。该表将被上传到服务器，并在作为名称为 «name» 临时表。

示例：

```
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
```

当使用HTTP接口时，外部数据以 multipart/form-data 格式传递。每个表作为一个单独的文件传输。表名取自文件名。«query_string» 传递参数 «name_format»、«name_types» 和 «name_structure»，其中 «name» 是这些参数对应的表的名称。参数的含义与使用命令行客户端时的含义相同。

示例：

```
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
```

对于分布式查询，将临时表发送到所有远程服务器。

缓冲区

缓冲数据写入 RAM 中，周期性地将数据刷新到另一个表。在读取操作时，同时从缓冲区和另一个表读取数据。

```
Buffer(database, table, num_layers, min_time, max_time, min_rows, max_rows, min_bytes, max_bytes)
```

引擎的参数：`database`，`table` - 要刷新数据的表。可以使用返回字符串的常量表达式而不是数据库名称。`num_layers` - 并行层数。在物理上，该表将表示为 `num_layers` 个独立缓冲区。建议值为 16。

`min_time`，`max_time`，`min_rows`，`max_rows`，`min_bytes`，`max_bytes` - 从缓冲区刷新数据的条件。

如果满足所有 «`min`» 条件或至少一个 «`max`» 条件，则从缓冲区刷新数据并将其写入目标表。`min_time`，`max_time` — 从第一次写入缓冲区时起以秒为单位的时间条件。`min_rows`，`max_rows` - 缓冲区中行数的条件。

`min_bytes`，`max_bytes` - 缓冲区中字节数的条件。

写入时，数据从 `num_layers` 个缓冲区中随机插入。或者，如果插入数据的大小足够大（大于 `max_rows` 或 `max_bytes`），则会绕过缓冲区将其写入目标表。

每个 «`num_layers`» 缓冲区刷新数据的条件是分别计算。例如，如果 `num_layers = 16` 且 `max_bytes = 1000000000`，则最大RAM消耗将为 1.6 GB。

示例：

```
CREATE TABLE merge.hits_buffer AS merge.hits ENGINE = Buffer(merge, hits, 16, 10, 100, 10000, 1000000, 10000000, 100000000)
```

创建一个 «`merge.hits_buffer`» 表，其结构与 «`merge.hits`» 相同，并使用 `Buffer` 引擎。写入此表时，数据缓冲在 RAM 中，然后写入 «`merge.hits`» 表。创建了 16 个缓冲区。如果已经过了 100 秒，或者已写入 100 万行，或者已写入 100 MB 数据，则刷新每个缓冲区的数据；或者如果同时已经过了 10 秒并且已经写入了 10,000 行和 10 MB 的数据。例如，如果只写了一行，那么在 100 秒之后，都会被刷新。但是如果写了很多行，数据将会更快地刷新。

当服务器停止时，使用 `DROP TABLE` 或 `DETACH TABLE`，缓冲区数据也会刷新到目标表。

可以为数据库和表名在单个引号中设置空字符串。这表示没有目的地表。在这种情况下，当达到数据刷新条件时，缓冲器被简单地清除。这可能对于保持数据窗口在内存中是有用的。

从 `Buffer` 表读取时，将从缓冲区和目标表（如果有）处理数据。

请注意，`Buffer` 表不支持索引。换句话说，缓冲区中的数据被完全扫描，对于大缓冲区来说可能很慢。（对于目标表中的数据，将使用它支持的索引。）

如果 `Buffer` 表中的列集与目标表中的列集不匹配，则会插入两个表中存在的列的子集。

如果类型与 `Buffer` 表和目标表中的某列不匹配，则会在服务器日志中输入错误消息并清除缓冲区。

如果在刷新缓冲区时目标表不存在，则会发生同样的情况。

如果需要为目标表和 Buffer 表运行 ALTER，我们建议先删除 Buffer 表，为目标表运行 ALTER，然后再次创建 Buffer 表。

如果服务器异常重启，缓冲区中的数据将丢失。

PREWHERE，FINAL 和 SAMPLE 对缓冲表不起作用。这些条件将传递到目标表，但不用于处理缓冲区中的数据。因此，我们建议只使用 Buffer 表进行写入，同时从目标表进行读取。

将数据添加到缓冲区时，其中一个缓冲区被锁定。如果同时从表执行读操作，则会导致延迟。

插入到 Buffer 表中的数据可能以不同的顺序和不同的块写入目标表中。因此，Buffer 表很难用于正确写入 CollapsingMergeTree。为避免出现问题，您可以将 «num_layers» 设置为 1。

如果目标表是复制表，则在写入 Buffer 表时会丢失复制表的某些预期特征。数据部分的行次序和大小的随机变化导致数据不能去重，这意味着无法对复制表进行可靠的 «exactly once» 写入。

由于这些缺点，我们只建议在极少数情况下使用 Buffer 表。

当在单位时间内从大量服务器接收到太多 INSERTs 并且在插入之前无法缓冲数据时使用 Buffer 表，这意味着这些 INSERTs 不能足够快地执行。

请注意，一次插入一行数据是没有意义的，即使对于 Buffer 表也是如此。这将只产生每秒几千行的速度，而插入更大的数据块每秒可以产生超过一百万行（参见 «性能» 部分）。

视图

用于构建视图（有关更多信息，请参阅 CREATE VIEW 查询）。它不存储数据，仅存储指定的 SELECT 查询。从表中读取时，它会运行此查询（并从查询中删除所有不必要的列）。

集合

始终存在于 RAM 中的数据集。它适用于 IN 运算符的右侧（请参见 «IN 运算符» 部分）。

可以使用 INSERT 向表中插入数据。新元素将添加到数据集中，而重复项将被忽略。但是不能对此类型表执行 SELECT 语句。检索数据的唯一方法是在 IN 运算符的右半部分使用它。

数据始终存在于 RAM 中。对于 INSERT，插入数据块也会写入磁盘上的表目录。启动服务器时，此数据将加载到 RAM。也就是说，重新启动后，数据仍然存在。

对于强制服务器重启，磁盘上的数据块可能会丢失或损坏。在数据块损坏的情况下，可能需要手动删除包含损坏数据的文件。

表引擎

表引擎（即表的类型）决定了：

- 数据的存储方式和位置，写到哪里以及从哪里读取数据
- 支持哪些查询以及如何支持。
- 并发数据访问。
- 索引的使用（如果存在）。
- 是否可以执行多线程请求。
- 数据复制参数。

引擎类型

MergeTree

适用于高负载任务的最通用和功能最强大的表引擎。这些引擎的共同特点是可以快速插入数据并进行后续的后台数据处理。MergeTree系列引擎支持数据复制（使用Replicated*的引擎版本），分区和一些其他引擎不支持的其他功能。

该类型的引擎：

- MergeTree
- ReplacingMergeTree
- SummingMergeTree
- AggregatingMergeTree
- CollapsingMergeTree
- VersionedCollapsingMergeTree
- GraphiteMergeTree

日志

具有最小功能的轻量级引擎。当您需要快速写入许多小表（最多约100万行）并在以后整体读取它们时，该类型的引擎是最有效的。

该类型的引擎：

- TinyLog
- StripeLog
- Log

集成引擎

用于与其他的数据存储与处理系统集成的引擎。

该类型的引擎：

- Kafka
- MySQL
- ODBC
- JDBC
- HDFS

用于其他特定功能的引擎

该类型的引擎：

- Distributed
- MaterializedView
- Dictionary
- Merge
- File
- Null
- Set

- [Join](#)
- [URL](#)
- [View](#)
- [Memory](#)
- [Buffer](#)

虚拟列

虚拟列是表引擎组成的一部分，它在对应的表引擎的源代码中定义。

您不能在 `CREATE TABLE` 中指定虚拟列，并且虚拟列不会包含在 `SHOW CREATE TABLE` 和 `DESCRIBE TABLE` 的查询结果中。虚拟列是只读的，所以您不能向虚拟列中写入数据。

如果想要查询虚拟列中的数据，您必须在 `SELECT` 查询中包含虚拟列的名字。`SELECT *` 不会返回虚拟列的内容。

若您创建的表中有一列与虚拟列的名字相同，那么虚拟列将不能再被访问。我们不建议您这样做。为了避免这种列名的冲突，虚拟列的名字一般都以下划线开头。

SQL参考

ClickHouse 支持以下形式的查询：

- [SELECT](#)
- [INSERT INTO](#)
- [CREATE](#)
- [ALTER](#)
- [其他类型的查询](#)

SQL语法

ClickHouse 有 2 类解析器：完整 SQL 解析器（递归式解析器），以及数据格式解析器（快速流式解析器）。除了 `INSERT` 查询，其它情况下仅使用完整 SQL 解析器。

`INSERT` 查询会同时使用 2 种解析器：

```
INSERT INTO t VALUES (1, 'Hello, world'), (2, 'abc'), (3, 'def')
```

含 `INSERT INTO t VALUES` 的部分由完整 SQL 解析器处理，包含数据的部分 `(1, 'Hello, world'), (2, 'abc'), (3, 'def')` 交给快速流式解析器解析。通过设置参数 `input_format_values_interpret_expressions`，你也可以对数据部分开启完整 SQL 解析器。当 `input_format_values_interpret_expressions = 1` 时，ClickHouse 优先采用快速流式解析器来解析数据。如果失败，ClickHouse 再尝试用完整 SQL 解析器来处理，就像处理 SQL `expression` 一样。

数据可以采用任何格式。当 CH 接收到请求时，服务端先在内存中计算不超过 `max_query_size` 字节的数据（默认 1 mb），然后剩下部分交给快速流式解析器。

当 `INSERT` 语句中使用 `Values` 格式时，看起来数据部分的解析和解析 `SELECT` 中的表达式相同，但并不是这样的。`Values` 格式有非常多的限制。

本文的剩余部分涵盖了完整 SQL 解析器。关于格式解析的更多信息，参见 [Formats](#) 章节。

空白{#spaces}

SQL语句的语法结构部分之间（标识符之间、部分符号之间、包括SQL的起始和结束）可以有任意的空白字符，这些空字符类型包括：空格字符，tab制表符，换行符，CR符，换页符等。

注释

ClickHouse支持SQL风格或C语言风格的注释：

- SQL风格的注释以`--`开始，直到行末，`--`后紧跟的空格可以忽略
- C语言风格的注释以`/*`开始，以`*/`结束，可以跨行，同样可以省略`/*`后的空格

关键字

以下场景的关键字是大小写不敏感的：

- 标准SQL。例如，`SELECT`, `select` 和 `SeLeCt` 都是允许的
- 在某些流行的RDBMS中被实现的关键字，例如，`DateTime` 和 `datetime`是一样的

你可以在系统表 `system.data_type_families` 中检查某个数据类型的名称是否是大小写敏感型。

和标准SQL相反，所有其它的关键字都是大小写敏感的，包括函数名称。

关键字不是保留的；它们仅在相应的上下文中才会被认为是关键字。如果你使用和关键字同名的 **标识符**，需要使用双引号或反引号将它们包含起来。例如：如果表 `table_name` 包含列 `"FROM"`，那么 `SELECT "FROM" FROM table_name` 是合法的

标识符

标识符包括：

- 集群、数据库、表、分区、列的名称
- 函数
- 数据类型
- 表达式别名

变量名可以被括起或不括起，后者是推荐做法。

没有括起的变量名，必须匹配正则表达式 `^[a-zA-Z_][0-9a-zA-Z_]*$`，并且不能和 **关键字**相同，合法的标识符名称：`x`, `_1`, `X_y_Z123` 等。

如果想使用和关键字同名的变量名称，或者在变量名称中包含其它符号，你需要通过双引号或反引号，例如：`"id"`, ``id``

字符

字符包含数字，字母，括号，NULL值等字符。

数字

数字类型字符会被做如下解析：

- 首先，当做64位的有符号整数，使用函数 `strtoull`
- 如果失败，解析成64位无符号整数，同样使用函数 `strtoull`
- 如果还失败了，试图解析成浮点型数值，使用函数 `strtod`
- 最后，以上情形都不符合时，返回异常

数字类型的值类型为能容纳该值的最小数据类型。

例如：1 解析成 UInt8型，256 则解析成 UInt16。更多信息，参见 [数据类型](#)

例如: 1, 18446744073709551615, 0xDEADBEEF, 01, 0.1, 1e100, -1e-100, inf, nan.

字符串

ClickHouse只支持用单引号包含的字符串。特殊字符可通过反斜杠进行转义。下列转义字符都有相应的实际值：`\b`, `\f`, `\r`, `\n`, `\t`, `\0`, `\a`, `\v`, `\xHH`。其它情况下，以`\c`形式出现的转义字符，当`c`表示任意字符时，转义字符会转换成`c`。这意味着你可以使用`\``和`\\"`。该值将拥有[String](#)类型。

在字符串中，你至少需要对`'`和`\``进行转义。单引号可以使用单引号转义，例如`'\t\'s'`和`'\t\"s'`是相同的。

复合字符串

数组都是使用方括号进行构造 [1, 2, 3]，元组则使用圆括号 (1, 'Hello, world!', 2)

从技术上来讲，这些都不是字符串，而是包含创建数组和元组运算符的表达式。

创建一个数组必须至少包含一个元素，创建一个元组至少包含2个元素

当元组出现在 `SELECT` 查询的 `IN` 部分时，是一种例外情形。查询结果可以包含元组，但是元组类型不能保存到数据库中（除非表采用[内存表引擎](#)）

NULL值

代表不存在的值。

为了能在表字段中存储NULL值，该字段必须声明为[空值](#)类型。

根据数据的格式（输入或输出），NULL值有不同的表现形式。更多信息参见文档[数据格式](#)

在处理 `NULL` 时存在很多细微差别。例如，比较运算的至少一个参数为 `NULL`，则该结果也是 `NULL`。与之类似的还有乘法运算，加法运算，以及其他运算。更多信息，请参阅每种运算的文档部分。

在语句中，可以通过 `IS NULL` 以及 `IS NOT NULL` 运算符，以及 `isNull`、`isNotNull` 函数来检查 `NULL` 值

函数

函数调用的写法，类似于一个标识符后接被圆括号包含的参数列表（可能为空）。与标准SQL不同，圆括号是必须的，不管参数列表是否为空。例如：`now()`。

函数分为常规函数和聚合函数（参见“Aggregate functions”一章）。有些聚合函数包含2个参数列表，第一个参数列表中的参数被称为“parameters”。不包含“parameters”的聚合函数语法和常规函数是一样的。

运算符

在查询解析阶段，运算符会被转换成对应的函数，使用时请注意它们的优先级。例如：

表达式 `1 + 2 * 3 + 4` 会被解析成 `plus(plus(1, multiply(2, 3)), 4)`.

数据类型及数据库/表引擎

`CREATE` 语句中的数据类型和表引擎写法与变量或函数类似。

换句话说，它们可以包含或不包含用括号包含的参数列表。更多信息，参见“数据类型,” “数据表引擎” 和 “`CREATE` 语句”等章节

表达式别名

别名是用户对表达式的自定义名称

`expr AS alias`

■ AS — 用于定义别名的关键字。可以对表或select语句中的列定义别名(AS 可以省略)

例如, `SELECT table_name_alias.column_name FROM table_name table_name_alias.`

在 **CAST函数** 中, AS有其它含义。请参见该函数的说明部分。

■ expr — 任意CH支持的表达式。

例如, `SELECT column_name * 2 AS double FROM some_table`

■ alias — expr 的名称。别名必须符合 **标识符** 语法。

例如, `SELECT "table t".column_name FROM table_name AS "table t"`.

用法注意

别名在当前查询或子查询中是全局可见的，你可以在查询语句的任何位置对表达式定义别名

别名在当前查询的子查询及不同子查询中是不可见的。例如，执行如下查询SQL: `SELECT (SELECT sum(b.a) + num FROM b) - a.a AS num FROM a`, ClickHouse会提示异常 `Unknown identifier: num.`

如果给select子查询语句的结果列定义其别名，那么在外层可以使用该别名。例如, `SELECT n + m FROM (SELECT 1 AS n, 2 AS m).`

注意列的别名和表的别名相同时的情形，考虑如下示例：

```
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.
```

在这个示例中，先声明了表 `t` 以及列 `b`。然后，在查询数据时，又定义了别名 `sum(b) AS b`。由于别名是全局的，ClickHouse使用表达式 `sum(b)` 来替换表达式 `argMax(a, b)` 中的变量 `b`。这种替换导致出现异常。

星号

select查询中，星号可以代替表达式使用。详情请参见“`select`”部分

表达式

表达式是函数、标识符、字符、使用运算符的语句、括号中的表达式、子查询或星号。它也可以包含别名。

表达式列表是用逗号分隔的一个或多个表达式。

反过来，函数和运算符可以将表达式作为参数。

选择查询

`SELECT` 查询执行数据检索。默认情况下，请求的数据返回给客户端，同时结合 **INSERT INTO** 可以被转发到不同的表。

语法

```
[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 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]
[UNION ALL ...]
[INTO OUTFILE filename]
[FORMAT format]
```

所有子句都是可选的，但紧接在 `SELECT` 后面的必需表达式列表除外，更详细的请看 [下面](#).

每个可选子句的具体内容在单独的部分中进行介绍，这些部分按与执行顺序相同的顺序列出：

- [WITH 子句](#)
- [FROM 子句](#)
- [SAMPLE 子句](#)
- [JOIN 子句](#)
- [PREWHERE 子句](#)
- [WHERE 子句](#)
- [GROUP BY 子句](#)
- [LIMIT BY 子句](#)
- [HAVING 子句](#)
- [SELECT 子句](#)
- [DISTINCT 子句](#)
- [LIMIT 子句](#)
- [UNION ALL 子句](#)
- [INTO OUTFILE 子句](#)
- [FORMAT 子句](#)

SELECT 子句

[表达式](#) 指定 `SELECT` 子句是在上述子句中的所有操作完成后计算的。这些表达式的工作方式就好像它们应用于结果中的单独行一样。如果表达式 `SELECT` 子句包含聚合函数，然后 ClickHouse 将使用 [GROUP BY](#) 聚合参数应用在聚合函数和表达式上。

如果在结果中包含所有列，请使用星号 (*) 符号。例如，`SELECT * FROM ...`

将结果中的某些列与 [re2](#) 正则表达式匹配，可以使用 [COLUMNS](#) 表达。

```
COLUMNS('regexp')
```

例如表:

```
CREATE TABLE default.col_names (aa Int8, ab Int8, bc Int8) ENGINE = TinyLog
```

以下查询所有列名包含 `a`。

```
SELECT COLUMNS('a') FROM col_names
```

aa	ab
1	1

所选列不按字母顺序返回。

您可以使用多个 `COLUMNS` 表达式并将函数应用于它们。

例如:

```
SELECT COLUMNS('a'), COLUMNS('c'), toTypeName(COLUMNS('c')) FROM col_names
```

aa	ab	bc	toTypeName(bc)
1	1	1	Int8

返回的每一列 `COLUMNS` 表达式作为单独的参数传递给函数。如果函数支持其他参数，您也可以将其他参数传递给函数。使用函数时要小心，如果函数不支持传递给它的参数，ClickHouse 将抛出异常。

例如:

```
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.
```

该例子中，`COLUMNS('a')` 返回两列：`aa` 和 `ab`. `COLUMNS('c')` 返回 `bc` 列。该 `+` 运算符不能应用于3个参数，因此 ClickHouse 抛出一个带有相关消息的异常。

匹配的列 `COLUMNS` 表达式可以具有不同的数据类型。如果 `COLUMNS` 不匹配任何列，并且是在 `SELECT` 唯一的表达式，ClickHouse 则抛出异常。

星号

您可以在查询的任何部分使用星号替代表达式。进行查询分析、时，星号将展开为所有表的列（不包括 `MATERIALIZED` 和 `ALIAS` 列）。只有少数情况下使用星号是合理的：

- 创建转储表时。
- 对于只包含几列的表，例如系统表。
- 获取表中列的信息。在这种情况下，设置 `LIMIT 1`. 但最好使用 `DESC TABLE` 查询。

- 当对少量列使用 `PREWHERE` 进行强过滤时。
- 在子查询中（因为外部查询不需要的列从子查询中排除）。

在所有其他情况下，我们不建议使用星号，因为它只给你一个列DBMS的缺点，而不是优点。换句话说，不建议使用星号。

极端值

除结果之外，还可以获取结果列的最小值和最大值。要做到这一点，设置 `extremes` 设置为 `1`。最小值和最大值是针对数字类型、日期和带有时间的日期计算的。对于其他类型列，输出默认值。

分别的额外计算两行 - 最小值和最大值。这额外的两行采用输出格式为 `JSON*`, `TabSeparated*`, 和 `Pretty*` `formats`，与其他行分开。它们不以其他格式输出。

为 `JSON*` 格式时，极端值单独的输出在 ‘`extremes`’ 字段。为 `TabSeparated*` 格式时，此行来的主要结果集后，然后显示 ‘`totals`’ 字段。它前面有一个空行（在其他数据之后）。在 `Pretty*` 格式时，该行在主结果之后输出为一个单独的表，然后显示 ‘`totals`’ 字段。

极端值在 `LIMIT` 之前被计算，但在 `LIMIT BY` 之后被计算。然而，使用 `LIMIT offset, size`，`offset` 之前的行都包含在 `extremes`。在流请求中，结果还可能包括少量通过 `LIMIT` 过滤的行。

备注

您可以在查询的任何部分使用同义词（AS 别名）。

`GROUP BY` 和 `ORDER BY` 子句不支持位置参数。这与MySQL相矛盾，但符合标准SQL。例如，`GROUP BY 1, 2` 将被理解为根据常量分组 (i.e. aggregation of all rows into one).

实现细节

如果查询省略 `DISTINCT`, `GROUP BY` , `ORDER BY` , `IN` , `JOIN` 子查询，查询将被完全流处理，使用 $O(1)$ 量的RAM。若未指定适当的限制，则查询可能会消耗大量RAM:

- `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`

有关详细信息，请参阅部分 “`Settings`”。可以使用外部排序（将临时表保存到磁盘）和外部聚合。

ALL 子句

`SELECT ALL` 和 `SELECT` 不带 `DISTINCT` 是一样的。

- 如果指定了 ALL，则忽略它。
- 如果同时指定了 ALL 和 DISTINCT，则会抛出异常。

ALL 也可以在聚合函数中指定，具有相同的效果（空操作）。例如：

```
SELECT sum(ALL number) FROM numbers(10);
```

等于

```
SELECT sum(number) FROM numbers(10);
```

ARRAY JOIN子句

对于包含数组列的表来说是一种常见的操作，用于生成一个新表，该表具有包含该初始列中的每个单独数组元素的列，而其他列的值将被重复显示。这是 ARRAY JOIN 语句最基本的场景。

它可以被视为执行 JOIN 并具有数组或嵌套数据结构。类似于 **arrayJoin** 功能，但该子句功能更广泛。

语法：

```
SELECT <expr_list>
FROM <left_subquery>
[LEFT] ARRAY JOIN <array>
[WHERE|PREWHERE <expr>]
...
```

您只能在 SELECT 查询指定一个 ARRAY JOIN 。

ARRAY JOIN 支持的类型有：

- **ARRAY JOIN** - 一般情况下，空数组不包括在结果中 JOIN。
- **LEFT ARRAY JOIN** - 的结果 JOIN 包含具有空数组的行。空数组的值设置为数组元素类型的默认值（通常为 0、空字符串或 NULL）。

基本 ARRAY JOIN 示例

下面的例子展示 ARRAY JOIN 和 LEFT ARRAY JOIN 的用法，让我们创建一个表包含一个 Array 的列并插入值：

```
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	[]

下面的例子使用 ARRAY JOIN 子句：

```
SELECT s, arr
FROM arrays_test
ARRAY JOIN arr;
```

s	arr
Hello	1
Hello	2
World	3
World	4
World	5

下一个示例使用 `LEFT ARRAY JOIN` 子句：

```
SELECT s, arr
FROM arrays_test
LEFT ARRAY JOIN arr;
```

s	arr
Hello	1
Hello	2
World	3
World	4
World	5
Goodbye	0

使用别名

在使用 `ARRAY JOIN` 时可以为数组指定别名，数组元素可以通过此别名访问，但数组本身则通过原始名称访问。示例：

```
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

可以使用别名与外部数组执行 `ARRAY JOIN`。例如：

```
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

在 `ARRAY JOIN` 中，多个数组可以用逗号分隔，在这例子中 `JOIN` 与它们同时执行（直接 `sum`，而不是笛卡尔积）。请注意，所有数组必须具有相同的大小。示例：

```
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

下面的例子使用 `arrayEnumerate` 功能：

```
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]

具有嵌套数据结构的数组连接

`ARRAY JOIN` 也适用于 嵌套数据结构：

```
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

当指定嵌套数据结构的名称 `ARRAY JOIN`，意思是一样的 `ARRAY JOIN` 它包含的所有数组元素。下面列出了示例：

```
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

这种变化也是有道理的：

```
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]

可以将别名用于嵌套数据结构，以便选择 `JOIN` 结果或源数组。例如：

```
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]

使用功能 `arrayEnumerate` 的例子：

```
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

实现细节

运行时优化查询执行顺序 `ARRAY JOIN`。虽然 `ARRAY JOIN` 必须始终之前指定 `WHERE/PREWHERE` 子句中的查询，从技术上讲，它们可以以任何顺序执行，除非结果 `ARRAY JOIN` 用于过滤。处理顺序由查询优化器控制。

DISTINCT子句

如果 `SELECT DISTINCT` 被声明，则查询结果中只保留唯一行。因此，在结果中所有完全匹配的行集合中，只有一行被保留。

空处理

`DISTINCT` 适用于 `NULL` 就好像 `NULL` 是一个特定的值，并且 `NULL==NULL`。换句话说，在 `DISTINCT` 结果，不同的组合 `NULL` 仅发生一次。它不同于 `NULL` 在大多数其他情况中的处理方式。

替代办法

通过应用可以获得相同的结果 `GROUP BY` 在同一组值指定为 `SELECT` 子句，并且不使用任何聚合函数。但与 `GROUP BY` 有几个不同的地方：

- `DISTINCT` 可以与 `GROUP BY` 一起使用。
- 当 `ORDER BY` 被省略并且 `LIMIT` 被定义时，在读取所需数量的不同行后立即停止运行。
- 数据块在处理时输出，而无需等待整个查询完成运行。

限制

`DISTINCT` 不支持当 `SELECT` 包含有数组的列。

例子

ClickHouse 支持使用 `DISTINCT` 和 `ORDER BY` 在一个查询中的不同的列。`DISTINCT` 子句在 `ORDER BY` 子句前被执行。

示例表：

a	b
2	1
1	2
3	3
2	4

当执行 `SELECT DISTINCT a FROM t1 ORDER BY b ASC` 来查询数据，我们得到以下结果：

a
2
1
3

如果我们改变排序方向 `SELECT DISTINCT a FROM t1 ORDER BY b DESC`, 我们得到以下结果:

a
3
1
2

行 2, 4 排序前被切割。

在编程查询时考虑这种实现特性。

格式化子句

ClickHouse 支持广泛的 **序列化格式** 可用于查询结果等。有多种方法可以选择格式化 `SELECT` 的输出，其中之一是指定 `FORMAT format` 在查询结束时以任何特定格式获取结果集。

特定的格式方便使用，与其他系统集成或增强性能。

默认格式

如果 `FORMAT` 被省略则使用默认格式，这取决于用于访问 ClickHouse 服务器的设置和接口。为 **HTTP 接口** 和 **命令行客户端** 在批处理模式下，默认格式为 `TabSeparated`。对于交互模式下的命令行客户端，默认格式为 `PrettyCompact`（它生成紧凑的人类可读表）。

实现细节

使用命令行客户端时，数据始终以内部高效格式通过网络传递 (`Native`)。客户端独立解释 `FORMAT` 查询子句并格式化数据本身（以减轻网络和服务器的额外负担）。

FROM 子句

`FROM` 子句指定从以下数据源中读取数据：

- 表
- 子查询
- 表函数

`JOIN` 和 `ARRAY JOIN` 子句也可以用来扩展 `FROM` 的功能

子查询是另一个 `SELECT` 可以指定在 `FROM` 后的括号内的查询。

`FROM` 子句可以包含多个数据源，用逗号分隔，这相当于在他们身上执行 `CROSS JOIN`

FINAL 修饰符

当 `FINAL` 被指定，ClickHouse 会在返回结果之前完全合并数据，从而执行给定表引擎合并期间发生的所有数据转换。

它适用于从使用 `MergeTree`-引擎族（除了 `GraphiteMergeTree`）。还支持：

- **Replicated** 版本 MergeTree 引擎
- **View, Buffer, Distributed**，和 **MaterializedView** 在其他引擎上运行的引擎，只要是它们底层是 **MergeTree**-引擎表即可。

现在使用 **FINAL** 修饰符的 **SELECT** 查询启用了并发执行，这会快一点。但是仍然存在缺陷（见下）。**max_final_threads** 设置使用的最大线程数限制。

缺点

使用的查询 **FINAL** 执行速度比类似的查询慢一点，因为：

- 在查询执行期间合并数据。
- 查询与 **FINAL** 除了读取查询中指定的列之外，还读取主键列。

在大多数情况下，避免使用 **FINAL**。常见的方法是使用假设后台进程的不同查询 **MergeTree** 引擎还没有发生，并通过应用聚合（例如，丢弃重复项）来处理它。

实现细节

如果 **FROM** 子句被省略，数据将从读取 **system.one** 表。

该 **system.one** 表只包含一行（此表满足与其他 DBMS 中的 **DUAL** 表有相同的作用）。

若要执行查询，将从相应的表中提取查询中列出的所有列。外部查询不需要的任何列都将从子查询中抛出。

如果查询未列出任何列（例如，**SELECT count() FROM t**），无论如何都会从表中提取一些列（首选是最小的列），以便计算行数。

GROUP BY 子句

GROUP BY 子句将 **SELECT** 查询结果转换为聚合模式，其工作原理如下：

- **GROUP BY** 子句包含表达式列表（或单个表达式 -- 可以认为是长度为1的列表）。这份名单充当“grouping key”，而每个单独的表达式将被称为“key expressions”。
- 在所有的表达式在 **SELECT**, **HAVING**，和 **ORDER BY** 子句中 必须 基于键表达式进行计算 或 上 聚合函数 在非键表达式（包括纯列）上。换句话说，从表中选择的每个列必须用于键表达式或聚合函数内，但不能同时使用。
- 聚合结果 **SELECT** 查询将包含尽可能多的行，因为有唯一值“grouping key”在源表中。通常这会显着减少行数，通常是数量级，但不一定：如果所有行数保持不变“grouping key”值是不同的。

注

还有一种额外的方法可以在表上运行聚合。如果查询仅在聚合函数中包含表列，则 **GROUP BY** 可以省略，并且通过一个空的键集合来假定聚合。这样的查询总是只返回一行。

空处理

对于分组，ClickHouse解释 **NULL** 作为一个值，并且 **NULL==NULL**。它不同于 **NULL** 在大多数其他上下文中的处理方式。

这里有一个例子来说明这意味着什么。

假设你有一张表：

x	y
1	2
2	NULL
3	2
3	3
3	NULL

查询 `SELECT sum(x), y FROM t_null_big GROUP BY y` 结果:

sum(x)	y
4	2
3	3
5	NULL

你可以看到 `GROUP BY` 为 `y = NULL` 总结 `x`，仿佛 `NULL` 是这个值。

如果你通过几个键 `GROUP BY`，结果会给你选择的所有组合，就好像 `NULL` 是一个特定的值。

WITH TOTAL 修饰符

如果 `WITH TOTALS` 被指定，将计算另一行。此行将具有包含默认值（零或空行）的关键列，以及包含跨所有行计算值的聚合函数列（“total”值）。

这个额外的行仅产生于 `JSON*`, `TabSeparated*`, 和 `Pretty*` 格式，与其他行分开:

- 在 `JSON*` 格式，这一行是作为一个单独的输出 ‘totals’ 字段。
- 在 `TabSeparated*` 格式，该行位于主结果之后，前面有一个空行（在其他数据之后）。
- 在 `Pretty*` 格式时，该行在主结果之后作为单独的表输出。
- 在其他格式中，它不可用。

`WITH TOTALS` 可以以不同的方式运行时 `HAVING` 是存在的。该行为取决于 `totals_mode` 设置。

配置总和处理

默认情况下，`totals_mode = 'before_having'`. 在这种情况下，‘totals’ 是跨所有行计算，包括那些不通过具有和 `max_rows_to_group_by`.

其他替代方案仅包括通过具有在 ‘totals’，并与设置不同的行为 `max_rows_to_group_by` 和 `group_by_overflow_mode = 'any'`.

`after_having_exclusive` – Don't include rows that didn't pass through `max_rows_to_group_by`. 换句话说，‘totals’ 将有少于或相同数量的行，因为它会 `max_rows_to_group_by` 被省略。

`after_having_inclusive` – Include all the rows that didn't pass through ‘`max_rows_to_group_by`’ 在 ‘totals’. 换句话说，‘totals’ 将有多个或相同数量的行，因为它会 `max_rows_to_group_by` 被省略。

`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`’ 在 ‘totals’. 否则，不包括它们。

`totals_auto_threshold` – By default, 0.5. The coefficient for `after_having_auto`.

如果 `max_rows_to_group_by` 和 `group_by_overflow_mode = 'any'` 不使用，所有的变化 `after_having` 是相同的，你可以使用它们中的任何一个（例如，`after_having_auto`）。

您可以使用 `WITH TOTALS` 在子查询中，包括在子查询 `JOIN` 子句（在这种情况下，将各自的总值合并）。

例子

示例：

```
SELECT
    count(),
    median(FetchTiming > 60 ? 60 : FetchTiming),
    count() - sum(Refresh)
FROM hits
```

但是，与标准SQL相比，如果表没有任何行（根本没有任何行，或者使用 `WHERE` 过滤之后没有任何行），则返回一个空结果，而不是来自包含聚合函数初始值的行。

相对于MySQL（并且符合标准SQL），您无法获取不在键或聚合函数（常量表达式除外）中的某些列的某些值。要解决此问题，您可以使用‘any’聚合函数（获取第一个遇到的值）或‘min/max’。

示例：

```
SELECT
    domainWithoutWWW(URL) AS domain,
    count(),
    any>Title AS title -- getting the first occurred page header for each domain.
FROM hits
GROUP BY domain
```

对于遇到的每个不同的键值，`GROUP BY` 计算一组聚合函数值。

`GROUP BY` 不支持数组列。

不能将常量指定为聚合函数的参数。示例：`sum(1)`. 相反，你可以摆脱常数。示例：`count()`.

实现细节

聚合是面向列的DBMS最重要的功能之一，因此它的实现是ClickHouse中最优化的部分之一。默认情况下，聚合使用哈希表在内存中完成。它有40+的特殊化自动选择取决于“grouping key”数据类型。

在外部存储器中分组

您可以启用将临时数据转储到磁盘以限制内存使用期间 `GROUP BY`.

该 `max_bytes_before_external_group_by` 设置确定倾销的阈值RAM消耗 `GROUP BY` 临时数据到文件系统。如果设置为0（默认值），它将被禁用。

使用时 `max_bytes_before_external_group_by`，我们建议您设置 `max_memory_usage` 大约两倍高。这是必要的，因为聚合有两个阶段：读取数据和形成中间数据（1）和合并中间数据（2）。将数据转储到文件系统只能在阶段1中发生。如果未转储临时数据，则阶段2可能需要与阶段1相同的内存量。

例如，如果 `max_memory_usage` 设置为100000000000，你想使用外部聚合，这是有意义的设置

`max_bytes_before_external_group_by` 到100000000000，和 `max_memory_usage` 到200000000000。当触发外部聚合（如果至少有一个临时数据转储）时，RAM的最大消耗仅略高于 `max_bytes_before_external_group_by`.

通过分布式查询处理，在远程服务器上执行外部聚合。为了使请求者服务器只使用少量的RAM，设置 `distributed_aggregation_memory_efficient` 到1。

当合并数据刷新到磁盘时，以及当合并来自远程服务器的结果时，`distributed_aggregation_memory_efficient` 设置被启用，消耗高达 `1/256 * the_number_of_threads` 从RAM的总量。

当启用外部聚合时，如果数据量小于 `max_bytes_before_external_group_by`（例如数据没有被flushed），查询执行速度和不在外部聚合的速度一样快。如果临时数据被flushed到外部存储，执行的速度会慢几倍（大概是三倍）。

如果你有一个 `ORDER BY` 用一个 `LIMIT` 后 `GROUP BY`，然后使用的RAM的量取决于数据的量 `LIMIT`，不是在整个表。但如果 `ORDER BY` 没有 `LIMIT`，不要忘记启用外部排序 (`max_bytes_before_external_sort`)。

HAVING 子句

允许过滤由 `GROUP BY` 生成的聚合结果。它类似于 `WHERE`，但不同的是 `WHERE` 在聚合之前执行，而 `HAVING` 之后进行。

可以从 `SELECT` 生成的聚合结果中通过他们的别名来执行 `HAVING` 子句。或者 `HAVING` 子句可以筛选查询结果中未返回的其他聚合的结果。

限制

`HAVING` 如果不执行聚合则无法使用。使用 `WHERE` 则相反。

INTO OUTFILE 子句

添加 `INTO OUTFILE filename` 子句（其中`filename`是字符串）`SELECT query` 将其输出重定向到客户端上的指定文件。

实现细节

- 此功能是在可用 `命令行客户端` 和 `clickhouse-local`。因此通过 `HTTP接口` 发送查询将会失败。
- 如果具有相同文件名的文件已经存在，则查询将失败。
- 默认值 `输出格式` 是 `TabSeparated`（就像在命令行客户端批处理模式中一样）。

JOIN子句

`Join`通过使用一个或多个表的公共值合并来自一个或多个表的列来生成新表。它是支持SQL的数据库中的常见操作，它对应于 `关系代数` 加入。一个表连接的特殊情况通常被称为“self-join”。

语法:

```
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>) ...
```

从表达式 `ON` 从子句和列 `USING` 子句被称为“join keys”。除非另有说明，加入产生一个 `笛卡尔积` 从具有匹配的行“join keys”，这可能会产生比源表更多的行的结果。

支持的联接类型

所有标准 `SQL JOIN` 支持类型:

- `INNER JOIN`，只返回匹配的行。
- `LEFT OUTER JOIN`，除了匹配的行之外，还返回左表中的非匹配行。
- `RIGHT OUTER JOIN`，除了匹配的行之外，还返回右表中的非匹配行。
- `FULL OUTER JOIN`，除了匹配的行之外，还会返回两个表中的非匹配行。
- `CROSS JOIN`，产生整个表的笛卡尔积，“join keys”是不指定。

`JOIN` 没有指定类型暗指 `INNER`. 关键字 `OUTER` 可以安全地省略。替代语法 `CROSS JOIN` 在指定多个表 `FROM` 用逗号分隔。

ClickHouse中提供的其他联接类型:

- `LEFT SEMI JOIN` 和 `RIGHT SEMI JOIN`,白名单“join keys”，而不产生笛卡尔积。
- `LEFT ANTI JOIN` 和 `RIGHT ANTI JOIN`，黑名单“join keys”，而不产生笛卡尔积。
- `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.

严格

注

可以使用以下方式复盖默认的严格性值 `join_default_striictness` 设置。

Also the behavior of ClickHouse server for ANY JOIN operations depends on the `any_join_distinct_right_table_keys` setting.

ASOF JOIN使用

`ASOF JOIN` 当您需要连接没有完全匹配的记录时非常有用。

该算法需要表中的特殊列。该列需要满足:

- 必须包含有序序列。
- 可以是以下类型之一: `Int` , `UInt` , `Float*` , `Date` , `DateTime` , `Decimal*`.
- 不能是`JOIN`子句中唯一的列

语法 `ASOF JOIN ... ON:`

```
SELECT expressions_list
FROM table_1
ASOF LEFT JOIN table_2
ON equi_cond AND closest_match_cond
```

您可以使用任意数量的相等条件和一个且只有一个最接近的匹配条件。例如, `SELECT count() FROM table_1 ASOF LEFT JOIN table_2 ON table_1.a == table_2.b AND table_2.t <= table_1.t.`

支持最接近匹配的运算符: `>`, `>=`, `<`, `<=`.

语法 `ASOF JOIN ... USING:`

```
SELECT expressions_list
FROM table_1
ASOF JOIN table_2
USING (equi_column1, ... equi_columnN, asof_column)
```

`table_1.asof_column >= table_2.asof_column` 中, `ASOF JOIN` 使用 `equi_columnX` 来进行条件匹配, `asof_column` 用于 `JOIN`最接近匹配。`asof_column` 列总是在最后一个 `USING` 条件中。

例如,参考下表:

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会从 `table_2` 中的用户事件时间戳找出和 `table_1` 中用户事件时间戳中最近的一个时间戳，来满足最接近匹配的条件。如果有得话，则相等的时间戳值是最接近的值。在此例中，`user_id` 列可用于条件匹配，`ev_time` 列可用于最接近匹配。在此例中，`event_1_1` 可以 JOIN `event_2_1`，`event_1_2` 可以JOIN `event_2_3`，但是 `event_2_2` 不能被JOIN。

注

ASOF JOIN在 **JOIN** 表引擎中 不受 支持。

分布式联接

有两种方法可以执行涉及分布式表的join:

- 当使用正常 **JOIN**，将查询发送到远程服务器。为了创建正确的表，在每个子查询上运行子查询，并使用此表执行联接。换句话说，在每个服务器上单独形成右表。
- 使用时 **GLOBAL ... JOIN**，首先请求者服务器运行一个子查询来计算正确的表。此临时表将传递到每个远程服务器，并使用传输的临时数据对其进行查询。

使用时要小心 **GLOBAL**. 有关详细信息，请参阅 [分布式子查询](#) 科。

使用建议

处理空单元格或空单元格

在连接表时，可能会出现空单元格。设置 **join_use_nulls** 定义ClickHouse如何填充这些单元格。

如果 **JOIN** 键是 **可为空** 字段，其中至少有一个键具有值的行 **NULL** 没有加入。

语法

在指定的列 **USING** 两个子查询中必须具有相同的名称，并且其他列必须以不同的方式命名。您可以使用别名更改子查询中的列名。

该 **USING** 子句指定一个或多个要联接的列，这将建立这些列的相等性。列的列表设置不带括号。不支持更复杂的连接条件。

语法限制

对于多个 **JOIN** 单个子句 **SELECT** 查询:

- 通过以所有列 * 仅在联接表时才可用，而不是子查询。
- 该 **PREWHERE** 条款不可用。

为 **ON**, **WHERE**, 和 **GROUP BY** 条款:

- 任意表达式不能用于 **ON**, **WHERE**, 和 **GROUP BY** 子句，但你可以定义一个表达式 **SELECT** 子句，然后通过别名在这些子句中使用它。

性能

当运行 `JOIN`，与查询的其他阶段相关的执行顺序没有优化。连接（在右表中搜索）在过滤之前运行 `WHERE` 和聚集之前。

每次使用相同的查询运行 `JOIN`，子查询再次运行，因为结果未缓存。为了避免这种情况，使用特殊的 [加入我们](#) 表引擎，它是一个用于连接的准备好的数组，总是在RAM中。

在某些情况下，使用效率更高 `IN` 而不是 `JOIN`。

如果你需要一个 `JOIN` 对于连接维度表（这些是包含维度属性的相对较小的表，例如广告活动的名称），`JOIN` 由于每个查询都会重新访问正确的表，因此可能不太方便。对于这种情况下，有一个“external dictionaries”您应该使用的功能 `JOIN`。有关详细信息，请参阅 [外部字典](#) 科。

内存限制

默认情况下，ClickHouse使用 [哈希联接](#) 算法。ClickHouse采取 `<right_table>` 并在RAM中为其创建哈希表。在某个内存消耗阈值之后，ClickHouse回退到合并联接算法。

如果需要限制联接操作内存消耗，请使用以下设置：

- `max_rows_in_join` — Limits number of rows in the hash table.
- `max_bytes_in_join` — Limits size of the hash table.

当任何这些限制达到，ClickHouse作为 `join_overflow_mode` 设置指示。

例子

示例：

```
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

`LIMIT m` 允许选择结果中起始的 `m` 行。

`LIMIT n, m` 允许选择从跳过第一个结果后的行 `n` 行。与 `LIMIT m OFFSET n` 语法是等效的。

`n` 和 `m` 必须是非负整数。

如果没有 `ORDER BY` 子句显式排序结果，结果的行选择可能是任意的和非确定性的。

`LIMIT ... WITH TIES` 修饰符

如果为 `LIMIT n[,m]` 设置了 `WITH TIES`，并且声明了 `ORDER BY expr_list`, 除了得到无修饰符的结果（正常情况下的 `limit n`, 前`n`行数据），还会返回与第`n`行具有相同排序字段的行(即如果第`n+1`行的字段与第`n`行 拥有相同的排序字段，同样返回该结果.)

此修饰符可以与：`ORDER BY ... WITH FILL modifier` 组合使用.

例如以下查询：

```
SELECT * FROM (
    SELECT number%50 AS n FROM numbers(100)
) ORDER BY n LIMIT 0,5
```

返回

n
0
0
1
1
2

添加 `WITH TIES` 修饰符后

```
SELECT * FROM (
    SELECT number%50 AS n FROM numbers(100)
) ORDER BY n LIMIT 0,5 WITH TIES
```

则返回了以下的数据行

n
0
0
1
1
2
2

虽然指定了 `LIMIT 5`, 但第6行的 `n` 字段值为 2，与第5行相同，因此也作为满足条件的记录返回。

简而言之，该修饰符可理解为是否增加“并列行”的数据。

``` sql,

```

```

```
LIMIT BY子句 {#limit-by-clause}
```

与查询`LIMIT n BY expressions`子句选择第一个`n`每个不同值的行`expressions`。`LIMIT BY`可以包含任意数量的[表达式](#sql-reference-syntax-md)。

ClickHouse支持以下语法变体：

- `LIMIT [offset\_value, ]n BY expressions`
- `LIMIT n OFFSET offset\_value BY expressions`

在查询处理过程中，ClickHouse会选择按排序键排序的数据。排序键使用以下命令显式设置[ORDER BY](#sql-reference-statements-select-order-by-md)子句或隐式作为表引擎的属性。然后ClickHouse应用`LIMIT n BY expressions`并返回第一`n`每个不同组合的行`expressions`。如果`OFFSET`被指定，则对于每个数据块属于一个不同的组合`expressions`，ClickHouse跳过`offset\_value`从块开始的行数，并返回最大值`n`行的结果。如果`offset\_value`如果数据块中的行数大于数据块中的行数，ClickHouse将从该块返回零行。

!!! note "注"

`LIMIT BY`是不相关的[LIMIT](#sql-reference-statements-select-limit-md)。它们都可以在同一个查询中使用。

```
例 {#examples}
```

样例表：

```
``` sql
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);
```

查询：

```
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

该 `SELECT * FROM limit_by ORDER BY id, val LIMIT 2 OFFSET 1 BY id` 查询返回相同的结果。

以下查询返回每个引用的前5个引用 `domain, device_type` 最多可与100行配对 (`LIMIT n BY + LIMIT`)。

```
SELECT
    domainWithoutWWW(URL) AS domain,
    domainWithoutWWW(REFERER_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

ORDER BY 子句包含一个表达式列表，每个表达式都可以用 DESC (降序) 或 ASC (升序) 修饰符确定排序方向。如果未指定方向，默认是 ASC，所以它通常被省略。排序方向适用于单个表达式，而不适用于整个列表。示例: ORDER BY Visits DESC, SearchPhrase

对于排序表达式列表具有相同值的行以任意顺序输出，也可以是非确定性的（每次都不同）。

如果省略ORDER BY子句，则行的顺序也是未定义的，并且可能也是非确定性的。

特殊值的排序

有两种方法 NaN 和 NULL 排序顺序：

- 默认情况下或与 NULLS LAST 修饰符：首先是值，然后 NaN，然后 NULL.
- 与 NULLS FIRST 修饰符：第一 NULL，然后 NaN，然后其他值。

示例

对于表

x	y
1	NULL
2	2
1	nan
2	2
3	4
5	6
6	nan
7	NULL
6	7
8	9

运行查询 SELECT * FROM t_null_nan ORDER BY y NULLS FIRST 获得：

x	y
1	NULL
7	NULL
1	nan
6	nan
2	2
2	2
3	4
5	6
6	7
8	9

当对浮点数进行排序时，Nan与其他值是分开的。无论排序顺序如何，Nan都在最后。换句话说，对于升序排序，它们被放置为好像它们比所有其他数字大，而对于降序排序，它们被放置为好像它们比其他数字小。

排序规则支持

对于按字符串值排序，可以指定排序规则（比较）。示例: `ORDER BY SearchPhrase COLLATE 'tr'`-对于按关键字升序排序，使用土耳其字母，不区分大小写，假设字符串是UTF-8编码。`COLLATE` 可以按顺序独立地指定或不按每个表达式。如果 `ASC` 或 `DESC` 被指定, `COLLATE` 在它之后指定。使用时 `COLLATE`，排序始终不区分大小写。

我们只建议使用 `COLLATE` 对于少量行的最终排序，因为排序与 `COLLATE` 比正常的按字节排序效率低。

实现细节

更少的RAM使用，如果一个足够小 `LIMIT` 除了指定 `ORDER BY`. 否则，所花费的内存量与用于排序的数据量成正比。对于分布式查询处理，如果 `GROUP BY` 省略排序，在远程服务器上部分完成排序，并将结果合并到请求者服务器上。这意味着对于分布式排序，要排序的数据量可以大于单个服务器上的内存量。

如果没有足够的RAM，则可以在外部存储器中执行排序（在磁盘上创建临时文件）。使用设置 `max_bytes_before_external_sort` 为此目的。如果将其设置为0（默认值），则禁用外部排序。如果启用，则当要排序的数据量达到指定的字节数时，将对收集的数据进行排序并转储到临时文件中。读取所有数据后，将合并所有已排序的文件并输出结果。文件被写入到 `/var/lib/clickhouse/tmp/` 目录中的配置（默认情况下，但你可以使用 `tmp_path` 参数来更改此设置）。

运行查询可能占用的内存比 `max_bytes_before_external_sort` 大. 因此，此设置的值必须大大小于 `max_memory_usage`. 例如，如果您的服务器有128GB的RAM，并且您需要运行单个查询，请设置 `max_memory_usage` 到100GB，和 `max_bytes_before_external_sort` 至80GB。

外部排序的工作效率远远低于在RAM中进行排序。

ORDER BY Expr WITH FILL Modifier

此修饰符可以与 `LIMIT ... WITH TIES modifier` 进行组合使用.

可以在 `ORDER BY expr` 之后用可选的 `FROM expr`, `TO expr` 和 `STEP expr` 参数来设置 `WITH FILL` 修饰符。所有 `expr` 列的缺失值将被顺序填充，而其他列将被填充为默认值。

使用以下语法填充多列，在 `ORDER BY` 部分的每个字段名称后添加带有可选参数的 `WITH FILL` 修饰符。

```
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` 仅适用于具有数字（所有类型的浮点，小数，整数）或日期/日期时间类型的字段。

当未定义 `FROM const_expr` 填充顺序时，则使用 `ORDER BY` 中的最小 `expr` 字段值。

如果未定义 `TO const_expr` 填充顺序，则使用 `ORDER BY` 中的最大 `expr` 字段值。

当定义了 `STEP const_numeric_expr` 时，对于数字类型，`const_numeric_expr` 将 `as is` 解释为 `days` 作为日期类型，将 `seconds` 解释为 `DateTime` 类型。

如果省略了 `STEP const_numeric_expr`，则填充顺序使用 `1.0` 表示数字类型，`1 day` 表示日期类型，`1 second` 表示日期时间类型。

例如下面的查询：

```
SELECT n, source FROM (
  SELECT toFloat32(number % 10) AS n, 'original' AS source
  FROM numbers(10) WHERE number % 3 = 1
) ORDER BY n
```

返回

n	source
1	original
4	original
7	original

但是如果配置了 `WITH FILL` 修饰符

```
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
```

返回

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 when we have multiple fields `ORDER BY field2 WITH FILL, field1 WITH FILL` order of filling will follow the order of fields in `ORDER BY` clause.

对于我们有多个字段 `ORDER BY field2 WITH FILL, field1 WITH FILL` 的情况，填充顺序将遵循 `ORDER BY` 子句中字段的顺序。

示例：

```

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;

```

[返回](#)

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

字段 `d1` 没有填充并使用默认值，因为我们没有 `d2` 值的重复值，并且无法正确计算 `d1` 的顺序。

以下查询中 `ORDER BY` 中的字段将被更改

```

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;

```

[返回](#)

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 子句

`Prewhere` 是更有效地进行过滤的优化。默认情况下，即使在 `PREWHERE` 子句未显式指定。它也会自动移动 `WHERE` 条件到 `prewhere` 阶段。`PREWHERE` 子句只是控制这个优化，如果你认为你知道如何做得比默认情况下更好才去控制它。

使用 `prewhere` 优化，首先只读取执行 `prewhere` 表达式所需的列。然后读取运行其余查询所需的其他列，但只读取 `prewhere` 表达式所在的那些块 “`true`” 至少对于一些行。如果有很多块，其中 `prewhere` 表达式是 “`false`” 对于所有行和 `prewhere` 需要比查询的其他部分更少的列，这通常允许从磁盘读取更少的数据以执行查询。

手动控制Prewhere

该子句具有与 WHERE 相同的含义，区别在于从表中读取数据。当手动控制 PREWHERE 对于查询中的少数列使用的过滤条件，但这些过滤条件提供了强大的数据过滤。这减少了要读取的数据量。

查询可以同时指定 PREWHERE 和 WHERE。在这种情况下，PREWHERE 先于 WHERE。

如果 optimize_move_to_prewhere 设置为 0，启发式自动移动部分表达式 WHERE 到 PREWHERE 被禁用。

限制

PREWHERE 只有支持 *MergeTree 族系列引擎的表。

采样子句

该 SAMPLE 子句允许近似于 SELECT 查询处理。

启用数据采样时，不会对所有数据执行查询，而只对特定部分数据（样本）执行查询。例如，如果您需要计算所有访问的统计信息，只需对所有访问的 1/10 分数执行查询，然后将结果乘以 10 即可。

近似查询处理在以下情况下可能很有用：

- 当你有严格的时间需求（如 <100ms），但你不能通过额外的硬件资源来满足他们的成本。
- 当您的原始数据不准确时，所以近似不会明显降低质量。
- 业务需求的目标是近似结果（为了成本效益，或者向高级用户推销确切结果）。

注

您只能使用采样中的表 MergeTree 族，并且只有在表创建过程中指定了采样表达式（请参阅 [MergeTree 引擎](#)）。

下面列出了数据采样的功能：

- 数据采样是一种确定性机制。同样的结果 SELECT .. SAMPLE 查询始终是相同的。
- 对于不同的表，采样工作始终如一。对于具有单个采样键的表，具有相同系数的采样总是选择相同的可能数据子集。例如，用户 Id 的示例采用来自不同表的所有可能的用户 Id 的相同子集的行。这意味着您可以在子查询中使用采样 IN 此外，您可以使用 JOIN。
- 采样允许从磁盘读取更少的数据。请注意，您必须正确指定采样键。有关详细信息，请参阅 [创建 MergeTree 表](#)。

为 SAMPLE 子句支持以下语法：

SAMPLE Clause Syntax

产品描述

SAMPLE k

这里 k 是从 0 到 1 的数字。

查询执行于 k 数据的分数。例如，SAMPLE 0.1 对 10% 的数据运行查询。Read more SAMPLE n 这里 n 是足够大的整数。该查询是在至少一个样本上执行的 n 行（但不超过这个）。例如，SAMPLE 10000000 在至少 10,000,000 行上运行查询。Read more SAMPLE k OFFSET m 这里 k 和 m 是从 0 到 1 的数字。查询在以下示例上执行 k 数据的分数。用于采样的数据由以下偏移 m 分数。Read more

SAMPLE K

这里 k 从 0 到 1 的数字（支持小数和小数表示法）。例如，SAMPLE 1/2 或 SAMPLE 0.5.

在一个 SAMPLE k 子句，样品是从 k 数据的分数。示例如下所示：

```
SELECT
    Title,
    count() * 10 AS PageViews
FROM hits_distributed
SAMPLE 0.1
WHERE
    CounterID = 34
GROUP BY Title
ORDER BY PageViews DESC LIMIT 1000
```

在此示例中，对 0.1(10%) 数据的样本执行查询。聚合函数的值不会自动修正，因此要获得近似结果，值 count() 手动乘以 10。

SAMPLE N

这里 n 是足够大的整数。例如，SAMPLE 10000000。

在这种情况下，查询在至少一个样本上执行 n 行（但不超过这个）。例如，SAMPLE 10000000 在至少 10,000,000 行上运行查询。

由于数据读取的最小单位是一个颗粒（其大小由 index_granularity 设置），是有意义的设置一个样品，其大小远大于颗粒。

使用时 SAMPLE n 子句，你不知道处理了哪些数据的相对百分比。所以你不知道聚合函数应该乘以的系数。使用 _sample_factor 虚拟列得到近似结果。

该 _sample_factor 列包含动态计算的相对系数。当您执行以下操作时，将自动创建此列 创建 具有指定采样键的表。的使用示例 _sample_factor 列如下所示。

让我们考虑表 visits，其中包含有关网站访问的统计信息。第一个示例演示如何计算页面浏览量：

```
SELECT sum(PageViews * _sample_factor)
FROM visits
SAMPLE 10000000
```

下一个示例演示如何计算访问总数：

```
SELECT sum(_sample_factor)
FROM visits
SAMPLE 10000000
```

下面的示例显示了如何计算平均会话持续时间。请注意，您不需要使用相对系数来计算平均值。

```
SELECT avg(Duration)
FROM visits
SAMPLE 10000000
```

SAMPLE K OFFSET M

这里 k 和 m 是从 0 到 1 的数字。示例如下所示。

示例 1

```
SAMPLE 1/10
```

在此示例中，示例是所有数据的十分之一：

[+-----]

示例2

```
SAMPLE 1/10 OFFSET 1/2
```

这里，从数据的后半部分取出10%的样本。

[-----+-----]

UNION ALL子句

你可以使用 UNION ALL 结合任意数量的 SELECT 来扩展其结果。示例：

```
SELECT CounterID, 1 AS table, tolnt64(count()) AS c
  FROM test.hits
 GROUP BY CounterID

UNION ALL

SELECT CounterID, 2 AS table, sum(Sign) AS c
  FROM test.visits
 GROUP BY CounterID
 HAVING c > 0
```

结果列通过它们的索引进行匹配（在内部的顺序 SELECT）。如果列名称不匹配，则从第一个查询中获取最终结果的名称。

对联合执行类型转换。例如，如果合并的两个查询具有相同的字段与非-Nullable 和 Nullable 从兼容类型的类型，由此产生的 UNION ALL 有一个 Nullable 类型字段。

属于以下部分的查询 UNION ALL 不能用圆括号括起来。 ORDER BY 和 LIMIT 应用于单独的查询，而不是最终结果。如果您需要将转换应用于最终结果，则可以将所有查询 UNION ALL 在子查询中 FROM 子句。

限制

只有 UNION ALL 支持。 UNION (UNION DISTINCT) 不支持。如果你需要 UNION DISTINCT，你可以写 SELECT DISTINCT 子查询中包含 UNION ALL.

实现细节

属于 UNION ALL 的查询可以同时运行，并且它们的结果可以混合在一起。

WHERE

WHERE 子句允许过滤从 FROM 子句 SELECT.

如果有一个 WHERE 子句，它必须包含一个表达式与 UInt8 类型。这通常是一个带有比较和逻辑运算符的表达式。此表达式计算结果为0的行将从进一步的转换或结果中解释出来。

WHERE 如果基础表引擎支持，则根据使用索引和分区修剪的能力评估表达式。

注

有一个叫做过滤优化 prewhere 的东西。

WITH子句

本节提供对公共表表达式的支持 (CTE) ，所以结果 WITH 子句可以在其余部分中使用 SELECT 查询。

限制

1. 不支持递归查询。
2. 当在section中使用子查询时，它的结果应该是只有一行的标量。
3. Expression的结果在子查询中不可用。

例

示例1： 使用常量表达式作为“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
```

示例2： 从SELECT子句列表中逐出sum(bytes)表达式结果

```
WITH sum(bytes) as s
SELECT
    formatReadableSize(s),
    table
FROM system.parts
GROUP BY table
ORDER BY s
```

例3： 使用标量子查询的结果

```
/* 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
```

例4： 在子查询中重用表达式

作为子查询中表达式使用的当前限制的解决方法，您可以复制它。

```
WITH ['hello'] AS hello
SELECT
    hello,
    *
FROM
(
    WITH ['hello'] AS hello
    SELECT hello
)
```

```
hello      hello  
['hello'] | ['hello'] |
```

CREATE语法

CREATE语法包含以下子集:

- **DATABASE**

CREATE DATABASE

创建数据库.

```
CREATE DATABASE [IF NOT EXISTS] db_name [ON CLUSTER cluster] [ENGINE = engine(...)]
```

条件

IF NOT EXISTS

如果db_name数据库已经存在，则ClickHouse不会创建新数据库并且：

- 如果指定了子句，则不会引发异常。
- 如果未指定子句，则抛出异常。

ON CLUSTER

ClickHouse在指定集群的所有服务器上创建db_name数据库。更多细节在 [Distributed DDL article](#).

ENGINE

[MySQL](#) 允许您从远程MySQL服务器检索数据。默认情况下，ClickHouse使用自己的[database engine](#). 还有一个[lazy](#)引擎。

CREATE TABLE

Creates a new table. This query can have various syntax forms depending on a use case.

By default, tables are created only on the current server. Distributed DDL queries are implemented as [ON CLUSTER clause](#), which is [described separately](#).

Syntax Forms

With Explicit Schema

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]  
(  
    name1 [type1] [NULL|NOT NULL] [DEFAULT|MATERIALIZED|ALIAS expr1] [compression_codec] [TTL expr1],  
    name2 [type2] [NULL|NOT NULL] [DEFAULT|MATERIALIZED|ALIAS expr2] [compression_codec] [TTL expr2],  
    ...  
) ENGINE = engine
```

Creates a table named `name` in the `db` database or the current database if `db` is not set, with the structure specified in brackets and the `engine` engine.

The structure of the table is a list of column descriptions, secondary indexes and constraints . If **primary key** is supported by the engine, it will be indicated as parameter for the table engine.

A column description is `name type` in the simplest case. Example: `RegionID UInt32`.

Expressions can also be defined for default values (see below).

If necessary, primary key can be specified, with one or more key expressions.

With a Schema Similar to Other Table

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name AS [db2.]name2 [ENGINE = engine]
```

Creates a table with the same structure as another table. You can specify a different engine for the table. If the engine is not specified, the same engine will be used as for the `db2.name2` table.

From a Table Function

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name AS table_function()
```

Creates a table with the same result as that of the **table function** specified. The created table will also work in the same way as the corresponding table function that was specified.

From SELECT query

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name[(name1 [type1], name2 [type2], ...)] ENGINE = engine AS SELECT ...
```

Creates a table with a structure like the result of the `SELECT` query, with the `engine` engine, and fills it with data from `SELECT`. Also you can explicitly specify columns description.

If the table already exists and `IF NOT EXISTS` is specified, the query won't do anything.

There can be other clauses after the `ENGINE` clause in the query. See detailed documentation on how to create tables in the descriptions of **table engines**.

Example

Query:

```
CREATE TABLE t1 (x String) ENGINE = Memory AS SELECT 1;
SELECT x, toTypeName(x) FROM t1;
```

Result:

x	toTypeName(x)
1	String

NULL Or NOT NULL Modifiers

`NULL` and `NOT NULL` modifiers after data type in column definition allow or do not allow it to be **Nullable**.

If the type is not `Nullable` and if `NULL` is specified, it will be treated as `Nullable`; if `NOT NULL` is specified, then no. For example, `INT NULL` is the same as `Nullable(INT)`. If the type is `Nullable` and `NULL` or `NOT NULL` modifiers are specified, the exception will be thrown.

See also [data_type_default_nullable](#) setting.

Default Values

The column description can specify an expression for a default value, in one of the following ways: `DEFAULT expr`, `MATERIALIZED expr`, `ALIAS expr`.

Example: `URLDomain String DEFAULT domain(URL)`.

If an expression for the default value is not defined, the default values will be set to zeros for numbers, empty strings for strings, empty arrays for arrays, and `1970-01-01` for dates or zero unix timestamp for `DateTime`, `NULL` for `Nullable`.

If the default expression is defined, the column type is optional. If there isn't an explicitly defined type, the default expression type is used. Example: `EventDate DEFAULT toDate(EventTime)` – the 'Date' type will be used for the 'EventDate' column.

If the data type and default expression are defined explicitly, this expression will be cast to the specified type using type casting functions. Example: `Hits UInt32 DEFAULT 0` means the same thing as `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 do not contain loops. For `INSERT`, it checks that expressions are resolvable – that all columns they can be calculated from have been passed.

DEFAULT

`DEFAULT expr`

Normal default value. If the `INSERT` query does not specify the corresponding column, it will be filled in by computing the corresponding expression.

MATERIALIZED

`MATERIALIZED expr`

Materialized expression. Such a column can't be specified for `INSERT`, because it is always calculated. For an `INSERT` without a list of columns, these columns are not considered. In addition, this column is not substituted when using an asterisk in a `SELECT` query. This is to preserve the invariant that the dump obtained using `SELECT *` can be inserted back into the table using `INSERT` without specifying the list of columns.

ALIAS

`ALIAS expr`

Synonym. Such a column isn't stored in the table at all.

Its values can't be inserted in a table, and it is not substituted when using an asterisk in a `SELECT` query. It can be used in `SELECT`s if the alias is expanded during query parsing.

When using the `ALTER` query to add new columns, old data for these columns is not written. Instead, when reading old data that does not have values for the new columns, expressions are computed on the fly by default. However, if running the expressions requires different columns that are not indicated in the query, these columns will additionally be read, but only for the blocks of data that need it.

If you add a new column to a table but later change its default expression, the values used for old data will change (for data where values were not stored on the disk). Note that when running background merges, data for columns that are missing in one of the merging parts is written to the merged part.

It is not possible to set default values for elements in nested data structures.

Primary Key

You can define a [primary key](#) when creating a table. Primary key can be specified in two ways:

- Inside the column list

```
CREATE TABLE db.table_name
(
    name1 type1, name2 type2, ...,
    PRIMARY KEY(expr1[, expr2,...])
)
ENGINE = engine;
```

- Outside the column list

```
CREATE TABLE db.table_name
(
    name1 type1, name2 type2, ...
)
ENGINE = engine
PRIMARY KEY(expr1[, expr2,...]);
```

Warning

You can't combine both ways in one query.

Constraints

Along with columns descriptions constraints could be defined:

```
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` could by any boolean expression. If constraints are defined for the table, each of them will be checked for every row in `INSERT` query. If any constraint is not satisfied — server will raise an exception with constraint name and checking expression.

Adding large amount of constraints can negatively affect performance of big `INSERT` queries.

TTL Expression

Defines storage time for values. Can be specified only for MergeTree-family tables. For the detailed description, see [TTL for columns and tables](#).

Column Compression Codecs

By default, ClickHouse applies the `lz4` compression method. For `MergeTree`-engine family you can change the default compression method in the [compression](#) section of a server configuration.

You can also define the compression method for each individual column in the [CREATE TABLE](#) query.

```
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>
...
```

The `Default` codec can be specified to reference default compression which may depend on different settings (and properties of data) in runtime.

Example: `value UInt64 CODEC(Default)` — the same as lack of codec specification.

Also you can remove current CODEC from the column and use default compression from config.xml:

```
ALTER TABLE codec_example MODIFY COLUMN float_value CODEC(Default);
```

Codecs can be combined in a pipeline, for example, `CODEC(Delta, Default)`.

Warning

You can't decompress ClickHouse database files with external utilities like `lz4`. Instead, use the special [clickhouse-compressor](#) utility.

Compression is supported for the following table engines:

- [MergeTree](#) family. Supports column compression codecs and selecting the default compression method by [compression](#) settings.
- [Log](#) family. Uses the `lz4` compression method by default and supports column compression codecs.
- [Set](#). Only supported the default compression.
- [Join](#). Only supported the default compression.

ClickHouse supports general purpose codecs and specialized codecs.

General Purpose Codecs

Codecs:

- `NONE` — No compression.
- `LZ4` — Lossless [data compression algorithm](#) used by default. Applies LZ4 fast compression.
- `LZ4HC[(level)]` — LZ4 HC (high compression) algorithm with configurable level. Default level: 9. Setting `level <= 0` applies the default level. Possible levels: [1, 12]. Recommended level range: [4, 9].
- `ZSTD[(level)]` — [ZSTD compression algorithm](#) with configurable `level`. Possible levels: [1, 22]. Default value: 1.

High compression levels are useful for asymmetric scenarios, like compress once, decompress repeatedly. Higher levels mean better compression and higher CPU usage.

Specialized Codecs

These codecs are designed to make compression more effective by using specific features of data. Some of these codecs do not compress data themselves. Instead, they prepare the data for a common purpose codec, which compresses it better than without this preparation.

Specialized codecs:

- **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` are used for storing delta values, so `delta_bytes` is the maximum size of raw values. Possible `delta_bytes` values: 1, 2, 4, 8. The default value for `delta_bytes` is `sizeof(type)` if equal to 1, 2, 4, or 8. In all other cases, it's 1.
- **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: A Fast, Scalable, In-Memory Time Series Database](#).
- **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: A Fast, Scalable, In-Memory Time Series Database](#).
- **T64** — Compression approach that crops unused high bits of values in integer data types (including `Enum`, `Date` and `DateTime`). At each step of its algorithm, codec takes a block of 64 values, puts them into 64x64 bit matrix, transposes it, crops the unused bits of values and returns the rest as a sequence. Unused bits are the bits, that do not differ between maximum and minimum values in the whole data part for which the compression is used.

`DoubleDelta` and `Gorilla` codecs are used in Gorilla TSDB as the components of its compressing algorithm. Gorilla approach is effective in scenarios when there is a sequence of slowly changing values with their timestamps. Timestamps are effectively compressed by the `DoubleDelta` codec, and values are effectively compressed by the `Gorilla` codec. For example, to get an effectively stored table, you can create it in the following configuration:

```
CREATE TABLE codec_example
(
    timestamp DateTime CODEC(DoubleDelta),
    slow_values Float32 CODEC(Gorilla)
)
ENGINE = MergeTree()
```

Temporary Tables

ClickHouse supports temporary tables which have the following characteristics:

- Temporary tables disappear when the session ends, including if the connection is lost.
- A temporary table uses the Memory engine only.
- The DB can't be specified for a temporary table. It is created outside of databases.

- Impossible to create a temporary table with distributed DDL query on all cluster servers (by using `ON CLUSTER`): this table exists only in the current session.
- If a temporary table has the same name as another one and a query specifies the table name without specifying the DB, the temporary table will be used.
- For distributed query processing, temporary tables used in a query are passed to remote servers.

To create a temporary table, use the following syntax:

```
CREATE TEMPORARY TABLE [IF NOT EXISTS] table_name
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
)
```

In most cases, temporary tables are not created manually, but when using external data for a query, or for distributed (`GLOBAL`) `IN`. For more information, see the appropriate sections

It's possible to use tables with `ENGINE = Memory` instead of temporary tables.

REPLACE TABLE

'`REPLACE`' query allows you to update the table atomically.

Note

This query is supported only for **Atomic** database engine.

If you need to delete some data from a table, you can create a new table and fill it with a `SELECT` statement that does not retrieve unwanted data, then drop the old table and rename the new one:

```
CREATE TABLE myNewTable AS myOldTable;
INSERT INTO myNewTable SELECT * FROM myOldTable WHERE CounterID <12345;
DROP TABLE myOldTable;
RENAME TABLE myNewTable TO myOldTable;
```

Instead of above, you can use the following:

```
REPLACE TABLE myOldTable SELECT * FROM myOldTable WHERE CounterID <12345;
```

Syntax

```
{CREATE [OR REPLACE] | REPLACE} TABLE [db.]table_name
```

All syntax forms for `CREATE` query also work for this query. `REPLACE` for a non-existent table will cause an error.

Examples:

Consider the table:

```
CREATE DATABASE base ENGINE = Atomic;
CREATE OR REPLACE TABLE base.t1 (n UInt64, s String) ENGINE = MergeTree ORDER BY n;
INSERT INTO base.t1 VALUES (1, 'test');
SELECT * FROM base.t1;
```

n	s
1	test

Using `REPLACE` query to clear all data:

```
CREATE OR REPLACE TABLE base.t1 (n UInt64, s Nullable(String)) ENGINE = MergeTree ORDER BY n;
INSERT INTO base.t1 VALUES (2, null);
SELECT * FROM base.t1;
```

n	s
2	\N

Using `REPLACE` query to change table structure:

```
REPLACE TABLE base.t1 (n UInt64) ENGINE = MergeTree ORDER BY n;
INSERT INTO base.t1 VALUES (3);
SELECT * FROM base.t1;
```

n
3

COMMENT Clause

You can add a comment to the table when you creating it.

Note

The comment is supported for all table engines except **Kafka**, **RabbitMQ** and **EmbeddedRocksDB**.

Syntax

```
CREATE TABLE db.table_name
(
    name1 type1, name2 type2, ...
)
ENGINE = engine
COMMENT 'Comment'
```

Example

Query:

```
CREATE TABLE t1 (x String) ENGINE = Memory COMMENT 'The temporary table';
SELECT name, comment FROM system.tables WHERE name = 't1';
```

Result:

name	comment
t1	The temporary table

CREATE VIEW

创建一个新视图。有两种类型的视图：普通视图和物化视图。

Normal

语法：

```
CREATE [OR REPLACE] VIEW [IF NOT EXISTS] [db.]table_name [ON CLUSTER] AS SELECT ...
```

普通视图不存储任何数据。他们只是在每次访问时从另一个表执行读取。换句话说，普通视图只不过是一个保存的查询。从视图中读取时，此保存的查询用作**FROM**子句中的子查询。

例如，假设您已经创建了一个视图：

```
CREATE VIEW view AS SELECT ...
```

并写了一个查询：

```
SELECT a, b, c FROM view
```

这个查询完全等同于使用子查询：

```
SELECT a, b, c FROM (SELECT ...) view
```

Materialized

```
CREATE MATERIALIZED VIEW [IF NOT EXISTS] [db.]table_name [ON CLUSTER] [TO[db.]name] [ENGINE = engine] [POPULATE] AS SELECT ...
```

物化视图存储由相应的**SELECT**管理。

创建不带**TO [db].[table]**的物化视图时，必须指定**ENGINE** – 用于存储数据的表引擎。

使用**TO [db].[table]** 创建物化视图时，不得使用**POPULATE**。

一个物化视图的实现是这样的：当向**SELECT**中指定的表插入数据时，插入数据的一部分被这个**SELECT**查询转换，结果插入到视图中。

重要

ClickHouse 中的物化视图更像是插入触发器。如果视图查询中有一些聚合，则它仅应用于一批新插入的数据。对源表现有数据的任何更改（如更新、删除、删除分区等）都不会更改物化视图。

如果指定**POPULATE**，则在创建视图时将现有表数据插入到视图中，就像创建一个**CREATE TABLE ... AS SELECT ...**一样。否则，查询仅包含创建视图后插入表中的数据。我们不建议使用**POPULATE**，因为在创建视图期间插入表中的数据不会插入其中。

`SELECT` 查询可以包含`DISTINCT`、`GROUP BY`、`ORDER BY`、`LIMIT`……请注意，相应的转换是在每个插入数据块上独立执行的。例如，如果设置了`GROUP BY`，则在插入期间聚合数据，但仅在插入数据的单个数据包内。数据不会被进一步聚合。例外情况是使用独立执行数据聚合的`ENGINE`，例如`SummingMergeTree`。

在物化视图上执行`ALTER`查询有局限性，因此可能不方便。如果物化视图使用构造`TO [db.]name`，你可以`DETACH`视图，为目标表运行`ALTER`，然后`ATTACH`先前分离的（`DETACH`）视图。

请注意，物化视图受`optimize_on_insert`设置的影响。在插入视图之前合并数据。

视图看起来与普通表相同。例如，它们列在`SHOW TABLES`查询的结果中。

删除视图，使用`DROP VIEW`。`DROP TABLE`也适用于视图。

Live View (实验性)

重要

这是一项实验性功能，可能会在未来版本中以向后不兼容的方式进行更改。

使用`allow_experimental_live_view`设置启用实时视图和`WATCH`查询的使用。输入命令`set allow_experimental_live_view = 1`。

```
CREATE LIVE VIEW [IF NOT EXISTS] [db.]table_name [WITH [TIMEOUT [value_in_sec] [AND]] [REFRESH [value_in_sec]]] AS SELECT ...
```

实时视图存储相应`SELECT`查询的结果，并在查询结果更改时随时更新。查询结果以及与新数据结合所需的部分结果存储在内存中，为重复查询提供更高的性能。当使用`WATCH`查询更改查询结果时，实时视图可以提供推送通知。

实时视图是通过插入到查询中指定的最里面的表来触发的。

实时视图的工作方式类似于分布式表中查询的工作方式。但不是组合来自不同服务器的部分结果，而是将当前数据的部分结果与新数据的部分结果组合在一起。当实时视图查询包含子查询时，缓存的部分结果仅存储在最里面的子查询中。

限制

- `Table function`不支持作为最里面的表。
- 没有插入的表，例如`dictionary`, `system table`, `normal view`, `materialized view`不会触发实时视图。
- 只有可以将旧数据的部分结果与新数据的部分结果结合起来的查询才有效。实时视图不适用于需要完整数据集来计算最终结果或必须保留聚合状态的聚合的查询。
- 不适用于在不同节点上执行插入的复制或分布式表。
- 不能被多个表触发。

`WITH REFRESH`强制定期更新实时视图，在某些情况下可以用作解决方法。

Monitoring Changes

您可以使用`WATCH`命令监视`LIVE VIEW`查询结果的变化

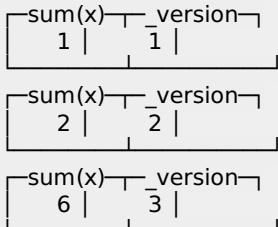
```
WATCH [db.]live_view
```

示例：

```
CREATE TABLE mt (x Int8) Engine = MergeTree ORDER BY x;
CREATE LIVE VIEW lv AS SELECT sum(x) FROM mt;
```

在对源表进行并行插入时观看实时视图。

```
WATCH lv;
```



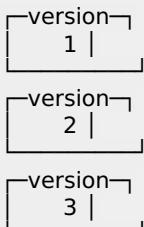
```
INSERT INTO mt VALUES (1);
INSERT INTO mt VALUES (2);
INSERT INTO mt VALUES (3);
```

添加**EVENTS**子句只获取更改事件。

```
WATCH [db.]live_view EVENTS;
```

示例：

```
WATCH lv EVENTS;
```



你可以执行**SELECT**与任何常规视图或表格相同的方式查询实时视图。如果查询结果被缓存，它将立即返回结果而不在基础表上运行存储的查询。

```
SELECT * FROM [db.]live_view WHERE ...
```

Force Refresh

您可以使用`ALTER LIVE VIEW [db.]table_name REFRESH`语法。

WITH TIMEOUT 条件

当使用**WITH TIMEOUT**子句创建实时视图时，**WATCH**观察实时视图的查询。

```
CREATE LIVE VIEW [db.]table_name WITH TIMEOUT [value_in_sec] AS SELECT ...
```

如果未指定超时值，则由指定的值temporary_live_view_timeout决定。

示例：

```
CREATE TABLE mt (x Int8) Engine = MergeTree ORDER BY x;
CREATE LIVE VIEW lv WITH TIMEOUT 15 AS SELECT sum(x) FROM mt;
```

WITH REFRESH条件

当使用WITH REFRESH子句创建实时视图时，它将在自上次刷新或触发后经过指定的秒数后自动刷新。

```
CREATE LIVE VIEW [db.]table_name WITH REFRESH [value_in_sec] AS SELECT ...
```

如果未指定刷新值，则由指定的值periodic_live_view_refresh决定。

示例：

```
CREATE LIVE VIEW lv WITH REFRESH 5 AS SELECT now();
WATCH lv
```

now()	_version
2021-02-21 08:47:05	1
2021-02-21 08:47:10	2
2021-02-21 08:47:15	3

您可以使用AND子句组合WITH TIMEOUT和WITH REFRESH子句。

```
CREATE LIVE VIEW [db.]table_name WITH TIMEOUT [value_in_sec] AND REFRESH [value_in_sec] AS SELECT ...
```

示例：

```
CREATE LIVE VIEW lv WITH TIMEOUT 15 AND REFRESH 5 AS SELECT now();
```

15 秒后，如果没有活动的WATCH查询，实时视图将自动删除。

```
WATCH lv
```

```
Code: 60. DB::Exception: Received from localhost:9000. DB::Exception: Table default.lv does not exist..
```

Usage

实时视图表的最常见用途包括：

- 为查询结果更改提供推送通知以避免轮询。
- 缓存最频繁查询的结果以提供即时查询结果。
- 监视表更改并触发后续选择查询。
- 使用定期刷新从系统表中查看指标。

CREATE DICTIONARY

Creates a new [external dictionary](#) with given [structure](#), [source](#), [layout](#) and [lifetime](#).

Syntax:

```
CREATE DICTIONARY [IF NOT EXISTS] [db.]dictionary_name [ON CLUSTER cluster]
(
    key1 type1 [DEFAULT|EXPRESSION expr1] [IS_OBJECT_ID],
    key2 type2 [DEFAULT|EXPRESSION expr2],
    attr1 type2 [DEFAULT|EXPRESSION expr3] [HIERARCHICAL|INJECTIVE],
    attr2 type2 [DEFAULT|EXPRESSION expr4] [HIERARCHICAL|INJECTIVE]
)
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})
```

External dictionary structure consists of attributes. Dictionary attributes are specified similarly to table columns. The only required attribute property is its type, all other properties may have default values.

`ON CLUSTER` clause allows creating dictionary on a cluster, see [Distributed DDL](#).

Depending on dictionary [layout](#) one or more attributes can be specified as dictionary keys.

For more information, see [External Dictionaries](#) section.

CREATE FUNCTION

Creates a user defined function from a lambda expression. The expression must consist of function parameters, constants, operators, or other function calls.

Syntax

```
CREATE FUNCTION name AS (parameter0, ...) -> expression
```

A function can have an arbitrary number of parameters.

There are a few restrictions:

- The name of a function must be unique among user defined and system functions.
- Recursive functions are not allowed.
- All variables used by a function must be specified in its parameter list.

If any restriction is violated then an exception is raised.

Example

Query:

```
CREATE FUNCTION linear_equation AS (x, k, b) -> k*x + b;
SELECT number, linear_equation(number, 2, 1) FROM numbers(3);
```

Result:

number	<code>plus(multiply(2, number), 1)</code>
0	1
1	3
2	5

A **conditional function** is called in a user defined function in the following query:

```
CREATE FUNCTION parity_str AS (n) -> if(n % 2, 'odd', 'even');
SELECT number, parity_str(number) FROM numbers(3);
```

Result:

number	<code>if(modulo(number, 2), 'odd', 'even')</code>
0	even
1	odd
2	even

CREATE USER

Creates **user accounts**.

Syntax:

```
CREATE USER [IF NOT EXISTS | OR REPLACE] name1 [ON CLUSTER cluster_name1]
[, name2 [ON CLUSTER cluster_name2] ...]
[NOT IDENTIFIED | IDENTIFIED {[WITH {no_password | plaintext_password | sha256_password | sha256_hash |
double_sha1_password | double_sha1_hash} ] BY {'password' | 'hash'}} | {WITH Idap SERVER 'server_name'} | {WITH
kerberos [REALM 'realm']} ]
[HOST {LOCAL | NAME 'name' | REGEXP 'name_regexp' | IP 'address' | LIKE 'pattern'} [...]| ANY | NONE]
[DEFAULT ROLE role [...]]
[DEFAULT DATABASE database | NONE]
[GRANTEES {user | role | ANY | NONE} [...] [EXCEPT {user | role} [...]]]
[SETTINGS variable [= value] [MIN [=] min_value] [MAX [=] max_value] [READONLY | WRITABLE] | PROFILE
'profile_name'] [...]
```

ON CLUSTER clause allows creating users on a cluster, see [Distributed DDL](#).

Identification

There are multiple ways of user identification:

- IDENTIFIED WITH no_password
- IDENTIFIED WITH plaintext_password BY 'qwerty'
- IDENTIFIED WITH sha256_password BY 'qwerty' or IDENTIFIED BY 'password'
- IDENTIFIED WITH sha256_hash BY 'hash'
- IDENTIFIED WITH double_sha1_password BY 'qwerty'
- IDENTIFIED WITH double_sha1_hash BY 'hash'
- IDENTIFIED WITH Idap SERVER 'server_name'
- IDENTIFIED WITH kerberos or IDENTIFIED WITH kerberos REALM 'realm'

User Host

User host is a host from which a connection to ClickHouse server could be established. The host can be specified in the `HOST` query section in the following ways:

- `HOST IP 'ip_address_or_subnetwork'` — User can connect to ClickHouse server only from the specified IP address or a **subnetwork**. Examples: `HOST IP '192.168.0.0/16'`, `HOST IP '2001:DB8::/32'`. For use in production, only specify `HOST IP` elements (IP addresses and their masks), since using `host` and `host_regexp` might cause extra latency.
- `HOST ANY` — User can connect from any location. This is a 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** regular expressions when specifying user hosts. For example, `HOST NAME REGEXP '.*\.mysite\.com'`.
- `HOST LIKE 'template'` — Allows you to use the **LIKE** operator to filter the user hosts. For example, `HOST LIKE '%'` is equivalent to `HOST ANY`, `HOST LIKE '%.mysite.com'` filters all the hosts in the `mysite.com` domain.

Another way of specifying host is to use `@` syntax following the username. Examples:

- `CREATE USER mira@'127.0.0.1'` — Equivalent to the `HOST IP` syntax.
- `CREATE USER mira@'localhost'` — Equivalent to the `HOST LOCAL` syntax.
- `CREATE USER mira@'192.168.%.%'` — Equivalent to the `HOST LIKE` syntax.

Warning

ClickHouse treats `user_name@'address'` as a username as a whole. Thus, technically you can create multiple users with the same `user_name` and different constructions after `@`. However, we do not recommend to do so.

GRANTEES Clause

Specifies users or roles which are allowed to receive **privileges** from this user on the condition this user has also all required access granted with **GRANT OPTION**. Options of the `GRANTEES` clause:

- `user` — Specifies a user this user can grant privileges to.
- `role` — Specifies a role this user can grant privileges to.
- `ANY` — This user can grant privileges to anyone. It's the default setting.
- `NONE` — This user can grant privileges to none.

You can exclude any user or role by using the `EXCEPT` expression. For example, `CREATE USER user1 GRANTEES ANY EXCEPT user2`. It means if `user1` has some privileges granted with `GRANT OPTION` it will be able to grant those privileges to anyone except `user2`.

Examples

Create the user account `mira` protected by the password `qwerty`:

```
CREATE USER mira HOST IP '127.0.0.1' IDENTIFIED WITH sha256_password BY 'qwerty';
```

`mira` should start client app at the host where the ClickHouse server runs.

Create the user account `john`, assign roles to it and make this roles default:

```
CREATE USER john DEFAULT ROLE role1, role2;
```

Create the user account `john` and make all his future roles default:

```
CREATE USER john DEFAULT ROLE ALL;
```

When some role is assigned to `john` in the future, it will become default automatically.

Create the user account `john` and make all his future roles default excepting `role1` and `role2`:

```
CREATE USER john DEFAULT ROLE ALL EXCEPT role1, role2;
```

Create the user account `john` and allow him to grant his privileges to the user with `jack` account:

```
CREATE USER john GRANTEEES jack;
```

CREATE ROLE

Creates new **roles**. Role is a set of **privileges**. A **user** assigned a role gets all the privileges of this role.

Syntax:

```
CREATE ROLE [IF NOT EXISTS | OR REPLACE] name1 [, name2 ...]
[SETTINGS variable [= value] [MIN [=] min_value] [MAX [=] max_value] [READONLY|WRITABLE] | PROFILE
'profile_name'] [...]
```

Managing Roles

A user can be assigned multiple roles. Users can apply their assigned roles in arbitrary combinations by the **SET ROLE** statement. The final scope of privileges is a combined set of all the privileges of all the applied roles. If a user has privileges granted directly to its user account, they are also combined with the privileges granted by roles.

User can have default roles which apply at user login. To set default roles, use the **SET DEFAULT ROLE** statement or the **ALTER USER** statement.

To revoke a role, use the **REVOKE** statement.

To delete role, use the **DROP ROLE** statement. The deleted role is being automatically revoked from all the users and roles to which it was assigned.

Examples

```
CREATE ROLE accountant;
GRANT SELECT ON db.* TO accountant;
```

This sequence of queries creates the role `accountant` that has the privilege of reading data from the accounting database.

Assigning the role to the user `mira`:

```
GRANT accountant TO mira;
```

After the role is assigned, the user can apply it and execute the allowed queries. For example:

```
SET ROLE accountant;
SELECT * FROM db.*;
```

CREATE ROW POLICY

Creates a **row policy**, i.e. a filter used to determine which rows a user can read from a table.

Warning

Row policies makes sense only for users with readonly access. If user can modify table or copy partitions between tables, it defeats the restrictions of row policies.

Syntax:

```
CREATE [ROW] POLICY [IF NOT EXISTS | OR REPLACE] policy_name1 [ON CLUSTER cluster_name1] ON [db1.]table1
[, policy_name2 [ON CLUSTER cluster_name2] ON [db2.]table2 ...]
[FOR SELECT] USING condition
[AS {PERMISSIVE | RESTRICTIVE}]
[TO {role1 [, role2 ...] | ALL | ALL EXCEPT role1 [, role2 ...]}]
```

USING Clause

Allows to specify a condition to filter rows. An user will see a row if the condition is calculated to non-zero for the row.

TO Clause

In the section **TO** you can provide a list of users and roles this policy should work for. For example, `CREATE ROW POLICY ... TO accountant, john@localhost`.

Keyword **ALL** means all the ClickHouse users including current user. Keyword **ALL EXCEPT** allow to exclude some users from the all users list, for example, `CREATE ROW POLICY ... TO ALL EXCEPT accountant, john@localhost`

Note

If there are no row policies defined for a table then any user can **SELECT** all the row from the table. Defining one or more row policies for the table makes the access to the table depending on the row policies no matter if those row policies are defined for the current user or not. For example, the following policy

```
CREATE ROW POLICY pol1 ON mydb.table1 USING b=1 TO mira, peter
```

forbids the users **mira** and **peter** to see the rows with **b != 1**, and any non-mentioned user (e.g., the user **paul**) will see no rows from **mydb.table1** at all.

If that's not desirable it can't be fixed by adding one more row policy, like the following:

```
CREATE ROW POLICY pol2 ON mydb.table1 USING 1 TO ALL EXCEPT mira, peter
```

AS Clause

It's allowed to have more than one policy enabled on the same table for the same user at the one time. So we need a way to combine the conditions from multiple policies.

By default policies are combined using the boolean `OR` operator. For example, the following policies

```
CREATE ROW POLICY pol1 ON mydb.table1 USING b=1 TO mira, peter
CREATE ROW POLICY pol2 ON mydb.table1 USING c=2 TO peter, antonio
```

enables the user `peter` to see rows with either `b=1` or `c=2`.

The `AS` clause specifies how policies should be combined with other policies. Policies can be either permissive or restrictive. By default policies are permissive, which means they are combined using the boolean `OR` operator.

A policy can be defined as restrictive as an alternative. Restrictive policies are combined using the boolean `AND` operator.

Here is the general formula:

```
row_is_visible = (one or more of the permissive policies' conditions are non-zero) AND
                  (all of the restrictive policies's conditions are non-zero)
```

For example, the following policies

```
CREATE ROW POLICY pol1 ON mydb.table1 USING b=1 TO mira, peter
CREATE ROW POLICY pol2 ON mydb.table1 USING c=2 AS RESTRICTIVE TO peter, antonio
```

enables the user `peter` to see rows only if both `b=1 AND c=2`.

ON CLUSTER Clause

Allows creating row policies on a cluster, see [Distributed DDL](#).

Examples

```
CREATE ROW POLICY filter1 ON mydb.mytable USING a<1000 TO accountant, john@localhost
```

```
CREATE ROW POLICY filter2 ON mydb.mytable USING a<1000 AND b=5 TO ALL EXCEPT mira
```

```
CREATE ROW POLICY filter3 ON mydb.mytable USING 1 TO admin
```

CREATE QUOTA

Creates a `quota` that can be assigned to a user or a role.

Syntax:

```
CREATE QUOTA [IF NOT EXISTS | OR REPLACE] name [ON CLUSTER cluster_name]
    [KEYED BY {user_name | ip_address | client_key | client_key,user_name | client_key,ip_address} | NOT KEYED]
    [FOR [RANDOMIZED] INTERVAL number {second | minute | hour | day | week | month | quarter | year}
        {MAX { {queries | query_selects | query_inserts | errors | result_rows | result_bytes | read_rows | read_bytes |
            execution_time} = number } [...] | NO LIMITS | TRACKING ONLY} [...]]
    [TO {role [...] | ALL | ALL EXCEPT role [...]}]
```

Keys `user_name`, `ip_address`, `client_key`, `client_key`, `user_name` and `client_key`, `ip_address` correspond to the fields in the [system.quotas](#) table.

Parameters `queries`, `query_selects`, `query_inserts`, `errors`, `result_rows`, `result_bytes`, `read_rows`, `read_bytes`, `execution_time` correspond to the fields in the [system.quotas_usage](#) table.

`ON CLUSTER` clause allows creating quotas on a cluster, see [Distributed DDL](#).

Examples

Limit the maximum number of queries for the current user with 123 queries in 15 months constraint:

```
CREATE QUOTA qA FOR INTERVAL 15 month MAX queries = 123 TO CURRENT_USER;
```

For the default user limit the maximum execution time with half a second in 30 minutes, and limit the maximum number of queries with 321 and the maximum number of errors with 10 in 5 quarters:

```
CREATE QUOTA qB FOR INTERVAL 30 minute MAX execution_time = 0.5, FOR INTERVAL 5 quarter MAX queries = 321, errors = 10 TO default;
```

CREATE SETTINGS PROFILE

Creates [settings profiles](#) that can be assigned to a user or a role.

Syntax:

```
CREATE SETTINGS PROFILE [IF NOT EXISTS | OR REPLACE] TO name1 [ON CLUSTER cluster_name1]
    [, name2 [ON CLUSTER cluster_name2] ...]
    [SETTINGS variable [= value] [MIN [=] min_value] [MAX [=] max_value] [READONLY|WRITABLE] | INHERIT
    'profile_name'] [...]
```

`ON CLUSTER` clause allows creating settings profiles on a cluster, see [Distributed DDL](#).

Example

Create the `max_memory_usage_profile` settings profile with value and constraints for the `max_memory_usage` setting and assign it to user `robin`:

```
CREATE SETTINGS PROFILE max_memory_usage_profile SETTINGS max_memory_usage = 100000001 MIN 90000000
MAX 110000000 TO robin
```

ALTER

`ALTER` 仅支持 *MergeTree , Merge 以及 Distributed 等引擎表。
该操作有多种形式。

列操作

改变表结构：

```
ALTER TABLE [db].name [ON CLUSTER cluster] ADD|DROP|CLEAR|COMMENT|MODIFY COLUMN ...
```

在语句中，配置一个或多个用逗号分隔的动作。每个动作是对某个列实施的操作行为。

支持下列动作：

- **ADD COLUMN** — 添加列
- **DROP COLUMN** — 删除列
- **CLEAR COLUMN** — 重置列的值
- **COMMENT COLUMN** — 给列增加注释说明
- **MODIFY COLUMN** — 改变列的值类型，默认表达式以及TTL

这些动作将在下文中进行详述。

增加列

```
ADD COLUMN [IF NOT EXISTS] name [type] [default_expr] [codec] [AFTER name_after]
```

使用指定的 `name`, `type`, `codec` 以及 `default_expr` (请参见 [Default expressions](#))，往表中增加新的列。

如果sql中包含 `IF NOT EXISTS`，执行语句时如果列已经存在，CH不会报错。如果指定 `AFTER name_after` (表中另一个列的名称)，则新的列会加在指定列的后面。否则，新的列将被添加到表的末尾。注意，不能将新的列添加到表的开始位置，`name_after` 可以是执行该动作时已经在表中存在的任意列。

添加列仅仅是改变原有表的结构不会对已有数据产生影响。执行完 `ALTER` 后磁盘中也不会出现新的数据。如果查询表时列的数据为空，那么CH会使用列的默认值来进行填充（如果有默认表达式，则使用这个；或者用0或空字符串）。当数据块完成合并(参见[MergeTree](#))后，磁盘中会出现该列的数据。

这种方式允许 `ALTER` 语句能马上执行。不需要增加原有数据的大小。

示例：

```
ALTER TABLE visits ADD COLUMN browser String AFTER user_id
```

删除列

```
DROP COLUMN [IF EXISTS] name
```

通过指定 `name` 删除列。如果语句包含 `IF EXISTS`，执行时遇到不存在的列也不会报错。

从文件系统中删除数据。由于是删除列的整个文件，该语句几乎是立即执行完成的。

示例：

```
ALTER TABLE visits DROP COLUMN browser
```

清空列

```
CLEAR COLUMN [IF EXISTS] name IN PARTITION partition_name
```

重置指定分区中列的值。分区名称 `partition_name` 请参见 [怎样设置分区表达式](#)

如果语句中包含 `IF EXISTS`，遇到不存在的列，sql执行不会报错。

示例：

```
ALTER TABLE visits CLEAR COLUMN browser IN PARTITION tuple()
```

增加注释

```
COMMENT COLUMN [IF EXISTS] name 'comment'
```

给列增加注释说明。如果语句中包含 `IF EXISTS`，遇到不存在的列，sql执行不会报错。

每个列都可以包含注释。如果列的注释已经存在，新的注释会替换旧的。

注释信息保存在 `DESCRIBE TABLE`查询的 `comment_expression` 字段中。

示例：

```
ALTER TABLE visits COMMENT COLUMN browser 'The table shows the browser used for accessing the site.'
```

修改列

```
MODIFY COLUMN [IF EXISTS] name [type] [default_expr] [TTL]
```

该语句可以改变 `name` 列的属性：

- Type
- Default expression
- TTL

有关修改列TTL的示例，请参见 [Column TTL](#)。

如果语句中包含 `IF EXISTS`，遇到不存在的列，sql执行不会报错。

当改变列的类型时，列的值也被转换了，如同对列使用 `toType`函数一样。如果只改变了默认表达式，该语句几乎不会做任何复杂操作，并且几乎是立即执行完成的。

示例：

```
ALTER TABLE visits MODIFY COLUMN browser Array(String)
```

改变列的类型是唯一的复杂型动作 - 它改变了数据文件的内容。对于大型表，执行起来要花费较长的时间。

该操作分为如下处理步骤：

- 为修改的数据准备新的临时文件
- 重命名原来的文件
- 将新的临时文件改名为原来的数据文件名
- 删除原来的文件

仅仅在第一步是耗费时间的。如果该阶段执行失败，那么数据没有变化。如果执行后续的步骤中失败了，数据可以手动恢复。例外的情形是，当原来的文件从文件系统中被删除了，但是新的数据没有写入到临时文件中并且丢失了。

列操作的 `ALTER`行为是可以被复制的。这些指令会保存在ZooKeeper中，这样每个副本节点都能执行它们。所有的 `ALTER`将按相同的顺序执行。

The query waits for the appropriate actions to be completed on the other replicas.

然而，改变可复制表的列是可以被中断的，并且所有动作都以异步方式执行。

ALTER 操作限制

ALTER 操作允许在嵌套的数据结构中创建和删除单独的元素（列），但是不是整个嵌套结构。添加一个嵌套数据结构的列时，你可以用类似这样的名称 `name.nested_name` 及类型 `Array(T)` 来操作。嵌套数据结构等同于列名前带有同样前缀的多个数组列。

不支持对 `primary key` 或者 `sampling key` 中的列（在 `ENGINE` 表达式中用到的列）进行删除操作。改变包含在 `primary key` 中的列的类型时，如果操作不会导致数据的变化（例如，往 `Enum` 中添加一个值，或者将 `DateTime` 类型改成 `UInt32`），那么这种操作是可行的。

如果 ALTER 操作不足以完成你想要的表变动操作，你可以创建一张新的表，通过 `INSERT SELECT` 将数据拷贝进去，然后通过 `RENAME` 将新的表改成和原有表一样的名称，并删除原有的表。你可以使用 `clickhouse-copier` 代替 `INSERT SELECT`。

ALTER 操作会阻塞对表的所有读写操作。换句话说，当一个大的 `SELECT` 语句和 `ALTER` 同时执行时，`ALTER` 会等待，直到 `SELECT` 执行结束。与此同时，当 `ALTER` 运行时，新的 `sql` 语句将会等待。

对于不存储数据的表（例如 `Merge` 及 `Distributed` 表），`ALTER` 仅仅改变了自身的表结构，不会改变从属的表结构。例如，对 `Distributed` 表执行 `ALTER` 操作时，需要对其它包含该表的服务器执行该操作。

key 表达式的修改

支持下列表达式：

```
MODIFY ORDER BY new_expression
```

该操作仅支持 `MergeTree` 系列表（含 `replicated` 表）。它会将表的 `排序键` 变成 `new_expression`（元组表达式）。主键仍保持不变。

该操作是轻量级的，仅会改变元数据。

跳过索引来更改数据

该操作仅支持 `MergeTree` 系列表（含 `replicated` 表）。

下列操作是允许的：

- `ALTER TABLE [db].name ADD INDEX name expression TYPE type GRANULARITY value [FIRST|AFTER name]` - 在表的元数据中增加索引说明
- `ALTER TABLE [db].name DROP INDEX name` - 从表的元数据中删除索引描述，并从磁盘上删除索引文件

由于只改变表的元数据或者删除文件，因此该操作是轻量级的，也可以被复制到其它节点（通过 `Zookeeper` 同步索引元数据）

更改约束

参见 `constraints` 查看更多信息。

通过下面的语法，可以添加或删除约束：

```
ALTER TABLE [db].name ADD CONSTRAINT constraint_name CHECK expression;
ALTER TABLE [db].name DROP CONSTRAINT constraint_name;
```

上述语句会从表中增加或删除约束的元数据，因此会被立即处理。

对已有数据的约束检查 将不会执行。

对可复制表的操作可通过 `Zookeeper` 传播到其它副本节点。

更改分区及文件块

允许进行下列关于 **partitions** 的操作：

- **DETACH PARTITION** — 将分区数据移动到 `detached`，并且忘记它
- **DROP PARTITION** — 删除一个partition.
- **ATTACH PART|PARTITION** — 将`detached` 目录中的分区重新添加到表中.
- **ATTACH PARTITION FROM** — 从表中复制数据分区到另一张表，并添加分区
- **REPLACE PARTITION** — 从表中复制数据分区到其它表及副本
- **MOVE PARTITION TO TABLE** — 从表中复制数据分区到其它表.
- **CLEAR COLUMN IN PARTITION** — 重置分区中某个列的值
- **CLEAR INDEX IN PARTITION** — 重置分区中指定的二级索引
- **FREEZE PARTITION** — 创建分区的备份
- **FETCH PARTITION** — 从其它服务器上下载分
- **MOVE PARTITION|PART** — 将分区/数据块移动到另外的磁盘/卷

分区剥离

```
ALTER TABLE table_name DETACH PARTITION partition_expr
```

将指定分区的数据移动到 `detached` 目录。服务器会忽略被分离的数据分区。只有当你使用 **ATTACH** 时，服务器才会知晓这部分数据。

示例：

```
ALTER TABLE visits DETACH PARTITION 201901
```

从 [如何设置分区表达式](#) 章节中获取分区表达式的设置说明。

当执行操作以后，可以对 `detached` 目录的数据进行任意操作，例如删除文件，或者放着不管。

该操作是可以复制的，它会将所有副本节点上的数据移动到 `detached` 目录。注意仅能在副本的leader节点上执行该操作。想了解副本是否是leader节点，需要在 `system.replicas` 表执行 `SELECT` 操作。或者，可以很方便的在所有副本节点上执行 `DETACH`操作，但除leader外其它的副本节点会抛出异常。

删除分区

```
ALTER TABLE table_name DROP PARTITION partition_expr
```

从表中删除指定分区。该操作会将分区标记为不活跃的，然后在大约10分钟内删除全部数据。

在 [如何设置分区表达式](#) 中获取分区表达式的设置说明。

该操作是可复制的，副本节点的数据也将被删除。

删除已剥离的分区|数据块

```
ALTER TABLE table_name DROP DETACHED PARTITION|PART partition_expr
```

从`detached`目录中删除指定分区的特定部分或所有数据。访问 [如何设置分区表达式](#) 可获取设置分区表达式的详细信息。

关联分区|数据块

```
ALTER TABLE table_name ATTACH PARTITION|PART partition_expr
```

从detached目录中添加数据到数据表。可以添加整个分区的数据，或者单独的数据块。例如：

```
ALTER TABLE visits ATTACH PARTITION 201901;
ALTER TABLE visits ATTACH PART 201901_2_2_0;
```

访问[如何设置分区表达式](#)可获取设置分区表达式的详细信息。

该操作是可以复制的。副本启动器检查detached目录是否有数据。如果有，该操作会检查数据的完整性。如果一切正常，该操作将数据添加到表中。其它副本节点通过副本启动器下载这些数据。

因此可以在某个副本上将数据放到detached目录，然后通过ALTER ... ATTACH操作将这部分数据添加到该表的所有副本。

从...关联分区

```
ALTER TABLE table2 ATTACH PARTITION partition_expr FROM table1
```

该操作将table1表的数据分区复制到table2表的已有分区。注意table1表的数据不会被删除。

为保证该操作能成功运行，下列条件必须满足：

- 2张表必须有相同的结构
- 2张表必须有相同的分区键

替换分区

```
ALTER TABLE table2 REPLACE PARTITION partition_expr FROM table1
```

该操作将table1表的数据分区复制到table2表，并替换table2表的已有分区。注意table1表的数据不会被删除。

为保证该操作能成功运行，下列条件必须满足：

- 2张表必须有相同的结构
- 2张表必须有相同的分区键

将分区移动到表

```
ALTER TABLE table_source MOVE PARTITION partition_expr TO TABLE table_dest
```

该操作将table_source表的数据分区移动到table_dest表，并删除table_source表的数据。

为保证该操作能成功运行，下列条件必须满足：

- 2张表必须有相同的结构
- 2张表必须有相同的分区键
- 2张表必须属于相同的引擎系列（可复制表或不可复制表）
- 2张表必须有相同的存储方式

清空分区的列

```
ALTER TABLE table_name CLEAR COLUMN column_name IN PARTITION partition_expr
```

重置指定分区的特定列的值。如果建表时使用了 `DEFAULT` 语句，该操作会将列的值重置为该默认值。

示例：

```
ALTER TABLE visits CLEAR COLUMN hour in PARTITION 201902
```

冻结分区

```
ALTER TABLE table_name FREEZE [PARTITION partition_expr]
```

该操作为指定分区创建一个本地备份。如果 `PARTITION` 语句省略，该操作会一次性为所有分区创建备份。

Note

整个备份过程不需要停止服务

注意对于老式的表，可以指定分区名前缀（例如，'2019'），然后该操作会创建所有对应分区的备份。访问 [如何设置分区表达式](#) 可获取设置分区表达式的详细信息。

在执行操作的同时，对于数据快照，该操作会创建到表数据的硬链接。硬链接放置在 `/var/lib/clickhouse/shadow/N/...`，也就是：

- `/var/lib/clickhouse/` 服务器配置文件中指定的CH工作目录
- `N` 备份的增长序号

Note

如果你使用 [多个磁盘存储数据表](#)，

那么每个磁盘上都有 `shadow/N` 目录，用来保存 `PARTITION` 表达式对应的数据块。

备份内部也会创建和 `/var/lib/clickhouse/` 内部一样的目录结构。该操作在所有文件上执行'chmod'，禁止往里写入数据

当备份创建完毕，你可以从 `/var/lib/clickhouse/shadow/` 复制数据到远端服务器，然后删除本地数据。注意 `ALTER t FREEZE PARTITION` 操作是不能复制的，它仅在本地服务器上创建本地备份。

该操作创建备份几乎是即时的（但是首先它会等待相关表的当前操作执行完成）

`ALTER TABLE t FREEZE PARTITION` 仅仅复制数据，而不是元数据信息。要复制表的元数据信息，拷贝这个文件 `/var/lib/clickhouse/metadata/database/table.sql`

从备份中恢复数据，按如下步骤操作：

1. 如果表不存在，先创建。查看.sql 文件获取执行语句（将ATTACH 替换成 CREATE）。
2. 从 备份的 `data/database/table/` 目录中将数据复制到 `/var/lib/clickhouse/data/database/table/detached/` 目录
3. 运行 `ALTER TABLE t ATTACH PARTITION` 操作，将数据添加到表中

恢复数据不需要停止服务进程。

想了解备份及数据恢复的更多信息，请参见 [数据备份](#)。

删除分区的索引

```
ALTER TABLE table_name CLEAR INDEX index_name IN PARTITION partition_expr
```

该操作和 `CLEAR COLUMN` 类似，但是它重置的是索引而不是列的数据。

获取分区

```
ALTER TABLE table_name FETCH PARTITION partition_expr FROM 'path-in-zookeeper'
```

从另一服务器上下载分区数据。仅支持可复制引擎表。

该操作做了如下步骤：

1. 从指定数据分片上下载分区。在 `path-in-zookeeper` 这一参数你必须设置 Zookeeper 中该分片的 `path` 值。
2. 然后将已下载的数据放到 `table_name` 表的 `detached` 目录下。通过 `ATTACH PARTITION|PART` 将数据加载到表中。

示例：

```
ALTER TABLE users FETCH PARTITION 201902 FROM '/clickhouse/tables/01-01/visits';
ALTER TABLE users ATTACH PARTITION 201902;
```

注意：

- `ALTER ... FETCH PARTITION` 操作不支持复制，它仅在本地服务器上将分区移动到 `detached` 目录。
- `ALTER TABLE ... ATTACH` 操作是可复制的。它将数据添加到所有副本。数据从某个副本的 `detached` 目录中添加进来，然后添加到邻近的副本。

在开始下载之前，系统检查分区是否存在以及和表结构是否匹配。然后从健康的副本集中自动选择最合适副本。

虽然操作叫做 `ALTER TABLE`，但是它并不能改变表结构，也不会立即改变表中可用的数据。

移动分区 | 数据块

将 MergeTree 引擎表的分区或数据块移动到另外的卷/磁盘中。参见 [使用多个块设备存储数据](#)

```
ALTER TABLE table_name MOVE PARTITION|PART partition_expr TO DISK|VOLUME 'disk_name'
```

`ALTER TABLE t MOVE` 操作：

- 不支持复制，因为不同副本可以有不同的存储方式
- 如果指定的磁盘或卷没有配置，返回错误。如果存储方式中设定的数据移动条件不能满足，该操作同样报错。
- 这种情况也会报错：即将移动的数据已经由后台进程在进行移动操作时，并行的 `ALTER TABLE t MOVE` 操作或者作为后台数据合并的结果。这种情形下用户不能任何额外的动作。

示例：

```
ALTER TABLE hits MOVE PART '20190301_14343_16206_438' TO VOLUME 'slow'
ALTER TABLE hits MOVE PARTITION '2019-09-01' TO DISK 'fast_ssds'
```

如何设置分区表达式

通过不同方式在 `ALTER ... PARTITION` 操作中设置分区表达式：

- `system.parts` 表 `partition` 列的某个值，例如，`ALTER TABLE visits DETACH PARTITION 201901`
- 表的列表达式。支持常量及常量表达式。例如，`ALTER TABLE visits DETACH PARTITION toYYYYMM(toDate('2019-01-25'))`
- 使用分区 ID。分区 ID 是字符串变量（可能的话有较好的可读性），在文件系统和 ZooKeeper 中作为分区名称。分区 ID 必须配置在 `PARTITION ID` 中，用单引号包含，例如，`ALTER TABLE visits DETACH PARTITION ID '201901'`
- 在 `ALTER ATTACH PART` 和 `DROP DETACHED PART` 操作中，要配置块的名称，使用 `system.detached_parts` 表中 `name` 列的字符串值，例如：`ALTER TABLE visits ATTACH PART '201901_1_1_0'`

设置分区时，引号使用要看分区表达式的类型。例如，对于 `String` 类型，需要设置用引号(')包含的名称。对于 `Date` 和 `Int*` 引号就不需要了。

对于老式的表，可以用数值 `201901` 或字符串 `'201901'` 来设置分区。新式的表语法严格和类型一致（类似于 `VALUES` 输入的解析）

上述所有规则同样适用于 `OPTIMIZE` 操作。在对未分区的表进行 `OPTIMIZE` 操作时，如果需要指定唯一的分区，这样设置表达式 `PARTITION tuple()`。例如：

```
OPTIMIZE TABLE table_not_partitioned PARTITION tuple() FINAL;
```

`ALTER ... PARTITION` 操作的示例在 `00502_custom_partitioning_local` 和 `00502_custom_partitioning_replicated_zookeeper` 提供了演示。

更改表的TTL

通过以下形式的请求可以修改 `table TTL`

```
ALTER TABLE table-name MODIFY TTL ttl-expression
```

ALTER操作的同步性

对于不可复制的表，所有 `ALTER` 操作都是同步执行的。对于可复制的表，`ALTER` 操作会将指令添加到 ZooKeeper 中，然后会尽快的执行它们。然而，该操作可以等待其它所有副本将指令执行完毕。

对于 `ALTER ... ATTACH|DETACH|DROP` 操作，可以通过设置 `replication_alter_partitions_sync` 来启用等待。可用参数值：
- 不需要等待; 1 - 仅等待自己执行(默认); 2 - 等待所有节点

Mutations

`Mutations` 是一类允许对表的行记录进行删除或更新的 `ALTER` 操作。相较于标准的 `UPDATE` 和 `DELETE` 用于少量行操作而言，`Mutations` 用来对表的很多行进行重量级的操作。该操作支持 `MergeTree` 系列表，包含支持复制功能的表。

已有的表已经支持 `mutations` 操作（不需要转换）。但是在首次对表进行 `mutation` 操作以后，它的元数据格式变得和之前的版本不兼容，并且不能回退到之前版本。

目前可用的命令：

```
ALTER TABLE [db.]table DELETE WHERE filter_expr
```

`filter_expr` 必须是 `UInt8` 型。该操作将删除表中 `filter_expr` 表达式值为非 0 的列

```
ALTER TABLE [db.]table UPDATE column1 = expr1 [, ...] WHERE filter_expr
```

`filter_expr` 必须是 `UInt8` 型。该操作将更新表中各行 `filter_expr` 表达式值为非 0 的指定列的值。通过 `CAST` 操作将值转换成对应列的类型。不支持对用于主键或分区键表达式的列进行更新操作。

```
ALTER TABLE [db.]table MATERIALIZE INDEX name IN PARTITION partition_name
```

该操作更新 `partition_name` 分区中的二级索引 `name`。

单次操作可以包含多个逗号分隔的命令。

对于 *MergeTree 引擎表，`mutation` 操作通过重写整个数据块来实现。没有原子性保证 - 被 `mutation` 操作的数据会被替换，在 `mutation` 期间开始执行的 `SELECT` 查询能看到所有已经完成 `mutation` 的数据，以及还没有被 `mutation` 替换的数据。

mutation总是按照它们的创建顺序来排序并以同样顺序在每个数据块中执行。mutation操作也会部分的和Insert操作一起排序 - 在mutation提交之前插入的数据会参与mutation操作，在mutation提交之后的插入的数据则不会参与mutation。注意mutation从来不会阻塞插入操作。

mutation操作在提交后（对于可复制表，添加到Zookeeper,对于不可复制表，添加到文件系统）立即返回。mutation操作本身是根据系统的配置参数异步执行的。要跟踪mutation的进度，可以使用系统表 `system.mutations`。已经成功提交的mutation操作在服务重启后仍会继续执行。一旦mutation完成提交，就不能回退了，但是如果因为某种原因操作被卡住了，可以通过 `KILL MUTATION` 操作来取消它的执行。

已完成的mutations记录不会立即删除（要保留的记录数量由 `finished_mutations_to_keep` 这一参数决定）。之前的mutation记录会被删除。

修改用户

修改CH的用户账号

语法

```
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'] [,....]
```

说明

要使用 `ALTER USER`，你必须拥有 `ALTER USER` 操作的权限

Examples

设置默认角色：

```
ALTER USER user DEFAULT ROLE role1, role2
```

如果角色之前没分配给用户，CH会抛出异常。

将所有分配的角色设为默认

```
ALTER USER user DEFAULT ROLE ALL
```

如果以后给用户分配了某个角色，它将自动成为默认角色

将除了 `role1` 和 `role2`之外的其它角色 设为默认

```
ALTER USER user DEFAULT ROLE ALL EXCEPT role1, role2
```

修改角色

修改角色.

语法

```
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'] [...]
```

修改row policy

修改row policy.

语法

```
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 [...]}]
```

修改配额quotas

修改配额quotas.

语法

```
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 [...]}]
```

修改settings配置

修改settings配置.

语法

```
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'] [...]
```

SYSTEM Queries

- RELOAD EMBEDDED DICTIONARIES
- RELOAD DICTIONARIES
- RELOAD DICTIONARY
- DROP DNS CACHE
- DROP MARK CACHE
- DROP UNCOMPRESSED CACHE
- DROP COMPILED EXPRESSION CACHE

- `DROP REPLICA`
- `FLUSH LOGS`
- `RELOAD CONFIG`
- `SHUTDOWN`
- `KILL`
- `STOP DISTRIBUTED SENDS`
- `FLUSH DISTRIBUTED`
- `START DISTRIBUTED SENDS`
- `STOP MERGES`
- `START MERGES`
- `STOP TTL MERGES`
- `START TTL MERGES`
- `STOP MOVES`
- `START MOVES`
- `STOP FETCHES`
- `START FETCHES`
- `STOP REPLICATED SENDS`
- `START REPLICATED SENDS`
- `STOP REPLICATION QUEUES`
- `START REPLICATION QUEUES`
- `SYNC REPLICA`
- `RESTART REPLICA`
- `RESTART REPLICAS`

RELOAD EMBEDDED DICTIONARIES]

重新加载所有[内置字典](#)。默认情况下内置字典是禁用的。

总是返回‘OK.’，不管这些内置字典的更新结果如何。

RELOAD DICTIONARIES

重载已经被成功加载过的所有字典。

默认情况下，字典是延时加载的（[`dictionaries_lazy_load`](#)），不是在服务启动时自动加载，而是在第一次使用`dictGet`函数或通过`SELECT from tables with ENGINE = Dictionary`进行访问时被初始化。这个命令`SYSTEM RELOAD DICTIONARIES`就是针对这类表进行重新加载的。

RELOAD DICTIONARY `Dictionary_name`

完全重新加载指定字典 `dictionary_name`，不管该字典的状态如何(LOADED / NOT_LOADED / FAILED)。不管字典的更新结果如何，总是返回 `OK`。

字典的状态可以通过查询 `system.dictionaries` 表来检查。

```
SELECT name, status FROM system.dictionaries;
```

DROP DNS CACHE

重置CH的dns缓存。有时候（对于旧的ClickHouse版本）当某些底层环境发生变化时（修改其它Clickhouse服务器的ip或字典所在服务器的ip），需要使用该命令。

更多自动化的缓存管理相关信息，参见`disable_internal_dns_cache`, `dns_cache_update_period`这些参数。

DROP MARK CACHE

重置mark缓存。在进行ClickHouse开发或性能测试时使用。

DROP REPLICA

使用下面的语句可以删除已经无效的副本。

```
SYSTEM DROP REPLICA 'replica_name' FROM TABLE database.table;
SYSTEM DROP REPLICA 'replica_name' FROM DATABASE database;
SYSTEM DROP REPLICA 'replica_name';
SYSTEM DROP REPLICA 'replica_name' FROM ZKPATH '/path/to/table/in/zk';
```

该操作将副本的路径从Zookeeper中删除。当副本失效，并且由于该副本已经不存在导致它的元数据不能通过 `DROP TABLE` 从zookeeper中删除，这种情形下可以使用该命令。它只会删除失效或过期的副本，不会删除本地的副本。请使用 `DROP TABLE` 来删除本地副本。`DROP REPLICA` 不会删除任何表，并且不会删除磁盘上的任何数据或元数据信息。

第1条语句：删除 `database.table` 表的 `replica_name` 副本的元数据

第2条语句：删除 `database` 数据库的所有 `replica_name` 副本的元数据

第3条语句：删除本地服务器所有 `replica_name` 副本的元数据

第4条语句：用于在表的其它所有副本都删除时，删除已失效副本的元数据。使用时需要明确指定表的路径。该路径必须和创建表时 `ReplicatedMergeTree` 引擎的第一个参数一致。

DROP UNCOMPRESSED CACHE

重置未压缩数据的缓存。用于ClickHouse开发和性能测试。

管理未压缩数据缓存的参数，使用以下的服务器级别设置 `uncompressed_cache_size` 以及 `query/user/profile` 级别设置 `use_uncompressed_cache`

DROP COMPILED EXPRESSION CACHE

重置已编译的表达式缓存。用于ClickHouse开发和性能测试。

当 `query/user/profile` 启用配置项 `compile-expressions` 时，编译的表达式缓存开启。

FLUSH LOGS

将日志信息缓冲数据刷入系统表（例如`system.query_log`）。调试时允许等待不超过7.5秒。当信息队列为空时，会创建系统表。

RELOAD CONFIG

重新加载ClickHouse的配置。用于当配置信息存放在ZooKeeper时。

SHUTDOWN

关闭ClickHouse服务（类似于 `service clickhouse-server stop / kill {$pid_clickhouse-server}`）

KILL

关闭ClickHouse进程（`kill -9 {$ pid_clickhouse-server}`）

Managing Distributed Tables

ClickHouse可以管理 **distribute** 表。当用户向这类表插入数据时，ClickHouse首先为需要发送到集群节点的数据创建一个队列，然后异步的发送它们。你可以维护队列的处理过程，通过 **STOP DISTRIBUTED SENDS**, **FLUSH DISTRIBUTED**, 以及 **START DISTRIBUTED SENDS**。你也可以设置 `insert_distributed_sync` 参数来以同步的方式插入分布式数据。

STOP DISTRIBUTED SENDS

当向分布式表插入数据时，禁用后台的分布式数据分发。

```
SYSTEM STOP DISTRIBUTED SENDS [db.]<distributed_table_name>
```

FLUSH DISTRIBUTED

强制让ClickHouse同步向集群节点同步发送数据。如果有节点失效，ClickHouse抛出异常并停止插入操作。当所有节点都恢复上线时，你可以重试之前的操作直到成功执行。

```
SYSTEM FLUSH DISTRIBUTED [db.]<distributed_table_name>
```

START DISTRIBUTED SENDS

当向分布式表插入数据时，允许后台的分布式数据分发。

```
SYSTEM START DISTRIBUTED SENDS [db.]<distributed_table_name>
```

Managing MergeTree Tables

ClickHouse可以管理 **MergeTree** 表的后台处理进程。

STOP MERGES

为MergeTree系列引擎表停止后台合并操作。

```
SYSTEM STOP MERGES [[db.]merge_tree_family_table_name]
```

Note

`DETACH / ATTACH` 表操作会在后台进行表的 `merge` 操作，甚至当所有 **MergeTree** 表的合并操作已经停止的情况下。

START MERGES

为MergeTree系列引擎表启动后台合并操作。

```
SYSTEM START MERGES [[db.]merge_tree_family_table_name]
```

STOP TTL MERGES

根据 [TTL expression](#)，为 MergeTree 系列引擎表停止后台删除旧数据。

不管表存在与否，都返回 `OK.`。当数据库不存在时返回错误。

```
SYSTEM STOP TTL MERGES [[db.]merge_tree_family_table_name]
```

START TTL MERGES

根据 [TTL expression](#)，为 MergeTree 系列引擎表启动后台删除旧数据。不管表存在与否，都返回 `OK.`。当数据库不存在时返回错误。

```
SYSTEM START TTL MERGES [[db.]merge_tree_family_table_name]
```

STOP MOVES

根据 [TTL expression](#)，为 MergeTree 系列引擎表停止后台移动数据。不管表存在与否，都返回 `OK.`。当数据库不存在时返回错误。

```
SYSTEM STOP MOVES [[db.]merge_tree_family_table_name]
```

START MOVES

根据 [TTL expression](#)，为 MergeTree 系列引擎表启动后台移动数据。不管表存在与否，都返回 `OK.`。当数据库不存在时返回错误。

```
SYSTEM STOP MOVES [[db.]merge_tree_family_table_name]
```

Managing ReplicatedMergeTree Tables

管理 [ReplicatedMergeTree](#) 表的后台复制相关进程。

STOP FETCHES

停止后台获取 ReplicatedMergeTree 系列引擎表中插入的数据块。

不管表引擎类型如何或表/数据库是否存，都返回 `OK.`。

```
SYSTEM STOP FETCHES [[db.]replicated_merge_tree_family_table_name]
```

START FETCHES

启动后台获取 ReplicatedMergeTree 系列引擎表中插入的数据块。

不管表引擎类型如何或表/数据库是否存，都返回 `OK.`。

```
SYSTEM START FETCHES [[db.]replicated_merge_tree_family_table_name]
```

STOP REPLICATED SENDS

停止通过后台分发 ReplicatedMergeTree 系列引擎表中新插入的数据块到集群的其它副本节点。

```
SYSTEM STOP REPLICATED SENDS [[db.]replicated_merge_tree_family_table_name]
```

START REPLICATED SENDS

启动通过后台分发 ReplicatedMergeTree 系列引擎表中新插入的数据块到集群的其它副本节点。

```
SYSTEM START REPLICATED SENDS [[db.]replicated_merge_tree_family_table_name]
```

STOP REPLICATION QUEUES

停止从Zookeeper中获取 ReplicatedMergeTree系列表的复制队列的后台任务。可能的后台任务类型包含：merges, fetches, mutation，带有 ON CLUSTER的ddl语句

```
SYSTEM STOP REPLICATION QUEUES [[db.]replicated_merge_tree_family_table_name]
```

START REPLICATION QUEUES

启动从Zookeeper中获取 ReplicatedMergeTree系列表的复制队列的后台任务。可能的后台任务类型包含：merges, fetches, mutation，带有 ON CLUSTER的ddl语句

```
SYSTEM START REPLICATION QUEUES [[db.]replicated_merge_tree_family_table_name]
```

SYNC REPLICA

直到 ReplicatedMergeTree表将要和集群的其它副本进行同步之前会一直运行。如果当前对表的获取操作禁用的话，在达到 receive_timeout之前会一直运行。

```
SYSTEM SYNC REPLICA [db.]replicated_merge_tree_family_table_name
```

RESTART REPLICA

重置 ReplicatedMergeTree表的Zookeeper会话状态。该操作会以Zookeeper为参照，对比当前状态，有需要的情况下将任务添加到ZooKeeper队列。

基于ZooKeeper的日期初始化复制队列，类似于 ATTACH TABLE语句。短时间内不能对表进行任何操作。

```
SYSTEM RESTART REPLICA [db.]replicated_merge_tree_family_table_name
```

RESTART REPLICAS

重置所有 ReplicatedMergeTree表的ZooKeeper会话状态。该操作会以Zookeeper为参照，对比当前状态，有需要的情况下将任务添加到ZooKeeper队列。

SHOW 查询

SHOW CREATE TABLE

```
SHOW CREATE [TEMPORARY] [TABLE|DICTIONARY] [db.]table [INTO OUTFILE filename] [FORMAT format]
```

返回单个字符串类型的 ‘statement’列，其中只包含了一个值 - 用来创建指定对象的 CREATE 语句。

SHOW DATABASES

```
SHOW DATABASES [INTO OUTFILE filename] [FORMAT format]
```

打印所有的数据库列表，该查询等同于 SELECT name FROM system.databases [INTO OUTFILE filename] [FORMAT format]

SHOW PROCESSLIST

```
SHOW PROCESSLIST [INTO OUTFILE filename] [FORMAT format]
```

输出 `system.processes` 表的内容，包含有当前正在处理的请求列表，除了 `SHOW PROCESSLIST` 查询。

`SELECT * FROM system.processes` 查询返回和当前请求相关的所有数据

提示 (在控制台执行):

```
$ watch -n1 "clickhouse-client --query='SHOW PROCESSLIST'"
```

SHOW TABLES

显示表的清单

```
SHOW [TEMPORARY] TABLES [{FROM | IN} <db>] [LIKE '<pattern>' | WHERE expr] [LIMIT <N>] [INTO OUTFILE <filename>] [FORMAT <format>]
```

如果未使用 `FROM` 字句，该查询返回当前数据库的所有表清单

可以用下面的方式获得和 `SHOW TABLES`一样的结果：

```
SELECT name FROM system.tables WHERE database = <db> [AND name LIKE <pattern>] [LIMIT <N>] [INTO OUTFILE <filename>] [FORMAT <format>]
```

示例

下列查询获取最前面的2个位于 `system` 库中且表名包含 `co` 的表。

```
SHOW TABLES FROM system LIKE '%co%' LIMIT 2
```

name
aggregate_function_combinators
collations

SHOW DICTIONARIES

以列表形式显示 `外部字典`。

```
SHOW DICTIONARIES [FROM <db>] [LIKE '<pattern>'] [LIMIT <N>] [INTO OUTFILE <filename>] [FORMAT <format>]
```

如果 `FROM` 字句没有指定，返回当前数据库的字典列表

可以通过下面的查询获取和 `SHOW DICTIONARIES` 相同的结果：

```
SELECT name FROM system.dictionaries WHERE database = <db> [AND name LIKE <pattern>] [LIMIT <N>] [INTO OUTFILE <filename>] [FORMAT <format>]
```

示例

下列查询获取最前面的2个位于 `system` 库中且名称包含 `reg` 的字典表。

```
SHOW DICTIONARIES FROM db LIKE '%reg%' LIMIT 2
```

```
name  
regions |  
region_names |
```

SHOW GRANTS

显示用户的权限

语法

```
SHOW GRANTS [FOR user]
```

如果未指定用户，输出当前用户的权限

SHOW CREATE USER

显示 **user creation** 用到的参数。

SHOW CREATE USER 不会输出用户的密码信息

语法

```
SHOW CREATE USER [name | CURRENT_USER]
```

SHOW CREATE ROLE

显示 **role creation** 中用到的参数。

语法

```
SHOW CREATE ROLE name
```

SHOW CREATE ROW POLICY

显示 **row policy creation** 中用到的参数

语法

```
SHOW CREATE [ROW] POLICY name ON [database.]table
```

SHOW CREATE QUOTA

显示 **quota creation** 中用到的参数

语法

```
SHOW CREATE QUOTA [name | CURRENT]
```

SHOW CREATE SETTINGS PROFILE

显示 **settings profile creation** 中用到的参数

语法

```
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

授权

- 给ClickHouse的用户或角色赋予 **权限**
- 将角色分配给用户或其他角色

取消权限，使用 **REVOKE**语句。查看已授权的权限请使用 **SHOW GRANTS**。

授权操作语法

```
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** — 权限类型
- **role** — 用户角色
- **user** — 用户账号

WITH GRANT OPTION 授予 **user** 或 **role**执行 **GRANT** 操作的权限。用户可将在自身权限范围内的权限进行授权
WITH REPLACE OPTION 以当前sql里的新权限替代掉 **user** 或 **role**的旧权限，如果没有该选项则是追加授权。

角色分配的语法

```
GRANT [ON CLUSTER cluster_name] role [...] TO {user | another_role | CURRENT_USER} [...] [WITH ADMIN OPTION] [WITH REPLACE OPTION]
```

- **role** — 角色
- **user** — 用户

WITH ADMIN OPTION 授予 **user** 或 **role** 执行**ADMIN OPTION** 的权限

WITH REPLACE OPTION 以当前sql里的新**role**替代掉 **user** 或 **role**的旧**role**，如果没有该选项则是追加**roles**。

用法

使用 `GRANT`，你的账号必须有 `GRANT OPTION`的权限。用户只能将在自身权限范围内的权限进行授权

例如，管理员有权通过下面的语句给 `john`账号添加授权

```
GRANT SELECT(x,y) ON db.table TO john WITH GRANT OPTION
```

这意味着 `john` 有权限执行以下操作：

- `SELECT x,y FROM db.table.`
- `SELECT x FROM db.table.`
- `SELECT y FROM db.table.`

`john` 不能执行 `SELECT z FROM db.table`。同样的 `SELECT * FROM db.table` 也是不允许的。执行这个查询时，`CH`不会返回任何数据，甚至 `x` 和 `y`列。唯一的例外是，当表仅包含 `x`和`y`列时。这种情况下，`CH`返回所有数据。

同样 `john` 有权执行 `GRANT OPTION`，因此他能给其它账号进行和自己账号权限范围相同的授权。

可以使用`*`号代替表或库名进行授权操作。例如，`GRANT SELECT ON db.* TO john` 操作运行 `john`对 `db`库的所有表执行 `SELECT`查询。同样，你可以忽略库名。在这种情形下，权限将指向当前的数据库。例如，`GRANT SELECT ON* to john` 对当前数据库的所有表指定授权，`GRANT SELECT ON mytable to john`对当前数据库的 `mytable`表进行授权。

访问 `system`数据库总是被允许的（因为这个数据库用来处理sql操作）

可以一次给多个账号进行多种授权操作。`GRANT SELECT,INSERT ON *.* TO john,robin` 允许 `john`和`robin` 账号对任意数据库的任意表执行 `INSERT`和 `SELECT`操作。

权限

权限是指执行特定操作的许可

权限有层级结构。一组允许的操作依赖相应的权限范围。

权限的层级：

- `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

如何对待该层级的示例：

- ALTER 权限包含所有其它 ALTER * 的权限
- ALTER CONSTRAINT 包含 ALTER ADD CONSTRAINT 和 ALTER DROP CONSTRAINT 权限

权限被应用到不同级别。 Knowing of a level suggests syntax available for privilege.

级别（由低到高）：

- COLUMN - 可以授权到列，表，库或者全局
- TABLE - 可以授权到表，库，或全局
- VIEW - 可以授权到视图，库，或全局
- DICTIONARY - 可以授权到字典，库，或全局
- DATABASE - 可以授权到数据库或全局
- GLABLE - 可以授权到全局
- GROUP - 不同级别的权限分组。当授予 GROUP 级别的权限时，根据所用的语法，只有对应分组中的权限才会被分配。

允许的语法示例：

- GRANT SELECT(x) ON db.table TO user
- GRANT SELECT ON db.* TO user

不允许的语法示例：

- GRANT CREATE USER(x) ON db.table TO user
- GRANT CREATE USER ON db.* TO user

特殊的权限 ALL 将所有权限授予给用户或角色

默认情况下，一个用户账号或角色没有可授予的权限

如果用户或角色没有任何权限，它将显示为 NONE 权限

有些操作根据它们的实现需要一系列的权限。例如，**RENAME** 操作需要以下权限来执行：SELECT, CREATE TABLE, INSERT 和 DROP TABLE。

SELECT

允许执行 **SELECT** 查询

权限级别: COLUMN.

说明

有该权限的用户可以对指定的表和库的指定列进行 **SELECT** 查询。如果用户查询包含了其它列则结果不返回数据。

考虑如下的授权语句：

```
GRANT SELECT(x,y) ON db.table TO john
```

该权限允许 **john** 对 **db.table** 表的列 **x,y** 执行任意 **SELECT** 查询，例如 **SELECT x FROM db.table**。**john** 不能执行 **SELECT z FROM db.table** 以及 **SELECT * FROM db.table**。执行这个查询时，CH 不会返回任何数据，甚至 **x** 和 **y** 列。唯一的例外是，当表仅包含 **x** 和 **y** 列时。这种情况下，CH 返回所有数据。

INSERT

允许执行 **INSERT** 操作。

权限级别: COLUMN.

说明

有该权限的用户可以对指定的表和库的指定列进行 **INSERT** 操作。如果用户查询包含了其它列则结果不返回数据。

示例

```
GRANT INSERT(x,y) ON db.table TO john
```

该权限允许 **john** 对 **db.table** 表的列 **x,y** 执行数据插入操作

ALTER

允许根据下列权限层级执行 **ALTER** 操作

- **ALTER**. 级别: COLUMN.

- ALTER TABLE. 级别: GROUP
 - ALTER UPDATE. 级别: COLUMN. 别名: UPDATE
 - ALTER DELETE. 级别: COLUMN. 别名: DELETE
 - ALTER COLUMN. 级别: GROUP
 - ALTER ADD COLUMN. 级别: COLUMN. 别名: ADD COLUMN
 - ALTER DROP COLUMN. 级别: COLUMN. 别名: DROP COLUMN
 - ALTER MODIFY COLUMN. 级别: COLUMN. 别名: MODIFY COLUMN
 - ALTER COMMENT COLUMN. 级别: COLUMN. 别名: COMMENT COLUMN
 - ALTER CLEAR COLUMN. 级别: COLUMN. 别名: CLEAR COLUMN
 - ALTER RENAME COLUMN. 级别: COLUMN. 别名: RENAME COLUMN
 - ALTER INDEX. 级别: GROUP. 别名: INDEX
 - ALTER ORDER BY. 级别: TABLE. 别名: ALTER MODIFY ORDER BY, MODIFY ORDER BY
 - ALTER ADD INDEX. 级别: TABLE. 别名: ADD INDEX
 - ALTER DROP INDEX. 级别: TABLE. 别名: DROP INDEX
 - ALTER MATERIALIZE INDEX. 级别: TABLE. 别名: MATERIALIZE INDEX
 - ALTER CLEAR INDEX. 级别: TABLE. 别名: CLEAR INDEX
 - ALTER CONSTRAINT. 级别: GROUP. 别名: CONSTRAINT
 - ALTER ADD CONSTRAINT. 级别: TABLE. 别名: ADD CONSTRAINT
 - ALTER DROP CONSTRAINT. 级别: TABLE. 别名: DROP CONSTRAINT
 - ALTER TTL. 级别: TABLE. 别名: ALTER MODIFY TTL, MODIFY TTL
 - ALTER MATERIALIZE TTL. 级别: TABLE. 别名: MATERIALIZE TTL
 - ALTER SETTINGS. 级别: TABLE. 别名: ALTER SETTING, ALTER MODIFY SETTING, MODIFY SETTING
 - ALTER MOVE PARTITION. 级别: TABLE. 别名: ALTER MOVE PART, MOVE PARTITION, MOVE PART
 - ALTER FETCH PARTITION. 级别: TABLE. 别名: FETCH PARTITION
 - ALTER FREEZE PARTITION. 级别: TABLE. 别名: FREEZE PARTITION
- ALTER VIEW 级别: GROUP
 - ALTER VIEW REFRESH. 级别: VIEW. 别名: ALTER LIVE VIEW REFRESH, REFRESH VIEW
 - ALTER VIEW MODIFY QUERY. 级别: VIEW. 别名: ALTER TABLE MODIFY QUERY

如何对待该层级的示例：

- ALTER 权限包含所有其它 ALTER * 的权限
- ALTER CONSTRAINT 包含 ALTER ADD CONSTRAINT 和 ALTER DROP CONSTRAINT 权限

备注

- MODIFY SETTING 权限允许修改表的引擎设置。它不会影响服务的配置参数
- ATTACH 操作需要 CREATE 权限。
- DETACH 操作需要 DROP 权限。

- 要通过 **KILL MUTATION** 操作来终止mutation, 你需要有发起mutation操作的权限。例如, 当你想终止 **ALTER UPDATE** 操作时, 需要有 **ALTER UPDATE**, **ALTER TABLE**, 或 **ALTER** 权限

CREATE

允许根据下面的权限层级来执行 **CREATE** 和 **ATTACH** DDL语句:

- CREATE**. 级别: **GROUP**
 - CREATE DATABASE**. 级别: **DATABASE**
 - CREATE TABLE**. 级别: **TABLE**
 - CREATE VIEW**. 级别: **VIEW**
 - CREATE DICTIONARY**. 级别: **DICTIONARY**
 - CREATE TEMPORARY TABLE**. 级别: **GLOBAL**

备注

- 删除已创建的表, 用户需要 **DROP** 权限

DROP

允许根据下面的权限层级来执行 **DROP** 和 **DETACH**:

- DROP**. 级别:
 - DROP DATABASE**. 级别: **DATABASE**
 - DROP TABLE**. 级别: **TABLE**
 - DROP VIEW**. 级别: **VIEW**
 - DROP DICTIONARY**. 级别: **DICTIONARY**

TRUNCATE

允许执行 **TRUNCATE**.

权限级别: **TABLE**.

OPTIMIZE

允许执行 **OPTIMIZE TABLE**.

权限级别: **TABLE**.

SHOW

允许根据下面的权限层级来执行 **SHOW**, **DESCRIBE**, **USE**, 和 **EXISTS**:

- SHOW**. 级别: **GROUP**
 - SHOW DATABASES**. 级别: **DATABASE**. 允许执行 **SHOW DATABASES**, **SHOW CREATE DATABASE**, **USE <database>**.
 - SHOW TABLES**. 级别: **TABLE**. 允许执行 **SHOW TABLES**, **EXISTS <table>**, **CHECK <table>**.
 - SHOW COLUMNS**. 级别: **COLUMN**. 允许执行 **SHOW CREATE TABLE**, **DESCRIBE**.
 - SHOW DICTIONARIES**. 级别: **DICTIONARY**. 允许执行 **SHOW DICTIONARIES**, **SHOW CREATE DICTIONARY**, **EXISTS <dictionary>**.

备注

用户同时拥有 SHOW 权限，当用户对指定表，字典或数据库有其它的权限时。

KILL QUERY

允许根据下面的权限层级来执行 KILL:

权限级别: GLOBAL.

备注

KILL QUERY 权限允许用户终止其它用户提交的操作。

访问管理

允许用户执行管理用户/角色和行规则的操作:

- ACCESS MANAGEMENT. 级别: GROUP
 - CREATE USER. 级别: GLOBAL
 - ALTER USER. 级别: GLOBAL
 - DROP USER. 级别: GLOBAL
 - CREATE ROLE. 级别: GLOBAL
 - ALTER ROLE. 级别: GLOBAL
 - DROP ROLE. 级别: GLOBAL
 - ROLE ADMIN. 级别: GLOBAL
 - CREATE ROW POLICY. 级别: GLOBAL. 别名: CREATE POLICY
 - ALTER ROW POLICY. 级别: GLOBAL. 别名: ALTER POLICY
 - DROP ROW POLICY. 级别: GLOBAL. 别名: DROP POLICY
 - CREATE QUOTA. 级别: GLOBAL
 - ALTER QUOTA. 级别: GLOBAL
 - DROP QUOTA. 级别: GLOBAL
 - CREATE SETTINGS PROFILE. 级别: GLOBAL. 别名: CREATE PROFILE
 - ALTER SETTINGS PROFILE. 级别: GLOBAL. 别名: ALTER PROFILE
 - DROP SETTINGS PROFILE. 级别: GLOBAL. 别名: DROP PROFILE
 - SHOW ACCESS. 级别: GROUP
 - SHOW_USERS. 级别: GLOBAL. 别名: SHOW CREATE USER
 - SHOW_ROLES. 级别: GLOBAL. 别名: SHOW CREATE ROLE
 - SHOW_ROW_POLICIES. 级别: GLOBAL. 别名: SHOW POLICIES, SHOW CREATE ROW POLICY, SHOW CREATE POLICY
 - SHOW_QUOTAS. 级别: GLOBAL. 别名: SHOW CREATE QUOTA
 - SHOW_SETTINGS_PROFILES. 级别: GLOBAL. 别名: SHOW PROFILES, SHOW CREATE SETTINGS PROFILE, SHOW CREATE PROFILE

ROLE ADMIN 权限允许用户对角色进行分配以及撤回，包括根据管理选项尚未分配的角色

SYSTEM

允许根据下面的权限层级来执行 **SYSTEM** :

- SYSTEM. 级别: GROUP
 - SYSTEM SHUTDOWN. 级别: GLOBAL. 别名: SYSTEM KILL, SHUTDOWN
 - SYSTEM DROP CACHE. 别名: DROP CACHE
 - SYSTEM DROP DNS CACHE. 级别: GLOBAL. 别名: SYSTEM DROP DNS, DROP DNS CACHE, DROP DNS
 - SYSTEM DROP MARK CACHE. 级别: GLOBAL. 别名: SYSTEM DROP MARK, DROP MARK CACHE, DROP MARKS
 - SYSTEM DROP UNCOMPRESSED CACHE. 级别: GLOBAL. 别名: SYSTEM DROP UNCOMPRESSED, DROP UNCOMPRESSED CACHE, DROP UNCOMPRESSED
 - SYSTEM RELOAD. 级别: GROUP
 - SYSTEM RELOAD CONFIG. 级别: GLOBAL. 别名: RELOAD CONFIG
 - SYSTEM RELOAD DICTIONARY. 级别: GLOBAL. 别名: SYSTEM RELOAD DICTIONARIES, RELOAD DICTIONARY, RELOAD DICTIONARIES
 - SYSTEM RELOAD EMBEDDED DICTIONARIES. 级别: GLOBAL. 别名: RELOAD EMBEDDED DICTIONARIES
 - SYSTEM MERGES. 级别: TABLE. 别名: SYSTEM STOP MERGES, SYSTEM START MERGES, STOP MERGES, START MERGES
 - SYSTEM TTL MERGES. 级别: TABLE. 别名: SYSTEM STOP TTL MERGES, SYSTEM START TTL MERGES, STOP TTL MERGES, START TTL MERGES
 - SYSTEM FETCHES. 级别: TABLE. 别名: SYSTEM STOP FETCHES, SYSTEM START FETCHES, STOP FETCHES, START FETCHES
 - SYSTEM MOVES. 级别: TABLE. 别名: SYSTEM STOP MOVES, SYSTEM START MOVES, STOP MOVES, START MOVES
 - SYSTEM SENDS. 级别: GROUP. 别名: SYSTEM STOP SENDS, SYSTEM START SENDS, STOP SENDS, START SENDS
 - SYSTEM DISTRIBUTED SENDS. 级别: TABLE. 别名: SYSTEM STOP DISTRIBUTED SENDS, SYSTEM START DISTRIBUTED SENDS, STOP DISTRIBUTED SENDS, START DISTRIBUTED SENDS
 - SYSTEM REPLICATED SENDS. 级别: TABLE. 别名: SYSTEM STOP REPLICATED SENDS, SYSTEM START REPLICATED SENDS, STOP REPLICATED SENDS, START REPLICATED SENDS
 - SYSTEM REPLICATION QUEUES. 级别: TABLE. 别名: SYSTEM STOP REPLICATION QUEUES, SYSTEM START REPLICATION QUEUES, STOP REPLICATION QUEUES, START REPLICATION QUEUES
 - SYSTEM SYNC REPLICA. 级别: TABLE. 别名: SYNC REPLICA
 - SYSTEM RESTART REPLICA. 级别: TABLE. 别名: RESTART REPLICA
 - SYSTEM FLUSH. 级别: GROUP
 - SYSTEM FLUSH DISTRIBUTED. 级别: TABLE. 别名: FLUSH DISTRIBUTED
 - SYSTEM FLUSH LOGS. 级别: GLOBAL. 别名: FLUSH LOGS

SYSTEM RELOAD EMBEDDED DICTIONARIES 权限隐式的通过操作 SYSTEM RELOAD DICTIONARY ON *.* 来进行授权.

内省introspection

允许使用 **introspection** 函数.

- INTROSPECTION. 级别: GROUP. 别名: INTROSPECTION FUNCTIONS
 - addressToLine. 级别: GLOBAL
 - addressToSymbol. 级别: GLOBAL
 - demangle. 级别: GLOBAL

数据源

允许在 **table engines** 和 **table functions** 中使用外部数据源。

- SOURCES. 级别: GROUP
 - FILE. 级别: GLOBAL
 - URL. 级别: GLOBAL
 - REMOTE. 级别: GLOBAL
 - YSQL. 级别: GLOBAL
 - ODBC. 级别: GLOBAL
 - JDBC. 级别: GLOBAL
 - HDFS. 级别: GLOBAL
 - S3. 级别: GLOBAL

SOURCES 权限允许使用所有数据源。当然也可以单独对每个数据源进行授权。要使用数据源时，还需要额外的权限。

示例:

- 创建 MySQL table engine, 需要 CREATE TABLE (ON db.table_name) 和 MYSQL权限。4
- 要使用 mysql table function, 需要 CREATE TEMPORARY TABLE 和 MYSQL 权限

dictGet

- dictGet. 别名: dictHas, dictGetHierarchy, dictIsIn

允许用户执行 **dictGet**, **dictHas**, **dictGetHierarchy**, **dictIsIn** 等函数.

权限级别: DICTIONARY.

示例

- GRANT dictGet ON mydb.mydictionary TO john
- GRANT dictGet ON mydictionary TO john

ALL

对规定的实体（列，表，库等）给用户或角色授予所有权限

NONE

不授予任何权限

ADMIN OPTION

ADMIN OPTION 权限允许用户将他们的角色分配给其它用户

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]
```

权限取消

取消用户或角色的权限

语法

取消用户的权限

```
REVOKE [ON CLUSTER cluster_name] privilege[(column_name [,...])] [,....] ON {db.table|db.*|*.*|table|*} FROM {user | CURRENT_USER} [,....] | ALL | ALL EXCEPT {user | CURRENT_USER} [,....]
```

取消用户的角色

```
REVOKE [ON CLUSTER cluster_name] [ADMIN OPTION FOR] role [,....] FROM {user | role | CURRENT_USER} [,....] | ALL | ALL EXCEPT {user_name | role_name | CURRENT_USER} [,....]
```

说明

要取消某些权限，可使用比要撤回的权限更大范围的权限。例如，当用户有 `SELECT (x,y)` 权限时，管理员可执行 `REVOKE SELECT(x,y) ...`, 或 `REVOKE SELECT * ...`, 甚至是 `REVOKE ALL PRIVILEGES ...` 来取消原有权限。

取消部分权限

可以取消部分权限。例如，当用户有 `SELECT *.*` 权限时，可以通过授予对部分库或表的读取权限来撤回原有权限。

示例

授权 `john` 账号能查询所有库的所有表，除了 `account` 库。

```
GRANT SELECT ON *.* TO john;
REVOKE SELECT ON accounts.* FROM john;
```

授权 `mira` 账号能查询 `accounts.staff` 表的所有列，除了 `wage` 这一列。

```
GRANT SELECT ON accounts.staff TO mira;
REVOKE SELECT(wage) ON accounts.staff FROM mira;
```

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.

杂项查询

ATTACH

与 `CREATE` 类似，但有所区别

- 使用关键词 `ATTACH`
- 查询不会在磁盘上创建数据。但会假定数据已经在对应位置存放，同时将与表相关的信息添加到服务器。
执行 `ATTACH` 查询后，服务器将知道表已经被创建。

如果表之前已分离 (`DETACH`)，意味着其结构是已知的，可以使用简要的写法来建立表，即不需要定义表结构的 Schema 细节。

```
ATTACH TABLE [IF NOT EXISTS] [db.]name [ON CLUSTER cluster]
```

启动服务器时会自动触发此查询。

服务器将表的元数据作为文件存储 `ATTACH` 查询，它只是在启动时运行。有些表例外，如系统表，它们是在服务器上显式指定的。

CHECK TABLE

检查表中的数据是否已损坏。

```
CHECK TABLE [db.]name
```

`CHECK TABLE` 查询会比较存储在服务器上的实际文件大小与预期值。如果文件大小与存储的值不匹配，则表示数据已损坏。例如，这可能是由查询执行期间的系统崩溃引起的。

查询返回一行结果，列名为 `result`，该行的值为 `布尔值` 类型：

- 0-表中的数据已损坏；
- 1-数据保持完整性；

该 `CHECK TABLE` 查询支持下表引擎：

- `Log`
- `TinyLog`
- `StripeLog`
- `MergeTree` 家族

对其他不支持的表引擎的表执行会导致异常。

来自 `*Log` 家族的引擎不提供故障自动数据恢复。使用 `CHECK TABLE` 查询及时跟踪数据丢失。

对于 MergeTree 家族引擎，`CHECK TABLE` 查询显示本地服务器上表的每个单独数据部分的检查状态。

如果数据已损坏

如果表已损坏，则可以将未损坏的数据复制到另一个表。要做到这一点：

1. 创建一个与损坏的表结构相同的新表。请执行查询 `CREATE TABLE <new_table_name> AS <damaged_table_name>`。
2. 将 `max_threads` 值设置为 1，以在单个线程中处理下一个查询。要这样做，请运行查询 `SET max_threads = 1`。
3. 执行查询 `INSERT INTO <new_table_name> SELECT * FROM <damaged_table_name>`。此请求将未损坏的数据从损坏的表复制到另一个表。只有损坏部分之前的数据才会被复制。
4. 重新启动 `clickhouse-client` 以重置 `max_threads` 值。

DESCRIBE TABLE

查看表的描述信息，返回各列的 Schema，语法如下：

```
DESC|DESCRIBE TABLE [db.]table [INTO OUTFILE filename] [FORMAT format]
```

返回以下 `String` 类型列：

- `name` — 列名。
- `type` — 列的类型。
- `default_type` — 默认表达式 (DEFAULT, MATERIALIZED 或 ALIAS) 中使用的子句。如果没有指定默认表达式，则列包含一个空字符串。
- `default_expression` — DEFAULT 子句中指定的值。
- `comment_expression` — 注释信息。

嵌套数据结构以“expanded”格式输出。每列分别显示，列名后加点号。

DETACH

从服务器中删除目标表信息（删除对象是表），执行查询后，服务器视作该表已经不存在。

```
DETACH TABLE [IF EXISTS] [db.]name [ON CLUSTER cluster]
```

这不会删除表的数据或元数据。在下一次服务器启动时，服务器将读取元数据并再次查找该表。

也可以不停止服务器的情况下，使用前面介绍的 `ATTACH` 查询来重新关联该表（系统表除外，没有为它们存储元数据）。

DROP

删除已经存在的实体。如果指定 `IF EXISTS`，则如果实体不存在，则不返回错误。

建议使用时添加 `IF EXISTS` 修饰符。

DROP DATABASE

删除 `db` 数据库中的所有表，然后删除 `db` 数据库本身。

语法：

```
DROP DATABASE [IF EXISTS] db [ON CLUSTER cluster]
```

DROP TABLE

删除表。

语法:

```
DROP [TEMPORARY] TABLE [IF EXISTS] [db.]name [ON CLUSTER cluster]
```

DROP DICTIONARY

删除字典。

语法:

```
DROP DICTIONARY [IF EXISTS] [db.]name
```

DROP USER

删除用户。

语法:

```
DROP USER [IF EXISTS] name [...] [ON CLUSTER cluster_name]
```

DROP ROLE

删除角色。

同时该角色所拥有的权限也会被收回。

语法:

```
DROP ROLE [IF EXISTS] name [...] [ON CLUSTER cluster_name]
```

DROP ROW POLICY

删除行策略。

已删除行策略将从分配该策略的所有实体撤销。

语法:

```
DROP [ROW] POLICY [IF EXISTS] name [...] ON [database.]table [...] [ON CLUSTER cluster_name]
```

DROP QUOTA

删除配额。

已删除的配额将从分配该配额的所有实体撤销。

语法:

```
DROP QUOTA [IF EXISTS] name [...] [ON CLUSTER cluster_name]
```

DROP SETTINGS PROFILE

删除settings配置。

已删除的settings配置将从分配该settings配置的所有实体撤销。

语法:

```
DROP [SETTINGS] PROFILE [IF EXISTS] name [...] [ON CLUSTER cluster_name]
```

DROP VIEW

删除视图。视图也可以通过 `DROP TABLE` 删除，但是 `DROP VIEW` 检查 `[db.]name` 是视图。

语法:

```
DROP VIEW [IF EXISTS] [db.]name [ON CLUSTER cluster]
```

EXISTS

```
EXISTS [TEMPORARY] [TABLE|DICTIONARY] [db.]name [INTO OUTFILE filename] [FORMAT format]
```

返回单个 `UInt8` 类型的列，其中包含单个值 `0` 如果表或数据库不存在，或 `1` 如果该表存在于指定的数据库中。

KILL QUERY

```
KILL QUERY [ON CLUSTER cluster]
WHERE <where expression to SELECT FROM system.processes query>
[SYNC|ASYNC|TEST]
[FORMAT format]
```

尝试强制终止当前正在运行的查询。

要终止的查询是使用 `KILL` 查询的 `WHERE` 子句定义的标准从 `system.processes` 表中选择的。

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
```

只读用户只能停止自己提交的查询。

默认情况下，使用异步版本的查询 (`ASYNC`)，不需要等待确认查询已停止。

而相对的，终止同步版本 (`SYNC`) 的查询会显示每步停止时间。

返回信息包含 `kill_status` 列，该列可以采用以下值:

1. ‘finished’ – 查询已成功终止。
2. ‘waiting’ – 发送查询信号终止后，等待查询结束。
3. 其他值，会解释为什么查询不能停止。

测试查询 (`TEST`) 仅检查用户的权限，并显示要停止的查询列表。

KILL MUTATION

```
KILL MUTATION [ON CLUSTER cluster]
WHERE <where expression to SELECT FROM system.mutations query>
[TEST]
[FORMAT format]
```

尝试取消和删除当前正在执行的 **mutations**。要取消的mutation是使用 **KILL** 查询的WHERE子句指定的过滤器从**system.mutations** 表中选择的。

测试查询 (**TEST**) 仅检查用户的权限并显示要停止的mutations列表。

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'
```

当mutation卡住且无法完成时，该查询是有用的(例如，当mutation查询中的某些函数在应用于表中包含的数据时抛出异常)。

Mutation已经做出的更改不会回滚。

OPTIMIZE

```
OPTIMIZE TABLE [db.]name [ON CLUSTER cluster] [PARTITION partition | PARTITION ID 'partition_id'] [FINAL]  
[DEDUPLICATE]
```

此查询尝试初始化 **MergeTree**家族的表引擎的表中未计划合并数据部分。

该 **OPTIMIZE** 查询也支持 **MaterializedView** 和 **Buffer** 引擎。不支持其他表引擎。

当 **OPTIMIZE** 与 **ReplicatedMergeTree** 家族的表引擎一起使用时，ClickHouse将创建一个合并任务，并等待所有节点上的执行（如果 `replication_alter_partitions_sync` 设置已启用）。

- 如果 **OPTIMIZE** 出于任何原因不执行合并，它不通知客户端。要启用通知，请使用 **optimize_throw_if_noop** 设置。
- 如果您指定 **PARTITION**，仅优化指定的分区。[如何设置分区表达式](#)。
- 如果您指定 **FINAL**，即使所有数据已经在一部分中，也会执行优化。
- 如果您指定 **DEDUPLICATE**，则将对完全相同的行进行重复数据删除（所有列进行比较），这仅适用于**MergeTree**引擎。

警告

OPTIMIZE 无法修复 “Too many parts” 错误。

RENAME

重命名一个或多个表。

```
RENAME TABLE [db11.]name11 TO [db12.]name12, [db21.]name21 TO [db22.]name22, ... [ON CLUSTER cluster]
```

所有表都在全局锁定下重命名。重命名表是一个轻型操作。如果您在TO之后指定了另一个数据库，则表将被移动到此数据库。但是，包含数据库的目录必须位于同一文件系统中（否则，将返回错误）。

如果您在一个查询中重命名多个表，这是一个非原子操作，它可能被部分执行，其他会话中的查询可能会接收错误 Table ... doesn't exist ...。

SET

```
SET param = value
```

为当前会话的 **设置** `param` 分配值 `value`。 您不能以这种方式更改 **服务器设置**。

您还可以在单个查询中从指定的设置配置文件中设置所有值。

```
SET profile = 'profile-name-from-the-settings-file'
```

有关详细信息，请参阅 **设置**。

SET ROLE

激活当前用户的角色。

```
SET ROLE {DEFAULT | NONE | role [...] | ALL | ALL EXCEPT role [...]}
```

SET DEFAULT ROLE

将默认角色设置为用户。

默认角色在用户登录时自动激活。 您只能将以前授予的角色设置为默认值。 如果角色没有授予用户，ClickHouse会抛出异常。

```
SET DEFAULT ROLE {NONE | role [...] | ALL | ALL EXCEPT role [...]} TO {user|CURRENT_USER} [...]
```

示例

为用户设置多个默认角色:

```
SET DEFAULT ROLE role1, role2, ... TO user
```

将所有授予的角色设置为用户的默认角色:

```
SET DEFAULT ROLE ALL TO user
```

清除用户的默认角色:

```
SET DEFAULT ROLE NONE TO user
```

将所有授予的角色设置为默认角色，但其中一些角色除外:

```
SET DEFAULT ROLE ALL EXCEPT role1, role2 TO user
```

TRUNCATE

```
TRUNCATE TABLE [IF EXISTS] [db.]name [ON CLUSTER cluster]
```

从表中删除所有数据。 当省略 IF EXISTS子句时，如果该表不存在，则查询返回错误。

该 **TRUNCATE** 查询不支持 **View**, **File**, **URL** 和 **Null** 表引擎。

USE

```
USE db
```

用于设置会话的当前数据库。

当前数据库用于搜索表，如果数据库没有在查询中明确定义与表名之前的点。

使用HTTP协议时无法进行此查询，因为没有会话的概念。

DESCRIBE TABLE Statement

```
DESC|DESCRIBE TABLE [db.]table [INTO OUTFILE filename] [FORMAT format]
```

Returns the following String type columns:

- `name` — Column name.
- `type` — Column type.
- `default_type` — Clause that is used in **default expression** (`DEFAULT`, `MATERIALIZED` or `ALIAS`). Column contains an empty string, if the default expression isn't specified.
- `default_expression` — Value specified in the `DEFAULT` clause.
- `comment_expression` — Comment text.

Nested data structures are output in “expanded” format. Each column is shown separately, with the name after a dot.

Column Manipulations

A set of queries that allow changing the table structure.

Syntax:

```
ALTER TABLE [db].name [ON CLUSTER cluster] ADD|DROP|CLEAR|COMMENT|MODIFY COLUMN ...
```

In the query, specify a list of one or more comma-separated actions.

Each action is an operation on a column.

The following actions are supported:

- **ADD COLUMN** — Adds a new column to the table.
- **DROP COLUMN** — Deletes the column.
- **RENAME COLUMN** — Renames an existing 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.
- **MODIFY COLUMN REMOVE** — Removes one of the column properties.

These actions are described in detail below.

ADD COLUMN

```
ADD COLUMN [IF NOT EXISTS] name [type] [default_expr] [codec] [AFTER name_after | FIRST]
```

Adds a new column to the table with the specified name, type, codec and default_expr (see the section [Default expressions](#)).

If the IF NOT EXISTS clause is included, the query won't return an error if the column already exists. If you specify AFTER name_after (the name of another column), the column is added after the specified one in the list of table columns. If you want to add a column to the beginning of the table use the FIRST clause. Otherwise, the column is added to the end of the table. For a chain of actions, name_after can be the name of a column that is added in one of the previous actions.

Adding a column just changes the table structure, without performing any actions with data. The data does not appear on the disk after ALTER. If the data is missing for a column when reading from the table, it is filled in with default values (by performing the default expression if there is one, or using zeros or empty strings). The column appears on the disk after merging data parts (see [MergeTree](#)).

This approach allows us to complete the ALTER query instantly, without increasing the volume of old data.

Example:

```
ALTER TABLE alter_test ADD COLUMN Added1 UInt32 FIRST;
ALTER TABLE alter_test ADD COLUMN Added2 UInt32 AFTER NestedColumn;
ALTER TABLE alter_test ADD COLUMN Added3 UInt32 AFTER ToDrop;
DESC alter_test FORMAT TSV;
```

```
Added1 UInt32
CounterID UInt32
StartDate Date
UserID UInt32
VisitID UInt32
NestedColumn.A Array(UInt8)
NestedColumn.S Array(String)
Added2 UInt32
ToDrop UInt32
Added3 UInt32
```

DROP COLUMN

```
DROP COLUMN [IF EXISTS] name
```

Deletes the column with the name name. If the IF EXISTS clause is specified, the query won't return an error if the column does not exist.

Deletes data from the file system. Since this deletes entire files, the query is completed almost instantly.

Warning

You can't delete a column if it is referenced by [materialized view](#). Otherwise, it returns an error.

Example:

```
ALTER TABLE visits DROP COLUMN browser
```

RENAME COLUMN

```
RENAME COLUMN [IF EXISTS] name to new_name
```

Renames the column `name` to `new_name`. If the `IF EXISTS` clause is specified, the query won't return an error if the column does not exist. Since renaming does not involve the underlying data, the query is completed almost instantly.

NOTE: Columns specified in the key expression of the table (either with `ORDER BY` or `PRIMARY KEY`) cannot be renamed. Trying to change these columns will produce SQL Error [524].

Example:

```
ALTER TABLE visits RENAME COLUMN webBrowser TO browser
```

CLEAR COLUMN

```
CLEAR COLUMN [IF EXISTS] name IN PARTITION partition_name
```

Resets all data in a column for a specified partition. Read more about setting the partition name in the section [How to specify the partition expression](#).

If the `IF EXISTS` clause is specified, the query won't return an error if the column does not exist.

Example:

```
ALTER TABLE visits CLEAR COLUMN browser IN PARTITION tuple()
```

COMMENT COLUMN

```
COMMENT COLUMN [IF EXISTS] name 'Text comment'
```

Adds a comment to the column. If the `IF EXISTS` clause is specified, the query won't return an error if the column does not exist.

Each column can have one comment. If a comment already exists for the column, a new comment overwrites the previous comment.

Comments are stored in the `comment_expression` column returned by the [DESCRIBE TABLE](#) query.

Example:

```
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] [codec] [TTL] [AFTER name_after | FIRST]
```

This query changes the `name` column properties:

- Type
- Default expression

- Compression Codec
- TTL

For examples of columns compression CODECS modifying, see [Column Compression Codecs](#).

For examples of columns TTL modifying, see [Column TTL](#).

If the `IF EXISTS` clause is specified, the query won't return an error if the column does not exist.

The query also can change the order of the columns using `FIRST | AFTER` clause, see [ADD COLUMN](#) description.

When changing the type, values are converted as if the `toType` functions were applied to them. If only the default expression is changed, the query does not do anything complex, and is completed almost instantly.

Example:

```
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.

The `ALTER` query is atomic. For MergeTree tables it is also lock-free.

The `ALTER` query for changing columns is replicated. The instructions are saved in ZooKeeper, then each replica applies them. All `ALTER` queries are run in the same order. The query waits for the appropriate actions to be completed on the other replicas. However, a query to change columns in a replicated table can be interrupted, and all actions will be performed asynchronously.

MODIFY COLUMN REMOVE

Removes one of the column properties: `DEFAULT`, `ALIAS`, `MATERIALIZED`, `CODEC`, `COMMENT`, `TTL`.

Syntax:

```
ALTER TABLE table_name MODIFY column_name REMOVE property;
```

Example

Remove TTL:

```
ALTER TABLE table_with_ttl MODIFY COLUMN column_ttl REMOVE TTL;
```

See Also

- [REMOVE TTL](#).

Limitations

The `ALTER` query lets you create and delete separate elements (columns) in nested data structures, but not whole nested data structures. To add a nested data structure, you can add columns with a name like `name.nested_name` and the type `Array(T)`. A nested data structure is equivalent to multiple array columns with a name that has the same prefix before the dot.

There is no support for deleting columns in the primary key or the sampling key (columns that are used in the `ENGINE` expression). Changing the type for columns that are included in the primary key is only possible if this change does not cause the data to be modified (for example, you are allowed to add values to an `Enum` or to change a type from `DateTime` to `UInt32`).

If the `ALTER` query is not sufficient to make the table changes you need, you can create a new table, copy the data to it using the `INSERT SELECT` query, then switch the tables using the `RENAME` query and delete the old table. You can use the [clickhouse-copier](#) as an alternative to the `INSERT SELECT` query.

The `ALTER` query blocks all reads and writes for the table. In other words, if a long `SELECT` is running at the time of the `ALTER` query, the `ALTER` query will wait for it to complete. At the same time, all new queries to the same table will wait while this `ALTER` is running.

For tables that do not store data themselves (such as `Merge` and `Distributed`), `ALTER` just changes the table structure, and does not change the structure of subordinate tables. For example, when running `ALTER` for a `Distributed` table, you will also need to run `ALTER` for the tables on all remote servers.

Manipulating Partitions and Parts

The following operations with `partitions` are available:

- `DETACH PARTITION` — Moves a partition to the `detached` directory and forget it.
- `DROP PARTITION` — Deletes a partition.
- `ATTACH PART|PARTITION` — Adds a part or partition from the `detached` directory to the table.
- `ATTACH PARTITION FROM` — Copies the data partition from one table to another and adds.
- `REPLACE PARTITION` — Copies the data partition from one table to another and replaces.
- `MOVE PARTITION TO TABLE` — Moves the data partition from one table to another.
- `CLEAR COLUMN IN PARTITION` — Resets the value of a specified column in a partition.
- `CLEAR INDEX IN PARTITION` — Resets the specified secondary index in a partition.
- `FREEZE PARTITION` — Creates a backup of a partition.
- `UNFREEZE PARTITION` — Removes a backup of a partition.
- `FETCH PARTITION|PART` — Downloads a part or partition from another server.
- `MOVE PARTITION|PART` — Move partition/data part to another disk or volume.
- `UPDATE IN PARTITION` — Update data inside the partition by condition.
- `DELETE IN PARTITION` — Delete data inside the partition by condition.

DETACH PARTITION|PART

```
ALTER TABLE table_name DETACH PARTITION|PART partition_expr
```

Moves all data for the specified partition to the `detached` directory. The server forgets about the detached data partition as if it does not exist. The server will not know about this data until you make the `ATTACH` query.

Example:

```
ALTER TABLE mt DETACH PARTITION '2020-11-21';
ALTER TABLE mt DETACH PART 'all_2_2_0';
```

Read about setting the partition expression in a section [How to specify the partition expression](#).

After the query is executed, you can do whatever you want with the data in the detached directory — delete it from the file system, or just leave it.

This query is replicated – it moves the data to the `detached` directory on all replicas. Note that you can execute this query only on a leader replica. To find out if a replica is a leader, perform the `SELECT` query to the `system.replicas` table. Alternatively, it is easier to make a `DETACH` query on all replicas - all the replicas throw an exception, except the leader replicas (as multiple leaders are allowed).

DROP PARTITION|PART

```
ALTER TABLE table_name DROP PARTITION|PART partition_expr
```

Deletes the specified partition from the table. This query tags the partition as inactive and deletes data completely, approximately in 10 minutes.

Read about setting the partition expression in a section [How to specify the partition expression](#).

The query is replicated – it deletes data on all replicas.

Example:

```
ALTER TABLE mt DROP PARTITION '2020-11-21';
ALTER TABLE mt DROP PART 'all_4_4_0';
```

DROP DETACHED PARTITION|PART

```
ALTER TABLE table_name DROP DETACHED PARTITION|PART partition_expr
```

Removes the specified part or all parts of the specified partition from `detached`.

Read more about setting the partition expression in a section [How to specify the partition expression](#).

ATTACH PARTITION|PART

```
ALTER TABLE table_name ATTACH PARTITION|PART partition_expr
```

Adds data to the table from the `detached` directory. It is possible to add data for an entire partition or for a separate part. Examples:

```
ALTER TABLE visits ATTACH PARTITION 201901;
ALTER TABLE visits ATTACH PART 201901_2_2_0;
```

Read more about setting the partition expression in a section [How to specify the partition expression](#).

This query is replicated. The replica-initiator checks whether there is data in the `detached` directory. If data exists, the query checks its integrity. If everything is correct, the query adds the data to the table.

If the non-initiator replica, receiving the attach command, finds the part with the correct checksums in its own `detached` folder, it attaches the data without fetching it from other replicas.

If there is no part with the correct checksums, the data is downloaded from any replica having the part.

You can put data to the `detached` directory on one replica and use the `ALTER ... ATTACH` query to add it to the table on all replicas.

ATTACH PARTITION FROM

```
ALTER TABLE table2 ATTACH PARTITION partition_expr FROM table1
```

This query copies the data partition from `table1` to `table2`.

Note that data will be deleted neither from `table1` nor from `table2`.

For the query to run successfully, the following conditions must be met:

- Both tables must have the same structure.
- Both tables must have the same partition key.

REPLACE PARTITION

```
ALTER TABLE table2 REPLACE PARTITION partition_expr FROM table1
```

This query copies the data partition from the `table1` to `table2` and replaces existing partition in the `table2`.

Note that data won't be deleted from `table1`.

For the query to run successfully, the following conditions must be met:

- Both tables must have the same structure.
- Both tables must have the same partition key.

MOVE PARTITION TO TABLE

```
ALTER TABLE table_source MOVE PARTITION partition_expr TO TABLE table_dest
```

This query moves the data partition from the `table_source` to `table_dest` with deleting the data from `table_source`.

For the query to run successfully, the following conditions must be met:

- Both tables must have the same structure.
- Both tables must have the same partition key.
- Both tables must be the same engine family (replicated or non-replicated).
- Both tables must have the same storage policy.

CLEAR COLUMN IN PARTITION

```
ALTER TABLE table_name CLEAR COLUMN column_name IN PARTITION partition_expr
```

Resets all values in the specified column in a partition. If the `DEFAULT` clause was determined when creating a table, this query sets the column value to a specified default value.

Example:

```
ALTER TABLE visits CLEAR COLUMN hour IN PARTITION 201902
```

FREEZE PARTITION

```
ALTER TABLE table_name FREEZE [PARTITION partition_expr]
```

This query creates a local backup of a specified partition. If the PARTITION clause is omitted, the query creates the backup of all partitions at once.

Note

The entire backup process is performed without stopping the server.

Note that for old-styled tables you can specify the prefix of the partition name (for example, `2019`) - then the query creates the backup for all the corresponding partitions. Read about setting the partition expression in a section [How to specify the partition expression](#).

At the time of execution, for a data snapshot, the query creates hardlinks to a table data. Hardlinks are placed in the directory `/var/lib/clickhouse/shadow/N/...`, where:

- `/var/lib/clickhouse/` is the working ClickHouse directory specified in the config.
- `N` is the incremental number of the backup.

Note

If you use **a set of disks for data storage in a table**, the `shadow/N` directory appears on every disk, storing data parts that matched by the `PARTITION` expression.

The same structure of directories is created inside the backup as inside `/var/lib/clickhouse/`. The query performs `chmod` for all files, forbidding writing into them.

After creating the backup, you can copy the data from `/var/lib/clickhouse/shadow/` to the remote server and then delete it from the local server. Note that the `ALTER t FREEZE PARTITION` query is not replicated. It creates a local backup only on the local server.

The query creates backup almost instantly (but first it waits for the current queries to the corresponding table to finish running).

`ALTER TABLE t FREEZE PARTITION` copies only the data, not table metadata. To make a backup of table metadata, copy the file `/var/lib/clickhouse/metadata/database/table.sql`

To restore data from a backup, do the following:

1. Create the table if it does not exist. To view the query, use the `.sql` file (replace `ATTACH` in it with `CREATE`).
2. Copy the data from the `data/database/table/` directory inside the backup to the `/var/lib/clickhouse/data/database/table/detached/` directory.
3. Run `ALTER TABLE t ATTACH PARTITION` queries to add the data to a table.

Restoring from a backup does not require stopping the server.

For more information about backups and restoring data, see the [Data Backup](#) section.

UNFREEZE PARTITION

```
ALTER TABLE 'table_name' UNFREEZE [PARTITION 'part_expr'] WITH NAME 'backup_name'
```

Removes freezed partitions with the specified name from the disk. If the PARTITION clause is omitted, the query removes the backup of all partitions at once.

CLEAR INDEX IN PARTITION

```
ALTER TABLE table_name CLEAR INDEX index_name IN PARTITION partition_expr
```

The query works similar to CLEAR COLUMN, but it resets an index instead of a column data.

FETCH PARTITION|PART

```
ALTER TABLE table_name FETCH PARTITION|PART partition_expr FROM 'path-in-zookeeper'
```

Downloads a partition from another server. This query only works for the replicated tables.

The query does the following:

1. Downloads the partition|part from the specified shard. In 'path-in-zookeeper' you must specify a path to the shard in ZooKeeper.
2. Then the query puts the downloaded data to the detached directory of the table_name table. Use the **ATTACH PARTITION|PART** query to add the data to the table.

For example:

1. FETCH PARTITION

```
ALTER TABLE users FETCH PARTITION 201902 FROM '/clickhouse/tables/01-01/visits';
ALTER TABLE users ATTACH PARTITION 201902;
```

2. FETCH PART

```
ALTER TABLE users FETCH PART 201901_2_2_0 FROM '/clickhouse/tables/01-01/visits';
ALTER TABLE users ATTACH PART 201901_2_2_0;
```

Note that:

- The **ALTER ... FETCH PARTITION|PART** query isn't replicated. It places the part or partition to the **detached** directory only on the local server.
- The **ALTER TABLE ... ATTACH** query is replicated. It adds the data to all replicas. The data is added to one of the replicas from the **detached** directory, and to the others - from neighboring replicas.

Before downloading, the system checks if the partition exists and the table structure matches. The most appropriate replica is selected automatically from the healthy replicas.

Although the query is called **ALTER TABLE**, it does not change the table structure and does not immediately change the data available in the table.

MOVE PARTITION|PART

Moves partitions or data parts to another volume or disk for MergeTree-engine tables. See [Using Multiple Block Devices for Data Storage](#).

```
ALTER TABLE table_name MOVE PARTITION|PART partition_expr TO DISK|VOLUME 'disk_name'
```

The `ALTER TABLE t MOVE` query:

- Not replicated, because different replicas can have different storage policies.
- Returns an error if the specified disk or volume is not configured. Query also returns an error if conditions of data moving, that specified in the storage policy, can't be applied.
- Can return an error in the case, when data to be moved is already moved by a background process, concurrent `ALTER TABLE t MOVE` query or as a result of background data merging. A user shouldn't perform any additional actions in this case.

Example:

```
ALTER TABLE hits MOVE PART '20190301_14343_16206_438' TO VOLUME 'slow'  
ALTER TABLE hits MOVE PARTITION '2019-09-01' TO DISK 'fast_ssds'
```

UPDATE IN PARTITION

Manipulates data in the specifies partition matching the specified filtering expression. Implemented as a [mutation](#).

Syntax:

```
ALTER TABLE [db.]table UPDATE column1 = expr1 [, ...] [IN PARTITION partition_id] WHERE filter_expr
```

Example

```
ALTER TABLE mt UPDATE x = x + 1 IN PARTITION 2 WHERE p = 2;
```

See Also

- [UPDATE](#)

DELETE IN PARTITION

Deletes data in the specifies partition matching the specified filtering expression. Implemented as a [mutation](#).

Syntax:

```
ALTER TABLE [db.]table DELETE [IN PARTITION partition_id] WHERE filter_expr
```

Example

```
ALTER TABLE mt DELETE IN PARTITION 2 WHERE p = 2;
```

See Also

- [DELETE](#)

How to Set Partition Expression

You can specify the partition expression in `ALTER ... PARTITION` queries in different ways:

- As a value from the `partition` column of the `system.parts` table. For example, `ALTER TABLE visits DETACH PARTITION 201901`.
- As a tuple of expressions or constants that matches (in types) the table partitioning keys tuple. In the case of a single element partitioning key, the expression should be wrapped in the `tuple(...)` function. For example, `ALTER TABLE visits DETACH PARTITION tuple(toYYYYMM(toDate('2019-01-25')))`.
- Using the partition ID. Partition ID is a string identifier of the partition (human-readable, if possible) that is used as the names of partitions in the file system and in ZooKeeper. The partition ID must be specified in the `PARTITION ID` clause, in a single quotes. For example, `ALTER TABLE visits DETACH PARTITION ID '201901'`.
- In the `ALTER ATTACH PART` and `DROP DETACHED PART` query, to specify the name of a part, use string literal with a value from the `name` column of the `system.detached_parts` table. For example, `ALTER TABLE visits ATTACH PART '201901_1_1_0'`.

Usage of quotes when specifying the partition depends on the type of partition expression. For example, for the `String` type, you have to specify its name in quotes (''). For the `Date` and `Int*` types no quotes are needed.

All the rules above are also true for the `OPTIMIZE` query. If you need to specify the only partition when optimizing a non-partitioned table, set the expression `PARTITION tuple()`. For example:

```
OPTIMIZE TABLE table_not_partitioned PARTITION tuple() FINAL;
```

`IN PARTITION` specifies the partition to which the `UPDATE` or `DELETE` expressions are applied as a result of the `ALTER TABLE` query. New parts are created only from the specified partition. In this way, `IN PARTITION` helps to reduce the load when the table is divided into many partitions, and you only need to update the data point-by-point.

The examples of `ALTER ... PARTITION` queries are demonstrated in the tests `00502_custom_partitioning_local` and `00502_custom_partitioning_replicated_zookeeper`.

Table Settings Manipulations

There is a set of queries to change table settings. You can modify settings or reset them to default values. A single query can change several settings at once.

If a setting with the specified name does not exist, then the query raises an exception.

Syntax

```
ALTER TABLE [db].name [ON CLUSTER cluster] MODIFY|RESET SETTING ...
```

Note

These queries can be applied to `MergeTree` tables only.

MODIFY SETTING

Changes table settings.

Syntax

```
MODIFY SETTING setting_name=value [, ...]
```

Example

```
CREATE TABLE example_table (id UInt32, data String) ENGINE=MergeTree() ORDER BY id;  
ALTER TABLE example_table MODIFY SETTING max_part_loading_threads=8, max_parts_in_total=50000;
```

RESET SETTING

Resets table settings to their default values. If a setting is in a default state, then no action is taken.

Syntax

```
RESET SETTING setting_name [, ...]
```

Example

```
CREATE TABLE example_table (id UInt32, data String) ENGINE=MergeTree() ORDER BY id  
SETTINGS max_part_loading_threads=8;  
ALTER TABLE example_table RESET SETTING max_part_loading_threads;
```

See Also

- [MergeTree settings](#)

ALTER TABLE ... DELETE Statement

```
ALTER TABLE [db.]table [ON CLUSTER cluster] DELETE WHERE filter_expr
```

Deletes data matching the specified filtering expression. Implemented as a [mutation](#).

Note

The `ALTER TABLE` prefix makes this syntax different from most other systems supporting SQL. It is intended to signify that unlike similar queries in OLTP databases this is a heavy operation not designed for frequent use.

The `filter_expr` must be of type `UInt8`. The query deletes rows in the table for which this expression takes a non-zero value.

One query can contain several commands separated by commas.

The synchronicity of the query processing is defined by the `mutations_sync` setting. By default, it is asynchronous.

See also

- [Mutations](#)
- [Synchronicity of ALTER Queries](#)

- [mutations_sync](#) setting

ALTER TABLE ... UPDATE Statements

```
ALTER TABLE [db.]table UPDATE column1 = expr1 [, ...] WHERE filter_expr
```

Manipulates data matching the specified filtering expression. Implemented as a [mutation](#).

Note

The `ALTER TABLE` prefix makes this syntax different from most other systems supporting SQL. It is intended to signify that unlike similar queries in OLTP databases this is a heavy operation not designed for frequent use.

The `filter_expr` must be of type `UInt8`. This query updates values of specified columns to the values of corresponding expressions in rows for which the `filter_expr` takes a non-zero value. Values are casted to the column type using the `CAST` operator. Updating columns that are used in the calculation of the primary or the partition key is not supported.

One query can contain several commands separated by commas.

The synchronicity of the query processing is defined by the [mutations_sync](#) setting. By default, it is asynchronous.

See also

- [Mutations](#)
- [Synchronicity of ALTER Queries](#)
- [mutations_sync](#) setting

Manipulating Key Expressions

```
ALTER TABLE [db].name [ON CLUSTER cluster] MODIFY ORDER BY new_expression
```

The command changes the [sorting key](#) of the table to `new_expression` (an expression or a tuple of expressions). Primary key remains the same.

The command is lightweight in a sense that it only changes metadata. To keep the property that data part rows are ordered by the sorting key expression you cannot add expressions containing existing columns to the sorting key (only columns added by the `ADD COLUMN` command in the same `ALTER` query, without default column value).

Note

It only works for tables in the `MergeTree` family (including [replicated](#) tables).

Manipulating Sampling-Key Expressions

Syntax:

```
ALTER TABLE [db].name [ON CLUSTER cluster] MODIFY SAMPLE BY new_expression
```

The command changes the **sampling key** of the table to `new_expression` (an expression or a tuple of expressions).

The command is lightweight in the sense that it only changes metadata. The primary key must contain the new sample key.

Note

It only works for tables in the **MergeTree** family (including **replicated** tables).

操作数据跳过索引

可以使用以下操作：

- `ALTER TABLE [db].name ADD INDEX name expression TYPE type GRANULARITY value [FIRST|AFTER name]` - 向表元数据添加索引描述。
- `ALTER TABLE [db].name DROP INDEX name` - 从表元数据中删除索引描述并从磁盘中删除索引文件。
- `ALTER TABLE [db.]table MATERIALIZE INDEX name IN PARTITION partition_name` - 查询在分区 `partition_name` 中重建二级索引 `name`。操作为 **mutation**.

前两个命令是轻量级的，它们只更改元数据或删除文件。

Also, they are replicated, syncing indices metadata via ZooKeeper.

此外，它们会被复制，会通过ZooKeeper同步索引元数据。

注意

索引操作仅支持具有以下特征的表 *MergeTree引擎 (包括**replicated**)。

Manipulating Data Skipping Indices

The following operations are available:

- `ALTER TABLE [db].name ADD INDEX name expression TYPE type GRANULARITY value [FIRST|AFTER name]` Adds index description to tables metadata.
- `ALTER TABLE [db].name DROP INDEX name` - Removes index description from tables metadata and deletes index files from disk.
- `ALTER TABLE [db.]table MATERIALIZE INDEX name IN PARTITION partition_name` - The query rebuilds the secondary index `name` in the partition `partition_name`. Implemented as a **mutation**. To rebuild index over the whole data in the table you need to remove `IN PARTITION` from query.

The first two commands are lightweight in a sense that they only change metadata or remove files.

Also, they are replicated, syncing indices metadata via ZooKeeper.

Note

Index manipulation is supported only for tables with ***MergeTree** engine (including **replicated** variants).

Manipulating Constraints

Constraints could be added or deleted using following syntax:

```
ALTER TABLE [db].name ADD CONSTRAINT constraint_name CHECK expression;
ALTER TABLE [db].name DROP CONSTRAINT constraint_name;
```

See more on [constraints](#).

Queries will add or remove metadata about constraints from table so they are processed immediately.

Warning

Constraint check **will not be executed** on existing data if it was added.

All changes on replicated tables are broadcasted to ZooKeeper and will be applied on other replicas as well.

Manipulations with Table TTL

MODIFY TTL

You can change **table TTL** with a request of the following form:

```
ALTER TABLE table_name MODIFY TTL ttl_expression;
```

REMOVE TTL

TTL-property can be removed from table with the following query:

```
ALTER TABLE table_name REMOVE TTL
```

Example

Consider the table with table **TTL**:

```

CREATE TABLE table_with_ttl
(
    event_time DateTime,
    UserID UInt64,
    Comment String
)
ENGINE MergeTree()
ORDER BY tuple()
TTL event_time + INTERVAL 3 MONTH;
SETTINGS min_bytes_for_wide_part = 0;

INSERT INTO table_with_ttl VALUES (now(), 1, 'username1');

INSERT INTO table_with_ttl VALUES (now() - INTERVAL 4 MONTH, 2, 'username2');

```

Run `OPTIMIZE` to force `TTL` cleanup:

```

OPTIMIZE TABLE table_with_ttl FINAL;
SELECT * FROM table_with_ttl FORMAT PrettyCompact;

```

Second row was deleted from table.

event_time	UserID	Comment
2020-12-11 12:44:57	1	username1

Now remove table `TTL` with the following query:

```
ALTER TABLE table_with_ttl REMOVE TTL;
```

Re-insert the deleted row and force the `TTL` cleanup again with `OPTIMIZE`:

```

INSERT INTO table_with_ttl VALUES (now() - INTERVAL 4 MONTH, 2, 'username2');
OPTIMIZE TABLE table_with_ttl FINAL;
SELECT * FROM table_with_ttl FORMAT PrettyCompact;

```

The `TTL` is no longer there, so the second row is not deleted:

event_time	UserID	Comment
2020-12-11 12:44:57	1	username1
2020-08-11 12:44:57	2	username2

See Also

- More about the [TTL-expression](#).
- Modify column [with TTL](#).

ALTER USER

Changes ClickHouse user accounts.

Syntax:

```
ALTER USER [IF EXISTS] name1 [ON CLUSTER cluster_name1] [RENAME TO new_name1]
[, name2 [ON CLUSTER cluster_name2] [RENAME TO new_name2] ...]
[NOT IDENTIFIED | IDENTIFIED {[WITH {no_password | plaintext_password | sha256_password | sha256_hash |
double_sha1_password | double_sha1_hash}] BY {'password' | 'hash'}} | {WITH Ildap SERVER 'server_name'} | {WITH
kerberos [REALM 'realm']}]
[[ADD | DROP] HOST {LOCAL | NAME 'name' | REGEXP 'name_regex' | IP 'address' | LIKE 'pattern'} [,...] | ANY |
NONE]
[DEFAULT ROLE role [,...] | ALL | ALL EXCEPT role [,...] ]
[GRANTEES {user | role | ANY | NONE} [,...] [EXCEPT {user | role} [,...]]]
[SETTINGS variable [= value] [MIN [=] min_value] [MAX [=] max_value] [READONLY | WRITABLE] | PROFILE
'profile_name'] [,...]
```

To use `ALTER USER` you must have the **ALTER USER** privilege.

GRANTEES Clause

Specifies users or roles which are allowed to receive **privileges** from this user on the condition this user has also all required access granted with **GRANT OPTION**. Options of the `GRANTEES` clause:

- `user` — Specifies a user this user can grant privileges to.
- `role` — Specifies a role this user can grant privileges to.
- `ANY` — This user can grant privileges to anyone. It's the default setting.
- `NONE` — This user can grant privileges to none.

You can exclude any user or role by using the `EXCEPT` expression. For example, `ALTER USER user1 GRANTEES ANY EXCEPT user2`. It means if `user1` has some privileges granted with `GRANT OPTION` it will be able to grant those privileges to anyone except `user2`.

Examples

Set assigned roles as default:

```
ALTER USER user DEFAULT ROLE role1, role2
```

If roles aren't previously assigned to a user, ClickHouse throws an exception.

Set all the assigned roles to default:

```
ALTER USER user DEFAULT ROLE ALL
```

If a role is assigned to a user in the future, it will become default automatically.

Set all the assigned roles to default, excepting `role1` and `role2`:

```
ALTER USER user DEFAULT ROLE ALL EXCEPT role1, role2
```

Allows the user with `john` account to grant his privileges to the user with `jack` account:

```
ALTER USER john GRANTEES jack;
```

ALTER QUOTA

Changes quotas.

Syntax:

```
ALTER QUOTA [IF EXISTS] name [ON CLUSTER cluster_name]
  [RENAME TO new_name]
  [KEYED BY {user_name | ip_address | client_key | client_key,user_name | client_key,ip_address} | NOT KEYED]
  [FOR [RANDOMIZED] INTERVAL number {second | minute | hour | day | week | month | quarter | year}
    {MAX { {queries | query_selects | query_inserts | errors | result_rows | result_bytes | read_rows | read_bytes | execution_time} = number } [...] | NO LIMITS | TRACKING ONLY} [...] | TO {role [...] | ALL | ALL EXCEPT role [...]}]
```

Keys `user_name`, `ip_address`, `client_key`, `client_key, user_name` and `client_key, ip_address` correspond to the fields in the [system.quotas](#) table.

Parameters `queries`, `query_selects`, `'query_inserts'`, `errors`, `result_rows`, `result_bytes`, `read_rows`, `read_bytes`, `execution_time`` correspond to the fields in the [system.quotas_usage](#) table.

ON CLUSTER clause allows creating quotas on a cluster, see [Distributed DDL](#).

Examples

Limit the maximum number of queries for the current user with 123 queries in 15 months constraint:

```
ALTER QUOTA IF EXISTS qA FOR INTERVAL 15 month MAX queries = 123 TO CURRENT_USER;
```

For the default user limit the maximum execution time with half a second in 30 minutes, and limit the maximum number of queries with 321 and the maximum number of errors with 10 in 5 quarters:

```
ALTER QUOTA IF EXISTS qB FOR INTERVAL 30 minute MAX execution_time = 0.5, FOR INTERVAL 5 quarter MAX queries = 321, errors = 10 TO default;
```

ALTER ROLE

Changes roles.

Syntax:

```
ALTER ROLE [IF EXISTS] name1 [ON CLUSTER cluster_name1] [RENAME TO new_name1]
  [, name2 [ON CLUSTER cluster_name2] [RENAME TO new_name2] ...]
  [SETTINGS variable [= value] [MIN [=] min_value] [MAX [=] max_value] [READONLY|WRITABLE] | PROFILE
  'profile_name'] [...]
```

ALTER ROW POLICY

Changes row policy.

Syntax:

```
ALTER [ROW] POLICY [IF EXISTS] name1 [ON CLUSTER cluster_name1] ON [database1.]table1 [RENAME TO
new_name1]
  [, name2 [ON CLUSTER cluster_name2] ON [database2.]table2 [RENAME TO new_name2] ...]
  [AS {PERMISSIVE | RESTRICTIVE}]
  [FOR SELECT]
  [USING {condition | NONE}][,...]
  [TO {role [...] | ALL | ALL EXCEPT role [...]}]
```

ALTER SETTINGS PROFILE

Changes settings profiles.

Syntax:

```
ALTER SETTINGS PROFILE [IF EXISTS] TO name1 [ON CLUSTER cluster_name1] [RENAME TO new_name1]
[, name2 [ON CLUSTER cluster_name2] [RENAME TO new_name2] ...]
[SETTINGS variable [= value] [MIN [=] min_value] [MAX [=] max_value] [READONLY|WRITABLE] | INHERIT
'profile_name'] [...]
```

Manipulating Projections

The following operations with **projections** are available:

- **ALTER TABLE [db].name ADD PROJECTION name AS SELECT <COLUMN LIST EXPR> [GROUP BY] [ORDER BY]**- Adds projection description to tables metadata.
- **ALTER TABLE [db].name DROP PROJECTION name** - Removes projection description from tables metadata and deletes projection files from disk.
- **ALTER TABLE [db.]table MATERIALIZE PROJECTION name IN PARTITION partition_name** - The query rebuilds the projection name in the partition partition_name. Implemented as a **mutation**.
- **ALTER TABLE [db.]table CLEAR PROJECTION name IN PARTITION partition_name** - Deletes projection files from disk without removing description.

The commands **ADD**, **DROP** and **CLEAR** are lightweight in a sense that they only change metadata or remove files.

Also, they are replicated, syncing projections metadata via ZooKeeper.

Note

Projection manipulation is supported only for tables with ***MergeTree** engine (including **replicated** variants).

ALTER TABLE ... MODIFY QUERY Statement

You can modify **SELECT** query that was specified when a **materialized view** was created with the **ALTER TABLE ... MODIFY QUERY** statement. Use it when the materialized view was created without the **TO [db.]name** clause. The **allow_experimental_alter_materialized_view_structure** setting must be enabled.

If a materialized view uses the **TO [db.]name** construction, you must **DETACH** the view, run **ALTER TABLE** query for the target table, and then **ATTACH** the previously detached (**DETACH**) view.

Example

```
CREATE TABLE src_table (`a` UInt32) ENGINE = MergeTree ORDER BY a;
CREATE MATERIALIZED VIEW mv (`a` UInt32) ENGINE = MergeTree ORDER BY a AS SELECT a FROM src_table;
INSERT INTO src_table (a) VALUES (1), (2);
SELECT * FROM mv;
```

a
1
2

```
ALTER TABLE mv MODIFY QUERY SELECT a * 2 as a FROM src_table;
INSERT INTO src_table (a) VALUES (3), (4);
SELECT * FROM mv;
```

a
6
8

a
1
2

ALTER LIVE VIEW Statement

ALTER LIVE VIEW ... REFRESH statement refreshes a [Live view](#). See [Force Live View Refresh](#).

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语法

删除现有实体。如果指定了IF EXISTS子句，如果实体不存在，这些查询不会返回错误。

DROP DATABASE

删除db数据库中的所有表，然后删除db数据库本身。

语法:

```
DROP DATABASE [IF EXISTS] db [ON CLUSTER cluster]
```

DROP TABLE

删除数据表

语法:

```
DROP [TEMPORARY] TABLE [IF EXISTS] [db.]name [ON CLUSTER cluster]
```

DROP DICTIONARY

删除字典。

语法:

```
DROP DICTIONARY [IF EXISTS] [db.]name
```

DROP USER

删除用户。

语法:

```
DROP USER [IF EXISTS] name [,....] [ON CLUSTER cluster_name]
```

DROP ROLE

删除角色。删除的角色将从分配给它的所有实体中撤消。

语法:

```
DROP ROLE [IF EXISTS] name [,....] [ON CLUSTER cluster_name]
```

DROP ROW POLICY

删除行策略。删除的行策略从分配到它的所有实体中撤消。

语法:

```
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.

语法:

```
DROP QUOTA [IF EXISTS] name [,....] [ON CLUSTER cluster_name]
```

DROP SETTINGS PROFILE

删除配置文件。已删除的设置配置文件将从分配给它的所有实体中撤销。

语法:

```
DROP [SETTINGS] PROFILE [IF EXISTS] name [,....] [ON CLUSTER cluster_name]
```

DROP VIEW

删除视图。视图也可以通过DROP TABLE命令删除，但DROP VIEW会检查[db.]name是否是一个视图。

语法:

```
DROP VIEW [IF EXISTS] [db.]name [ON CLUSTER cluster]
```

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.

OPTIMIZE Statement

This query tries to initialize an unscheduled merge of data parts for tables.

Warning

OPTIMIZE can't fix the Too many parts error.

Syntax

```
OPTIMIZE TABLE [db.]name [ON CLUSTER cluster] [PARTITION partition | PARTITION ID 'partition_id'] [FINAL]  
[DEDUPLICATE [BY expression]]
```

The OPTIMIZE query is supported for MergeTree family, 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 replicas (if the replication_alter_partitions_sync setting is set to 2) or on current replica (if the replication_alter_partitions_sync setting is set to 1).

- If OPTIMIZE does not perform a merge for any reason, it does not 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. Also merge is forced even if concurrent merges are performed.
- If you specify DEDUPLICATE, then completely identical rows (unless by-clause is specified) will be deduplicated (all columns are compared), it makes sense only for the MergeTree engine.

You can specify how long (in seconds) to wait for inactive replicas to execute OPTIMIZE queries by 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.

BY expression

If you want to perform deduplication on custom set of columns rather than on all, you can specify list of columns explicitly or use any combination of *, **COLUMNS** or **EXCEPT** expressions. The explicitly written or implicitly expanded list of columns must include all columns specified in row ordering expression (both primary and sorting keys) and partitioning expression (partitioning key).

Note

Notice that * behaves just like in SELECT: **MATERIALIZED** and **ALIAS** columns are not used for expansion.

Also, it is an error to specify empty list of columns, or write an expression that results in an empty list of columns, or deduplicate by an **ALIAS** column.

Syntax

```
OPTIMIZE TABLE table DEDUPLICATE; -- all columns
OPTIMIZE TABLE table DEDUPLICATE BY *; -- excludes MATERIALIZED and ALIAS columns
OPTIMIZE TABLE table DEDUPLICATE BY colX,colY,colZ;
OPTIMIZE TABLE table DEDUPLICATE BY * EXCEPT colX;
OPTIMIZE TABLE table DEDUPLICATE BY * EXCEPT (colX, colY);
OPTIMIZE TABLE table DEDUPLICATE BY COLUMNS('column-matched-by-regex');
OPTIMIZE TABLE table DEDUPLICATE BY COLUMNS('column-matched-by-regex') EXCEPT colX;
OPTIMIZE TABLE table DEDUPLICATE BY COLUMNS('column-matched-by-regex') EXCEPT (colX, colY);
```

Examples

Consider the table:

```
CREATE TABLE example (
    primary_key Int32,
    secondary_key Int32,
    value UInt32,
    partition_key UInt32,
    materialized_value UInt32 MATERIALIZED 12345,
    aliased_value UInt32 ALIAS 2,
    PRIMARY KEY primary_key
) ENGINE=MergeTree
PARTITION BY partition_key
ORDER BY (primary_key, secondary_key);
```

```
INSERT INTO example (primary_key, secondary_key, value, partition_key)
VALUES (0, 0, 0, 0), (0, 0, 0, 0), (1, 1, 2, 2), (1, 1, 2, 3), (1, 1, 3, 3);
```

```
SELECT * FROM example;
```

Result:

primary_key	secondary_key	value	partition_key
0	0	0	0
0	0	0	0
1	1	2	2
1	1	2	3
1	1	3	3

When columns for deduplication are not specified, all of them are taken into account. Row is removed only if all values in all columns are equal to corresponding values in previous row:

```
OPTIMIZE TABLE example FINAL DEDUPLICATE;
```

```
SELECT * FROM example;
```

Result:

primary_key	secondary_key	value	partition_key
1	1	2	2
0	0	0	0
1	1	2	3
1	1	3	3

When columns are specified implicitly, the table is deduplicated by all columns that are not ALIAS or MATERIALIZED. Considering the table above, these are `primary_key`, `secondary_key`, `value`, and `partition_key` columns:

```
OPTIMIZE TABLE example FINAL DEDUPLICATE BY *;
```

```
SELECT * FROM example;
```

Result:

primary_key	secondary_key	value	partition_key
1	1	2	2
0	0	0	0
1	1	2	3
1	1	3	3

Deduplicate by all columns that are not ALIAS or MATERIALIZED and explicitly not `value`: `primary_key`, `secondary_key`, and `partition_key` columns.

```
OPTIMIZE TABLE example FINAL DEDUPLICATE BY * EXCEPT value;
```

```
SELECT * FROM example;
```

Result:

primary_key	secondary_key	value	partition_key
1	1	2	2
0	0	0	0

primary_key	secondary_key	value	partition_key
1	1	2	3

Deduplicate explicitly by primary_key, secondary_key, and partition_key columns:

```
OPTIMIZE TABLE example FINAL DEDUPLICATE BY primary_key, secondary_key, partition_key;
```

```
SELECT * FROM example;
```

Result:

primary_key	secondary_key	value	partition_key
1	1	2	2
0	0	0	0
1	1	2	3

Deduplicate by any column matching a regex: primary_key, secondary_key, and partition_key columns:

```
OPTIMIZE TABLE example FINAL DEDUPLICATE BY COLUMNS('.*_key');
```

```
SELECT * FROM example;
```

Result:

primary_key	secondary_key	value	partition_key
0	0	0	0
1	1	2	2
1	1	2	3

RENAME语法

重命名数据库、表或字典。可以在单个查询中重命名多个实体。

请注意，具有多个实体的**RENAME**查询是非原子操作。要以原子方式交换实体名称，请使用**EXCHANGE**语法。

注意

RENAME仅支持**Atomic**数据库引擎。

语法

```
RENAME DATABASE|TABLE|DICTIONARY name TO new_name [,...] [ON CLUSTER cluster]
```

RENAME DATABASE

重命名数据库。

语法

```
RENAME DATABASE atomic_database1 TO atomic_database2 [,...] [ON CLUSTER cluster]
```

RENAME TABLE

重命名一个或多个表

重命名表是一个轻量级的操作。如果在TO之后传递一个不同的数据库，该表将被移动到这个数据库。但是，包含数据库的目录必须位于同一文件系统中。否则，返回错误。

如果在一个查询中重命名多个表，则该操作不是原子操作。可能会部分执行，其他会话中可能会得到Table ... does not exist ... 错误。

语法

```
RENAME TABLE [db1.]name1 TO [db2.]name2 [,...] [ON CLUSTER cluster]
```

示例

```
RENAME TABLE table_A TO table_A_bak, table_B TO table_B_bak;
```

RENAME DICTIONARY

重命名一个或多个词典。此查询可用于在数据库之间移动字典。

语法

```
RENAME DICTIONARY [db0.]dict_A TO [db1.]dict_B [,...] [ON CLUSTER cluster]
```

参考

- [Dictionaries](#)

EXCHANGE语法

以原子方式交换两个表或字典的名称。

此任务也可以通过使用[RENAME](#)来完成，但在这种情况下操作不是原子的。

注意

EXCHANGE仅支持Atomic数据库引擎。

语法

```
EXCHANGE TABLES|DICTIONARIES [db0.]name_A AND [db1.]name_B
```

EXCHANGE TABLES

交换两个表的名称。

语法

```
EXCHANGE TABLES [db0.]table_A AND [db1.]table_B
```

EXCHANGE DICTIONARIES

交换两个字典的名称。

语法

```
EXCHANGE DICTIONARIES [db0.]dict_A AND [db1.]dict_B
```

参考

- [Dictionaries](#)

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.

CREATE DATABASE

该查询用于根据指定名称创建数据库。

```
CREATE DATABASE [IF NOT EXISTS] db_name
```

数据库其实只是用于存放表的一个目录。

如果查询中存在**IF NOT EXISTS**，则当数据库已经存在时，该查询不会返回任何错误。

CREATE TABLE

对于CREATE TABLE，存在以下几种方式。

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
) ENGINE = engine
```

在指定的'db'数据库中创建一个名为'name'的表，如果查询中没有包含'db'，则默认使用当前选择的数据库作为'db'。后面的是包含在括号中的表结构以及表引擎的声明。

其中表结构声明是一个包含一组列描述声明的组合。如果表引擎是支持索引的，那么可以在表引擎的参数中对其进行说明。

在最简单的情况下，列描述是指名称 类型这样的子句。例如：RegionID UInt32。

但是也可以为列另外定义默认值表达式（见后文）。

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name AS [db2.]name2 [ENGINE = engine]
```

创建一个与db2.name2具有相同结构的表，同时你可以对其指定不同的表引擎声明。如果没有表引擎声明，则创建的表将与db2.name2使用相同的表引擎。

```
CREATE TABLE [IF NOT EXISTS] [db.]table_name ENGINE = engine AS SELECT ...
```

使用指定的引擎创建一个与SELECT子句的结果具有相同结构的表，并使用SELECT子句的结果填充它。

以上所有情况，如果指定了**IF NOT EXISTS**，那么在该表已经存在的情况下，查询不会返回任何错误。在这种情况下，查询几乎不会做任何事情。

在ENGINE子句后还可能存在一些其他的子句，更详细的信息可以参考[表引擎](#) 中关于建表的描述。

默认值

在列描述中你可以通过以下方式之一为列指定默认表达式：**DEFAULT expr**，**MATERIALIZED expr**，**ALIAS expr**。

示例：`URLDomain String DEFAULT domain(URL)`。

如果在列描述中未定义任何默认表达式，那么系统将会根据类型设置对应的默认值，如：数值类型为零、字符串类型为空字符串、数组类型为空数组、日期类型为'1970-01-01'以及时间类型为 zero unix timestamp。

如果定义了默认表达式，则可以不定义列的类型。如果没有明确的定义类的类型，则使用默认表达式的类型。例如：`EventDate DEFAULT toDate(EventTime)` - 最终'EventDate'将使用'Date'作为类型。

如果同时指定了默认表达式与列的类型，则将使用类型转换函数将默认表达式转换为指定的类型。例如：`Hits UInt32 DEFAULT 0`与`Hits UInt32 DEFAULT toUInt32(0)`意思相同。

默认表达式可以包含常量或表的任意其他列。当创建或更改表结构时，系统将会运行检查，确保不会包含循环依赖。对于**INSERT**，它仅检查表达式是否是可以解析的 - 它们可以从中计算出所有需要的列的默认值。

DEFAULT expr

普通的默认值，如果INSERT中不包含指定的列，那么将通过表达式计算它的默认值并填充它。

MATERIALIZED expr

物化表达式，被该表达式指定的列不能包含在INSERT的列表中，因为它总是被计算出来的。

对于INSERT而言，不需要考虑这些列。

另外，在SELECT查询中如果包含星号，此列不会被用来替换星号，这是因为考虑到数据转储，在使用SELECT *查询出的结果总能够被'INSERT'回表。

ALIAS expr

别名。这样的列不会存储在表中。

它的值不能够通过INSERT写入，同时使用SELECT查询星号时，这些列也不会被用来替换星号。

但是它们可以显示的用于SELECT中，在这种情况下，在查询分析中别名将被替换。

当使用ALTER查询对添加新的列时，不同于为所有旧数据添加这个列，对于需要在旧数据中查询新列，只会在查询时动态计算这个新列的值。但是如果新列的默认表示中依赖其他列的值进行计算，那么同样会加载这些依赖的列的数据。

如果你向表中添加一个新列，并在之后的一段时间后修改它的默认表达式，则旧数据中的值将会被改变。请注意，在运行后台合并时，缺少的列的值将被计算后写入到合并后的数据部分中。

不能够为nested类型的列设置默认值。

制约因素

随着列描述约束可以定义：

```
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 可以通过任何布尔表达式。如果为表定义了约束，则将为表中的每一行检查它们中的每一行 INSERT query. If any constraint is not satisfied — server will raise an exception with constraint name and checking expression.

添加大量的约束会对big的性能产生负面影响 INSERT 查询。

Ttl表达式

定义值的存储时间。只能为MergeTree系列表指定。有关详细说明，请参阅 [列和表的TTL](#).

列压缩编解ecs

默认情况下，ClickHouse应用以下定义的压缩方法 [服务器设置](#)，列。您还可以定义在每个单独的列的压缩方法 CREATE TABLE 查询。

```
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>
...
```

如果指定了编解ec，则默认编解码器不适用。编解码器可以组合在一个流水线中，例如，CODEC(Delta, ZSTD)。要为您的项目选择最佳的编解码器组合，请通过类似于Altinity中描述的基准测试[新编码提高ClickHouse效率](#)文章。

警告

您无法使用外部实用程序解压缩ClickHouse数据库文件，如 lz4。相反，使用特殊的 [环板compressor](#) 实用程序。

下表引擎支持压缩：

- [MergeTree](#) 家庭
- [日志](#) 家庭
- [设置](#)
- [加入我们](#)

ClickHouse 支持通用编解码器和专用编解ecs。

专业编解ecs

这些编解码器旨在通过使用数据的特定功能使压缩更有效。其中一些编解码器不压缩数据本身。相反，他们准备的数据用于共同目的的编解ec，其压缩它比没有这种准备更好。

专业编解ecs：

- [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 用于存储增量值，所以 delta_bytes 是原始值的最大大小。可能 delta_bytes 值:1,2,4,8. 默认值 delta_bytes 是 sizeof(type) 如果等于1, 2, 4 或8。在所有其他情况下，它是1。
- [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：一个快速、可扩展的内存时间序列数据库](#).
- [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：一个快速、可扩展的内存时间序列数据库](#).
- [T64](#) — Compression approach that crops unused high bits of values in integer data types (including [Enum](#), [Date](#) 和 [DateTime](#))。在算法的每个步骤中，编解码器采用64个值块，将它们放入64x64位矩阵中，对其进行转置，裁剪未使用的值位并将其余部分作为序列返回。未使用的位是使用压缩的整个数据部分的最大值和最小值之间没有区别的位。

[DoubleDelta](#) 和 [Gorilla](#) 编解码器在Gorilla TSDB中用作其压缩算法的组件。大猩猩的方法是有效的情况下，当有缓慢变化的值与他们的时间戳序列。时间戳是由有效地压缩 [DoubleDelta](#) 编解ec，和值有效地由压缩 [Gorilla](#) 编解ec 例如，要获取有效存储的表，可以在以下配置中创建它：

```
CREATE TABLE codec_example
(
    timestamp DateTime CODEC(DoubleDelta),
    slow_values Float32 CODEC(Gorilla)
)
ENGINE = MergeTree()
```

通用编解ecs

编解ecs:

- **NONE** — No compression.
- **LZ4** — Lossless **数据压缩算法** 默认情况下使用。应用**LZ4**快速压缩。
- **LZ4HC[(level)]** — LZ4 HC (high compression) algorithm with configurable level. Default level: 9. Setting **level <= 0** 应用默认级别。可能的水平:[1, 12]。推荐级别范围:[4, 9]。
- **ZSTD[(level)]** — **ZSTD压缩算法** 可配置 **level**. 可能的水平:[1, 22]。默认值:1。

高压缩级别对于非对称场景非常有用，例如压缩一次，重复解压缩。更高的级别意味着更好的压缩和更高的CPU使用率。

临时表

ClickHouse支持临时表，其具有以下特征：

- 当会话结束时，临时表将随会话一起消失，这包含链接中断。
- 临时表仅能够使用**Memory**表引擎。
- 无法为临时表指定数据库。它是在数据库之外创建的。
- 如果临时表与另一个表名称相同，那么当在查询时没有显示的指定**db**的情况下，将优先使用临时表。
- 对于分布式处理，查询中使用的临时表将被传递到远程服务器。

可以使用下面的语法创建一个临时表：

```
CREATE TEMPORARY TABLE [IF NOT EXISTS] table_name [ON CLUSTER cluster]
(
    name1 [type1] [DEFAULT|MATERIALIZED|ALIAS expr1],
    name2 [type2] [DEFAULT|MATERIALIZED|ALIAS expr2],
    ...
)
```

大多数情况下，临时表不是手动创建的，只有在分布式查询处理中使用**(GLOBAL) IN**时为外部数据创建。更多信息，可以参考相关章节。

分布式DDL查询（ON CLUSTER子句）

对于 **CREATE**，**DROP**，**ALTER**，以及**RENAME**查询，系统支持其运行在整个集群上。

例如，以下查询将在**cluster**集群的所有节点上创建名为**all_hits**的**Distributed**表：

```
CREATE TABLE IF NOT EXISTS all_hits ON CLUSTER cluster (p Date, i Int32) ENGINE = Distributed(cluster, default, hits)
```

为了能够正确的运行这种查询，每台主机必须具有相同的**cluster**声明（为了简化配置的同步，你可以使用**zookeeper**的方式进行配置）。同时这些主机还必须链接到**zookeeper**服务器。

这个查询将最终在集群的每台主机上运行，即使一些主机当前处于不可用状态。同时它还保证了所有的查询在单台主机中的执行顺序。

CREATE VIEW

```
CREATE [MATERIALIZED] VIEW [IF NOT EXISTS] [db.]table_name [TO[db.]name] [ENGINE = engine] [POPULATE] AS  
SELECT ...
```

创建一个视图。它存在两种可选择的类型：普通视图与物化视图。

普通视图不存储任何数据，只是执行从另一个表中的读取。换句话说，普通视图只是保存了视图的查询，当从视图中查询时，此查询被作为子查询用于替换FROM子句。

举个例子，假设你已经创建了一个视图：

```
CREATE VIEW view AS SELECT ...
```

还有一个查询：

```
SELECT a, b, c FROM view
```

这个查询完全等价于：

```
SELECT a, b, c FROM (SELECT ...)
```

物化视图存储的数据是由相应的SELECT查询转换得来的。

在创建物化视图时，你还必须指定表的引擎 - 将会使用这个表引擎存储数据。

目前物化视图的工作原理：当将数据写入到物化视图中SELECT子句所指定的表时，插入的数据会通过SELECT子句查询进行转换并将最终结果插入到视图中。

如果创建物化视图时指定了POPULATE子句，则在创建时将该表的数据插入到物化视图中。就像使用CREATE TABLE ... AS SELECT ...一样。否则，物化视图只会包含在物化视图创建后的新写入的数据。我们不推荐使用POPULATE，因为在视图创建期间写入的数据将不会写入其中。

当一个SELECT子句包含DISTINCT, GROUP BY, ORDER BY, LIMIT时，请注意，这些仅会在插入数据时在每个单独的数据块上执行。例如，如果你在其中包含了GROUP BY，则只会在查询期间进行聚合，但聚合范围仅限于单个批的写入数据。数据不会进一步被聚合。但是当你使用一些其他数据聚合引擎时这是例外的，如：SummingMergeTree。

目前对物化视图执行ALTER是不支持的。因此这可能是不方便的。如果物化视图是使用的TO [db.]name的方式进行构建的，你可以使用DETACH语句先将视图剥离，然后使用ALTER运行在目标表上，然后使用ATTACH将之前剥离的表重新加载进来。

视图看起来和普通的表相同。例如，你可以通过SHOW TABLES查看到它们。

没有单独的删除视图的语法。如果要删除视图，请使用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})
```

INSERT INTO 语句

INSERT INTO 语句主要用于向系统中添加数据。

查询的基本格式：

```
INSERT INTO [db.]table [(c1, c2, c3)] VALUES (v11, v12, v13), (v21, v22, v23), ...
```

您可以在查询中指定要插入的列的列表，如：[(c1, c2, c3)]。您还可以使用列匹配器的表达式，例如*和/或修饰符，例如 APPLY, EXCEPT, REPLACE。

例如，考虑该表：

```
SHOW CREATE insert_select_testtable;
```

```
CREATE TABLE insert_select_testtable
(
    `a` Int8,
    `b` String,
    `c` Int8
)
ENGINE = MergeTree()
ORDER BY a
```

```
INSERT INTO insert_select_testtable (*) VALUES (1, 'a', 1);
```

如果要在除了'b'列以外的所有列中插入数据，您需要传递和括号中选择的列数一样多的值：

```
INSERT INTO insert_select_testtable (* EXCEPT(b)) Values (2, 2);
```

```
SELECT * FROM insert_select_testtable;
```

a	b	c
2		2

a	b	c
1	a	1

在这个示例中，我们看到插入的第二行的a和c列的值由传递的值填充，而b列由默认值填充。

对于存在于表结构中但不存在于插入列表中的列，它们将会按照如下方式填充数据：

- 如果存在DEFAULT表达式，根据DEFAULT表达式计算被填充的值。
- 如果没有定义DEFAULT表达式，则填充零或空字符串。

如果 strict_insert_defaults=1，你必须在查询中列出所有没有定义DEFAULT表达式的列。

数据可以以ClickHouse支持的任何 输入输出格式 传递给INSERT。格式的名称必须显示的指定在查询中：

```
INSERT INTO [db.]table [(c1, c2, c3)] FORMAT format_name data_set
```

例如，下面的查询所使用的输入格式就与上面INSERT ... VALUES的中使用的输入格式相同：

```
INSERT INTO [db.]table [(c1, c2, c3)] FORMAT Values (v11, v12, v13), (v21, v22, v23), ...
```

ClickHouse会清除数据前所有的空白字符与一行摘要信息（如果需要的话）。所以在进行查询时，我们建议您将数据放入到输入输出格式名称后的新的一行中去（如果数据是以空白字符开始的，这将非常重要）。

示例：

```
INSERT INTO t FORMAT TabSeparated
11 Hello, world!
22 Qwerty
```

在使用命令行客户端或HTTP客户端时，你可以将具体的查询语句与数据分开发送。更多具体信息，请参考[«客户端»部分](#)。

使用SELECT的结果写入

```
INSERT INTO [db.]table [(c1, c2, c3)] SELECT ...
```

写入与SELECT的列的对应关系是使用位置来进行对应的，尽管它们在SELECT表达式与INSERT中的名称可能是不同的。如果需要，会对它们执行对应的类型转换。

除了VALUES格式之外，其他格式中的数据都不允许出现诸如now()，1 + 2等表达式。VALUES格式允许您有限度的使用这些表达式，但是不建议您这么做，因为执行这些表达式总是低效的。

系统不支持的其他用于修改数据的查询：UPDATE, DELETE, REPLACE, MERGE, UPSERT, INSERT UPDATE。
但是，您可以使用ALTER TABLE ... DROP PARTITION查询来删除一些旧的数据。

性能的注意事项

在进行INSERT时将会对写入的数据进行一些处理，按照主键排序，按照月份对数据进行分区等。所以如果在您的写入数据中包含多个月份的混合数据时，将会显著的降低INSERT的性能。为了避免这种情况：

- 数据总是以尽量大的batch进行写入，如每次写入100,000行。
- 数据在写入ClickHouse前预先的对数据进行分组。

在以下的情况下，性能不会下降：

- 数据总是被实时的写入。
- 写入的数据已经按照时间排序。

分布式DDL查询(ON CLUSTER条件)

默认情况下，CREATE、DROP、ALTER和RENAME查询仅影响执行它们的当前服务器。在集群设置中，可以使用ON CLUSTER子句以分布式方式运行此类查询。

例如，以下查询在cluster中的每个主机上创建all_hits Distributed表：

```
CREATE TABLE IF NOT EXISTS all_hits ON CLUSTER cluster (p Date, i Int32) ENGINE = Distributed(cluster, default, hits)
```

为了正确运行这些查询，每个主机必须具有相同的集群定义（为了简化同步配置，您可以使用ZooKeeper替换）。他们还必须连接到ZooKeeper服务器。

本地版本的查询最终会在集群中的每台主机上执行，即使某些主机当前不可用。

警告

在单个主机内执行查询的顺序是有保证的。

函数

ClickHouse中至少存在两种类型的函数 - 常规函数（它们称之为«函数»）和聚合函数。常规函数的工作就像分别为每一行执行一次函数计算一样（对于每一行，函数的结果不依赖于其他行）。聚合函数则从各行累积一组值（即函数的结果以来整个结果集）。

在本节中，我们将讨论常规函数。有关聚合函数，请参阅«聚合函数»一节。

* - 'arrayJoin'函数与表函数均属于第三种类型的函数。 *

强类型

与标准SQL相比，ClickHouse具有强类型。换句话说，它不会在类型之间进行隐式转换。每个函数适用于特定的一组类型。这意味着有时您需要使用类型转换函数。

常见的子表达式消除

查询中具有相同AST（相同语句或语法分析结果相同）的所有表达式都被视为具有相同的值。这样的表达式被连接并执行一次。通过这种方式也可以消除相同的子查询。

结果类型

所有函数都只能够返回一个返回值。结果类型通常由参数的类型决定。但tupleElement函数（`a.N`运算符）和toFixedString函数是例外的。

常量

为了简单起见，某些函数的某些参数只能是常量。例如，LIKE运算符的右参数必须是常量。

几乎所有函数都为常量参数返回常量。除了用于生成随机数的函数。

'now'函数为在不同时间运行的查询返回不同的值，但结果被视为常量，因为常量在单个查询中很重要。

常量表达式也被视为常量（例如，LIKE运算符的右半部分可以由多个常量构造）。

对于常量和非常量参数，可以以不同方式实现函数（执行不同的代码）。但是，对于包含相同数据的常量和非常量参数它们的结果应该是一致的。

NULL值处理

函数具有以下行为：

- 如果函数的参数至少一个是«NULL»，则函数结果也是«NULL»。
- 在每个函数的描述中单独指定的特殊行为。在ClickHouse源代码中，这些函数具有«UseDefaultImplementationForNulls = false»。

不可变性

函数不能更改其参数的值 - 任何更改都将作为结果返回。因此，计算单独函数的结果不依赖于在查询中写入函数的顺序。

错误处理

如果数据无效，某些函数可能会抛出异常。在这种情况下，将取消查询并将错误信息返回给客户端。对于分布式处理，当其中一个服务器发生异常时，其他服务器也会尝试中止查询。

表达式参数的计算

在几乎所有编程语言中，某些函数可能无法预先计算其中一个参数。这通常是运算符`&&`，`||`和`? :`。

但是在ClickHouse中，函数（运算符）的参数总是被预先计算。这是因为一次评估列的整个部分，而不是分别计算每一行。

执行分布式查询处理的功能

对于分布式查询处理，在远程服务器上执行尽可能多的查询处理阶段，并且在请求者服务器上执行其余阶段（合并中间结果和之后的所有内容）。

这意味着可以在不同的服务器上执行功能。

例如，在查询`SELECT f (sum (g (x))) FROM distributed_table GROUP BY h (y)` 中，

- 如果`distributed_table`至少有两个分片，则在远程服务器上执行函数'g'和'h'，并在请求服务器上执行函数'f'。
- 如果`distributed_table`只有一个分片，则在该分片的服务器上执行所有'f'，'g'和'h'功能。

函数的结果通常不依赖于它在哪个服务器上执行。但是，有时这很重要。

例如，使用字典的函数时将使用运行它们的服务器上存在的字典。

另一个例子是`hostName`函数，它返回运行它的服务器的名称，以便在`SELECT`查询中对服务器进行`GROUP BY`。

如果查询中的函数在请求服务器上执行，但您需要在远程服务器上执行它，则可以将其包装在«any»聚合函数中，或将其添加到«GROUP BY»中。

算术函数

对于所有算术函数，结果类型为结果适合的最小数值类型（如果存在这样的类型）。最小数值类型是根据数值的位数，是否有符号以及是否是浮点类型而同时进行的。如果没有足够的位，则采用最高位类型。

例如：

```
SELECT toTypeName(0), toTypeName(0 + 0), toTypeName(0 + 0 + 0), toTypeName(0 + 0 + 0 + 0)
```

toTypeName(0)	toTypeName(plus(0, 0))	toTypeName(plus(plus(0, 0), 0))	toTypeName(plus(plus(plus(0, 0), 0), 0))
UInt8	UInt16	UInt32	UInt64

算术函数适用于`UInt8`，`UInt16`，`UInt32`，`UInt64`，`Int8`，`Int16`，`Int32`，`Int64`，`Float32`或`Float64`中的任何类型。

溢出的产生方式与C++相同。

plus(a, b), a + b operator

计算数值的总和。

您还可以将`Date`或`DateTime`与整数进行相加。在`Date`的情况下，和整数相加整数意味着添加相应的天数。对于`DateTime`，这意味着添加相应的秒数。

minus(a, b), a - b operator

计算数值之间的差，结果总是有符号的。

您还可以将`Date`或`DateTime`与整数进行相减。见上面的'plus'。

multiply(a, b), a * b operator

计算数值的乘积。

divide(a, b), a / b operator

计算数值的商。结果类型始终是浮点类型。

它不是整数除法。对于整数除法，请使用'intDiv'函数。

当除以零时，你得到'inf'，'- inf'或'nan'。

intDiv(a,b)

计算数值的商，向下舍入取整（按绝对值）。

除以零或将最小负数除以-1时抛出异常。

intDivOrZero(a,b)

与'intDiv'的不同之处在于它在除以零或将最小负数除以-1时返回零。

modulo(a, b), a % b operator

计算除法后的余数。

如果参数是浮点数，则通过删除小数部分将它们预转换为整数。

其余部分与C++中的含义相同。截断除法用于负数。

除以零或将最小负数除以-1时抛出异常。

moduloOrZero(a, b)

和**modulo**不同之处在于，除以0时结果返回0

negate(a), -a operator

通过改变数值的符号位对数值取反，结果总是有符号的

abs(a)

计算数值 (a) 的绝对值。也就是说，如果 $a < 0$ ，它返回- a 。对于无符号类型，它不执行任何操作。对于有符号整数类型，它返回无符号数。

gcd(a,b)

返回数值的最大公约数。

除以零或将最小负数除以-1时抛出异常。

lcm(a,b)

返回数值的最小公倍数。

除以零或将最小负数除以-1时抛出异常。

比较函数

比较函数始终返回0或1 (UInt8)。

可以比较以下类型：

- 数字

- String 和 FixedString

- 日期

- 日期时间

以上每个组内的类型均可互相比较，但是对于不同组的类型间不能够进行比较。

例如，您无法将日期与字符串进行比较。您必须使用函数将字符串转换为日期，反之亦然。

字符串按字节进行比较。较短的字符串小于以其开头并且至少包含一个字符的所有字符串。

等于, `a=b` 和 `a==b` 运算符

不等于, `a!=b` 和 `a<>b` 运算符

少, `<` 运算符

大于, `>` 运算符

小于等于, `<=` 运算符

大于等于, `>=` 运算符

逻辑函数

逻辑函数可以接受任何数字类型的参数，并返回 UInt8 类型的 0 或 1。

当向函数传递零时，函数将判定为 «false»，否则，任何其他非零的值都将被判定为 «true»。

和, `AND` 运算符

或, `OR` 运算符

非, `NOT` 运算符

异或, `XOR` 运算符

类型转换函数

数值类型转换常见的问题

当你把一个值从一个类型转换为另外一个类型的时候，你需要注意的是这是一个不安全的操作，可能导致数据的丢失。数据丢失一般发生在你将一个大的数据类型转换为小的数据类型的时候，或者你把两个不同的数据类型相互转换的时候。

ClickHouse 和 C++ 有相同的类型转换行为。

`toInt(8|16|32|64)`

转换一个输入值为 Int 类型。这个函数包括：

- `toInt8(expr)` — 结果为 Int8 数据类型。
- `toInt16(expr)` — 结果为 Int16 数据类型。
- `toInt32(expr)` — 结果为 Int32 数据类型。
- `toInt64(expr)` — 结果为 Int64 数据类型。

参数

- `expr` — 表达式 返回一个数字或者代表数值类型的字符串。不支持二进制、八进制、十六进制的数字形式，有效数字之前的0也会被忽略。

返回值

整形在 `Int8`, `Int16`, `Int32`, 或者 `Int64` 的数据类型。

函数使用 **rounding towards zero** 原则，这意味着会截断丢弃小数部分的数值。

`Nan` 和 `Inf` 转换是不确定的。具体使用的时候，请参考 [数值类型转换常见的问题](#)。

例子

```
SELECT tolnt64(nan), tolnt32(32), tolnt16('16'), tolnt8(8.8)
```

tolnt64(nan)	tolnt32(32)	tolnt16('16')	tolnt8(8.8)
-9223372036854775808	32	16	8

tolnt(8|16|32|64)OrZero

这个函数需要一个字符类型的入参，然后尝试把它转为 `Int (8 | 16 | 32 | 64)`，如果转换失败直接返回0。

例子

```
select tolnt64OrZero('123123'), tolnt8OrZero('123qwe123')
```

tolnt64OrZero('123123')	tolnt8OrZero('123qwe123')
123123	0

tolnt(8|16|32|64)OrNull

这个函数需要一个字符类型的入参，然后尝试把它转为 `Int (8 | 16 | 32 | 64)`，如果转换失败直接返回 `NULL`。

例子

```
select tolnt64OrNull('123123'), tolnt8OrNull('123qwe123')
```

tolnt64OrNull('123123')	tolnt8OrNull('123qwe123')
123123	NULL

toUInt(8|16|32|64)

转换一个输入值到 `UInt` 类型。这个函数包括：

- `toUInt8(expr)` — 结果为 `UInt8` 数据类型。
- `toUInt16(expr)` — 结果为 `UInt16` 数据类型。
- `toUInt32(expr)` — 结果为 `UInt32` 数据类型。
- `toUInt64(expr)` — 结果为 `UInt64` 数据类型。

参数

- `expr` — 表达式返回一个数字或者代表数值类型的字符串。不支持二进制、八进制、十六进制的数字形式，有效数字之前的0也会被忽略。

返回值

整形在 `UInt8`, `UInt16`, `UInt32`, 或者 `UInt64` 的数据类型。

函数使用 **rounding towards zero** 原则，这意味着会截断丢弃小数部分的数值。

对于负数和 **NaN and Inf** 来说转换的结果是不确定的。如果你传入一个负数，比如：`'-32'`，ClickHouse 会抛出异常。具体使用的时候，请参考 [数值类型转换常见的问题](#)。

例子

```
SELECT toUInt64(nan), toUInt32(-32), toUInt16('16'), toUInt8(8.8)
```

toUInt64(nan)	toUInt32(-32)	toUInt16('16')	toUInt8(8.8)
9223372036854775808	4294967264	16	8

`toUInt(8|16|32|64)OrZero`

`toUInt(8|16|32|64)OrNull`

`toFloat(32|64)`

`toFloat(32|64)OrZero`

`toFloat(32|64)OrNull`

`toDate`

`toDateOrZero`

`toDateOrNull`

`dateTime`

`dateTimeOrZero`

`dateTimeOrNull`

`toDecimal(32|64|128)`

转换 `value` 到 **Decimal** 类型的值，其中精度为 `S`。`value` 可以是一个数字或者一个字符串。`S` 指定小数位的精度。

- `toDecimal32(value, S)`
- `toDecimal64(value, S)`
- `toDecimal128(value, S)`

`toDecimal(32|64|128)OrNull`

转换一个输入的字符到 **Nullable(Decimal(P,S))** 类型的数据。这个函数包括：

- `toDecimal32OrNull(expr, S)` — 结果为 `Nullable(Decimal32(S))` 数据类型。
- `toDecimal64OrNull(expr, S)` — 结果为 `Nullable(Decimal64(S))` 数据类型。
- `toDecimal128OrNull(expr, S)` — 结果为 `Nullable(Decimal128(S))` 数据类型。

如果在解析输入值发生错误的时候你希望得到一个 `NONE` 值而不是抛出异常，你可以使用该函数。

参数

- `expr` — 表达式 返回一个 `String` 类型的数据。 ClickHouse 倾向于文本类型的表示带小数类型的数值，比如 '`1.111`'。
- `S` — 小数位的精度。

返回值

`Nullable(Decimal(P,S))` 类型的数据，包括：

- 如果有的话，小数位 `S`。
- 如果解析错误或者输入的数字的小数位多于 `S`，那结果为 `NONE`。

例子

```
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

转换输入值为 `Decimal(P,S)` 类型数据。这个函数包括：

- `toDecimal32OrZero(expr, S)` — 结果为 `Decimal32(S)` 数据类型。
- `toDecimal64OrZero(expr, S)` — 结果为 `Decimal64(S)` 数据类型。
- `toDecimal128OrZero(expr, S)` — 结果为 `Decimal128(S)` 数据类型。

当解析错误的时候，你不需要抛出异常而希望得到 `0` 值，你可以使用该函数。

参数

- `expr` — 表达式 返回一个 `String` 类型的数据。 ClickHouse 倾向于文本类型的表示带小数类型的数值，比如 '`1.111`'。
- `S` — 小数位的精度。

返回值

A value in the `Nullable(Decimal(P,S))` data type. The value contains:

- 如果有的话，小数位 `S`。

- 如果解析错误或者输入的数字的小数位多于S,那结果为小数位精度为S的0。

例子

```
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

这些函数用于在数字、字符串（不包含`FixedString`）`Date`以及`DateTime`之间互相转换。
所有的函数都接受一个参数。

当将其他类型转换到字符串或从字符串转换到其他类型时，使用与`TabSeparated`格式相同的规则对字符串的值进行格式化或解析。如果无法解析字符串则抛出异常并取消查询。

当将`Date`转换为数字或反之，`Date`对应Unix时间戳的天数。

将`DateTime`转换为数字或反之，`DateTime`对应Unix时间戳的秒数。

`toDate/toDateTime`函数的日期和日期时间格式定义如下：

YYYY-MM-DD
YYYY-MM-DD hh:mm:ss

例外的是，如果将`UInt32`、`Int32`、`UInt64`或`Int64`类型的数值转换为`Date`类型，并且其对应的值大于等于65536，则该数值将被解析成unix时间戳（而不是对应的天数）。这意味着允许写入`'toDate(unix_timestamp)'`这种常见情况，否则这将是错误的，并且需要便携更加繁琐的`'toDate(toDateTime(unix_timestamp))'`。

`Date`与`DateTime`之间的转换以更为自然的方式进行：通过添加空的`time`或删除`time`。

数值类型之间的转换与C++中不同数字类型之间的赋值相同的规则。

此外，`DateTime`参数的`toString`函数可以在第二个参数中包含时区名称。例如：`Asia/Yekaterinburg`在这种情况下，时间根据指定的时区进行格式化。

```
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

另请参阅`toUnixTimestamp`函数。

toFixedString(s,N)

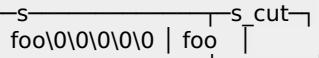
将String类型的参数转换为FixedString(N)类型的值（具有固定长度N的字符串）。N必须是一个常量。如果字符串的字节数少于N，则向右填充空字节。如果字符串的字节数多于N，则抛出异常。

toStringCutToZero(s)

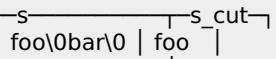
接受String或FixedString参数。返回String，其内容在找到的第一个零字节处被截断。

示例：

```
SELECT toFixedString('foo', 8) AS s, toStringCutToZero(s) AS s_cut
```



```
SELECT toFixedString('foo\0bar', 8) AS s, toStringCutToZero(s) AS s_cut
```



reinterpretAsUInt(8|16|32|64)

reinterpretAsInt(8|16|32|64)

reinterpretAsFloat(32|64)

reinterpretAsDate

reinterpretAsDateTime

这些函数接受一个字符串，并将放在字符串开头的字节解释为主机顺序中的数字（little endian）。如果字符串不够长，则函数就像使用必要数量的空字节填充字符串一样。如果字符串比需要的长，则忽略额外的字节。Date被解释为Unix时间戳的天数，DateTime被解释为Unix时间戳。

reinterpretAsString

此函数接受数字、Date或DateTime，并返回一个字符串，其中包含表示主机顺序（小端）的相应值的字节。从末尾删除空字节。例如，UInt32类型值255是一个字节长的字符串。

reinterpretAsFixedString

此函数接受数字、Date或DateTime，并返回包含表示主机顺序（小端）的相应值的字节的FixedString。从末尾删除空字节。例如，UInt32类型值255是一个长度为一个字节的FixedString。

CAST(x, T)

将'x'转换为't'数据类型。还支持语法CAST (x AS t)

示例：

```
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
```

timestamp	datetime	date	string	fixed_string
2016-06-15 23:00:00	2016-06-15 23:00:00	2016-06-15	2016-06-15 23:00:00	2016-06-15 23:00:00\0\0\0

将参数转换为 FixedString(N)，仅适用于 String 或 FixedString(N) 类型的参数。

支持将数据转换为 可为空。例如：

```
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(Year|Quarter|Month|Week|Day|Hour|Minute|Second)

把一个数值类型的值转换为 Interval 类型的数据。

语法

```
toIntervalSecond(number)
toIntervalMinute(number)
toIntervalHour(number)
toIntervalDay(number)
toIntervalWeek(number)
toIntervalMonth(number)
toIntervalQuarter(number)
toIntervalYear(number)
```

参数

- number — 正整数，持续的时间。

返回值

- 时间的 Interval 值。

例子

```
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

把 **String** 类型的时间日期转换为 **DateTime** 数据类型。

该函数可以解析 **ISO 8601**，**RFC 1123 - 5.2.14 RFC-822 Date and Time Specification** 或者 ClickHouse 的一些别的时间日期格式。

语法

```
parseDateTimeBestEffort(time_string [, time_zone]);
```

参数

- **time_string** — 字符类型的时间和日期。
- **time_zone** — 字符类型的时区。

非标准格式的支持

- 9位或者10位的数字时间，**unix timestamp**。
- 时间和日期组成的字符串：YYYYMMDDhhmmss, DD/MM/YYYY hh:mm:ss, DD-MM-YY hh:mm, YYYY-MM-DD hh:mm:ss 等。
- 只有日期的字符串：YYYY, YYYYMM, YYYY*MM, DD/MM/YYYY, DD-MM-YY 等。
- 只有天和时间：DD, DD hh, DD hh:mm。这种情况下 YYYY-MM 默认为 2000-01。
- 包含时间日期以及时区信息：YYYY-MM-DD hh:mm:ss ±hh:mm 等。例如：2020-12-12 17:36:00 -5:00。

对于所有的格式来说，这个函数通过全称或者第一个三个字符的月份名称来解析月份，比如：24/DEC/18, 24-Dec-18, 01-September-2018。

返回值

- **DateTime** 类型数据。

例子

查询：

```
SELECT parseDateTimeBestEffort('12/12/2020 12:12:57')
AS parseDateTimeBestEffort;
```

结果：

```
parseDateTimeBestEffort
2020-12-12 12:12:57 |
```

查询：

```
SELECT parseDateTimeBestEffort('Sat, 18 Aug 2018 07:22:16 GMT', 'Europe/Moscow')
AS parseDateTimeBestEffort
```

结果：

```
parseDateTimeBestEffort
2018-08-18 10:22:16 |
```

查询：

```
SELECT parseDateTimeBestEffort('1284101485')
AS parseDateTimeBestEffort
```

结果：

```
parseDateTimeBestEffort
2015-07-07 12:04:41 |
```

查询：

```
SELECT parseDateTimeBestEffort('2018-12-12 10:12:12')
AS parseDateTimeBestEffort
```

结果：

```
parseDateTimeBestEffort
2018-12-12 10:12:12 |
```

查询：

```
SELECT parseDateTimeBestEffort('10 20:19')
```

结果：

```
parseDateTimeBestEffort('10 20:19')
2000-01-10 20:19:00 |
```

除此之外

- ISO 8601 announcement by @xkcd
- RFC 1123
- toDate

- [toDateTime](#)

parseDateTimeBestEffortOrNull

这个函数和[parseDateTimeBestEffort](#)基本一致，除了无法解析返回结果为NULL。

parseDateTimeBestEffortOrZero

这个函数和[parseDateTimeBestEffort](#)基本一致，除了无法解析返回结果为0。

toLowCardinality

把输入值转换为[LowCardinality](#)的相同类型的数据。

如果要把[LowCardinality](#)类型的数据转换为其他类型，使用[CAST](#)函数。比如：`CAST(x as String)`。

语法

```
toLowCardinality(expr)
```

参数

- `expr` — 表达式为支持的数据类型的一种。

返回值

- `expr`的结果。

类型：`LowCardinality(expr_result_type)`

例子

查询：

```
SELECT toLowCardinality('1')
```

结果：

```
toLowCardinality('1')  
1
```

toUnixTimestamp64Milli

toUnixTimestamp64Micro

toUnixTimestamp64Nano

把一个[DateTime64](#)类型的数据转换为[Int64](#)类型的数据，结果包含固定亚秒的精度。输入的值是变大还是变低依赖于输入的精度。需要注意的是输出的值是一个UTC的时间戳，不是同一个时区的[DateTime64](#)值。

语法

```
toUnixTimestamp64Milli(value)
```

参数

- `value` — 任何精度的DateTime64类型的数据。

返回值

- `value` Int64类型数据。

例子

查询:

```
WITH toDateTime64('2019-09-16 19:20:12.345678910', 6) AS dt64
SELECT toUnixTimestamp64Milli(dt64)
```

结果:

```
toUnixTimestamp64Milli(dt64)─
1568650812345 |
```

```
WITH toDateTime64('2019-09-16 19:20:12.345678910', 6) AS dt64
SELECT toUnixTimestamp64Nano(dt64)
```

结果:

```
toUnixTimestamp64Nano(dt64)─
1568650812345678000 |
```

fromUnixTimestamp64Milli

fromUnixTimestamp64Micro

fromUnixTimestamp64Nano

把Int64类型的数据转换为DateTime64类型的数据，结果包含固定的亚秒精度和可选的时区。输入的值是变大还是变低依赖于输入的精度。需要注意的是输入的值是一个UTC的时间戳，不是一个包含时区的时间戳。

语法

```
fromUnixTimestamp64Milli(value [, ti])
```

参数

- `value` — Int64类型的数据，可以是任意精度。
- `timezone` — String类型的时区

返回值

- `value` DateTime64`类型的数据。

例子

```
WITH CAST(1234567891011, 'Int64') AS i64
SELECT fromUnixTimestamp64Milli(i64, 'UTC')
```

```
└─fromUnixTimestamp64Milli(i64, 'UTC')─  
  2009-02-13 23:31:31.011 |
```

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](#)

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.

Arguments are **maps** or two **arrays**, where the first array represent keys, and the second array contains values for the each key.

For array arguments the number of elements in **keys** and **values** must be the same for each row.

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.

Arguments

Mapped arrays:

- **keys** — Array of keys. [Array\(Int\)](#).
- **values** — Array of values. [Array\(Int\)](#).

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  
{[1,2,3,4,5],[11,22,0,44,0]} | type  
| 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']
```

IN运算符相关函数

in,notIn,globalIn,globalNotIn

请参阅 [IN 运算符](#) 部分。

tuple(x, y, ...), 运算符 (x, y, ...)

函数用于对多个列进行分组。

对于具有类型T1, T2, ...的列，它返回包含这些列的元组 (T1, T2, ...)。执行该函数没有任何成本。元组通常用作IN运算符的中间参数值，或用于创建lambda函数的形参列表。元组不能写入表。

tupleElement(tuple, n), 运算符 x.N

用于从元组中获取列的函数

'N'是列索引，从1开始。N必须是正整数常量，并且不大于元组的大小。

执行该函数没有任何成本。

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 $2+4\cdot n$, 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.
- $[(a, b), (c, d) \dots]$ — 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](#).

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 Engineering 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  
89184926cdbfffff |
```

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 SELECT h3GetFaces(599686042433355775) as faces;
```

Result:

```
faces  
[7] |
```

内省功能

您可以使用本章中描述的函数来反省 [ELF](#) 和 [DWARF](#) 用于查询分析。

警告

这些功能很慢，可能会强加安全考虑。

对于内省功能的正确操作：

- 安装 `clickhouse-common-static-dbg` 包。
- 设置 `allow_introspection_functions` 设置为 1。

出于安全考虑，内省函数默认是关闭的。

ClickHouse将探查器报告保存到 [trace_log](#) 系统表。确保正确配置了表和探查器。

addressToLine

将ClickHouse服务器进程内的虚拟内存地址转换为ClickHouse源代码中的文件名和行号。

如果您使用官方的ClickHouse软件包，您需要安装 `clickhouse-common-static-dbg` 包。

语法

```
addressToLine(address_of_binary_instruction)
```

参数

- `address_of_binary_instruction` ([UInt64](#)) — 正在运行进程的指令地址。

返回值

- 源代码文件名和行号（用冒号分隔的行号）

```
示例, `/build/obj-x86_64-linux-gnu/../src/Common/ThreadPool.cpp:199`, where `199` is a line number.
```

- 如果函数找不到调试信息，返回二进制文件的名称。
- 如果地址无效，返回空字符串。

类型: **字符串**.

示例

启用内省功能:

```
SET allow_introspection_functions=1
```

从中选择第一个字符串 `trace_log` 系统表:

```
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]
```

该 `trace` 字段包含采样时的堆栈跟踪。

获取单个地址的源代码文件名和行号:

```
SELECT addressToLine(94784076370703) \G
```

Row 1:

```
addressToLine(94784076370703): /build/obj-x86_64-linux-gnu/../src/Common/ThreadPool.cpp:199
```

将函数应用于整个堆栈跟踪:

```
SELECT
    arrayStringConcat(arrayMap(x -> addressToLine(x), trace), '\n') AS trace_source_code_lines
FROM system.trace_log
LIMIT 1
\G
```

该 `arrayMap` 功能允许处理的每个单独的元素 `trace` 阵列由 `addressToLine` 功能。这种处理的结果，你在看 `trace_source_code_lines` 列的输出。

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

将ClickHouse服务器进程内的虚拟内存地址转换为ClickHouse对象文件中的符号。

语法

```
addressToSymbol(address_of_binary_instruction)
```

参数

- `address_of_binary_instruction` (`UInt64`) — Address of instruction in a running process.

返回值

- 来自ClickHouse对象文件的符号。
- 如果地址无效，返回空字符串。

类型: `字符串`.

示例

启用内省功能:

```
SET allow_introspection_functions=1
```

从中选择第一个字符串 `trace_log` 系统表:

```
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:
[94138803686098,94138815010911,94138815096522,94138815101224,94138815102091,94138814222988,94138806823642,94138814457211,94138806823642,94138814457211,94138806823642,94138806795179,94138806796144,94138753770094,94138753771646,94138753760572,94138852407232,140399185266395,140399178045583]
```

该 `trace` 字段包含采样时的堆栈跟踪。

获取单个地址的符号:

```
SELECT addressToSymbol(94138803686098) \G
```

Row 1:

```
addressToSymbol(94138803686098):
_ZNK2DB24IAggregateFunctionHelperINS_20AggregateFunctionSumImmNS_24AggregateFunctionSumDataImEEEEEE1
9addBatchSinglePlaceEmPcPPKNS_7IColumnEPNS_5ArenaE
```

将函数应用于整个堆栈跟踪:

```
SELECT
    arrayStringConcat(arrayMap(x -> addressToSymbol(x), trace), '\n') AS trace_symbols
FROM system.trace_log
LIMIT 1
\G
```

该 `arrayMap` 功能允许处理的每个单独的元素 `trace` 阵列由 `addressToSymbols` 功能。这种处理的结果，你在看 `trace_symbols` 列的输出。

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_5BlockERNNS_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_iteratorISO_E
_ZZN20ThreadFromGlobalPoolC4IZN14ThreadPoolImplIS_E12scheduleImplvvEET_St8functionIFvvEEiSt8optionalImEEU
vE1_JEEEOS4_DpOT0_ENKUivE_c1Ev
_ZN14ThreadPoolImplISt6threadE6workerEST14_List_iteratorISO_E
execute_native_thread_routine
start_thread
clone
```

demangle

转换一个符号，您可以使用 `addressToSymbol` 函数到 C++ 函数名。

语法

```
demangle(symbol)
```

参数

- `symbol` (`字符串`) — Symbol from an object file.

返回值

- C++ 函数的名称。
- 如果符号无效，则为空字符串。

类型: `字符串`.

示例

启用内省功能:

```
SET allow_introspection_functions=1
```

从中选择第一个字符串 `trace_log` 系统表:

```
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:
[94138803686098,94138815010911,94138815096522,94138815101224,94138815102091,94138814222988,94138806823642,94138814457211,94138806823642,94138806795179,94138806796144,94138753770094,94138753771646,94138753760572,94138852407232,140399185266395,140399178045583]
```

该 `trace` 字段包含采样时的堆栈跟踪。

获取单个地址的函数名称:

```
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
```

将函数应用于整个堆栈跟踪:

```
SELECT
    arrayStringConcat(arrayMap(x -> demangle(addressToSymbol(x)), trace), '\n') AS trace_functions
FROM system.trace_log
LIMIT 1
\G
```

该 `arrayMap` 功能允许处理的每个单独的元素 `trace` 阵列由 `demangle` 功能。这种处理的结果，你在看 `trace_functions` 列的输出。

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*> >>&, 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_handler<void (), DB::AsynchronousBlockInputStream::next()>::_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

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>
```

GEO 函数

大圆形距离

使用 **great-circle distance** 公式计算地球表面两点之间的距离。

```
greatCircleDistance(lon1Deg, lat1Deg, lon2Deg, lat2Deg)
```

输入参数

- **lon1Deg** — 第一个点的经度，单位：度，范围： [-180°, 180°]。
- **lat1Deg** — 第一个点的纬度，单位：度，范围： [-90°, 90°]。
- **lon2Deg** — 第二个点的经度，单位：度，范围： [-180°, 180°]。
- **lat2Deg** — 第二个点的纬度，单位：度，范围： [-90°, 90°]。

正值对应北纬和东经，负值对应南纬和西经。

返回值

地球表面的两点之间的距离，以米为单位。

当输入参数值超出规定的范围时将抛出异常。

示例

```
SELECT greatCircleDistance(55.755831, 37.617673, -55.755831, -37.617673)
```

```
greatCircleDistance(55.755831, 37.617673, -55.755831, -37.617673) —  
14132374.194975413 |
```

尖尖的人

检查指定的点是否至少包含在指定的一个椭圆中。
下述中的坐标是几何图形在笛卡尔坐标系中的位置。

```
pointInEllipses(x, y, x0, y0, a0, b0,...,xn, yn, an, bn)
```

输入参数

- **x, y** — 平面上某个点的坐标。

■ x_i, y_i — 第 i 个椭圆的中心坐标。

■ a_i, b_i — 以 x, y 坐标为单位的第 i 个椭圆的轴。

输入参数的个数必须是 $2+4\cdot n$ ，其中 n 是椭圆的数量。

返回值

如果该点至少包含在一个椭圆中，则返回1；否则，则返回0。

示例

```
SELECT pointInEllipses(55.755831, 37.617673, 55.755831, 37.617673, 1.0, 2.0)
```

```
pointInEllipses(55.755831, 37.617673, 55.755831, 37.617673, 1., 2.)  
1 |
```

pointInPolygon

检查指定的点是否包含在指定的多边形中。

```
pointInPolygon((x, y), [(a, b), (c, d) ...], ...)
```

输入参数

■ (x, y) — 平面上某个点的坐标。元组类型，包含坐标的两个数字。

■ $[(a, b), (c, d) \dots]$ — 多边形的顶点。阵列类型。每个顶点由一对坐标 (a, b) 表示。顶点可以按顺时针或逆时针指定。顶点的个数应该大于等于3。同时只能是常量的。

■ 该函数还支持镂空的多边形（切除部分）。如果需要，可以使用函数的其他参数定义需要切除部分的多边形。(The function does not support non-simply-connected polygons.)

返回值

如果坐标点存在在多边形范围内，则返回1。否则返回0。

如果坐标位于多边形的边界上，则该函数可能返回1，或可能返回0。

示例

```
SELECT pointInPolygon((3., 3.), [(6, 0), (8, 4), (5, 8), (0, 2)]) AS res
```

```
res  
1 |
```

geohashEncode

将经度和纬度编码为geohash-string，请参阅 (<http://geohash.org/>,<https://en.wikipedia.org/wiki/Geohash>)。

```
geohashEncode(longitude, latitude, [precision])
```

输入值

■ **longitude** - 要编码的坐标的经度部分。其值应在 $[-180^\circ, 180^\circ]$ 范围内

- `latitude` - 要编码的坐标的纬度部分。其值应在 $[-90^\circ, 90^\circ]$ 范围内
- `precision` - 可选，生成的geohash-string的长度，默认为`12`。取值范围为`[1,12]`。任何小于`1`或大于`12`的值都会默认转换为`12`。

返回值

- 坐标编码的字符串（使用base32编码的修改版本）。

示例

```
SELECT geohashEncode(-5.60302734375, 42.593994140625, 0) AS res
```

```
res  
ezs42d000000 |
```

geohashDecode

将任何geohash编码的字符串解码为经度和纬度。

输入值

- `encoded string` - geohash编码的字符串。

返回值

- `(longitude, latitude)` - 经度和纬度的`Float64`值的2元组。

示例

```
SELECT geohashDecode('ezs42') AS res
```

```
res  
(-5.60302734375,42.60498046875) |
```

geoToH3

计算指定的分辨率的H3索引(`lon, lat`)。

```
geoToH3(lon, lat, resolution)
```

输入值

- `lon` — 经度。`Float64`类型。
- `lat` — 纬度。`Float64`类型。
- `resolution` — 索引的分辨率。取值范围为：`[0, 15]`。`UInt8`类型。

返回值

- H3中六边形的索引值。

- 发生异常时返回`0`。

`UInt64`类型。

示例

```
SELECT geoToH3(37.79506683, 55.71290588, 15) as h3Index
```

```
h3Index  
644325524701193974
```

geohashesInBox

计算在指定精度下计算最小包含指定的经纬范围的最小图形的geohash数组。

输入值

- `longitude_min` - 最小经度。其值应在 $[-180^\circ, 180^\circ]$ 范围内
- `latitude_min` - 最小纬度。其值应在 $[-90^\circ, 90^\circ]$ 范围内
- `longitude_max` - 最大经度。其值应在 $[-180^\circ, 180^\circ]$ 范围内
- `latitude_max` - 最大纬度。其值应在 $[-90^\circ, 90^\circ]$ 范围内
- `precision` - geohash的精度。其值应在`[1, 12]`内的`UInt8`类型的数字

请注意，上述所有的坐标参数必须同为`Float32`或`Float64`中的一种类型。

返回值

- 包含指定范围内的指定精度的geohash字符串数组。注意，您不应该依赖返回数组中geohash的顺序。
- `[]` - 当传入的最小经纬度大于最大经纬度时将返回一个空数组。

请注意，如果生成的数组长度超过10000时，则函数将抛出异常。

示例

```
SELECT geohashesInBox(24.48, 40.56, 24.785, 40.81, 4) AS thasos
```

```
thasos  
['sx1q','sx1r','sx32','sx1w','sx1x','sx38'] |
```

Hash函数

Hash函数可以用于将元素不可逆的伪随机打乱。

halfMD5

计算字符串的MD5。然后获取结果的前8个字节并将它们作为`UInt64`（大端）返回。

此函数相当低效（500万个短字符串/秒/核心）。

如果您不需要一定使用MD5，请使用'sipHash64'函数。

MD5

计算字符串的MD5并将结果放入`FixedString(16)`中返回。

如果您只是需要一个128位的hash，同时不需要一定使用MD5，请使用`'sipHash128'`函数。

如果您要获得与`md5sum`程序相同的输出结果，请使用`lower(hex(MD5(s)))`。

sipHash64

计算字符串的SipHash。

接受`String`类型的参数，返回`UInt64`。

SipHash是一种加密哈希函数。它的处理性能至少比MD5快三倍。

有关详细信息，请参阅链接：<https://131002.net/siphash/>

sipHash128

计算字符串的SipHash。

接受`String`类型的参数，返回`FixedString(16)`。

与`sipHash64`函数的不同在于它的最终计算结果为128位。

cityHash64

计算任意数量字符串的CityHash64或使用特定实现的Hash函数计算任意数量其他类型的Hash。

对于字符串，使用CityHash算法。这是一个快速的非加密哈希函数，用于字符串。

对于其他类型的参数，使用特定实现的Hash函数，这是一种快速的非加密的散列函数。

如果传递了多个参数，则使用CityHash组合这些参数的Hash结果。

例如，您可以计算整个表的checksum，其结果取决于行的顺序：`SELECT sum(cityHash64(*)) FROM table`。

intHash32

为任何类型的整数计算32位的哈希。

这是相对高效的非加密Hash函数。

intHash64

从任何类型的整数计算64位哈希码。

它的工作速度比`intHash32`函数快。

SHA1

SHA224

SHA256

计算字符串的SHA-1，SHA-224或SHA-256，并将结果字节集返回为`FixedString(20)`，`FixedString(28)`或`FixedString(32)`。

该函数相当低效（SHA-1大约500万个短字符串/秒/核心，而SHA-224和SHA-256大约220万个短字符串/秒/核心）。

我们建议仅在必须使用这些Hash函数且无法更改的情况下使用这些函数。

即使在这些情况下，我们仍建议将函数采用在写入数据时使用预算算的方式将其计算完毕。而不是在`SELECT`中计算它们。

URLHash(url[,N])

一种快速的非加密哈希函数，用于规范化的从URL获得的字符串。

`URLHash(s)` - 从一个字符串计算一个哈希，如果结尾存在尾随符号/，?或#则忽略。

`URLHash (s, N)` - 计算URL层次结构中字符串到N级别的哈希值，如果末尾存在尾随符号/，?或#则忽略。

URL的层级与URLHierarchy中的层级相同。此函数被用于Yandex.Metrica。

farmHash64

计算字符串的FarmHash64。

接受一个String类型的参数。返回UInt64。

有关详细信息，请参阅链接：[FarmHash64](#)

javaHash

计算字符串的JavaHash。

接受一个String类型的参数。返回Int32。

有关更多信息，请参阅链接：[JavaHash](#)

hiveHash

计算字符串的HiveHash。

接受一个String类型的参数。返回Int32。

与[JavaHash](#)相同，但不会返回负数。

metroHash64

计算字符串的MetroHash。

接受一个String类型的参数。返回UInt64。

有关详细信息，请参阅链接：[MetroHash64](#)

jumpConsistentHash

计算UInt64的JumpConsistentHash。

接受UInt64类型的参数。返回Int32。

有关更多信息，请参见链接：[JumpConsistentHash](#)

murmurHash2_32,murmurHash2_64

计算字符串的MurmurHash2。

接受一个String类型的参数。返回UInt64或UInt32。

有关更多信息，请参阅链接：[MurmurHash2](#)

murmurHash3_32,murmurHash3_64,murmurHash3_128

计算字符串的MurmurHash3。

接受一个String类型的参数。返回UInt64或UInt32或FixedString(16)。

有关更多信息，请参阅链接：[MurmurHash3](#)

xxHash32,xxHash64

计算字符串的xxHash。

接受一个String类型的参数。返回UInt64或UInt32。

有关更多信息，请参见链接：[xxHash](#)

IP 函数

IPv4NumToString(num)

接受一个UInt32（大端）表示的IPv4的地址，返回相应IPv4的字符串表现形式，格式为A.B.C.D（以点分割的十进制数字）。

IPv4StringToNum(s)

与IPv4NumToString函数相反。如果IPv4地址格式无效，则返回0。

IPv4NumToStringClassC(num)

与IPv4NumToString类似，但使用xxx替换最后一个字节。

示例：

```
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

由于使用'xxx'是不规范的，因此将来可能会更改。我们建议您不要依赖此格式。

IPv6NumToString(x)

接受FixedString(16)类型的二进制格式的IPv6地址。以文本格式返回此地址的字符串。

IPv6映射的IPv4地址以::ffff:111.222.33。例如：

```
SELECT IPv6NumToString(toFixedString(unhex('2A0206B800000000000000000000000011'), 16)) AS addr
```

```
addr
2a02:6b8::11 |
```

```
SELECT
    IPv6NumToString(ClientIP6) AS k,
    count() AS c
FROM hits_all
WHERE EventDate = today() AND substring(ClientIP6, 1, 12) != unhex('0000000000000000FFFF')
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:eeee	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)

与IPv6NumToString的相反。如果IPv6地址格式无效，则返回空字节字符串。
十六进制可以是大写的或小写的。

IPv4ToIntPv6(x)

接受一个UInt32类型的IPv4地址，返回FixedString(16)类型的IPv6地址。例如：

```
SELECT IPv6NumToString(IPv4ToIntPv6(IPv4StringToNum('192.168.0.1'))) AS addr
```

addr
::ffff:192.168.0.1

cutIPv6(x,bitsToCutForIPv6,bitsToCutForIPv4)

接受一个FixedString(16)类型的IPv6地址，返回一个String，这个String中包含了删除指定位之后的地址的文本格式。例如：

```

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:cafe:0	::ffff:192.168.0.0

ツ古カツ益ツ催ツ団ツ法ツ人),

接受一个IPv4地址以及一个UInt8类型的CIDR。返回包含子网最低范围以及最高范围的元组。

```
SELECT IPv4CIDRToRange(toIPv4('192.168.5.2'), 16)
```

```
└─IPv4CIDRToRange(toIPv4('192.168.5.2'), 16)─  
  ('192.168.0.0','192.168.255.255')
```

ツ暗エツ汎環催ツ団ツ法ツ人),

接受一个**IPv6**地址以及一个UInt8类型的CIDR。返回包含子网最低范围以及最高范围的元组。

```
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(字符串)

IPv4StringToNum()的别名，它采用字符串形式的**IPv4**地址并返回**IPv4**类型的值，该二进制值等于**IPv4StringToNum()**返回的值。

```
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(字符串)

IPv6StringToNum()的别名，它采用字符串形式的**IPv6**地址并返回**IPv6**类型的值，该二进制值等于**IPv6StringToNum()**返回的值。

```
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 函数

在 Yandex.Metrica 中，用户使用 JSON 作为访问参数。为了处理这些 JSON，实现了一些函数。（尽管在大多数情况下，JSON 是预先进行额外处理的，并将结果值放在单独的列中。）所有的这些函数都进行了尽可能的假设。以使函数能够尽快的完成工作。

我们对 JSON 格式做了如下假设：

1. 字段名称（函数的参数）必须使常量。
2. 字段名称必须使用规范的编码。例如：`visitParamHas('{"abc":"def"}', 'abc') = 1`，但是`visitParamHas('"\u0061\u0062\u0063":"def"', 'abc') = 0`
3. 函数可以随意的在多层嵌套结构下查找字段。如果存在多个匹配字段，则返回第一个匹配字段。
4. JSON 除字符串文本外不存在空格字符。

visitParamHas(参数, 名称)

检查是否存在 «name» 名称的字段

visitParamExtractUInt(参数, 名称)

将名为 «name» 的字段的值解析成 UInt64。如果这是一个字符串字段，函数将尝试从字符串的开头解析一个数字。如果该字段不存在，或无法从它中解析到数字，则返回 0。

visitParamExtractInt(参数, 名称)

与 visitParamExtractUInt 相同，但返回 Int64。

visitParamExtractFloat(参数, 名称)

与 visitParamExtractUInt 相同，但返回 Float64。

visitParamExtractBool(参数, 名称)

解析 true/false 值。其结果是 UInt8 类型的。

visitParamExtractRaw(参数, 名称)

返回字段的值，包含空格符。

示例：

```
visitParamExtractRaw('{"abc":"\\n\\u0000"}', 'abc') = "\\n\\u0000"
visitParamExtractRaw('{"abc":{"def":[1,2,3]}}', 'abc') = '{"def":[1,2,3]}'
```

visitParamExtractString(参数, 名称)

使用双引号解析字符串。这个值没有进行转义。如果转义失败，它将返回一个空白字符串。

示例：

```
visitParamExtractString('{"abc":"\\n\\u0000"}', 'abc') = '\\n\\0'
visitParamExtractString('{"abc":"\\u263a"}', 'abc') = '@'
visitParamExtractString('{"abc":"\\u263"}', 'abc') = "
visitParamExtractString('{"abc":"hello"}', 'abc') = "
```

目前不支持\uXXXX\uYYYY这些字符编码，这些编码不在基本多文种平面中（它们被转化为CESU-8而不是UTF-8）。

以下函数基于[simdjson](#)，专为更复杂的JSON解析要求而设计。但上述假设2仍然适用。

JSONHas(json[, indices_or_keys]...)

如果JSON中存在该值，则返回1。

如果该值不存在，则返回0。

示例：

```
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可以是零个或多个参数的列表，每个参数可以是字符串或整数。

- String = 按成员名称访问JSON对象成员。
- 正整数 = 从头开始访问第n个成员/成员名称。
- 负整数 = 从末尾访问第n个成员/成员名称。

您可以使用整数来访问JSON数组和JSON对象。

例如：

```
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]...)

返回JSON数组或JSON对象的长度。

如果该值不存在或类型错误，将返回0。

示例：

```
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]...)

返回JSON值的类型。

如果该值不存在，将返回Null。

示例：

```
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]...)

解析JSON并提取值。这些函数类似于visitParam*函数。

如果该值不存在或类型错误，将返回0。

示例：

```
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]...)

解析JSON并提取字符串。此函数类似于visitParamExtractString函数。

如果该值不存在或类型错误，则返回空字符串。

该值未转义。如果unescaping失败，则返回一个空字符串。

示例：

```
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)

解析JSON并提取给定ClickHouse数据类型的值。

这是以前的JSONExtract<type>函数的变体。这意味着JSONExtract(..., 'String')返回与JSONExtractString()返回完全相同。JSONExtract(..., 'Float64')返回与JSONExtractFloat()返回完全相同。

示例：

```
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)

从JSON中解析键值对，其中值是给定的ClickHouse数据类型。

示例：

```
SELECT JSONExtractKeysAndValues('{"x": {"a": 5, "b": 7, "c": 11}}', 'x', 'Int8') = [(('a',5),('b',7),('c',11))];
```

JSONExtractRaw(json[, indices_or_keys]...)

返回JSON的部分。

如果部件不存在或类型错误，将返回空字符串。

示例：

```
select JSONExtractRaw('{"a": "hello", "b": [-100, 200.0, 300]}', 'b') = '[-100, 200.0, 300]'
```

Nullable处理函数

isNull

检查参数是否为NULL。

```
isNull(x)
```

参数

■ **x** — 一个非复合数据类型的值。

返回值

■ 1 如果**x**为NULL。

■ 0 如果**x**不为NULL。

示例

存在以下内容的表

x	y
1	NULL
2	3

对其进行查询

```
:) SELECT x FROM t_null WHEREisNull(y)
```

```
SELECT x  
FROM t_null  
WHERE isNull(y)
```

x
1

1 rows in set. Elapsed: 0.010 sec.

isNotNull

检查参数是否不为 **NULL**。

```
isNotNull(x)
```

参数：

- **x** — 一个非复合数据类型的值。

返回值

- 0 如果**x**为 **NULL**。
- 1 如果**x**不为 **NULL**。

示例

存在以下内容的表

x	y
1	NULL
2	3

对其进行查询

```
:) SELECT x FROM t_null WHERE isNotNull(y)
```

```
SELECT x  
FROM t_null  
WHERE isNotNull(y)
```

x
2

1 rows in set. Elapsed: 0.010 sec.

合并

检查从左到右是否传递了«**NULL**»参数并返回第一个非'**NULL**'参数。

```
coalesce(x,...)
```

参数：

- 任何数量的非复合类型的参数。所有参数必须与数据类型兼容。

返回值

- 第一个非'NULL`参数。
- **NULL**，如果所有参数都是'NULL`。

示例

考虑可以指定多种联系客户的方式的联系人列表。

name	mail	phone	icq
client 1	NULL	123-45-67	123
client 2	NULL	NULL	NULL

mail和phone字段是String类型，但icq字段是UInt32，所以它需要转换为String。

从联系人列表中获取客户的第一个可用联系方式：

```
:) SELECT coalesce(mail, phone, CAST(icq,'Nullable(String)')) FROM aBook
```

```
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

```
2 rows in set. Elapsed: 0.006 sec.
```

ifNull

如果第一个参数为«NULL»，则返回第二个参数的值。

```
ifNull(x,alt)
```

参数:

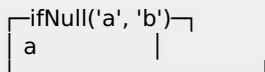
- **x** — 要检查«NULL»的值。
- **alt** — 如果x为'NULL`，函数返回的值。

返回值

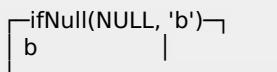
- 价值 **x**，如果 **x** 不是 **NULL**.
- 价值 **alt**，如果 **x** 是 **NULL**.

示例

```
SELECT ifNull('a', 'b')
```



```
SELECT ifNull(NULL, 'b')
```



nullIf

如果参数相等，则返回NULL。

```
nullIf(x, y)
```

参数：

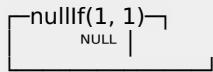
x, y — 用于比较的值。它们必须是类型兼容的，否则将抛出异常。

返回值

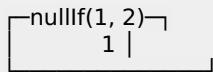
- 如果参数相等，则为NULL。
- 如果参数不相等，则为x值。

示例

```
SELECT nullIf(1, 1)
```



```
SELECT nullIf(1, 2)
```



assumeNotNull

将可为空类型的值转换为非Nullable类型的值。

```
assumeNotNull(x)
```

参数：

- x — 原始值。

返回值

- 如果x不为NULL，返回非Nullable类型的原始值。
- 如果x为NULL，返回对应非Nullable类型的默认值。

示例

存在如下t_null表。

```
SHOW CREATE TABLE t_null
```

statement

```
CREATE TABLE default.t_null ( x Int8, y Nullable(Int8) ENGINE = TinyLog |
```

x	y
1	NULL
2	3

将列y作为assumeNotNull函数的参数。

```
SELECT assumeNotNull(y) FROM t_null
```

```
assumeNotNull(y)|  
0 |  
3 |
```

```
SELECT toTypeName(assumeNotNull(y)) FROM t_null
```

```
toTypeName(assumeNotNull(y))|  
Int8 |  
Int8 |
```

可调整

将参数的类型转换为Nullable。

```
toNullable(x)
```

参数：

- x — 任何非复合类型的值。

返回值

- 输入的值，但其类型为Nullable。

示例

```
SELECT toTypeName(10)
```

```
toTypeName(10)|  
UInt8 |
```

```
SELECT toTypeName(toNullable(10))
```

```
toTypeName(toNullable(10))|  
Nullable(UInt8) |
```

URL函数

所有这些功能都不遵循RFC。它们被最大程度简化以提高性能。

URL截取函数

如果URL中没有要截取的内容则返回空字符串。

协议

返回URL的协议。例如：http、ftp、mailto、magnet...。

域

获取域名。

domainwithoutww

返回域名并删除第一个'www.'。

topLevelDomain

返回顶级域名。例如：.ru。

第一重要的元素分区域

返回«第一个有效子域名»。这并不是一个标准概念，仅用于Yandex.Metrica。如果顶级域名为'com'，'net'，'org'或者'co'则第一个有效子域名为二级域名。否则则返回三级域名。例如，`irstSignificantSubdomain('https://news.yandex.ru/')` = 'yandex'，`irstSignificantSubdomain ('https://news.yandex.com.tr/')` = 'yandex'。一些实现细节在未来可能会进行改变。

cutToFirstSignificantSubdomain

返回包含顶级域名与第一个有效子域名之间的内容（请参阅上面的内容）。

例如，`cutToFirstSignificantSubdomain('https://news.yandex.com.tr/')` = 'yandex.com.tr'.

路径

返回URL路径。例如：`/top/news.html`，不包含请求参数。

pathFull

与上面相同，但包括请求参数和fragment。例如：`/top/news.html?page=2#comments`

查询字符串

返回请求参数。例如：`page=1&lr=213`。请求参数不包含问号已经# 以及# 之后所有的内容。

片段

返回URL的fragment标识。fragment不包含#。

querystring andfragment

返回请求参数和fragment标识。例如：`page=1#29390`。

extractURLParameter(URL,name)

返回URL请求参数中名称为'name'的参数。如果不存在则返回一个空字符串。如果存在多个匹配项则返回第一个相匹配的。此函数假设参数名称与参数值在url中的编码方式相同。

extractURLParameters(URL)

返回一个数组，其中以`name=value`的字符串形式返回url的所有请求参数。不以任何编码解析任何内容。

extractURLParameterNames(URL)

返回一个数组，其中包含url的所有请求参数的名称。不以任何编码解析任何内容。

URLHierarchy(URL)

返回一个数组，其中包含以/切割的URL的所有内容。?将被包含在URL路径以及请求参数中。连续的分割符号被记为一个。

Urlpathhierarchy(URL)

与上面相同，但结果不包含协议和host部分。/element(root)不包括在内。该函数用于在Yandex.Metric中实现导出URL的树形结构。

```
URLPathHierarchy('https://example.com/browse/CONV-6788') =  
[  
  '/browse/',  
  '/browse/CONV-6788'  
]
```

decodeURLComponent(URL)

返回已经解码的URL。

例如：

```
SELECT decodeURLComponent('http://127.0.0.1:8123/?query=SELECT%201%3B') AS DecodedURL;
```

DecodedURL
http://127.0.0.1:8123/?query=SELECT 1; |

删除URL中的部分内容

如果URL中不包含指定的部分，则URL不变。

cutWWW

删除开始的第一个'www.'。

cutQueryString

删除请求参数。问号也将被删除。

cutFragment

删除fragment标识。#同样也会被删除。

cutquerystring andfragment

删除请求参数以及fragment标识。问号以及#也会被删除。

cutURLParameter(URL,name)

删除URL中名称为'name'的参数。改函数假设参数名称以及参数值经过URL相同的编码。

UUID函数

下面列出了所有UUID的相关函数

generateuidv4

生成一个UUID（[版本4](#)）。

```
generateUUIDv4()
```

返回值

UUID类型的值。

使用示例

此示例演示如何在表中创建UUID类型的列，并对其写入数据。

```
:) CREATE TABLE t_uuid (x UUID) ENGINE=TinyLog  
:) INSERT INTO t_uuid SELECT generateUUIDv4()  
:) SELECT * FROM t_uuid
```

```
f4bf890f-f9dc-4332-ad5c-0c18e73f28e9 |
```

toUUID(x)

将String类型的值转换为UUID类型的值。

```
toUUID(String)
```

返回值

UUID类型的值

使用示例

```
:) SELECT toUUID('61f0c404-5cb3-11e7-907b-a6006ad3dba0') AS uuid
```

```
61f0c404-5cb3-11e7-907b-a6006ad3dba0 |
```

UUIDStringToNum

接受一个String类型的值，其中包含36个字符且格式为xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx，将其转换为UUID的数值并以[固定字符串\(16\)](#)将其返回。

```
UUIDStringToNum(String)
```

返回值

[固定字符串\(16\)](#)

使用示例

```
:) SELECT
  '612f3c40-5d3b-217e-707b-6a546a3d7b29' AS uuid,
  UUIDStringToNum(uuid) AS bytes
```

uuid	bytes
612f3c40-5d3b-217e-707b-6a546a3d7b29	a/<@]:!~p{jTj={}

UUIDNumToString

接受一个**固定字符串(16)**类型的值，返回其对应的String表现形式。

```
UUIDNumToString(FixedString(16))
```

返回值

字符串。

使用示例

```
SELECT
  'a/<@]:!~p{jTj={} AS bytes,
  UUIDNumToString(toFixedString(bytes, 16)) AS uuid
```

bytes	uuid
a/<@]:!~p{jTj={}	612f3c40-5d3b-217e-707b-6a546a3d7b29

另请参阅

- [dictgetuid](#)

arrayJoin 函数

这是一个非常有用的函数。

普通函数不会更改结果集的行数，而只是计算每行中的值（map）。

聚合函数将多行压缩到一行中（fold或reduce）。

'arrayJoin'函数获取每一行并将他们展开到多行（unfold）。

此函数将数组作为参数，并将该行在结果集中复制数组元素个数。

除了应用此函数的列中的值之外，简单地复制列中的所有值；它被替换为相应的数组值。

查询可以使用多个arrayJoin函数。在这种情况下，转换被执行多次。

请注意SELECT查询中的ARRAY JOIN语法，它提供了更广泛的可能性。

示例：

```
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]

位图函数

位图函数用于对两个位图对象进行计算，对于任何一个位图函数，它都将返回一个位图对象，例如and，or，xor，not等。

位图对象有两种构造方法。一个是由聚合函数groupBitmapState构造的，另一个是由Array Object构造的。同时还可以将位图对象转化为数组对象。

我们使用RoaringBitmap实际存储位图对象，当基数小于或等于32时，它使用Set保存。当基数大于32时，它使用RoaringBitmap保存。这也是为什么低基数集的存储更快的原因。

有关RoaringBitmap的更多信息，请参阅：[RoaringBitmap](#)。

bitmapBuild

从无符号整数数组构建位图对象。

```
bitmapBuild(array)
```

参数

- **array** – 无符号整数数组.

示例

```
SELECT bitmapBuild([1, 2, 3, 4, 5]) AS res
```

bitmapToArray

将位图转换为整数数组。

```
bitmapToArray(bitmap)
```

参数

- **bitmap** – 位图对象.

示例

```
SELECT bitmapToArray(bitmapBuild([1, 2, 3, 4, 5])) AS res
```

```
res  
[1,2,3,4,5] |
```

bitmapSubsetInRange

将位图指定范围（不包含range_end）转换为另一个位图。

```
bitmapSubsetInRange(bitmap, range_start, range_end)
```

参数

- **bitmap** – 位图对象.

- `range_start` – 范围起始点（含）。
- `range_end` – 范围结束点（不含）。

示例

```
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

将位图指定范围（起始点和数目上限）转换为另一个位图。

```
bitmapSubsetLimit(bitmap, range_start, limit)
```

参数

- `bitmap` – 位图对象。
- `range_start` – 范围起始点（含）。
- `limit` – 子位图基数上限。

示例

```
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
```

```
res
[30,31,32,33,100,200,500] |
```

subBitmap

将位图跳过`offset`个元素，限制大小为`limit`个的结果转换为另一个位图。

```
subBitmap(bitmap, offset, limit)
```

参数

- `bitmap` – 位图对象。
- `offset` – 跳过多少个元素。
- `limit` – 子位图基数上限。

示例

```
SELECT
bitmapToArray(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(10), toUInt32(10)) AS res
```

```
res
[10,11,12,13,14,15,16,17,18,19] |
```

bitmapContains

检查位图是否包含指定元素。

```
bitmapContains(haystack, needle)
```

参数

- `haystack` – 位图对象.
- `needle` – 元素，类型 UInt32.

示例

```
SELECT bitmapContains(bitmapBuild([1,5,7,9]), toUInt32(9)) AS res
```

```
res
1 |
```

bitmapHasAny

与 `hasAny(array, array)` 类似，如果位图有任何公共元素则返回1，否则返回0。

对于空位图，返回0。

```
bitmapHasAny(bitmap,bitmap)
```

参数

- `bitmap` – bitmap 对象。

示例

```
SELECT bitmapHasAny(bitmapBuild([1,2,3]),bitmapBuild([3,4,5])) AS res
```

```
res
1 |
```

bitmapHasAll

与 `hasAll(array, array)` 类似，如果第一个位图包含第二个位图的所有元素，则返回1，否则返回0。

如果第二个参数是空位图，则返回1。

```
bitmapHasAll(bitmap,bitmap)
```

参数

- `bitmap` – `bitmap` 对象。

示例

```
SELECT bitmapHasAll(bitmapBuild([1,2,3]),bitmapBuild([3,4,5])) AS res
```

```
res  
0 |
```

位图和

为两个位图对象进行与操作，返回一个新的位图对象。

```
bitmapAnd(bitmap1,bitmap2)
```

参数

- `bitmap1` – 位图对象。
- `bitmap2` – 位图对象。

示例

```
SELECT bitmapToArray(bitmapAnd(bitmapBuild([1,2,3]),bitmapBuild([3,4,5]))) AS res
```

```
res  
[3] |
```

位图或

为两个位图对象进行或操作，返回一个新的位图对象。

```
bitmapOr(bitmap1,bitmap2)
```

参数

- `bitmap1` – 位图对象。
- `bitmap2` – 位图对象。

示例

```
SELECT bitmapToArray(bitmapOr(bitmapBuild([1,2,3]),bitmapBuild([3,4,5]))) AS res
```

```
res  
[1,2,3,4,5] |
```

bitmapXor

为两个位图对象进行异或操作，返回一个新的位图对象。

```
bitmapXor(bitmap1,bitmap2)
```

参数

- `bitmap1` – 位图对象。
- `bitmap2` – 位图对象。

示例

```
SELECT bitmapToArray(bitmapXor(bitmapBuild([1,2,3]),bitmapBuild([3,4,5]))) AS res
```

```
res  
[1,2,4,5] |
```

bitmapAndnot

计算两个位图的差异，返回一个新的位图对象。

```
bitmapAndnot(bitmap1,bitmap2)
```

参数

- `bitmap1` – 位图对象。
- `bitmap2` – 位图对象。

示例

```
SELECT bitmapToArray(bitmapAndnot(bitmapBuild([1,2,3]),bitmapBuild([3,4,5]))) AS res
```

```
res  
[1,2] |
```

bitmapCardinality

返回一个UInt64类型的数值，表示位图对象的基数。

```
bitmapCardinality(bitmap)
```

参数

- `bitmap` – 位图对象。

示例

```
SELECT bitmapCardinality(bitmapBuild([1, 2, 3, 4, 5])) AS res
```

```
res  
5 |
```

bitmapMin

返回一个UInt64类型的数值，表示位图中的最小值。如果位图为空则返回UINT32_MAX。

```
bitmapMin(bitmap)
```

参数

- **bitmap** – 位图对象。

示例

```
SELECT bitmapMin(bitmapBuild([1, 2, 3, 4, 5])) AS res
```

```
res  
1 |
```

bitmapMax

返回一个UInt64类型的数值，表示位图中的最大值。如果位图为空则返回0。

```
bitmapMax(bitmap)
```

参数

- **bitmap** – 位图对象。

示例

```
SELECT bitmapMax(bitmapBuild([1, 2, 3, 4, 5])) AS res
```

```
res  
5 |
```

位图和标准性

为两个位图对象进行与操作，返回结果位图的基数。

```
bitmapAndCardinality(bitmap1,bitmap2)
```

参数

- **bitmap1** – 位图对象。
- **bitmap2** – 位图对象。

示例

```
SELECT bitmapAndCardinality(bitmapBuild([1,2,3]),bitmapBuild([3,4,5])) AS res;
```

res
1

bitmapOrCardinality

为两个位图进行或运算，返回结果位图的基数。

```
bitmapOrCardinality(bitmap1,bitmap2)
```

参数

- **bitmap1** – 位图对象。
- **bitmap2** – 位图对象。

示例

```
SELECT bitmapOrCardinality(bitmapBuild([1,2,3]),bitmapBuild([3,4,5])) AS res;
```

res
5

bitmapXorCardinality

为两个位图进行异或运算，返回结果位图的基数。

```
bitmapXorCardinality(bitmap1,bitmap2)
```

参数

- **bitmap1** – 位图对象。
- **bitmap2** – 位图对象。

示例

```
SELECT bitmapXorCardinality(bitmapBuild([1,2,3]),bitmapBuild([3,4,5])) AS res;
```

res
4

位图和非标准性

计算两个位图的差异，返回结果位图的基数。

```
bitmapAndnotCardinality(bitmap1,bitmap2)
```

参数

- `bitmap1` - 位图对象。
- `bitmap2` - 位图对象。

示例

```
SELECT bitmapAndNotCardinality(bitmapBuild([1,2,3]),bitmapBuild([3,4,5])) AS res;
```

```
res  
2 |
```

位操作函数

位操作函数适用于 `UInt8`, `UInt16`, `UInt32`, `UInt64`, `Int8`, `Int16`, `Int32`, `Int64`, `Float32` 或 `Float64` 中的任何类型。

结果类型是一个整数，其位数等于其参数的最大位。如果至少有一个参数为有符数字，则结果为有符数字。如果参数是浮点数，则将其强制转换为 `Int64`。

`bitAnd(a,b)`

`bitOr(a,b)`

`bitXor(a,b)`

`bitNot(a)`

`bitShiftLeft(a,b)`

`bitShiftRight(a,b)`

`bitRotateLeft(a,b)`

`bitRotateRight(a,b)`

`bitTest(a,b)`

`bitTestAll(a,b)`

`bitTestAny(a,b)`

使用 Yandex.Metrica 字典函数

为了使下面的功能正常工作，服务器配置必须指定获取所有 Yandex.Metrica 字典的路径和地址。Yandex.Metrica 字典在任何这些函数的第一次调用时加载。如果无法加载引用列表，则会引发异常。

有关创建引用列表的信息，请参阅《字典》部分。

多个地理基

ClickHouse 支持同时使用多个备选地理基（区域层次结构），以支持某些地区所属国家的各种观点。

该 ‘clickhouse-server’ config 指定具有区域层次结构的文

件::<path_to_regions_hierarchy_file>/opt/geo/regions_hierarchy.txt</path_to_regions_hierarchy_file>

除了这个文件，它还搜索附近有_ 符号和任何后缀附加到名称（文件扩展名之前）的文件。

例如，它还会找到该文件 /opt/geo/regions_hierarchy_ua.txt，如果存在。

ua 被称为字典键。对于没有后缀的字典，键是空字符串。

所有字典都在运行时重新加载（每隔一定数量的秒重新加载一次，如builtin_dictionaries_reload_interval config 参数中定义，或默认情况下每小时一次）。但是，可用字典列表在服务器启动时定义一次。

所有处理区域的函数都在末尾有一个可选参数—字典键。它被称为地基。

示例：

```
regionToCountry(RegionID) - 使用默认路径: /opt/geo/regions_hierarchy.txt  
regionToCountry(RegionID, '') - 使用默认路径: /opt/geo/regions_hierarchy.txt  
regionToCountry(RegionID, 'ua') - 使用字典中的'ua' 键: /opt/geo/regions_hierarchy_ua.txt
```

regionToCity(id[, geobase])

从 Yandex geobase 接收一个 UInt32 数字类型的区域ID。如果该区域是一个城市或城市的一部分，它将返回相应城市的区域ID。否则，返回0。

regionToArea(id[, geobase])

将区域转换为区域（地理数据库中的类型5）。在所有其他方式，这个功能是一样的 ‘regionToCity’.

```
SELECT DISTINCT regionToName(regionToArea(toUInt32(number), 'ua'))  
FROM system.numbers  
LIMIT 15
```

```
regionToName(regionToArea(toUInt32(number), '\ua'))—  
|  
Moscow and Moscow region  
St. Petersburg and Leningrad region  
Belgorod region  
Ivanovsk region  
Kaluga region  
Kostroma region  
Kursk region  
Lipetsk region  
Orlov region  
Ryazan region  
Smolensk region  
Tambov region  
Tver region  
Tula region
```

regionToDistrict(id[,geobase])

将区域转换为联邦区（地理数据库中的类型4）。在所有其他方式，这个功能是一样的 ‘regionToCity’.

```
SELECT DISTINCT regionToName(regionToDistrict(toUInt32(number), 'ua'))  
FROM system.numbers  
LIMIT 15
```

```

regionToName(regionToDistrict(toUInt32(number), '\ua\'))|
  Central federal district
  Northwest federal district
  South federal district
  North Caucases federal district
  Privolga federal district
  Ural federal district
  Siberian federal district
  Far East federal district
  Scotland
  Faroe Islands
  Flemish region
  Brussels capital region
  Wallonia
  Federation of Bosnia and Herzegovina

```

regionToCountry(id[, geobase])

将区域转换为国家。在所有其他方式，这个功能是一样的‘regionToCity’。

示例: `regionToCountry(toUInt32(213)) = 225` 转换莫斯科 (213) 到俄罗斯 (225)。

regionToContinent(id[, geobase])

将区域转换为大陆。在所有其他方式，这个功能是一样的‘regionToCity’。

示例: `regionToContinent(toUInt32(213)) = 10001` 将莫斯科 (213) 转换为欧亚大陆 (10001)。

regionToTopContinent (#regiontotopcontinent)

查找该区域层次结构中最高的大陆。

语法

```
regionToTopContinent(id[, geobase])
```

参数

- `id` — Yandex geobase 的区域 ID. **UInt32**.
- `geobase` — 字典的建. 参阅 [Multiple Geobases](#). **String**, 可选.

返回值

- 顶级大陆的标识符(当您在区域层次结构中攀爬时，是后者)。
- 0，如果没有。

类型: **UInt32**.

regionToPopulation(id[, geobase])

获取区域的人口。

人口可以记录在文件与地球基。请参阅《外部词典》部分。

如果没有为该区域记录人口，则返回0。

在Yandex地理数据库中，可能会为子区域记录人口，但不会为父区域记录人口。

regionIn(lhs,rhs[, 地理数据库])

检查是否 ‘lhs’ 属于一个区域 ‘rhs’ 区域。如果属于UInt8，则返回等于1的数字，如果不属于则返回0。这种关系是反射的——任何地区也属于自己。

regionHierarchy(id[, geobase])

从 Yandex geobase 接收一个 UInt32 数字类型的区域ID。返回一个区域ID数组，由传递的区域和链上的所有父节点组成。

示例: `regionHierarchy(toUInt32(213)) = [213,1,3,225,10001,10000].`

regionToName(id[, lang])

从 Yandex geobase 接收一个 UInt32 数字类型的区域ID。带有语言名称的字符串可以作为第二个参数传递。支持的语言有:ru, en, ua, uk, by, kz, tr。如果省略第二个参数，则使用' ru '语言。如果不支持该语言，则抛出异常。返回一个字符串-对应语言的区域名称。如果指定ID的区域不存在，则返回一个空字符串。

`ua` 和 `uk` 都意味着乌克兰。

其他函数

主机名()

返回一个字符串，其中包含执行此函数的主机的名称。对于分布式处理，如果在远程服务器上执行此函数，则将返回远程服务器主机的名称。

basename

在最后一个斜杠或反斜杠后的字符串文本。此函数通常用于从路径中提取文件名。

`basename(expr)`

参数

- `expr` — 任何一个返回字符串结果的表达式。字符串

返回值

一个String类型的值，其包含：

- 在最后一个斜杠或反斜杠后的字符串文本内容。

如果输入的字符串以斜杆或反斜杆结尾，例如：`/` 或 `c:\`，函数将返回一个空字符串。

- 如果输入的字符串中不包含斜杆或反斜杠，函数返回输入字符串本身。

示例

`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)

以文本格式（以制表符分隔）向控制台输出值时，计算近似宽度。

系统使用此函数实现Pretty格式。

以文本格式（制表符分隔）将值输出到控制台时，计算近似宽度。

这个函数被系统用于实现漂亮的格式。

NULL 表示为对应于 NULL 在 Pretty 格式。

```
SELECT visibleWidth(NULL)
```

visibleWidth(NULL)	4
--------------------	---

toTypeName(x)

返回包含参数的类型名称的字符串。

如果将NULL作为参数传递给函数，那么它返回Nullable (Nothing) 类型，它对应于ClickHouse中的内部NULL。

块大小()

获取Block的大小。

在ClickHouse中，查询始终工作在Block（包含列的部分的集合）上。此函数允许您获取调用其的块的大小。

实现(x)

将一个常量列变为一个非常量列。

在ClickHouse中，非常量列和常量列在内存中的表示方式不同。尽管函数对于常量列和非常量总是返回相同的结果，但它们的工作方式可能完全不同（执行不同的代码）。此函数用于调试这种行为。

ignore(...)

接受任何参数，包括NULL。始终返回0。

但是，函数的参数总是被计算的。该函数可以用于基准测试。

睡眠 (秒)

在每个Block上休眠'seconds'秒。可以是整数或浮点数。

sleepEachRow (秒)

在每行上休眠'seconds'秒。可以是整数或浮点数。

当前数据库()

返回当前数据库的名称。

当您需要在CREATE TABLE中的表引擎参数中指定数据库，您可以使用此函数。

isFinite(x)

接受Float32或Float64类型的参数，如果参数不是infinite且不是NaN，则返回1，否则返回0。

isInfinite(x)

接受Float32或Float64类型的参数，如果参数是infinite，则返回1，否则返回0。注意NaN返回0。

isNaN(x)

接受Float32或Float64类型的参数，如果参数是Nan，则返回1，否则返回0。

hasColumnInTable(['hostname'][, 'username'][, 'password']][,] ‘database’, ‘table’, ‘column’)

接受常量字符串：数据库名称、表名称和列名称。如果存在列，则返回等于1的UInt8常量表达式，否则返回0。如果设置了hostname参数，则测试将在远程服务器上运行。

如果表不存在，该函数将引发异常。

对于嵌套数据结构中的元素，该函数检查是否存在列。对于嵌套数据结构本身，函数返回0。

酒吧

使用unicode构建图表。

bar(x, min, max, width) 当x = max时，绘制一个宽度与(x - min)成正比且等于width的字符带。

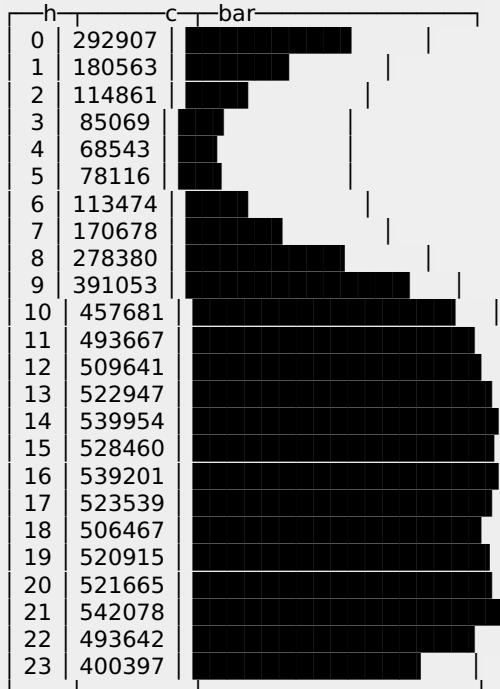
参数:

- **x** — 要显示的尺寸。
- **min, max** — 整数常量，该值必须是Int64。
- **width** — 常量，可以是正整数或小数。

字符带的绘制精度是符号的八分之一。

示例:

```
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

根据定义，将某些元素转换为其他元素。

此函数有两种使用方式：

1. `transform(x, array_from, array_to, default)`

`x` - 要转换的值。

`array_from` - 用于转换的常量数组。

`array_to` - 将'from'中的值转换为的常量数组。

`default` - 如果'x'不等于'from'中的任何值，则默认转换的值。

`array_from` 和 `array_to` - 拥有相同大小的数组。

类型约束:

`transform(T, Array(T), Array(U), U) -> U`

`T`和`U`可以是`String`，`Date`，`DateTime`或任意数值类型的。

对于相同的字母（`T`或`U`），如果数值类型，那么它们不可不完全匹配的，只需要具备共同的类型即可。

例如，第一个参数是`Int64`类型，第二个参数是`Array(UInt16)`类型。

如果'x'值等于'array_from'数组中的一个元素，它将从'array_to'数组返回一个对应的元素（下标相同）。否则，它返回'default'。如果'array_from'匹配到了多个元素，则返回第一个匹配的元素。

示例:

```

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

1. transform(x, array_from, array_to)

与第一种不同在于省略了'default'参数。

如果'x'值等于'array_from'数组中的一个元素，它将从'array_to'数组返回相应的元素（下标相同）。否则，它返回'x'。

类型约束：

`transform(T, Array(T), Array(T)) -> T`

示例：

```
SELECT
    transform(domain(Rferer), ['yandex.ru', 'google.ru', 'vk.com'], ['www.yandex', 'example.com']) AS s,
    count() AS c
FROM test.hits
GROUP BY domain(Rferer)
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)

接受大小（字节数）。返回带有后缀（KiB, MiB等）的字符串。

示例：

```
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

至少(a,b)

返回a和b中的最小值。

最伟大(a,b)

返回a和b的最大值。

碌莽碌time拢time()

返回服务正常运行的秒数。

版本()

以字符串形式返回服务器的版本。

时区()

返回服务器的时区。

blockNumber

返回行所在的Block的序列号。

rowNumberInBlock

返回行所在Block中行的序列号。针对不同的Block始终重新计算。

rowNumberInAllBlocks()

返回行所在结果集中的序列号。此函数仅考虑受影响的Block。

运行差异(x)

计算数据块中相邻行的值之间的差异。

对于第一行返回0，并为每个后续行返回与前一行的差异。

函数的结果取决于受影响的Block和Block中的数据顺序。

如果使用ORDER BY创建子查询并从子查询外部调用该函数，则可以获得预期结果。

示例：

```
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

运行差异启动与第一值

与[运行差异](#)相同，区别在于第一行返回第一行的值，后续每个后续行返回与上一行的差值。

MACNumToString(num)

接受一个 UInt64 类型的数字。 将其解释为 big endian 的 MAC 地址。 返回包含相应 MAC 地址的字符串，格式为 AA:BB:CC:DD:EE:FF（以冒号分隔的十六进制形式的数字）。

MACStringToNum(s)

与 MACNumToString 相反。 如果 MAC 地址格式无效，则返回 0。

MACStringToOUI(s)

接受格式为 AA:BB:CC:DD:EE:FF（十六进制形式的冒号分隔数字）的 MAC 地址。 返回前三个八位字节作为 UInt64 编号。 如果 MAC 地址格式无效，则返回 0。

getSizeOfEnumType

返回枚举中的枚举数量。

```
getSizeOfEnumType(value)
```

参数：

- value — Enum 类型的值。

返回值

- Enum 的枚举数量。
- 如果类型不是 Enum，则抛出异常。

示例

```
SELECT getSizeOfEnumType( CAST('a' AS Enum8('a' = 1, 'b' = 2) ) ) AS x
```

```
[x  
2 |
```

toColumnName

返回在 RAM 中列的数据类型的名称。

```
toColumnName(value)
```

参数：

- value — 任何类型的值。

返回值

- 一个字符串，其内容是 value 在 RAM 中的类型名称。

toTypeName 与 **toColumnName** 的区别示例

```
:) select toTypeName(cast('2018-01-01 01:02:03' AS DateTime))
SELECT toTypeName(CAST('2018-01-01 01:02:03', 'DateTime'))
  toTypeName(CAST('2018-01-01 01:02:03', 'DateTime'))—
    DateTime
```

1 rows in set. Elapsed: 0.008 sec.

```
:) select toColumnTypeNames(cast('2018-01-01 01:02:03' AS DateTime))
SELECT toColumnTypeNames(CAST('2018-01-01 01:02:03', 'DateTime'))
  toColumnTypeNames(CAST('2018-01-01 01:02:03', 'DateTime'))—
    Const(UInt32)
```

该示例显示 `DateTime` 数据类型作为 `Const(UInt32)` 存储在内存中。

dumpColumnStructure

输出在 RAM 中的数据结果的详细信息。

```
dumpColumnStructure(value)
```

参数:

- `value` — 任何类型的值。

返回值

- 一个字符串，其内容是 `value` 在 RAM 中的数据结构的详细描述。

示例

```
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

输出数据类型的默认值。

不包括用户设置的自定义列的默认值。

```
defaultValueOfArgumentType(expression)
```

参数:

- `expression` — 任意类型的值或导致任意类型值的表达式。

返回值

- 数值类型返回 0。
- 字符串类型返回空的字符串。
- 可为空类型返回 `NULL`。

示例

```
:) SELECT defaultValueOfArgumentType( CAST(1 AS Int8) )  
SELECT defaultValueOfArgumentType(CAST(1, 'Int8'))  
defaultValueOfArgumentType(CAST(1, 'Int8'))—  
0 |  
  
1 rows in set. Elapsed: 0.002 sec.  
  
:) SELECT defaultValueOfArgumentType( CAST(1 AS Nullable(Int8) ) )  
SELECT defaultValueOfArgumentType(CAST(1, 'Nullable(Int8') ))  
defaultValueOfArgumentType(CAST(1, 'Nullable(Int8') ))—  
NULL |  
  
1 rows in set. Elapsed: 0.002 sec.
```

indexHint

输出符合索引选择范围内的所有数据，同时不实用参数中的表达式进行过滤。

传递给函数的表达式参数将不会被计算，但ClickHouse使用参数中的表达式进行索引过滤。

返回值

- 1。

示例

这是一个包含ontime测试数据集的测试表。

```
SELECT count() FROM ontime
```

```
count()  
4276457 |
```

该表使用(FlightDate, (Year, FlightDate))作为索引。

对该表进行如下的查询：

```
:) SELECT FlightDate AS k, count() FROM ontime GROUP BY k ORDER BY k
```

```
SELECT  
  FlightDate AS k,  
  count()  
FROM ontime  
GROUP BY k  
ORDER BY k ASC
```

```
k   count()  
2017-01-01 | 13970 |  
2017-01-02 | 15882 |  
.....  
2017-09-28 | 16411 |  
2017-09-29 | 16384 |  
2017-09-30 | 12520 |
```

273 rows in set. Elapsed: 0.072 sec. Processed 4.28 million rows, 8.55 MB (59.00 million rows/s., 118.01 MB/s.)

在这个查询中，由于没有使用索引，所以ClickHouse将处理整个表的所有数据(Processed 4.28 million rows)。使用下面的查询尝试使用索引进行查询：

```
:) SELECT FlightDate AS k, count() FROM ontime WHERE k = '2017-09-15' GROUP BY k ORDER BY k
```

```
SELECT
  FlightDate AS k,
  count()
FROM ontime
WHERE k = '2017-09-15'
GROUP BY k
ORDER BY k ASC
```

k	count()
2017-09-15	16428

1 rows in set. Elapsed: 0.014 sec. Processed 32.74 thousand rows, 65.49 KB (2.31 million rows/s., 4.63 MB/s.)

在最后一行的显示中，通过索引ClickHouse处理的行数明显减少（Processed 32.74 thousand rows）。

现在将表达式k = '2017-09-15'传递给indexHint函数：

```
:) SELECT FlightDate AS k, count() FROM ontime WHERE indexHint(k = '2017-09-15') GROUP BY k ORDER BY k
```

```
SELECT
  FlightDate AS k,
  count()
FROM ontime
WHERE indexHint(k = '2017-09-15')
GROUP BY k
ORDER BY k ASC
```

k	count()
2017-09-14	7071
2017-09-15	16428
2017-09-16	1077
2017-09-30	8167

4 rows in set. Elapsed: 0.004 sec. Processed 32.74 thousand rows, 65.49 KB (8.97 million rows/s., 17.94 MB/s.)

对于这个请求，根据ClickHouse显示ClickHouse与上一次相同的方式应用了索引（Processed 32.74 thousand rows）。但是，最终返回的结果集中并没有根据k = '2017-09-15'表达式进行过滤结果。

由于ClickHouse中使用稀疏索引，因此在读取范围时（本示例中为相邻日期），“额外”的数据将包含在索引结果中。使用indexHint函数可以查看到它们。

复制

使用单个值填充一个数组。

用于arrayJoin的内部实现。

```
replicate(x, arr)
```

参数：

- arr — 原始数组。ClickHouse创建一个与原始数据长度相同的新数组，并用值x填充它。
- x — 生成的数组将被填充的值。

输出

- 一个被x填充的数组。

示例

```
SELECT replicate(1, ['a', 'b', 'c'])  
replicate(1, ['a', 'b', 'c'])  
[1,1,1]
```

文件系统可用

返回磁盘的剩余空间信息（以字节为单位）。使用配置文件中的path配置评估此信息。

文件系统容量

返回磁盘的容量信息，以字节为单位。使用配置文件中的path配置评估此信息。

最后聚会

获取聚合函数的状态。返回聚合结果（最终状态）。

跑累积

获取聚合函数的状态并返回其具体的值。这是从第一行到当前行的所有行累计的结果。

例如，获取聚合函数的状态（示例runningAccumulate(uniqState(UserID)))），对于数据块的每一行，返回所有先前行和当前行的状态合并后的聚合函数的结果。

因此，函数的结果取决于分区中数据块的顺序以及数据块中行的顺序。

joinGet('join_storage_table_name', 'get_column', join_key)

使用指定的连接键从Join类型引擎的表中获取数据。

modelEvaluate(model_name, ...)

使用外部模型计算。

接受模型的名称以及模型的参数。返回Float64类型的值。

throwIf(x)

如果参数不为零则抛出异常。

取整函数

楼(x[,N])

返回小于或等于x的最大舍入数。该函数使用参数乘 $1/10^N$ ，如果 $1/10^N$ 不精确，则选择最接近的精确的适当数据类型的数。

'N'是一个整数常量，可选参数。默认为0，这意味着不对其进行舍入。

'N'可以是负数。

示例: floor(123.45, 1) = 123.4, floor(123.45, -1) = 120.

x是任何数字类型。结果与其为相同类型。

对于整数参数，使用负'N'值进行舍入是有意义的（对于非负«N»，该函数不执行任何操作）。

如果取整导致溢出（例如，floor(-128, -1)），则返回特定于实现的结果。

ceil(x[,N]), 天花板(x[,N])

返回大于或等于'x'的最小舍入数。在其他方面，它与'floor'功能相同（见上文）。

圆形(x[,N])

将值取整到指定的小数位数。

该函数按顺序返回最近的数字。如果给定数字包含多个最近数字，则函数返回其中最接近偶数的数字（银行的取整方式）。

```
round(expression [, decimal_places])
```

参数：

- **expression** — 要进行取整的数字。可以是任何返回数字类型的表达式。
- **decimal_places** — 整数类型。
 - 如果`decimal-places > 0`，则该函数将值舍入小数点右侧。
 - 如果`decimal-places < 0`，则该函数将小数点左侧的值四舍五入。
 - 如果`decimal-places = 0`，则该函数将该值舍入为整数。在这种情况下，可以省略参数。

返回值：

与输入数字相同类型的取整后的数字。

示例

使用示例

```
SELECT number / 2 AS x, round(x) FROM system.numbers LIMIT 3
```

x	round(divide(number, 2))
0	0
0.5	0
1	1

取整的示例

取整到最近的数字。

```
round(3.2, 0) = 3  
round(4.1267, 2) = 4.13  
round(22,-1) = 20  
round(467,-2) = 500  
round(-467,-2) = -500
```

银行的取整。

```
round(3.5) = 4  
round(4.5) = 4  
round(3.55, 1) = 3.6  
round(3.65, 1) = 3.6
```

roundToExp2(num)

接受一个数字。如果数字小于1，则返回0。否则，它将数字向下舍入到最接近的（整个非负）2的x次幂。

圆形饱和度(num)

接受一个数字。如果数字小于1，则返回0。否则，它将数字向下舍入为集合中的数字：

1, 10, 30, 60, 120, 180, 240, 300, 600, 1200, 1800, 3600, 7200, 18000, 36000。此函数用于Yandex.Metrica报表中计算会话的持续时长。

圆数(num)

接受一个数字。如果数字小于18，则返回0。否则，它将数字向下舍入为集合中的数字：18, 25, 35, 45, 55。此函数用于Yandex.Metrica报表中用户年龄的计算。

roundDown(num,arr)

接受一个数字，将其向下舍入到指定数组中的元素。如果该值小于数组中的最低边界，则返回最低边界。

字典函数

有关连接和配置外部词典的信息，请参阅[外部词典](#)。

dictGetUInt8,dictGetInt8,dictGetUInt16,dictGetInt16,dictGetUInt32,dictGetInt32,dictGetUInt64,dictGetInt64

dictGetFloat32,dictGetFloat64

dictGetDate,dictGetDateTime

dictGetuid

dictGetString

`dictGetT('dict_name', 'attr_name', id)`

- 使用'id'键获取dict_name字典中attr_name属性的值。dict_name和attr_name是常量字符串。id必须是UInt64。如果字典中没有id键，则返回字典描述中指定的默认值。

dictGetTOrDefault

`dictGetTOrDefault('dict_name', 'attr_name', id, default)`

与dictGetT函数相同，但默认值取自函数的最后一个参数。

dictIsIn

`dictIsIn ('dict_name', child_id, ancestor_id)`

- 对于'dict_name'分层字典，查找'child_id'键是否位于'ancestor_id'内（或匹配'ancestor_id'）。返回UInt8。

独裁主义

`dictGetHierarchy('dict_name', id)`

- 对于'dict_name'分层字典，返回从'id'开始并沿父元素链继续的字典键数组。返回Array (UInt64)

dictHas

```
dictHas('dict_name', id)
```

- 检查字典是否存在指定的id。如果不存在，则返回0；如果存在，则返回1。

字符串函数

empty

对于空字符串返回1，对于非空字符串返回0。

结果类型是UInt8。

如果字符串包含至少一个字节，则该字符串被视为非空字符串，即使这是一个空格或空字符。

该函数也适用于数组。

notEmpty

对于空字符串返回0，对于非空字符串返回1。

结果类型是UInt8。

该函数也适用于数组。

length

返回字符串的字节长度。

结果类型是UInt64。

该函数也适用于数组。

lengthUTF8

假定字符串以UTF-8编码组成的文本，返回此字符串的Unicode字符长度。如果传入的字符串不是UTF-8编码，则函数可能返回一个预期外的值（不会抛出异常）。

结果类型是UInt64。

char_length,CHAR_LENGTH

假定字符串以UTF-8编码组成的文本，返回此字符串的Unicode字符长度。如果传入的字符串不是UTF-8编码，则函数可能返回一个预期外的值（不会抛出异常）。

结果类型是UInt64。

character_length,CHARACTER_LENGTH

假定字符串以UTF-8编码组成的文本，返回此字符串的Unicode字符长度。如果传入的字符串不是UTF-8编码，则函数可能返回一个预期外的值（不会抛出异常）。

结果类型是UInt64。

lower, lcase

将字符串中的ASCII转换为小写。

upper, ucase

将字符串中的ASCII转换为大写。

lowerUTF8

将字符串转换为小写，函数假设字符串是以UTF-8编码文本的字符集。

同时函数不检测语言。因此对土耳其人来说，结果可能不完全正确。

如果UTF-8字节序列的长度对于代码点的大写和小写不同，则该代码点的结果可能不正确。

如果字符串包含一组非UTF-8的字节，则将引发未定义行为。

upperUTF8

将字符串转换为大写，函数假设字符串是以UTF-8编码文本的字符集。

同时函数不检测语言。因此对土耳其人来说，结果可能不完全正确。

如果UTF-8字节序列的长度对于代码点的大写和小写不同，则该代码点的结果可能不正确。

如果字符串包含一组非UTF-8的字节，则将引发未定义行为。

isValidUTF8

检查字符串是否为有效的UTF-8编码，是则返回1，否则返回0。

toValidUTF8

用◆（U+FFFD）字符替换无效的UTF-8字符。所有连续的无效字符都会被替换为一个替换字符。

```
toValidUTF8( input_string )
```

参数：

- `input_string` — 任何一个字符串类型的对象。

返回值：有效的UTF-8字符串。

示例

```
SELECT toValidUTF8('\x61\xF0\x80\x80\x80b')
```

```
toValidUTF8('a◆◆◆◆b')  
a◆b
```

reverse

反转字符串。

reverseUTF8

以Unicode字符为单位反转UTF-8编码的字符串。如果字符串不是UTF-8编码，则可能获取到一个非预期的结果（不会抛出异常）。

format(pattern, s0, s1, ...)

使用常量字符串pattern格式化其他参数。pattern字符串中包含由大括号{}包围的«替换字段»。未被包含在大括号中的任何内容都被视为文本内容，它将原样保留在返回值中。如果你需要在文本内容中包含一个大括号字符，它可以通过加倍来转义：{{和{{'{}'}}}}。字段名称可以是数字（从零开始）或空（然后将它们视为连续数字）

```
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
```

concat(s1, s2, ...)

将参数中的多个字符串拼接，不带分隔符。

concatAssumeInjective(s1, s2, ...)

与**concat**相同，区别在于，你需要保证concat(s1, s2, s3) -> s4是单射的，它将用于GROUP BY的优化。

substring(s,offset,length),mid(s,offset,length),substr(s,offset,length)

以字节为单位截取指定位置字符串，返回以'offset'位置为开头，长度为'length'的子串。'offset'从1开始（与标准SQL相同）。'offset'和'length'参数必须是常量。

substringUTF8(s,offset,length)

与'substring'相同，但其操作单位为Unicode字符，函数假设字符串是以UTF-8进行编码的文本。如果不是则可能返回一个预期外的结果（不会抛出异常）。

appendTrailingCharIfAbsent(s,c)

如果's'字符串非空并且末尾不包含'c'字符，则将'c'字符附加到末尾。

convertCharset(s,from,to)

返回从'from'中的编码转换为'to'中的编码的字符串's'。

base64Encode(s)

将字符串's'编码成base64

base64Decode(s)

使用base64将字符串解码成原始字符串。如果失败则抛出异常。

tryBase64Decode(s)

使用base64将字符串解码成原始字符串。但如果出现错误，将返回空字符串。

endsWith(s,后缀)

返回是否以指定的后缀结尾。如果字符串以指定的后缀结束，则返回1，否则返回0。

startsWith (s, 前缀)

返回是否以指定的前缀开头。如果字符串以指定的前缀开头，则返回1，否则返回0。

trimLeft(s)

返回一个字符串，用于删除左侧的空白字符。

trimRight(s)

返回一个字符串，用于删除右侧的空白字符。

trimBoth(s)

返回一个字符串，用于删除任一侧的空白字符。

字符串拆分合并函数

splitByChar (分隔符, s)

将字符串以'separator'拆分成多个子串。'separator'必须为仅包含一个字符的字符串常量。

返回拆分后的子串的数组。如果分隔符出现在字符串的开头或结尾，或者如果有多个连续的分隔符，则将在对应位置填充空的子串。

splitByString(分隔符, s)

与上面相同，但它使用多个字符的字符串作为分隔符。该字符串必须为非空。

arrayStringConcat(arr[, 分隔符])

使用separator将数组中列出的字符串拼接起来。'separator'是一个可选参数：一个常量字符串，默认情况下设置为空字符串。

返回拼接后的字符串。

alphaTokens(s)

从范围a-z和A-Z中选择连续字节的子字符串。返回子字符串数组。

示例：

```
SELECT alphaTokens('abca1abc')
alphaTokens('abca1abc') └
[ 'abca', 'abc' ] ┌
```

字符串搜索函数

下列所有函数在默认的情况下区分大小写。对于不区分大小写的搜索，存在单独的变体。

位置（大海捞针），定位（大海捞针）

在字符串haystack中搜索子串needle。

返回子串的位置（以字节为单位），从1开始，如果未找到子串，则返回0。

对于不区分大小写的搜索，请使用函数positionCaseInsensitive。

positionUTF8(大海捞针)

与position相同，但位置以Unicode字符返回。此函数工作在UTF-8编码的文本字符集中。如非此编码的字符集，则返回一些非预期结果（他不会抛出异常）。

对于不区分大小写的搜索，请使用函数positionCaseInsensitiveUTF8。

多搜索分配（干草堆，[针₁，针₂，..., needle_n])

与position相同，但函数返回一个数组，其中包含所有匹配needle_i的位置。

对于不区分大小写的搜索或/和UTF-8格式，使用函

数multiSearchAllPositionsCaseInsensitive，multiSearchAllPositionsUTF8，multiSearchAllPositionsCaseInsensitiveUTF8。

multiSearchFirstPosition(大海捞针,[针₁，针₂，..., needle_n])

与 `position` 相同，但返回在 `haystack` 中与 `needles` 字符串匹配的最左偏移。

对于不区分大小写的搜索或/和 UTF-8 格式，使用函

数 `multiSearchFirstPositionCaseInsensitive`，`multiSearchFirstPositionUTF8`，`multiSearchFirstPositionCaseInsensitiveUTF8`。

multiSearchFirstIndex(大海捞针,[针₁, 针₂, ..., needle_n])

返回在字符串 `haystack` 中最先查找到的 `needlei` 的索引 `i` (从 1 开始)，没有找到任何匹配项则返回 0。

对于不区分大小写的搜索或/和 UTF-8 格式，使用函

数 `multiSearchFirstIndexCaseInsensitive`，`multiSearchFirstIndexUTF8`，`multiSearchFirstIndexCaseInsensitiveUTF8`。

多搜索 (大海捞针,[针₁, 针₂, ..., needle_n])

如果 `haystack` 中至少存在一个 `needlei` 匹配则返回 1，否则返回 0。

对于不区分大小写的搜索或/和 UTF-8 格式，使用函

数 `multiSearchAnyCaseInsensitive`，`multiSearchAnyUTF8`，`multiSearchAnyCaseInsensitiveUTF8`。

注意

在所有 `multiSearch*` 函数中，由于实现规范，`needles` 的数量应小于 2^8 。

匹配 (大海捞针，模式)

检查字符串是否与 `pattern` 正则表达式匹配。`pattern` 可以是一个任意的 `re2` 正则表达式。`re2` 正则表达式的语法比 `Perl` 正则表达式的语法存在更多限制。

如果不匹配返回 0，否则返回 1。

请注意，反斜杠符号 (\) 用于在正则表达式中转义。由于字符串中采用相同的符号来进行转义。因此，为了在正则表达式中转义符号，必须在字符串文字中写入两个反斜杠 (\)。

正则表达式与字符串一起使用，就像它是一组字节一样。正则表达式中不能包含空字节。

对于在字符串中搜索子字符串的模式，最好使用 `LIKE` 或 «`position`»，因为它们更加高效。

multiMatchAny (大海捞针，[模式₁, 模式₂, ..., pattern_n])

与 `match` 相同，但如果所有正则表达式都不匹配，则返回 0；如果任何模式匹配，则返回 1。它使用 [超扫描](#) 库。对于在字符串中搜索子字符串的模式，最好使用 «`multisearchany`»，因为它更高效。

注意

任何 `haystack` 字符串的长度必须小于 $2^{32} \times \text{sup--}$ 字节，否则抛出异常。这种限制是因为 `hyperscan API` 而产生的。

multiMatchAnyIndex (大海捞针，[模式₁, 模式₂, ..., pattern_n])

与 `multiMatchAny` 相同，但返回与 `haystack` 匹配的任何内容的索引位置。

multiFuzzyMatchAny(干草堆, 距离,[模式₁, 模式₂, ..., pattern_n])

与 `multiMatchAny` 相同，但如果在 `haystack` 能够查找到任何模式匹配能够在指定的 [编辑距离](#) 内进行匹配，则返回 1。此功能也处于实验模式，可能非常慢。有关更多信息，请参阅 [hyperscan 文档](#)。

multiFuzzyMatchAnyIndex(大海捞针, 距离, [模式₁, 模式₂, ..., pattern_n])

与 multiFuzzyMatchAny 相同，但返回匹配项的匹配能容的索引位置。

注意

multiFuzzyMatch* 函数不支持 UTF-8 正则表达式，由于 hyperscan 限制，这些表达式被按字节解析。

注意

如要关闭所有 hyperscan 函数的使用，请设置 SET allow_hyperscan = 0;。

提取（大海捞针，图案）

使用正则表达式截取字符串。如果 'haystack' 与 'pattern' 不匹配，则返回空字符串。如果正则表达式中不包含子模式，它将获取与整个正则表达式匹配的子串。否则，它将获取与第一个子模式匹配的子串。

extractAll (大海捞针，图案)

使用正则表达式提取字符串的所有片段。如果 'haystack' 与 'pattern' 正则表达式不匹配，则返回一个空字符串。否则返回所有与正则表达式匹配的字符串数组。通常，行为与 'extract' 函数相同（它采用第一个子模式，如果没有子模式，则采用整个表达式）。

像（干草堆，模式），干草堆像模式运算符

检查字符串是否与简单正则表达式匹配。

正则表达式可以包含的元符号有 % 和 _。

% 表示任何字节数（包括零字符）。

_ 表示任何一个字节。

可以使用反斜杠 (\) 来对元符号进行转义。请参阅 «match» 函数说明中有关转义的说明。

对于像 %needle% 这样的正则表达式，改函数与 position 函数一样快。

对于其他正则表达式，函数与 'match' 函数相同。

不喜欢（干草堆，模式），干草堆不喜欢模式运算符

与 'like' 函数返回相反的结果。

大海捞针)

基于 4-gram 计算 haystack 和 needle 之间的距离：计算两个 4-gram 集合之间的对称差异，并用它们的基数和对其进行归一化。返回 0 到 1 之间的任何浮点数 - 越接近 0 则表示越多的字符串彼此相似。如果常量的 needle 或 haystack 超过 32KB，函数将抛出异常。如果非常量的 haystack 或 needle 字符串超过 32Kb，则距离始终为 1。

对于不区分大小写的搜索或 / 和 UTF-8 格式，使用函

数 ngramDistanceCaseInsensitive, ngramDistanceUTF8, ngramDistanceCaseInsensitiveUTF8。

ツ暗エツ氾环催ツ団ツ法ツ人)

与 ngramDistance 相同，但计算 needle 和 haystack 之间的非对称差异——needle 的 n-gram 减去 needle 归一化 n-gram。可用于模糊字符串搜索。

对于不区分大小写的搜索或/和UTF-8格式，使用函

数ngramSearchCaseInsensitive，ngramSearchUTF8，ngramSearchCaseInsensitiveUTF8。

注意

对于UTF-8，我们使用3-gram。所有这些都不是完全公平的n-gram距离。我们使用2字节哈希来散列n-gram，然后计算这些哈希表之间的（非）对称差异 - 可能会发生冲突。对于UTF-8不区分大小写的格式，我们不使用公平的tolower函数 - 我们将每个Unicode字符字节的第5位（从零开始）和字节的第一位归零 - 这适用于拉丁语，主要用于所有西里尔字母。

字符串替换函数

replaceOne(haystack, pattern, replacement)

用'replacement'子串替换'haystack'中第一次出现的'pattern'子串（如果存在）。
'pattern'和'replacement'必须是常量。

replaceAll(haystack, pattern, replacement), replace(haystack, pattern, replacement)

用'replacement'子串替换'haystack'中出现的所有'pattern'子串。

replaceRegexpOne(haystack, pattern, replacement)

使用'pattern'正则表达式的替换。'pattern'可以是任意一个有效的re2正则表达式。
如果存在与'pattern'正则表达式匹配的匹配项，仅替换第一个匹配项。
模式pattern可以指定为'replacement'。此模式可以包含替代\0-\9。
替代\0包含了整个正则表达式。替代\1-\9对应于子模式编号。要在模板中使用反斜杠\，请使用\\将其转义。
另外还请记住，字符串字面值(literal)需要额外的转义。

示例1. 将日期转换为美国格式：

```
SELECT DISTINCT
  EventDate,
  replaceRegexpOne(toString(EventDate), '(\d{4})-(\d{2})-(\d{2})', '\1/\2/\3') 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

示例2. 复制字符串十次：

```
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! |
```

replaceRegexpAll(haystack, pattern, replacement)

与replaceRegexpOne相同，但会替换所有出现的匹配项。例如：

```
SELECT replaceRegexpAll('Hello, World!', '.', '\\0\\0') AS res
```

```
res
HHeelllloo,, WWoorrlldd!! |
```

作为例外，对于空子字符串，正则表达式只会进行一次替换。

示例：

```
SELECT replaceRegexpAll('Hello, World!', '^', 'here: ') AS res
```

```
res
here: Hello, World! |
```

regexpQuoteMeta(s)

该函数用于在字符串中的某些预定义字符之前添加反斜杠。

预定义字符：`\0, \\", |, (,), ^, $, ., [,], ?, *, +, {, :, -`。

这个实现与re2::RE2::QuoteMeta略有不同。它以`\0`转义零字节，而不是`\x00`，并且只转义必需的字符。

有关详细信息，请参阅链接：[RE2](#)

数学函数

以下所有的函数都返回一个Float64类型的数值。返回结果总是以尽可能最大精度返回，但还是可能与机器中可表示最接近该值的数字不同。

e()

返回一个接近数学常量e的Float64数字。

pi()

返回一个接近数学常量π的Float64数字。

exp(x)

接受一个数值类型的参数并返回它的指数。

log(x),ln(x)

接受一个数值类型的参数并返回它的自然对数。

exp2(x)

接受一个数值类型的参数并返回它的 2^x 次幂。

log2(x)

接受一个数值类型的参数并返回它的底2对数。

exp10(x)

接受一个数值类型的参数并返回它的 10^x 次幂。

log10(x)

接受一个数值类型的参数并返回它的底10对数。

sqrt(x)

接受一个数值类型的参数并返回它的平方根。

cbrt(x)

接受一个数值类型的参数并返回它的立方根。

erf(x)

如果'x'是非负数，那么 $\text{erf}(x / \sigma\sqrt{2})$ 是具有正态分布且标准偏差为« σ »的随机变量的值与预期值之间的距离大于« x »。

示例（三西格玛准则）：

```
SELECT erf(3 / sqrt(2))
```

```
erf(divide(3, sqrt(2)))—  
0.9973002039367398 |
```

erfc(x)

接受一个数值参数并返回一个接近 $1 - \text{erf}(x)$ 的Float64数字，但不会丢失大« x »值的精度。

lgamma(x)

返回 x 的绝对值的自然对数的伽玛函数。

tgamma(x)

返回 x 的伽玛函数。

sin(x)

返回 x 的三角正弦值。

cos(x)

返回 x 的三角余弦值。

tan(x)

返回x的三角正切值。

asin(x)

返回x的反三角正弦值。

acos(x)

返回x的反三角余弦值。

atan(x)

返回x的反三角正切值。

pow(x,y),power(x,y)

接受x和y两个参数。返回x的y次方。

intExp2

接受一个数值类型的参数并返回它的2的x次幂 (UInt64)。

intExp10

接受一个数值类型的参数并返回它的10的x次幂 (UInt64)。

数组函数

empty

对于空数组返回1，对于非空数组返回0。

结果类型是UInt8。

该函数也适用于字符串。

notEmpty

对于空数组返回0，对于非空数组返回1。

结果类型是UInt8。

该函数也适用于字符串。

length

返回数组中的元素个数。

结果类型是UInt64。

该函数也适用于字符串。

emptyArrayUInt8,emptyArrayUInt16,emptyArrayUInt32,empty/

emptyArrayInt8,emptyArrayInt16,emptyArrayInt32,emptyArray

emptyArrayFloat32,emptyArrayFloat64

emptyArrayDate , emptyArrayDateTime

emptyArrayString

不接受任何参数并返回适当类型的空数组。

emptyArrayToSingle

接受一个空数组并返回一个仅包含一个默认值元素的数组。

range(N)

返回从0到N-1的数字数组。

以防万一，如果在数据块中创建总长度超过100,000,000个元素的数组，则抛出异常。

array(x1, ...), operator [x1, ...]

使用函数的参数作为数组元素创建一个数组。

参数必须是常量，并且具有最小公共类型的类型。必须至少传递一个参数，否则将不清楚要创建哪种类型的数组。也就是说，你不能使用这个函数来创建一个空数组（为此，使用上面描述的'emptyArray *'函数）。

返回'Array (T)'类型的结果，其中'T'是传递的参数中最小的公共类型。

arrayConcat

合并参数中传递的所有数组。

```
arrayConcat(arrays)
```

参数

- arrays – 任意数量的阵列类型的参数。

示例

```
SELECT arrayConcat([1, 2], [3, 4], [5, 6]) AS res
```

```
res  
[1,2,3,4,5,6] |
```

arrayElement(arr,n), 运算符arr[n]

从数组arr中获取索引为«n»的元素。 n 必须是任何整数类型。

数组中的索引从一开始。

支持负索引。在这种情况下，它选择从末尾开始编号的相应元素。例如，arr [-1]是数组中的最后一项。

如果索引超出数组的边界，则返回默认值（数字为0，字符串为空字符串等）。

has(arr,elem)

检查'arr'数组是否具有'elem'元素。

如果元素不在数组中，则返回0;如果在，则返回1。

NULL 值的处理。

```
SELECT has([1, 2, NULL], NULL)
```

```
has([1, 2, NULL], NULL)  
1 |
```

hasAll

检查一个数组是否是另一个数组的子集。

```
hasAll(set, subset)
```

参数

- **set** – 具有一组元素的任何类型的数组。
- **subset** – 任何类型的数组，其元素应该被测试为**set**的子集。

返回值

- **1**，如果**set**包含**subset**中的所有元素。
- **0**，否则。

特殊的定义

- 空数组是任何数组的子集。
- «Null»作为数组中的元素值进行处理。
- 忽略两个数组中的元素值的顺序。

示例

`SELECT hasAll([], [])` 返回1。

`SELECT hasAll([1, Null], [Null])` 返回1。

`SELECT hasAll([1.0, 2, 3, 4], [1, 3])` 返回1。

`SELECT hasAll(['a', 'b'], ['a'])` 返回1。

`SELECT hasAll([1], ['a'])` 返回0。

`SELECT hasAll([[1, 2], [3, 4]], [[1, 2], [3, 5]])` 返回0。

hasAny

检查两个数组是否存在交集。

```
hasAny(array1, array2)
```

参数

- **array1** – 具有一组元素的任何类型的数组。
- **array2** – 具有一组元素的任何类型的数组。

返回值

- **1**，如果**array1**和**array2**存在交集。
- **0**，否则。

特殊的定义

- «Null»作为数组中的元素值进行处理。
- 忽略两个数组中的元素值的顺序。

示例

SELECT hasAny([1], []) 返回 0.

SELECT hasAny([Null], [Null, 1]) 返回 1.

SELECT hasAny([-128, 1., 512], [1]) 返回 1.

SELECT hasAny([[1, 2], [3, 4]], ['a', 'c']) 返回 0.

SELECT hasAll([[1, 2], [3, 4]], [[1, 2], [1, 2]]) 返回 1.

indexOf(arr,x)

返回数组中第一个'x'元素的索引（从1开始），如果'x'元素不存在在数组中，则返回0。

示例：

```
:) SELECT indexOf([1,3,NULL,NULL],NULL)
SELECT indexOf([1, 3, NULL, NULL], NULL)
  └─ indexOf([1, 3, NULL, NULL], NULL) ─
      3 |
```

设置为«NULL»的元素将作为普通的元素值处理。

countEqual(arr,x)

返回数组中等于x的元素的个数。相当于arrayCount (elem -> elem = x, arr)。

NULL值将作为单独的元素值处理。

示例：

```
SELECT countEqual([1, 2, NULL, NULL], NULL)
  └─ countEqual([1, 2, NULL, NULL], NULL) ─
      2 |
```

arrayEnumerate(arr)

返回 Array [1, 2, 3, ..., length (arr)]

此功能通常与ARRAY JOIN一起使用。它允许在应用ARRAY JOIN后为每个数组计算一次。例如：

```
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

在此示例中，Reaches是转换次数（应用ARRAY JOIN后接收的字符串），Hits是浏览量（ARRAY JOIN之前的字符串）。在这种特殊情况下，您可以更轻松地获得相同的结果：

```
SELECT
    sum(length(GoalsReached)) AS Reaches,
    count() AS Hits
FROM test.hits
WHERE (CounterID = 160656) AND notEmpty(GoalsReached)
```

Reaches	Hits
95606	31406

此功能也可用于高阶函数。例如，您可以使用它来获取与条件匹配的元素的数组索引。

arrayEnumerateUniq(arr, ...)

返回与源数组大小相同的数组，其中每个元素表示与其下标对应的源数组元素在源数组中出现的次数。

例如：arrayEnumerateUniq ([10,20,10,30]) = [1,1,2,1]。

使用ARRAY JOIN和数组元素的聚合时，此函数很有用。

示例：

```
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

在此示例中，每个GoalID都计算转换次数（目标嵌套数据结构中的每个元素都是达到的目标，我们称之为转换）和会话数。如果没有ARRAY JOIN，我们会将会话数计为总和（Sign）。但在这种特殊情况下，行乘以嵌套的Goals结构，因此为了在此之后计算每个会话一次，我们将一个条件应用于arrayEnumerateUniq (Goals.ID) 函数的值。

arrayEnumerateUniq函数可以使用与参数大小相同的多个数组。在这种情况下，对于所有阵列中相同位置的元素元组，考虑唯一性。

```
SELECT arrayEnumerateUniq([1, 1, 1, 2, 2, 2], [1, 1, 2, 1, 1, 2]) AS res
```

```
res  
[1,2,1,1,2,1] |
```

当使用带有嵌套数据结构的ARRAY JOIN并在此结构中跨多个元素进一步聚合时，这是必需的。

arrayPopBack

从数组中删除最后一项。

```
arrayPopBack(array)
```

参数

- **array** – 数组。

示例

```
SELECT arrayPopBack([1, 2, 3]) AS res
```

```
res  
[1,2] |
```

arrayPopFront

从数组中删除第一项。

```
arrayPopFront(array)
```

参数

- **array** – 数组。

示例

```
SELECT arrayPopFront([1, 2, 3]) AS res
```

```
res  
[2,3] |
```

arrayPushBack

添加一个元素到数组的末尾。

```
arrayPushBack(array, single_value)
```

参数

- **array** – 数组。

- `single_value` – 单个值。只能将数字添加到带数字的数组中，并且只能将字符串添加到字符串数组中。添加数字时，ClickHouse会自动为数组的数据类型设置`single_value`类型。有关ClickHouse中数据类型的更多信息，请参阅«[数据类型](#)»。可以是'NULL`。该函数向数组添加一个«NULL»元素，数组元素的类型转换为`Nullable``。

示例

```
SELECT arrayPushBack(['a'], 'b') AS res
```

res

['a','b'] |

arrayPushFront

将一个元素添加到数组的开头。

```
arrayPushFront(array, single_value)
```

参数

- `array` – 数组。
- `single_value` – 单个值。只能将数字添加到带数字的数组中，并且只能将字符串添加到字符串数组中。添加数字时，ClickHouse会自动为数组的数据类型设置`single_value`类型。有关ClickHouse中数据类型的更多信息，请参阅«[数据类型](#)»。可以是'NULL`。该函数向数组添加一个«NULL»元素，数组元素的类型转换为`Nullable``。

示例

```
SELECT arrayPushFront(['b'], 'a') AS res
```

res

['a','b'] |

arrayResize

更改数组的长度。

```
arrayResize(array, size[, extender])
```

参数：

- `array` — 数组。
- `size` — 数组所需的长度。
 - 如果`size`小于数组的原始大小，则数组将从右侧截断。
 - 如果`size`大于数组的初始大小，则使用`extender`值或数组项的数据类型的默认值将数组扩展到右侧。
- `extender` — 扩展数组的值。可以是'NULL`。

返回值：

一个`size`长度的数组。

调用示例

```
SELECT arrayResize([1], 3)
```

```
arrayResize([1], 3)─  
[1,0,0] ┊
```

```
SELECT arrayResize([1], 3, NULL)
```

```
arrayResize([1], 3, NULL)─  
[1,NULL,NULL] ┊
```

arraySlice

返回一个子数组，包含从指定位置的指定长度的元素。

```
arraySlice(array, offset[, length])
```

参数

- **array** – 数组。
- **offset** – 数组的偏移。正值表示左侧的偏移量，负值表示右侧的缩进值。数组下标从1开始。
- **length** - 子数组的长度。如果指定负值，则该函数返回[offset, array_length - length]。如果省略该值，则该函数返回[offset, the_end_of_array]。

示例

```
SELECT arraySlice([1, 2, NULL, 4, 5], 2, 3) AS res
```

```
res─  
[2,NULL,4] ┊
```

设置为«NULL»的数组元素作为普通的数组元素值处理。

arraySort([func,] arr, ...)

以升序对arr数组的元素进行排序。如果指定了func函数，则排序顺序由func函数的调用结果决定。如果func接受多个参数，那么arraySort函数也将解析与func函数参数相同数量的数组参数。更详细的示例在arraySort的末尾。

整数排序示例:

```
SELECT arraySort([1, 3, 3, 0]);
```

```
arraySort([1, 3, 3, 0])─  
[0,1,3,3] ┊
```

字符串排序示例:

```
SELECT arraySort(['hello', 'world', '!']);
```

```
arraySort(['hello', 'world', '!'])  
[!, 'hello', 'world'] |
```

NULL, NaN和Inf的排序顺序：

```
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 是数组中的第一个。
- NULL 是数组中的最后一个。
- NaN 在NULL的前面。
- Inf 在NaN的前面。

注意：arraySort是高阶函数。您可以将lambda函数作为第一个参数传递给它。在这种情况下，排序顺序由lambda函数的调用结果决定。

让我们来看一下如下示例：

```
SELECT arraySort((x) -> -x, [1, 2, 3]) as res;
```

```
res  
[3,2,1] |
```

对于源数组的每个元素，lambda函数返回排序键，即[1 -> -1, 2 -> -2, 3 -> -3]。由于arraySort函数按升序对键进行排序，因此结果为[3,2,1]。因此，(x) -> -x lambda函数将排序设置为降序。

lambda函数可以接受多个参数。在这种情况下，您需要为arraySort传递与lambda参数个数相同的数组。函数使用第一个输入的数组中的元素组成返回结果；使用接下来传入的数组作为排序键。例如：

```
SELECT arraySort((x, y) -> y, ['hello', 'world'], [2, 1]) as res;
```

```
res  
['world', 'hello'] |
```

这里，在第二个数组 ([2, 1]) 中定义了第一个数组 (['hello', 'world']) 的相应元素的排序键，即['hello' -> 2, 'world' -> 1]。由于lambda函数中没有使用x，因此源数组中的实际值不会影响结果的顺序。所以，'world'将是结果中的第一个元素，'hello'将是结果中的第二个元素。

其他示例如下所示。

```
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] |
```

注意

为了提高排序效率，使用了施瓦茨变换。

arrayReverseSort([func,] arr, ...)

以降序对arr数组的元素进行排序。如果指定了func函数，则排序顺序由func函数的调用结果决定。如果func接受多个参数，那么arrayReverseSort函数也将解析与func函数参数相同数量的数组作为参数。更详细的示例在arrayReverseSort的末尾。

整数排序示例：

```
SELECT arrayReverseSort([1, 3, 3, 0]);
```

```
arrayReverseSort([1, 3, 3, 0])  
[3,3,1,0] |
```

字符串排序示例：

```
SELECT arrayReverseSort(['hello', 'world', '!']);
```

```
arrayReverseSort(['hello', 'world', '!'])  
['world','hello','!'] |
```

NULL, NaN和Inf的排序顺序：

```
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 是数组中的第一个。
- NULL 是数组中的最后一个。
- NaN 在NULL的前面。
- -Inf 在NaN的前面。

注意：arraySort是**高阶函数**。您可以将lambda函数作为第一个参数传递给它。如下示例所示。

```
SELECT arrayReverseSort((x) -> -x, [1, 2, 3]) AS res;
```

```
res  
[1,2,3] |
```

数组按以下方式排序：

数组按以下方式排序：

1. 首先，根据lambda函数的调用结果对源数组 ([1, 2, 3]) 进行排序。结果是[3, 2, 1]。
2. 反转上一步获得的数组。所以，最终的结果是[1, 2, 3]。

lambda函数可以接受多个参数。在这种情况下，您需要为arrayReverseSort传递与lambda参数个数相同的数组。函数使用第一个输入的数组中的元素组成返回结果；使用接下来传入的数组作为排序键。例如：

```
SELECT arrayReverseSort((x, y) -> y, ['hello', 'world'], [2, 1]) AS res;
```

```
res  
['hello','world'] |
```

在这个例子中，数组按以下方式排序：

1. 首先，根据lambda函数的调用结果对源数组 (['hello', 'world']) 进行排序。其中，在第二个数组 ([2,1]) 中定义了源数组中相应元素的排序键。所以，排序结果['world', 'hello']。
2. 反转上一步骤中获得的排序数组。所以，最终的结果是['hello', 'world']。

其他示例如下所示。

```
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, ...)

如果传递一个参数，则计算数组中不同元素的数量。

如果传递了多个参数，则它计算多个数组中相应位置的不同元素元组的数量。

如果要获取数组中唯一项的列表，可以使用arrayReduce ('groupUniqArray', arr)。

arrayJoin(arr)

一个特殊的功能。请参见[«ArrayJoin函数»部分](#)。

arrayDifference(arr)

返回一个数组，其中包含所有相邻元素对之间的差值。例如：

```
SELECT arrayDifference([1, 2, 3, 4])
```

```
arrayDifference([1, 2, 3, 4])  
[0,1,1,1]
```

arrayDistinct(arr)

返回一个包含所有数组中不同元素的数组。例如：

```
SELECT arrayDistinct([1, 2, 2, 3, 1])
```

```
arrayDistinct([1, 2, 2, 3, 1])  
[1,2,3]
```

arrayEnumerateDense(arr)

返回与源数组大小相同的数组，指示每个元素首次出现在源数组中的位置。例如：

arrayEnumerateDense ([10,20,10,30]) = [1,2,1,3]。

arrayIntersect(arr)

返回所有数组元素的交集。例如：

```
SELECT  
    arrayIntersect([1, 2], [1, 3], [2, 3]) AS no_intersect,  
    arrayIntersect([1, 2], [1, 3], [1, 4]) AS intersect
```

```
no_intersect  
[] | [1] |
```

arrayReduce(agg_func, arr1, ...)

将聚合函数应用于数组并返回其结果。如果聚合函数具有多个参数，则此函数可应用于相同大小的多个数组。

arrayReduce ('agg_func', arr1, ...) - 将聚合函数agg_func应用于数组arr1 ...。如果传递了多个数组，则相应位置上的元素将作为多个参数传递给聚合函数。例如：SELECT arrayReduce ('max', [1,2,3]) = 3

arrayReverse(arr)

返回与源数组大小相同的数组，包含反转源数组的所有元素的结果。

时间日期函数

支持时区。

所有的时间日期函数都可以在第二个可选参数中接受时区参数。示例：Asia / Yekaterinburg。在这种情况下，它们使用指定的时区而不是本地（默认）时区。

```
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

仅支持与UTC相差一整小时的时区。

toTimeZone

将Date或DateTime转换为指定的时区。时区是Date/DateTime类型的属性。表字段或结果集的列的内部值（秒数）不会更改，列的类型会更改，并且其字符串表示形式也会相应更改。

```
SELECT
    toDateTime('2019-01-01 00:00:00', 'UTC') AS time_utc,
    toTypeName(time_utc) AS type_utc,
   .toInt32(time_utc) AS int32utc,
    toTimezone(time_utc, 'Asia/Yekaterinburg') AS time_yekat,
    toTypeName(time_yekat) AS type_yekat,
   .toInt32(time_yekat) AS int32yekat,
    toTimezone(time_utc, 'US/Samoa') AS time_samoa,
    toTypeName(time_samoa) AS type_samoa,
   .toInt32(time_samoa) AS int32samoa
FORMAT Vertical;
```

Row 1:

```
time_utc: 2019-01-01 00:00:00
type_utc: DateTime('UTC')
int32utc: 1546300800
time_yekat: 2019-01-01 05:00:00
type_yekat: DateTime('Asia/Yekaterinburg')
int32yekat: 1546300800
time_samoa: 2018-12-31 13:00:00
type_samoa: DateTime('US/Samoa')
int32samoa: 1546300800
```

toTimezone(time_utc, 'Asia/Yekaterinburg') 把 `DateTime('UTC')` 类型转换为 `DateTime('Asia/Yekaterinburg')`。内部值 (Unixtimestamp) 1546300800 保持不变，但是字符串表示(toString() 函数的结果值) 由 `time_utc: 2019-01-01 00:00:00` 转换为 `time_yekat: 2019-01-01 05:00:00`。

toYear

将Date或DateTime转换为包含年份编号（AD）的UInt16类型的数字。

toQuarter

将Date或DateTime转换为包含季度编号的UInt8类型的数字。

toMonth

将Date或DateTime转换为包含月份编号（1-12）的UInt8类型的数字。

toDayOfYear

将Date或DateTime转换为包含一年中的某一天的编号的UInt16（1-366）类型的数字。

toDayOfMonth

将Date或DateTime转换为包含一月中的某一天的编号的UInt8（1-31）类型的数字。

toDayOfWeek

将Date或DateTime转换为包含一周中的某一天的编号的UInt8（周一1，周日7）类型的数字。

toHour

将DateTime转换为包含24小时制（0-23）小时数的UInt8数字。

这个函数假设如果时钟向前移动，它是一个小时，发生在凌晨2点，如果时钟被移回，它是一个小时，发生在凌晨3点（这并非总是如此 - 即使在莫斯科时钟在不同的时间两次改变）。

toMinute

将DateTime转换为包含一小时中分钟数（0-59）的UInt8数字。

toSecond

将DateTime转换为包含一分钟中秒数（0-59）的UInt8数字。

闰秒不计算在内。

toUnixTimestamp

对于DateTime参数：将值转换为UInt32类型的数字-Unc时间戳（https://en.wikipedia.org/wiki/Unix_time）。

对于String参数：根据时区将输入字符串转换为日期时间（可选的第二个参数，默认使用服务器时区），并返回相应的unix时间戳。

语法

```
toUnixTimestamp(datetime)
toUnixTimestamp(str, [timezone])
```

返回值

- 返回 unix timestamp.

类型: UInt32.

示例

查询:

```
SELECT toUnixTimestamp('2017-11-05 08:07:47', 'Asia/Tokyo') AS unix_timestamp
```

结果:

```
unix_timestamp
1509836867 |
```

toStartOfYear

将Date或DateTime向前取整到本年的第一天。

返回Date类型。

toStartOfISOYear

将Date或DateTime向前取整到ISO本年的第一天。

返回Date类型。

toStartOfQuarter

将Date或DateTime向前取整到本季度的第一天。

返回Date类型。

toStartOfMonth

将Date或DateTime向前取整到本月的第一天。

返回Date类型。

注意

解析不正确日期的行为是特定于实现的。 ClickHouse可能会返回零日期，抛出异常或执行«natural»溢出。

toMonday

将Date或DateTime向前取整到本周的星期一。

返回Date类型。

toStartOfWeek(t[,mode])

按mode将Date或DateTime向前取整到最近的星期日或星期一。

返回Date类型。

mode参数的工作方式与toWeek()的mode参数完全相同。 对于单参数语法，mode使用默认值0。

toStartOfDay

将DateTime向前取整到今天的开始。

toStartOfHour

将DateTime向前取整到当前小时的开始。

toStartOfMinute

将DateTime向前取整到当前分钟的开始。

toStartOfSecond

将DateTime向前取整到当前秒数的开始。

语法

```
toStartOfSecond(value[, timezone])
```

参数

- value — 时间和日期**DateTime64**.

- `timezone` — 返回值的Timezone (可选参数)。如果未指定将使用 `value` 参数的时区。String。

返回值

- 输入值毫秒部分为零。

类型: `DateTime64`.

示例

不指定时区查询:

```
WITH toDateTime64('2020-01-01 10:20:30.999', 3) AS dt64
SELECT toStartOfSecond(dt64);
```

结果:

```
toStartOfSecond(dt64) └
2020-01-01 10:20:30.000 |
```

指定时区查询:

```
WITH toDateTime64('2020-01-01 10:20:30.999', 3) AS dt64
SELECT toStartOfSecond(dt64, 'Europe/Moscow');
```

结果:

```
toStartOfSecond(dt64, 'Europe/Moscow') └
2020-01-01 13:20:30.000 |
```

参考

- Timezone 服务器配置选项。

toStartOfFiveMinute

将DateTime以五分钟为单位向前取整到最接近的时间点。

toStartOfTenMinutes

将DateTime以十分钟为单位向前取整到最接近的时间点。

toStartOfFifteenMinutes

将DateTime以十五分钟为单位向前取整到最接近的时间点。

toStartOfInterval(time_or_data, 间隔x单位[,time_zone])

这是名为`toStartOf*`的所有函数的通用函数。例如，

`toStartOfInterval (t, INTERVAL 1 year)` 返回与`toStartOfYear (t)`相同的结果，

`toStartOfInterval (t, INTERVAL 1 month)` 返回与`toStartOfMonth (t)`相同的结果，

`toStartOfInterval (t, INTERVAL 1 day)` 返回与`toStartOfDay (t)`相同的结果，

`toStartOfInterval (t, INTERVAL 15 minute)` 返回与`toStartOfFifteenMinutes (t)`相同的结果。

toTime

将DateTime中的日期转换为一个固定的日期，同时保留时间部分。

toRelativeYearNum

将Date或DateTime转换为年份的编号，从过去的某个固定时间点开始。

toRelativeQuarterNum

将Date或DateTime转换为季度的数字，从过去的某个固定时间点开始。

toRelativeMonthNum

将Date或DateTime转换为月份的编号，从过去的某个固定时间点开始。

toRelativeWeekNum

将Date或DateTime转换为星期数，从过去的某个固定时间点开始。

toRelativeDayNum

将Date或DateTime转换为当天的编号，从过去的某个固定时间点开始。

toRelativeHourNum

将DateTime转换为小时数，从过去的某个固定时间点开始。

toRelativeMinuteNum

将DateTime转换为分钟数，从过去的某个固定时间点开始。

toRelativeSecondNum

将DateTime转换为秒数，从过去的某个固定时间点开始。

toISOYear

将Date或DateTime转换为包含ISO年份的UInt16类型的编号。

toISOWeek

将Date或DateTime转换为包含ISO周数的UInt8类型的编号。

toWeek(date[,mode])

返回Date或DateTime的周数。两个参数形式可以指定星期是从星期日还是星期一开始，以及返回值应在0到53还是从1到53的范围内。如果省略了mode参数，则默认 模式为0。

toISOWeek()是一个兼容函数，等效于toWeek(date,3)。

下表描述了mode参数的工作方式。

Mode	First day of week	Range	Week 1 is the first week ...
0	Sunday	0-53	with a Sunday in this year
1	Monday	0-53	with 4 or more days this year
2	Sunday	1-53	with a Sunday in this year

Mode	First day of week	Range	Week 1 is the first week ...
3	Monday	1-53	with 4 or more days this year
4	Sunday	0-53	with 4 or more days this year
5	Monday	0-53	with a Monday in this year
6	Sunday	1-53	with 4 or more days this year
7	Monday	1-53	with a Monday in this year
8	Sunday	1-53	contains January 1
9	Monday	1-53	contains January 1

对于象“with 4 or more days this year,”的mode值，根据ISO 8601：1988对周进行编号：

- 如果包含1月1日的一周在后一年度中有4天或更多天，则为第1周。
- 否则，它是上一年的最后一周，下周是第1周。

对于像“contains January 1”的mode值，包含1月1日的那周为本年度的第1周。

```
toWeek(date, [, mode][, Timezone])
```

参数

- date – Date 或 DateTime.
- mode – 可选参数，取值范围 [0,9]，默认0。
- Timezone – 可选参数，可其他时间日期转换参数的行为一致。

示例

```
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(date[,mode])

返回Date的年和周。结果中的年份可能因为Date为该年份的第一周和最后一周而于Date的年份不同。

mode参数的工作方式与toWeek()的mode参数完全相同。对于单参数语法，mode使用默认值0。

toISOYear()是一个兼容函数，等效于intDiv(toYearWeek(date,3),100).

示例

```
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

date_trunc

将Date或DateTime按指定的单位向前取整到最接近的时间点。

语法

```
date_trunc(unit, value[, timezone])
```

别名: dateTrunc.

参数

- **unit** — 单位. **String**.

可选值:

- second
- minute
- hour
- day
- week
- month
- quarter
- year

- **value** — **DateTime** 或者 **DateTime64**.

- **timezone** — **Timezone name** 返回值的时区(可选值)。如果未指定将使用**value**的时区。 **String**.

返回值

- 按指定的单位向前取整后的**DateTime**。

类型: **Datetime**.

示例

不指定时区查询:

```
SELECT now(), date_trunc('hour', now());
```

结果:

now()	date_trunc('hour', now())
2020-09-28 10:40:45	2020-09-28 10:00:00

指定时区查询:

```
SELECT now(), date_trunc('hour', now(), 'Europe/Moscow');
```

结果:

now()	date_trunc('hour', now(), 'Europe/Moscow')
2020-09-28 10:46:26	2020-09-28 13:00:00

参考

- [toStartOfInterval](#)

now

返回当前日期和时间。

语法

```
now([timezone])
```

参数

- **timezone** — [Timezone name](#) 返回结果的时区(可选参数). [String](#).

返回值

- 当前日期和时间。

类型: [Datetime](#).

示例

不指定时区查询:

```
SELECT now();
```

结果:

now()
2020-10-17 07:42:09

指定期区查询:

```
SELECT now('Europe/Moscow');
```

结果:

now('Europe/Moscow')
2020-10-17 10:42:23

today

不接受任何参数并在请求执行时的某一刻返回当前日期(Date)。

其功能与'toDate (now())'相同。

yesterday

不接受任何参数并在请求执行时的某一刻返回昨天的日期(Date)。

其功能与'today() - 1'相同。

timeSlot

将时间向前取整半小时。

此功能用于Yandex.Metrica，因为如果跟踪标记显示单个用户的连续综合浏览量在时间上严格超过此数量，则半小时是将会话分成两个会话的最短时间。这意味着 (tag id, user id, time slot) 可用于搜索相应会话中包含的综合浏览量。

toYYYYMM

将Date或DateTime转换为包含年份和月份编号的UInt32类型的数字 (YYYY * 100 + MM)。

toYYYYMMDD

将Date或DateTime转换为包含年份和月份编号的UInt32类型的数字 (YYYY * 10000 + MM * 100 + DD)。

toYYYYMMDDhhmmss

将Date或DateTime转换为包含年份和月份编号的UInt64类型的数字 (YYYY * 10000000000 + MM * 100000000 + DD * 1000000 + hh * 10000 + mm * 100 + ss)。

addYears, addMonths, addWeeks, addDays, addHours, addMinutes, addSeconds, addQuarters

函数将一段时间间隔添加到Date/DateTime，然后返回Date/DateTime。例如：

```
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

subtractYears, subtractMonths, subtractWeeks, subtractDays, su

函数将Date/DateTime减去一段时间间隔，然后返回Date/DateTime。例如：

```
WITH
    toDate('2019-01-01') AS date,
    toDateTime('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

返回两个Date或DateTime类型之间的时差。

语法

```
dateDiff('unit', startdate, enddate, [timezone])
```

参数

- **unit** — 返回结果的时间单位。 **String**.

支持的时间单位: second, minute, hour, day, week, month, quarter, year.

- **startdate** — 第一个待比较值。 **Date** 或 **DateTime**.

- **enddate** — 第二个待比较值。 **Date** 或 **DateTime**.

- **timezone** — 可选参数。 如果指定了，则同时适用于**startdate**和**enddate**。如果不指定，则使用**startdate**和**enddate**的时区。如果两个时区不一致，则结果不可预料。

返回值

以**unit**为单位的**startdate**和**enddate**之间的时差。

类型: **int**.

示例

查询:

```
SELECT dateDiff('hour', toDateTime('2018-01-01 22:00:00'), toDateTime('2018-01-02 23:00:00'));
```

结果:

```
dateDiff('hour', toDateTime('2018-01-01 22:00:00'), toDateTime('2018-01-02 23:00:00'))—  
25 |
```

timeSlots(StartTime, Duration[, Size])

它返回一个时间数组，其中包括从从«StartTime»开始到«StartTime + Duration 秒»内的所有符合«size»（以秒为单位）步长的时间点。其中«size»是一个可选参数，默认为1800。

例如，`timeSlots(toDateTime('2012-01-01 12:20:00'), 600) = [toDateTime ('2012-01-01 12:00:00') , toDateTime ('2012-01-01 12:30:00')]`。

这对于搜索在相应会话中综合浏览量是非常有用的。

formatDateTime

函数根据给定的格式字符串来格式化时间。请注意：格式字符串必须是常量表达式，例如：单个结果列不能有多种格式字符串。

语法

```
formatDateTime(Time, Format[, Timezone])
```

返回值

根据指定格式返回的日期和时间。

支持的格式修饰符

使用格式修饰符来指定结果字符串的样式。«Example» 列是对2018-01-02 22:33:44的格式化结果。

修饰符	描述	示例
%C	年除以100并截断为整数(00-99)	20
%d	月中的一天，零填充 (01-31)	02
%D	短MM/DD/YY日期，相当于%m/%d/%y	01/02/2018
%e	月中的一天，空格填充 (1-31)	2
%F	短YYYY-MM-DD日期，相当于%Y-%m-%d	2018-01-02
%G	ISO周号的四位数年份格式，从基于周的年份由 ISO 8601定义 标准计算得出，通常仅对%V有用	2018
%g	两位数的年份格式，与ISO 8601一致，四位数表示法的缩写	18
%H	24小时格式 (00-23)	22
%I	12小时格式 (01-12)	10
%j	一年中的一天 (001-366)	002
%m	月份为十进制数 (01-12)	01
%M	分钟(00-59)	33
%n	换行符("\n")	
%p	AM或PM指定	PM
%Q	季度 (1-4)	1
%R	24小时HH:MM时间，相当于%H:%M	22:33
%S	秒 (00-59)	44
%t	水平制表符('t')	
%T	ISO8601时间格式(HH:MM:SS)，相当于%H:%M:%S	22:33:44
%u	ISO8601工作日为数字，星期一为1(1-7)	2
%V	ISO8601周编号(01-53)	01

修饰符	描述	示例
%w	工作日为十进制数，周日为0(0-6)	2
%y	年份，最后两位数字（00-99）	18
%Y	年	2018
%%	%符号	%

示例

查询：

```
SELECT formatDateTime(toDate('2010-01-04'), '%g')
```

结果：

```
formatDateTime(toDate('2010-01-04'), '%g')—  
10 |
```

FROM_UNIXTIME

当只有单个整数类型的参数时，它的作用与 `toDateTime` 相同，并返回 `DateTime` 类型。

例如：

```
SELECT FROM_UNIXTIME(423543535)
```

```
FROM_UNIXTIME(423543535)—  
1983-06-04 10:58:55 |
```

当有两个参数时，第一个是整型或 `DateTime`，第二个是常量格式字符串，它的作用与 `formatDateTime` 相同，并返回 `String` 类型。

例如：

```
SELECT FROM_UNIXTIME(1234334543, '%Y-%m-%d %R:%S') AS DateTime
```

```
DateTime—  
2009-02-11 14:42:23 |
```

机器学习函数

evalMLMethod (预测)

使用拟合回归模型的预测请使用 `evalMLMethod` 函数。请参阅 `linearRegression` 中的链接。

随机线性回归

`stochasticLinearRegression`聚合函数使用线性模型和MSE损失函数实现随机梯度下降法。 使用`evalMLMethod`来预测新数据。

请参阅示例和注释[此处](#)。

随机逻辑回归

`stochasticLogisticRegression`聚合函数实现了二元分类问题的随机梯度下降法。 使用`evalMLMethod`来预测新数据。请参阅示例和注释[此处](#)。

条件函数

if

控制条件分支。 与大多数系统不同，ClickHouse始终评估两个表达式 `then` 和 `else`。

语法

```
SELECT if(cond, then, else)
```

如果条件 `cond` 的计算结果为非零值，则返回表达式 `then` 的结果，并且跳过表达式 `else` 的结果（如果存在）。如果 `cond` 为零或 `NULL`，则将跳过 `then` 表达式的结果，并返回 `else` 表达式的结果（如果存在）。

参数

- `cond` - 条件结果可以为零或不为零。 类型是 `UInt8`, `Nullable(UInt8)` 或 `NULL`。
- `then` - 如果满足条件则返回的表达式。
- `else` - 如果不满足条件则返回的表达式。

返回值

该函数执行 `then` 和 `else` 表达式并返回其结果，这取决于条件 `cond` 最终是否为零。

示例

查询:

```
SELECT if(1, plus(2, 2), plus(2, 6))
```

结果:

```
plus(2, 2)  
4 |
```

查询:

```
SELECT if(0, plus(2, 2), plus(2, 6))
```

结果:

```
plus(2, 6)
  8 |
```

- `then` 和 `else` 必须具有最低的通用类型。

示例：

给定表`LEFT_RIGHT`:

```
SELECT *
FROM LEFT_RIGHT
```

left	right
NULL	4
1	3
2	2
3	1
4	NULL

下面的查询比较了 `left` 和 `right` 的值:

```
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

注意：在此示例中未使用'NULL'值，请检查[条件中的NULL值](#)部分。

三元运算符

与 `if` 函数相同。

语法: `cond ? then : else`

如果`cond != 0`则返回`then`，如果`cond = 0`则返回`else`。

- `cond`必须是`UInt8`类型，`then`和`else`必须存在最低的共同类型。
- `then`和`else`可以是NULL

multilf

允许您在查询中更紧凑地编写CASE运算符。

```
multilf(cond_1, then_1, cond_2, then_2...else)
```

参数：

- `cond_N` — 函数返回`then_N`的条件。
- `then_N` — 执行时函数的结果。

- `else` — 如果没有满足任何条件，则为函数的结果。

该函数接受 $2N + 1$ 参数。

返回值

该函数返回值«`then_N`»或«`else`»之一，具体取决于条件`cond_N`。

示例

再次使用表 `LEFT_RIGHT`。

```
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

直接使用条件结果

条件结果始终为 `0`、`1` 或 `NULL`。因此，你可以像这样直接使用条件结果：

```
SELECT left < right AS is_small
FROM LEFT_RIGHT
```

is_small
NULL
1
0
0
NULL

条件中的`NULL`值

当条件中包含 `NULL` 值时，结果也将为 `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

因此，如果类型是 `Nullable`，你应该仔细构造查询。

以下示例说明这一点。

```
SELECT
    left,
    right,
    multilif(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

编码函数

char

返回长度为传递参数数量的字符串，并且每个字节都有对应参数的值。接受数字Numeric类型的多个参数。如果参数的值超出了UInt8数据类型的范围，则将其转换为UInt8，并可能进行舍入和溢出。

语法

```
char(number_1, [number_2, ..., number_n]);
```

参数

- `number_1, number_2, ..., number_n` — 数值参数解释为整数。类型: [Int](#), [Float](#).

返回值

- 给定字节数的字符串。

类型: [String](#)。

示例

查询:

```
SELECT char(104.1, 101, 108.9, 108.9, 111) AS hello
```

结果:

```
hello  
hello |
```

你可以通过传递相应的字节来构造任意编码的字符串。这是UTF-8的示例:

查询:

```
SELECT char(0xD0, 0xBF, 0xD1, 0x80, 0xD0, 0xB8, 0xD0, 0xB2, 0xD0, 0xB5, 0xD1, 0x82) AS hello;
```

结果:

```
hello  
привет
```

查询:

```
SELECT char(0xE4, 0xBD, 0xA0, 0xE5, 0xA5, 0xBD) AS hello;
```

结果:

```
hello  
你好 |
```

hex

接受 `String`，`unsigned integer`，`Date` 或 `DateTime` 类型的参数。返回包含参数的十六进制表示的字符串。使用大写字母 A-F。不使用 `0x` 前缀或 `h` 后缀。对于字符串，所有字节都简单地编码为两个十六进制数字。数字转换为大端（«易阅读»）格式。对于数字，去除其中较旧的零，但仅限整个字节。例如，`hex (1) = '01'`。`Date` 被编码为自 Unix 时间开始以来的天数。`DateTime` 编码为自 Unix 时间开始以来的秒数。

unhex(str)

接受包含任意数量的十六进制数字的字符串，并返回包含相应字节的字符串。支持大写和小写字母 A-F。十六进制数字的数量不必是偶数。如果是奇数，则最后一位数被解释为 00-0F 字节的低位。如果参数字符串包含除十六进制数字以外的任何内容，则返回一些实现定义的结果（不抛出异常）。

如果要将结果转换为数字，可以使用 «reverse» 和 «reinterpretAsType» 函数。

UUIDStringToNum(str)

接受包含 36 个字符的字符串，格式为 «123e4567-e89b-12d3-a456-426655440000»，并将其转化为 `FixedString (16)` 返回。

UUIDNumToString(str)

接受 `FixedString (16)` 值。返回包含 36 个字符的文本格式的字符串。

bitmaskToList(num)

接受一个整数。返回一个字符串，其中包含一组 2 的幂列表，其列表中的所有值相加等于这个整数。列表使用逗号分割，按升序排列。

bitmaskToArray(num)

接受一个整数。返回一个 `UInt64` 类型数组，其中包含一组 2 的幂列表，其列表中的所有值相加等于这个整数。数组中的数字按升序排列。

随机函数

随机函数使用非加密方式生成伪随机数字。

所有随机函数都只接受一个参数或不接受任何参数。

您可以向它传递任何类型的参数，但传递的参数将不会使用在任何随机数生成过程中。

此参数的唯一目的是防止公共子表达式消除，以便在相同的查询中使用相同的随机函数生成不同的随机数。

rand, rand32

返回一个UInt32类型的随机数字，所有UInt32类型的数字被生成的概率均相等。此函数线性同于的方式生成随机数。

rand64

返回一个UInt64类型的随机数字，所有UInt64类型的数字被生成的概率均相等。此函数线性同于的方式生成随机数。

randConstant

返回一个UInt32类型的随机数字，该函数不同之处在于仅为每个数据块参数一个随机数。

高阶函数

-> 运算符, lambda(params, expr) 函数

用于描述一个lambda函数用来传递给其他高阶函数。箭头的左侧有一个形式参数，它可以是一个标识符或多个标识符所组成的元祖。箭头的右侧是一个表达式，在这个表达式中可以使用形式参数列表中的任何一个标识符或表的任何一个列名。

示例: `x -> 2 * x, str -> str != Referer.`

高阶函数只能接受lambda函数作为其参数。

高阶函数可以接受多个参数的lambda函数作为其参数，在这种情况下，高阶函数需要同时传递几个长度相等的数组，这些数组将被传递给lambda参数。

除了'arrayMap'和'arrayFilter'以外的所有其他函数，都可以省略第一个参数（lambda函数）。在这种情况下，默认返回数组元素本身。

arrayMap(func, arr1, ...)

将arr

将从'func'函数的原始应用程序获得的数组返回到'arr'数组中的每个元素。

返回从原始应用程序获得的数组 'func' 函数中的每个元素 'arr' 阵列。

arrayFilter(func, arr1, ...)

返回一个仅包含以下元素的数组 'arr1' 对于哪个 'func' 返回0以外的内容。

示例:

```
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] |
```

arrayCount([func,] arr1, ...)

返回数组arr中非零元素的数量，如果指定了'func'，则通过'func'的返回值确定元素是否为非零元素。

arrayExists([func,] arr1, ...)

返回数组'arr'中是否存在非零元素，如果指定了'func'，则使用'func'的返回值确定元素是否为非零元素。

arrayAll([func,] arr1, ...)

返回数组'arr'中是否存在为零的元素，如果指定了'func'，则使用'func'的返回值确定元素是否为零元素。

arraySum([func,] arr1, ...)

计算arr数组的总和，如果指定了'func'，则通过'func'的返回值计算数组的总和。

arrayFirst(func, arr1, ...)

返回数组中第一个匹配的元素，函数使用'func'匹配所有元素，直到找到第一个匹配的元素。

arrayFirstIndex(func, arr1, ...)

返回数组中第一个匹配的元素的下标索引，函数使用'func'匹配所有元素，直到找到第一个匹配的元素。

arrayCumSum([func,] arr1, ...)

返回源数组部分数据的总和，如果指定了func函数，则使用func的返回值计算总和。

示例：

```
SELECT arrayCumSum([1, 1, 1, 1]) AS res
```

```
res  
[1, 2, 3, 4] |
```

arrayCumSumNonNegative(arr)

与arrayCumSum相同，返回源数组部分数据的总和。不同于arrayCumSum，当返回值包含小于零的值时，该值替换为零，后续计算使用零继续计算。例如：

```
SELECT arrayCumSumNonNegative([1, 1, -4, 1]) AS res
```

```
res  
[1,2,0,1] |
```

arraySort([func,] arr1, ...)

返回升序排序arr1的结果。如果指定了func函数，则排序顺序由func的结果决定。

Schwartzian变换用于提高排序效率。

示例：

```
SELECT arraySort((x, y) -> y, ['hello', 'world'], [2, 1]);
```

```
res  
['world', 'hello'] |
```

请注意，`NULL`和`NaN`在最后（`NaN`在`NULL`之前）。例如：

```
SELECT arraySort([1, nan, 2, NULL, 3, nan, 4, NULL])
```

```
arraySort([1, nan, 2, NULL, 3, nan, 4, NULL]) ->  
[1,2,3,4,nan,NaN,NULL,NULL] |
```

arrayReverseSort([func,] arr1, ...)

返回降序排序`arr1`的结果。如果指定了`func`函数，则排序顺序由`func`的结果决定。

请注意，`NULL`和`NaN`在最后（`NaN`在`NULL`之前）。例如：

```
SELECT arrayReverseSort([1, nan, 2, NULL, 3, nan, 4, NULL])
```

```
arrayReverseSort([1, nan, 2, NULL, 3, nan, 4, NULL]) ->  
[4,3,2,1,nan,NaN,NULL,NULL] |
```

聚合函数

聚合函数如数据库专家预期的方式 **正常** 工作。

ClickHouse还支持：

- **参数聚合函数**，它接受除列之外的其他参数。
- **组合器**，这改变了聚合函数的行为。

空处理

在聚合过程中，所有`NULL`被跳过。

例：

考虑这个表：

x	y
1	2
2	NULL
3	2
3	3
3	NULL

比方说，你需要计算`y`列的总数：

```
SELECT sum(y) FROM t_null_big
```

sum(y)
7

现在你可以使用 `groupArray` 函数用 `y` 列创建一个数组:

```
SELECT groupArray(y) FROM t_null_big
```

groupArray(y)
[2,2,3]

在 `groupArray` 生成的数组中不包括 `NULL`。

Count

计数行数或非空值。

ClickHouse 支持以下 `count` 语法:

- `count(expr)` 或 `COUNT(DISTINCT expr)`。
- `count()` 或 `COUNT(*)`. 该 `count()` 语法是 ClickHouse 特定的。

参数

该函数可以采取:

- 零参数。
- 一个 [表达式](#)。

返回值

- 如果没有参数调用函数，它会计算行数。
- 如果 [表达式](#) 被传递，则该函数计数此表达式返回非 `null` 的次数。如果表达式返回 [可为空](#) 类型的值，`count` 的结果仍然不是 `Nullable`。如果表达式对于所有的行都返回 `NULL`，则该函数返回 `0`。

在这两种情况下，返回值的类型为 [UInt64](#)。

详细信息

ClickHouse 支持 `COUNT(DISTINCT ...)` 语法，这种结构的行为取决于 `count_distinctImplementation` 设置。它定义了用于执行该操作的 `uniq*` 函数。默认值是 `uniqExact` 函数。

`SELECT count() FROM table` 这个查询未被优化，因为表中的条目数没有单独存储。它从表中选择一个小列并计算其值的个数。

示例

示例 1:

```
SELECT count() FROM t
```

```
count()  
5 |
```

示例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 |
```

这个例子表明 `count(DISTINCT num)` 是通过 `count_distinctImplementation` 的设定值 `uniqExact` 函数来执行的。

min

计算最小值。

max

计算最大值。

sum

计算总和。

只适用于数字。

avg

计算算术平均值。

语法

```
avg(x)
```

参数

- `x` — 输入值，必须是 `Integer`, `Float`, 或 `Decimal`。

返回值

- 算术平均值，总是 `Float64` 类型。
- 输入参数 `x` 为空时返回 `Nan`。

示例

查询:

```
SELECT avg(x) FROM values('x UInt8', 0, 1, 2, 3, 4, 5);
```

结果:

avg(x)	2.5
--------	-----

示例

创建一个临时表:

查询:

```
CREATE table test (t UInt8) ENGINE = Memory;
```

获取算术平均值:

查询:

```
SELECT avg(t) FROM test;
```

结果:

avg(x)	nan
--------	-----

any

选择第一个遇到的值。

查询可以以任何顺序执行，甚至每次都以不同的顺序执行，因此此函数的结果是不确定的。

要获得确定的结果，您可以使用 'min' 或 'max' 功能，而不是 'any'.

在某些情况下，可以依靠执行的顺序。这适用于 `SELECT` 来自使用 `ORDER BY` 的子查询的情况。

当一个 `SELECT` 查询具有 `GROUP BY` 子句或至少一个聚合函数，ClickHouse（相对于 MySQL）要求在所有表达式 `SELECT`, `HAVING`, 和 `ORDER BY` 子句可以从键或聚合函数计算。换句话说，从表中选择的每个列必须在键或聚合函数内使用。要获得像 MySQL 这样的行为，您可以将其他列放在 `any` 聚合函数。

stddevPop

结果等于 `[varPop]` ([../../sql-reference/aggregate-functions/reference/varpop.md](#)) 的平方根。

注

该函数使用数值不稳定的算法。如果你需要 **数值稳定性** 在计算中，使用 `stddevPopStable` 函数。它的工作速度较慢，但提供较低的计算错误。

stddevSamp

结果等于 `[varSamp](../../../../sql-reference/aggregate-functions/reference/varsamp.md)` 的平方根。

注

该函数使用数值不稳定的算法。如果你需要 [数值稳定性](#) 在计算中，使用 `stddevSampStable` 函数。它的工作速度较慢，但提供较低的计算错误。

varPop(x)

计算 $\sum((x - \bar{x})^2) / n$ ，这里 n 是样本大小， \bar{x} 是 x 的平均值。

换句话说，计算一组数据的离差。返回 `Float64`。

注

该函数使用数值不稳定的算法。如果你需要 [数值稳定性](#) 在计算中，使用 `varPopStable` 函数。它的工作速度较慢，但提供较低的计算错误。

varSamp

计算 $\sum((x - \bar{x})^2) / (n - 1)$ ，这里 n 是样本大小， \bar{x} 是 x 的平均值。

它表示随机变量的方差的无偏估计，如果传递的值形成其样本。

返回 `Float64`。当 $n \leq 1$ ，返回 $+\infty$ 。

注

该函数使用数值不稳定的算法。如果你需要 [数值稳定性](#) 在计算中，使用 `varSampStable` 函数。它的工作速度较慢，但提供较低的计算错误。

covarPop

语法

```
covarPop(x, y)
```

计算 $\sum((x - \bar{x})(y - \bar{y})) / n$ 的值。

注

该函数使用数值不稳定的算法。如果你需要 [数值稳定性](#) 在计算中，使用 `covarPopStable` 函数。它的工作速度较慢，但提供了较低的计算错误。

聚合函数列表

标准聚合函数:

- `count`
- `min`
- `max`
- `sum`
- `avg`
- `any`
- `stddevPop`
- `stddevSamp`
- `varPop`
- `varSamp`
- `covarPop`
- `covarSamp`

ClickHouse 特有的聚合函数:

- `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`
 - `simpleLinearRegression`
 - `stochasticLinearRegression`
 - `stochasticLogisticRegression`
 - `categoricalInformationValue`
-

covarSamp

语法

```
covarSamp(x, y)
```

计算 $\sum((x - \bar{x})(y - \bar{y})) / (n - 1)$ 的值。

返回 `Float64`。当 `n <= 1`, 返回 $+\infty$ 。

注

该函数使用数值不稳定的算法。如果你需要 [数值稳定性](#) 在计算中，使用 `covarSampStable` 函数。它的工作速度较慢，但提供较低的计算错误。

anyHeavy

选择一个频繁出现的值，使用 [heavy hitters](#) 算法。如果某个值在查询的每个执行线程中出现的情况超过一半，则返回此值。通常情况下，结果是不确定的。

```
anyHeavy(column)
```

参数

- `column` – The column name。

示例

使用 [OnTime](#) 数据集，并选择在 `AirlineID` 列任何频繁出现的值。

查询:

```
SELECT anyHeavy(AirlineID) AS res  
FROM ontime;
```

结果:

```
res  
19690
```

anyLast

选择遇到的最后一个值。

其结果和 [any](#) 函数一样是不确定的。

argMin

语法: `argMin(arg, val)` 或 `argMin(tuple(arg, val))`

计算 `val` 最小值对应的 `arg` 值。如果 `val` 最小值存在几个不同的 `arg` 值，输出遇到的第一个(`arg`)值。

这个函数的 `Tuple` 版本将返回 `val` 最小值对应的 `tuple`。本函数适合和 `SimpleAggregateFunction` 搭配使用。

示例:

输入表:

user	salary
director	5000
manager	3000
worker	1000

查询:

```
SELECT argMin(user, salary), argMin(tuple(user, salary)) FROM salary;
```

结果:

argMin(user, salary)	argMin(tuple(user, salary))
worker	('worker', 1000)

argMax

计算 `val` 最大值对应的 `arg` 值。如果 `val` 最大值存在几个不同的 `arg` 值，输出遇到的第一个值。

这个函数的 Tuple 版本将返回 `val` 最大值对应的元组。本函数适合和 `SimpleAggregateFunction` 搭配使用。

语法

```
argMax(arg, val)
```

或

```
argMax(tuple(arg, val))
```

参数

- `arg` — Argument.
- `val` — Value.

返回值

- `val` 最大值对应的 `arg` 值。

类型: 匹配 `arg` 类型。

对于输入中的元组:

- 元组 `(arg, val)`, 其中 `val` 最大值, `arg` 是对应的值。

类型: **元组**。

示例

输入表:

user	salary
director	5000
manager	3000
worker	1000

查询：

```
SELECT argMax(user, salary), argMax(tuple(user, salary), salary), argMax(tuple(user, salary)) FROM salary;
```

结果：

```
argMax(user, salary)---argMax(tuple(user, salary), salary)---argMax(tuple(user, salary))---  
director      | ('director',5000)           | ('director',5000)           |
```

avgWeighted

计算 **加权算术平均值**。

语法

```
avgWeighted(x, weight)
```

参数

- **x** — 值。
- **weight** — 值的加权。

x 和 **weight** 的类型必须是

整数, 或

浮点数, 或

定点数,

但是可以不一样。

返回值

- **Nan**。如果所有的权重都等于0 或所提供的权重参数是空。
- 加权平均值。其他。

类型：总是**Float64**.

示例

查询：

```
SELECT avgWeighted(x, w)  
FROM values('x Int8, w Int8', (4, 1), (1, 0), (10, 2))
```

结果：

```
avgWeighted(x, weight)---  
8 |
```

示例

查询:

```
SELECT avgWeighted(x, w)
FROM values('x Int8, w Int8', (0, 0), (1, 0), (10, 0))
```

结果:

```
avgWeighted(x, weight)─
nan |
```

示例

查询:

```
CREATE table test (t UInt8) ENGINE = Memory;
SELECT avgWeighted(t) FROM test
```

结果:

```
avgWeighted(x, weight)─
nan |
```

corr

语法

```
`corr(x, y)`
```

计算 Pearson 相关系数: $\Sigma((x - \bar{x})(y - \bar{y})) / \sqrt{\Sigma((x - \bar{x})^2) * \Sigma((y - \bar{y})^2)}$ 。

注

该函数使用数值不稳定的算法。如果你需要 [数值稳定性](#) 在计算中，使用 `corrStable` 函数。它的工作速度较慢，但提供较低的计算错误。

topK

返回指定列中近似最常见值的数组。生成的数组按值的近似频率降序排序（而不是值本身）。

实现了[过滤节省空间](#)算法，使用基于 `reduce-and-combine` 的算法，借鉴[并行节省空间](#)。

语法

```
topK(N)(x)
```

此函数不提供保证的结果。在某些情况下，可能会发生错误，并且可能会返回不是最高频的值。

我们建议使用 `N < 10` 值，`N` 值越大，性能越低。最大值 `N = 65536`。

参数

- `N` — 要返回的元素数。

如果省略该参数，则使用默认值 `10`。

参数

- `x` - (要计算频次的)值。

示例

就拿 `OnTime` 数据集来说，选择 `AirlineID` 列中出现最频繁的三个。

```
SELECT topK(3)(AirlineID) AS res  
FROM ontime
```

```
res  
[19393,19790,19805] |
```

topKWeighted

类似于 `topK` 但需要一个整数类型的附加参数 - `weight`。每个输入都被记入 `weight` 次频率计算。

语法

```
topKWeighted(N)(x, weight)
```

参数

- `N` — 要返回的元素数。

参数

- `x` - (要计算频次的)值。

- `weight` — 权重。`UInt8` 类型。

返回值

返回具有最大近似权重总和的值数组。

示例

查询：

```
SELECT topKWeighted(10)(number, number) FROM numbers(1000)
```

结果：

```
topKWeighted(10)(number, number)  
[999,998,997,996,995,994,993,992,991,990] |
```

groupArray

语法

```
groupArray(x)
或
groupArray(max_size)(x)
```

创建参数值的数组。

值可以按任何（不确定）顺序添加到数组中。

第二个版本（带有 `max_size` 参数）将结果数组的大小限制为 `max_size` 个元素。

例如，`groupArray(1)(x)` 相当于 `[any(x)]`。

在某些情况下，您仍然可以依赖执行顺序。这适用于SELECT(查询)来自使用了 ORDER BY 子查询的情况。

groupUniqArray

语法

```
groupUniqArray(x)
或
groupUniqArray(max_size)(x)
```

从不同的参数值创建一个数组。内存消耗和 `uniqExact` 函数是一样的。

第二个版本（带有 `max_size` 参数）将结果数组的大小限制为 `max_size` 个元素。

例如，`groupUniqArray(1)(x)` 相当于 `[any(x)]`。

groupArrayInsertAt

在指定位置向数组中插入一个值。

语法

```
groupArrayInsertAt(default_x, size)(x, pos);
```

如果在一个查询中将多个值插入到同一位置，则该函数的行为方式如下：

- 如果在单个线程中执行查询，则使用第一个插入的值。
- 如果在多个线程中执行查询，则结果值是未确定的插入值之一。

参数

- `x` — 要插入的值。生成所支持的数据类型(数据)的表达式。
- `pos` — 指定元素 `x` 将被插入的位置。数组中的索引编号从零开始。`UInt32`。
- `default_x` — 在空位置替换的默认值。可选参数。生成 `x` 数据类型(数据)的表达式。如果 `default_x` 未定义，则默认值被使用。
- `size` — 结果数组的长度。可选参数。如果使用该参数，必须指定默认值 `default_x`。`UInt32`。

返回值

- 具有插入值的数组。

类型: **阵列**。

示例

查询:

```
SELECT groupArrayInsertAt(toString(number), number * 2) FROM numbers(5);
```

结果:

```
groupArrayInsertAt(toString(number), multiply(number, 2))—  
['0','1','2','3','4'] |
```

查询:

```
SELECT groupArrayInsertAt('-')(toString(number), number * 2) FROM numbers(5);
```

结果:

```
groupArrayInsertAt('')(toString(number), multiply(number, 2))—  
['0','-','1','-','2','-','3','-','4'] |
```

查询:

```
SELECT groupArrayInsertAt('-', 5)(toString(number), number * 2) FROM numbers(5);
```

结果:

```
groupArrayInsertAt('', 5)(toString(number), multiply(number, 2))—  
['0','1','2'] |
```

在一个位置多线程插入数据。

查询:

```
SELECT groupArrayInsertAt(number, 0) FROM numbers_mt(10) SETTINGS max_block_size = 1;
```

作为这个查询的结果，你会得到 [0,9] 范围的随机整数。例如:

```
groupArrayInsertAt(number, 0)—  
[7] |
```

groupArrayMovingSum

计算输入值的移动和。

语法

```
groupArrayMovingSum(numbers_for_summing)
groupArrayMovingSum(window_size)(numbers_for_summing)
```

该函数可以将窗口大小作为参数。如果未指定，则该函数的窗口大小等于列中的行数。

参数

- `numbers_for_summing` — 表达式 生成数值数据类型值。
- `window_size` — 窗口大小。

返回值

- 与输入数据大小相同的数组。
对于输入数据类型是 `Decimal` 数组元素类型是 `Decimal128`。
对于其他的数值类型，获取其对应的 `NearestFieldType`。

示例

样表：

```
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

查询：

```
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

计算输入值的移动平均值。

语法

```
groupArrayMovingAvg(numbers_for_summing)
groupArrayMovingAvg(window_size)(numbers_for_summing)
```

该函数可以将窗口大小作为参数。如果未指定，则该函数的窗口大小等于列中的行数。

参数

- `numbers_for_summing` — 表达式 生成数值数据类型值。
- `window_size` — 窗口大小。

返回值

- 与输入数据大小相同的数组。

对于输入数据类型是 `Integer`,

和 `floating-point`,

对应的返回值类型是 `Float64`。

对于输入数据类型是 `Decimal` 返回值类型是 `Decimal128`。

该函数对于 `Decimal128` 使用 `四舍五入到零`，它截断无意义的小数位来保证结果的数据类型。

示例

样表 t:

```
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

查询:

```
SELECT
    groupArrayMovingAvg(int) AS I,
    groupArrayMovingAvg(float) AS F,
    groupArrayMovingAvg(dec) AS D
FROM t
```

D	F
[0.25,0.75,1.75,3.5]	[0.2750000059604645,0.8250000178813934,1.9250000417232513,3.8499999940395355]
[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
```

F	D
[0.5,1.5,3,5.5]	[0.550000011920929,1.6500000357627869,3.3000000715255737,6.049999952316284]
[0.55,1.65,3.30,6.08]	

groupArraySample

构建一个参数值的采样数组。

结果数组的大小限制为 `max_size` 个元素。参数值被随机选择并添加到数组中。

语法

```
groupArraySample(max_size[, seed])(x)
```

参数

- `max_size` — 结果数组的最大长度。`UInt64`。
- `seed` — 随机数发生器的种子。可选。`UInt64`。默认值: `123456`。
- `x` — 参数 (列名 或者 表达式)。

返回值

- 随机选取参数 `x` (的值)组成的数组。

类型: `Array`.

示例

样表 `colors`:

id	color
1	red
2	blue
3	green
4	white
5	orange

使用列名做参数查询:

```
SELECT groupArraySample(3)(color) as newcolors FROM colors;
```

结果：

```
newcolors  
['white','blue','green'] |
```

使用列名和不同的(随机数)种子查询：

```
SELECT groupArraySample(3, 987654321)(color) as newcolors FROM colors;
```

结果：

```
newcolors  
['red','orange','green'] |
```

使用表达式做参数查询：

```
SELECT groupArraySample(3)(concat('light-', color)) as newcolors FROM colors;
```

结果：

```
newcolors  
['light-blue','light-orange','light-green'] |
```

groupBitAnd

对于数字序列按位应用 **AND**。

语法

```
groupBitAnd(expr)
```

参数

expr – 结果为 **UInt*** 类型的表达式。

返回值

UInt* 类型的值。

示例

测试数据：

```
binary    decimal
00101100 = 44
00011100 = 28
00001101 = 13
01010101 = 85
```

查询:

```
SELECT groupBitAnd(num) FROM t
```

`num` 是包含测试数据的列。

结果:

```
binary  decimal  
00000100 = 4
```

groupBitOr

对于数字序列按位应用 OR。

语法

```
groupBitOr(expr)
```

参数

`expr` – 结果为 `UInt*` 类型的表达式。

返回值

`UInt*` 类型的值。

示例

测试数据::

```
binary  decimal  
00101100 = 44  
00011100 = 28  
00001101 = 13  
01010101 = 85
```

查询:

```
SELECT groupBitOr(num) FROM t
```

`num` 是包含测试数据的列。

结果:

```
binary  decimal  
01111101 = 125
```

groupBitXor

对于数字序列按位应用 XOR。

语法

```
groupBitXor(expr)
```

参数

`expr` – 结果为 `UInt*` 类型的表达式。

返回值

`UInt*` 类型的值。

示例

测试数据:

```
binary  decimal
00101100 = 44
00011100 = 28
00001101 = 13
01010101 = 85
```

查询:

```
SELECT groupBitXor(num) FROM t
```

`num` 是包含测试数据的列。

结果:

```
binary  decimal
01101000 = 104
```

groupBitmap

从无符号整数列进行位图或聚合计算，返回 `UInt64` 类型的基数，如果添加后缀 `State`，则返回位图对象。

语法

```
groupBitmap(expr)
```

参数

`expr` – 结果为 `UInt*` 类型的表达式。

返回值

`UInt64` 类型的值。

示例

测试数据:

```
UserID
1
1
2
3
```

查询：

```
SELECT groupBitmap(UserID) as num FROM t
```

结果：

```
num  
3
```

groupBitmapAnd

计算位图列的 AND，返回 UInt64 类型的基数，如果添加后缀 State，则返回 位图对象。

语法

```
groupBitmapAnd(expr)
```

参数

expr – 结果为 AggregateFunction(groupBitmap, UInt*) 类型的表达式。

返回值

UInt64 类型的值。

示例

```
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

计算位图列的 OR，返回 UInt64 类型的基数，如果添加后缀 State，则返回 位图对象。

语法

```
groupBitmapOr(expr)
```

参数

`expr` – 结果为 `AggregateFunction(groupBitmap, UInt*)` 类型的表达式。

返回值

`UInt64` 类型的值。

示例

```
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

计算位图列的 `XOR`，返回 `UInt64` 类型的基数，如果添加后缀 `State`，则返回 [位图对象](#)。

语法

```
groupBitmapXor(expr)
```

参数

`expr` – 结果为 `AggregateFunction(groupBitmap, UInt*)` 类型的表达式。

返回值

`UInt64` 类型的值。

示例

```

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]
```

sumWithOverflow

使用与输入参数相同的数据类型计算结果的数字总和。如果总和超过此数据类型的最大值，则使用溢出进行计算。

只适用于数字。

deltaSum

计算连续行之间的差值和。如果差值为负，则忽略。

语法

```
deltaSum(value)
```

参数

- `value` — 必须是 整型 或者 浮点型 。

返回值

- `Integer or Float` 型的算术差值和。

示例

查询:

```
SELECT deltaSum(arrayJoin([1, 2, 3]));
```

结果:

```
└deltaSum(arrayJoin([1, 2, 3]))┘
  2 |
```

查询:

```
SELECT deltaSum(arrayJoin([1, 2, 3, 0, 3, 4, 2, 3]));
```

结果:

```
deltaSum(arrayJoin([1, 2, 3, 0, 3, 4, 2, 3]))—  
    7 |
```

查询:

```
SELECT deltaSum(arrayJoin([2.25, 3, 4.5]));
```

结果:

```
deltaSum(arrayJoin([2.25, 3, 4.5]))—  
    2.25 |
```

参见

- [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

语法

```
sumMap(key, value)
或
sumMap(Tuple(key, value))
```

根据 `key` 数组中指定的键对 `value` 数组进行求和。

传递 `key` 和 `value` 数组与传递 `key` 和 `value` 的两个数组是同义的。

要总计的每一行的 `key` 和 `value` (数组)元素的数量必须相同。

返回两个数组组成的一个元组: 排好序的 `key` 和对应 `key` 的 `value` 之和。

示例:

```
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]

minMap

语法

```
minMap(key, value)
或
minMap(Tuple(key, value))
```

根据 `key` 数组中指定的键对 `value` 数组计算最小值。

传递 `key` 和 `value` 数组的元组与传递 `key` 和 `value` 的两个数组是同义的。

要总计的每一行的 `key` 和 `value` (数组)元素的数量必须相同。

返回两个数组组成的元组: 排好序的 `key` 和对应 `key` 的 `value` 计算值(最小值)。

示例

```
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

语法

```
maxMap(key, value)
或
maxMap(Tuple(key, value))
```

根据 `key` 数组中指定的键对 `value` 数组计算最大值。

传递 `key` 和 `value` 数组的元组与传递 `key` 和 `value` 的两个数组是同义的。

要总计的每一行的 `key` 和 `value` (数组)元素的数量必须相同。

返回两个数组组成的元组: 排好序的 `key` 和对应 `key` 的 `value` 计算值(最大值)。

示例:

```
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.

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)
0.999999999999999	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

initializeAggregation

初始化你输入行的聚合。用于后缀是 `State` 的函数。

用它来测试或处理 `AggregateFunction` 和 `AggregationMergeTree` 类型的列。

语法

```
initializeAggregation (aggregate_function, column_1, column_2)
```

参数

- `aggregate_function` — 聚合函数名。这个函数的状态 — 正创建的。`String`。
- `column_n` — 将其转换为函数的参数的列。`String`。

返回值

返回输入行的聚合结果。返回类型将与 `initializeAggregation` 用作第一个参数的函数的返回类型相同。

例如，对于后缀为 `State` 的函数，返回类型将是 `AggregateFunction`。

示例

查询：

```
SELECT uniqMerge(state) FROM (SELECT initializeAggregation('uniqState', number % 3) AS state FROM system.numbers LIMIT 10000);
```

结果：

```
└─uniqMerge(state)─  
   | 3 |  
   └─────────┘
```

skewPop

计算给定序列的 [偏度] (<https://en.wikipedia.org/wiki/Skewness>)。

语法

```
skewPop(expr)
```

参数

`expr` — 表达式 返回一个数字。

返回值

给定分布的偏度。类型 — `Float64`

示例

```
SELECT skewPop(value) FROM series_with_value_column;
```

skewSamp

计算给定序列的 [样本偏度] (<https://en.wikipedia.org/wiki/Skewness>)。

如果传递的值形成其样本，它代表了一个随机变量的偏度的无偏估计。

语法

```
skewSamp(expr)
```

参数

`expr` — 表达式 返回一个数字。

返回值

给定分布的偏度。类型 — `Float64`。如果 `n <= 1` (`n` 样本的大小)，函数返回 `nan`。

示例

```
SELECT skewSamp(value) FROM series_with_value_column;
```

kurtPop

计算给定序列的 **峰度**。

语法

```
kurtPop(expr)
```

参数

expr — 结果为数字的 **表达式**。

返回值

给定分布的峰度。类型 — **Float64**

示例

```
SELECT kurtPop(value) FROM series_with_value_column;
```


kurtSamp {#kurtsamp}

计算给定序列的 [峰度样本](<https://en.wikipedia.org/wiki/Kurtosis>)。

它表示随机变量峰度的无偏估计，如果传递的值形成其样本。

****语法****

```
``` sql  
kurtSamp(expr)
```

## 参数

**expr** — 结果为数字的 **表达式**。

## 返回值

给定序列的峰度。类型 — **Float64**。如果 **n <= 1** (**n** 是样本的大小)，则该函数返回 **nan**。

## 示例

```
SELECT kurtSamp(value) FROM series_with_value_column;
```

# uniq

计算参数的不同值的近似数量。

## 语法

```
uniq(x[, ...])
```

## 参数

该函数采用可变数量的参数。 参数可以是 `Tuple`, `Array`, `Date`, `DateTime`, `String`, 或数字类型。

## 返回值

- `UInt64` 类型数值。

## 实现细节

功能:

- 计算聚合中所有参数的哈希值，然后在计算中使用它。
- 使用自适应采样算法。对于计算状态，该函数使用最多 65536 个元素哈希值的样本。

这个算法是非常精确的，并且对于 CPU 来说非常高效。如果查询包含一些这样的函数，那和其他聚合函数相比 `uniq` 将是几乎一样快。

- 确定性地提供结果（它不依赖于查询处理顺序）。

我们建议在几乎所有情况下使用此功能。

## 参见

- [uniqCombined](#)
- [uniqCombined64](#)
- [uniqHLL12](#)
- [uniqExact](#)

# uniqExact

计算不同参数值的准确数目。

## 语法

```
uniqExact(x[, ...])
```

如果你绝对需要一个确切的结果，使用 `uniqExact` 函数。否则使用 `uniq` 函数。

`uniqExact` 函数比 `uniq` 使用更多的内存，因为状态的大小随着不同值的数量的增加而无界增长。

## 参数

该函数采用可变数量的参数。参数可以是 `Tuple`, `Array`, `Date`, `DateTime`, `String`, 或数字类型。

## 参见

- [uniq](#)
- [uniqCombined](#)
- [uniqHLL12](#)

# uniqCombined

计算不同参数值的近似数量。

## 语法

```
uniqCombined(HLL_precision)(x[, ...])
```

该 `uniqCombined` 函数是计算不同值数量的不错选择。

## 参数

该函数采用可变数量的参数。参数可以是 `Tuple`, `Array`, `Date`, `DateTime`, `String`, 或数字类型。

`HLL_precision` 是以2为底的单元格数的对数 **HyperLogLog**。可选，您可以将该函数用作 `uniqCombined(x[, ...])`。

`HLL_precision` 的默认值是17，这是有效的96KiB的空间（ $2^{17}$ 个单元，每个6比特）。

## 返回值

- 一个 **UInt64** 类型的数字。

## 实现细节

功能：

- 为聚合中的所有参数计算哈希（`String` 类型用64位哈希，其他32位），然后在计算中使用它。

- 使用三种算法的组合：数组、哈希表和包含错误修正表的 **HyperLogLog**。

少量的不同的值，使用数组。值再多一些，使用哈希表。对于大量的数据来说，使用 **HyperLogLog**，**HyperLogLog** 占用一个固定的内存空间。

- 确定性地提供结果（它不依赖于查询处理顺序）。

## 注

由于它对非 `String` 类型使用32位哈希，对于基数显著大于 `UINT_MAX`，结果将有非常高的误差(误差将在几百亿不同值之后迅速提高)，因此这种情况，你应该使用 **uniqCombined64**

相比于 `uniq` 函数，该 `uniqCombined`:

- 消耗内存要少几倍。
- 计算精度高出几倍。
- 通常具有略低的性能。在某些情况下，`uniqCombined` 可以表现得比 `uniq` 好，例如，使用通过网络传输大量聚合状态的分布式查询。

## 参见

- [uniq](#)
- [uniqCombined64](#)
- [uniqHLL12](#)
- [uniqExact](#)

## uniqCombined64

和 `uniqCombined`一样，但对于所有数据类型使用64位哈希。

# uniqHLL12

计算不同参数值的近似数量，使用 [HyperLogLog](#) 算法。

语法

```
uniqHLL12(x[, ...])
```

参数

该函数采用可变数量的参数。 参数可以是 `Tuple`, `Array`, `Date`, `DateTime`, `String`，或数字类型。

返回值

返回值

- 一个 [UInt64](#)类型的数字。

实现细节

功能:

- 计算聚合中所有参数的哈希值，然后在计算中使用它。
- 使用 [HyperLogLog](#) 算法来近似不同参数值的数量。

使用 $2^{12}$ 个5比特单元。 状态的大小略大于2.5KB。 对于小数据集（<10K元素），结果不是很准确（误差高达10%）。但是，对于高基数数据集（10K-100M），结果相当准确，最大误差约为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).

- 提供确定结果（它不依赖于查询处理顺序）。

我们不建议使用此函数。 在大多数情况下，使用 [uniq](#) 或 [uniqCombined](#) 函数。

参见

- [uniq](#)
- [uniqCombined](#)
- [uniqExact](#)

# 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

计算数字序列的近似分位数。

此函数应用[水塘抽样][reservoir sampling] ([https://en.wikipedia.org/wiki/Reservoir\\_sampling](https://en.wikipedia.org/wiki/Reservoir_sampling))，使用高达8192的水塘大小和随机数发生器采样。

结果是不确定的。要获得精确的分位数，使用 [quantileExact](#) 函数。

当在一个查询中使用多个不同层次的 `quantile*` 时，内部状态不会被组合（即查询的工作效率低于组合情况）。在这种情况下，使用 [quantiles](#) 函数。

## 语法

```
quantile(level)(expr)
```

别名: `median`。

## 参数

- `level` — 分位数层次。可选参数。从0到1的一个float类型的常量。我们推荐 `level` 值的范围为 [0.01, 0.99]。默认值：0.5。当 `level=0.5` 时，该函数计算 [中位数](#)。
- `expr` — 求值表达式，类型为数值类型[data types](#), [Date](#) 或 [DateTime](#)。

## 返回值

- 指定层次的分位数。

类型:

- [Float64](#) 用于数字数据类型输入。
- [Date](#) 如果输入值是 `Date` 类型。
- [DateTime](#) 如果输入值是 `DateTime` 类型。

## 示例

输入表:

val
1
1
2
3

查询:

```
SELECT quantile(val) FROM t
```

结果:

quantile(val)
1.5

参见

- [中位数](#)
- [分位数](#)

## quantiles

语法

```
quantiles(level1, level2, ...)(x)
```

所有分位数函数(quantile)也有相应的分位数(quantiles)函数: `quantiles`, `quantilesDeterministic`, `quantilesTiming`, `quantilesTimingWeighted`, `quantilesExact`, `quantilesExactWeighted`, `quantilesTDigest`。这些函数一次计算所列的级别的所有分位数，并返回结果值的数组。

## quantileExact

准确计算数字序列的分位数。

为了准确计算，所有输入的数据被合并为一个数组，并且部分的排序。因此该函数需要  $O(n)$  的内存， $n$  为输入数据的个数。但是对于少量数据来说，该函数还是非常有效的。

当在一个查询中使用多个不同层次的 `quantile*` 时，内部状态不会被组合（即查询的工作效率低于组合情况）。在这种情况下，使用 `quantiles` 函数。

语法

```
quantileExact(level)(expr)
```

别名: `medianExact`。

参数

- `level` — 分位数层次。可选参数。从0到1的一个float类型的常量。我们推荐 `level` 值的范围为 [0.01, 0.99]。默认值：0.5。当 `level=0.5` 时，该函数计算中位数。
- `expr` — 求值表达式，类型为数值类型[data types](#), [Date](#) 或 [DateTime](#)。

## 返回值

- 指定层次的分位数。

类型：

- **Float64** 对于数字数据类型输入。
- **日期** 如果输入值具有 **Date** 类型。
- **日期时间** 如果输入值具有 **DateTime** 类型。

## 示例

查询：

```
SELECT quantileExact(number) FROM numbers(10)
```

结果：

```
quantileExact(number)─
5 |
```

## quantileExactLow

和 `quantileExact` 相似，准确计算数字序列的分位数。

为了准确计算，所有输入的数据被合并为一个数组，并且全排序。这排序算法的复杂度是  $O(N \cdot \log(N))$ ，其中  $N = \text{std}::\text{distance}(\text{first}, \text{last})$  比较。

返回值取决于分位数级别和所选取的元素数量，即如果级别是 `0.5`，函数返回偶数元素的低位中位数，奇数元素的中位数。中位数计算类似于 python 中使用的 `median_low` 的实现。

对于所有其他级别，返回 `level * size_of_array` 值所对应的索引的元素值。

例如：

```
SELECT quantileExactLow(0.1)(number) FROM numbers(10)
```

```
quantileExactLow(0.1)(number)─
1 |
```

当在一个查询中使用多个不同层次的 `quantile*` 时，内部状态不会被组合（即查询的工作效率低于组合情况）。在这种情况下，使用 `quantiles` 函数。

## 语法

```
quantileExactLow(level)(expr)
```

别名：`medianExactLow`。

## 参数

- `level` — 分位数层次。可选参数。从 `0` 到 `1` 的一个 `float` 类型的常量。我们推荐 `level` 值的范围为 `[0.01, 0.99]`。默认值：`0.5`。当 `level=0.5` 时，该函数计算 中位数。
- `expr` — — 求值表达式，类型为数值类型 `data types`, `Date` 或 `DateTime`。

## 返回值

- 指定层次的分位数。

类型：

- **Float64** 用于数字数据类型输入。
- **Date** 如果输入值是 **Date** 类型。
- **DateTime** 如果输入值是 **DateTime** 类型。

## 示例

查询：

```
SELECT quantileExactLow(number) FROM numbers(10)
```

结果：

```
quantileExactLow(number)
4 |
```

## quantileExactHigh

和 `quantileExact` 相似，准确计算数字序列的 **分位数**。

为了准确计算，所有输入的数据被合并为一个数组，并且全排序。这排序 **算法** 的复杂度是  $O(N \cdot \log(N))$ ，其中  $N = \text{std}::\text{distance}(\text{first}, \text{last})$  比较。

返回值取决于分位数级别和所选取的元素数量，即如果级别是 `0.5`，函数返回偶数元素的低位中位数，奇数元素的中位数。中位数计算类似于 `python` 中使用的 `median_high` 的实现。

对于所有其他级别，返回 `level * size_of_array` 值所对应的索引的元素值。

这个实现与当前的 `quantileExact` 实现完全相似。

当在一个查询中使用多个不同层次的 `quantile*` 时，内部状态不会被组合（即查询的工作效率低于组合情况）。在这种情况下，使用 `quantiles` 函数。

## 语法

```
quantileExactHigh(level)(expr)
```

别名： `medianExactHigh`。

## 参数

- `level` — 分位数层次。可选参数。从 `0` 到 `1` 的一个 `float` 类型的常量。我们推荐 `level` 值的范围为 `[0.01, 0.99]`。默认值：`0.5`。当 `level=0.5` 时，该函数计算 **中位数**。
- `expr` — 求值表达式，类型为数值类型 **data types**, **Date** 或 **DateTime**。

## 返回值

- 指定层次的分位数。

类型：

- **Float64** 用于数字数据类型输入。
- **Date** 如果输入值是 **Date** 类型。
- **DateTime** 如果输入值是 **DateTime** 类型。

#### 示例

查询:

```
SELECT quantileExactHigh(number) FROM numbers(10)
```

结果:

```
quantileExactHigh(number)
5 |
```

#### 参见

- [中位数](#)
- [分位数](#)

## quantileExactWeighted

考虑到每个元素的权重，然后准确计算数值序列的**分位数**。

为了准确计算，所有输入的数据被合并为一个数组，并且部分的排序。每个输入值需要根据 **weight** 计算求和。该算法使用哈希表。正因为如此，在数据重复较多的时候使用的内存是少于**quantileExact**的。您可以使用此函数代替 **quantileExact** 并指定 **weight** 为 1。

当在一个查询中使用多个不同层次的 **quantile\*** 时，内部状态不会被组合（即查询的工作效率低于组合情况）。在这种情况下，使用 **quantiles** 函数。

#### 语法

```
quantileExactWeighted(level)(expr, weight)
```

别名: **medianExactWeighted**。

#### 参数

- **level** — 分位数层次。可选参数。从0到1的一个**float**类型的常量。我们推荐 **level** 值的范围为 [0.01, 0.99]。默认值：0.5。当 **level=0.5** 时，该函数计算 **中位数**。
- **expr** — 求值表达式，类型为数值类型**data types**, **Date** 或 **DateTime**。
- **weight** — 权重序列。权重是一个数据出现的数值。

#### 返回值

- 指定层次的分位数。

类型:

- **Float64** 对于数字数据类型输入。
- **日期** 如果输入值具有 **Date** 类型。
- **日期时间** 如果输入值具有 **DateTime** 类型。

## 示例

输入表:

n	val
0	3
1	2
2	1
5	4

查询:

```
SELECT quantileExactWeighted(n, val) FROM t
```

结果:

```
quantileExactWeighted(n, val)
1 |
```

## 参见

- [中位数](#)
- [分位数](#)

## quantileTiming

使用确定的精度计算数字数据序列的[分位数](#)。

结果是确定性的（它不依赖于查询处理顺序）。该函数针对描述加载网页时间或后端响应时间等分布的序列进行了优化。

当在一个查询中使用多个不同层次的 `quantile*` 时，内部状态不会被组合（即查询的工作效率低于组合情况）。在这种情况下，使用 `quantiles` 函数。

### 语法

```
quantileTiming(level)(expr)
```

别名: `medianTiming`。

### 参数

- `level` — 分位数层次。可选参数。从0到1的一个float类型的常量。我们推荐 `level` 值的范围为 [0.01, 0.99]。默认值：0.5。当 `level=0.5` 时，该函数计算 [中位数](#)。
- `expr` — 求值表达式 返回 [Float\\*](#) 类型数值。
  - 如果输入负值，那结果是不可预期的。
  - 如果输入值大于30000（页面加载时间大于30s），那我们假设为30000。

### 精度

计算是准确的，如果：

- 值的总数不超过5670。

- 总数值超过5670，但页面加载时间小于1024ms。

否则，计算结果将四舍五入到16毫秒的最接近倍数。

## 注

对于计算页面加载时间分位数，此函数比**quantile**更有效和准确。

## 返回值

- 指定层次的分位数。

类型: `Float32`。

## 注

如果没有值传递给函数（当使用 `quantileTimingIf`），`Nan` 被返回。这样做的目的是将这些案例与导致零的案例区分开来。参见 `ORDER BY clause` 对于 `Nan` 值排序注意事项。

## 示例

输入表:

response_time
72
112
126
145
104
242
313
168
108

查询:

```
SELECT quantileTiming(response_time) FROM t
```

结果:

quantileTiming(response_time)
126

## 参见

- [中位数](#)
- [分位数](#)

# quantileTimingWeighted

根据每个序列成员的权重，使用确定的精度计算数字序列的分位数。

结果是确定性的（它不依赖于查询处理顺序）。该函数针对描述加载网页时间或后端响应时间等分布的序列进行了优化。

当在一个查询中使用多个不同层次的 `quantile*` 时，内部状态不会被组合（即查询的工作效率低于组合情况）。在这种情况下，使用 `quantiles` 功能。

## 语法

```
quantileTimingWeighted(level)(expr, weight)
```

别名: `medianTimingWeighted`。

## 参数

- `level` — 分位数层次。可选参数。从 0 到 1 的一个 `float` 类型的常量。我们推荐 `level` 值的范围为 [0.01, 0.99]。默认值：0.5。当 `level=0.5` 时，该函数计算 **中位数**。
- `expr` — 求值 **表达式** 返回 `Float*` 类型数值。
  - 如果输入负值，那结果是不可预期的。
  - 如果输入值大于 30000（页面加载时间大于 30s），那我们假设为 30000。
- `weight` — 权重序列。权重是一个数据出现的数值。

## 精度

计算是准确的，如果：

- 值的总数不超过 5670。
- 总数值超过 5670，但页面加载时间小于 1024ms。

否则，计算结果将四舍五入到 16 毫秒的最接近倍数。

## 注

对于计算页面加载时间分位数，此函数比 `quantile` 更有效和准确。

## 返回值

- 指定层次的分位数。

类型: `Float32`。

## 注

如果没有值传递给函数（当使用 `quantileTimingIf`），`Nan` 被返回。这样做的目的是将这些案例与导致零的案例区分开来。

参见 `ORDER BY clause` 对于 `Nan` 值排序注意事项。

## 示例

输入表:

response_time	weight
68	1
104	2
112	3
126	2
138	1
162	1

查询:

```
SELECT quantileTimingWeighted(response_time, weight) FROM t
```

结果:

```
quantileTimingWeighted(response_time, weight)
112 |
```

## quantilesTimingWeighted

类似于 `quantileTimingWeighted`，但接受多个分位数层次参数，并返回一个由这些分位数值组成的数组。

示例

输入表:

response_time	weight
68	1
104	2
112	3
126	2
138	1
162	1

查询:

```
SELECT quantilesTimingWeighted(0.5, 0.99)(response_time, weight) FROM t
```

结果:

```
quantilesTimingWeighted(0.5, 0.99)(response_time, weight)
[112,162] |
```

参见

- [中位数](#)
- [分位数](#)

## quantileDeterministic

计算数字序列的近似分位数。

此功能适用 [水塘抽样](#)，使用储存器最大到8192和随机数发生器进行采样。结果是非确定性的。要获得精确的分位数，请使用 [quantileExact](#) 功能。

当在一个查询中使用多个不同层次的 `quantile*` 时，内部状态不会被组合（即查询的工作效率低于组合情况）。在这种情况下，使用 [quantiles](#) 功能。

## 语法

```
quantileDeterministic(level)(expr, determinator)
```

别名: `medianDeterministic`。

## 参数

- `level` — 分位数层次。可选参数。从0到1的一个float类型的常量。我们推荐 `level` 值的范围为 [0.01, 0.99]。默认值：0.5。当 `level=0.5` 时，该函数计算 [中位数](#)。
- `expr` — 求值表达式，类型为数值类型[data types](#), [Date](#) 或 [DateTime](#)。
- `determinator` — 一个数字，其hash被用来代替在水塘抽样中随机生成的数字，这样可以保证取样的确定性。你可以使用用户ID或者事件ID等任何正数，但是如果相同的 `determinator` 出现多次，那结果很可能不正确。

## 返回值

- 指定层次的近似分位数。

## 类型:

- [Float64](#) 用于数字数据类型输入。
- [Date](#) 如果输入值是 [Date](#) 类型。
- [DateTime](#) 如果输入值是 [DateTime](#) 类型。

## 示例

输入表:

val
1
1
2
3

查询:

```
SELECT quantileDeterministic(val, 1) FROM t
```

结果:

```
quantileDeterministic(val, 1)
1.5 |
```

## 参见

- [中位数](#)
- [分位数](#)

# quantileTDigest

使用 **t-digest** 算法计算数字序列近似分位数。

最大误差为 1%。内存消耗为  $\log(n)$ ，这里  $n$  是值的个数。结果取决于运行查询的顺序，并且是不确定的。

该函数的性能低于 **quantile** 或 **quantileTiming** 的性能。从状态大小和精度的比值来看，这个函数比 **quantile** 更优秀。

当在一个查询中使用多个不同层次的 **quantile\*** 时，内部状态不会被组合（即查询的工作效率低于组合情况）。在这种情况下，使用 **quantiles** 函数。

## 语法

```
quantileTDigest(level)(expr)
```

别名: `medianTDigest`。

## 参数

- **level** — 分位数层次。可选参数。从 0 到 1 的一个 **float** 类型的常量。我们推荐 **level** 值的范围为 [0.01, 0.99]。默认值：0.5。当 **level=0.5** 时，该函数计算 **中位数**。
- **expr** — 求值表达式，类型为数值类型 **data types**, **Date** 或 **DateTime**。

## 返回值

- 指定层次的分位数。

类型：

- **Float64** 用于数字数据类型输入。
- **Date** 如果输入值是 **Date** 类型。
- **DateTime** 如果输入值是 **DateTime** 类型。

## 示例

查询：

```
SELECT quantileTDigest(number) FROM numbers(10)
```

结果：

```
quantileTDigest(number)
4.5 |
```

## 参见

- **中位数**
- **分位数**

# quantileTDigestWeighted

使用 **t-digest** 算法计算数字序列近似 **分位数**。该函数考虑了每个序列成员的权重。最大误差为 1%。内存消耗为  $\log(n)$ ，这里 **n** 是值的个数。

该函数的性能低于 **quantile** 或 **quantileTiming** 的性能。从状态大小和精度的比值来看，这个函数比 **quantile** 更优秀。

结果取决于运行查询的顺序，并且是不确定的。

当在一个查询中使用多个不同层次的 **quantile\*** 时，内部状态不会被组合（即查询的工作效率低于组合情况）。在这种情况下，使用 **quantiles** 函数。

## 语法

```
quantileTDigestWeighted(level)(expr, weight)
```

别名: **medianTDigestWeighted**。

## 参数

- **level** — 分位数层次。可选参数。从 0 到 1 的一个 **float** 类型的常量。我们推荐 **level** 值的范围为 [0.01, 0.99]。默认值：0.5。当 **level=0.5** 时，该函数计算 **中位数**。
- **expr** — 求值表达式，类型为数值类型 **data types**, **Date** 或 **DateTime**。
- **weight** — 权重序列。权重是一个数据出现的数值。

## 返回值

- 指定层次的分位数。

类型：

- **Float64** 用于数字数据类型输入。
- **Date** 如果输入值是 **Date** 类型。
- **DateTime** 如果输入值是 **DateTime** 类型。

## 示例

查询：

```
SELECT quantileTDigestWeighted(number, 1) FROM numbers(10)
```

结果：

```
quantileTDigestWeighted(number, 1)─
4.5 |
```

## 参见

- **中位数**
- **分位数**

# 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](#)

# simpleLinearRegression

执行简单（一维）线性回归。

语法

```
simpleLinearRegression(x, y)
```

参数

- `x` — `x`轴。
- `y` — `y`轴。

返回值

符合 $y = a*x + b$ 的常量 ( $a, b$ )。

示例

```
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

该函数实现随机线性回归。它支持自定义参数的学习率、L2正则化系数、微批，并且具有少量更新权重的方法（[Adam](#)（默认），[simple SGD](#)，[Momentum](#)，[Nesterov](#)）。

参数

有4个可自定义的参数。它们按顺序传递给函数，但不需要传递所有四个参数——将使用默认值，然而好的模型需要一些参数调整。

语法

```
stochasticLinearRegression(1.0, 1.0, 10, 'SGD')
```

1. `learning rate` 当执行梯度下降步骤时，步长的系数。过大的学习率可能会导致模型的权重无限大。默认值为 `0.00001`。
2. `L2 regularization coefficient` 这可能有助于防止过度拟合。默认值为 `0.1`。
3. `mini-batch size` 设置元素的数量，这些元素将被计算和求和以执行梯度下降的一个步骤。纯随机下降使用一个元素，但是具有小批量（约10个元素）使梯度步骤更稳定。默认值为 `15`。
4. `method for updating weights` 他们是: `Adam` (默认情况下), `SGD`, `Momentum`, `Nesterov`。`Momentum` 和 `Nesterov` 需要更多的计算和内存，但是它们恰好在收敛速度和随机梯度方法的稳定性方面是有用的。

## 使用

`stochasticLinearRegression` 用于两个步骤：拟合模型和预测新数据。为了拟合模型并保存其状态以供以后使用，我们使用 `-State` 组合器，它基本上保存了状态（模型权重等）。

为了预测我们使用函数 `evalMLMethod`，这需要一个状态作为参数以及特征来预测。

### 1. 拟合

可以使用这种查询。

```
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;
```

在这里，我们还需要将数据插入到 `train_data` 表。参数的数量不是固定的，它只取决于传入 `linearRegressionState` 的参数数量。它们都必须是数值。

注意，目标值(我们想学习预测的)列作为第一个参数插入。

### 2. 预测

在将状态保存到表中之后，我们可以多次使用它进行预测，甚至与其他状态合并，创建新的更好的模型。

```
WITH (SELECT state FROM your_model) AS model SELECT
evalMLMethod(model, param1, param2) FROM test_data
```

查询将返回一列预测值。注意，`evalMLMethod` 的第一个参数是 `AggregateFunctionState` 对象，接下来是特征列。

`test_data` 是一个类似 `train_data` 的表 但可能不包含目标值。

### 注

- 要合并两个模型，用户可以创建这样的查询：

```
sql SELECT state1 + state2 FROM your_models
```

其中 `your_models` 表包含这两个模型。此查询将返回新的 `AggregateFunctionState` 对象。

- 如果没有使用 `-State` 组合器，用户可以自己的目的获取所创建模型的权重，而不保存模型。

```
sql SELECT stochasticLinearRegression(0.01)(target, param1, param2) FROM train_data
```

这样的查询将拟合模型，并返回其权重——首先是权重，对应模型的参数，最后一个偏差。所以在上面的例子中，查询将返回一个具有3个值的列。

### 参见

- 随机指标逻辑回归
- 线性回归和逻辑回归之间的差异

## stochasticLogisticRegression

该函数实现随机逻辑回归。它可以用于二进制分类问题，支持与`stochasticLinearRegression`相同的自定义参数，并以相同的方式工作。

### 参数

参数与stochasticLinearRegression中的参数完全相同:

learning rate, l2 regularization coefficient, mini-batch size, method for updating weights.

欲了解更多信息，参见 [参数] (#agg\_functions-stochasticlinearregression-parameters).

## 语法

```
stochasticLogisticRegression(1.0, 1.0, 10, 'SGD')
```

### 1. 拟合

参考[stochasticLinearRegression](#stochasticlinearregression-usage-fitting) `拟合` 章节文档。

预测标签的取值范围为[-1, 1]

### 2. 预测

使用已经保存的state我们可以预测标签为`1`的对象的概率。

```
```sql
WITH (SELECT state FROM your_model) AS model SELECT
evalMLMethod(model, param1, param2) FROM test_data
```
```

查询结果返回一个列的概率。注意`evalMLMethod`的第一个参数是`AggregateFunctionState`对象，接下来的参数是列的特性。

我们也可以设置概率的范围，这样需要给元素指定不同的标签。

```
```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)
```
```

结果是标签。

`test\_data`是一个像`train\_data`一样的表，但是不包含目标值。

## 参见

- [随机指标线性回归](#)
- [线性回归和逻辑回归之间的差异](#)

## categoricalInformationValue

对于每个类别计算  $(P(tag = 1) - P(tag = 0))(\log(P(tag = 1)) - \log(P(tag = 0)))$ 。

```
categoricalInformationValue(category1, category2, ..., tag)
```

结果指示离散（分类）要素如何使用 [category1, category2, ...] 有助于使用学习模型预测tag的值。

## studentTTest

对两个总体的样本应用t检验。

## 语法

```
studentTTest(sample_data, sample_index)
```

两个样本的值都在 `sample_data` 列中。如果 `sample_index` 等于 0，则该行的值属于第一个总体的样本。反之属于第二个总体的样本。

零假设是总体的均值相等。假设为方差相等的正态分布。

## 参数

- `sample_data` — 样本数据。`Integer`, `Float` 或 `Decimal`。

- `sample_index` — 样本索引。`Integer`。

## 返回值

元组，有两个元素：

- 计算出的t统计量。`Float64`。

- 计算出的p值。`Float64`。

## 示例

输入表：

| sample_data | sample_index |
|-------------|--------------|
| 20.3        | 0            |
| 21.1        | 0            |
| 21.9        | 1            |
| 21.7        | 0            |
| 19.9        | 1            |
| 21.8        | 1            |

查询：

```
SELECT studentTTest(sample_data, sample_index) FROM student_ttest;
```

结果：

```
studentTTest(sample_data, sample_index)
(-0.21739130434783777, 0.8385421208415731)
```

## 参见

- [Student's t-test](#)

- [welchTTest function](#)

## welchTTest

对两个总体的样本应用 Welch t检验。

## 语法

```
welchTTest(sample_data, sample_index)
```

两个样本的值都在 `sample_data` 列中。如果 `sample_index` 等于 0，则该行的值属于第一个总体的样本。反之属于第二个总体的样本。

零假设是群体的均值相等。假设为正态分布。总体可能具有不相等的方差。

## 参数

- `sample_data` — 样本数据。`Integer`, `Float` 或 `Decimal`。
- `sample_index` — 样本索引。`Integer`。

## 返回值

元组，有两个元素：

- 计算出的t统计量。`Float64`。
- 计算出的p值。`Float64`。

## 示例

输入表：

| sample_data | sample_index |
|-------------|--------------|
| 20.3        | 0            |
| 22.1        | 0            |
| 21.9        | 0            |
| 18.9        | 1            |
| 20.3        | 1            |
| 19          | 1            |

查询：

```
SELECT welchTTest(sample_data, sample_index) FROM welch_ttest;
```

结果：

```
welchTTest(sample_data, sample_index)
(2.7988719532211235, 0.051807360348581945)
```

## 参见

- [Welch's t-test](#)
- [studentTTest function](#)

# mannWhitneyUTest

对两个总体的样本应用 Mann-Whitney 秩检验。

## 语法

```
mannWhitneyUTest[(alternative[, continuity_correction])](sample_data, sample_index)
```

两个样本的值都在 `sample_data` 列中。如果 `sample_index` 等于 0，则该行的值属于第一个总体的样本。反之属于第二个总体的样本。

零假设是两个总体随机相等。也可以检验单边假设。该检验不假设数据具有正态分布。

## 参数

- `sample_data` — 样本数据。`Integer`, `Float` 或 `Decimal`。

- `sample_index` — 样本索引。`Integer`。

## 参数

- `alternative` — 供选假设。(可选，默认值是: 'two-sided'。) `String`。
  - 'two-sided';
  - 'greater';
  - 'less'。
- `continuity_correction` — 如果不为0，那么将对p值进行正态近似的连续性修正。(可选，默认：1。) `UInt64`。

## 返回值

元组，有两个元素：

- 计算出U统计量。`Float64`。
- 计算出的p值。`Float64`。

## 示例

输入表：

| sample_data | sample_index |
|-------------|--------------|
| 10          | 0            |
| 11          | 0            |
| 12          | 0            |
| 1           | 1            |
| 2           | 1            |
| 3           | 1            |

查询：

```
SELECT mannWhitneyUTest('greater')(sample_data, sample_index) FROM mww_ttest;
```

结果：

```
mannWhitneyUTest('greater')(sample_data, sample_index)
(9,0.04042779918503192)
```

## 参见

- [Mann-Whitney U test](#)
- [Stochastic ordering](#)

# median

`median*` 函数是 `quantile*` 函数的别名。它们计算数字数据样本的中位数。

函数：

- `median` — `quantile`别名。
- `medianDeterministic` — `quantileDeterministic`别名。

- `medianExact` — `quantileExact`别名。
- `medianExactWeighted` — `quantileExactWeighted`别名。
- `medianTiming` — `quantileTiming`别名。
- `medianTimingWeighted` — `quantileTimingWeighted`别名。
- `medianTDigest` — `quantileTDigest`别名。
- `medianTDigestWeighted` — `quantileTDigestWeighted`别名。

#### 示例

输入表:

| val |
|-----|
| 1   |
| 1   |
| 2   |
| 3   |

查询:

```
SELECT medianDeterministic(val, 1) FROM t
```

结果:

| medianDeterministic(val, 1) |
|-----------------------------|
| 1.5                         |

## rankCorr

计算等级相关系数。

#### 语法

```
rankCorr(x, y)
```

#### 参数

- `x` — 任意值。`Float32` 或 `Float64`。
- `y` — 任意值。`Float32` 或 `Float64`。

#### 返回值

- 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.

类型: `Float64`。

## 示例

查询：

```
SELECT rankCorr(number, number) FROM numbers(100);
```

结果：

```
rankCorr(number, number)─
1 |
```

查询：

```
SELECT roundBankers(rankCorr(exp(number), sin(number)), 3) FROM numbers(100);
```

结果：

```
roundBankers(rankCorr(exp(number), sin(number)), 3)─
-0.037 |
```

## 参见

- 斯皮尔曼等级相关系数 [Spearman's rank correlation coefficient](#)

## 聚合函数组合器

聚合函数的名称可以附加一个后缀。这改变了聚合函数的工作方式。

### -If

-If可以加到任何聚合函数之后。加了-If之后聚合函数需要接受一个额外的参数，一个条件（Uint8类型），如果条件满足，那聚合函数处理当前的行数据，如果不满足，那返回默认值（通常是0或者空字符串）。

例：`sumIf(column, cond)`, `countIf(cond)`, `avgIf(x, cond)`, `quantilesTimingIf(level1, level2)(x, cond)`, `argMinIf(arg, val, cond)` 等等。

使用条件聚合函数，您可以一次计算多个条件的聚合，而无需使用子查询和 JOIN 例如，在 Yandex.Metrica，条件聚合函数用于实现段比较功能。

### -Array

-Array后缀可以附加到任何聚合函数。在这种情况下，聚合函数采用的参数 ‘Array(T)’ 类型（数组）而不是 ‘T’ 类型参数。如果聚合函数接受多个参数，则它必须是长度相等的数组。在处理数组时，聚合函数的工作方式与所有数组元素的原始聚合函数类似。

示例1：`sumArray(arr)` - 总计所有的所有元素 ‘arr’ 阵列。在这个例子中，它可以更简单地编写：`sum(arraySum(arr))`。

示例2：`uniqArray(arr)` - 计算‘arr’中唯一元素的个数。这可以是一个更简单的方法：`uniq(arrayJoin(arr))`，但它并不总是可以添加 ‘arrayJoin’ 到查询。

如果和-If组合，‘Array’ 必须先来，然后 ‘If’。例：`uniqArrayIf(arr, cond)`, `quantilesTimingArrayIf(level1, level2)(arr, cond)`。由于这个顺序，该 ‘cond’ 参数不会是数组。

### -State

如果应用此combinator，则聚合函数不会返回结果值（例如唯一值的数量 `uniq` 函数），但是返回聚合的中间状态（对于 `uniq`，返回的是计算唯一值的数量的哈希表）。这是一个 `AggregateFunction(...)` 可用于进一步处理或存储在表中以完成稍后的聚合。

要使用这些状态，请使用：

- `AggregatingMergeTree` 表引擎。
- `finalizeAggregation` 功能。
- `runningAccumulate` 功能。
- `-Merge` combinator
- `-MergeState` combinator

## -Merge

如果应用此组合器，则聚合函数将中间聚合状态作为参数，组合状态以完成聚合，并返回结果值。

## -MergeState

以与-Merge 相同的方式合并中间聚合状态。但是，它不会返回结果值，而是返回中间聚合状态，类似于-State。

## -ForEach

将表的聚合函数转换为聚合相应数组项并返回结果数组的数组的聚合函数。例如, `sumForEach` 对于数组 [1, 2], [3, 4, 5] 和 [6, 7] 返回结果 [10, 13, 5] 之后将相应的数组项添加在一起。

## -OrDefault

更改聚合函数的行为。

如果聚合函数没有输入值，则使用此组合器它返回其返回数据类型的默认值。适用于可以采用空输入数据的聚合函数。

`-OrDefault` 可与其他组合器一起使用。

语法

```
<aggFunction>OrDefault(x)
```

参数

- `x` — 聚合函数参数。

返回值

如果没有要聚合的内容，则返回聚合函数返回类型的默认值。

类型取决于所使用的聚合函数。

示例

查询:

```
SELECT avg(number), avgOrDefault(number) FROM numbers(0)
```

结果:

|             |   |                      |
|-------------|---|----------------------|
| avg(number) | — | avgOrDefault(number) |
| nan         |   | 0                    |

还有 `-OrDefault` 可与其他组合器一起使用。当聚合函数不接受空输入时，它很有用。

查询：

```
SELECT avgOrDefaultIf(x, x > 10)
FROM
(
 SELECT toDecimal32(1.23, 2) AS x
)
```

结果：

|                                   |
|-----------------------------------|
| avgOrDefaultIf(x, greater(x, 10)) |
| 0.00                              |

## -OrNull

更改聚合函数的行为。

此组合器将聚合函数的结果转换为 **可为空** 数据类型。如果聚合函数没有值来计算它返回 **NULL**.

`-OrNull` 可与其他组合器一起使用。

语法

```
<aggFunction>OrNull(x)
```

参数

- `x` — Aggregate function parameters.

返回值

- 聚合函数的结果，转换为 **Nullable** 数据类型。

- **NULL**，如果没有进行聚合。

类型: **Nullable(aggregate function return type)**.

示例

添加 `-orNull` 到聚合函数的末尾。

查询：

```
SELECT sumOrNull(number), toTypeName(sumOrNull(number)) FROM numbers(10) WHERE number > 10
```

结果：

|                   |   |                               |
|-------------------|---|-------------------------------|
| sumOrNull(number) | — | toTypeName(sumOrNull(number)) |
| NULL              |   | Nullable(UInt64)              |

还有 `-OrNull` 可与其他组合器一起使用。当聚合函数不接受空输入时，它很有用。

查询：

```
SELECT avgOrNullIf(x, x > 10)
FROM
(
 SELECT toDecimal32(1.23, 2) AS x
)
```

结果：

```
avgOrNullIf(x, greater(x, 10)) ━
 NULL |
```

## -Resample

允许您将数据划分为组，然后单独聚合这些组中的数据。通过将一列中的值拆分为间隔来创建组。

```
<aggFunction>Resample(start, end, step)(<aggFunction_params>, resampling_key)
```

参数

- `start` — `resampling_key` 开始值。
- `stop` — `resampling_key` 结束边界。区间内部不包含 `stop` 值，即 `[start, stop)`.
- `step` — 分组的步长。The `aggFunction` 在每个子区间上独立执行。
- `resampling_key` — 取样列，被用来分组.
- `aggFunction_params` — `aggFunction` 参数。

返回值

- `aggFunction` 每个子区间的結果，結果为数组。

示例

考虑一下 `people` 表具有以下数据的表结构：

| name   | age | wage |
|--------|-----|------|
| John   | 16  | 10   |
| Alice  | 30  | 15   |
| Mary   | 35  | 8    |
| Evelyn | 48  | 11.5 |
| David  | 62  | 9.9  |
| Brian  | 60  | 16   |

让我们得到人的名字，他们的年龄在于的时间间隔 `[30,60)` 和 `[60,75)`。由于我们使用整数表示的年龄，我们得到的年龄 `[30, 59]` 和 `[60,74]` 间隔。

要在数组中聚合名称，我们使用 `groupArray` 聚合函数。这需要一个参数。在我们的例子中，它是 `name` 列。`groupArrayResample` 函数应该使用 `age` 按年龄聚合名称，要定义所需的时间间隔，我们传入 `30, 75, 30` 参数给 `groupArrayResample` 函数。

```
SELECT groupArrayResample(30, 75, 30)(name, age) FROM people
```

```
groupArrayResample(30, 75, 30)(name, age)——
[['Alice','Mary','Evelyn'],['David','Brian']] |
```

考虑结果。

John 没有被选中，因为他太年轻了。 其他人按照指定的年龄间隔进行分配。

现在让我们计算指定年龄间隔内的总人数和平均工资。

```
SELECT
 countResample(30, 75, 30)(name, age) AS amount,
 avgResample(30, 75, 30)(wage, age) AS avg_wage
FROM people
```

```
amount——avg_wage——
[3,2] | [11.5,12.949999809265137] |
```

## 参数聚合函数

一些聚合函数不仅可以接受参数列（用于压缩），也可以接收常量的初始化参数。这种语法是接受两个括号的参数，第一个数初始化参数，第二个是入参。

## histogram

计算自适应直方图。 它不能保证精确的结果。

```
histogram(number_of_bins)(values)
```

该函数使用 [流式并行决策树算法](#)。当新数据输入函数时，hist图分区的边界将被调整。在通常情况下，箱的宽度不相等。

### 参数

`number_of_bins` — 直方图bin个数，这个函数会自动计算bin的数量，而且会尽量使用指定值，如果无法做到，那就使用更小的bin个数。

`values` — 表达式 输入值。

### 返回值

- `Array` 的 `Tuples` 如下：

```
```  
[(lower_1, upper_1, height_1), ... (lower_N, upper_N, height_N)]  
  
- `lower` — bin的下边界。  
- `upper` — bin的上边界。  
- `height` — bin的计算权重。
```

示例

```
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)] |
```

您可以使用 **bar** 功能，例如：

```
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
)
```



在这种情况下，您应该记住您不知道直方图bin边界。

sequenceMatch(pattern)(timestamp, cond1, cond2, ...)

检查序列是否包含与模式匹配的事件链。

```
sequenceMatch(pattern)(timestamp, cond1, cond2, ...)
```

警告

在同一秒钟发生的事件可能以未定义的顺序排列在序列中，影响结果。

参数

- **pattern** — 模式字符串。参考 [模式语法](#).
- **timestamp** — 包含时间的列。典型的时间类型是：`Date` 和 `DateTime`。您还可以使用任何支持的 `UInt` 数据类型。
- **cond1, cond2** — 事件链的约束条件。数据类型是：`UInt8`。最多可以传递32个条件参数。该函数只考虑这些条件中描述的事件。如果序列包含未在条件中描述的数据，则函数将跳过这些数据。

返回值

- 1，如果模式匹配。
- 0，如果模式不匹配。

类型: UInt8.

模式语法

- `(?N)` — 在位置N匹配条件参数。 条件在编号 [1, 32] 范围。 例如, `(?1)` 匹配传递给 `cond1` 参数。
- `.*` — 匹配任何事件的数字。 不需要条件参数来匹配这个模式。
- `(?t operator value)` — 分开两个事件的时间。 例如 : `(?1)(?t>1800)(?2)` 匹配彼此发生超过1800秒的事件。 这些事件之间可以存在任意数量的任何事件。 您可以使用 `>=, >, <, <=, ==` 运算符。

例

考虑在数据 `t` 表:

time	number
1	1
2	3
3	2

执行查询:

```
SELECT sequenceMatch('(?1)(?2)')(time, number = 1, number = 2) FROM t
```

```
sequenceMatch('(?1)(?2)')(time, equals(number, 1), equals(number, 2))—  
1 |
```

该函数找到了数字2跟随数字1的事件链。 它跳过了它们之间的数字3，因为该数字没有被描述为事件。 如果我们想在搜索示例中给出的事件链时考虑这个数字，我们应该为它创建一个条件。

```
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 |
```

在这种情况下，函数找不到与模式匹配的事件链，因为数字3的事件发生在1和2之间。 如果在相同的情况下，我们检查了数字4的条件，则序列将与模式匹配。

```
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 |
```

另请参阅

- [sequenceCount](#)

sequenceCount(pattern)(time, cond1, cond2, ...)

计算与模式匹配的事件链的数量。该函数搜索不重叠的事件链。当前链匹配后，它开始搜索下一个链。

警告

在同一秒钟发生的事件可能以未定义的顺序排列在序列中，影响结果。

`sequenceCount(pattern)(timestamp, cond1, cond2, ...)`

参数

- `pattern` — 模式字符串。参考：[模式语法](#)。
- `timestamp` — 包含时间的列。典型的时间类型是：`Date` 和 `DateTime`。您还可以使用任何支持的 `UInt` 数据类型。
- `cond1, cond2` — 事件链的约束条件。数据类型是：`UInt8`。最多可以传递32个条件参数。该函数只考虑这些条件下描述的事件。如果序列包含未在条件中描述的数据，则函数将跳过这些数据。

返回值

- 匹配的非重叠事件链数。

类型：`UInt64`。

示例

考虑在数据 `t` 表：

time	number
1	1
2	3
3	2
4	1
5	3
6	2

计算数字2在数字1之后出现的次数以及它们之间的任何其他数字：

```
SELECT sequenceCount('(?1).*(?2)')(time, number = 1, number = 2) FROM t
```

```
sequenceCount('(?1).*(?2)')(time, equals(number, 1), equals(number, 2))—  
2 |
```

另请参阅

- [sequenceMatch](#)

windowFunnel

搜索滑动时间窗中的事件链，并计算从链中发生的最大事件数。

该函数采用如下算法：

- 该函数搜索触发链中的第一个条件并将事件计数器设置为1。这是滑动窗口启动的时刻。
- 如果来自链的事件在窗口内顺序发生，则计数器将递增。如果事件序列中断，则计数器不再增加。
- 如果数据在不同的完成点具有多个事件链，则该函数将仅输出最长链的大小。

语法

```
windowFunnel(window, [mode, [mode, ... ]])(timestamp, cond1, cond2, ..., condN)
```

参数

- **window** — 滑动窗户的大小，表示事件链中第一个事件和最后一个事件的最大间隔。单位取决于**timestamp**。用表达式来表示则是：`timestamp of cond1 <= timestamp of cond2 <= ... <= timestamp of condN <= timestamp of cond1 + window`。
- **mode** - 这是一个可选的参数，可以设置一个或多个参数。
 - '`strict_deduplication`' - 如果事件链中出现相同的条件，则会停止进一步搜索。
 - '`strict_order`' - 不允许其他事件的介入。例如：在A->B->D->C的情况下，它在D停止继续搜索A->B->C，最大事件数为2。
 - '`strict_increase`' - 事件链中的时间戳必须严格上升。
- **timestamp** — 包含时间戳的列。数据类型支持：**日期**, **日期时间** 和其他无符号整数类型（请注意，即使时间戳支持 UInt64 类型，它的值也不能超过 Int64 最大值，即 $2^{63}-1$ ）。
- **cond** — 事件链的约束条件。**UInt8** 类型。

返回值

滑动时间窗口内连续触发条件链的最大数目。

对选择中的所有链进行了分析。

类型: **Integer**.

示例

确定设定的时间段是否足以让用户选择手机并在在线商店中购买两次。

设置以下事件链:

1. 用户登录到其在应用商店中的帐户 (`eventID = 1003`).
2. 用户搜索手机 (`eventID = 1007, product = 'phone'`).
3. 用户下了订单 (`eventID = 1009`).
4. 用户再次下订单 (`eventID = 1010`).

输入表:

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

了解用户 `user_id` 可以在 2019 的 1-2 月期间通过链条多远。

查询:

```

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

```

结果：

level	c
4	1

Retention

该函数将一组条件作为参数，类型为1到32个 UInt8 类型的参数，用来表示事件是否满足特定条件。

任何条件都可以指定为参数（如 **WHERE**）。

除了第一个以外，条件成对适用：如果第一个和第二个是真的，第二个结果将是真的，如果第一个和第三个是真的，第三个结果将是真的，等等。

语法

```
retention(cond1, cond2, ..., cond32);
```

参数

- **cond** — 返回 UInt8 结果（1或0）的表达式。

返回值

数组为1或0。

- 1 — 条件满足。
- 0 — 条件不满足。

类型: UInt8.

示例

让我们考虑使用 **retention** 功能的一个例子，以确定网站流量。

1. 举例说明，先创建一张表。

```

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);

```

输入表：

查询:

```
SELECT * FROM retention_test
```

结果:

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. 按唯一ID uid 对用户进行分组，使用 retention 功能。

查询:

```
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
```

结果:

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. 计算每天的现场访问总数。

查询:

```

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
)
    
```

结果:

r1	r2	r3
5	5	5

条件:

- `r1`-2020-01-01期间访问该网站的独立访问者数量（`cond1` 条件）。
- `r2`-在2020-01-01和2020-01-02之间的特定时间段内访问该网站的唯一访问者的数量 (`cond1` 和 `cond2` 条件)。
- `r3`-在2020-01-01和2020-01-03之间的特定时间段内访问该网站的唯一访问者的数量 (`cond1` 和 `cond3` 条件)。

uniqUpTo(N)(x)

计算小于或者等于N的不同参数的个数。如果结果大于N，那返回N+1。

建议使用较小的Ns，比如：10。N的最大值为100。

对于聚合函数的状态，它使用的内存量等于 $1+N \times$ 一个字节值的大小。

对于字符串，它存储8个字节的非加密哈希。也就是说，计算是近似的字符串。

该函数也适用于多个参数。

它的工作速度尽可能快，除了使用较大的N值并且唯一值的数量略小于N的情况。

用法示例:

问题：产出一个不少于五个唯一用户的关键字报告

解决方案：写group by查询语句 HAVING uniqUpTo(4)(UserID) >= 5

sumMapFiltered(keys_to_keep)(keys, values)

和 [sumMap](#) 基本一致，除了一个键数组作为参数传递。这在使用高基数key时尤其有用。

表函数

表函数是用来构造表的方法。

您可以在以下位置使用表函数：

- `SELECT` 查询的 `FROM` 子句。

创建临时表的方法，该临时表仅在当前查询中可用。当查询完成后，该临时表将被删除。

- `CREATE TABLE AS \<table_function(>` 查询。

这是创建表的方法之一。

警告

如果 `allow_ddl` 设置被禁用，则不能使用表函数。

函数	描述
<code>file</code>	创建一个file引擎表。
<code>merge</code>	创建一个merge引擎表。
<code>numbers</code>	创建一个单列的表，其中包含整数。
<code>remote</code>	允许您访问远程服务器，而无需创建分布式表。
<code>url</code>	创建一个URL引擎表。
<code>mysql</code>	创建一个MySQL引擎表。
<code>jdbc</code>	创建一个JDBC引擎表。
<code>odbc</code>	创建一个ODBC引擎表。
<code>hdfs</code>	创建一个HDFS引擎表。

file

从文件创建表。此表函数类似于 `url` 和 `hdfs`。

`file` 函数可用于对 `File` 表中的数据进行 `SELECT` 和 `INSERT` 查询。

语法

```
file(path, format, structure)
```

参数

- `path` — `user_files_path` 中文件的相对路径。在只读模式下，文件路径支持以下通配符: `*`, `?`, `{abc,def}` 和 `{N..M}`，其中 `N, M` 是数字，`'abc', 'def'` 是字符串。
- `format` — 文件的格式。
- `structure` — 表的结构。格式 `'column1_name column1_type, column2_name column2_type, ...'`。

返回值

具有指定结构的表，用于读取或写入指定文件中的数据。

示例

设置 `user_files_path` 和文件 `test.csv` 的内容:

```
$ 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
```

从 `test.csv` 中的表中获取数据，并从表中选择前两行:

```
SELECT * FROM file('test.csv', 'CSV', 'column1 UInt32, column2 UInt32, column3 UInt32') LIMIT 2;
```

column1	column2	column3
1	2	3
3	2	1

从 CSV 文件获取包含 3 列 `UInt32` 类型的表的前 10 行:

```
SELECT * FROM file('test.csv', 'CSV', 'column1 UInt32, column2 UInt32, column3 UInt32') LIMIT 10;
```

将文件中的数据插入表中:

```
INSERT INTO FUNCTION file('test.csv', 'CSV', 'column1 UInt32, column2 UInt32, column3 UInt32') VALUES (1, 2, 3), (3, 2, 1);
SELECT * FROM file('test.csv', 'CSV', 'column1 UInt32, column2 UInt32, column3 UInt32');
```

column1	column2	column3
1	2	3
3	2	1

路径中的通配符

多个路径组件可以具有通配符。对于要处理的文件必须存在并与整个路径模式匹配（不仅后缀或前缀）。

- `*` — 替换任意数量的任何字符，除了 / 包括空字符串。
- `?` — 替换任何单个字符。
- `{some_string,another_string,yet_another_one}` — 替换任何字符串 'some_string', 'another_string', 'yet_another_one'。
- `{N..M}` — 替换范围从N到M的任何数字（包括两个边界）。

使用 `{}` 的构造类似于 **remote** 表函数。

示例

假设我们有几个文件，这些文件具有以下相对路径：

- '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'

查询这些文件中的行数：

```
SELECT count(*)  
FROM file('{some,another}_dir/some_file_{1..3}', 'TSV', 'name String, value UInt32')
```

查询这两个目录的所有文件中的行数：

```
SELECT count(*)  
FROM file('{some,another}_dir/*', 'TSV', 'name String, value UInt32')
```

警告

如果您的文件列表包含带前导零的数字范围，请对每个数字分别使用带有大括号的结构或使用 `?`。

示例

从名为 `file000, file001, ..., file999` 的文件中查询数据：

```
SELECT count(*)  
FROM file('big_dir/file{0..9}{0..9}{0..9}', 'CSV', 'name String, value UInt32')
```

虚拟列

- `_path` — 文件路径。
- `_file` — 文件名称。

另请参阅

- **虚拟列**

merge

`merge(db_name, 'tables_regex')` – 创建一个临时Merge表。有关更多信息，请参见“Table engines, Merge”。

表结构取自遇到的第一个与正则表达式匹配的表。

numbers

`numbers(N)` – 返回一个包含单个 ‘number’ 列(UInt64)的表，其中包含从0到N-1的整数。

`numbers(N, M)` – 返回一个包含单个 ‘number’ 列(UInt64)的表，其中包含从N到(N+M-1)的整数。

类似于 `system.numbers` 表，它可用于测试和生成连续的值，`numbers(N, M)` 比 `system.numbers` 更有效。

以下查询是等价的：

```
SELECT * FROM numbers(10);
SELECT * FROM numbers(0, 10);
SELECT * FROM system.numbers LIMIT 10;
```

示例：

```
-- 生成2010-01-01至2010-12-31的日期序列
select toDate('2010-01-01') + number as d FROM numbers(365);
```

url

`url` 函数从 URL 创建一个具有给定 `format` 和 `structure` 的表。

`url` 函数可用于对 `URL` 表中的数据进行 `SELECT` 和 `INSERT` 的查询中。

语法

```
url(URL, format, structure)
```

参数

- `URL` — HTTP或HTTPS服务器地址，它可以接受 `GET` 或 `POST` 请求 (对应于 `SELECT` 或 `INSERT` 查询)。类型: `String`。
- `format` — 数据格式。类型: `String`。
- `structure` — 以 '`UserID UInt64, Name String`' 格式的表结构。确定列名和类型。类型: `String`。

返回值

A table with the specified format and structure and with data from the defined URL.

示例

获取一个表的前3行，该表是从HTTP服务器获取的包含 `String` 和 `UInt32` 类型的列，以 `CSV` 格式返回。

```
SELECT * FROM url('http://127.0.0.1:12345/', CSV, 'column1 String, column2 UInt32') LIMIT 3;
```

将 `URL` 的数据插入到表中：

```
CREATE TABLE test_table (column1 String, column2 UInt32) ENGINE=Memory;
INSERT INTO FUNCTION url('http://127.0.0.1:8123/?query=INSERT+INTO+test_table+FORMAT+CSV', 'CSV', 'column1
String, column2 UInt32') VALUES ('http interface', 42);
SELECT * FROM test_table;
```

postgresql

允许对存储在远程 PostgreSQL 服务器上的数据进行 `SELECT` 和 `INSERT` 查询。

语法

```
postgresql('host:port', 'database', 'table', 'user', 'password'[, `schema`])
```

参数

- `host:port` — PostgreSQL 服务器地址.
- `database` — 远程数据库名称.
- `table` — 远程表名称.
- `user` — PostgreSQL 用户.
- `password` — 用户密码.
- `schema` — 非默认的表结构. 可选.

返回值

一个表对象，其列数与原 PostgreSQL 表的列数相同。

Note

在 `INSERT` 查询中，为了区分表函数 `postgresql(..)` 和表名以及表的列名列表，你必须使用关键字 `FUNCTION` 或 `TABLE FUNCTION`。请看下面的例子。

实施细节

`SELECT` 查询在 PostgreSQL 上以 `COPY (SELECT ...)` TO `STDOUT` 的方式在只读的 PostgreSQL 事务中运行，每次在 `SELECT` 查询后提交。

简单的 `WHERE` 子句，如 `=`、`!=`、`>`、`>=`、`<`、`<=` 和 `IN`，在 PostgreSQL 服务器上执行。

所有的连接、聚合、排序，`IN [数组]` 条件和 `LIMIT` 采样约束只有在对 PostgreSQL 的查询结束后才会在 ClickHouse 中执行。

PostgreSQL 上的 `INSERT` 查询以 `COPY "table_name" (field1, field2, ... fieldN)` FROM `STDIN` 的方式在 PostgreSQL 事务中运行，每次 `INSERT` 语句后自动提交。

PostgreSQL 数组类型将转换为 ClickHouse 数组。

Note

要小心，在 PostgreSQL 中，像 `Integer[]` 这样的数组数据类型列可以在不同的行中包含不同维度的数组，但在 ClickHouse 中，只允许在所有的行中有相同维度的多维数组。

支持设置 PostgreSQL 字典源中 Replicas 的优先级。地图中的数字越大，优先级就越低。0 代表最高的优先级。

示例

PostgreSQL 中的表:

```
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)
```

从 ClickHouse 检索数据:

```
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

插入数据:

```
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

使用非默认的表结构:

```
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', 'postgresql_user', 'password', 'nice.schema');
```

另请参阅

- [PostgreSQL 表引擎](#)
- [使用 PostgreSQL 作为外部字典的来源](#)

jdbc

`jdbc(datasource, schema, table)` - 返回通过 JDBC 驱动程序连接的表。

此表函数需要单独的 [clickhouse-jdbc-bridge](#) 程序才能运行。

它支持可空类型（基于查询的远程表的 DDL）。

示例

```
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', 'select toInt32OrZero("") as num')
```

```
SELECT *
FROM jdbc('mysql-dev?p1=233', 'num Int32', 'select toInt32OrZero("") 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

返回通过 [ODBC](#) 连接的表。

```
odbc(connection_settings, external_database, external_table)
```

参数：

- `connection_settings` — 在 `odbc.ini` 文件中连接设置的部分的名称。
- `external_database` — 外部 DBMS 的数据库名。
- `external_table` — `external_database` 数据库中的表名。

为了安全地实现 ODBC 连接，ClickHouse 使用单独的程序 `clickhouse-odbc-bridge`。如果 ODBC 驱动程序直接从 `clickhouse-server` 加载，则驱动程序问题可能会导致 ClickHouse 服务器崩溃。当需要时，ClickHouse 自动启动 `clickhouse-odbc-bridge`。ODBC 桥程序是从与 `clickhouse-server` 相同的软件包安装的。

外部表中字段包含的 `NULL` 值将转换为基本数据类型的默认值。例如，如果远程 MySQL 表字段包含 `INT NULL` 类型，则将被转换为 0 (`ClickHouse Int32` 数据类型的默认值)。

用法示例

通过 ODBC 从本地安装的 MySQL 获取数据

这个例子检查 Ubuntu Linux 18.04 和 MySQL 服务器 5.7。

确保已经安装了 unixODBC 和 MySQL 连接器。

默认情况下（如果从软件包安装），ClickHouse 以用户 `clickhouse` 启动。因此，您需要在 MySQL 服务器中创建和配置此用户。

```
$ sudo mysql
```

```
mysql> CREATE USER 'clickhouse'@'localhost' IDENTIFIED BY 'clickhouse';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'clickhouse'@'clickhouse' WITH GRANT OPTION;
```

然后在 `/etc/odbc.ini` 中配置连接。

```
$ cat /etc/odbc.ini
[mysqlconn]
DRIVER = /usr/local/lib/libmyodbc5w.so
SERVER = 127.0.0.1
PORT = 3306
DATABASE = test
USERNAME = clickhouse
PASSWORD = clickhouse
```

您可以使用 unixODBC 安装的 `isql` 实用程序检查连接。

```
$ isql -v mysqlconn
+-----+
| Connected!
|                               |
...
```

MySQL 中的表：

```
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)
```

从 ClickHouse 中的 MySQL 表中检索数据：

```
SELECT * FROM odbc('DSN=mysqlconn', 'test', 'test')
```

int_id	int_nullable	float	float_nullable
1	0	2	0

另请参阅

- [ODBC 外部字典](#)
- [ODBC 表引擎](#).

hdfs

根据HDFS中的文件创建表。该表函数类似于 [url](#) 和 [文件](#)。

```
hdfs(URI, format, structure)
```

输入参数

- `URI` — HDFS中文件的相对URI。在只读模式下，文件路径支持以下通配符: `*`, `?`, `{abc,def}` 和 `{N..M}`，其中 `N, M` 是数字, `'abc'`, `'def'` 是字符串。
- `format` — 文件的格式。
- `structure` — 表的结构。格式 `'column1_name column1_type, column2_name column2_type, ...'`。

返回值

具有指定结构的表，用于读取或写入指定文件中的数据。

示例

表来自 `hdfs://hdfs1:9000/test` 并从中选择前两行:

```
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

路径中的通配符

多个路径组件可以具有通配符。对于要处理的文件必须存在并与整个路径模式匹配（不仅后缀或前缀）。

- `*` — 替换任意数量的任何字符，除了 / 包括空字符串。
- `?` — 替换任何单个字符。
- `{some_string,another_string,yet_another_one}` — 替换任何字符串 `'some_string'`, `'another_string'`, `'yet_another_one'`。
- `{N..M}` — 替换范围从 `N` 到 `M` 的任何数字（包括两个边界）。

使用 `{}` 的构造类似于 [remote](#)(表)函数。

示例

1. 假设我们在HDFS上有几个带有以下URI的文件:

- `'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'`

2. 查询这些文件中的行数:

```
SELECT count(*)  
FROM hdfs('hdfs://hdfs1:9000/{some,another}_dir/some_file_{1..3}', 'TSV', 'name String, value UInt32')
```

3. 查询这两个目录的所有文件中的行数:

```
SELECT count(*)  
FROM hdfs('hdfs://hdfs1:9000/{some,another}_dir/*', 'TSV', 'name String, value UInt32')
```

警告

如果您的文件列表包含带前导零的数字范围，请对每个数字分别使用带有大括号的结构或使用 `?`。

示例

从名为 `file000, file001, ..., file999` 的文件中查询数据:

```
SELECT count(*)  
FROM hdfs('hdfs://hdfs1:9000/big_dir/file{0..9}{0..9}{0..9}', 'CSV', 'name String, value UInt32')
```

虚拟列

- `_path` — 文件路径。
 - `_file` — 文件名称。
- 另请参阅
- [虚拟列](#)

S3 表函数

提供类似于表的接口来 `select/insert` Amazon S3 中的文件。这个表函数类似于 `hdfs`，但提供了 S3 特有的功能。

语法

```
s3(path, [aws_access_key_id, aws_secret_access_key], format, structure, [compression])
```

参数

- `path` — 带有文件路径的 Bucket url。在只读模式下支持以下通配符: `*`, `?`, `{abc,def}` 和 `{N..M}` 其中 `N, M` 是数字, `'abc'`, `'def'` 是字符串。更多信息见[下文](#)。
- `format` — 文件的[格式](#)。
- `structure` — 表的结构。格式像这样 `'column1_name column1_type, column2_name column2_type, ...'`。
- `compression` — 压缩类型。支持的值: `none, gzip/gz, brotli/br, xz/LZMA, zstd/zst`。参数是可选的。默认情况下，通过文件扩展名自动检测压缩类型。

返回值

一个具有指定结构的表，用于读取或写入指定文件中的数据。

示例

从 S3 文件<https://storage.yandexcloud.net/my-test-bucket-768/data.csv>中选择表格的前两行:

```
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

类似的情况，但来源是gzip压缩的文件:

```
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

用法

假设我们在S3上有几个文件，URI如下:

- '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/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_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_4.csv'

计算以数字1至3结尾的文件的总行数:

```
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

计算这两个目录中所有文件的行的总量:

```
SELECT count(*)  
FROM s3('https://storage.yandexcloud.net/my-test-bucket-768/{some,another}_prefix/*', 'CSV', 'name String, value UInt32')
```

```
count()  
24 |
```

Warning

如果文件列表中包含有从零开头的数字范围，请对每个数字分别使用带括号的结构，或者使用?。

计算名为 file-000.csv, file-001.csv, ..., file-999.csv 文件的总行数:

```
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 |
```

插入数据到 test-data.csv.gz 文件:

```
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);
```

从已有的表插入数据到 test-data.csv.gz 文件:

```
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;
```

另请参阅

- [S3 引擎](#)

input

`input(structure)` - 表函数，可以有效地将发送给服务器的数据转换为具有给定结构的数据并将其插入到具有其他结构的表中。

`structure` - 发送到服务器的数据结构的格式 '`column1_name column1_type, column2_name column2_type, ...'`。
例如, '`id UInt32, name String`'。

该函数只能在 `INSERT SELECT` 查询中使用，并且只能使用一次，但在其他方面，行为类似于普通的表函数
(例如，它可以用于子查询等。)。

数据可以像普通 `INSERT` 查询一样发送，并以必须在查询末尾指定的任何可用 [格式](#)
传递 (与普通 `INSERT SELECT` 不同)。

该函数的主要特点是，当服务器从客户端接收数据时，它会同时根据 `SELECT` 子句中的表达式列表将其转换，并插入到目标表中。

不会创建包含所有已传输数据的临时表。

例

- 让 `test` 表具有以下结构 (`a String, b String`)

并且 `data.csv` 中的数据具有不同的结构 (`col1 String, col2 Date, col3 Int32`)。

将数据从 `data.csv` 插入到 `test` 表中，同时进行转换的查询如下所示：

```
$ cat data.csv | clickhouse-client --query="INSERT INTO test SELECT lower(col1), col3 * col3 FROM input('col1 String, col2 Date, col3 Int32') FORMAT CSV";
```

- 如果 `data.csv` 包含与表 `test` 相同结构 `test_structure` 的数据，那么这两个查询是相等的：

```
$ 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

生成具用给定的模式的随机数据。

允许用数据来填充测试表。

支持所有可以存储在表中的数据类型，`LowCardinality` 和 `AggregateFunction` 除外。

```
generateRandom('name TypeName[, name TypeName]...', [, 'random_seed'][, 'max_string_length'][, 'max_array_length']);
```

参数

- `name` — 对应列的名称。
- `TypeName` — 对应列的类型。
- `max_array_length` — 生成数组的最大长度。默认为10。
- `max_string_length` — 生成字符串的最大长度。默认为10。
- `random_seed` — 手动指定随机种子以产生稳定的结果。如果为NULL-种子是随机生成的。

返回值

具有请求模式的表对象。

用法示例

```
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 asterics 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

mysql

允许对存储在远程MySQL服务器上的数据执行SELECT和INSERT查询。

语法

```
mysql('host:port', 'database', 'table', 'user', 'password'[, replace_query, 'on_duplicate_clause']);
```

参数

- `host:port` — MySQL服务器地址.
- `database` — 远程数据库名称.
- `table` — 远程表名称.
- `user` — MySQL用户.
- `password` — 用户密码.
- `replace_query` — 将INSERT INTO查询转换为REPLACE INTO的标志。如果`replace_query=1``，查询被替换。
- `on_duplicate_clause` — 添加ON DUPLICATE KEY `on_duplicate_clause`表达式到INSERT查询。明确规定只能使用`replace_query = 0`，如果你同时设置`replace_query = 1`和`on_duplicate_clause``，ClickHouse将产生异常。

示例：`INSERT INTO t (c1,c2) VALUES ('a', 2) ON DUPLICATE KEY UPDATE c2 = c2 + 1`

`on_duplicate_clause`这里是`UPDATE c2 = c2 + 1`。请查阅MySQL文档，来找到可以和ON DUPLICATE KEY一起使用的`on_duplicate_clause`子句。

简单的 WHERE 子句如`=, !=, >, >=, <, <=`将即时在MySQL服务器上执行。其余的条件和 LIMIT 只有在对MySQL的查询完成后，才会在ClickHouse中执行采样约束。

支持使用|并列进行多副本查询，示例如下：

```
SELECT name FROM mysql(`mysql{1|2|3}:3306`, 'mysql_database', 'mysql_table', 'user', 'password');
```

或

```
SELECT name FROM mysql(`mysql1:3306|mysql2:3306|mysql3:3306`, 'mysql_database', 'mysql_table', 'user', 'password');
```

返回值

与原始MySQL表具有相同列的表对象。

注意

在INSERT查询中为了区分mysql(...)与带有列名列表的表名的表函数，你必须使用关键字FUNCTION或TABLE FUNCTION。查看如下示例。

用法示例

MySQL中的表：

```
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)
```

从ClickHouse中查询数据：

```
SELECT * FROM mysql('localhost:3306', 'test', 'test', 'bayonet', '123')
```

int_id	int_nullable	float	float_nullable
1	NULL	2	NULL

替换和插入：

```
INSERT INTO FUNCTION mysql('localhost:3306', 'test', 'test', 'bayonet', '123', 1) (int_id, float) VALUES (1, 3);
INSERT INTO TABLE FUNCTION mysql('localhost:3306', 'test', 'test', 'bayonet', '123', 0, 'UPDATE int_id = int_id + 1')
(int_id, float) VALUES (1, 4);
SELECT * FROM mysql('localhost:3306', 'test', 'test', 'bayonet', '123');
```

int_id	float
1	3
2	4

另请参阅

- 该 ‘MySQL’ 表引擎
- 使用 MySQL 作为外部字典的来源

remote, remoteSecure

允许您访问远程服务器，而无需创建 Distributed 表。remoteSecure - 与 remote 相同，但是会使用加密链接。

这两个函数都可以在 SELECT 和 INSERT 查询中使用。

语法：

```
remote('addresses_expr', db, table[, 'user'][, 'password'], sharding_key])
remote('addresses_expr', db.table[, 'user'][, 'password'], sharding_key)
remoteSecure('addresses_expr', db, table[, 'user'][, 'password'], sharding_key)
remoteSecure('addresses_expr', db.table[, 'user'][, 'password'], sharding_key)
```

参数

- `addresses_expr` – 代表远程服务器地址的一个表达式。可以只是单个服务器地址。服务器地址可以是 `host:port` 或 `host`。

`host` 可以指定为服务器名称，或是IPV4或IPV6地址。IPv6地址在方括号中指定。

`port` 是远程服务器上的TCP端口。如果省略端口，则 `remote` 使用服务器配置文件中的 `tcp_port`（默认情况为，9000），`remoteSecure` 使用 `tcp_port_secure`（默认情况为，9440）。

IPv6地址需要指定端口。

类型: `String`。

- `db` — 数据库名。类型: `String`。
- `table` — 表名。类型: `String`。
- `user` — 用户名。如果未指定用户，则使用 `default`。类型: `String`。
- `password` — 用户密码。如果未指定密码，则使用空密码。类型: `String`。
- `sharding_key` — 分片键以支持在节点之间分布数据。例如: `insert into remote('127.0.0.1:9000,127.0.0.2', db, table, 'default', rand())`。类型: `UInt32`。

返回值

来自远程服务器的数据集。

用法

使用 `remote` 表函数没有创建一个 `Distributed` 表更优，因为在这种情况下，将为每个请求重新建立服务器连接。此外，如果设置了主机名，则会解析这些名称，并且在使用各种副本时不会计入错误。在处理大量查询时，始终优先创建 `Distributed` 表，不要使用 `remote` 表函数。

该 `remote` 表函数可以在以下情况下是有用的:

- 访问特定服务器进行数据比较、调试和测试。
- 在多个ClickHouse集群之间的用户研究目的的查询。
- 手动发出的不频繁分布式请求。
- 每次重新定义服务器集的分布式请求。

地址

```
example01-01-1
example01-01-1:9000
localhost
127.0.0.1
[::]:9000
[2a02:6b8:0:1111::11]:9000
```

多个地址可以用逗号分隔。在这种情况下，ClickHouse将使用分布式处理，因此它将将查询发送到所有指定的地址（如具有不同数据的分片）。

```
example01-01-1,example01-02-1
```

表达式的一部分可以用大括号指定。前面的示例可以写成如下:

```
example01-0{1,2}-1
```

大括号可以包含由两个点（非负整数）分隔的数字范围。在这种情况下，范围将扩展为生成分片地址的一组值。如果第一个数字以零开头，则使用相同的零对齐形成值。前面的示例可以写成如下：

```
example01-{01..02}-1
```

如果您有多对大括号，它会生成相应集合的直接乘积。

大括号中的地址和部分地址可以用管道符号(|)分隔。在这种情况下，相应的地址集被解释为副本，并且查询将被发送到第一个正常副本。但是，副本将按照当前load_balancing设置的顺序进行迭代。此示例指定两个分片，每个分片都有两个副本：

```
example01-{01..02}-{1|2}
```

生成的地址数由常量限制。目前这是1000个地址。

示例

从远程服务器选择数据：

```
SELECT * FROM remote('127.0.0.1', db.remote_engine_table) LIMIT 3;
```

将远程服务器中的数据插入表中：

```
CREATE TABLE remote_table (name String, value UInt32) ENGINE=Memory;
INSERT INTO FUNCTION remote('127.0.0.1', currentDatabase(), 'remote_table') VALUES ('test', 42);
SELECT * FROM remote_table;
```

字典

字典是一个映射（键 -> 属性），是方便各种类型的参考清单。

ClickHouse支持一些特殊函数配合字典在查询中使用。将字典与函数结合使用比将JOIN操作与引用表结合使用更简单、更有效。

NULL值不能存储在字典中。

ClickHouse支持：

- 内置字典，这些字典具有特定的函数集。
- 插件（外部）字典，这些字典拥有一个函数集。

外部字典

您可以从各种数据源添加自己的字典。字典的数据源可以是本地文本或可执行文件、HTTP(s)资源或其他DBMS。有关详细信息，请参阅“[外部字典的来源](#)”。

ClickHouse：

- 完全或部分存储在RAM中的字典。
- 定期更新字典并动态加载缺失的值。换句话说，字典可以动态加载。
- 允许创建外部字典与xml文件或[DDL查询](#)。

外部字典的配置可以位于一个或多个xml文件中。 配置的路径在指定 `dictionaries_config` 参数。

字典可以在服务器启动或首次使用时加载，具体取决于 `dictionaries_lazy_load` 设置。

该 [字典](#) 系统表包含有关在服务器上配置的字典的信息。对于每个字典，你可以在那里找到：

- 字典的状态。
- 配置参数。
- 度量指标，如为字典分配的RAM量或自成功加载字典以来的查询数量。

字典配置文件具有以下格式：

```
<yandex>
  <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>

</yandex>
```

你可以 [配置](#) 同一文件中的任意数量的字典。

[字典的DDL查询](#) 在服务器配置中不需要任何其他记录。它们允许使用字典作为一流的实体，如表或视图。

注意

您可以通过在一个小字典中描述它来转换小字典的值 `SELECT` 查询（见 [变换](#) 功能）。此功能与外部字典无关。

另请参阅

- [配置外部字典](#)
- [在内存中存储字典](#)
- [字典更新](#)
- [外部字典的来源](#)
- [字典键和字段](#)
- [使用外部字典的函数](#)

配置外部字典

如果使用xml文件配置字典，则比字典配置具有以下结构：

```

<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>

```

相应的 **DDL-查询** 具有以下结构:

```

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_].
- **来源** — Source of the dictionary.
- **布局** — Dictionary layout in memory.
- **结构** — Structure of the dictionary . A key and attributes that can be retrieved by this key.
- **使用寿命** — Frequency of dictionary updates.

在内存中存储字典

有多种方法可以将字典存储在内存中。

我们建议 **平**, **散列** 和 **complex_key_hashed**. 其提供最佳的处理速度。

不建议使用缓存, 因为性能可能较差, 并且难以选择最佳参数。阅读更多的部分 “**缓存**”.

有几种方法可以提高字典性能:

- 调用该函数以使用后的字典 **GROUP BY**.
- 将要提取的属性标记为“注射”。如果不同的属性值对应于不同的键，则称为注射属性。所以当 **GROUP BY** 使用由键获取属性值的函数，此函数会自动取出 **GROUP BY**.

ClickHouse为字典中的错误生成异常。错误示例:

- 无法加载正在访问的字典。
- 查询错误 **cached** 字典

您可以查看外部字典的列表及其状态 **system.dictionaries** 桌子

配置如下所示：

```
<yandex>
  <dictionary>
    ...
    <layout>
      <layout_type>
        <!-- layout settings -->
      </layout_type>
    </layout>
    ...
  </dictionary>
</yandex>
```

相应的 [DDL-查询](#)：

```
CREATE DICTIONARY (...)

  LAYOUT(LAYOUT_TYPE(param value)) -- layout settings
```

在内存中存储字典的方法

- 平
- 散列
- [sparse_hashed](#)
- [缓存](#)
- [直接](#)
- [range_hashed](#)
- [complex_key_hashed](#)
- [complex_key_cache](#)
- [ip_trie](#)

平

字典以平面数组的形式完全存储在内存中。字典使用多少内存？量与最大键的大小（在使用的空间中）成正比。

字典键具有 `UInt64` 类型和值限制为 500,000。如果在创建字典时发现较大的键，ClickHouse将引发异常，不会创建字典。

支持所有类型的来源。更新时，数据（来自文件或表）将完整读取。

此方法在存储字典的所有可用方法中提供了最佳性能。

配置示例：

```
<layout>
  <flat />
</layout>
```

或

```
LAYOUT(FLAT())
```

散列

该字典以哈希表的形式完全存储在内存中。字典中可以包含任意数量的带有任意标识符的元素，在实践中，键的数量可以达到数千万项。

支持所有类型的来源。更新时，数据（来自文件或表）将完整读取。

配置示例：

```
<layout>
<hashed />
</layout>
```

或

```
LAYOUT(HASHED())
```

sparse_hashed

类似于 `hashed`，但使用更少的内存，有利于更多的CPU使用率。

配置示例：

```
<layout>
<sparse_hashed />
</layout>
```

```
LAYOUT(SPARSE_HASHED())
```

complex_key_hashed

这种类型的存储是用于复合 **键**。类似于 `hashed`。

配置示例：

```
<layout>
<complex_key_hashed />
</layout>
```

```
LAYOUT(COMPLEX_KEY_HASHED())
```

range_hashed

字典以哈希表的形式存储在内存中，其中包含有序范围及其相应值的数组。

此存储方法的工作方式与散列方式相同，除了键之外，还允许使用日期/时间（任意数字类型）范围。

示例：该表格包含每个广告客户的折扣，格式为：

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

要对日期范围使用示例，请定义 `range_min` 和 `range_max` 中的元素 [结构](#)。这些元素必须包含元素 `name` 和 `type`（如果 `type` 没有指定，则默认类型将使用 `-Date`）。`type` 可以是任何数字类型（`Date/DateTime/UInt64/Int32/others`）。

示例：

```
<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>
...
...
```

或

```
CREATE DICTIONARY somedict (
    id UInt64,
    first Date,
    last Date
)
PRIMARY KEY id
LAYOUT(RANGE_HASHED())
RANGE(MIN first MAX last)
```

要使用这些字典，您需要将附加参数传递给 `dictGetT` 函数，为其选择一个范围：

```
dictGetT('dict_name', 'attr_name', id, date)
```

此函数返回指定的值 `ids` 和包含传递日期的日期范围。

算法的详细信息：

- 如果 `id` 未找到或范围未找到 `id`，它返回字典的默认值。
- 如果存在重叠范围，则可以使用任意范围。
- 如果范围分隔符是 `NULL` 或无效日期（如 `1900-01-01` 或 `2039-01-01`），范围保持打开状态。范围可以在两侧打开。

配置示例：

```

<yandex>
  <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>
</yandex>

```

或

```

CREATE DICTIONARY somedict(
  Abcdef UInt64,
  StartTimeStamp UInt64,
  EndTimeStamp UInt64,
  XXXType String DEFAULT ''
)
PRIMARY KEY Abcdef
RANGE(MIN StartTimeStamp MAX EndTimeStamp)

```

缓存

字典存储在具有固定数量的单元格的缓存中。 这些单元格包含经常使用的元素。

搜索字典时，首先搜索缓存。 对于每个数据块，所有在缓存中找不到或过期的密钥都从源请求，使用 `SELECT attrs... FROM db.table WHERE id IN (k1, k2, ...)`。 然后将接收到的数据写入高速缓存。

对于缓存字典，过期 [使用寿命](#) 可以设置高速缓存中的数据。 如果更多的时间比 `lifetime` 自从在单元格中加载数据以来，单元格的值不被使用，并且在下次需要使用时重新请求它。

这是存储字典的所有方法中最不有效的。 缓存的速度在很大程度上取决于正确的设置和使用场景。 缓存类型字典只有在命中率足够高（推荐99%或更高）时才能表现良好。 您可以查看平均命中率 `system.dictionaries` 桌子

要提高缓存性能，请使用以下子查询 `LIMIT`，并从外部调用字典函数。

支持 [来源](#):MySQL的,ClickHouse的,可执行文件,HTTP.

设置示例：

```

<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>

```

或

```
LAYOUT(CACHE(SIZE_IN_CELLS 1000000000))
```

设置足够大的缓存大小。你需要尝试选择细胞的数量：

1. 设置一些值。
2. 运行查询，直到缓存完全满。
3. 使用评估内存消耗 `system.dictionaries` 桌子
4. 增加或减少单元数，直达到所需的内存消耗。

警告

不要使用ClickHouse作为源，因为处理随机读取的查询速度很慢。

complex_key_cache

这种类型的存储是用于复合 [键](#)，类似于 `cache`。

直接

字典不存储在内存中，并且在处理请求期间直接转到源。

字典键具有 `UInt64` 类型。

所有类型的 [来源](#)，除了本地文件，支持。

配置示例：

```
<layout>
  <direct />
</layout>
```

或

```
LAYOUT(DIRECT())
```

ip_trie

这种类型的存储用于将网络前缀（IP地址）映射到ASN等元数据。

示例：该表包含网络前缀及其对应的AS号码和国家代码：

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

使用此类布局时，结构必须具有复合键。

示例：

```
<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>
```

或

```
CREATE DICTIONARY somedict (
  prefix String,
  asn UInt32,
  cca2 String DEFAULT '??'
)
PRIMARY KEY prefix
```

该键必须只有一个包含允许的IP前缀的字符串类型属性。还不支持其他类型。

对于查询，必须使用相同的函数 (`dictGetT` 与元组) 至于具有复合键的字典：

```
dictGetT('dict_name', 'attr_name', tuple(ip))
```

该函数采用任一 `UInt32` 对于IPv4，或 `FixedString(16)` 碌莽祿Ipv6拢IPv6：

```
dictGetString('prefix', 'asn', tuple(IPv6StringToNum('2001:db8::1')))
```

还不支持其他类型。该函数返回与此IP地址对应的前缀的属性。如果有重叠的前缀，则返回最具体的前缀。

数据存储在一个 `trie`。它必须完全适合RAM。

字典更新

ClickHouse定期更新字典。完全下载字典的更新间隔和缓存字典的无效间隔在 `<lifetime>` 在几秒钟内标记。

字典更新（除首次使用的加载之外）不会阻止查询。在更新期间，将使用旧版本的字典。如果在更新过程中发生错误，则将错误写入服务器日志，并使用旧版本的字典继续查询。

设置示例：

```
<dictionary>
...
<lifetime>300</lifetime>
...
</dictionary>
```

CREATE DICTIONARY (...)

...
LIFETIME(300)
...

设置 `<lifetime>0</lifetime>` (`LIFETIME(0)`) 防止字典更新。

您可以设置升级的时间间隔，ClickHouse将在此范围内选择一个统一的随机时间。为了在大量服务器上升级时分配字典源上的负载，这是必要的。

设置示例：

```
<dictionary>
...
<lifetime>
  <min>300</min>
  <max>360</max>
</lifetime>
...
</dictionary>
```

或

LIFETIME(MIN 300 MAX 360)

如果 `<min>0</min>` 和 `<max>0</max>`，ClickHouse不会按超时重新加载字典。

在这种情况下，如果字典配置文件已更改，ClickHouse可以更早地重新加载字典 `SYSTEM RELOAD DICTIONARY` 命令被执行。

升级字典时，ClickHouse服务器根据字典的类型应用不同的逻辑 [来源](#)：

升级字典时，ClickHouse服务器根据字典的类型应用不同的逻辑 [来源](#)：

- 对于文本文件，它检查修改的时间。如果时间与先前记录的时间不同，则更新字典。
- 对于MyISAM表，修改的时间使用检查 `SHOW TABLE STATUS` 查询。
- 默认情况下，每次都会更新来自其他来源的字典。

对于MySQL（InnoDB），ODBC和ClickHouse源代码，您可以设置一个查询，只有在字典真正改变时才会更新字典，而不是每次都更新。为此，请按照下列步骤操作：

- 字典表必须具有在源数据更新时始终更改的字段。
- 源的设置必须指定检索更改字段的查询。ClickHouse服务器将查询结果解释为一行，如果此行相对于其以前的状态发生了更改，则更新字典。指定查询 `<invalidate_query>` 字段中的设置 [来源](#)。

设置示例：

```
<dictionary>
...
<odbc>
...
<invalidate_query>SELECT update_time FROM dictionary_source where id = 1</invalidate_query>
</odbc>
...
</dictionary>
```

或

```
...
SOURCE(ODBC(... invalidate_query 'SELECT update_time FROM dictionary_source where id = 1'))
...
```

外部字典的来源

外部字典可以从许多不同的来源连接。

如果使用 `xml-file` 配置字典，则配置如下所示：

```
<yandex>
<dictionary>
...
<source>
<source_type>
<!-- Source configuration --&gt;
&lt;/source_type&gt;
&lt;/source&gt;
...
&lt;/dictionary&gt;
...
&lt;/yandex&gt;</pre>
```

在情况下 **DDL-查询**，相等的配置将看起来像：

```
CREATE DICTIONARY dict_name (...)

...
SOURCE(SOURCE_TYPE(param1 val1 ... paramN valN)) -- Source configuration
...
```

源配置在 `source` 科。

对于源类型 **本地文件**, **可执行文件**, **HTTP(s)**, **ClickHouse**
可选设置：

```
<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>
```

或

```
SOURCE(FILE(path './user_files/os.tsv' format 'TabSeparated'))
SETTINGS(format_csv_allow_single_quotes = 0)
```

来源类型 (source_type):

- 本地文件
- 可执行文件
- HTTP(s)
- DBMS
 - ODBC
 - MySQL
 - ClickHouse
 - MongoDB
 - Redis

本地文件

设置示例:

```
<source>
  <file>
    <path>/opt/dictionaries/os.tsv</path>
    <format>TabSeparated</format>
  </file>
</source>
```

或

```
SOURCE(FILE(path './user_files/os.tsv' format 'TabSeparated'))
```

设置字段:

- path – The absolute path to the file.
- format – The file format. All the formats described in “[格式](#)” 支持。

可执行文件

使用可执行文件取决于 [字典如何存储在内存中](#)。如果字典存储使用 cache 和 complex_key_cache，ClickHouse通过向可执行文件的STDIN发送请求来请求必要的密钥。否则，ClickHouse将启动可执行文件并将其输出视为字典数据。

设置示例:

```
<source>
  <executable>
    <command>cat /opt/dictionaries/os.tsv</command>
    <format>TabSeparated</format>
  </executable>
</source>
```

或

```
SOURCE(EXECUTABLE(command 'cat /opt/dictionaries/os.tsv' format 'TabSeparated'))
```

设置字段:

- `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 “[格式](#)” 支持。

Http(s)

使用HTTP (s) 服务器取决于 [字典如何存储在内存中](#). 如果字典存储使用 `cache` 和 `complex_key_cache` , ClickHouse通过通过发送请求请求必要的密钥 `POST` 方法。

设置示例:

```
<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>
```

或

```
SOURCE(HTTP(
  url 'http://[::1]/os.tsv'
  format 'TabSeparated'
  credentials(user 'user' password 'password')
  headers(header(name 'API-KEY' value 'key'))
))
```

为了让ClickHouse访问HTTPS资源，您必须 [配置openSSL](#) 在服务器配置中。

设置字段:

- `url` – The source URL.
- `format` – The file format. All the formats described in “[格式](#)” 支持。
- `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

您可以使用此方法连接具有ODBC驱动程序的任何数据库。

设置示例:

```
<source>
  <odbc>
    <db>DatabaseName</db>
    <table>ShemaName.TableName</table>
    <connection_string>DSN=some_parameters</connection_string>
    <invalidate_query>SQL_QUERY</invalidate_query>
  </odbc>
</source>
```

或

```
SOURCE(ODBC(
  db 'DatabaseName'
  table 'SchemaName.TableName'
  connection_string 'DSN=some_parameters'
  invalidate_query 'SQL_QUERY'
))
```

设置字段:

- db – Name of the database. Omit it if the database name is set in the `<connection_string>` 参数。
- 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 [更新字典](#).

ClickHouse接收来自ODBC-driver的引用符号，并将查询中的所有设置引用到driver，因此有必要根据数据库中的表名大小写设置表名。

如果您在使用Oracle时遇到编码问题，请参阅相应的 [FAQ](#) 文章。

ODBC字典功能的已知漏洞

注意

通过ODBC驱动程序连接参数连接到数据库时 `Servername` 可以取代。在这种情况下，值 `USERNAME` 和 `PASSWORD` 从 `odbc.ini` 被发送到远程服务器，并且可能会受到损害。

不安全使用示例

让我们为PostgreSQL配置unixODBC。的内容 `/etc/odbc.ini`:

```
[gregtest]
Driver = /usr/lib/pgsqlodbc.so
Servername = localhost
PORT = 5432
DATABASE = test_db
##OPTION = 3
USERNAME = test
PASSWORD = test
```

如果然后进行查询，例如

```
SELECT * FROM odbc('DSN=gregtest;Servername=some-server.com', 'test_db');
```

ODBC驱动程序将发送的值 `USERNAME` 和 `PASSWORD` 从 `odbc.ini` 到 `some-server.com`.

连接Postgresql的示例

Ubuntu操作系统。

为PostgreSQL安装unixODBC和ODBC驱动程序:

```
$ sudo apt-get install -y unixodbc odbcinst odbc-postgresql
```

配置 /etc/odbc.ini (或 ~/.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中的字典配置:

```
<yandex>
  <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>
</yandex>
```

或

```
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)
```

您可能需要编辑 `odbc.ini` 使用驱动程序指定库的完整路径 `DRIVER=/usr/local/lib/psqlodbcw.so.`

连接MS SQL Server的示例

Ubuntu操作系统。

安装驱动程序:::

```
$ sudo apt-get install tdsodbc freetds-bin sqsh
```

配置驱动程序:

```
$ cat /etc/freetds/freetds.conf
...
[MSSQL]
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
...
[MSSQL]
Description = FreeTDS
Driver      = FreeTDS
Servername  = MSSQL
Database    = test
UID        = test
PWD        = test
Port       = 1433
```

在ClickHouse中配置字典:

```

<yandex>
  <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>
</yandex>

```

或

```

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

设置示例：

```

<source>
  <mysql>
    <port>3306</port>
    <user>clickhouse</user>
    <password>qwerty</password>
    <replica>
      <host>example01-1</host>
      <priorty>1</priorty>
    </replica>
    <replica>
      <host>example01-2</host>
      <priorty>1</priorty>
    </replica>
    <replica>
      <host>example01-3</host>
      <priorty>1</priorty>
    </replica>
    <db>db_name</db>
    <table>table_name</table>
    <where>id=10</where>
    <invalidate_query>SQL_QUERY</invalidate_query>
  </mysql>
</source>

```

或

```
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'
))
```

设置字段：

- **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` 例如，mysql中的子句, `id > 10 AND id < 20`. 可选参数。
- **invalidate_query** – Query for checking the dictionary status. Optional parameter. Read more in the section [更新字典](#).

MySQL可以通过套接字在本地主机上连接。要做到这一点，设置 `host` 和 `socket`.

设置示例：

```
<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>
```

或

```
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'
))
```

ClickHouse

设置示例：

```
<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>
```

或

```
SOURCE(CLICKHOUSE(
    host 'example01-01-1'
    port 9000
    user 'default'
    password ""
    db 'default'
    table 'ids'
    where 'id=10'
))
```

设置字段：

- **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 **分布** 表并在后续配置中输入它。
- **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 [更新字典](#).

Mongodb

设置示例：

```
<source>
  <mongodb>
    <host>localhost</host>
    <port>27017</port>
    <user></user>
    <password></password>
    <db>test</db>
    <collection>dictionary_source</collection>
  </mongodb>
</source>
```

或

```
SOURCE(MONGO(
  host 'localhost'
  port 27017
  user ""
  password ""
  db 'test'
  collection 'dictionary_source'
))
```

设置字段：

- **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

设置示例：

```
<source>
  <redis>
    <host>localhost</host>
    <port>6379</port>
    <storage_type>simple</storage_type>
    <db_index>0</db_index>
  </redis>
</source>
```

或

```
SOURCE(REDIS(
  host 'localhost'
  port 6379
  storage_type 'simple'
  db_index 0
))
```

设置字段：

- **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` 适用于简单源和散列单键源, `hash_map` 用于具有两个键的散列源。不支持具有复杂键的范围源和缓存源。可以省略, 默认值为 `simple`.
- `db_index` – The specific numeric index of Redis logical database. May be omitted, default value is 0.

字典键和字段

该 `<structure>` 子句描述可用于查询的字典键和字段。

XML描述:

```
<dictionary>
  <structure>
    <id>
      <name>Id</name>
    </id>

    <attribute>
      <!-- Attribute parameters -->
    </attribute>

    ...
  </structure>
</dictionary>
```

属性在元素中描述:

- `<id>` — 键列.
- `<attribute>` — 数据列. 可以有多个属性。

DDL查询:

```
CREATE DICTIONARY dict_name (
  Id UInt64,
  -- attributes
)
PRIMARY KEY Id
...
```

查询正文中描述了属性:

- `PRIMARY KEY` — 键列
- `AttrName AttrType` — 数据列. 可以有多个属性。

键

ClickHouse 支持以下类型的键:

- 数字键。 `UInt64`. 在定义 `<id>` 标记或使用 `PRIMARY KEY` 关键字。
- 复合密钥。 组不同类型的值。 在标签中定义 `<key>` 或 `PRIMARY KEY` 关键字。

Xml结构可以包含 `<id>` 或 `<key>`. DDL-查询必须包含单个 `PRIMARY KEY`.

警告

不能将键描述为属性。

数字键

类型: UInt64.

配置示例:

```
<id>
  <name>Id</name>
</id>
```

配置字段:

- name - The name of the column with keys.

对于DDL-查询:

```
CREATE DICTIONARY (
  Id UInt64,
  ...
)
PRIMARY KEY Id
...
```

- PRIMARY KEY - The name of the column with keys.

复合密钥

键可以是一个 tuple 从任何类型的字段。该 布局 在这种情况下，必须是 complex_key_hashed 或 complex_key_cache.

提示

复合键可以由单个元素组成。例如，这使得可以使用字符串作为键。

键结构在元素中设置 `<key>`. 键字段的格式与字典的格式相同 [属性](#). 示例:

```
<structure>
  <key>
    <attribute>
      <name>field1</name>
      <type>String</type>
    </attribute>
    <attribute>
      <name>field2</name>
      <type>UInt32</type>
    </attribute>
    ...
  </key>
  ...
```

或

```
CREATE DICTIONARY (
  field1 String,
  field2 String
  ...
)
PRIMARY KEY field1, field2
...
```

对于查询 `dictGet*` 函数中，一个元组作为键传递。示例: `dictGetString('dict_name', 'attr_name', tuple('string for field1', num_for_field2))`.

属性

配置示例:

```
<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>
```

或

```
CREATE DICTIONARY somename (
    Name ClickHouseDataType DEFAULT '' EXPRESSION rand64() HIERARCHICAL INJECTIVE IS_OBJECT_ID
)
```

配置字段:

标签	产品描述	必填项
<code>name</code>	列名称。	是
<code>type</code>	<code>ClickHouse</code> 数据类型。 <code>ClickHouse</code> 尝试将字典中的值转换为指定的数据类型。例如，对于MySQL，该字段可能是 <code>TEXT</code> , <code>VARCHAR</code> , 或 <code>BLOB</code> 在MySQL源表中，但它可以上传为 <code>String</code> 在克里克豪斯 可为空 不支持。	是
<code>null_value</code>	非现有元素的默认值。 在示例中，它是一个空字符串。你不能使用 <code>NULL</code> 在这个领域。	是
<code>expression</code>	<code>表达式</code> <code>ClickHouse</code> 对该值执行。 表达式可以是远程SQL数据库中的列名。因此，您可以使用它为远程列创建别名。 默认值：无表达式。	非也。
<code>hierarchical</code>	如果 <code>true</code> ，该属性包含当前键的父键值。看 分层字典 . 默认值: <code>false</code> .	非也。
<code>injective</code>	标志，显示是否 <code>id -> attribute</code> 图像是 注射 . 如果 <code>true</code> ， <code>ClickHouse</code> 可以自动放置后 <code>GROUP BY</code> 子句注入字典的请求。通常它显着减少了这种请求的数量。 默认值: <code>false</code> .	非也。

标签	产品描述	必填项
is_object_id	显示是否通过以下方式对MongoDB文档执行查询的标志 ObjectId. 默认值: <code>false</code> .	非也。

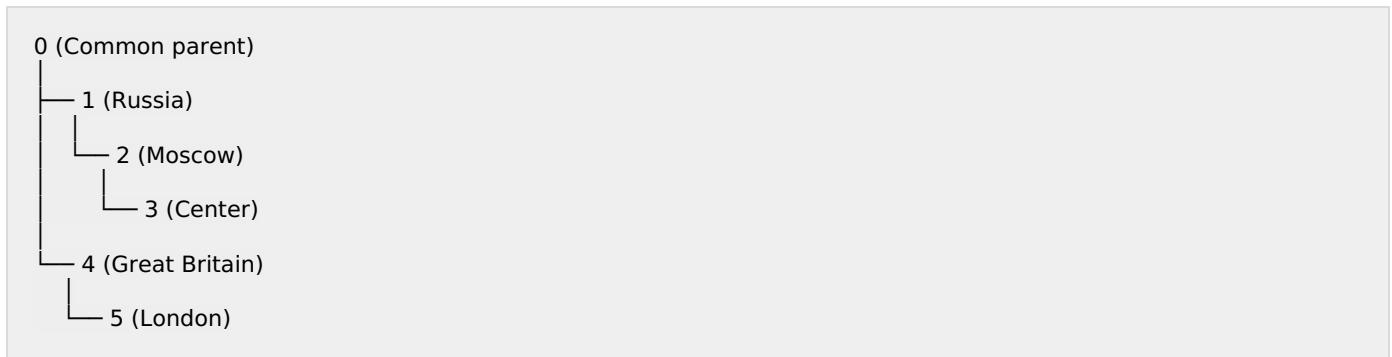
另请参阅

- [使用外部字典的函数.](#)

分层字典

ClickHouse支持分层字典与 [数字键](#).

看看下面的层次结构:



这种层次结构可以表示为下面的字典表。

region_id	parent_region	region_name
1	0	俄罗斯
2	1	莫斯科
3	2	中心
4	0	英国
5	4	伦敦

此表包含一列 `parent_region` 包含该元素的最近父项的键。

ClickHouse支持 [等级](#) 属性为 [外部字典](#) 属性。此属性允许您配置类似于上述的分层字典。

该 [独裁主义](#) 函数允许您获取元素的父链。

对于我们的例子，`dictionary`的结构可以是以下内容:

```

<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(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(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.

内部字典

ClickHouse包含用于处理地理数据库的内置功能。

这使您可以:

- 使用区域的ID以所需语言获取其名称。
- 使用区域ID获取城市、地区、联邦区、国家或大陆的ID。

- 检查一个区域是否属于另一个区域。
- 获取父区域链。

所有功能支持“translocality,”能够同时使用不同的角度对区域所有权。有关详细信息，请参阅部分“[Functions for working with Yandex.Metrica dictionaries](#)”。

在默认包中禁用内部字典。

要启用它们，请取消注释参数 `path_to_regions_hierarchy_file` 和 `path_to_regions_names_files` 在服务器配置文件中。

Geobase从文本文件加载。

将 `regions_hierarchy*.txt` 文件到 `path_to_regions_hierarchy_file` 目录。此配置参数必须包含指向 `regions_hierarchy.txt` 文件（默认区域层次结构）和其他文件 (`regions_hierarchy_ua.txt`) 必须位于同一目录中。

把 `regions_names_*.txt` 在文件 `path_to_regions_names_files` 目录。

您也可以自己创建这些文件。文件格式如下:

`regions_hierarchy*.txt` : TabSeparated (无标题) , 列:

- 地区ID (UInt32)
- 父区域ID (UInt32)
- 区域类型 (UInt8) : 1-大陆, 3-国家, 4-联邦区, 5-地区, 6-城市;其他类型没有价值
- 人口 (UInt32) — optional column

`regions_names_*.txt` : TabSeparated (无标题) , 列:

- 地区ID (UInt32)
- 地区名称 (String) — Can't contain tabs or line feeds, even escaped ones.

平面阵列用于存储在RAM中。出于这个原因，Id不应该超过一百万。

字典可以在不重新启动服务器的情况下更新。但是，不会更新可用字典集。

对于更新，将检查文件修改时间。如果文件已更改，则更新字典。

检查更改的时间间隔在 `builtin_dictionaries_reload_interval` 参数。

字典更新（首次使用时加载除外）不会阻止查询。在更新期间，查询使用旧版本的字典。如果在更新过程中发生错误，则将错误写入服务器日志，并使用旧版本的字典继续查询。

我们建议定期使用**geobase**更新字典。在更新期间，生成新文件并将其写入单独的位置。一切准备就绪后，将其重命名为服务器使用的文件。

还有与操作系统标识符和Yandex的工作功能。**Metrica**搜索引擎，但他们不应该被使用。

数据类型

ClickHouse可以在数据表中存储多种数据类型。

本节描述 **ClickHouse** 支持的数据类型，以及使用或者实现它们时（如果有的话）的注意事项。

你可以在系统表 `system.data_type_families` 中检查数据类型名称是否区分大小写。

UUID

通用唯一标识符(UUID)是一个16字节的数字，用于标识记录。有关UUID的详细信息，参见[维基百科](#)。

UUID类型值的示例如下:

61f0c404-5cb3-11e7-907b-a6006ad3dba0

如果在插入新记录时未指定UUID列的值，则UUID值将用零填充：

```
00000000-0000-0000-0000-000000000000
```

如何生成

要生成UUID值，ClickHouse提供了 [generateUUIDv4](#) 函数。

用法示例

示例1

这个例子演示了创建一个具有UUID类型列的表，并在表中插入一个值。

```
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

示例2

在这个示例中，插入新记录时未指定UUID列的值。

```
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

限制

UUID数据类型只支持 [字符串](#) 数据类型也支持的函数(比如, [min](#), [max](#), 和 [count](#))。

算术运算不支持UUID数据类型 (例如, [abs](#)) 或聚合函数, 例如 [sum](#) 和 [avg](#).

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](#)

Datetime64

此类型允许以日期 (date) 加时间 (time) 的形式来存储一个时刻的时间值，具有定义的亚秒精度

时间刻度大小（精度）： $10^{-\text{精度}}$ 秒

语法：

```
DateTime64(precision, [timezone])
```

在内部，此类型以 `Int64` 类型将数据存储为自 Linux 纪元开始 (1970-01-01 00:00:00 UTC) 的时间刻度数 (ticks)。时间刻度的分辨率由 `precision` 参数确定。此外，`DateTime64` 类型可以像存储其他数据列一样存储时区信息，时区会影响 `DateTime64` 类型的值如何以文本格式显示，以及如何解析以字符串形式指定的时间数据 ('2020-01-01 05:00:01.000')。时区不存储在表的行中（也不在 `resultset` 中），而是存储在列的元数据中。详细信息请参考 [DateTime](#) 数据类型。

示例

1. 创建一个具有 `DateTime64` 类型列的表，并向其中插入数据：

```
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

- 将日期时间作为integer类型插入时，它会被视为适当缩放的Unix时间戳(UTC)。1546300800000（精度为3）表示'2019-01-01 00:00:00' UTC. 不过，因为 timestamp 列指定了 Europe/Moscow (UTC+3) 的时区，当作为字符串输出时，它将显示为 '2019-01-01 03:00:00'
- 当把字符串作为日期时间插入时，它会被赋予时区信息。'2019-01-01 00:00:00' 将被认为处于 Europe/Moscow 时区并被存储为 1546290000000.

2. 过滤 DateTime64 类型的值

```
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

与 DateTime 不同，DateTime64 类型的值不会自动从 String 类型的值转换过来

3. 获取 DateTime64 类型值的时区信息：

```
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. 时区转换

```
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

另请参阅

- [类型转换函数](#)
- [用于处理日期和时间的函数](#)
- [用于处理数组的函数](#)
- [date_time_input_format 配置](#)
- [date_time_output_format 配置](#)

- `timezone` 服务器配置参数
- 用于处理日期和时间的算子
- `Date` 数据类型
- `DateTime` 数据类型

低基数类型

把其它数据类型转变为字典编码类型。

语法

```
LowCardinality(data_type)
```

参数

- `data_type` — `String`, `FixedString`, `Date`, `DateTime`, 包括数字类型, 但是`Decimal`除外。对一些数据类型来说, `LowCardinality` 并不高效, 详查[allow_suspicious_low_cardinality_types](#)设置描述。

描述

`LowCardinality` 是一种改变数据存储和数据处理方法的概念。ClickHouse会把 `LowCardinality` 所在的列进行[dictionary coding](#)。对很多应用来说, 处理字典编码的数据可以显著的增加[SELECT](#)查询速度。

使用 `LowCardinality` 数据类型的效率依赖于数据的多样性。如果一个字典包含少于10000个不同的值, 那么ClickHouse可以进行更高效的数据存储和处理。反之如果字典多于10000, 效率会表现的更差。

当使用字符类型的时候, 可以考虑使用 `LowCardinality` 代替[Enum](#)。`LowCardinality` 通常更加灵活和高效。

例子

创建一个 `LowCardinality` 类型的列：

```
CREATE TABLE lc_t
(
    `id` UInt16,
    `strings` LowCardinality(String)
)
ENGINE = MergeTree()
ORDER BY id
```

相关的设置和函数

设置:

- `low_cardinality_max_dictionary_size`
- `low_cardinality_use_single_dictionary_for_part`
- `low_cardinality_allow_in_native_format`
- `allow_suspicious_low_cardinality_types`

函数:

- `toLowCardinality`

参考

- 高效低基数类型.
- 使用低基数类型减少ClickHouse的存储成本 - 来自Instana工程师的分享.
- 字符优化 (俄语视频分享). 英语分享.

域

Domain类型是特定实现的类型，它总是与某个现存的基础类型保持二进制兼容的同时添加一些额外的特性，以能够在维持磁盘数据不变的情况下使用这些额外的特性。目前ClickHouse暂不支持自定义domain类型。

如果你可以在一个地方使用与Domain类型二进制兼容的基础类型，那么在相同的地方您也可以使用Domain类型，例如：

- 使用Domain类型作为表中列的类型
- 对Domain类型的列进行读/写数据
- 如果与Domain二进制兼容的基础类型可以作为索引，那么Domain类型也可以作为索引
- 将Domain类型作为参数传递给函数使用
- 其他

Domains的额外特性

- 在执行SHOW CREATE TABLE 或 DESCRIBE TABLE时，其对应的列总是展示为Domain类型的名称
- 在INSERT INTO domain_table(domain_column) VALUES(...)中输入数据总是以更人性化的格式进行输入
- 在SELECT domain_column FROM domain_table中数据总是以更人性化的格式输出
- 在INSERT INTO domain_table FORMAT CSV ...中，实现外部源数据以更人性化的格式载入

Domains类型的限制

- 无法通过ALTER TABLE将基础类型的索引转换为Domain类型的索引。
- 当从其他列或表插入数据时，无法将string类型的值隐式地转换为Domain类型的值。
- 无法对存储为Domain类型的值添加约束。

IPv4

IPv4是与UInt32类型保持二进制兼容的Domain类型，其用于存储IPv4地址的值。它提供了更为紧凑的二进制存储的同时保持识别可读性更加友好的输入输出格式。

基本使用

```
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				

同时您也可以使用IPv4类型的列作为主键：

```
CREATE TABLE hits (url String, from IPv4) ENGINE = MergeTree() ORDER BY from;
```

在写入与查询时，IPv4类型能够识别可读性更加友好的输入输出格式：

```
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

同时它提供更为紧凑的二进制存储格式：

```
SELECT toTypeName(from), hex(from) FROM hits LIMIT 1;
```

toTypeName(from)	hex(from)
IPv4	B7F7E83A

不可隐式转换为除UInt32以外的其他类型。如果要将IPv4类型的值转换成字符串，你可以使用IPv4NumToString()显示的进行转换：

```
SELECT toTypeName(s), IPv4NumToString(from) as s FROM hits LIMIT 1;
```

toTypeName(IPv4NumToString(from))	s
String	183.247.232.58

或可以使用CAST将它转换为UInt32类型：

```
SELECT toTypeName(i), CAST(from as UInt32) as i FROM hits LIMIT 1;
```

toTypeName(CAST(from, 'UInt32'))	i
UInt32	3086477370

IPv6

IPv6是与FixedString(16)类型保持二进制兼容的Domain类型，其用于存储IPv6地址的值。它提供了更为紧凑的二进制存储的同时支持识别可读性更加友好的输入输出格式。

基本用法

```
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				

同时您也可以使用 IPv6 类型的列作为主键：

```
CREATE TABLE hits (url String, from IPv6) ENGINE = MergeTree() ORDER BY from;
```

在写入与查询时，IPv6 类型能够识别可读性更加友好的输入输出格式：

```
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

同时它提供更为紧凑的二进制存储格式：

```
SELECT toTypeName(from), hex(from) FROM hits LIMIT 1;
```

toTypeName(from)	hex(from)
IPv6	200144C8012926320033000002520002

不可隐式转换为除 `FixedString(16)` 以外的其他类型。如果要将 IPv6 类型的值转换成字符串，你可以使用 `IPv6NumToString()` 显示的进行转换：

```
SELECT toTypeName(s), IPv6NumToString(from) as s FROM hits LIMIT 1;
```

toTypeName(IPv6NumToString(from))	s
String	2001:44c8:129:2632:33:0:252:2

或使用 `CAST` 将其转换为 `FixedString(16)`：

```
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	toTypeName(pg)
[(20,20),(50,20),(50,50),(20,50)],[(30,30),(50,50),(50,30)]	Polygon

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, typeName(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

AggregateFunction(name, types_of_arguments...)

聚合函数的中间状态，可以通过聚合函数名称加-`State`后缀的形式得到它。与此同时，当您需要访问该类型的最终状态数据时，您需要以相同的聚合函数名加-`Merge`后缀的形式来得到最终状态数据。

`AggregateFunction` — 参数化的数据类型。

参数

■ 聚合函数名

如果函数具备多个参数列表，请在此处指定其他参数列表中的值。

■ 聚合函数参数的类型

示例

```
CREATE TABLE t
(
    column1 AggregateFunction(uniq, UInt64),
    column2 AggregateFunction(anyIf, String, UInt8),
    column3 AggregateFunction(quantiles(0.5, 0.9), UInt64)
) ENGINE = ...
```

上述中的 `uniq`，`anyIf`（任何+如果）以及 `分位数` 都为 ClickHouse 中支持的聚合函数。

使用指南

数据写入

当需要写入数据时，您需要将数据包含在 `INSERT SELECT` 语句中，同时对于 `AggregateFunction` 类型的数据，您需要使用对应的以 `-State` 为后缀的函数进行处理。

函数使用示例

```
uniqState(UserID)
quantilesState(0.5, 0.9)(SendTiming)
```

不同于 `uniq` 和 `quantiles` 函数返回聚合结果的最终值，以 `-State` 后缀的函数总是返回 `AggregateFunction` 类型的数据的中间状态。

对于 `SELECT` 而言，`AggregateFunction` 类型总是以特定的二进制形式展现在所有的输出格式中。例如，您可以使用 `SELECT` 语句将函数的状态数据转储为 `TabSeparated` 格式的同时使用 `INSERT` 语句将数据转储回去。

数据查询

当从 `AggregatingMergeTree` 表中查询数据时，对于 `AggregateFunction` 类型的字段，您需要使用以 `-Merge` 为后缀的相同聚合函数来聚合数据。对于非 `AggregateFunction` 类型的字段，请将它们包含在 `GROUP BY` 子句中。

以 `-Merge` 为后缀的聚合函数，可以将多个 `AggregateFunction` 类型的中间状态组合计算为最终的聚合结果。

例如，如下的两个查询返回的结果总是一致：

```
SELECT uniq(UserID) FROM table

SELECT uniqMerge(state) FROM (SELECT uniqState(UserID) AS state FROM table GROUP BY RegionID)
```

使用示例

请参阅 [AggregatingMergeTree](#) 的说明

Decimal(P,S), Decimal32(S), Decimal64(S), Decimal128(S)

有符号的定点数，可在加、减和乘法运算过程中保持精度。对于除法，最低有效数字会被丢弃（不舍入）。

参数

- P - 精度。有效范围 : [1:38] , 决定可以有多少个十进制数字 (包括分数) 。
- S - 规模。有效范围 : [0 : P] , 决定数字的小数部分中包含的小数位数。

对于不同的 P 参数值 Decimal 表示 , 以下例子都是同义的 :

-P从[1:9]-对于Decimal32(S)
-P从[10:18]-对于Decimal64(小号)
-P从[19:38]-对于Decimal128 (S)

十进制值范围

- Decimal32(S) - (-1 * $10^{(9-S)}$, $1 \times 10^{(9-S)}$)
- Decimal64(S) - (-1 * $10^{(18-S)}$, $1 \times 10^{(18-S)}$)
- Decimal128(S) - (-1 * $10^{(38-S)}$, $1 \times 10^{(38-S)}$)

例如 , Decimal32(4) 可以表示 -99999.9999 至 99999.9999 的数值 , 步长为 0.0001 。

内部表示方式

数据采用与自身位宽相同的有符号整数存储。这个数在内存中实际范围会高于上述范围 , 从 String 转换到十进制数的时候会做对应的检查。

由于现代CPU不支持128位数字 , 因此 Decimal128 上的操作由软件模拟。所以 Decimal128 的运算速度明显慢于 Decimal32/Decimal64 。

运算和结果类型

对 Decimal 的二进制运算导致更宽的结果类型 (无论参数的顺序如何) 。

- Decimal64(S1) <op> Decimal32(S2) -> Decimal64(S)
- Decimal128(S1) <op> Decimal32(S2) -> Decimal128(S)
- Decimal128(S1) <op> Decimal64(S2) -> Decimal128(S)

精度变化的规则 :

- 加法 , 减法 : S = max(S1, S2) 。
- 乘法 : S = S1 + S2 。
- 除法 : S = S1 。

对于 Decimal 和整数之间的类似操作 , 结果是与参数大小相同的十进制。

未定义 Decimal 和 Float32/Float64 之间的函数。要执行此类操作 , 您可以使用 : toDecimal32 、 toDecimal64 、 toDecimal128 或 toFloat32 , toFloat64 , 需要显式地转换其中一个参数。注意 , 结果将失去精度 , 类型转换是昂贵的操作。

Decimal 上的一些函数返回结果为 Float64 (例如 , var 或 stddev) 。对于其中一些 , 中间计算发生在 Decimal 中。对于此类函数 , 尽管结果类型相同 , 但 Float64 和 Decimal 中相同数据的结果可能不同。

溢出检查

在对 Decimal 类型执行操作时 , 数值可能会发生溢出。分数中的过多数字被丢弃 (不是舍入的) 。整数中的过多数字将导致异常。

```
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.
```

检查溢出会导致计算变慢。如果已知溢出不可能，则可以通过设置`decimal_check_overflow`来禁用溢出检查，在这种情况下，溢出将导致结果不正确：

```
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 |
```

溢出检查不仅发生在算术运算上，还发生在比较运算上：

```
SELECT toDecimal32(1, 8) < 100
DB::Exception: Can't compare.
```

Enum8,Enum16

包括 `Enum8` 和 `Enum16` 类型。`Enum` 保存 `'string' = integer` 的对应关系。在 ClickHouse 中，尽管用户使用的是字符串常量，但所有含有 `Enum` 数据类型的操作都是按照包含整数的值来执行。这在性能方面比使用 `String` 数据类型更有效。

- `Enum8` 用 `'String' = Int8` 对描述。
- `Enum16` 用 `'String' = Int16` 对描述。

用法示例

创建一个带有一个枚举 `Enum8('hello' = 1, 'world' = 2)` 类型的列：

```
CREATE TABLE t_enum
(
    x Enum8('hello' = 1, 'world' = 2)
)
ENGINE = TinyLog
```

这个 `x` 列只能存储类型定义中列出的值：`'hello'`或`'world'`。如果您尝试保存任何其他值，ClickHouse 抛出异常。

```
:) INSERT INTO t_enum VALUES ('hello'), ('world'), ('hello')
INSERT INTO t_enum VALUES
Ok.
3 rows in set. Elapsed: 0.002 sec.

:) insert into t_enum values('a')
INSERT INTO t_enum VALUES

Exception on client:
Code: 49. DB::Exception: Unknown element 'a' for type Enum8('hello' = 1, 'world' = 2)
```

当您从表中查询数据时，ClickHouse 从 `Enum` 中输出字符串值。

```
SELECT * FROM t_enum
```

x
hello
world
hello

如果需要看到对应行的数值，则必须将 `Enum` 值转换为整数类型。

```
SELECT CAST(x, 'Int8') FROM t_enum
```

CAST(x, 'Int8')
1
2
1

在查询中创建枚举值，您还需要使用 `CAST`。

```
SELECT toTypeName(CAST('a', 'Enum8(''a'' = 1, ''b'' = 2)'))
toTypeName(CAST('a', 'Enum8(''a'' = 1, ''b'' = 2)'))—
Enum8('a' = 1, 'b' = 2)
```

规则及用法

`Enum8` 类型的每个值范围是 `-128 ... 127`，`Enum16` 类型的每个值范围是 `-32768 ... 32767`。所有的字符串或者数字都必须是不一样的。允许存在空字符串。如果某个 `Enum` 类型被指定了（在表定义的时候），数字可以是任意顺序。然而，顺序并不重要。

`Enum` 中的字符串和数值都不能是 `NULL`。

`Enum` 包含在 `可为空` 类型中。因此，如果您使用此查询创建一个表

```
CREATE TABLE t_enum_nullable
(
    x Nullable( Enum8('hello' = 1, 'world' = 2) )
)
ENGINE = TinyLog
```

不仅可以存储 `'hello'` 和 `'world'`，还可以存储 `NULL`。

```
INSERT INTO t_enum_nullable Values('hello'),('world'),(NULL)
```

在内存中，Enum 列的存储方式与相应数值的 Int8 或 Int16 相同。

当以文本方式读取的时候，ClickHouse 将值解析成字符串然后去枚举值的集合中搜索对应字符串。如果没有找到，会抛出异常。当读取文本格式的时候，会根据读取到的字符串去找对应的数值。如果没有找到，会抛出异常。

当以文本形式写入时，ClickHouse 将值解析成字符串写入。如果列数据包含垃圾数据（不是来自有效集合的数字），则抛出异常。Enum 类型以二进制读取和写入的方式与 Int8 和 Int16 类型一样的。

隐式默认值是数值最小的值。

在 ORDER BY, GROUP BY, IN, DISTINCT 等等中，Enum 的行为与相应的数字相同。例如，按数字排序。对于等式运算符和比较运算符，Enum 的工作机制与它们在底层数值上的工作机制相同。

枚举值不能与数字进行比较。枚举可以与常量字符串进行比较。如果与之比较的字符串不是有效Enum值，则将引发异常。可以使用 IN 运算符来判断一个 Enum 是否存在于某个 Enum 集合中，其中集合中的 Enum 需要用字符串表示。

大多数具有数字和字符串的运算并不适用于Enums；例如，Enum 类型不能和一个数值相加。但是，Enum有一个原生的 `toString` 函数，它返回它的字符串值。

Enum 值使用 `toT` 函数可以转换成数值类型，其中 T 是一个数值类型。若 T 恰好对应 Enum 的底层数值类型，这个转换是零消耗的。

Enum 类型可以被 ALTER 无成本地修改对应集合的值。可以通过 ALTER 操作来增加或删除 Enum 的成员（只要表没有用到该值，删除都是安全的）。作为安全保障，改变之前使用过的 Enum 成员将抛出异常。

通过 ALTER 操作，可以将 `Enum8` 转成 `Enum16`，反之亦然，就像 `Int8` 转 `Int16`一样。

Float32,Float64

浮点数。

类型与以下 C 语言中类型是相同的：

- `Float32 - float`
- `Float64 - double`

我们建议您尽可能以整数形式存储数据。例如，将固定精度的数字转换为整数值，例如货币数量或页面加载时间用毫秒为单位表示

使用浮点数

- 对浮点数进行计算可能引起四舍五入的误差。

```
SELECT 1 - 0.9
```

```
minus(1, 0.9) └  
0.0999999999999998 |
```

- 计算的结果取决于计算方法（计算机系统的处理器类型和体系结构）
- 浮点计算结果可能是诸如无穷大（INF）和«非数字»（NaN）。对浮点数计算的时候应该考虑到这点。
- 当一行行阅读浮点数的时候，浮点数的结果可能不是机器最近显示的数值。

NaN和Inf

与标准SQL相比，ClickHouse 支持以下类别的浮点数：

- Inf – 正无穷

```
SELECT 0.5 / 0
```

```
divide(0.5, 0) └  
      inf |
```

- -Inf – 负无穷

```
SELECT -0.5 / 0
```

```
divide(-0.5, 0) └  
      -inf |
```

- NaN – 非数字

```
SELECT 0 / 0
```

```
divide(0, 0) └  
      nan |
```

可以在 [ORDER BY 子句](#) 查看更多关于 NaN 排序的规则。

SimpleAggregateFunction

`SimpleAggregateFunction(name, types_of_arguments...)` 数据类型存储聚合函数的当前值，并不像 `AggregateFunction` 那样存储其全部状态。这种优化可以应用于具有以下属性函数：将函数 `f` 应用于行集合 `S1 UNION ALL S2` 的结果，可以通过将 `f` 分别应用于行集合的部分，然后再将 `f` 应用于结果来获得： $f(S1 \text{ UNION ALL } S2) = f(f(S1) \text{ UNION ALL } f(S2))$ 。这个属性保证了部分聚合结果足以计算出合并的结果，所以我们不必存储和处理任何额外的数据。

支持以下聚合函数：

- any
- anyLast
- min
- max
- sum
- sumWithOverflow
- groupBitAnd
- groupBitOr
- groupBitXor

- `groupArrayArray`
- `groupUniqArrayArray`
- `sumMap`
- `minMap`
- `maxMap`
- `argMin`
- `argMax`

注

`SimpleAggregateFunction(func, Type)` 的值外观和存储方式于 `Type` 相同，所以你不需要应用带有 `-Merge/-State` 后缀的函数。

`SimpleAggregateFunction` 的性能优于具有相同聚合函数的 `AggregateFunction`。

参数

- 聚合函数的名称。
- 聚合函数参数的类型。

示例

```
CREATE TABLE simple (id UInt64, val SimpleAggregateFunction(sum, Double)) ENGINE=AggregatingMergeTree ORDER BY id;
```

Tuple(T1, T2, ...)

元组，其中每个元素都有单独的 [类型](#)。

不能在表中存储元组（除了内存表）。它们可以用于临时列分组。在查询中，`IN` 表达式和带特定参数的 `lambda` 函数可以来对临时列进行分组。更多信息，请参阅 [IN 操作符](#) 和 [高阶函数](#)。

元组可以是查询的结果。在这种情况下，对于JSON以外的文本格式，括号中的值是逗号分隔的。在JSON格式中，元组作为数组输出（在方括号中）。

创建元组

可以使用函数来创建元组：

```
tuple(T1, T2, ...)
```

创建元组的示例：

```
:) SELECT tuple(1,'a') AS x, toTypeName(x)

SELECT
  (1, 'a') AS x,
  toTypeName(x)

x-----toTypeName(tuple(1, 'a'))-----
|(1,'a') | Tuple(UInt8, String) |

```

1 rows in set. Elapsed: 0.021 sec.

元组中的数据类型

在动态创建元组时，ClickHouse 会自动为元组的每一个参数赋予最小可表达的类型。如果参数为 **NULL**，那这个元组对应元素是 **可为空**。

自动数据类型检测示例：

```
SELECT tuple(1, NULL) AS x, toTypeName(x)

SELECT
  (1, NULL) AS x,
  toTypeName(x)

x-----toTypeName(tuple(1, NULL))-----
|(1,NULL) | Tuple(UInt8, Nullable(Nothing)) |

```

1 rows in set. Elapsed: 0.002 sec.

UInt8, UInt16, UInt32, UInt64, Int8, Int16, Int32, Int64

固定长度的整型，包括有符号整型或无符号整型。

整型范围

- Int8-[-128:127]
- Int16-[-32768:32767]
- Int32-[-2147483648:2147483647]
- Int64-[-9223372036854775808:9223372036854775807]

无符号整型范围

- UInt8-[0:255]
- UInt16-[0:65535]
- UInt32-[0:4294967295]
- UInt64-[0:18446744073709551615]

可为空（类型名称）

允许用特殊标记 (**NULL**) 表示«缺失值»，可以与 **TypeName** 的正常值存放一起。例如，**Nullable(Int8)** 类型的列可以存储 **Int8** 类型值，而没有值的行将存储 **NULL**。

对于 `TypeName`，不能使用复合数据类型 `阵列` 和 `元组`。复合数据类型可以包含 `Nullable` 类型值，例如 `Array Nullable(Int8)`。

`Nullable` 类型字段不能包含在表索引中。

除非在 ClickHouse 服务器配置中另有说明，否则 `NONE` 是任何 `Nullable` 类型的默认值。

存储特性

要在表的列中存储 `Nullable` 类型值，ClickHouse 除了使用带有值的普通文件外，还使用带有 `NONE` 掩码的单独文件。掩码文件中的条目允许 ClickHouse 区分每个表行的 `NONE` 和相应数据类型的默认值。由于附加了新文件，`Nullable` 列与类似的普通文件相比消耗额外的存储空间。

注意点

使用 `Nullable` 几乎总是对性能产生负面影响，在设计数据库时请记住这一点

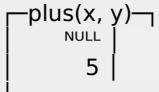
掩码文件中的条目允许 ClickHouse 区分每个表行的对应数据类型的«`NONE`»和默认值由于有额外的文件，«`Nullable`»列比普通列消耗更多的存储空间

用法示例

```
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
```



固定字符串

固定长度 `N` 的字符串（`N` 必须是严格的正自然数）。

您可以使用下面的语法对列声明为 `FixedString` 类型：

```
<column_name> FixedString(N)
```

其中 `N` 表示自然数。

当数据的长度恰好为 `N` 个字节时，`FixedString` 类型是高效的。在其他情况下，这可能会降低效率。

可以有效存储在 `FixedString` 类型的列中的值的示例：

- 二进制表示的 IP 地址（IPv6 使用 `FixedString(16)`）
- 语言代码（`ru_RU, en_US ...`）
- 货币代码（`USD, RUB ...`）

- 二进制表示的哈希值（MD5使用FixedString(16)，SHA256使用FixedString(32)）

请使用 **UUID** 数据类型来存储 UUID 值，。

当向 ClickHouse 中插入数据时，

- 如果字符串包含的字节数少于 `N`，将对字符串末尾进行空字节填充。
- 如果字符串包含的字节数大于 N，将抛出 **Too large value for FixedString(N)** 异常。

当做数据查询时，ClickHouse 不会删除字符串末尾的空字节。如果使用 **WHERE** 子句，则须要手动添加空字节以匹配 **FixedString** 的值。以下示例阐明了如何将 **WHERE** 子句与 **FixedString** 一起使用。

考虑带有 **FixedString(2)** 列的表：

name
b

查询语句 `SELECT * FROM FixedStringTable WHERE a = 'b'` 不会返回任何结果。请使用空字节来填充筛选条件。

```
SELECT * FROM FixedStringTable  
WHERE a = 'b\0'
```

a
b

这种方式与 MySQL 的 **CHAR** 类型的方式不同（MySQL 中使用空格填充字符串，并在输出时删除空格）。

请注意，**FixedString(N)** 的长度是个常量。仅由空字符组成的字符串，函数 **length** 返回值为 N，而函数 **empty** 的返回值为 1。

字符串

字符串可以任意长度的。它可以包含任意的字节集，包含空字节。因此，字符串类型可以代替其他 DBMSs 中的 **VARCHAR**、**BLOB**、**CLOB** 等类型。

编码

ClickHouse 没有编码的概念。字符串可以是任意的字节集，按它们原本的方式进行存储和输出。

若需存储文本，我们建议使用 **UTF-8** 编码。至少，如果你的终端使用 **UTF-8**（推荐），这样读写就不需要进行任何的转换了。

同样，对不同的编码文本 ClickHouse 会有不同处理字符串的函数。

比如，**length** 函数可以计算字符串包含的字节数组的长度，然而 **lengthUTF8** 函数是假设字符串以 **UTF-8** 编码，计算的是字符串包含的 **Unicode** 字符的长度。

布尔值

没有单独的类型来存储布尔值。可以使用 **UInt8** 类型，取值限制为 0 或 1。

日期

日期类型，用两个字节存储，表示从 **1970-01-01**（无符号）到当前的日期值。允许存储从 Unix 纪元开始到编译阶段定义的上限阈值常量（目前上限是 2106 年，但最终完全支持的年份为 2105）。最小值输出为 **1970-01-01**。

日期中没有存储时区信息。

日期时间

时间戳类型。用四个字节（无符号的）存储 Unix 时间戳）。允许存储与日期类型相同的范围内的值。最小值为 1970-01-01 00:00:00。时间戳类型值精确到秒（没有闰秒）。

时区

使用启动客户端或服务器时的系统时区，时间戳是从文本（分解为组件）转换为二进制并返回。在文本格式中，有关夏令时的信息会丢失。

默认情况下，客户端连接到服务的时候会使用服务端时区。您可以通过启用客户端命令行选项 `--use_client_time_zone` 来设置使用客户端时间。

因此，在处理文本日期时（例如，在保存文本转储时），请记住在夏令时更改期间可能存在歧义，如果时区发生更改，则可能存在匹配数据的问题。

阵列(T)

由 T 类型元素组成的数组。

T 可以是任意类型，包含数组类型。但不推荐使用多维数组，ClickHouse 对多维数组的支持有限。例如，不能存储在 MergeTree 表中存储多维数组。

创建数组

您可以使用 array 函数来创建数组：

```
array(T)
```

您也可以使用方括号：

```
[]
```

创建数组示例：

```
:) SELECT array(1, 2) AS x, toTypeName(x)
```

```
SELECT
```

```
[1, 2] AS x,  
toTypeName(x)
```

```
x ── toTypeName(array(1, 2)) ─  
[1,2] | Array(UInt8) |
```

1 rows in set. Elapsed: 0.002 sec.

```
:) SELECT [1, 2] AS x, toTypeName(x)
```

```
SELECT
```

```
[1, 2] AS x,  
toTypeName(x)
```

```
x ── toTypeName([1, 2]) ─  
[1,2] | Array(UInt8) |
```

1 rows in set. Elapsed: 0.002 sec.

使用数据类型

ClickHouse会自动检测数组元素，并根据元素计算出存储这些元素最小的数据类型。如果在元素中存在 **NULL** 或存在 **可为空** 类型元素，那么数组的元素类型将会变成 **可为空**。

如果 ClickHouse 无法确定数据类型，它将产生异常。当尝试同时创建一个包含字符串和数字的数组时会发生这种情况 (`SELECT array(1, 'a')`)。

自动数据类型检测示例：

```
:) SELECT array(1, 2, NULL) AS x, toTypeName(x)
```

```
SELECT
```

```
[1, 2, NULL] AS x,  
toTypeName(x)
```

```
x ── toTypeName(array(1, 2, NULL)) ─  
[1,2,NULL] | Array(Nullable(UInt8)) |
```

1 rows in set. Elapsed: 0.002 sec.

如果您尝试创建不兼容的数据类型数组，ClickHouse 将引发异常：

```
:) SELECT array(1, 'a')
```

```
SELECT [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.

0 rows in set. Elapsed: 0.246 sec.

Nested(Name1 Type1, Name2 Type2, ...)

嵌套数据结构类似于嵌套表。嵌套数据结构的参数（列名和类型）与 CREATE 查询类似。每个表可以包含任意多行嵌套数据结构。

示例：

```

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)

```

上述示例声明了 `Goals` 这种嵌套数据结构，它包含访客转化相关的数据（访客达到的目标）。在 ‘visits’ 表中每一行都可以对应零个或者任意个转化数据。

只支持一级嵌套。嵌套结构的列中，若列的类型是数组类型，那么该列其实和多维数组是相同的，所以目前嵌套层级的支持很局限（MergeTree 引擎中不支持存储这样的列）

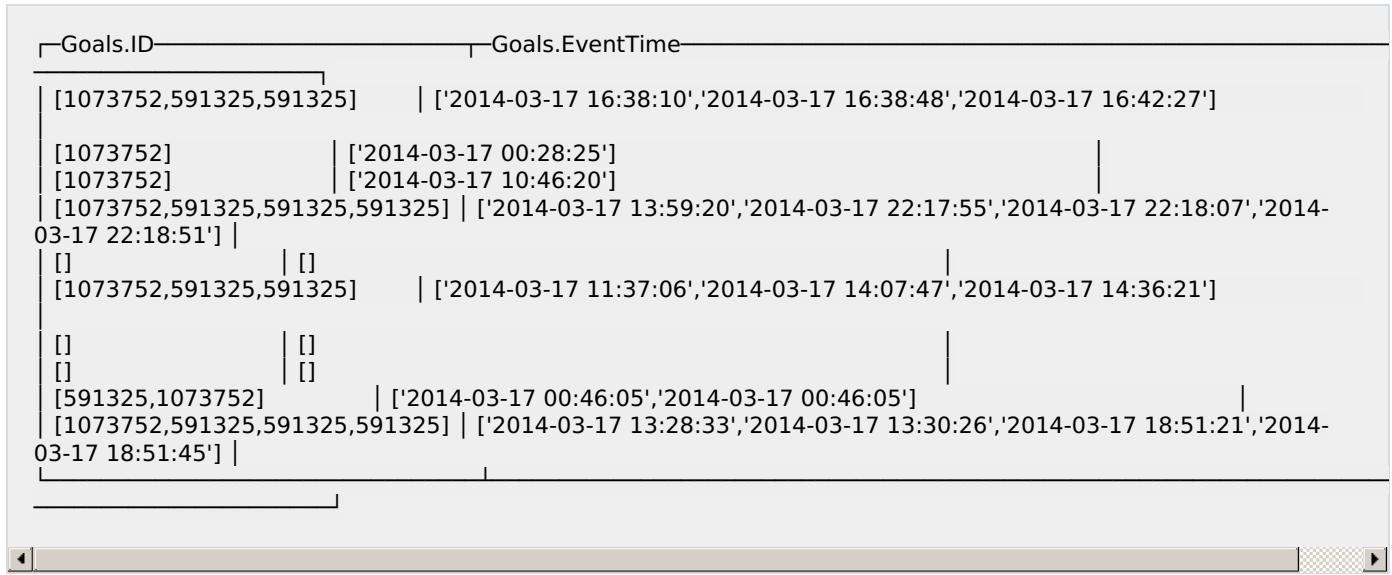
大多数情况下，处理嵌套数据结构时，会指定一个单独的列。为了这样实现，列的名称会与点号连接起来。这些列构成了一组匹配类型。在同一条嵌套数据中，所有的列都具有相同的长度。

示例：

```

SELECT
    Goals.ID,
    Goals.EventTime
FROM test.visits
WHERE CounterID = 101500 AND length(Goals.ID) < 5
LIMIT 10

```



所以可以简单地把嵌套数据结构当做是所有列都是相同长度的多列数组。

`SELECT` 查询只有在使用 `ARRAY JOIN` 的时候才可以指定整个嵌套数据结构的名称。更多信息，参考 «`ARRAY JOIN` 子句»。示例：

```
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

不能对整个嵌套数据结构执行 SELECT。只能明确列出属于它一部分列。

对于 INSERT 查询，可以单独地传入所有嵌套数据结构中的列数组（假如它们是单独的列数组）。在插入过程中，系统会检查它们是否有相同的长度。

对于 DESCRIBE 查询，嵌套数据结构中的列会以相同的方式分别列出来。

ALTER 查询对嵌套数据结构的操作非常有限。

嵌套数据结构

Interval类型

表示时间和日期间隔的数据类型家族。**INTERVAL** 运算的结果类型。

警告

Interval 数据类型值不能存储在表中。

结构:

- 时间间隔作为无符号整数值。
- 时间间隔的类型。

支持的时间间隔类型:

- SECOND
- MINUTE
- HOUR
- DAY
- WEEK
- MONTH

- QUARTER

- YEAR

对于每个时间间隔类型，都有一个单独的数据类型。例如，DAY 间隔对应于 IntervalDay 数据类型：

```
SELECT toTypeName(INTERVAL 4 DAY)
```

```
toTypeName(toIntervalDay(4))  
IntervalDay
```

使用说明

您可以在与 **日期** 和 **日期时间** 类型值的算术运算中使用 **Interval** 类型值。例如，您可以将4天添加到当前时间：

```
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 |
```

不同类型的间隔不能合并。你不能使用诸如 **4 DAY 1 HOUR** 的时间间隔。以小于或等于时间间隔最小单位的单位来指定间隔，例如，时间间隔 **1 day and an hour** 可以表示为 **25 HOUR** 或 **90000 SECOND**.

你不能对 **Interval** 类型的值执行算术运算，但你可以向 **Date** 或 **DateTime** 数据类型的值添加不同类型的时间间隔，例如：

```
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 |
```

以下查询将导致异常：

```
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..

另请参阅

- **INTERVAL** 操作
- **toInterval** 类型转换函数

没什么

此数据类型的唯一目的是表示不是期望值的情况。所以不能创建一个 **Nothing** 类型的值。

例如，文本 **NULL** 的类型为 **Nullable(Nothing)**。详情请见 [可为空](#)。

`Nothing` 类型也可以用来表示空数组：

```
:) SELECT toTypeName(array())
SELECT toTypeName([])
└─ toTypeName(array())
   └─ Array(Nothing)
```

1 rows in set. Elapsed: 0.062 sec.

特殊数据类型

特殊数据类型的值既不能存在表中也不能在结果中输出，但可用于查询的中间结果。

表达式

用于表示高阶函数中的Lambda表达式。

设置

可以用在 `IN` 表达式的右半部分。

Ansi Sql兼容性的ClickHouse SQL方言

注

本文依赖于表38, “Feature taxonomy and definition for mandatory features”, Annex F of ISO/IEC CD 9075-2:2013.

行为差异

下表列出了查询功能在ClickHouse中有效但不符合ANSI SQL标准的情况。

Feature ID	功能名称	差异
E011	数值 (Numeric) 数据类型	带小数点的数值文字被解释为近似值 (<code>Float64</code>) 而不是精确值 (<code>Decimal</code>)
E051-05	SELECT字段可以重命名	字段不仅仅在SELECT结果中可被重命名
E141-01	非空约束	表中每一列默认为 <code>NOT NULL</code>
E011-04	算术运算符	ClickHouse不会检查算法，并根据自定义规则更改结果数据类型，而是会溢出

功能匹配

Feature ID	功能名称	匹配	评论
E011	数字数据类型	部分	
E011-01	整型和小型数据类型	是	
E011-02	真实、双精度和浮点数据类型数据类型	部分	FLOAT(<binary_precision>), REAL 和 DOUBLE PRECISION 不支持
E011-03	十进制和数值数据类型	部分	只有 DECIMAL(p,s) 支持，而不是 NUMERIC
E011-04	算术运算符	是	
E011-05	数字比较	是	
E011-06	数字数据类型之间的隐式转换	否。	ANSI SQL允许在数值类型之间进行任意隐式转换，而 ClickHouse依赖于具有多个重载的函数而不是隐式转换
E021	字符串类型	部分	
E021-01	字符数据类型	否。	
E021-02	字符变化数据类型	否。	String 行为类似，但括号中没有长度限制
E021-03	字符文字	部分	不自动连接连续文字和字符集支持
E021-04	字符长度函数	部分	非也。 USING 条款
E021-05	OCTET_LENGTH函数	非也。	LENGTH 表现类似
E021-06	SUBSTRING	部分	不支持 SIMILAR 和 ESCAPE 条款，否 SUBSTRING_REGEX 备选案文
E021-07	字符串联	部分	非也。 COLLATE 条款
E021-08	上下功能	是	
E021-09	修剪功能	是	
E021-10	固定长度和可变长度字符串类型之间的隐式转换	否。	ANSI SQL允许在字符串类型之间进行任意隐式转换，而 ClickHouse依赖于具有多个重载的函数而不是隐式转换
E021-11	职位功能	部分	不支持 IN 和 USING 条款，否 POSITION_REGEX 备选案文
E021-12	字符比较	是	
E031	标识符	部分	
E031-01	分隔标识符	部分	Unicode文字支持有限

Feature ID	功能名称	匹配	评论
E031-02	小写标识符	是	
E031-03	尾部下划线	是	
E051	基本查询规范	部分	
E051-01	SELECT DISTINCT	是	
E051-02	GROUP BY子句	是	
E051-04	分组依据可以包含不在列 <code><select list></code>	是	
E051-05	选择项目可以重命名	是	
E051-06	有条款	是	
E051-07	合格*在选择列表中	是	
E051-08	FROM子句中的关联名称	是	
E051-09	重命名FROM子句中的列	否。	
E061	基本谓词和搜索条件	部分	
E061-01	比较谓词	是	
E061-02	谓词之间	部分	非也。 SYMMETRIC 和 ASYMMETRIC 条款
E061-03	在具有值列表的谓词中	是	
E061-04	像谓词	是	
E061-05	LIKE谓词：逃避条款	否。	
E061-06	空谓词	是	
E061-07	量化比较谓词	非 也。	
E061-08	存在谓词	非 也。	
E061-09	比较谓词中的子查询	是	
E061-11	谓词中的子查询	是	
E061-12	量化比较谓词中的子查询	否。	

Feature ID	功能名称	匹配	评论
E061-13	相关子查询	否。	
E061-14	搜索条件	是	
E071	基本查询表达式	部分	
E071-01	UNION DISTINCT table运算符	否。	
E071-02	联合所有表运算符	是	
E071-03	除了不同的表运算符	非也。	
E071-05	通过表运算符组合的列不必具有完全相同的数据类型	是	
E071-06	子查询中的表运算符	是	
E081	基本特权	部分	正在进行的工作
E091	设置函数	是	
E091-01	AVG	是	
E091-02	COUNT	是	
E091-03	MAX	是	
E091-04	MIN	是	
E091-05	SUM	是	
E091-06	全部量词	否。	
E091-07	不同的量词	部分	并非所有聚合函数都受支持
E101	基本数据操作	部分	
E101-01	插入语句	是	注：ClickHouse中的主键并不意味着 UNIQUE 约束
E101-03	搜索更新语句	否。	有一个 ALTER UPDATE 批量数据修改语句
E101-04	搜索的删除语句	否。	有一个 ALTER DELETE 批量数据删除声明
E111	单行SELECT语句	否。	
E121	基本光标支持	否。	

Feature ID	功能名称	匹配	评论
E121-01	DECLARE CURSOR	否。	
E121-02	按列排序不需要在选择列表中	否。	
E121-03	按顺序排列的值表达式	否。	
E121-04	公开声明	否。	
E121-06	定位更新语句	否。	
E121-07	定位删除语句	否。	
E121-08	关闭声明	否。	
E121-10	FETCH语句：隐式NEXT	否。	
E121-17	使用保持游标	否。	
E131	空值支持（空值代替值）	部分	一些限制适用
E141	基本完整性约束	部分	
E141-01	非空约束	是	注: NOT NULL 默认情况下，表列隐含
E141-02	非空列的唯一约束	否。	
E141-03	主键约束	否。	
E141-04	对于引用删除操作和引用更新操作，具有默认无操作的基本外键约束	否。	
E141-06	检查约束	是	
E141-07	列默认值	是	
E141-08	在主键上推断为非NULL	是	
E141-10	可以按任何顺序指定外键中的名称	否。	
E151	交易支持	否。	
E151-01	提交语句	否。	
E151-02	回滚语句	否。	
E152	基本设置事务语句	否。	

Feature ID	功能名称	匹配	评论
E152-01	SET TRANSACTION语句： 隔离级别SERIALIZABLE子句	否。	
E152-02	SET TRANSACTION语句： 只读和读写子句	否。	
E153	具有子查询的可更新查询	否。	
E161	SQL注释使用前导双减	是	
E171	SQLSTATE支持	否。	
E182	主机语言绑定	否。	
F031	基本架构操作	部分	
F031-01	CREATE TABLE语句创建持久基表	部分	否。 SYSTEM VERSIONING, ON COMMIT, GLOBAL, LOCAL, PRESERVE, DELETE, REF IS, WITH OPTIONS, UNDER, LIKE, PERIOD FOR 子句，不支持用户解析的数据类型
F031-02	创建视图语句	部分	否。 RECURSIVE, CHECK, UNDER, WITH OPTIONS 子句，不支持用户解析的数据类型
F031-03	赠款声明	是	
F031-04	ALTER TABLE语句：ADD COLUMN子句	部分	不支持 GENERATED 条款和系统时间段
F031-13	DROP TABLE语句： RESTRICT子句	否。	
F031-16	DROP VIEW语句： RESTRICT子句	否。	
F031-19	REVOKE语句：RESTRICT子句	否。	
F041	基本连接表	部分	
F041-01	Inner join (但不一定是 INNER关键字)	是	
F041-02	内部关键字	是	
F041-03	LEFT OUTER JOIN	是	
F041-04	RIGHT OUTER JOIN	是	

Feature ID	功能名称	匹配	评论
F041-05	可以嵌套外部连接	是	
F041-07	左侧或右侧外部联接中的内部表也可用于内部联接	是	
F041-08	支持所有比较运算符（不仅仅是=）	否。	
F051	基本日期和时间	部分	
F051-01	日期数据类型（包括对日期文字的支持）	部分	没有文字
F051-02	时间数据类型（包括对时间文字的支持），秒小数精度至少为0	否。	
F051-03	时间戳数据类型（包括对时间戳文字的支持），小数秒精度至少为0和6	否。	DateTime64 时间提供了类似的功能
F051-04	日期、时间和时间戳数据类型的比较谓词	部分	只有一种数据类型可用
F051-05	Datetime类型和字符串类型之间的显式转换	是	
F051-06	CURRENT_DATE	否。	today() 是相似的
F051-07	LOCALTIME	否。	now() 是相似的
F051-08	LOCALTIMESTAMP	否。	
F081	联盟和视图除外	部分	
F131	分组操作	部分	
F131-01	WHERE、GROUP BY和 HAVING子句在具有分组视图的查询中受支持	是	
F131-02	具有分组视图的查询中支持的多个表	是	
F131-03	设置具有分组视图的查询中支持的函数	是	
F131-04	具有分组依据和具有子句和分组视图的子查询	是	

Feature ID	功能名称	匹配	评论
F131-05	单行选择具有GROUP BY和具有子句和分组视图	非也。	
F181	多模块支持	否。	
F201	投函数	是	
F221	显式默认值	否。	
F261	案例表达式	是	
F261-01	简单案例	是	
F261-02	检索案例	是	
F261-03	NULLIF	是	
F261-04	COALESCE	是	
F311	架构定义语句	部分	
F311-01	CREATE SCHEMA	否。	
F311-02	为持久基表创建表	是	
F311-03	CREATE VIEW	是	
F311-04	CREATE VIEW: WITH CHECK OPTION	否。	
F311-05	赠款声明	是	
F471	标量子查询值	是	
F481	扩展空谓词	是	
F812	基本标记	否。	
T321	基本的SQL调用例程	否。	
T321-01	无重载的用户定义函数	否。	
T321-02	无重载的用户定义存储过程	否。	
T321-03	函数调用	否。	
T321-04	电话声明	否。	
T321-05	退货声明	否。	

Feature ID	功能名称	匹配	评论
T631	在一个列表元素的谓词中	是	

[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 (feature request)
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/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>

IN 操作符

该 `IN`, `NOT IN`, `GLOBAL IN`, 和 `GLOBAL NOT IN` 运算符是单独考虑的，因为它们的功能相当丰富。

运算符的左侧是单列或元组。

例：

```
SELECT UserID IN (123, 456) FROM ...
SELECT (CounterID, UserID) IN ((34, 123), (101500, 456)) FROM ...
```

如果左侧是索引中的单列，而右侧是一组常量，则系统将使用索引处理查询。

请不要列举太多具体的常量（比方说 几百万条）。如果数据集非常大，请把它放在一张临时表里（例如，参考章节[用于查询处理的外部数据](#)），然后使用子查询。

运算符的右侧可以是一组常量表达式、一组带有常量表达式的元组（如上面的示例所示），或括号中的数据库表或SELECT子查询的名称。

如果运算符的右侧是表的名称（例如，`UserID IN users`），这相当于子查询 `UserID IN (SELECT * FROM users)`。使用与查询一起发送的外部数据时，请使用此选项。例如，查询可以与一组用户 Id一起发送到 ‘users’ 应过滤的临时表。

如果运算符的右侧是具有Set引擎的表名（始终位于RAM中的准备好的数据集），则不会为每个查询重新创建数据集。

子查询可以指定多个用于筛选元组的列。

示例：

```
SELECT (CounterID, UserID) IN (SELECT CounterID, UserID FROM ...)
```

`IN` 运算符左侧和右侧的列应具有相同的类型。

`IN` 运算符和子查询可能出现在查询的任何部分，包括聚合函数和 `lambda` 函数。

示例：

```
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

对于3月17日后的每一天，计算3月17日访问该网站的用户所做的浏览量百分比。

`IN` 子句中的子查询始终只在单个服务器上运行一次。没有依赖子查询。

空处理

在请求处理过程中，`IN` 运算符假定运算的结果 `NULL` 总是等于 `0`，无论是否 `NULL` 位于操作员的右侧或左侧。`NULL` 值不包含在任何数据集中，彼此不对应，并且在以下情况下无法进行比较 `transform_null_in=0`。

下面是一个例子 `t_null` 表：

x	y
1	NULL
2	3

运行查询 `SELECT x FROM t_null WHERE y IN (NULL,3)` 为您提供以下结果：

x
2

你可以看到，在其中的行 `y = NULL` 被抛出的查询结果。这是因为 ClickHouse 无法决定是否 `NULL` 包含在 `(NULL,3)` 设置，返回 `0` 作为操作的结果，和 `SELECT` 从最终输出中排除此行。

```
SELECT y IN (NULL, 3)
FROM t_null
```

```
in(y, tuple(NULL, 3))—  
  0 |  
  1 |
```

分布式子查询

带子查询的IN-s有两个选项（类似于连接）：normal IN / JOIN 和 GLOBAL IN / GLOBAL JOIN. 它们在分布式查询处理的运行方式上有所不同。

注意

请记住，下面描述的算法可能会有不同的工作方式取决于 [设置 distributed_product_mode 设置](#)。

当使用常规IN时，查询被发送到远程服务器，并且它们中的每个服务器都在运行子查询 IN 或 JOIN 条款

使用时 GLOBAL IN / GLOBAL JOINS，首先所有的子查询都运行 GLOBAL IN / GLOBAL JOINS，并将结果收集在临时表中。然后将临时表发送到每个远程服务器，其中使用此临时数据运行查询。

对于非分布式查询，请使用常规 IN / JOIN.

在使用子查询时要小心 IN / JOIN 用于分布式查询处理的子句。

让我们来看看一些例子。假设集群中的每个服务器都有一个正常的 **local_table**. 每个服务器还具有 **distributed_table** 表与 分布 类型，它查看群集中的所有服务器。

对于查询 **distributed_table**，查询将被发送到所有远程服务器，并使用以下命令在其上运行 **local_table**.

例如，查询

```
SELECT uniq(UserID) FROM distributed_table
```

将被发送到所有远程服务器

```
SELECT uniq(UserID) FROM local_table
```

并且并行运行它们中的每一个，直到达到可以结合中间结果的阶段。然后将中间结果返回给请求者服务器并在其上合并，并将最终结果发送给客户端。

现在让我们检查一个查询IN:

```
SELECT uniq(UserID) FROM distributed_table WHERE CounterID = 101500 AND UserID IN (SELECT UserID FROM local_table WHERE CounterID = 34)
```

- 计算两个网站的受众的交集。

此查询将以下列方式发送到所有远程服务器

```
SELECT uniq(UserID) FROM local_table WHERE CounterID = 101500 AND UserID IN (SELECT UserID FROM local_table WHERE CounterID = 34)
```

换句话说，IN子句中的数据集将在每台服务器上独立收集，仅在每台服务器上本地存储的数据中收集。

如果您已经为此情况做好准备，并且已经将数据分散到群集服务器上，以便单个用户 Id 的数据完全驻留在单个服务器上，则这将正常和最佳地工作。在这种情况下，所有必要的数据将在每台服务器上本地提供。否则，结果将是不准确的。我们将查询的这种变体称为“local IN”。

若要更正数据在群集服务器上随机传播时查询的工作方式，可以指定 **distributed_table** 在子查询中。查询如下所示：

```
SELECT uniq(UserID) FROM distributed_table WHERE CounterID = 101500 AND UserID IN (SELECT UserID FROM distributed_table WHERE CounterID = 34)
```

此查询将以下列方式发送到所有远程服务器

```
SELECT uniq(UserID) FROM local_table WHERE CounterID = 101500 AND UserID IN (SELECT UserID FROM distributed_table WHERE CounterID = 34)
```

子查询将开始在每个远程服务器上运行。由于子查询使用分布式表，因此每个远程服务器上的子查询将重新发送到每个远程服务器

```
SELECT UserID FROM local_table WHERE CounterID = 34
```

例如，如果您有 100 台服务器的集群，则执行整个查询将需要 10,000 个基本请求，这通常被认为是不可接受的。

在这种情况下，应始终使用 **GLOBAL IN** 而不是 **IN**。让我们来看看它是如何工作的查询

```
SELECT uniq(UserID) FROM distributed_table WHERE CounterID = 101500 AND UserID GLOBAL IN (SELECT UserID FROM distributed_table WHERE CounterID = 34)
```

请求者服务器将运行子查询

```
SELECT UserID FROM distributed_table WHERE CounterID = 34
```

结果将被放在 RAM 中的临时表中。然后请求将被发送到每个远程服务器

```
SELECT uniq(UserID) FROM local_table WHERE CounterID = 101500 AND UserID GLOBAL IN _data1
```

和临时表 **_data1** 将通过查询发送到每个远程服务器（临时表的名称是实现定义的）。

这比使用正常 **IN** 更优化。但是，请记住以下几点：

1. 创建临时表时，数据不是唯一的。要减少通过网络传输的数据量，请在子查询中指定 **DISTINCT**。（你不需要为正常人做这个。）
2. 临时表将被发送到所有远程服务器。传输不考虑网络拓扑。例如，如果 10 个远程服务器驻留在与请求者服务器非常远程的数据中心中，则数据将通过通道发送 10 次到远程数据中心。使用 **GLOBAL IN** 时尽量避免使用大型数据集。
3. 将数据传输到远程服务器时，无法配置网络带宽限制。您可能会使网络过载。
4. 尝试跨服务器分发数据，以便您不需要定期使用 **GLOBAL IN**。
5. 如果您需要经常使用 **GLOBAL IN**，请规划 ClickHouse 集群的位置，以便单个副本组驻留在不超过一个数据中心中，并且它们之间具有快速网络，以便可以完全在单个数据中心内处理查询。

这也是有意义的，在指定一个本地表 **GLOBAL IN** 子句，以防此本地表仅在请求者服务器上可用，并且您希望在远程服务器上使用来自它的数据。

操作符

所有的操作符（运算符）都会在查询时依据他们的优先级及其结合顺序在被解析时转换为对应的函数。下面按优先级从高到低列出各组运算符及其对应的函数：

下标运算符

`a[N]` – 数组中的第`N`个元素；对应函数 `arrayElement(a, N)`

`a.N` – 元组中第`N`个元素；对应函数 `tupleElement(a, N)`

负号

`-a` – 对应函数 `negate(a)`

乘号、除号和取余

`a * b` – 对应函数 `multiply(a, b)`

`a / b` – 对应函数 `divide(a, b)`

`a % b` – 对应函数 `modulo(a, b)`

加号和减号

`a + b` – 对应函数 `plus(a, b)`

`a - b` – 对应函数 `minus(a, b)`

关系运算符

`a = b` – 对应函数 `equals(a, b)`

`a == b` – 对应函数 `equals(a, b)`

`a != b` – 对应函数 `notEquals(a, b)`

`a <> b` – 对应函数 `notEquals(a, b)`

`a <= b` – 对应函数 `lessOrEquals(a, b)`

`a >= b` – 对应函数 `greaterOrEquals(a, b)`

`a < b` – 对应函数 `less(a, b)`

`a > b` – 对应函数 `greater(a, b)`

`a LIKE s` – 对应函数 `like(a, b)`

`a NOT LIKE s` – 对应函数 `notLike(a, b)`

`a BETWEEN b AND c` – 等价于 `a >= b AND a <= c`

集合关系运算符

详见此节 **IN 相关操作符**。

`a IN ...` – 对应函数 `in(a, b)`

`a NOT IN ...` – 对应函数 `notIn(a, b)`

`a GLOBAL IN ...` – 对应函数 `globalIn(a, b)`

`a GLOBAL NOT IN ...` – 对应函数 `globalNotIn(a, b)`

逻辑非

`NOT a` – 对应函数 `not(a)`

逻辑与

`a AND b` – 对应函数 `and(a, b)`

逻辑或

`a OR b` – 对应函数 `or(a, b)`

条件运算符

`a ? b : c` – 对应函数 `if(a, b, c)`

注意:

条件运算符会先计算表达式`b`和表达式`c`的值，再根据表达式`a`的真假，返回相应的值。如果表达式`b`和表达式`c`是 `arrayJoin()` 函数，则不管表达式`a`是真是假，每行都会被复制展开。

使用日期和时间的操作员

EXTRACT

```
EXTRACT(part FROM date);
```

从给定日期中提取部件。例如，您可以从给定日期检索一个月，或从时间检索一秒钟。

该 `part` 参数指定要检索的日期部分。以下值可用:

- `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.

该 `part` 参数不区分大小写。

该 `date` 参数指定要处理的日期或时间。无论是 `日期` 或 `日期时间` 支持类型。

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'));
```

在下面的例子中，我们创建一个表，并在其中插入一个值 `DateTime` 类型。

```
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

你可以看到更多的例子 [测试](#).

INTERVAL

创建一个 [间隔](#)-应在算术运算中使用的类型值 [日期](#) 和 [日期时间](#)-类型值。

示例:

```
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

另请参阅

- [间隔](#) 数据类型
- [toInterval](#) 类型转换函数

CASE 条件表达式

```
CASE [x]
    WHEN a THEN b
    [WHEN ... THEN ...]
    [ELSE c]
END
```

如果指定了 `x`，该表达式会转换为 `transform(x, [a, ...], [b, ...], c)` 函数。否则转换为 `multilf(a, b, ..., c)`

如果该表达式中没有 `ELSE c` 子句，则默认值就是 `NULL`

但 `transform` 函数不支持 `NULL`

连接运算符

`s1 || s2` – 对应函数 `concat(s1, s2)`

创建 Lambda 函数

`x -> expr` – 对应函数 `lambda(x, expr)`

接下来的这些操作符因为其本身是括号没有优先级：

创建数组

`[x1, ...]` – 对应函数 `array(x1, ...)`

创建元组

`(x1, x2, ...)` – 对应函数 `tuple(x2, x2, ...)`

结合方式

所有的同级操作符从左到右结合。例如，`1 + 2 + 3` 会转换成 `plus(plus(1, 2), 3)`。

所以，有时他们会跟我们预期的不太一样。例如，`SELECT 4 > 2 > 3` 的结果是 0。

为了高效，`and` 和 `or` 函数支持任意多参数，一连串的 `AND` 和 `OR` 运算符会转换成其对应的单个函数。

判断是否为 NULL

ClickHouse 支持 `IS NULL` 和 `IS NOT NULL`。

IS NULL

- 对于 可为空 类型的值，`IS NULL` 会返回：
 - 1 值为 `NULL`
 - 0 否则
- 对于其他类型的值，`IS NULL` 总会返回 0

```
:) SELECT x+100 FROM t_null WHERE y IS NULL
```

```
SELECT x + 100  
FROM t_null  
WHEREisNull(y)
```

```
└─plus(x, 100)─  
    101 |
```

```
1 rows in set. Elapsed: 0.002 sec.
```

IS NOT NULL

- 对于 可为空 类型的值，`IS NOT NULL` 会返回：
 - 0 值为 `NULL`
 - 1 否则
- 对于其他类型的值，`IS NOT NULL` 总会返回 1

```
:) SELECT * FROM t_null WHERE y IS NOT NULL
```

```
SELECT *
FROM t_null
WHERE isNotNull(y)
```

x	y
2	3

1 rows in set. Elapsed: 0.002 sec.

ClickHouse指南

列出了如何使用 Clickhouse 解决各种任务的详细说明：

- [关于简单集群设置的教程](#)
- [在ClickHouse中应用CatBoost模型](#)

在ClickHouse中应用Catboost模型

CatBoost 是一个由[Yandex](#)开发的开源免费机器学习库。

通过这篇指导，您将学会如何用SQL建模，使用ClickHouse预先训练好的模型来推断数据。

在ClickHouse中应用CatBoost模型的一般过程：

1. [创建数据表](#).
2. [将数据插入到表中](#).
3. [将CatBoost集成到ClickHouse中](#)（可跳过）。
4. [从SQL运行模型推断](#).

有关训练CatBoost模型的详细信息，请参阅 [训练和模型应用](#).

先决条件

请先安装 [Docker](#)。

注

Docker 是一个软件平台，用户可以用来创建独立于其余系统、集成CatBoost和ClickHouse的容器。

在应用CatBoost模型之前：

1. 从容器仓库拉取docker映像 (<https://hub.docker.com/r/yandex/tutorial-catboost-clickhouse>) :

```
$ docker pull yandex/tutorial-catboost-clickhouse
```

此Docker映像包含运行CatBoost和ClickHouse所需的所有内容：代码、运行环境、库、环境变量和配置文件。

2. 确保已成功拉取Docker映像：

```
$ docker image ls
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
yandex/tutorial-catboost-clickhouse  latest   622e4d17945b  22 hours ago  1.37GB
```

3. 基于此映像启动一个Docker容器:

```
$ docker run -it -p 8888:8888 yandex/tutorial-catboost-clickhouse
```

1. 创建数据表

为训练样本创建ClickHouse表:

1. 在交互模式下启动ClickHouse控制台客户端:

```
$ clickhouse client
```

注

ClickHouse服务器已经在Docker容器内运行。

2. 使用以下命令创建表:

```
:) 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. 从ClickHouse控制台客户端退出:

```
:) exit
```

2. 将数据插入到表中

插入数据:

1. 运行以下命令:

```
$ clickhouse client --host 127.0.0.1 --query 'INSERT INTO amazon_train FORMAT CSVWithNames' <
~/amazon/train.csv
```

2. 在交互模式下启动ClickHouse控制台客户端:

```
$ clickhouse client
```

3. 确保数据已上传:

```
:) SELECT count() FROM amazon_train  
  
SELECT count()  
FROM amazon_train  
  
+-count()-+  
| 65538 |  
+-----+
```

3. 将CatBoost集成到ClickHouse中

注

可跳过。 Docker映像包含运行CatBoost和ClickHouse所需的所有内容。

CatBoost集成到ClickHouse步骤：

1. 构建评估库。

评估CatBoost模型的最快方法是编译 `libcatboostmodel.<so|dll|dylib>` 库文件。

有关如何构建库文件的详细信息，请参阅 [CatBoost文件](#)。

2. 创建一个新目录（位置与名称可随意指定），如 `data` 并将创建的库文件放入其中。 Docker映像已经包含了库 `data/libcatboostmodel.so`。

3. 创建一个新目录来放配置模型，如 `models`。

4. 创建一个模型配置文件，如 `models/amazon_model.xml`。

5. 描述模型配置：

```
<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. 将CatBoost库文件的路径和模型配置添加到ClickHouse配置：

```
<!-- 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. 运行从SQL推断的模型

测试模型是否正常，运行ClickHouse客户端 `$ clickhouse client`

让我们确保模型能正常工作：

```
:) 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

函数 **modelEvaluate** 返回带有多类模型的每类原始预测的元组。

执行预测：

```
:) 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

查看函数说明 **exp()**。

让我们计算样本的LogLoss:

```
:) 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

查看函数说明 **avg()** 和 **log()** 。

操作

ClickHouse操作手册由以下主要部分组成：

- 安装要求
- 监控
- 故障排除
- 使用建议
- 更新程序
- 访问权限
- 数据备份
- 配置文件
- 配额
- 系统表
- 服务器配置参数
- 如何用ClickHouse测试你的硬件
- 设置
- 实用工具

要求

CPU

对于从预构建的deb包进行安装，请使用具有x86_64架构并支持SSE4.2指令的CPU。要使用不支持SSE4.2或具有AArch64或PowerPC64LE体系结构的处理器运行ClickHouse，您应该从源代码构建ClickHouse。

ClickHouse实现并行数据处理并使用所有可用的硬件资源。在选择处理器时，考虑到ClickHouse在具有大量内核但时钟速率较低的配置中的工作效率要高于具有较少内核和较高时钟速率的配置。例如，具有2600MHz的16核心优于具有3600MHz的8核心。

建议使用睿频加速和超线程技术。它显着提高了典型工作负载的性能。

RAM

我们建议使用至少4GB的RAM来执行重要的查询。ClickHouse服务器可以使用少得多的RAM运行，但它需要处理查询的内存。

RAM所需的体积取决于：

- 查询的复杂性。

- 查询中处理的数据量。

要计算所需的RAM体积，您应该估计临时数据的大小 **GROUP BY**, **DISTINCT**, **JOIN** 和您使用的其他操作。

ClickHouse可以使用外部存储器来存储临时数据。看 [在外部存储器中分组](#) 有关详细信息。

交换文件

禁用生产环境的交换文件。

存储子系统

您需要有2GB的可用磁盘空间来安装ClickHouse。

数据所需的存储量应单独计算。评估应包括：

- 评估数据量。

您可以采取数据的样本并从中获取行的平均大小。然后将该值乘以计划存储的行数。

- 数据压缩系数。

要估计数据压缩系数，请将数据的样本加载到ClickHouse中，并将数据的实际大小与存储的表的大小进行比较。例如，点击流数据通常被压缩6-10倍。

要计算要存储的最终数据量，请将压缩系数应用于估计的数据量。如果计划将数据存储在多个副本中，则将估计的量乘以副本数。

网络

如果可能的话，使用10G或更高级别的网络。

网络带宽对于处理具有大量中间结果数据的分布式查询至关重要。此外，网络速度会影响复制过程。

软件

ClickHouse主要是为Linux系列操作系统开发的。推荐的Linux发行版是Ubuntu。`tzdata` 软件包应安装在系统中。

ClickHouse也可以在其他操作系统系列中工作。查看详细信息 [开始](#) 文档的部分。

监控

可以监控到：

- 硬件资源的利用率。
- ClickHouse 服务的指标。

硬件资源利用率

ClickHouse 本身不会去监控硬件资源的状态。

强烈推荐监控以下监控项：

- 处理器上的负载和温度。

可以使用 `dmesg`, `turbostat` 或者其他工具。

- 磁盘存储，RAM和网络的使用率。

ClickHouse 服务的指标。

ClickHouse服务本身具有用于自我状态监视指标。

要跟踪服务器事件，请观察服务器日志。请参阅配置文件的 **logger** 部分。

ClickHouse 收集的指标项：

- 服务用于计算的资源占用的各种指标。
- 关于查询处理的常见统计信息。

可以在 **系统指标**，**系统事件** 以及 **系统异步指标** 等系统表查看所有的指标项。

可以配置ClickHouse向**Graphite**推送监控信息并导入指标。参考**Graphite**监控配置文件。在配置指标导出之前，需要参考**Graphite官方教程**搭建**Graphite**服务。

此外，您可以通过HTTP API监视服务器可用性。将HTTP GET请求发送到/`ping`。如果服务器可用，它将以 `200 OK` 响应。

要监视服务器集群的配置，应设置**max_replica_delay_for_distributed_queries**参数并使用HTTP资源/`replicas_status`。如果副本可用，并且不延迟在其他副本之后，则对/`replicas_status`的请求将返回`200 OK`。如果副本滞后，请求将返回`503 HTTP_SERVICE_UNAVAILABLE`，包括有关待办事项大小的信息。

常见问题

- 安装
- 连接到服务器
- 查询处理
- 查询处理效率

安装

您无法使用Apt-get从ClickHouse存储库获取Deb软件包

- 检查防火墙设置。
- 如果出于任何原因无法访问存储库，请按照**开始**中的描述下载软件包，并使用命令 `sudo dpkg -i <packages>` 手动安装它们。除此之外你还需要 `tzdata` 包。

连接到服务器

可能出现的问题:

- 服务器未运行。
- 意外或错误的配置参数。

服务器未运行

检查服务器是否正在运行

命令:

```
$ sudo service clickhouse-server status
```

如果服务器没有运行，请使用以下命令启动它:

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

检查日志

主日志 `clickhouse-server` 默认情况是在 `/var/log/clickhouse-server/clickhouse-server.log` 下。

如果服务器成功启动，您应该看到字符串：

- `<Information> Application: starting up.` — Server started.
- `<Information> Application: Ready for connections.` — Server is running and ready for connections.

如果 `clickhouse-server` 启动失败与配置错误，你应该看到 `<Error>` 具有错误描述的字符串。例如：

```
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
```

如果在文件末尾没有看到错误，请从如下字符串开始查看整个文件：

```
<Information> Application: starting up.
```

如果您尝试在服务器上启动第二个实例 `clickhouse-server`，您将看到以下日志：

```
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
```

查看系统日志

如果你在 `clickhouse-server` 没有找到任何有用的信息或根本没有任何日志，您可以使用命令查看 `system.d`：

```
$ sudo journalctl -u clickhouse-server
```

在交互模式下启动**clickhouse**服务器

```
$ sudo -u clickhouse /usr/bin/clickhouse-server --config-file /etc/clickhouse-server/config.xml
```

此命令将服务器作为带有自动启动脚本标准参数的交互式应用程序启动。在这种模式下 `clickhouse-server` 打印控制台中的所有事件消息。

配置参数

检查：

- Docker设置。

如果您在IPv6网络中的Docker中运行ClickHouse，请确保 `network=host` 被设置。

■ 端点设置。

检查 `listen_host` 和 `tcp_port` 设置。

ClickHouse服务器默认情况下仅接受本地主机连接。

■ HTTP协议设置。

检查HTTP API的协议设置。

■ 安全连接设置。

检查:

- `tcp_port_secure` 设置。

- **SSL证书** 设置。

连接时使用正确的参数。例如，使用 `clickhouse-client` 的时候使用 `port_secure` 参数。

■ 用户设置。

您可能使用了错误的用户名或密码。

查询处理

如果ClickHouse无法处理查询，它会向客户端发送错误描述。在 `clickhouse-client` 您可以在控制台中获得错误的描述。如果您使用的是HTTP接口，ClickHouse会在响应正文中发送错误描述。例如：

```
$ 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
```

如果你使用 `clickhouse-client` 时设置了 `stack-trace` 参数，ClickHouse返回包含错误描述的服务器堆栈跟踪信息。

您可能会看到一条关于连接中断的消息。在这种情况下，可以重复查询。如果每次执行查询时连接中断，请检查服务器日志中是否存在错误。

查询处理效率

如果您发现ClickHouse工作速度太慢，则需要为查询分析服务器资源和网络的负载。

您可以使用`clickhouse-benchmark`实用程序来分析查询。它显示每秒处理的查询数、每秒处理的行数以及查询处理时间的百分位数。

更新

如果从deb包安装ClickHouse，请在服务器上执行以下命令:

```
$ sudo apt-get update
$ sudo apt-get install clickhouse-client clickhouse-server
$ sudo service clickhouse-server restart
```

如果您使用除推荐的deb包之外的其他方式安装ClickHouse，请使用适当的更新方法。

ClickHouse不支持分布式更新。该操作应在每个单独的服务器上连续执行。不要同时更新群集上的所有服务器，否则群集将在一段时间内不可用。

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`):

```
<yandex>
  <!-- ... -->
  <kerberos />
</yandex>
```

With principal specification:

```
<yandex>
  <!-- ... -->
  <kerberos>
    <principal>HTTP/clickhouse.example.com@EXAMPLE.COM</principal>
  </kerberos>
</yandex>
```

With filtering by realm:

```
<yandex>
  <!-- ... -->
  <kerberos>
    <realm>EXAMPLE.COM</realm>
  </kerberos>
</yandex>
```

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`):

```
<yandex>
  <!-- ... -->
  <users>
    <!-- ... -->
    <my_user>
      <!-- ... -->
      <kerberos>
        <realm>EXAMPLE.COM</realm>
      </kerberos>
    </my_user>
  </users>
</yandex>
```

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

```

<yandex>
  <!-- ... -->
  <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-ECDSSA-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>
</yandex>

```

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

```
<yandex>
  <!-- ... -->
  <users>
    <!-- ... -->
    <my_user>
      <!-- ... -->
      <ldap>
        <server>my_ldap_server</server>
      </ldap>
    </my_user>
  </users>
</yandex>
```

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`.

```
<yandex>
  <!-- ... -->
  <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>
</yandex>
```

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.

访问权限和账户管理

ClickHouse支持基于[RBAC](#)的访问控制管理。

ClickHouse权限实体包括：

- 用户账户
- 角色
- 行策略
- 设置描述
- 配额

你可以通过如下方式配置权限实体：

- 通过SQL驱动的工作流方式。

你需要[开启](#)这个功能。

- 服务端[配置文件](#) `users.xml` 和 `config.xml`.

我们建议你使用SQL工作流的方式。当然配置的方式也可以同时起作用，所以如果你正在用服务端配置的方式来管理权限和账户，你可以平滑的切换到SQL驱动的工作流方式。

警告

你无法同时使用两个配置的方式来管理同一个权限实体。

用法

默认ClickHouse提供了一个 `default` 账号，这个账号有所有的权限，但是不能使用SQL驱动方式的访问权限和账户管理。`default`主要用在用户名还未设置的情况，比如从客户端登录或者执行分布式查询。在分布式查询中如果服务端或者集群没有指定用户名密码那默认的账户就会被使用。

如果你刚开始使用ClickHouse，考虑如下场景：

1. 为 `default` 用户开启SQL驱动方式的访问权限和账户管理。
2. 使用 `default` 用户登录并且创建所需的所有用户。不要忘记创建管理员账户 (`GRANT ALL ON *.* WITH GRANT OPTION TO admin_user_account`)。
3. 限制 `default` 用户的权限并且禁用SQL驱动方式的访问权限和账户管理。

当前解决方案的特性

- 你甚至可以在数据库和表不存在的时候授予权限。
- 如果表被删除，和这张表关联的特权不会被删除。这意味着如果你创建一张同名的表，所有的特权仍旧有效。如果想删除这张表关联的特权，你可以执行 `REVOKE ALL PRIVILEGES ON db.table FROM ALL` 查询。
- 特权没有生命周期。

用户账户

用户账户是权限实体，用来授权操作ClickHouse，用户账户包含：

- 标识符信息。
- 特权用来定义用户可以执行的查询的范围。
- 可以连接到ClickHouse的主机。
- 指定或者默认的角色。
- 用户登录的时候默认的限制设置。
- 指定的设置描述。

特权可以通过`GRANT`查询授权给用户或者通过角色授予。如果想撤销特权，可以使用`REVOKE`查询。查询用户所有的特权，使用`SHOW GRANTS`语句。

查询管理：

- `CREATE USER`
- `ALTER USER`
- `DROP USER`
- `SHOW CREATE USER`

设置应用规则

对于一个用户账户来说，设置可以通过多种方式配置：通过角色扮演和设置描述。对于一个登陆的账号来说，如果一个设置对应了多个不同的权限实体，这些设置的应用规则如下（优先权从高到底）：

1. 用户账户设置。
2. 用户账号默认的角色设置。如果这个设置配置了多个角色，那设置的应用是没有规定的顺序。
3. 从设置描述分批给用户或者角色的设置。如果这个设置配置了多个角色，那设置的应用是没有规定的顺序。
4. 对所有服务器有效的默认或者**default profile**的设置。

角色

角色是权限实体的集合，可以被授予用户账号。

角色包括：

- 特权
- 设置和限制
- 分配的角色列表

查询管理：

- **CREATE ROLE**
- **ALTER ROLE**
- **DROP ROLE**
- **SET ROLE**
- **SET DEFAULT ROLE**
- **SHOW CREATE ROLE**

使用**GRANT** 查询可以把特权授予给角色。用**REVOKE**来撤回特权。

行策略

行策略是一个过滤器，用来定义哪些行数据可以被账户或者角色访问。对一个特定的表来说，行策略包括过滤器和使用这个策略的账户和角色。

查询管理：

- **CREATE ROW POLICY**
- **ALTER ROW POLICY**
- **DROP ROW POLICY**
- **SHOW CREATE ROW POLICY**

设置描述

设置描述是**设置**的汇总。设置汇总包括设置和限制，当然也包括这些描述的对象：角色和账户。

查询管理：

- **CREATE SETTINGS PROFILE**
- **ALTER SETTINGS PROFILE**
- **DROP SETTINGS PROFILE**
- **SHOW CREATE SETTINGS PROFILE**

配额

配额用来限制资源的使用情况。参考[配额](#)。

配额包括特定时间的限制条件和使用这个配额的账户和角色。

Management queries:

- [CREATE QUOTA](#)
- [ALTER QUOTA](#)
- [DROP QUOTA](#)
- [SHOW CREATE QUOTA](#)

开启SQL驱动方式的访问权限和账户管理

- 为配置的存储设置一个目录。

ClickHouse把访问实体的相关配置存储在[访问控制目录](#)，而这个目录可以通过服务端进行配置。

- 为至少一个账户开启SQL驱动方式的访问权限和账户管理。

默认情况下，SQL驱动方式的访问权限和账户管理对所有用户都是关闭的。你需要在 `users.xml` 中配置至少一个用户，并且把[权限管理](#)的值设置为1。

数据备份

尽管[副本](#)可以提供针对硬件的错误防护，但是它不能预防人为操作失误：数据的意外删除，错误表的删除或者错误集群上表的删除，以及导致错误数据处理或者数据损坏的软件bug。在很多案例中，这类意外可能会影响所有的副本。ClickHouse有内置的保护措施可以预防一些错误 — 例如，默认情况下[不能人工删除使用带有MergeTree引擎且包含超过50Gb数据的表](#)。但是，这些保护措施不能覆盖所有可能情况，并且这些措施可以被绕过。

为了有效地减少可能的人为错误，您应该提前仔细的准备备份和数据还原的策略。

不同公司有不同的可用资源和业务需求，因此不存在一个通用的解决方案可以应对各种情况下的ClickHouse备份和恢复。适用于1GB数据的方案可能并不适用于几十PB数据的情况。有多种具备各自优缺点的可能方法，将在下面对其进行讨论。最好使用几种方法而不是仅仅使用一种方法来弥补它们的各种缺点。。

注

需要注意的是，如果您备份了某些内容并且从未尝试过还原它，那么当您实际需要它时可能无法正常恢复（或者至少需要的时间比业务能够容忍的时间更长）。因此，无论您选择哪种备份方法，请确保自动还原过程，并定期在备用ClickHouse群集上演练。

将源数据复制到其它地方

通常摄入到ClickHouse的数据是通过某种持久队列传递的，例如[Apache Kafka](#)。在这种情况下，可以配置一组额外的订阅服务器，这些订阅服务器将在写入ClickHouse时读取相同的数据流，并将其存储在冷存储中。大多数公司已经有一些默认推荐的冷存储，可能是对象存储或分布式文件系统，如[HDFS](#)。

文件系统快照

某些本地文件系统提供快照功能（例如，[ZFS](#)），但它们可能不是提供实时查询的最佳选择。一个可能的解决方案是使用这种文件系统创建额外的副本，并将它们与用于SELECT查询的[分布式](#)表分离。任何修改数据的查询都无法访问此类副本上的快照。作为回报，这些副本可能具有特殊的硬件配置，每个服务器附加更多的磁盘，这将是经济高效的。

clickhouse-copier

clickhouse-copier 是一个多功能工具，最初创建它是为了用于重新切分pb大小的表。因为它能够在ClickHouse表和集群之间可靠地复制数据，所以它也可用于备份和还原数据。

对于较小的数据量，一个简单的 `INSERT INTO ... SELECT ...` 到远程表也可以工作。

part操作

ClickHouse允许使用 `ALTER TABLE ... FREEZE PARTITION ...` 查询以创建表分区的本地副本。这是利用硬链接(hardlink)到`/var/lib/clickhouse/shadow/`文件夹中实现的，所以它通常不会因为旧数据而占用额外的磁盘空间。创建的文件副本不由ClickHouse服务器处理，所以你可以把它们留在那里：你将有一个简单的备份，不需要任何额外的外部系统，但它仍然容易出现硬件问题。出于这个原因，最好将它们远程复制到另一个位置，然后删除本地副本。分布式文件系统和对象存储仍然是一个不错的选择，但是具有足够大容量的正常附加文件服务器也可以工作（在这种情况下，传输将通过网络文件系统或者也许是 `rsync` 来进行）。

数据可以使用 `ALTER TABLE ... ATTACH PARTITION ...` 从备份中恢复。

有关与分区操作相关的查询的详细信息，请参阅 [更改文档](#)。

第三方工具可用于自动化此方法: [clickhouse-backup](#).

采样查询探查器

ClickHouse运行允许分析查询执行的采样探查器。使用探查器，您可以找到在查询执行期间使用最频繁的源代码例程。您可以跟踪CPU时间和挂钟花费的时间，包括空闲时间。

使用概要分析器：

- 设置 `trace_log` 服务器配置部分。

本节配置 `trace_log` 系统表包含探查器运行的结果。它是默认配置的。请记住，此表中的数据仅对正在运行的服务器有效。服务器重新启动后，ClickHouse不会清理表，所有存储的虚拟内存地址都可能无效。

- 设置 `query_profiler_cpu_time_period_ns` 或 `query_profiler_real_time_period_ns` 设置。这两种设置可以同时使用。

这些设置允许您配置探查器计时器。由于这些是会话设置，您可以为整个服务器、单个用户或用户配置文件、交互式会话以及每个单个查询获取不同的采样频率。

默认采样频率为每秒一个采样，CPU和实时定时器都启用。该频率允许收集有关ClickHouse集群的足够信息。同时，使用此频率，profiler不会影响ClickHouse服务器的性能。如果您需要分析每个单独的查询，请尝试使用更高的采样频率。

分析 `trace_log` 系统表：

- 安装 `clickhouse-common-static-dbg` 包。看 [从DEB软件包安装](#)。
- 允许由内省功能 `allow_introspection_functions` 设置。

出于安全原因，默认情况下禁用内省功能。

- 使用 `addressToLine`, `addressToSymbol` 和 `demangle` 内省功能 获取函数名称及其在ClickHouse代码中的位置。要获取某些查询的配置文件，您需要从以下内容汇总数据 `trace_log` 桌子 您可以通过单个函数或整个堆栈跟踪聚合数据。

如果你需要想象 `trace_log` 信息，尝试 `flamegraph` 和 `测速镜`。

示例

在这个例子中，我们：

- 过滤 `trace_log` 数据由查询标识符和当前日期组成。
- 通过堆栈跟踪聚合。
- 使用内省功能，我们将得到一个报告：
 - 符号名称和相应的源代码函数。
 - 这些函数的源代码位置。

```

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<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
ThreadPoollImpl<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
ThreadPoollImpl<ThreadFromGlobalPool>::worker(std::list<ThreadFromGlobalPool>)
/usr/local/include/c++/9.1.0/x86_64-pc-linux-gnu/bits/gthr-default.h:748
ThreadFromGlobalPool::ThreadFromGlobalPool<ThreadPoollImpl<ThreadFromGlobalPool>::scheduleImpl<void>(std::function<void ()>, int, std::optional<unsigned long>){lambda()#3}>
(ThreadPoollImpl<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
ThreadPoollImpl<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 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, 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 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>::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 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/IO/BufferBase.h:99
```

_IPI_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/MergeTreeReaderStream.cpp:107
std::Function_handler<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
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
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&){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/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
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

```

系统表

引言

系统表提供的信息如下：

- 服务器的状态、进程以及环境。
- 服务器的内部进程。

系统表：

- 存储于 `system` 数据库。
- 仅提供数据读取功能。
- 不能被删除或更改，但可以对其进行分离(detach)操作。

大多数系统表将其数据存储在RAM中。一个ClickHouse服务在刚启动时便会创建此类系统表。

不同于其他系统表，系统日志表 `metric_log`, `query_log`, `query_thread_log`, `trace_log`, `part_log`, `crash_log` and `text_log` 默认采用 `MergeTree` 引擎并将其数据存储在文件系统中。如果人为的从文件系统中删除表，ClickHouse服务器会在下一次进行数据写入时再次创建空表。如果系统表结构在新版本中发生更改，那么ClickHouse会重命名当前表并创建一个新表。

用户可以通过在 `/etc/clickhouse-server/config.d/` 下创建与系统表同名的配置文件，或者在 `/etc/clickhouse-server/config.xml` 中设置相应配置项，来自定义系统日志表的结构。可供自定义的配置项如下：

- `database`: 系统日志表所在的数据库。这个选项目前已经不推荐使用。所有的系统日表都位于 `system` 库中。

- `table`: 接收数据写入的系统日志表。
- `partition_by`: 指定PARTITION BY表达式。
- `ttl`: 指定系统日志表TTL选项。
- `flush_interval_milliseconds`: 指定日志表数据刷新到磁盘的时间间隔。
- `engine`: 指定完整的表引擎定义。(以ENGINE =开头)。这个选项与`partition_by`以及`ttl`冲突。如果与两者一起设置，服务启动时会抛出异常并且退出。

配置定义的示例如下：

```
<yandex>
  <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>
</yandex>
```

默认情况下，表增长是无限的。可以通过TTL删除过期日志记录的设置来控制表的大小。你也可以使用分区功能MergeTree-引擎表。

系统指标的来源

用于收集ClickHouse服务器使用的系统指标：

- `CAP_NET_ADMIN` 能力。
- `procfs`（仅限于Linux）。

procfs

如果ClickHouse服务器没有 `CAP_NET_ADMIN` 能力，那么它将试图退回到 `ProcfsMetricsProvider`。`ProcfsMetricsProvider` 允许收集每个查询系统指标（包括CPU和I/O）。

如果系统上支持并启用procfs，ClickHouse server将收集如下指标：

- `OSCPUVirtualTimeMicroseconds`
- `OSCPUWaitMicroseconds`
- `OSIOWaitMicroseconds`
- `OSReadChars`
- `OSWriteChars`
- `OSReadBytes`
- `OSWriteBytes`

system.asynchronous_metric_log

包含以下内容的历史值 `system.asynchronous_log`（见 [系统。asynchronous_metrics](#)）

system.asynchronous_metrics

包含在后台定期计算的指标。例如，在使用的RAM量。

列：

- `metric` ([字符串](#)) — 指标名。
- `value` ([Float64](#)) — 指标值。

示例

```
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

另请参阅

- [监测](#) — ClickHouse监控的基本概念。
- [系统。指标](#) — 包含即时计算的指标。
- [系统。活动](#) — 包含出现的事件的次数。
- [系统。metric_log](#) — 包含[system.metrics](#) 和 [system.events](#)表中的指标的历史值。

system.clusters{ #system-clusters }

包含有关配置文件中可用的集群及其中的服务器的信息。

列：

- `cluster` ([String](#)) — 集群名。
- `shard_num` ([UInt32](#)) — 集群中的分片数，从1开始。
- `shard_weight` ([UInt32](#)) — 写数据时该分片的相对权重。
- `replica_num` ([UInt32](#)) — 分片的副本数量，从1开始。
- `host_name` ([String](#)) — 配置中指定的主机名。
- `host_address` ([String](#)) — 从DNS获取的主机IP地址。
- `port` ([UInt16](#)) — 连接到服务器的端口。
- `user` ([String](#)) — 连接到服务器的用户名。
- `errors_count` ([UInt32](#)) - 此主机无法访问副本的次数。
- `slowdowns_count` ([UInt32](#)) - 与对冲请求建立连接时导致更改副本的减速次数。
- `estimated_recovery_time` ([UInt32](#)) - 剩下的秒数，直到副本错误计数归零并被视为恢复正常。

请注意 `errors_count` 每个查询集群更新一次，但 `estimated_recovery_time` 按需重新计算。所以有可能是非零的情况 `errors_count` 和零 `estimated_recovery_time`，下一个查询将为零 `errors_count` 并尝试使用副本，就好像它没有错误。

另请参阅

- 表引擎分布式
- `distributed_replica_error_cap` 设置
- `distributed_replica_error_half_life` 设置

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.

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
```

Row 2:

```
database: default
table: users
name: contacts_null_idx
type: minmax
expr: assumeNotNull(contacts_null)
granularity: 1
```

system.data_type_families

包含有关受支持的 [数据类型](#) 的信息。

列字段包括：

- `name` ([String](#)) — 数据类型的名称。
- `case_insensitive` ([UInt8](#)) — 该属性显示是否可以在查询中以不区分大小写的方式使用数据类型名称。例如 `Date` 和 `date` 都是有效的。
- `alias_to` ([String](#)) — 名称为别名的数据类型名称。

示例

```
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

另请参阅

- [Syntax](#) — 关于所支持的语法信息。

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-00000000000
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-00000000000
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.functions

包含有关常规函数和聚合函数的信息。

列:

- `name(String)` – The name of the function.
- `is_aggregate(UInt8)` — Whether the function is aggregate.

举例

```
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.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.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.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:      1
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.query_log

包含已执行查询的相关信息，例如：开始时间、处理持续时间、错误消息。

注

此表不包含以下内容的摄取数据 `INSERT` 查询。

您可以更改`query_log`的设置，在服务器配置的 `query_log` 部分。

您可以通过设置 `log_queries=0` 来禁用`query_log`。我们不建议关闭此日志，因为此表中的信息对于解决问题很重要。

数据刷新的周期可通过 `flush_interval_milliseconds` 参数来设置 `query_log`。要强制刷新，请使用 `SYSTEM FLUSH LOGS`。

ClickHouse不会自动从表中删除数据。更多详情请看 [introduction](#)。

`system.query_log` 表注册两种查询：

1. 客户端直接运行的初始查询。
2. 由其他查询启动的子查询（用于分布式查询执行）。对于这些类型的查询，有关父查询的信息显示在 `initial_*` 列。

每个查询在 `query_log` 表中创建一或两行记录，这取决于查询的状态（见 `type` 列）：

- 如果查询执行成功，会创建 `type` 分别为 `QueryStart` 和 `QueryFinish` 的两行记录。
- 如果在查询处理过程中发生错误，会创建 `type` 分别为 `QueryStart` 和 `ExceptionWhileProcessing` 的两行记录。
- 如果在启动查询之前发生错误，则创建一行 `type` 为 `ExceptionBeforeStart` 的记录。

列：

- `type` (`Enum8`) — 执行查询时的事件类型。值：
 - '`QueryStart`' = 1 — 查询成功启动。
 - '`QueryFinish`' = 2 — 查询成功完成。
 - '`ExceptionBeforeStart`' = 3 — 查询执行前有异常。
 - '`ExceptionWhileProcessing`' = 4 — 查询执行期间有异常。
- `event_date` (`Date`) — 查询开始日期。
- `event_time` (`DateTime`) — 查询开始时间。
- `event_time_microseconds` (`DateTime64`) — 查询开始时间（毫秒精度）。
- `query_start_time` (`DateTime`) — 查询执行的开始时间。
- `query_start_time_microseconds` (`DateTime64`) — 查询执行的开始时间（毫秒精度）。
- `query_duration_ms` (`UInt64`) — 查询消耗的时间（毫秒）。
- `read_rows` (`UInt64`) — 从参与了查询的所有表和表函数读取的总行数。包括：普通的子查询, `IN` 和 `JOIN` 的子查询。对于分布式查询 `read_rows` 包括在所有副本上读取的行总数。每个副本发送它的 `read_rows` 值，并且查询的服务器-发起方汇总所有接收到的和本地的值。缓存卷不会影响此值。
- `read_bytes` (`UInt64`) — 从参与了查询的所有表和表函数读取的总字节数。包括：普通的子查询, `IN` 和 `JOIN` 的子查询。对于分布式查询 `read_bytes` 包括在所有副本上读取的字节总数。每个副本发送它的 `read_bytes` 值，并且查询的服务器-发起方汇总所有接收到的和本地的值。缓存卷不会影响此值。
- `written_rows` (`UInt64`) — 对于 `INSERT` 查询，为写入的行数。对于其他查询，值为 0。
- `written_bytes` (`UInt64`) — 对于 `INSERT` 查询时，为写入的字节数。对于其他查询，值为 0。
- `result_rows` (`UInt64`) — `SELECT` 查询结果的行数，或 `INSERT` 的行数。
- `result_bytes` (`UInt64`) — 存储查询结果的 RAM 量。
- `memory_usage` (`UInt64`) — 查询使用的内存。
- `query` (`String`) — 查询语句。
- `exception` (`String`) — 异常信息。
- `exception_code` (`Int32`) — 异常码。
- `stack_trace` (`String`) — `Stack Trace`。如果查询成功完成，则为空字符串。
- `is_initial_query` (`UInt8`) — 查询类型。可能的值：
 - 1 — 客户端发起的查询。
 - 0 — 由另一个查询发起的，作为分布式查询的一部分。
- `user` (`String`) — 发起查询的用户。
- `query_id` (`String`) — 查询 ID。

- `address (IPv6)` — 发起查询的客户端IP地址.
- `port (UInt16)` — 发起查询的客户端端口.
- `initial_user (String)` — 初始查询的用户名（用于分布式查询执行）.
- `initial_query_id (String)` — 运行初始查询的ID（用于分布式查询执行）.
- `initial_address (IPv6)` — 运行父查询的IP地址.
- `initial_port (UInt16)` — 发起父查询的客户端端口.
- `interface (UInt8)` — 发起查询的接口. 可能的值:
 - 1 — TCP.
 - 2 — HTTP.
- `os_user (String)` — 运行 `clickhouse-client` 的操作系统用户名.
- `client_hostname (String)` — 运行 `clickhouse-client` 或其他TCP客户端的机器的主机名。
- `client_name (String)` — `clickhouse-client` 或其他TCP客户端的名称。
- `client_revision (UInt32)` — `clickhouse-client` 或其他TCP客户端的Revision。
- `client_version_major (UInt32)` — `clickhouse-client` 或其他TCP客户端的Major version。
- `client_version_minor (UInt32)` — `clickhouse-client` 或其他TCP客户端的Minor version。
- `client_version_patch (UInt32)` — `clickhouse-client` 或其他TCP客户端的Patch component。
- `http_method (UInt8)` — 发起查询的HTTP方法. 可能值:
 - 0 — TCP接口的查询.
 - 1 — GET
 - 2 — POST
- `http_user_agent (String)` — The `UserAgent` The `UserAgent` header passed in the HTTP request。
- `quota_key (String)` — 在 `quotas` 配置里设置的“quota key”（见 `keyed`）.
- `revision (UInt32)` — ClickHouse revision.
- `ProfileEvents (Map(String, UInt64))` — Counters that measure different metrics. The description of them could be found in the table 系统。活动
- `Settings (Map(String, String))` — Names of settings that were changed when the client ran the query. To enable logging changes to settings, set the `log_query_settings` 参数为 1。
- `thread_ids (Array(UInt64))` — 参与查询的线程数.
- `Settings.Names (Array (String))` — 客户端运行查询时更改的设置的名称。要启用对设置的日志记录更改，请将 `log_query_settings` 参数设置为 1。
- `Settings.Values (Array (String))` — `Settings.Names` 列中列出的设置的值。

示例

```
SELECT * FROM system.query_log LIMIT 1 FORMAT Vertical;
```

Row 1:

```
type:          QueryStart
event_date:    2020-05-13
event_time:    2020-05-13 14:02:28
query_start_time: 2020-05-13 14:02:28
query_duration_ms: 0
read_rows:     0
read_bytes:    0
written_rows:  0
written_bytes: 0
result_rows:   0
result_bytes:  0
memory_usage: 0
query:         SELECT 1
exception_code: 0
exception:
stack_trace:
is_initial_query: 1
user:           default
query_id:       5e834082-6f6d-4e34-b47b-cd1934f4002a
address:        ::ffff:127.0.0.1
port:          57720
initial_user:   default
initial_query_id: 5e834082-6f6d-4e34-b47b-cd1934f4002a
initial_address: ::ffff:127.0.0.1
initial_port:   57720
interface:     1
os_user:        bayonet
client_hostname: clickhouse.ru-central1.internal
client_name:    ClickHouse client
client_revision: 54434
client_version_major: 20
client_version_minor: 4
client_version_patch: 1
http_method:   0
http_user_agent:
quota_key:
revision:      54434
thread_ids:    []
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':'2000000000','max_bytes_before_external_sort':'200000000000','allow_introspection_functions':'1'}
```

另请参阅

- [system.query_thread_log](#) — 这个表包含了每个查询执行线程的信息

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
':16,'SelectedRows':4,'SelectedBytes':48,'ContextLock':16,'RWLockAcquiredReadLocks':1,'RealTimeMicroseconds':698
,'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 `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.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_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::executeImpl(DB::QueryPipeline&, std::__1::shared_ptr<DB::IBlockInputStream> const&, std::__1::optional<DB::Pipe>)
DB::InterpreterSelectQuery::execute()
DB::InterpreterSelectWithUnionQuery::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.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
```

time_zone
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Asmera
Africa/Bamako
Africa/Bangui
Africa/Banjul
Africa/Bissau

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_regexp` ([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

如果未配置ZooKeeper，则该表不存在。 允许从配置中定义的ZooKeeper集群读取数据。

查询必须具有 'path' WHERE子句中的相等条件。 这是ZooKeeper中您想要获取数据的子路径。

查询 `SELECT * FROM system.zookeeper WHERE path = '/clickhouse'` 输出/clickhouse节点的对所有子路径的数据。

要输出所有根节点的数据，使用`path= '/'`。

如果在指定的路径 'path' 不存在，将引发异常。

查询`SELECT * FROM system.zookeeper WHERE path IN ('/', '/clickhouse')`输出/ 和 /clickhouse节点上所有子节点的数据。

如果在指定的 'path' 集合中有不存在的路径，将引发异常。

它可以用来做一批ZooKeeper路径查询。

列:

- `name` ([String](#)) — 节点的名字。
- `path` ([String](#)) — 节点的路径。
- `value` ([String](#)) — 节点的值。
- `dataLength` ([Int32](#)) — 节点的值长度。
- `numChildren` ([Int32](#)) — 子节点的个数。

- `czxid` (Int64) — 创建该节点的事务ID。
- `mzxid` (Int64) — 最后修改该节点的事务ID。
- `pzxid` (Int64) — 最后删除或者增加子节点的事务ID。
- `ctime` (DateTime) — 节点的创建时间。
- `mtime` (DateTime) — 节点的最后修改时间。
- `version` (Int32) — 节点版本：节点被修改的次数。
- `cversion` (Int32) — 增加或删除子节点的个数。
- `aversion` (Int32) — ACL的修改次数。
- `ephemeralOwner` (Int64) — 针对临时节点，拥有该节点的事务ID。

示例：

```
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
```

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 not requires 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:

- `xid` (`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_czid` (`Int64`) — The `zxid` of the change that caused this ZooKeeper node to be created.
- `stat_mzxid` (`Int64`) — The `zxid` 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](#)

系统。detached_parts

包含有关分离部分的信息 MergeTree 桌子 该 reason 列指定分离部件的原因。

对于用户分离的部件，原因是空的。这些部件可以附加 ALTER TABLE ATTACH PARTITION|PART 指挥部
有关其他列的说明，请参阅 系统。零件.

如果部件名称无效，某些列的值可能为 NULL. 这些部分可以删除 ALTER TABLE DROP DETACHED PART.

系统。graphite_retentions

包含有关参数的信息 graphite_rollup 这是在表中使用 *GraphiteMergeTree 引擎

列:

- config_name (字符串) - graphite_rollup 参数名称。
- regexp (String)-指标名称的模式。
- function (String)-聚合函数的名称。
- age (UInt64)-以秒为单位的数据的最小期限。
- precision (UInt64) -如何精确地定义以秒为单位的数据的年龄。
- priority (UInt16)-模式优先级。
- is_default (UInt8)-模式是否为默认值。
- Tables.database (Array(String))-使用数据库表名称的数组 config_name 参数。
- Tables.table (Array(String))-使用表名称的数组 config_name 参数。

系统。merge_tree_settings

包含有关以下设置的信息 MergeTree 桌子

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.

系统。metric_log

包含表中度量值的历史记录 system.metrics 和 system.events，定期刷新到磁盘。

打开指标历史记录收集 system.metric_log, 创建 /etc/clickhouse-server/config.d/metric_log.xml 具有以下内容:

```
<yandex>
  <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>
</yandex>
```

示例

```
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
...	

另请参阅

- 系统。[asynchronous_metrics](#) — Contains periodically calculated metrics.
- 系统。[活动](#) — Contains a number of events that occurred.
- 系统。[指标](#) — Contains instantly calculated metrics.
- [监测](#) — Base concepts of ClickHouse monitoring.

系统。numbers_mt

一样的 [系统。数字](#) 但读取是并行的。 这些数字可以以任何顺序返回。

用于测试。

系统。part_log

该 `system.part_log` 表只有当创建 [part_log](#) 指定了服务器设置。

此表包含与以下情况发生的事件有关的信息 [数据部分](#) 在 [MergeTree](#) 家庭表，例如添加或合并数据。

该 `system.part_log` 表包含以下列：

- `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’ 值，如果分区是由 `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.

该 `system.part_log` 表的第一个插入数据到后创建 MergeTree 桌子

系统。`query_thread_log`

包含有关执行查询的线程的信息，例如，线程名称、线程开始时间、查询处理的持续时间。

开始记录：

1. 在配置参数 `query_thread_log` 科。
2. 设置 `log_query_threads` 到1。

数据的冲洗周期设置在 `flush_interval_milliseconds` 的参数 `query_thread_log` 服务器设置部分。要强制冲洗，请使用 [SYSTEM FLUSH LOGS](#) 查询。

ClickHouse不会自动从表中删除数据。看 [导言](#) 欲了解更多详情。

列:

- `event_date` (日期) — The date when the thread has finished execution of the query.
- `event_time` (日期时间) — The date and time when the thread has finished execution of the query.
- `query_start_time` (日期时间) — 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` 查询, 写入的行数。对于其他查询, 列值为0。
- `written_bytes` (UInt64) — For `INSERT` 查询时, 写入的字节数。对于其他查询, 列值为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` (字符串) — 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` (字符串) — 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` (字符串) — Name of the user who initiated the current query.
- `query_id` (字符串) — 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` (字符串) — Name of the user who ran the initial query (for distributed query execution).
- `initial_query_id` (字符串) — 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` (字符串) — OS's username who runs `客户端`。
- `client_hostname` (字符串) — Hostname of the client machine where the `客户端` 或者运行另一个 TCP 客户端。

- `client_name` (字符串) — The 嘴环板 client 嘴嘉ツッ偲 或另一个TCP客户端名称。
- `client_revision` (UInt32) — Revision of the 嘴环板 client 嘴嘉ツッ偲 或另一个TCP客户端。
- `client_version_major` (UInt32) — Major version of the 嘴环板 client 嘴嘉ツッ偲 或另一个TCP客户端。
- `client_version_minor` (UInt32) — Minor version of the 嘴环板 client 嘴嘉ツッ偲 或另一个TCP客户端。
- `client_version_patch` (UInt32) — Patch component of the 嘴环板 client 嘴嘉ツッ偲 或另一个TCP客户端版本。
- `http_method` (UInt8) — HTTP method that initiated the query. Possible values:
 - 0 — The query was launched from the TCP interface.
 - 1 — GET 方法被使用。
 - 2 — POST 方法被使用。
- `http_user_agent` (字符串) — The UserAgent http 请求中传递的标头。
- `quota_key` (字符串) — The “quota key” 在指定 配额 设置 (见 keyed).
- `revision` (UInt32) — ClickHouse revision.
- `ProfileEvents` (数组 (字符串, UInt64)) — Counters that measure different metrics for this thread. The description of them could be found in the table 系统。活动.

示例

```
SELECT * FROM system.query_thread_log LIMIT 1 FORMAT Vertical
```

Row 1:

```
event_date:      2020-05-13
event_time:     2020-05-13 14:02:28
query_start_time: 2020-05-13 14:02:28
query_duration_ms: 0
read_rows:       1
read_bytes:      1
written_rows:    0
written_bytes:   0
memory_usage:    0
peak_memory_usage: 0
thread_name:    QueryPipelineEx
thread_id:       28952
master_thread_id: 28924
query:          SELECT 1
is_initial_query: 1
user:           default
query_id:        5e834082-6f6d-4e34-b47b-cd1934f4002a
address:         ::ffff:127.0.0.1
port:            57720
initial_user:    default
initial_query_id: 5e834082-6f6d-4e34-b47b-cd1934f4002a
initial_address: ::ffff:127.0.0.1
initial_port:    57720
interface:       1
os_user:         bayonet
client_hostname: clickhouse.ru-central1.internal
client_name:      ClickHouse client
client_revision:  54434
client_version_major: 20
client_version_minor: 4
client_version_patch: 1
http_method:     0
http_user_agent:
quota_key:
revision:        54434
ProfileEvents:
{'Query':1,'SelectQuery':1,'ReadCompressedBytes':36,'CompressedReadBufferBlocks':1,'CompressedReadBufferBytes':10,'IOBufferAllocs':1,'IOBufferAllocBytes':89,'ContextLock':15,'RWLockAcquiredReadLocks':1}
...
```

另请参阅

- 系统。[query_log](#) — Description of the `query_log` system table, which contains general information about query execution.

系统。`storage_policies`

包含有关存储策略和卷中定义的信息 [服务器配置](#).

列:

- `policy_name` ([字符串](#)) — Name of the storage policy.
- `volume_name` ([字符串](#)) — Volume name defined in the storage policy.
- `volume_priority` ([UInt64](#)) — Volume order number in the configuration.
- `disks` ([数组（字符串）](#)) — 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.

如果存储策略包含多个卷，则每个卷的信息将存储在表的单独行中。

系统。text_log

包含日志记录条目。进入该表的日志记录级别可以通过以下方式进行限制 `text_log.level` 服务器设置。

列：

- `event_date` (Date) — Date of the entry.
- `event_time` (DateTime) — Time of the entry.
- `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 或 'Fatal'.
 - 2 或 'Critical'.
 - 3 或 'Error'.
 - 4 或 'Warning'.
 - 5 或 'Notice'.
 - 6 或 'Information'.
 - 7 或 'Debug'.
 - 8 或 '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.

系统。trace_log

包含采样查询探查器收集的堆栈跟踪。

ClickHouse创建此表时 `trace_log` 服务器配置部分被设置。也是 `query_profiler_real_time_period_ns` 和 `query_profiler_cpu_time_period_ns` 应设置设置。

要分析日志，请使用 `addressToLine`, `addressToSymbol` 和 `demangle` 内省功能。

列：

- `event_date` (日期) — Date of sampling moment.
- `event_time` (日期时间) — Timestamp of the sampling moment.
- `timestamp_ns` (UInt64) — Timestamp of the sampling moment in nanoseconds.

- `revision` (`UInt32`) — ClickHouse server build revision.

通过以下方式连接到服务器 `clickhouse-client`，你看到的字符串类似于 `Connected to ClickHouse server version 19.18.1 revision 54429..` 该字段包含 `revision`，但不是 `version` 的服务器。

- `timer_type` (`枚举8`) — Timer type:

- `Real` 表示挂钟时间。
- `CPU` 表示CPU时间。

- `thread_number` (`UInt32`) — Thread identifier.

- `query_id` (`字符串`) — Query identifier that can be used to get details about a query that was running from the `query_log` 系统表.

- `trace` (`数组(UInt64)`) — Stack trace at the moment of sampling. Each element is a virtual memory address inside ClickHouse server process.

示例

```
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]
```

系统。一

此表包含一行，其中包含一行 `dummy UInt8` 列包含值 0。

如果使用此表 `SELECT` 查询不指定 `FROM` 条款

这类似于 `DUAL` 表在其他Dbms中找到。

系统。列

包含有关所有表中列的信息。

您可以使用此表获取类似于以下内容的信息 `DESCRIBE TABLE` 查询，但对于多个表一次。

该 `system.columns` 表包含以下列 (列类型显示在括号中):

- `database` (`String`) — Database name.
- `table` (`String`) — Table name.
- `name` (`String`) — Column name.
- `type` (`String`) — Column type.

- `default_kind` (String) — Expression type (DEFAULT, MATERIALIZED, ALIAS)为默认值，如果没有定义，则为空字符串。
- `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.

系统。副本

包含驻留在本地服务器上的复制表的信息和状态。

此表可用于监视。该表对于每个已复制的*表都包含一行。

示例：

```
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`)-数据库名称
 - `table` (`String`)-表名
 - `engine` (`String`)-表引擎名称
 - `is_leader` (`UInt8`)-副本是否是领导者。
一次只有一个副本可以成为领导者。领导者负责选择要执行的后台合并。
请注意，可以对任何可用且在ZK中具有会话的副本执行写操作，而不管该副本是否为leader。
 - `can_become_leader` (`UInt8`)-副本是否可以当选为领导者。
 - `is_READONLY` (`UInt8`)-副本是否处于只读模式。
如果配置没有ZooKeeper的部分，如果在ZooKeeper中重新初始化会话时发生未知错误，以及在ZooKeeper中重新初始化会话时发生未知错误，则此模式将打开。
 - `is_session_expired` (`UInt8`)-与ZooKeeper的会话已经过期。基本上一样 `is_READONLY`.
 - `future_parts` (`UInt32`)-由于尚未完成的插入或合并而显示的数据部分的数量。
 - `parts_to_check` (`UInt32`)-队列中用于验证的数据部分的数量。如果怀疑零件可能已损坏，则将其放入验证队列。
 - `zookeeper_path` (`String`)-在ZooKeeper中的表数据路径。
 - `replica_name` (`String`)-在动物园管理员副本名称。同一表的不同副本具有不同的名称。
 - `replica_path` (`String`)-在ZooKeeper中的副本数据的路径。与连接相同 '`zookeeper_path/replicas/replica_path`'.
 - `columns_version` (`Int32`)-表结构的版本号。指示执行ALTER的次数。如果副本有不同的版本，这意味着一些副本还没有做出所有的改变。
 - `queue_size` (`UInt32`)-等待执行的操作的队列大小。操作包括插入数据块、合并和某些其他操作。它通常与 `future_parts`.
 - `inserts_in_queue` (`UInt32`)-需要插入数据块的数量。插入通常复制得相当快。如果这个数字很大，这意味着有什么不对劲。
 - `merges_in_queue` (`UInt32`)-等待进行合并的数量。有时合并时间很长，因此此值可能长时间大于零。
 - `part_mutations_in_queue` (`UInt32`) -等待进行的突变的数量。
 - `queue_oldest_time` (`DateTime`)-如果 `queue_size` 大于0，显示何时将最旧的操作添加到队列中。
 - `inserts_oldest_time` (`DateTime`) -看 `queue_oldest_time`
 - `merges_oldest_time` (`DateTime`) -看 `queue_oldest_time`
 - `part_mutations_oldest_time` (`DateTime`) -看 `queue_oldest_time`
- 接下来的4列只有在有ZK活动会话的情况下才具有非零值。
- `log_max_index` (`UInt64`)-一般活动日志中的最大条目数。
 - `log_pointer` (`UInt64`)-副本复制到其执行队列的常规活动日志中的最大条目数加一。如果 `log_pointer` 比 `log_max_index`，有点不对劲。
 - `last_queue_update` (`DateTime`)-上次更新队列时。
 - `absolute_delay` (`UInt64`) -当前副本有多大滞后秒。
 - `total_replicas` (`UInt8`)-此表的已知副本总数。

- `active_replicas` (UInt8)-在ZooKeeper中具有会话的此表的副本的数量（即正常运行的副本的数量）。

如果您请求所有列，表可能会工作得有点慢，因为每行都会从ZooKeeper进行几次读取。

如果您没有请求最后4列（`log_max_index`，`log_pointer`，`total_replicas`，`active_replicas`），表工作得很快。

例如，您可以检查一切是否正常工作，如下所示：

```
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
```

如果这个查询没有返回任何东西，这意味着一切都很好。

系统。合并

包含有关MergeTree系列中表当前正在进行的合并和部件突变的信息。

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如果这个过程是一个部分突变。
- `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.

系统。字典

包含以下信息 [外部字典](#).

列:

- `database` (字符串) — Name of the database containing the dictionary created by DDL query. Empty string for other dictionaries.
- `name` (字符串) — 字典名称.
- `status` (枚举8) — 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](#) 查询, 超时, 字典配置已更改).
 - `FAILED_AND_RELOADING` — Could not load the dictionary as a result of an error and is loading now.
- `origin` (字符串) — Path to the configuration file that describes the dictionary.
- `type` (字符串) — Type of a dictionary allocation. 在内存中存储字典.
- `key` — 密钥类型: 数字键 (UInt64) or Composite key (字符串) — form "(type 1, type 2, ..., type n)".
- `attribute.names` (阵列(字符串)) — Array of 属性名称 由字典提供。
- `attribute.types` (阵列(字符串)) — Corresponding array of 属性类型 这是由字典提供。
- `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` (字符串) — Text describing the 数据源 为了字典
- `lifetime_min` (UInt64) — Minimum 使用寿命 在内存中的字典, 之后ClickHouse尝试重新加载字典 (如果 `invalidate_query` 被设置, 那么只有当它已经改变). 在几秒钟内设置。
- `lifetime_max` (UInt64) — Maximum 使用寿命 在内存中的字典, 之后ClickHouse尝试重新加载字典 (如果 `invalidate_query` 被设置, 那么只有当它已经改变). 在几秒钟内设置。
- `loading_start_time` (日期时间) — Start time for loading the dictionary.

- `last_successful_update_time` (日期时间) — 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` (字符串) — Text of the error that occurs when creating or reloading the dictionary if the dictionary couldn't be created.

示例

配置字典。

```
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())
```

确保字典已加载。

```
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	lifetime_min	lifetime_max	loading_start_time	last_successful_update_time	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	

系统。指标

包含可以立即计算或具有当前值的指标。例如，同时处理的查询的数量或当前副本的延迟。此表始终是最新的。

列：

- `metric` (字符串) — Metric name.
- `value` (Int64) — Metric value.
- `description` (字符串) — Metric description.

支持的指标列表，您可以在 [src/Common/CurrentMetrics.cpp](#) ClickHouse的源文件。

示例

```
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.

另请参阅

- 系统。[asynchronous_metrics](#) — Contains periodically calculated metrics.
- 系统。[活动](#) — Contains a number of events that occurred.
- 系统。[metric_log](#) — Contains a history of metrics values from tables `system.metrics` и `system.events`.
- [监测](#) — Base concepts of ClickHouse monitoring.

系统。数字

此表包含一个名为 UInt64 的列 `number` 它包含几乎所有从零开始的自然数。

您可以使用此表进行测试，或者如果您需要进行暴力搜索。

从此表中读取的内容不是并行的。

系统。数据库

此表包含一个名为"字符串"的列 '`name`' – the name of a database.

服务器知道的每个数据库在表中都有相应的条目。

该系统表用于实现 `SHOW DATABASES` 查询。

系统。活动

包含有关系统中发生的事件数的信息。例如，在表中，您可以找到多少 `SELECT` 自 ClickHouse 服务器启动以来已处理查询。

列:

- `event` ([字符串](#)) — Event name.

- `value` ([UInt64](#)) — Number of events occurred.
- `description` ([字符串](#)) — Event description.

示例

```
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.

另请参阅

- [系统。asynchronous_metrics](#) — Contains periodically calculated metrics.
- [系统。指标](#) — Contains instantly calculated metrics.
- [系统。metric_log](#) — Contains a history of metrics values from tables `system.metrics` и `system.events`.
- [监测](#) — Base concepts of ClickHouse monitoring.

系统。流程

该系统表用于实现 `SHOW PROCESSLIST` 查询。

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` 用户。该字段包含特定查询的用户名，而不是此查询启动的查询的用户名。
- `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` 查询请求者服务器上。
- `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` 设置。
- `query` (String) – The query text. For `INSERT`, it does not include the data to be inserted.
- `query_id` (String) – Query ID, if defined.

系统。磁盘

包含有关在定义的磁盘信息 [服务器配置](#).

列:

- `name` (字符串) — Name of a disk in the server configuration.
- `path` (字符串) — 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` 磁盘配置参数。

系统。storage_policies

包含有关存储策略和卷中定义的信息 [服务器配置](#).

列:

- `policy_name` (字符串) — Name of the storage policy.
- `volume_name` (字符串) — Volume name defined in the storage policy.
- `volume_priority` (UInt64) — Volume order number in the configuration.
- `disks` (数组 (字符串)) — 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.

如果存储策略包含多个卷，则每个卷的信息将存储在表的单独行中。

系统。突变

该表包含以下信息 [突变](#) MergeTree表及其进展。每个突变命令由一行表示。该表具有以下列:

数据库, 表 - 应用突变的数据库和表的名称。

mutation_id - 变异的ID 对于复制的表，这些Id对应于znode中的名称 `<table_path_in_zookeeper>/mutations/` 动物园管理员的目录。对于未复制的表，Id对应于表的数据目录中的文件名。

命令 - Mutation命令字符串 (查询后的部分 `ALTER TABLE [db.]table`).

create_time - 当这个突变命令被提交执行。

block_numbers.partition_id, block_numbers. 编号 - 嵌套列。对于复制表的突变，它包含每个分区的一条记录：分区ID和通过突变获取的块编号（在每个分区中，只有包含编号小于该分区中突变获取的块编号的块的 在非复制表中，所有分区中的块编号形成一个序列。这意味着对于非复制表的突变，该列将包含一条记录，其中包含由突变获取的单个块编号。

parts_to_do - 为了完成突变，需要突变的数据部分的数量。

is_done - 变异完成了?? 请注意，即使 `parts_to_do = 0` 由于长时间运行的INSERT将创建需要突变的新数据部分，因此可能尚未完成复制表的突变。

如果在改变某些部分时出现问题，以下列将包含其他信息：

latest_failed_part - 不能变异的最新部分的名称。

latest_fail_time - 最近的部分突变失败的时间。

latest_fail_reason - 导致最近部件变异失败的异常消息。

系统。表

包含服务器知道的每个表的元数据。分离的表不显示在 `system.tables`。

此表包含以下列（列类型显示在括号中）：

- `database` (String) — 表所在的数据库表名。
- `name` (String) — 表名。
- `engine` (String) — 表引擎名（不包含参数）。
- `is_temporary` (UInt8)-指示表是否是临时的标志。
- `data_path` (String)-文件系统中表数据的路径。
- `metadata_path` (String)-文件系统中表元数据的路径。
- `metadata_modification_time` (DateTime)-表元数据的最新修改时间。
- `dependencies_database` (数组(字符串))-数据库依赖关系。
- `dependencies_table` (数组(字符串)) - 表依赖关系 (**MaterializedView** 基于当前表的表)。
- `create_table_query` (String)-用于创建表的SQL语句。
- `engine_full` (String)-表引擎的参数。
- `partition_key` (String)-表中指定的分区键表达式。
- `sorting_key` (String)-表中指定的排序键表达式。
- `primary_key` (String)-表中指定的主键表达式。
- `sampling_key` (String)-表中指定的采样键表达式。
- `storage_policy` (字符串)-存储策略：
 - **MergeTree**
 - 分布

- `total_rows` (`Nullable(UInt64)`) - 总行数，如果可以快速确定表中的确切行数，否则行数为 `Null` (包括底层 `Buffer` 表)。
- `total_bytes` (`Nullable(UInt64)`) - 总字节数，如果可以快速确定存储表的确切字节数，否则字节数为 `Null` (即不包括任何底层存储)。
 - 如果表将数据存在磁盘上，返回实际使用的磁盘空间 (压缩后)。
 - 如果表在内存中存储数据，返回在内存中使用的近似字节数。
- `lifetime_rows` (`Nullable(UInt64)`) - 服务启动后插入的总行数(只针对 `Buffer` 表)。

`system.tables` 表被用于 `SHOW TABLES` 的查询实现中。

系统。表_engines

包含服务器支持的表引擎的描述及其功能支持信息。

此表包含以下列 (列类型显示在括号中):

- `name` (`String`) — The name of table engine.
- `supports_settings` (`UInt8`) — Flag that indicates if table engine supports `SETTINGS` 条款
- `supports_skipping_indices` (`UInt8`) — Flag that indicates if table engine supports 跳过索引.
- `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` 和 `SAMPLE_BY`.
- `supports_replication` (`UInt8`) — Flag that indicates if table engine supports 数据复制.
- `supports_deduplication` (`UInt8`) — Flag that indicates if table engine supports data deduplication.

示例:

```
SELECT *
FROM system.table_engines
WHERE name in ('Kafka', 'MergeTree', 'ReplicatedCollapsingMergeTree')
```

name	supports_settings	supports_skipping_indices	supports_sort_order	sup
	ports_ttl	supports_replication	supports_deduplication	ports
Kafka	1	0	0	0
MergeTree	1	1	1	0
ReplicatedCollapsingMergeTree	1	1	1	1

另请参阅

- 梅树家族 [查询子句](#)
- 卡夫卡 [设置](#)
- 加入我们 [设置](#)

系统。设置

包含有关当前用户的会话设置的信息。

列：

- `name` (字符串) — Setting name.
- `value` (字符串) — Setting value.
- `changed` (UInt8) — Shows whether a setting is changed from its default value.
- `description` (字符串) — Short setting description.
- `min` (可为空(字符串)) — Minimum value of the setting, if any is set via **制约因素**. 如果设置没有最小值，则包含 **NULL**.
- `max` (可为空(字符串)) — Maximum value of the setting, if any is set via **制约因素**. 如果设置没有最大值，则包含 **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.

示例

下面的示例演示如何获取有关名称包含的设置的信息 `min_i`.

```
SELECT *
FROM system.settings
WHERE name LIKE '%min_i%'
```

name	value	changed	description
<code>min_insert_block_size_rows</code>	1048576	0	Squash blocks passed to INSERT query to specified size in rows, if blocks are not big enough.
<code>min_insert_block_size_bytes</code>	268435456	0	Squash blocks passed to INSERT query to specified size in bytes, if blocks are not big enough.
<code>read_backoff_min_interval_between_events_ms</code>	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.

使用 `WHERE changed` 可以是有用的，例如，当你想检查：

- 配置文件中的设置是否正确加载并正在使用。
- 在当前会话中更改的设置。

```
SELECT * FROM system.settings WHERE changed AND name='load_balancing'
```

另请参阅

- [设置](#)
- [查询权限](#)
- [对设置的限制](#)

系统。贡献者

包含有关贡献者的信息。该顺序在查询执行时是随机的。

列:

- `name` (String) — Contributor (author) name from git log.

示例

```
SELECT * FROM system.contributors LIMIT 10
```

name
Olga Khvostikova
Max Vetrov
LiuYangkuan
svladykin
zamulla
Šimon Podlipský
BayoNet
Ilya Khomutov
Amy Krishnevsky
Loud_Scream

要在表中找出自己，请使用查询:

```
SELECT * FROM system.contributors WHERE name = 'Olga Khvostikova'
```

name
Olga Khvostikova

系统。零件

包含有关的部分信息 [MergeTree](#) 桌子

每行描述一个数据部分。

列:

- `partition` (String) – The partition name. To learn what a partition is, see the description of the [ALTER](#) 查询。

格式:

- `YYYYMM` 用于按月自动分区。
- `any_string` 手动分区时。
- `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` 通过索引粒度 (通常为8192) (此提示不适用于自适应粒度)。

- `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](#) 的压缩文件。
- `hash_of_uncompressed_files` (String) – [sipHash128](#) 未压缩的文件（带标记的文件，索引文件等。）。

- `uncompressed_hash_of_compressed_files` (`String`) – `sipHash128` 压缩文件中的数据，就好像它们是未压缩的。
- `bytes` (`UInt64`) – Alias for `bytes_on_disk`.
- `marks_size` (`UInt64`) – Alias for `marks_bytes`.

如何使用ClickHouse测试您的硬件

使用此指令，您可以在任何服务器上运行基本的ClickHouse性能测试，而无需安装ClickHouse软件包。

1. 转到“commits”页数：<https://github.com/ClickHouse/ClickHouse/commits/master>
2. 点击第一个绿色复选标记或红色十字与绿色“ClickHouse Build Check”然后点击“Details”附近链接“ClickHouse Build Check”。在一些提交中没有这样的链接，例如与文档的提交。在这种情况下，请选择具有此链接的最近提交。
3. 将链接复制到“clickhouse”二进制为amd64或aarch64。
4. ssh到服务器并使用wget下载它：

```
# 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. 下载配置：

```
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. 下载基准测试文件：

```
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. 根据下载测试数据 [Yandex梅里卡数据集](#) 说明 (“hits”表包含100万行)。

```
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. 运行服务器：

```
./clickhouse server
```

1. 检查数据：ssh到另一个终端中的服务器

```
./clickhouse client --query "SELECT count() FROM hits_100m_obfuscated"  
100000000
```

1. 编辑benchmark-new.sh，改变 clickhouse-client 到 ./clickhouse client 并添加 --max_memory_usage 1000000000000 参数。

```
mcedit benchmark-new.sh
```

1. 运行基准测试：

```
./benchmark-new.sh hits_100m_obfuscated
```

1. 将有关硬件配置的编号和信息发送到clickhouse-feedback@yandex-team.com

所有结果都在这里公布：<https://clickhouse.技术/基准/硬件/>

[experimental] OpenTelemetry Support

OpenTelemetry 是一个开放标准，用于从分布式应用程序收集踪迹和指标。ClickHouse 对 OpenTelemetry 有一些支持。

Warning

This is an experimental feature that will change in backwards-incompatible ways in future releases.

Supplying Trace Context to ClickHouse

ClickHouse 接受踪迹上下文 HTTP 头部，如由 W3C 推荐的那样。它也接受踪迹上下文通过一个原生协议，该协议用于 ClickHouse 服务器之间或客户端与服务器之间的通信。对于手动测试，踪迹上下文头部符合踪迹上下文推荐，可以使用 --opentelemetry-traceparent 和 --opentelemetry-tracestate 标志提供给 clickhouse-client。

如果未提供父踪迹上下文，则 ClickHouse 可以开始一个新的踪迹，其概率由 opentelemetry_start_trace_probability 配置设置控制。

Propagating the Trace Context

踪迹上下文在以下情况下被传播到下游服务：

- 对于远程 ClickHouse 服务器的查询，例如在使用 Distributed 表引擎时。
- url 表函数。踪迹上下文信息以 HTTP 头部的形式发送。

Tracing the ClickHouse Itself

ClickHouse 为每个查询创建踪迹 spans，并为某些查询执行阶段，例如查询规划或分布式查询，创建一些。

要使踪迹信息有用，必须将其导出到支持 OpenTelemetry 的监控系统，例如 Jaeger 或 Prometheus。ClickHouse 避免对特定的监控系统有依赖性，而是通过系统表提供踪迹数据。OpenTelemetry 踪迹 span 信息 required by the standard 存储在 system.opentelemetry_span_log 表中。

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:

```
<yandex>
  <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>
</yandex>
```

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:

```

<yandex>
  <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">00112233445566778899aabbcdddeeff</key_hex>
        <key_hex id="1">ffeeddccbba99887766554433221100</key_hex>
        <current_key_id>1</current_key_id>
      </disk_s3_encrypted>
    </disks>
  </storage_configuration>
</yandex>

```

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:

```
<yandex>
  <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>
</yandex>
```

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.

使用建议

CPU频率调节器

始终使用 `performance` 频率调节器。 `on-demand` 频率调节器在持续高需求的情况下，效果更差。

```
echo 'performance' | sudo tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
```

CPU限制

处理器可能会过热。 使用 `dmesg` 查看CPU的时钟速率是否由于过热而受到限制。

该限制也可以在数据中心级别外部设置。 您可以使用 `turbostat` 在负载下对其进行监控。

RAM

对于少量数据（压缩后约200GB），最好使用与数据量一样多的内存。

对于大量数据，以及在处理交互式（在线）查询时，应使用合理数量的RAM（128GB或更多），以便热数据子集适合页面缓存。

即使对于每台服务器约50TB的数据量，与64GB相比，使用128GB的RAM也可以显着提高查询性能。

不要禁用 `overcommit`。 `cat /proc/sys/vm/overcommit_memory` 的值应该为0或1。运行

```
$ echo 0 | sudo tee /proc/sys/vm/overcommit_memory
```

大页(Huge Pages)

始终禁用透明大页(`transparent huge pages`)。 它会干扰内存分配器，从而导致显着的性能下降。

```
echo 'never' | sudo tee /sys/kernel/mm/transparent_hugepage/enabled
```

使用 `perf top` 来查看内核在内存管理上花费的时间。

永久大页(`permanent huge pages`)也不需要被分配。

存储子系统

如果您的预算允许您使用SSD，请使用SSD。

如果没有，请使用硬盘。 SATA硬盘7200转就行了。

优先选择许多带有本地硬盘驱动器的服务器，而不是少量带有附加磁盘架的服务器。

但是对于存储极少查询的档案，架子可以使用。

RAID

当使用硬盘，你可以结合他们的RAID-10，RAID-5，RAID-6或RAID-50。

对于Linux，软件RAID更好（使用 `mdadm`）。我们不建议使用LVM。

当创建RAID-10，选择 `far` 布局。

如果您的预算允许，请选择RAID-10。

如果您有4个以上的磁盘，请使用RAID-6（首选）或RAID-50，而不是RAID-5。

当使用RAID-5、RAID-6或RAID-50时，始终增加`stripe_cache_size`，因为默认值通常不是最佳选择。

```
echo 4096 | sudo tee /sys/block/md2/md/stripe_cache_size
```

使用以下公式从设备数量和块大小中计算出确切的数量： $2 * \text{num_devices} * \text{chunk_size_in_bytes} / 4096$ 。

1024KB的块大小足以满足所有RAID配置。

切勿将块大小设置得太小或太大。

您可以在SSD上使用RAID-0。

无论使用哪种RAID，始终使用复制来保证数据安全。

启用有长队列的NCQ。对于HDD，选择CFQ调度程序，对于SSD，选择noop。不要减少‘readahead’设置。

对于HDD，启用写入缓存。

文件系统

Ext4是最可靠的选择。设置挂载选项 `noatime, nobarrier`。

XFS也是合适的，但它还没有经过ClickHouse的全面测试。

大多数其他文件系统也应该可以正常工作。具有延迟分配的文件系统工作得更好。

Linux内核

不要使用过时的Linux内核。

网络

如果使用的是IPv6，请增加路由缓存的大小。

3.2之前的Linux内核在IPv6实现方面存在许多问题。

如果可能的话，至少使用10GB的网络。1GB也可以工作，但对于使用数十TB的数据修补副本或处理具有大量中间数据的分布式查询，情况会更糟。

虚拟机监视器(Hypervisor)配置

如果您使用的是OpenStack，请在nova.conf中设置

```
cpu_mode=host-passthrough
```

。

如果您使用的是libvirt，请在XML配置中设置

```
<cpu mode='host-passthrough'/>
```

。

这对于ClickHouse能够通过 `cpuid` 指令获取正确的信息非常重要。

否则，当在旧的CPU型号上运行虚拟机监视器时，可能会导致 `Illegal instruction` 崩溃。

Zookeeper

您可能已经将ZooKeeper用于其他目的。如果它还没有超载，您可以使用相同的zookeeper。

最好使用新版本的Zookeeper – 3.4.9 或更高的版本。稳定的Liunx发行版中的Zookeeper版本可能已过时。

你永远不要使用手动编写的脚本在不同的Zookeeper集群之间传输数据，这可能会导致序列节点的数据不正确。出于相同的原因，永远不要使用 zkcopy 工具: <https://github.com/ksprojects/zkcopy/issues/15>

如果要将现有的ZooKeeper集群分为两个，正确的方法是增加其副本的数量，然后将其重新配置为两个独立的集群。

不要在ClickHouse所在的服务器上运行ZooKeeper。因为ZooKeeper对延迟非常敏感，而ClickHouse可能会占用所有可用的系统资源。

默认设置下，ZooKeeper 就像是一个定时炸弹：

当使用默认配置时，ZooKeeper服务器不会从旧的快照和日志中删除文件（请参阅autopurge），这是操作员的责任。

必须拆除炸弹。

下面的ZooKeeper（3.5.1）配置在 Yandex.Metrica 的生产环境中使用截至2017年5月20日：

zoo.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版本:

```

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参数:

```

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"

```

初始化:

```

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

```

配置文件

ClickHouse 支持多配置文件管理。主配置文件是 /etc/clickhouse-server/config.xml。其余文件须在目录 /etc/clickhouse-server/config.d。

!!! 注意：

所有配置文件必须是 XML 格式。此外，配置文件须有相同的跟元素，通常是 <yandex>。

主配置文件中的一些配置可以通过 replace 或 remove 属性被配置文件覆盖。

如果两者都未指定，则递归组合配置的内容，替换重复子项的值。

如果指定 replace 属性，则将整个元素替换为指定的元素。

如果指定 remove 属性，则删除该元素。

此外，配置文件还可指定 "substitutions"。如果一个元素有 incl 属性，则文件中的相应替换值将被使用。默认情况下，具有替换的文件的路径为 /etc/metrika.xml。这可以在服务配置中的 include_from 元素中被修改。替换值在这个文件的 /clickhouse/substitution_name 元素中被指定。如果 incl 中指定的替换值不存在，则将其记录在日志中。为防止 ClickHouse 记录丢失的替换，请指定 optional="true" 属性（例如，宏设置）。

替换也可以从 ZooKeeper 执行。为此，请指定属性 from_zk = "/path/to/node"。元素值被替换为 ZooKeeper 节点 /path/to/node 的内容。您还可以将整个 XML 子树放在 ZooKeeper 节点上，并将其完全插入到源元素中。

config.xml 文件可以指定单独的配置文件用于配置用户设置、配置文件及配额。可在 users_config 元素中指定其配置文件相对路径。其默认值是 users.xml。如果 users_config 被省略，用户设置、配置文件和配额则直接在 config.xml 中指定。

用户配置可以分为如 config.xml 和 config.d 等形式的单独配置文件。目录名称为配置 user_config 的值，去掉 .xml 后缀并与添加 .d。由于 users_config 配置默认值为 users.xml，所以目录名默认使用 users.d。例如，您可以为每个用户有单独的配置文件，如下所示：

```
$ cat /etc/clickhouse-server/users.d/alice.xml
```

```
<yandex>
<users>
  <alice>
    <profile>analytics</profile>
    <networks>
      <ip>::/0</ip>
    </networks>
    <password_sha256_hex>...</password_sha256_hex>
    <quota>analytics</quota>
  </alice>
</users>
</yandex>
```

对于每个配置文件，服务器还会在启动时生成 file-preprocessed.xml 文件。这些文件包含所有已完成的替换和复盖，并且它们旨在提供信息。如果 zookeeper 替换在配置文件中使用，但 ZooKeeper 在服务器启动时不可用，则服务器将从预处理的文件中加载配置。

服务器跟踪配置文件中的更改，以及执行替换和复盖时使用的文件和 ZooKeeper 节点，并动态重新加载用户和集群的设置。这意味着您可以在不重新启动服务器的情况下修改群集、用户及其设置。

配额

配额允许您在一段时间内限制资源使用情况，或者只是跟踪资源的使用。

配额在用户配置中设置。这通常是 'users.xml'.

The system also has a feature for limiting the complexity of a single query. See the section «Restrictions on query complexity»).

与查询复杂性限制相比，配额：

- 对可以在一段时间内运行的一组查询设置限制，而不是限制单个查询。
- 占用在所有远程服务器上用于分布式查询处理的资源。

让我们来看看部分 ‘users.xml’ 定义配额的文件。

```
<!-- 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>
```

默认情况下，配额只跟踪每小时的资源消耗，而不限制使用情况。

每次请求后，计算出的每个时间间隔的资源消耗将输出到服务器日志中。

```
<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>100000000000</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>500000000000</read_rows>
    <execution_time>7200</execution_time>
  </interval>
</statbox>
```

为 ‘statbox’ 配额，限制设置为每小时和每24小时（86,400秒）。时间间隔从实现定义的固定时刻开始计数。换句话说，24小时间隔不一定从午夜开始。

间隔结束时，将清除所有收集的值。在下一个小时内，配额计算将重新开始。

以下是可以限制的金额：

queries – The total number of requests.

errors – The number of queries that threw an exception.

`result_rows` – The total number of rows given as the 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).

如果在至少一个时间间隔内超出限制，则会引发异常，其中包含有关超出了哪个限制、哪个时间间隔以及新时间间隔开始时（何时可以再次发送查询）的文本。

Quotas can use the «quota key» feature in order to report on resources for multiple keys independently. Here is an example of this:

```
<!-- 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 />
```

配额分配给用户 ‘users’ 部分的配置。见“访问权限”部分。

For distributed query processing, the accumulated amounts are stored on the requestor server. So if the user goes to another server, the quota there will «start over».

服务器重新启动时，将重置配额。

clickhouse-local

`clickhouse-local` 模式可以使您能够对本地文件执行快速处理，而无需部署和配置 ClickHouse 服务器。

接受表示表格 `tables` 的数据，并使用 [ClickHouse SQL 方言](#) 查询它们。

`clickhouse-local` 使用与 ClickHouse Server 相同的核心，因此它支持大多数功能以及相同的格式和表引擎。

默认情况下 `clickhouse-local` 不能访问同一主机上的数据，但它支持使用 `--config-file` 方式加载服务器配置。

警告

不建议将生产服务器配置加载到 `clickhouse-local` 因为数据可以在人为错误的情况下被损坏。

对于临时数据，默认情况下会创建一个唯一的临时数据目录。

用途

基本用法：

```
clickhouse-local --structure "table_structure" --input-format "format_of_incoming_data" -q "query"
```

参数：

- `-S, --structure` — 输入数据的表结构。

- `-if, --input-format` — 输入格式化类型，默认是`TSV`。
- `-f, --file` — 数据路径，默认是`stdin`。
- `-q, --query` — 要查询的SQL语句使用;做分隔符。您必须指定`query`或`queries-file`选项。
- `-qf, --queries-file` - 包含执行查询的文件路径。您必须指定`query`或`queries-file`选项。
- `-N, --table` — 数据输出的表名，默认是`table`。
- `-of, --format, --output-format` — 输出格式化类型，默认是`TSV`。
- `-d, --database` — 默认数据库名，默认是`_local`。
- `--stacktrace` — 是否在出现异常时输出栈信息。
- `--echo` — 执行前打印查询。
- `--verbose` — `debug`显示查询的详细信息。
- `--logger.console` — 日志显示到控制台。
- `--logger.log` — 日志文件名。
- `--logger.level` — 日志级别。
- `--ignore-error` — 当查询失败时，不停止处理。
- `-c, --config-file` — 与ClickHouse服务器格式相同配置文件的路径，默认情况下配置为空。
- `--no-system-tables` — 不附加系统表。
- `--help` — `clickhouse-local`使用帮助信息。
- `-V, --version` — 打印版本信息并退出。

对于每个ClickHouse配置的参数，也可以单独使用，可以不使用`--config-file`指定。

示例

```
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
```

另一个示例，类似上一个使用示例：

```
$ 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"
Read 2 rows, 32.00 B in 0.000 sec., 4987 rows/sec., 77.93 KiB/sec.
1 2
3 4
```

你可以使用`stdin`或`--file`参数，打开任意数量的文件来使用多个文件`file table function`：

```
$ echo 1 | tee 1.tsv
1

$ echo 2 | tee 2.tsv
2

$ clickhouse-local --query "
  select * from file('1.tsv', TSV, 'a int') t1
  cross join file('2.tsv', TSV, 'b int') t2"
1 2
```

现在让我们查询每个Unix用户使用内存：

```
$ 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
...	

性能测试

连接到ClickHouse服务器并重复发送指定的查询。

语法：

```
$ echo "single query" | clickhouse-benchmark [keys]
```

或

```
$ clickhouse-benchmark [keys] <<< "single query"
```

如果要发送一组查询，请创建一个文本文件，并将每个查询的字符串放在此文件中。例如：

```
SELECT * FROM system.numbers LIMIT 10000000
SELECT 1
```

然后将此文件传递给标准输入 `clickhouse-benchmark`.

```
clickhouse-benchmark [keys] < queries_file
```

keys参数

- `-c N, --concurrency=N` — Number of queries that `clickhouse-benchmark` 同时发送。默认值：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`. 为 比较模式 您可以使用多个 `-h` 参数
- `-p N, --port=N` — Server port. Default value: 9000. For the 比较模式 您可以使用多个 `-p` 钥匙

- `-i N, --iterations=N` — 查询的总次数. Default value: 0.
- `-r, --randomize` — 有多个查询时，以随机顺序执行.
- `-s, --secure` — 使用TLS安全连接.
- `-t N, --timelimit=N` — Time limit in seconds. `clickhouse-benchmark` 达到指定的时间限制时停止发送查询。默认值：0（禁用时间限制）。
- `--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 比较模式 `clickhouse-benchmark` 执行 独立双样本学生的t测试 测试以确定两个分布是否与所选置信水平没有不同。
- `--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` 将报告输出到指定的JSON文件。
- `--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` 输出异常的堆栈跟踪。
- `--stage=WORD` — 查询请求的服务端处理状态. 在特定阶段Clickhouse会停止查询处理，并返回结果给`clickhouse-benchmark`。可能的值: `complete`, `fetch_columns`, `with_mergeable_state`. 默认值: `complete`.
- `--help` — Shows the help message.

如果你想在查询时应用上述的部分参数 **设置**，请将它们作为键传递 `--<session setting name>= SETTING_VALUE` 例如, `--max_memory_usage=1048576`.

输出

默认情况下, `clickhouse-benchmark` 按照 `--delay` 参数间隔输出结果。

报告示例:

```
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.
```

在结果报告中，您可以找到:

- 查询数量：参见`Queries executed`:字段。

- 状态码（按顺序给出）：

- ClickHouse服务器的连接信息。
- 已处理的查询数。
- QPS：服务端每秒处理的查询数量
- RPS：服务器每秒读取多少行
- MiB/s：服务器每秒读取多少字节的数据
- 结果RPS：服务端每秒生成多少行的结果集数据
- 结果MiB/s.服务端每秒生成多少字节的结果集数据
- 查询执行时间的百分比。

对比模式

`clickhouse-benchmark` 可以比较两个正在运行的ClickHouse服务器的性能。

要使用对比模式，分别为每个服务器配置各自的`--host`, `--port`参数。`clickhouse-benchmark` 会根据设置的参数建立到各个 Server的连接并发送请求。每个查询请求会随机发送到某个服务器。输出结果会按服务器分组输出

示例

```
$ 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-copier

将数据从一个群集中的表复制到另一个（或相同）群集中的表。

您可以运行多个 `clickhouse-copier` 不同服务器上的实例执行相同的作业。ZooKeeper用于同步进程。

开始后, `clickhouse-copier`:

- 连接到ZooKeeper并且接收:
 - 复制作业。
 - 复制作业的状态。
- 它执行的工作。

每个正在运行的进程都会选择源集群的“最接近”分片，然后将数据复制到目标集群，并在必要时重新分片数据。

`clickhouse-copier` 跟踪ZooKeeper中的更改，并实时应用它们。

为了减少网络流量，我们建议运行 `clickhouse-copier` 在源数据所在的同一服务器上。

运行Clickhouse-copier

该实用程序应手动运行:

```
clickhouse-copier --daemon --config zookeeper.xml --task-path /task/path --base-dir /path/to/dir
```

参数：

- **daemon** — 在守护进程模式下启动clickhouse-copier。
- **config** — zookeeper.xml文件的路径，其中包含用于连接ZooKeeper的参数。
- **task-path** — ZooKeeper节点的路径。该节点用于同步clickhouse-copier进程和存储任务。任务存储在\$task-path/description中。
- **task-file** — 可选的非必须参数，指定一个包含任务配置的参数文件，用于初始上传到ZooKeeper。
- **task-upload-force** — 即使节点已经存在，也强制上载task-file。
- **base-dir** — 日志和辅助文件的路径。启动时，clickhouse-copier在\$base-dir中创建clickhouse-copier_YYYYMMHHSS_<PID>子目录。如果省略此参数，则会在启动clickhouse-copier的目录中创建目录。

Zookeeper.xml格式

```
<yandex>
  <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>
</yandex>
```

复制任务的配置

```
<yandex>
  <!-- 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>
    ...
  </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>
        <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.
        <!-- 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>  
</yandex>
```

clickhouse-copier 跟踪更改 `/task/path/description` 并在飞行中应用它们。例如，如果你改变的值 `max_workers`，运行任务的进程数也会发生变化。

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
```

实用工具

- **本地查询** — 在不停止ClickHouse服务的情况下，对数据执行查询操作(类似于 `awk` 命令)。
- **跨集群复制** — 在不同集群间复制数据。
- **性能测试** — 连接到Clickhouse服务器，执行性能测试。

服务器配置

`builtin_dictionaries_reload_interval`

重新加载内置字典的间隔时间（以秒为单位）。

ClickHouse每x秒重新加载内置字典。这使得编辑字典“on the fly”，而无需重新启动服务器。

默认值:3600.

示例

```
<builtin_dictionaries_reload_interval>3600</builtin_dictionaries_reload_interval>
```

压缩

数据压缩配置 MergeTree 引擎表。

警告

如果您刚开始使用ClickHouse，请不要使用它。

配置模板:

```
<compression>
  <case>
    <min_part_size>...</min_part_size>
    <min_part_size_ratio>...</min_part_size_ratio>
    <method>...</method>
  </case>
  ...
</compression>
```

`<case>` 参数:

- `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` 或 `zstd`.

您可以配置多个 `<case>` 部分。

满足条件时的操作:

- 如果数据部分与条件集匹配，ClickHouse将使用指定的压缩方法。
- 如果数据部分匹配多个条件集，ClickHouse将使用第一个匹配的条件集。

如果没有满足数据部分的条件，ClickHouse使用 `lz4` 压缩。

示例

```
<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

默认数据库。

要获取数据库列表，请使用 **SHOW DATABASES** 查询。

示例

```
<default_database>default</default_database>
```

default_profile

默认配置文件。

配置文件位于 **user_config** 参数指定的文件中。

示例

```
<default_profile>default</default_profile>
```

dictionaries_config

外部字典的配置文件的路径。

路径：

- 指定相对于服务器配置文件的绝对路径或路径。
- 路径可以包含通配符*和?.

另请参阅 “[外部字典](#)”。

示例

```
<dictionaries_config>*_dictionary.xml</dictionaries_config>
```

dictionaries_lazy_load

延迟加载字典。

如果 **true**，然后在第一次使用时创建每个字典。如果字典创建失败，则使用该字典的函数将引发异常。

如果 **false**，服务器启动时创建所有字典，如果出现错误，服务器将关闭。

默认值为 **true**.

示例

```
<dictionaries_lazy_load>true</dictionaries_lazy_load>
```

format_schema_path

包含输入数据方案的目录路径，例如输入数据的方案 [CapnProto](#) 格式。

示例

```
<!-- Directory containing schema files for various input formats. -->
<format_schema_path>format_schemas/</format_schema_path>
```

石墨

将数据发送到 石墨。

设置：

- 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 系统。指标 桌子
- events – Sending deltas data accumulated for the time period from the 系统。活动 桌子
- events_cumulative – Sending cumulative data from the 系统。活动 桌子
- asynchronous_metrics – Sending data from the 系统。asynchronous_metrics 桌子

您可以配置多个 `<graphite>` 条款 例如，您可以使用它以不同的时间间隔发送不同的数据。

示例

```
<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

石墨细化数据的设置。

有关详细信息，请参阅 [GraphiteMergeTree](#).

示例

```
<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

通过HTTP连接到服务器的端口。

如果 https_port 被指定, **openSSL** 必须配置。

如果 http_port 指定时, 即使设置了OpenSSL配置, 也会忽略该配置。

示例

```
<https_port>9999</https_port>
```

http_server_default_response

访问ClickHouse HTTP(s)服务器时默认显示的页面。

默认值为 “Ok.” (最后有换行符)

示例

打开 <https://tabix.io/> 访问时 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

带替换的文件的路径。

有关详细信息, 请参阅部分 “[配置文件](#)”.

示例

```
<include_from>/etc/metrica.xml</include_from>
```

interserver_http_port

用于在ClickHouse服务器之间交换数据的端口。

示例

```
<interserver_http_port>9009</interserver_http_port>
```

interserver_http_host

其他服务器可用于访问此服务器的主机名。

如果省略, 它以相同的方式作为定义 `hostname-f` 指挥部

用于脱离特定的网络接口。

示例

```
<interserver_http_host>example.yandex.ru</interserver_http_host>
```

interserver_http_credentials

用户名和密码用于在以下期间进行身份验证 复制 与复制*引擎。这些凭据仅用于副本之间的通信，与ClickHouse客户端的凭据无关。服务器正在检查这些凭据以连接副本，并在连接到其他副本时使用相同的凭据。因此，这些凭据应该为集群中的所有副本设置相同。

默认情况下，不使用身份验证。

本节包含以下参数：

- `user` — `username`.
- `password` — `password`.

示例

```
<interserver_http_credentials>
  <user>admin</user>
  <password>222</password>
</interserver_http_credentials>
```

keep_alive_timeout

ClickHouse在关闭连接之前等待传入请求的秒数。默认为10秒。

示例

```
<keep_alive_timeout>10</keep_alive_timeout>
```

listen_host

对请求可能来自的主机的限制。如果您希望服务器回答所有这些问题，请指定`::`。

例：

```
<listen_host>::1</listen_host>
<listen_host>127.0.0.1</listen_host>
```

记录器

日志记录设置。

键：

- `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` and `errorlog`. 一旦文件到达 `size`，ClickHouse存档并重命名它，并在其位置创建一个新的日志文件。
- `count` – The number of archived log files that ClickHouse stores.

示例

```
<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>
```

还支持写入系统日志。配置示例：

```
<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>
```

键：

- `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` — 系统日志工具关键字 在大写字母与“LOG_”前缀：(LOG_USER, LOG_DAEMON, LOG_LOCAL3, 等等)。默认值: `LOG_USER` 如果 `address` 被指定, `LOG_DAEMON` otherwise.
- `format` – Message format. Possible values: `bsd` 和 `syslog`.

宏

复制表的参数替换。

如果不使用复制的表，则可以省略。

有关详细信息，请参阅部分“[创建复制的表](#)”。

示例

```
<macros incl="macros" optional="true" />
```

mark_cache_size

表引擎使用的标记缓存的近似大小（以字节为单位）[MergeTree](#) 家人

缓存为服务器共享，并根据需要分配内存。缓存大小必须至少为5368709120。

示例

```
<mark_cache_size>5368709120</mark_cache_size>
```

max_concurrent_queries

同时处理的请求的最大数量。

示例

```
<max_concurrent_queries>100</max_concurrent_queries>
```

max_connections

入站连接的最大数量。

示例

```
<max_connections>4096</max_connections>
```

max_open_files

打开文件的最大数量。

默认情况下: `maximum`.

我们建议在Mac OS X中使用此选项，因为 `getrlimit()` 函数返回一个不正确的值。

示例

```
<max_open_files>262144</max_open_files>
```

max_table_size_to_drop

限制删除表。

如果一个大小 `MergeTree` 表超过 `max_table_size_to_drop`（以字节为单位），您无法使用删除查询将其删除。

如果仍然需要在不重新启动ClickHouse服务器的情况下删除表，请创建 `<clickhouse-path>/flags/force_drop_table` 文件并运行DROP查询。

默认值：50GB。

值0表示您可以删除所有表而不受任何限制。

示例

```
<max_table_size_to_drop>0</max_table_size_to_drop>
```

merge_tree

微调中的表 `MergeTree`.

有关详细信息，请参阅 `MergeTreeSettings.h` 头文件。

示例

```
<merge_tree>
  <max_suspicious_broken_parts>5</max_suspicious_broken_parts>
</merge_tree>
```

openSSL

SSL客户端/服务器配置。

对SSL的支持由 `libpoco` 图书馆。该接口在文件中描述 `SSLManager.h`

服务器/客户端设置的密钥:

- 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` 包含证书。
- 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 A. 背景 同学们 可能的值: `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`. 可接受的值: `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`. 始终建议使用此参数，因为如果服务器缓存会话，以及客户端请求缓存，它有助于避免出现问题。默认值: `${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.

设置示例:

```

<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

记录与之关联的事件 MergeTree。例如，添加或合并数据。您可以使用日志来模拟合并算法并比较它们的特征。您可以可视化合并过程。

查询记录在 [系统。part_log](#) 表，而不是在一个单独的文件。您可以在以下命令中配置此表的名称 `table` 参数（见下文）。

使用以下参数配置日志记录：

- `database` – Name of the database.
- `table` – Name of the system table.
- `partition_by` – Sets a [自定义分区键](#)。
- `flush_interval_milliseconds` – Interval for flushing data from the buffer in memory to the table.

示例

```

<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>

```

路径

包含数据的目录的路径。

注

尾部斜杠是强制性的。

示例

```
<path>/var/lib/clickhouse/</path>
```

query_log

用于记录接收到的查询的设置 `log_queries=1` 设置。

查询记录在 `系统.query_log` 表，而不是在一个单独的文件。您可以更改表的名称 `table` 参数（见下文）。

使用以下参数配置日志记录：

- `database` – Name of the database.
- `table` – Name of the system table the queries will be logged in.
- `partition_by` – Sets a `自定义分区键` 为了一张桌子
- `flush_interval_milliseconds` – Interval for flushing data from the buffer in memory to the table.

如果该表不存在，ClickHouse将创建它。如果在ClickHouse服务器更新时查询日志的结构发生了更改，则会重命名具有旧结构的表，并自动创建新表。

示例

```
<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

设置用于记录接收到的查询的线程 `log_query_threads=1` 设置。

查询记录在 `系统.query_thread_log` 表，而不是在一个单独的文件。您可以更改表的名称 `table` 参数（见下文）。

使用以下参数配置日志记录：

- `database` – Name of the database.
- `table` – Name of the system table the queries will be logged in.
- `partition_by` – Sets a `自定义分区键` 对于一个系统表。
- `flush_interval_milliseconds` – Interval for flushing data from the buffer in memory to the table.

如果该表不存在，ClickHouse将创建它。如果更新ClickHouse服务器时查询线程日志的结构发生了更改，则会重命名具有旧结构的表，并自动创建新表。

示例

```
<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

设置为 `trace_log` 系统表操作。

参数:

- `database` — Database for storing a table.
- `table` — Table name.
- `partition_by` — 自定义分区键 对于一个系统表。
- `flush_interval_milliseconds` — Interval for flushing data from the buffer in memory to the table.

默认服务器配置文件 `config.xml` 包含以下设置部分:

```
<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

基于正则表达式的规则，在将查询以及所有日志消息存储在服务器日志中之前，这些规则将应用于查询以及所有日志消息，`system.query_log`, `system.text_log`, `system.processes` 表，并在日志中发送给客户端。这允许防止从SQL查询敏感数据泄漏（如姓名，电子邮件，个人标识符或信用卡号码）记录。

示例

```
<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>
```

配置字段:

- `name` - 规则的名称（可选）
- `regexp` - RE2 兼容正则表达式（强制性）
- `replace` - 敏感数据的替换字符串（可选，默认情况下-六个星号）

屏蔽规则应用于整个查询（以防止敏感数据从格式错误/不可解析的查询泄漏）。

`system.events` 表有计数器 `QueryMaskingRulesMatch` 其中具有匹配的查询屏蔽规则的总数。

对于分布式查询，每个服务器必须单独配置，否则，子查询传递给其他节点将被存储而不屏蔽。

remote_servers

所使用的集群的配置 分布 表引擎和由 `cluster` 表功能。

示例

```
<remote_servers incl="clickhouse_remote_servers" />
```

对于该值 `incl` 属性，请参阅部分 “[配置文件](#)”。

另请参阅

- [skip_unavailable_shards](#)

时区

服务器的时区。

指定为UTC时区或地理位置（例如，非洲/阿比让）的IANA标识符。

当DateTime字段输出为文本格式（打印在屏幕上或文件中）时，以及从字符串获取DateTime时，时区对于字符串和DateTime格式之间的转换是必需的。此外，如果在输入参数中没有收到时区，则时区用于处理时间和日期的函数。

示例

```
<timezone>Europe/Moscow</timezone>
```

tcp_port

通过TCP协议与客户端通信的端口。

示例

```
<tcp_port>9000</tcp_port>
```

tcp_port_secure

TCP端口，用于与客户端进行安全通信。使用它与 [OpenSSL](#) 设置。

可能的值

整数。

默认值

```
<tcp_port_secure>9440</tcp_port_secure>
```

mysql_port

通过MySQL协议与客户端通信的端口。

可能的值

整数。

示例

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

tmp_path

用于处理大型查询的临时数据的路径。

注

尾部斜杠是强制性的。

示例

```
<tmp_path>/var/lib/clickhouse/tmp/</tmp_path>
```

tmp_policy

从政策 `storage_configuration` 存储临时文件。
如果没有设置 `tmp_path` 被使用，否则被忽略。

注

- `move_factor` 被忽略

- `keep_free_space_bytes` 被忽略
- `max_data_part_size_bytes` 被忽略
 - 您必须在该政策中只有一个卷

uncompressed_cache_size

表引擎使用的未压缩数据的缓存大小（以字节为单位）`MergeTree`.

服务器有一个共享缓存。 内存按需分配。 如果选项使用缓存 `use_uncompressed_cache` 被启用。

在个别情况下，未压缩的缓存对于非常短的查询是有利的。

示例

```
<uncompressed_cache_size>8589934592</uncompressed_cache_size>
```

user_files_path

包含用户文件的目录。 在表函数中使用 [文件\(\)](#).

示例

```
<user_files_path>/var/lib/clickhouse/user_files/</user_files_path>
```

users_config

包含文件的路径:

- 用户配置。
- 访问权限。
- 设置配置文件。
- 配额设置。

示例

```
<users_config>users.xml</users_config>
```

zookeeper

包含允许ClickHouse与 **zookpeer** 集群。

ClickHouse使用ZooKeeper存储复制表副本的元数据。如果未使用复制的表，则可以省略此部分参数。

本节包含以下参数：

- **node** — ZooKeeper endpoint. You can set multiple endpoints.

例如：

```
<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**被用作根由ClickHouse服务器使用**znodes** 可选。
- **identity** — User and password, that can be required by ZooKeeper to give access to requested znodes. Optional.

配置示例

```
<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>
```

另请参阅

- [复制](#)
- [zookeeper管理指南](#)

use_minimalistic_part_header_in_zookeeper

ZooKeeper中数据部分头的存储方法。

此设置仅适用于 **MergeTree** 家人 它可以指定：

- 在全球范围内 **merge_tree** 一节 **config.xml** 文件

ClickHouse使用服务器上所有表的设置。您可以随时更改设置。当设置更改时，现有表会更改其行为。

- 对于每个表。

创建表时，指定相应的 [发动机设置](#)。即使全局设置更改，具有此设置的现有表的行为也不会更改。

可能的值

- 0 — Functionality is turned off.
- 1 — Functionality is turned on.

如果 `use_minimalistic_part_header_in_zookeeper = 1`，然后 [复制](#) 表存储的数据部分的头紧凑使用一个单一的 znode。如果表包含许多列，则此存储方法显着减少了 Zookeeper 中存储的数据量。

注意

申请后 `use_minimalistic_part_header_in_zookeeper = 1`，您不能将 ClickHouse 服务器降级到不支持此设置的版本。在集群中的服务器上升级 ClickHouse 时要小心。不要一次升级所有服务器。在测试环境中或在集群的几台服务器上测试 ClickHouse 的新版本更安全。

Data part headers already stored with this setting can't be restored to their previous (non-compact) representation.

默认值: 0.

disable_internal_dns_cache

禁用内部 DNS 缓存。推荐用于在系统中运行 ClickHouse 随着频繁变化的基础设施，如 Kubernetes。

默认值: 0.

dns_cache_update_period

更新存储在 ClickHouse 内部 DNS 缓存中的 IP 地址的周期（以秒为单位）。更新是在一个单独的系统线程中异步执行的。

默认值: 15.

服务器配置参数

本节包含无法在会话或查询级别更改的服务器设置的说明。

这些设置存储在 `config.xml` ClickHouse 服务器上的文件。

Other settings are described in the «[设置](#)» section.

在研究设置之前，请阅读 [配置文件](#) 部分和注意使用替换（的 `incl` 和 `optional` 属性）。

查询权限

ClickHouse 中的查询可以分为几种类型：

1. 读取数据查询: `SELECT`, `SHOW`, `DESCRIBE`, `EXISTS`.
2. 写入数据查询: `INSERT`, `OPTIMIZE`.
3. 更改设置查询: `SET`, `USE`.
4. [DDL](#) 查询: `CREATE`, `ALTER`, `RENAME`, `ATTACH`, `DETACH`, `DROP`, `TRUNCATE`.

5. KILL QUERY.

以下设置按查询类型规范用户权限:

- **只读** — Restricts permissions for all types of queries except DDL queries.
- **allow_ddl** — Restricts permissions for DDL queries.

KILL QUERY 可以与任何设置进行。

只读

限制读取数据、写入数据和更改设置查询的权限。

查看查询如何划分为多种类型 [以上](#).

可能的值:

- 0 — All queries are allowed.
- 1 — Only read data queries are allowed.
- 2 — Read data and change settings queries are allowed.

设置后 `readonly = 1`，用户无法更改 `readonly` 和 `allow_ddl` 当前会话中的设置。

使用时 GET 方法中的 **HTTP接口**, `readonly = 1` 自动设置。要修改数据，请使用 POST 方法。

设置 `readonly = 1` 禁止用户更改所有设置。有一种方法可以禁止用户从只更改特定设置，有关详细信息，请参阅 [对设置的限制](#).

默认值 : 0

allow_ddl

允许或拒绝 **DDL** 查询。

查看查询如何划分为多种类型 [以上](#).

可能的值:

- 0 — DDL queries are not allowed.
- 1 — DDL queries are allowed.

你不能执行 `SET allow_ddl = 1` 如果 `allow_ddl = 0` 对于当前会话。

默认值 : 1

设置配置文件

设置配置文件是以相同名称分组的设置的集合。

信息

ClickHouse还支持 **SQL驱动的工作流** 用于管理设置配置文件。我们建议使用它。

配置文件可以有任何名称。配置文件可以有任何名称。您可以为不同的用户指定相同的配置文件。您可以在设置配置文件中编写的最重要的事情是 `readonly=1`，这确保只读访问。

设置配置文件可以彼此继承。要使用继承，请指示一个或多个 `profile` 配置文件中列出的其他设置之前的设置。如果在不同的配置文件中定义了一个设置，则使用最新定义。

要应用配置文件中的所有设置，请设置 `profile` 设置。

示例：

安装 `web` 侧写

```
SET profile = 'web'
```

设置配置文件在用户配置文件中声明。这通常是 `users.xml`。

示例：

```
<!-- 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>10000000000</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>
```

该示例指定了两个配置文件：`default` 和 `web`。

该 `default` 配置文件有一个特殊用途：它必须始终存在并在启动服务器时应用。换句话说，`default` 配置文件包含默认设置。

该 `web` 配置文件是一个常规的配置文件，可以使用设置 `SET` 查询或在HTTP查询中使用URL参数。

对设置的限制

在设置的约束可以在定义 `profiles` 一节 `user.xml` 配置文件，并禁止用户更改一些设置与 `SET` 查询。

约束定义如下：

```
<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>
```

如果用户试图违反约束，将引发异常，并且设置不会更改。

支持三种类型的约束：`min`, `max`, `readonly`. 该 `min` 和 `max` 约束指定数值设置的上边界和下边界，并且可以组合使用。该 `readonly constraint` 指定用户根本无法更改相应的设置。

示例：让 `users.xml` 包括行：

```
<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>
```

以下查询都会引发异常：

```
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：该 `default` 配置文件具有特殊的处理：所有定义的约束 `default` 配置文件成为默认约束，因此它们限制所有用户，直到为这些用户显式覆盖它们。

用户设置

该 `users` 一节 `user.xml` 配置文件包含用户设置。

信息

ClickHouse还支持 **SQL驱动的工作流** 用于管理用户。我们建议使用它。

的结构 `users` 科:

```
<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>
```

用户名/密码

密码可以以明文或SHA256（十六进制格式）指定。

- 以明文形式分配密码（不推荐），把它放在一个 `password` 元素。

例如，`<password>qwerty</password>`. 密码可以留空。

- 要使用其SHA256散列分配密码，请将其放置在 `password_sha256_hex` 元素。

例如，

```
<password_sha256_hex>65e84be33532fb784c48129675f9eff3a682b27168c0ea744b2cf58ee02337c5</password_sha256_hex>.
```

如何从shell生成密码的示例：

```
PASSWORD=$(base64 < /dev/urandom | head -c8); echo "$PASSWORD"; echo -n "$PASSWORD" | sha256sum | tr -d '-'
```

结果的第一行是密码。第二行是相应的SHA256哈希。

- 为了与MySQL客户端兼容，密码可以在双SHA1哈希中指定。放进去 `password_double_sha1_hex` 元素。

例如, `<password_double_sha1_hex>08b4a0f1de6ad37da17359e592c8d74788a83eb0</password_double_sha1_hex>`.

如何从shell生成密码的示例:

```
PASSWORD=$(base64 < /dev/urandom | head -c8); echo "$PASSWORD"; echo -n "$PASSWORD" | sha1sum | tr -d '-' | xxd -r -p | sha1sum | tr -d '-'
```

结果的第一行是密码。第二行是相应的双SHA1哈希。

访问管理

此设置启用禁用使用SQL驱动 [访问控制和帐户管理](#) 对于用户。

可能的值:

- 0 — Disabled.
- 1 — Enabled.

默认值: 0。

用户名/网络

用户可以从中连接到ClickHouse服务器的网络列表。

列表中的每个元素都可以具有以下形式之一:

- `<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.

示例: `example01.host.ru`.

要检查访问, 将执行DNS查询, 并将所有返回的IP地址与对等地址进行比较。

- `<host_regexp>` — Regular expression for hostnames.

示例, `^example\d\d-\d\d-\d\.\host\.ru$`

要检查访问, a [DNS PTR查询](#) 对对等体地址执行, 然后应用指定的正则表达式。然后, 对PTR查询的结果执行另一个DNS查询, 并将所有接收到的地址与对等地址进行比较。我们强烈建议正则表达式以\$结尾。

DNS请求的所有结果都将被缓存, 直到服务器重新启动。

例

要从任何网络打开用户的访问权限, 请指定:

```
<ip>::/0</ip>
```

警告

从任何网络开放访问是不安全的, 除非你有一个防火墙正确配置或服务器没有直接连接到互联网。

若要仅从本地主机打开访问权限, 请指定:

```
<ip>::1</ip>
<ip>127.0.0.1</ip>
```

user_name/profile

您可以为用户分配设置配置文件。设置配置文件在单独的部分配置 `users.xml` 文件 有关详细信息，请参阅 [设置配置文件](#).

用户名/配额

配额允许您在一段时间内跟踪或限制资源使用情况。配额在配置 `quotas` 一节 `users.xml` 配置文件。

您可以为用户分配配额。有关配额配置的详细说明，请参阅 [配额](#).

用户名/数据库

在本节中，您可以限制ClickHouse返回的行 `SELECT` 由当前用户进行的查询，从而实现基本的行级安全性。

示例

以下配置强制该用户 `user1` 只能看到的行 `table1` 作为结果 `SELECT` 查询，其中的值 `id` 场是 `1000`。

```
<user1>
  <databases>
    <database_name>
      <table1>
        <filter>id = 1000</filter>
      </table1>
    </database_name>
  </databases>
</user1>
```

该 `filter` 可以是导致任何表达式 `UInt8`-键入值。它通常包含比较和逻辑运算符。从行 `database_name.table1` 其中，不会为此用户返回为 `0` 的筛选结果。过滤是不兼容的 `PREWHERE` 操作和禁用 `WHERE→PREWHERE` 优化。

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 $\text{pow}(1 * 1000, (1 + 299 - 150) / (300 - 150)) = 1000$ 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`.

查询复杂性的限制

对查询复杂性的限制是设置的一部分。

它们被用来从用户界面提供更安全的执行。

几乎所有的限制只适用于选择。对于分布式查询处理，每个服务器上分别应用限制。

Restrictions on the «maximum amount of something» can take the value 0, which means «unrestricted».

大多数限制也有一个 ‘`overflow_mode`’ 设置，这意味着超过限制时该怎么做。

它可以采用以下两个值之一: `throw` 或 `break`. 对聚合的限制(`group_by_overflow_mode`)也具有以下值 `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.

只读

值为0时，可以执行任何查询。

如果值为1，则只能执行读取请求（如`SELECT`和`SHOW`）。禁止写入和更改设置（插入，设置）的请求。

值为2时，可以处理读取查询（选择、显示）和更改设置（设置）。

启用只读模式后，您无法在当前会话中禁用它。

在HTTP接口中使用`GET`方法时，‘`readonly = 1`’ 自动设置。换句话说，对于修改数据的查询，您只能使用`POST`方法。您可以在`POST`正文或URL参数中发送查询本身。

`max_memory_usage`

用于在单个服务器上运行查询的最大RAM量。

在默认配置文件中，最大值为10GB。

该设置不考虑计算机上的可用内存量或内存总量。

该限制适用于单个服务器中的单个查询。

您可以使用 `SHOW PROCESSLIST` 查看每个查询的当前内存消耗。

此外，还会跟踪每个查询的内存消耗峰值并将其写入日志。

不监视某些聚合函数的状态的内存使用情况。

未完全跟踪聚合函数的状态的内存使用情况 `min`, `max`, `any`, `anyLast`, `argMin`, `argMax` 从 String 和 Array 争论。

内存消耗也受到参数的限制 `max_memory_usage_for_user` 和 `max_memory_usage_for_all_queries`.

`max_memory_usage_for_user`

用于在单个服务器上运行用户查询的最大RAM量。

默认值定义在 [Settings.h](#). 默认情况下，数额不受限制 (`max_memory_usage_for_user = 0`).

另请参阅说明 [max_memory_usage](#).

max_memory_usage_for_all_queries

用于在单个服务器上运行所有查询的最大RAM数量。

默认值定义在 [Settings.h](#). 默认情况下，数额不受限制 (`max_memory_usage_for_all_queries = 0`).

另请参阅说明 [max_memory_usage](#).

max_rows_to_read

可以在每个块（而不是每行）上检查以下限制。也就是说，限制可以打破一点。

运行查询时可从表中读取的最大行数。

max_bytes_to_read

运行查询时可以从表中读取的最大字节数（未压缩数据）。

read_overflow_mode

读取的数据量超过其中一个限制时该怎么办: ‘throw’ 或 ‘break’. 默认情况下，扔。

max_rows_to_group_by

从聚合接收的唯一密钥的最大数量。此设置允许您在聚合时限制内存消耗。

group_by_overflow_mode

当聚合的唯一键数超过限制时该怎么办: ‘throw’，‘break’，或 ‘any’. 默认情况下，扔。

使用 ‘any’ 值允许您运行GROUP BY的近似值。这种近似值的质量取决于数据的统计性质。

max_rows_to_sort

排序前的最大行数。这允许您在排序时限制内存消耗。

max_bytes_to_sort

排序前的最大字节数。

sort_overflow_mode

如果排序前收到的行数超过其中一个限制，该怎么办: ‘throw’ 或 ‘break’. 默认情况下，扔。

max_result_rows

限制结果中的行数。还检查子查询，并在运行分布式查询的部分时在远程服务器上。

max_result_bytes

限制结果中的字节数。与之前的设置相同。

result_overflow_mode

如果结果的体积超过其中一个限制，该怎么办: ‘throw’ 或 ‘break’. 默认情况下，扔。

使用 ‘break’ 类似于使用限制。

max_execution_time

最大查询执行时间（以秒为单位）。

此时，不会检查其中一个排序阶段，也不会在合并和最终确定聚合函数时进行检查。

timeout_overflow_mode

如果查询的运行时间长于 ‘max_execution_time’: ‘throw’ 或 ‘break’. 默认情况下，扔。

min_execution_speed

以每秒行为单位的最小执行速度。 检查每个数据块时 ‘timeout_before_checking_execution_speed’ 到期。 如果执行速度较低，则会引发异常。

timeout_before_checking_execution_speed

检查执行速度是不是太慢（不低于 ‘min_execution_speed’），在指定的时间以秒为单位已过期之后。

max_columns_to_read

单个查询中可从表中读取的最大列数。 如果查询需要读取更多列，则会引发异常。

max_temporary_columns

运行查询时必须同时保留在RAM中的最大临时列数，包括常量列。如果有比这更多的临时列，它会引发异常。

max_temporary_non_const_columns

同样的事情 ‘max_temporary_columns’，但不计数常数列。

请注意，常量列在运行查询时经常形成，但它们需要大约零计算资源。

max_subquery_depth

子查询的最大嵌套深度。如果子查询更深，则会引发异常。默认情况下，100。

max_pipeline_depth

最大管道深度。对应于查询处理期间每个数据块经历的转换数。在单个服务器的限制范围内计算。如果管道深度较大，则会引发异常。默认情况下，1000。

max_ast_depth

查询语法树的最大嵌套深度。如果超出，将引发异常。

此时，在解析过程中不会对其进行检查，而是仅在解析查询之后进行检查。也就是说，在分析过程中可以创建一个太深的语法树，但查询将失败。默认情况下，1000。

max_ast_elements

查询语法树中的最大元素数。如果超出，将引发异常。

与前面的设置相同，只有在解析查询后才会检查它。默认情况下，50,000。

max_rows_in_set

从子查询创建的IN子句中数据集的最大行数。

max_bytes_in_set

从子查询创建的IN子句中的集合使用的最大字节数（未压缩数据）。

set_overflow_mode

当数据量超过其中一个限制时该怎么办: ‘throw’ 或 ‘break’. 默认情况下，扔。

max_rows_in_distinct

使用 DISTINCT 时的最大不同行数。

max_bytes_in_distinct

使用 DISTINCT 时哈希表使用的最大字节数。

distinct_overflow_mode

当数据量超过其中一个限制时该怎么办: ‘throw’ 或 ‘break’. 默认情况下，扔。

max_rows_to_transfer

使用 GLOBAL IN 时，可以传递到远程服务器或保存在临时表中的最大行数。

max_bytes_to_transfer

使用 GLOBAL IN 时，可以传递到远程服务器或保存在临时表中的最大字节数（未压缩数据）。

transfer_overflow_mode

当数据量超过其中一个限制时该怎么办: ‘throw’ 或 ‘break’. 默认情况下，扔。

max_rows_in_join

Limits the number of rows in the hash table that is used when joining tables.

This setting applies to `SELECT ... JOIN` operations and the `Join` table engine.

If a query contains multiple joins, ClickHouse checks this setting for every intermediate result.

ClickHouse can proceed with different actions when the limit is reached. Use the `join_overflow_mode` setting to choose the action.

Possible values:

- Positive integer.
- 0 — Unlimited number of rows.

Default value: 0.

max_bytes_in_join

Limits the size in bytes of the hash table used when joining tables.

This setting applies to `SELECT ... JOIN` operations and `Join` table engine.

If the query contains joins, ClickHouse checks this setting for every intermediate result.

ClickHouse can proceed with different actions when the limit is reached. Use `join_overflow_mode` settings to choose the action.

Possible values:

- Positive integer.
- 0 — Memory control is disabled.

Default value: 0.

join_overflow_mode

Defines what action ClickHouse performs when any of the following join limits is reached:

- [max_bytes_in_join](#)
- [max_rows_in_join](#)

Possible values:

- `THROW` — ClickHouse throws an exception and breaks operation.
- `BREAK` — ClickHouse breaks operation and doesn't throw an exception.

Default value: `THROW`.

See Also

- [JOIN clause](#)
- [Join table engine](#)

max_bytes_before_external_group_by

Enables or disables execution of `GROUP BY` clauses in external memory. See [GROUP BY in external memory](#).

Possible values:

- Maximum volume of RAM (in bytes) that can be used by the single `GROUP BY` operation.
- 0 — `GROUP BY` in external memory disabled.

Default value: 0.

设置

分布_产品_模式

改变的行为 [分布式子查询](#).

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.

限制:

- 仅适用于IN和JOIN子查询。
- 仅当FROM部分使用包含多个分片的分布式表时。
- 如果子查询涉及包含多个分片的分布式表。
- 不用于表值 [远程](#) 功能。

可能的值:

- `deny` — Default value. Prohibits using these types of subqueries (returns the “Double-distributed in/JOIN 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/JOIN` query with `GLOBAL IN/GLOBAL JOIN`.
- `allow` — Allows the use of these types of subqueries.

enable_optimize_predicate_expression

打开谓词下推 `SELECT` 查询。

谓词下推可以显着减少分布式查询的网络流量。

可能的值:

- 0 — Disabled.
- 1 — Enabled.

默认值 : 1。

用途

请考虑以下查询:

1. `SELECT count() FROM test_table WHERE date = '2018-10-10'`
2. `SELECT count() FROM (SELECT * FROM test_table) WHERE date = '2018-10-10'`

如果 `enable_optimize_predicate_expression = 1`，则这些查询的执行时间相等，因为 ClickHouse 应用 `WHERE` 对子查询进行处理。

如果 `enable_optimize_predicate_expression = 0`，那么第二个查询的执行时间要长得多，因为 `WHERE` 子句适用于子查询完成后的所有数据。

fallback_to_stale_replicas_for_distributed_queries

如果更新的数据不可用，则强制对过期副本进行查询。看 [复制](#)。

ClickHouse 从表的过时副本中选择最相关的副本。

执行时使用 `SELECT` 从指向复制表的分布式表。

默认情况下，1（已启用）。

force_index_by_date

如果索引不能按日期使用，则禁用查询执行。

适用于 MergeTree 系列中的表。

如果 `force_index_by_date=1`，ClickHouse 检查查询是否具有可用于限制数据范围的 `date` 键条件。如果没有合适的条件，则会引发异常。但是，它不检查条件是否减少了要读取的数据量。例如，条件 `Date != '2000-01-01'` 即使它与表中的所有数据匹配（即运行查询需要完全扫描），也是可以接受的。有关 MergeTree 表中数据范围的详细信息，请参阅 [MergeTree](#)。

force_primary_key

如果无法按主键编制索引，则禁用查询执行。

适用于 MergeTree 系列中的表。

如果 `force_primary_key=1`，ClickHouse 检查查询是否具有可用于限制数据范围的主键条件。如果没有合适的条件，则会引发异常。但是，它不检查条件是否减少了要读取的数据量。有关 MergeTree 表中数据范围的详细信息，请参阅 [MergeTree](#)。

format_schema

当您使用需要架构定义的格式时，此参数非常有用，例如 [普罗托船长](#) 或 [Protobuf](#)。该值取决于格式。

fsync_metadata

启用或禁用 `fsync` 写作时 `.sql` 文件 默认情况下启用。

如果服务器有数百万个不断创建和销毁的小表，那么禁用它是有意义的。

enable_http_compression

在对 HTTP 请求的响应中启用或禁用数据压缩。

欲了解更多信息，请阅读 [HTTP 接口描述](#)。

可能的值：

- 0 — Disabled.
- 1 — Enabled.

默认值：0。

http_zlib_compression_level

在以下情况下，设置对 HTTP 请求的响应中的数据压缩级别 `enable_http_compression=1`。

可能的值：数字从1到9。

默认值：3。

http_native_compression_disable_checksumming_on_decompression

在从客户端解压缩 HTTP POST 数据时启用或禁用校验和验证。仅用于 ClickHouse 原生压缩格式（不用于 `gzip` 或 `deflate`）。

欲了解更多信息，请阅读 [HTTP 接口描述](#)。

可能的值：

- 0 — Disabled.
- 1 — Enabled.

默认值：0。

send_progress_in_http_headers

启用或禁用 `X-ClickHouse-Progress` Http 响应头 `clickhouse-server` 答复。

欲了解更多信息，请阅读 [HTTP 接口描述](#)。

可能的值：

- 0 — Disabled.
- 1 — Enabled.

默认值：0。

max_http_get_redirects

限制HTTP GET重定向跳数的最大数量 URL-发动机表。该设置适用于两种类型的表：由 `CREATE TABLE` 查询和由 `url` 表功能。

可能的值：

- 跳数的任何正整数。
- 0 — No hops allowed.

默认值：0。

input_format_allow_errors_num

设置从文本格式（CSV，TSV等）读取时可接受的错误的最大数量。).

默认值为0。

总是与它配对 `input_format_allow_errors_ratio`.

如果在读取行时发生错误，但错误计数器仍小于 `input_format_allow_errors_num`，ClickHouse忽略该行并移动到下一个。

如果两者 `input_format_allow_errors_num` 和 `input_format_allow_errors_ratio` 超出时，ClickHouse引发异常。

input_format_allow_errors_ratio

设置从文本格式（CSV，TSV等）读取时允许的最大错误百分比。).

错误百分比设置为介于0和1之间的浮点数。

默认值为0。

总是与它配对 `input_format_allow_errors_num`.

如果在读取行时发生错误，但错误计数器仍小于 `input_format_allow_errors_ratio`，ClickHouse忽略该行并移动到下一个。

如果两者 `input_format_allow_errors_num` 和 `input_format_allow_errors_ratio` 超出时，ClickHouse引发异常。

input_format_values_interpret_expressions

如果快速流解析器无法解析数据，则启用或禁用完整SQL解析器。此设置仅用于 值 格式在数据插入。有关语法分析的详细信息，请参阅 [语法](#) 科。

可能的值：

- 0 — Disabled.

在这种情况下，您必须提供格式化的数据。见 [格式](#) 科。

- 1 — Enabled.

在这种情况下，您可以使用SQL表达式作为值，但数据插入速度要慢得多。如果仅插入格式化的数据，则ClickHouse的行为就好像设置值为0。

默认值：1。

使用示例

插入 [日期时间](#) 使用不同的设置键入值。

```
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.

最后一个查询等效于以下内容:

```
SET input_format_values_interpret_expressions = 0;
INSERT INTO datetime_t SELECT now()
```

Ok.

input_format_values_deduce_templates_of_expressions

启用或禁用以下内容中的SQL表达式的模板扣除 **值** 格式。它允许解析和解释表达式 **Values** 如果连续行中的表达式具有相同的结构，速度要快得多。ClickHouse尝试推导表达式的模板，使用此模板解析以下行，并在一批成功解析的行上评估表达式。

可能的值:

- 0 — Disabled.
- 1 — Enabled.

默认值：1。

对于以下查询:

```
INSERT INTO test VALUES (lower('Hello')), (lower('world')), (lower('INSERT')), (upper('Values')), ...
```

- 如果 `input_format_values_interpret_expressions=1` 和 `format_values_deduce_templates_of_expressions=0`，表达式为每行分别解释（对于大量行来说，这非常慢）。
- 如果 `input_format_values_interpret_expressions=0` 和 `format_values_deduce_templates_of_expressions=1`，第一，第二和第三行中的表达式使用**template**解析 `lower(String)` 并一起解释，第四行中的表达式用另一个模板解析 `(upper(String))`。
- 如果 `input_format_values_interpret_expressions=1` 和 `format_values_deduce_templates_of_expressions=1`，与前面的情况相同，但如果不可能推导出模板，也允许回退到单独解释表达式。

input_format_values_accurate_types_of_literals

此设置仅在以下情况下使用 `input_format_values_deduce_templates_of_expressions = 1`。它可能发生，某些列的表达式具有相同的结构，但包含不同类型的数字文字，例如

```
(..., abs(0), ...),          -- UInt64 literal  
(..., abs(3.141592654), ...), -- Float64 literal  
(..., abs(-1), ...),        -- Int64 literal
```

可能的值:

- 0 — Disabled.

In this case, ClickHouse may use a more general type for some literals (e.g., `Float64` 或 `Int64` 而不是 `UInt64` 为 42) , 但它可能会导致溢出和精度问题。

- 1 — Enabled.

在这种情况下, ClickHouse会检查文本的实际类型, 并使用相应类型的表达式模板。在某些情况下, 可能会显着减慢表达式评估 `Values`.

默认值: 1。

input_format_defaults_for_omitted_fields

执行时 `INSERT` 查询时, 将省略的输入列值替换为相应列的默认值。此选项仅适用于 `JSONEachRow`, `CSV` 和 `TabSeparated` 格式。

注

启用此选项后, 扩展表元数据将从服务器发送到客户端。它会消耗服务器上的额外计算资源, 并可能降低性能。

可能的值:

- 0 — Disabled.
- 1 — Enabled.

默认值: 1。

input_format_tsv_empty_as_default

启用后, 将TSV中的空输入字段替换为默认值。对于复杂的默认表达式 `input_format_defaults_for_omitted_fields` 必须启用了。

默认情况下禁用。

input_format_null_as_default

如果输入数据包含 `NULL`, 但相应列的数据类型不 `Nullable(T)` (对于文本输入格式)。

input_format_skip_unknown_fields

启用或禁用跳过额外数据的插入。

写入数据时, 如果输入数据包含目标表中不存在的列, ClickHouse将引发异常。如果启用了跳过, ClickHouse不会插入额外的数据, 也不会引发异常。

支持的格式:

- `JSONEachRow`
- `CSVWithNames`

- [TabSeparatedWithNames](#)

- [TSKV](#)

可能的值:

- 0 — Disabled.

- 1 — Enabled.

默认值 : 0 。

input_format_import_nested_json

启用或禁用具有嵌套对象的JSON数据的插入。

支持的格式:

- [JSONEachRow](#)

可能的值:

- 0 — Disabled.

- 1 — Enabled.

默认值 : 0 。

另请参阅:

- [嵌套结构的使用 与 JSONEachRow 格式。](#)

input_format_with_names_use_header

启用或禁用插入数据时检查列顺序。

为了提高插入性能，如果您确定输入数据的列顺序与目标表中的列顺序相同，建议禁用此检查。

支持的格式:

- [CSVWithNames](#)

- [TabSeparatedWithNames](#)

可能的值:

- 0 — Disabled.

- 1 — Enabled.

默认值 : 1 。

date_time_input_format

允许选择日期和时间的文本表示的解析器。

该设置不适用于 [日期和时间功能](#).

可能的值:

- 'best_effort' — Enables extended parsing.

ClickHouse可以解析基本 YYYY-MM-DD HH:MM:SS 格式和所有 ISO 8601 日期和时间格式。例如, '2018-06-08T01:02:03.000Z'.

- 'basic' — Use basic parser.

ClickHouse只能解析基本的 YYYY-MM-DD HH:MM:SS 格式。例如, '2019-08-20 10:18:56'.

默认值: 'basic'.

另请参阅:

- [日期时间数据类型](#)。
- [用于处理日期和时间的函数](#)。

join_default_strictness

设置默认严格性 [加入子句](#).

可能的值:

- ALL — If the right table has several matching rows, ClickHouse creates a [笛卡尔积](#) 从匹配的行。这是正常的 JOIN 来自标准SQL的行为。
- 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 和 ALL 都是一样的
- ASOF — For joining sequences with an uncertain match.
- Empty string — If ALL 或 ANY 如果未在查询中指定，ClickHouse将引发异常。

默认值: ALL.

join_any_take_last_row

更改联接操作的行为 ANY 严格。

注意

此设置仅适用于 JOIN 操作与 [加入我们](#) 发动机表.

可能的值:

- 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.

默认值 : 0。

另请参阅:

- [JOIN子句](#)
- [联接表引擎](#)
- [join_default_strictness](#)

join_use_nulls

设置类型 **JOIN** 行为 合并表时，可能会出现空单元格。 ClickHouse根据此设置以不同的方式填充它们。

可能的值:

- 0 — The empty cells are filled with the default value of the corresponding field type.
- 1 — **JOIN** 其行为方式与标准SQL中的行为方式相同。 相应字段的类型将转换为 可为空，和空单元格填充 **NULL**.

默认值 : 0 。

max_block_size

在ClickHouse中，数据由块（列部分集）处理。 单个块的内部处理周期足够高效，但每个块都有明显的支出。 该 `max_block_size` 设置是建议从表中加载块的大小（行数）。 块大小不应该太小，以便每个块上的支出仍然明显，但不能太大，以便在第一个块处理完成后快速完成限制查询。 目标是避免在多个线程中提取大量列时占用太多内存，并且至少保留一些缓存局部性。

默认值 : 65,536 。

块的大小 `max_block_size` 并不总是从表中加载。 如果显然需要检索的数据较少，则处理较小的块。

preferred_block_size_bytes

用于相同的目的 `max_block_size`，但它通过使其适应块中的行数来设置推荐的块大小（以字节为单位）。

但是，块大小不能超过 `max_block_size` 行。

默认情况下 : 1,000,000。 它只有在从MergeTree引擎读取时才有效。

merge_tree_min_rows_for_concurrent_read

如果从a的文件中读取的行数 **MergeTree** 表超过 `merge_tree_min_rows_for_concurrent_read` 然后ClickHouse尝试在多个线程上从该文件执行并发读取。

可能的值:

- 任何正整数。

默认值:163840.

merge_tree_min_bytes_for_concurrent_read

如果从一个文件中读取的字节数 **MergeTree**-发动机表超过 `merge_tree_min_bytes_for_concurrent_read`，然后 ClickHouse尝试在多个线程中并发读取此文件。

可能的值:

- 任何正整数。

默认值:251658240.

merge_tree_min_rows_for_seek

如果要在一个文件中读取的两个数据块之间的距离小于 `merge_tree_min_rows_for_seek` 行，然后ClickHouse不查找文件，而是按顺序读取数据。

可能的值:

- 任何正整数。

默认值 : 0 。

merge_tree_min_bytes_for_seek

如果要在一个文件中读取的两个数据块之间的距离小于 `merge_tree_min_bytes_for_seek` 字节数，然后ClickHouse依次读取包含两个块的文件范围，从而避免了额外的寻道。

可能的值:

- 任何正整数。

默认值：0。

`merge_tree_coarse_index_granularity`

搜索数据时，ClickHouse会检查索引文件中的数据标记。如果ClickHouse发现所需的键在某个范围内，它将此范围划分为 `merge_tree_coarse_index_granularity` 子范围和递归地搜索所需的键。

可能的值:

- 任何正偶数整数。

默认值：8。

`merge_tree_max_rows_to_use_cache`

如果克里克豪斯应该阅读更多 `merge_tree_max_rows_to_use_cache` 在一个查询中的行，它不使用未压缩块的缓存。

未压缩块的缓存存储为查询提取的数据。ClickHouse使用此缓存来加快对重复的小查询的响应。此设置可保护缓存免受读取大量数据的查询的破坏。该 `uncompressed_cache_size` 服务器设置定义未压缩块的高速缓存的大小。

可能的值:

- 任何正整数。

Default value: 128×8192 .

`merge_tree_max_bytes_to_use_cache`

如果克里克豪斯应该阅读更多 `merge_tree_max_bytes_to_use_cache` 在一个查询中的字节，它不使用未压缩块的缓存。

未压缩块的缓存存储为查询提取的数据。ClickHouse使用此缓存来加快对重复的小查询的响应。此设置可保护缓存免受读取大量数据的查询的破坏。该 `uncompressed_cache_size` 服务器设置定义未压缩块的高速缓存的大小。

可能的值:

- 任何正整数。

默认值:2013265920.

`min_bytes_to_use_direct_io`

使用直接I/O访问存储磁盘所需的最小数据量。

ClickHouse在从表中读取数据时使用此设置。如果要读取的所有数据的总存储量超过 `min_bytes_to_use_direct_io` 字节，然后ClickHouse读取从存储磁盘的数据 `O_DIRECT` 选项。

可能的值:

- 0 — Direct I/O is disabled.
- 整数。

默认值：0。

`log_queries`

设置查询日志记录。

使用此设置发送到ClickHouse的查询将根据以下内容中的规则记录 [query_log](#) 服务器配置参数。

示例：

```
log_queries=1
```

log_queries_min_type

`query_log` 要记录的最小类型。

可能的值：

- `QUERY_START` (=1)
- `QUERY_FINISH` (=2)
- `EXCEPTION_BEFORE_START` (=3)
- `EXCEPTION_WHILE_PROCESSING` (=4)

默认值：`QUERY_START`.

可以用来限制哪些entiries将去 `query_log`，说你只有在错误中才感兴趣，那么你可以使用 `EXCEPTION_WHILE_PROCESSING`:

```
log_queries_min_type='EXCEPTION_WHILE_PROCESSING'
```

log_query_threads

设置查询线程日志记录。

ClickHouse使用此设置运行的查询线程将根据以下命令中的规则记录 [query_thread_log](#) 服务器配置参数。

示例：

```
log_query_threads=1
```

max_insert_block_size

要插入到表中的块的大小。

此设置仅适用于服务器形成块的情况。

例如，对于通过HTTP接口进行的插入，服务器会分析数据格式并形成指定大小的块。

但是当使用clickhouse-client时，客户端解析数据本身，并且 '`max_insert_block_size`' 服务器上的设置不会影响插入的块的大小。

使用`INSERT SELECT`时，该设置也没有目的，因为数据是使用在`SELECT`之后形成的相同块插入的。

默认值：1,048,576。

默认值略高于 `max_block_size`. 这样做的原因是某些表引擎 (*MergeTree) 在磁盘上为每个插入的块形成一个数据部分，这是一个相当大的实体。同样，*MergeTree 表在插入过程中对数据进行排序，并且足够大的块大小允许在RAM中对更多数据进行排序。

min_insert_block_size_rows

设置块中可以通过以下方式插入到表中的最小行数 `INSERT` 查询。较小尺寸的块被压扁成较大的块。

可能的值：

- 整数。

- 0 — Squashing disabled.

默认值：1048576。

min_insert_block_size_bytes

设置块中的最小字节数，可以通过以下方式插入到表中 `INSERT` 查询。较小尺寸的块被压扁成较大的块。

可能的值：

- 整数。
- 0 — Squashing disabled.

默认值：268435456。

max_replica_delay_for_distributed_queries

禁用分布式查询的滞后副本。看 [复制](#)。

以秒为单位设置时间。如果副本滞后超过设定值，则不使用此副本。

默认值：300。

执行时使用 `SELECT` 从指向复制表的分布式表。

max_threads

查询处理线程的最大数量，不包括用于从远程服务器检索数据的线程（请参阅 ‘`max_distributed_connections`’ 参数）。

此参数适用于并行执行查询处理管道的相同阶段的线程。

例如，当从表中读取时，如果可以使用函数来评估表达式，请使用 `WHERE` 进行过滤，并且至少使用并行方式对 `GROUP BY` 进行预聚合 ‘`max_threads`’ 线程数，然后 ‘`max_threads`’ 被使用。

默认值：物理CPU内核数。

如果一次在服务器上运行的 `SELECT` 查询通常少于一个，请将此参数设置为略小于实际处理器内核数的值。

对于由于限制而快速完成的查询，可以设置较低的 ‘`max_threads`’。例如，如果必要数量的条目位于每个块中，并且 `max_threads=8`，则会检索 8 个块，尽管只读取一个块就足够了。

越小 `max_threads` 值，较少的内存被消耗。

max_insert_threads

要执行的最大线程数 `INSERT SELECT` 查询。

可能的值：

- 0 (or 1) — `INSERT SELECT` 没有并行执行。
- 整数。大于 1。

默认值：0。

平行 `INSERT SELECT` 只有在 `SELECT` 部分并行执行，请参阅 [max_threads](#) 设置。

更高的值将导致更高的内存使用率。

max_compress_block_size

在压缩写入表之前，未压缩数据块的最大大小。默认情况下，1,048,576 (1MiB)。如果大小减小，则压缩率显着降低，压缩和解压缩速度由于高速缓存局部性而略微增加，并且内存消耗减少。通常没有任何理由更改此设置。

不要将用于压缩的块（由字节组成的内存块）与用于查询处理的块（表中的一组行）混淆。

min_compress_block_size

为 MergeTree"表。为了减少处理查询时的延迟，在写入下一个标记时，如果块的大小至少为‘min_compress_block_size’。默认情况下，65,536。

块的实际大小，如果未压缩的数据小于‘max_compress_block_size’，是不小于该值且不小于一个标记的数据量。

让我们来看看一个例子。假设‘index_granularity’在表创建期间设置为8192。

我们正在编写一个UInt32类型的列（每个值4个字节）。当写入8192行时，总数将是32KB的数据。由于min_compress_block_size=65,536，将为每两个标记形成一个压缩块。

我们正在编写一个字符串类型的URL列（每个值的平均大小60字节）。当写入8192行时，平均数据将略少于500KB。由于这超过65,536，将为每个标记形成一个压缩块。在这种情况下，当从单个标记范围内的磁盘读取数据时，额外的数据不会被解压缩。

通常没有任何理由更改此设置。

max_query_size

查询的最大部分，可以被带到RAM用于使用SQL解析器进行解析。

插入查询还包含由单独的流解析器（消耗O(1)RAM）处理的插入数据，这些数据不包含在此限制中。

默认值：256KiB。

interactive_delay

以微秒为单位的间隔，用于检查请求执行是否已被取消并发送进度。

默认值：100,000（检查取消并每秒发送十次进度）。

connect_timeout,receive_timeout,send_timeout

用于与客户端通信的套接字上的超时以秒为单位。

默认值：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.

默认值：0

poll_interval

锁定在指定秒数的等待循环。

默认值：10。

max_distributed_connections

与远程服务器同时连接的最大数量，用于分布式处理对单个分布式表的单个查询。我们建议设置不小于群集中服务器数量的值。

默认值：1024。

以下参数仅在创建分布式表（以及启动服务器时）时使用，因此没有理由在运行时更改它们。

distributed_connections_pool_size

与远程服务器同时连接的最大数量，用于分布式处理对单个分布式表的所有查询。 我们建议设置不小于群集中服务器数量的值。

默认值：1024。

connect_timeout_with_failover_ms

以毫秒为单位连接到分布式表引擎的远程服务器的超时，如果 ‘shard’ 和 ‘replica’ 部分用于群集定义。
如果不成功，将尝试多次连接到各种副本。

默认值：50。

connections_with_failover_max_tries

分布式表引擎的每个副本的最大连接尝试次数。

默认值：3。

极端

是否计算极值（查询结果列中的最小值和最大值）。 接受0或1。 默认情况下，0（禁用）。
有关详细信息，请参阅部分 “Extreme values”.

use_uncompressed_cache

是否使用未压缩块的缓存。 接受0或1。 默认情况下，0（禁用）。

使用未压缩缓存（仅适用于MergeTree系列中的表）可以在处理大量短查询时显着减少延迟并提高吞吐量。 为频繁发送短请求的用户启用此设置。 还要注意 **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.

对于至少读取大量数据（一百万行或更多行）的查询，将自动禁用未压缩缓存，以节省真正小型查询的空间。 这意味着你可以保持 ‘use_uncompressed_cache’ 设置始终设置为1。

replace_running_query

当使用HTTP接口时， ‘query_id’ 参数可以传递。 这是用作查询标识符的任何字符串。

如果来自同一用户的查询具有相同的 ‘query_id’ 已经存在在这个时候，行为取决于 ‘replace_running_query’ 参数。

0 (default) – Throw an exception (don't allow the query to run if a query with the same ‘query_id’ 已经运行) 。

1 – Cancel the old query and start running the new one.

YandexMetrica使用此参数设置为1来实现分段条件的建议。 输入下一个字符后，如果旧的查询还没有完成，应该取消。

stream_flush_interval_ms

适用于在超时的情况下或线程生成流式传输的表 **max_insert_block_size** 行。

默认值为7500。

值越小，数据被刷新到表中的频率就越高。 将该值设置得太低会导致性能较差。

load_balancing

指定用于分布式查询处理的副本选择算法。

ClickHouse 支持以下选择副本的算法：

- 随机（默认情况下）
- 最近的主机名
- 按顺序
- 第一次或随机

随机（默认情况下）

```
load_balancing = random
```

对每个副本计算错误数。查询发送到错误最少的副本，如果存在其中几个错误，则发送给其中任何一个。

缺点：不考虑服务器邻近度；如果副本具有不同的数据，则也会获得不同的数据。

最近的主机名

```
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).

例如，例如 01-01-1 和 example01-01-2.yandex.ru 在一个位置是不同的，而 example01-01-1 和 example01-02-2 在两个地方不同。

这种方法可能看起来很原始，但它不需要有关网络拓扑的外部数据，也不比较 IP 地址，这对于我们的 IPv6 地址来说会很复杂。

因此，如果存在等效副本，则首选按名称最接近的副本。

我们还可以假设，当向同一台服务器发送查询时，在没有失败的情况下，分布式查询也将转到同一台服务器。因此，即使在副本上放置了不同的数据，查询也会返回大多相同的结果。

按顺序

```
load_balancing = in_order
```

具有相同错误数的副本的访问顺序与配置中指定的顺序相同。

当您确切知道哪个副本是可取的时，此方法是适当的。

第一次或随机

```
load_balancing = first_or_random
```

此算法选择集合中的第一个副本，如果第一个副本不可用，则选择随机副本。它在跨复制拓扑设置中有效，但在其他配置中无用。

该 `first_or_random` 算法解决的问题 `in_order` 算法。与 `in_order`，如果一个副本出现故障，下一个副本将获得双重负载，而其余副本将处理通常的流量。使用时 `first_or_random` 算法中，负载均匀分布在仍然可用的副本之间。

prefer_localhost_replica

在处理分布式查询时，最好使用localhost副本启用/禁用该副本。

可能的值：

- 1 — ClickHouse always sends a query to the localhost replica if it exists.
- 0 — ClickHouse uses the balancing strategy specified by the [load_balancing](#) 设置。

默认值：1。

警告

如果使用此设置，请禁用此设置 [max_parallel_replicas](#).

totals_mode

如何计算总计时有存在，以及当max_rows_to_group_by和group_by_overflow_mode= 'any' 都在场。
请参阅部分“WITH TOTALS modifier”。

totals_auto_threshold

阈值 totals_mode = 'auto'.

请参阅部分“WITH TOTALS modifier”。

max_parallel_replicas

执行查询时每个分片的最大副本数。

为了保持一致性（以获取相同数据拆分的不同部分），此选项仅在设置了采样键时有效。
副本滞后不受控制。

compile_expressions

启用或禁用在运行时使用 LLVM 将常用的简单函数和运算符编译为本机代码。

可能的值：

- 0 — 禁用。
- 1 — 启用。

默认值：1。

min_count_to_compile_expression

在编译之前执行相同表达式的最小计数。

默认值：3。

output_format_json_quote_64bit_integers

如果该值为true，则在使用JSON*Int64和UInt64格式时，整数将显示在引号中（为了与大多数JavaScript实现兼容）；否则，整数将不带引号输出。

format_csv_delimiter

将字符解释为CSV数据中的分隔符。默认情况下，分隔符为`,`

input_format_csv_unquoted_null_literal_as_null

对于CSV输入格式，启用或禁用未引用的解析 `NULL` 作为文字（同义词 `\N`）。

`output_format_csv_crlf_end_of_line`

在CSV中使用DOS/Windows样式的行分隔符(CRLF)而不是Unix样式(LF)。

`output_format_tsv_crlf_end_of_line`

在TSV中使用DOC/Windows样式的行分隔符 (CRLF) 而不是Unix样式 (LF)。

`insert_quorum`

启用仲裁写入。

- 如果 `insert_quorum < 2`，仲裁写入被禁用。
- 如果 `insert_quorum >= 2`，仲裁写入已启用。

默认值：0。

仲裁写入

`INSERT` 只有当ClickHouse设法正确地将数据写入成功 `insert_quorum` 在复制品的 `insert_quorum_timeout`。如果由于任何原因，成功写入的副本数量没有达到 `insert_quorum`，写入被认为失败，ClickHouse将从已经写入数据的所有副本中删除插入的块。

仲裁中的所有副本都是一致的，即它们包含来自所有以前的数据 `INSERT` 查询。该 `INSERT` 序列线性化。

当读取从写入的数据 `insert_quorum`，您可以使用 `select_sequential_consistency` 选项。

ClickHouse生成异常

- 如果查询时可用副本的数量小于 `insert_quorum`。
- 在尝试写入数据时，以前的块尚未被插入 `insert_quorum` 的复制品。如果用户尝试执行 `INSERT` 前一个与 `insert_quorum` 完成。

另请参阅：

- `insert_quorum_timeout`
- `select_sequential_consistency`

`insert_quorum_timeout`

写入仲裁超时以秒为单位。如果超时已经过去，并且还没有发生写入，ClickHouse将生成异常，客户端必须重复查询以将相同的块写入相同的副本或任何其他副本。

默认值：60秒。

另请参阅：

- `insert_quorum`
- `select_sequential_consistency`

`select_sequential_consistency`

启用或禁用顺序一致性 `SELECT` 查询：

可能的值：

- 0 — Disabled.

- 1 — Enabled.

默认值：0。

用途

当启用顺序一致性时，ClickHouse允许客户端执行 `SELECT` 仅查询那些包含来自所有先前数据的副本 `INSERT` 查询执行 `insert_quorum`. 如果客户端引用了部分副本，ClickHouse将生成异常。 `SELECT`查询将不包括尚未写入副本仲裁的数据。

另请参阅：

- [insert_quorum](#)
- [insert_quorum_timeout](#)

insert_deduplicate

启用或禁用块重复数据删除 `INSERT` (对于复制的*表)。

可能的值：

- 0 — Disabled.
- 1 — Enabled.

默认值：1。

默认情况下，块插入到复制的表 `INSERT` 语句重复数据删除 (见 [数据复制](#)).

deduplicate_blocks_in_dependent_materialized_views

启用或禁用从已复制*表接收数据的实例化视图的重复数据删除检查。

可能的值：

0 — Disabled.
1 — Enabled.

默认值：0。

用途

默认情况下，重复数据删除不对实例化视图执行，而是在源表的上游执行。

如果由于源表中的重复数据删除而跳过了插入的块，则不会插入附加的实例化视图。这种行为的存在是为了允许将高度聚合的数据插入到实例化视图中，对于在实例化视图聚合之后插入的块相同，但是从源表中的不同插入派生的情况。

与此同时，这种行为“breaks”`INSERT` 署等性。如果一个 `INSERT` 进入主表是成功的，`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` 允许改变这种行为。重试时，实例化视图将收到重复插入，并自行执行重复数据删除检查，忽略源表的检查结果，并将插入由于第一次失败而丢失的行。

max_network_bytes

限制在执行查询时通过网络接收或传输的数据量 (以字节为单位)。此设置适用于每个单独的查询。

可能的值：

- 整数。

- 0 — Data volume control is disabled.

默认值：0。

max_network_bandwidth

限制通过网络进行数据交换的速度，以每秒字节为单位。此设置适用于每个查询。

可能的值：

- 整数。
- 0 — Bandwidth control is disabled.

默认值：0。

max_network_bandwidth_for_user

限制通过网络进行数据交换的速度，以每秒字节为单位。此设置适用于单个用户执行的所有并发运行的查询。

可能的值：

- 整数。
- 0 — Control of the data speed is disabled.

默认值：0。

max_network_bandwidth_for_all_users

限制通过网络交换数据的速度，以每秒字节为单位。此设置适用于服务器上同时运行的所有查询。

可能的值：

- 整数。
- 0 — Control of the data speed is disabled.

默认值：0。

count_distinct_implementation

指定其中的 `uniq*` 函数应用于执行 `COUNT(DISTINCT ...)` 建筑。

可能的值：

- `uniq`
- `uniqCombined`
- `uniqCombined64`
- `uniqHLL12`
- `uniqExact`

默认值：`uniqExact`。

skip_unavailable_shards

启用或禁用静默跳过不可用分片。

如果分片的所有副本都不可用，则视为不可用。副本在以下情况下不可用：

- ClickHouse出于任何原因无法连接到副本。

连接到副本时，ClickHouse会执行多次尝试。如果所有这些尝试都失败，则认为副本不可用。

- 副本无法通过DNS解析。

如果无法通过DNS解析副本的主机名，则可能指示以下情况：

- 副本的主机没有DNS记录。它可以发生在具有动态DNS的系统中，例如，**Kubernetes**，其中节点在停机期间可能无法解决问题，这不是错误。
- 配置错误。ClickHouse配置文件包含错误的主机名。

可能的值：

- 1 — skipping enabled.

如果分片不可用，ClickHouse将基于部分数据返回结果，并且不报告节点可用性问题。

- 0 — skipping disabled.

如果分片不可用，ClickHouse将引发异常。

默认值：0。

optimize_skip_unused_shards

对于在PREWHERE/WHERE中具有分片键条件的SELECT查询，启用或禁用跳过未使用的分片（假定数据是通过分片键分发的，否则不执行任何操作）。

默认值：0

force_optimize_skip_unused_shards

在以下情况下启用或禁用查询执行 `optimize_skip_unused_shards` 无法启用和跳过未使用的分片。如果跳过是不可能的，并且设置为启用异常将被抛出。

可能的值：

- 0-禁用（不抛出）
- 1-仅当表具有分片键时禁用查询执行
- 2-无论为表定义了分片键，都禁用查询执行

默认值：0

optimize_throw_if_noop

启用或禁用抛出异常，如果 **OPTIMIZE** 查询未执行合并。

默认情况下，**OPTIMIZE** 即使它没有做任何事情，也会成功返回。此设置允许您区分这些情况并在异常消息中获取原因。

可能的值：

- 1 — Throwing an exception is enabled.
- 0 — Throwing an exception is disabled.

默认值：0。

distributed_replica_error_half_life

- 类型：秒
- 默认值：60秒

控制清零分布式表中的错误的速度。如果某个副本在一段时间内不可用，累计出现5个错误，并且 `distributed_replica_error_half_life` 设置为1秒，则该副本在上一个错误发生3秒后视为正常。

另请参阅：

- 表引擎分布式
- `distributed_replica_error_cap`

`distributed_replica_error_cap`

- 类型：无符号int
- 默认值：1000

每个副本的错误计数上限为此值，从而防止单个副本累积太多错误。

另请参阅：

- 表引擎分布式
- `distributed_replica_error_half_life`

`distributed_directory_monitor_sleep_time_ms`

对于基本间隔 `分布` 表引擎发送数据。在发生错误时，实际间隔呈指数级增长。

可能的值：

- 毫秒的正整数。

默认值：100毫秒。

`distributed_directory_monitor_max_sleep_time_ms`

的最大间隔 `分布` 表引擎发送数据。限制在设置的区间的指数增长 `distributed_directory_monitor_sleep_time_ms` 设置。

可能的值：

- 毫秒的正整数。

默认值：30000毫秒（30秒）。

`distributed_directory_monitor_batch_inserts`

启用/禁用批量发送插入的数据。

当批量发送被启用时，`分布` 表引擎尝试在一个操作中发送插入数据的多个文件，而不是单独发送它们。批量发送通过更好地利用服务器和网络资源来提高集群性能。

可能的值：

- 1 — Enabled.
- 0 — Disabled.

默认值：0。

os_thread_priority

设置优先级 ([不错](#)) 对于执行查询的线程。当选择要在每个可用CPU内核上运行的下一个线程时，操作系统调度程序会考虑此优先级。

警告

要使用此设置，您需要设置 `CAP_SYS_NICE` 能力。该 `clickhouse-server` 软件包在安装过程中设置它。某些虚拟环境不允许您设置 `CAP_SYS_NICE` 能力。在这种情况下，`clickhouse-server` 在开始时显示关于它的消息。

可能的值:

- 您可以在范围内设置值 [-20, 19].

值越低意味着优先级越高。低螺纹 `nice` 与具有高值的线程相比，优先级值的执行频率更高。高值对于长时间运行的非交互式查询更为可取，因为这使得它们可以在到达时快速放弃资源，转而使用短交互式查询。

默认值：0。

query_profiler_real_time_period_ns

设置周期的实时时钟定时器 [查询探查器](#). 真正的时钟计时器计数挂钟时间。

可能的值:

- 正整数，以纳秒为单位。

推荐值:

- 10000000 (100 times a second) nanoseconds and less for single queries.
- 1000000000 (once a second) for cluster-wide profiling.

- 0用于关闭计时器。

类型: [UInt64](#).

默认值：10000000000纳秒（每秒一次）。

另请参阅:

- 系统表 [trace_log](#)

query_profiler_cpu_time_period_ns

设置周期的CPU时钟定时器 [查询探查器](#). 此计时器仅计算CPU时间。

可能的值:

- 纳秒的正整数。

推荐值:

- 10000000 (100 times a second) nanoseconds and more for single queries.
- 1000000000 (once a second) for cluster-wide profiling.

- 0用于关闭计时器。

类型: [UInt64](#).

默认值：10000000000纳秒。

另请参阅：

- 系统表 [trace_log](#)

allow_introspection_functions

启用禁用 反省函数 用于查询分析。

可能的值：

- 1 — Introspection functions enabled.
- 0 — Introspection functions disabled.

默认值：0。

另请参阅

- [采样查询探查器](#)
- 系统表 [trace_log](#)

input_format_parallel_parsing

- 类型：布尔
- 默认值：True

启用数据格式的保序并行分析。仅支持TSV，TCSV，CSV和JSONEachRow格式。

min_chunk_bytes_for_parallel_parsing

- 类型：无符号int
- 默认值：1MiB

以字节为单位的最小块大小，每个线程将并行解析。

output_format_avro_codec

设置用于输出Avro文件的压缩编解码器。

类型：字符串

可能的值：

- null — No compression
- deflate — Compress with Deflate (zlib)
- snappy — Compress with [活泼的](#)

默认值：snappy（如果可用）或 deflate。

output_format_avro_sync_interval

设置输出Avro文件的同步标记之间的最小数据大小（以字节为单位）。

类型：无符号int

可能的值：32（32字节）-1073741824（1GiB）

默认值：32768（32KiB）

format_avro_schema_registry_url

设置要与之一起使用的汇合架构注册表URL **AvroConfluent** 格式

类型：网址

默认值：空

background_pool_size

设置在表引擎中执行后台操作的线程数（例如，合并 **MergeTree** 引擎 表）。此设置在ClickHouse服务器启动时应用，不能在用户会话中更改。通过调整此设置，您可以管理CPU和磁盘负载。较小的池大小使用较少的CPU和磁盘资源，但后台进程推进速度较慢，最终可能会影响查询性能。

可能的值：

- 任何正整数。

默认值：16。

[原始文章](#)

transform_null_in

为**IN** 运算符启用**NULL** 值的相等性。

默认情况下，无法比较 **NULL** 值，因为 **NULL** 表示未定义的值。因此，比较 `expr = NULL` 必须始终返回 `false`。在此设置下，**NULL = NULL** 为**IN**运算符返回 `true`。

可能的值：

- 0 — 比较 **IN** 运算符中 **NULL** 值将返回 `false`。
- 1 — 比较 **IN** 运算符中 **NULL** 值将返回 `true`。

默认值：0。

例

考虑**null_in**表：

idx	i
1	1
2	NULL
3	3

查询：

```
SELECT idx, i FROM null_in WHERE i IN (1, NULL) SETTINGS transform_null_in = 0;
```

结果：

idx	i
1	1

查询：

```
SELECT idx, i FROM null_in WHERE i IN (1, NULL) SETTINGS transform_null_in = 1;
```

结果：

idx	i
1	1
2	NULL

另请参阅

- [IN 运算符中的 NULL 处理](#)

max_final_threads

设置使用 [FINAL](#) 限定符的 `SELECT` 查询，在数据读取阶段的最大并发线程数。

可能的值：

- 正整数。
- 0 or 1 — 禁用。此时 `SELECT` 查询单线程执行。

默认值: 16。

ClickHouse 云服务提供商

ClickHouse Cloud Service

Info

Detailed public description for ClickHouse cloud services is not ready yet, please [contact us](#) to learn more.

ClickHouse 商业支持服务提供商

Info

Detailed public description for ClickHouse support services is not ready yet, please [contact us](#) to learn more.

ClickHouse 商业服务

服务类别：

- [云](#)
- [支持](#)

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?

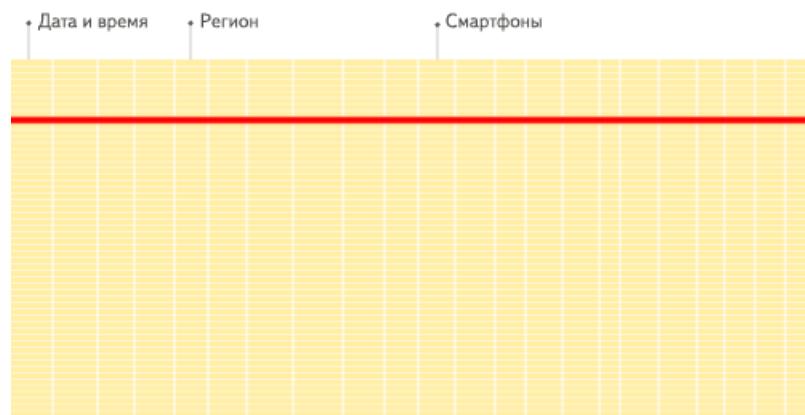
A columnar database stores data of each column independently. This allows to read data from disks only for those columns that are used in any given query. The cost is that operations that affect whole rows become proportionally more expensive. The synonym for a columnar database is a column-oriented database management system. ClickHouse is a typical example of such a system.

Key columnar database advantages are:

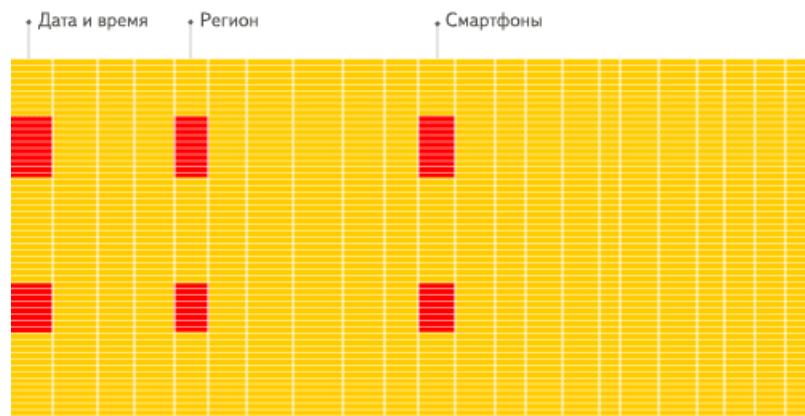
- Queries that use only a few columns out of many.
- Aggregating queries against large volumes of data.
- Column-wise data compression.

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
```

常见问题

为什么不使用MapReduce之类的产品呢？

我们可以将MapReduce这类的系统称为分布式计算系统，其reduce操作基于分布式排序。其中最常见的开源解决方案是 [Apache Hadoop](#)。Yandex使用他们的内部解决方案YT。

这些系统不适合在线查询，因为它们的延迟高。换句话说，它们不能用作Web接口的后端服务。这些系统对于实时数据更新是没有用的。如果操作的结果和所有中间结果（如果有的话）位于单个服务器的内存中，则分布式排序不是执行reduce操作的最佳方式，但这通常是在线查询的情况。在这种情况下，哈希表是执行reduce操作的最佳方式。优化map-reduce任务的常用方法是使用内存中的哈希表进行预聚合（部分reduce），用户手动执行此优化操作。分布式排序是运行简单map-reduce任务时性能降低的主要原因之一。

大多数MapReduce系统允许您在集群上执行任意代码。但是，声明性查询语言更适合OLAP，以便快速运行实验。例如，Hadoop包含Hive和Pig，Cloudera Impala或Shark（过时）for Spark，以及Spark SQL、Presto和Apache Drill。与专业系统相比，运行此类任务时的性能非常不理想，所以将这些系统用作Web接口的后端服务是不现实的，因为延迟相对较高。

如果我在通过ODBC使用Oracle时遇到编码问题，该怎么办？

如果您通过ODBC驱动程序使用Oracle作为外部字典的源，则需要为 `NLS_LANG` 在变量 `/etc/default/clickhouse`. 欲了解更多详情，请参阅 [Oracle NLS_常见问题](#).

示例

```
NLS_LANG=CHINESE_CHINA.ZHS16GBK
```

术语翻译约定

本文档用来维护从英文翻译成中文的术语集。

保持英文，不译

Parquet

英文 <-> 中文

Integer 整数

floating-point 浮点数

Fitting 拟合

Decimal 定点数

Tuple 元组

function 函数

array 数组/阵列

hash 哈希/散列

Parameters 参数

Arguments 参数

1. 对于array的翻译，保持初始翻译 数组/阵列 不变。

2. 对于倒装句。翻译时非直译，会调整语序。

比如，groupArrayInsertAt 翻译中

- `x` — [Expression] resulting in one of the [supported data types].

`x` — 生成所[支持的数据类型](#faq-数据)的[表达式]。

3. See also 参见

ClickHouse release v21.10, 2021-10-08

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 (Kseniia 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 (Kseniia 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 (Kseniia 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 (Ksenia 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.

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). (Kseniia 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). (Kseniia 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_regex_length and max_hyperscan_regex_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 new index data skipping minmax index format for proper Nullable support. #27250 ([Azat Khuzhin](#)).
- 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).
- Now `KeyConditions` can correctly skip nullable keys, including `isNull` and `isNotNull`.
<https://github.com/ClickHouse/ClickHouse/pull/12433>. #12455 (Amos Bird).

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 `NONE` (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_read_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 it 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 ([I1tsolaiki](#), [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 Levakov).
- 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 (Kseniia 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 (Kseniia 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 (Kseniia 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 ([Ksenia 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 ([Ksenia 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 endpoint. 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](#)) ([Kseniia 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 (Ksenia 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](#) ([Kseniia 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 (sundyl).

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 formated 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 NULL values in postgresql compatibility protocol. Closes [#22622](#). [#24857](#) ([Kseniia 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_read_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 (Ksenia 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 ([Kseniia 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 ([Kseniia 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](#) ([Kseniia 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 (Kseniia 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 (Kseniia 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. [#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](#) ([Ksenia 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 (changvvb).
- 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 (Izma) 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 if 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 PREWHERE 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 `Ilvm-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 `111111111.222`). 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 (mxzlxxy).
- 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](#) ([Ksenia 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 ldap 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 (Kseniia 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 whether `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 (sundylili).
- 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 (sundyl).
- 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`. #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 Authenticaiton 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) synax 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 xxx;`. [#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_coldown` 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 (Ksenia 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 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).
- 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).
- It 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 it's 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 ($> \text{min_chunk_bytes_for_parallel_parsing}$), 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` 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 (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).
- 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 desireable. It will not skip zero values in the result. Fixes #16087. #16631 (Ildus Kurbangaliev).
- Better update of ZooKeeper configuration in runtime. #16630 (sundyli).
- 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 TablesToDropQueueSize 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_regex_escaping_rule` 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 enumerate 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 (hcz).
- 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, FreeBSD) to clang-11. [#15622 \(Ilya Yatsishin\)](#).
- Now all test images use `llvm-symbolizer-11`. [#15069 \(alesapin\)](#).
- Allow to build with `llvm-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 or 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 argument` error 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 \(sundyli\)](#).
- 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. [#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 (hc).
- 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 (sundyli).
- 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 `dhpamparam.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. It's 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 (hcz).
- 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. #12756 (sundyli).
- 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 (hc).
- 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 (Ksenia 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. It's 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 ZooKeeper 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 (hcz).
- 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 (Darío).
- 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 (Matwey 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 `Ild` 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 (Matwey 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 `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 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 (hc).
- 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 downloading from other replicas. #10379 (alesapin)
- Add function `JSONExtractKeysAndValuesRaw` which extracts raw data from JSON objects #10378 (hc2)
- 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 setting is also useful with `allow_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 `querySYSTEM 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 (hcz)
- 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/relica_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 (hcz)
- 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](#))

- Now 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 flacky 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 columnfor 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 settings is also useful with allow_read_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 Yatishin](#))
- `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 `FixedSize` 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 `LLVME-executableFunction` 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 \(hcz\)](#).
- 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 `lz4` 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 INT_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 crt1.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 receives 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. 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. (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^I 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 (hcza)
- 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 `NONE` 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 `NONE` 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 NULL.

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 `Fedor 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 `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 `multilf`, `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 `NUL` 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.
- 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.

Roadmap

2021年Roadmap已公布供公开讨论查看[这里](#).

修复于ClickHouse Release 19.14.3.3, 2019-09-10

CVE-2019-15024

对ZooKeeper具有写访问权限并且可以运行ClickHouse所在网络上可用的自定义服务器的攻击者可以创建一个自定义的恶意服务器，该服务器将充当ClickHouse副本并在ZooKeeper中注册。当另一个副本将从恶意副本获取数据部分时，它可以强制clickhouse服务器写入文件系统上的任意路径。

作者：Yandex信息安全部队Eldar Zaitov

CVE-2019-16535

解压算法中的OOB-read、OOB-write和整数下溢可以通过本机协议实现RCE或DoS。

作者: Yandex信息安全部队Eldar Zaitov

CVE-2019-16536

恶意的经过身份验证的客户端可能会触发导致DoS的堆栈溢出。

作者: Yandex信息安全部队Eldar Zaitov

修复于ClickHouse Release 19.13.6.1, 2019-09-20

CVE-2019-18657

表函数url存在允许攻击者在请求中插入任意HTTP标头的漏洞。

作者: Nikita Tikhomirov

修复于ClickHouse Release 18.12.13, 2018-09-10

CVE-2018-14672

加载CatBoost模型的函数允许路径遍历和通过错误消息读取任意文件。

作者: Yandex信息安全部队Andrey Krasichkov

修复于Release 18.10.3, 2018-08-13

CVE-2018-14671

unixODBC允许从文件系统加载任意共享对象，从而导致远程代码执行漏洞。

作者: Yandex信息安全部队Andrey Krasichkov和Evgeny Sidorov

修复于ClickHouse Release 1.1.54388, 2018-06-28

CVE-2018-14668

remote表函数允许在user, password和default_database字段中使用任意符号，从而导致跨协议请求伪造攻击。

作者: Yandex信息安全部队Andrey Krasichkov

修复于ClickHouse Release 1.1.54390, 2018-07-06

CVE-2018-14669

ClickHouse MySQL客户端启用了LOAD DATA LOCAL INFILE功能，允许恶意MySQL数据库从连接的ClickHouse服务器读取任意文件。

作者: Yandex信息安全部队Andrey Krasichkov和Evgeny Sidorov

修复于ClickHouse Release 1.1.54131, 2017-01-10

CVE-2018-14670

deb包中的错误配置可能导致未经授权使用数据库。

作者: 英国国家网络安全中心 (NCSC)

ClickHouse变更及改动？

对于已经发布的版本，有一个[roadmap](#)和一个详细的[changelog](#)。

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** split 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.

浏览 ClickHouse 源代码

您可以使用 [Woboq](#) 在线代码浏览器 [点击这里](#)。它提供了代码导航和语义突出显示、搜索和索引。代码快照每天更新。

此外，您还可以像往常一样浏览源代码 [GitHub](#)

如果你希望了解哪种IDE较好，我们推荐使用CLion，QT Creator，VS Code和KDevelop（有注意事项）。您可以使用任何您喜欢的IDE。Vim和Emacs也可以。

如何在Linux上为AARCH64 (ARM64) 架构构建ClickHouse

这是当你有Linux机器，并希望使用它来构建的情况下 `clickhouse` 二进制文件将运行在另一个Linux机器上与AARCH64CPU架构。这适用于在Linux服务器上运行的持续集成检查。

Aarch64的交叉构建基于 [构建说明](#) 先跟着他们

安装Clang-8

按照以下说明操作<https://apt.llvm.org/>为您的Ubuntu或Debian设置。

例如，在Ubuntu Bionic中，您可以使用以下命令：

```
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
```

安装交叉编译工具集

```
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 xJf gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu.tar.xz -C build-aarch64/cmake/toolchain/linux-aarch64 --strip-components=1
```

建立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
```

生成的二进制文件将仅在具有AARCH64CPU体系结构的Linux上运行。

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 --multiquery --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 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 run 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 was written and all rejected inserts was 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 `uncrustify` tool to reformat your code. Configuration is in `uncrustify.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 single 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.

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 \
-DENABLE_CLICKHOUSE_SERVER=ON \
-DENABLE_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
ENABLE_CLICKHOUSE_ALL	ON	Enable all ClickHouse modes by default	The clickhouse tool multiple executors may be built a know what mode SERVER and CLIENT
ENABLE_CLICKHOUSE_BENCHMARK	ENABLE_CLICKHOUSE_ALL	Queries benchmarking mode	https://clickhouse.com/docs/en/interfaces/benchmark/
ENABLE_CLICKHOUSE_CLIENT	ENABLE_CLICKHOUSE_ALL	Client mode (interactive tui/shell that connects to the server)	
ENABLE_CLICKHOUSE_COMPRESSOR	ENABLE_CLICKHOUSE_ALL	Data compressor and decompressor	https://clickhouse.com/docs/en/interfaces/compressor/
ENABLE_CLICKHOUSE_COPIER	ENABLE_CLICKHOUSE_ALL	Inter-cluster data copying mode	https://clickhouse.com/docs/en/interfaces/copier/
ENABLE_CLICKHOUSE_EXTRACT_FROM_CONFIG	ENABLE_CLICKHOUSE_ALL	Configs processor (extract values etc.)	
ENABLE_CLICKHOUSE_FORMAT	ENABLE_CLICKHOUSE_ALL	Queries pretty-printer and formatter with syntax highlighting	
ENABLE_CLICKHOUSE_GIT_IMPORT	ENABLE_CLICKHOUSE_ALL	A tool to analyze Git repositories	https://present.clickhouse.com/doc/en/interfaces/git_import/

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	https://clickhouse.local/
ENABLE_CLICKHOUSE_OBFUSCATOR	ENABLE_CLICKHOUSE_ALL	Table data obfuscator (convert real data to benchmark-ready one)	https://clickhouse.obfuscator/
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	Enalbe 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_BASE64	ENABLE_LIBRARIES	Enable base64	
ENABLE_BROTLI	ENABLE_LIBRARIES	Enable brotli	
ENABLE_BZIP2	ENABLE_LIBRARIES	Enable bzip2 compression support	
ENABLE_CAPNP	ENABLE_LIBRARIES	Enable Cap'n Proto	
ENABLE_CASSANDRA	ENABLE_LIBRARIES	Enable Cassandra	
ENABLE_CCACHE	ENABLE_CCACHE_BY_DEFAULT	Speedup re-compilations using ccache (external tool)	https://ccach
ENABLE_CLANG_TIDY	OFF	Use clang-tidy static analyzer	https://clang.tidy/
ENABLE_CURL	ENABLE_LIBRARIES	Enable curl	
ENABLE_DATASKETCHES	ENABLE_LIBRARIES	Enable DataSketches	

Name	Default value	Description	Comment
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_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	
ENABLE_NLP	ENABLE_LIBRARIES	Enable NLP functions support	
ENABLE_NURAFF	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	

Name	Default value	Description	Comment
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	
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	

Name	Default value	Description	Comment
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	
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 we framework USE_INTERNAL_GPRC set to OFF to force gRPC framework installed in The external be installed running sudo libgrpc++- grpc
USE_INTERNAL_GTEST_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system Google Test instead of bundled	

Name	Default value	Description	Comment
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	
USE_INTERNAL_LIBGSSASL_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	

Name	Default value	Description	Comment
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	
USE_INTERNAL_PROTOBUF_LIBRARY	NOT_UNBUNDLED	Set to FALSE to use system protobuf instead of bundled. (Experimental. Set to OFF on your own risk)	Normally w USE_INTER to OFF to fo protobuf lib installed in The exten installed in sudo apt-g protobuf-cc
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	

Name	Default value	Description	Comment
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	
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
ARCH_NATIVE	0	Add <code>-march=native</code> compiler flag. This makes your binaries non-portable but more performant code may be generated. This option overrides <code>ENABLE_*</code> options for specific instruction set. Highly not recommended to use.	
CLICKHOUSE_SPLIT_BINARY	OFF	Make several binaries (<code>clickhouse-server</code> , <code>clickhouse-client</code> etc.) instead of one bundled	
COMPILER_PIPE	ON	-pipe compiler option	Less <code>/tmp</code> usage, mc

Name	Default value	Description	Comment
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. w/o the CMake will continue
GLIBC_COMPATIBILITY	ON	Enable compatibility with older glibc libraries.	Only for Linux, x86_64
LINKER_NAME	OFF	Linker name or full path	Example values: lld-link
MAKE_STATIC_LIBRARIES	USE_STATIC_LIBRARIES	Disable to make shared libraries	
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: -asan (ASan) -ubsan (UBSan) -tsan (TSan)
SPLIT_SHARED_LIBRARIES	OFF	Keep all internal libraries as separate .so files	DEVELOPER ONLY. If enabled, ClickHouse will be slower.
STRIP_DEBUG_SYMBOLS_FUNCTIONS	STRIP_DSF_DEFAULT	Do not generate debugger info for ClickHouse functions	Provides faster linking (but requires to debug some source variables)."

Name	Default value	Description	Comment
UNBUNDLED	OFF	Use system libraries instead of ones in contrib/	We recommend avc can't guarantee all i exists for enthusiast whole idea of using is deeply flawed. Us
USE_INCLUDE_WHAT_YOU_USE	OFF	Automatically reduce unneeded includes in source code (external tool)	https://github.com/i
USE_LIBCXX	NOT_UNBUNDLED	Use libc++ and libc++abi instead of libstdc++	
USE_SENTRY	ENABLE_LIBRARIES	Use Sentry	
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 expansion).
WEVERYTHING	ON	Enable -Weverything option with some exceptions.	Add some warnings. Wpedantic. Intended to be found useful. API
WITH_COVERAGE	OFF	Profile the resulting binary/binaries	Compiler-specific cc

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."
    ${STRIP_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.

ClickHouse 开发

ClickHouse 架构概述

ClickHouse 是一个真正的列式数据库管理系统 (DBMS)。在 ClickHouse 中，数据始终是按列存储的，包括矢量（向量或列块）执行的过程。只要有可能，操作都是基于矢量进行分派的，而不是单个的值，这被称为《矢量化查询执行》，它有利于降低实际的数据处理开销。

这个想法并不新鲜，其可以追溯到 APL 编程语言及其后代：A+、J、K 和 Q。矢量编程被大量用于科学数据处理中。即使在关系型数据库中，这个想法也不是什么新的东西：比如，矢量编程也被大量用于 Vectorwise 系统中。

通常有两种不同的加速查询处理的方法：矢量化查询执行和运行时代码生成。在后者中，动态地为每一类查询生成代码，消除了间接分派和动态分派。这两种方法中，并没有哪一种严格地比另一种好。运行时代码生成可以更好地将多个操作融合在一起，从而充分利用 CPU 执行单元和流水线。矢量化查询执行不是特别实用，因为它涉及必须写到缓存并读回的临时向量。如果 L2 缓存容纳不下临时数据，那么这将成为一个问题。但矢量化查询执行更容易利用 CPU 的 SIMD 功能。朋友写的一篇研究论文表明，将两种方法结合起来是更好的选择。ClickHouse 使用了矢量化查询执行，同时初步提供了有限的运行时动态代码生成。

列 (Columns)

要表示内存中的列（实际上是列块），需使用 `IColumn` 接口。该接口提供了用于实现各种关系操作符的辅助方法。几乎所有的操作都是不可变的：这些操作不会更改原始列，但是会创建一个新的修改后的列。比如，`IColumn::filter` 方法接受过滤字节掩码，用于 `WHERE` 和 `HAVING` 关系操作符中。另外的例子：`IColumn::permute` 方法支持 `ORDER BY` 实现，`IColumn::cut` 方法支持 `LIMIT` 实现等等。

不同的 `IColumn` 实现 (`ColumnUInt8`、`ColumnString` 等) 负责不同的列内存布局。内存布局通常是一个连续的数组。对于数据类型为整型的列，只是一个连续的数组，比如 `std::vector`。对于 `String` 列和 `Array` 列，则由两个向量组成：其中一个向量连续存储所有的 `String` 或数组元素，另一个存储每一个 `String` 或 `Array` 的起始元素在第一个向量中的偏移。而 `ColumnConst` 则仅在内存中存储一个值，但是看起来像一个列。

字段

尽管如此，有时候也可能需要处理单个值。表示单个值，可以使用 `Field`。`Field` 是 `UInt64`、`Int64`、`Float64`、`String` 和 `Array` 组成的联合。`IColumn` 拥有 `operator[]` 方法来获取第 `n` 个值成为一个 `Field`，同时也拥有 `insert` 方法将一个 `Field` 追加到一个列的末尾。这些方法并不高效，因为它们需要处理表示单一值的临时 `Field` 对象，但是有更高效的方法比如 `insertFrom` 和 `insertRangeFrom` 等。

`Field` 中并没有足够的关于一个表 (table) 的特定数据类型的信息。比如，`UInt8`、`UInt16`、`UInt32` 和 `UInt64` 在 `Field` 中均表示为 `UInt64`。

抽象漏洞

`IColumn` 具有用于数据的常见关系转换的方法，但这些方法并不能够满足所有需求。比如，`ColumnUInt64` 没有用于计算两列和的方法，`ColumnString` 没有用于进行子串搜索的方法。这些无法计算的例程在 `IColumn` 之外实现。

列 (Columns) 上的各种函数可以通过使用 `IColumn` 的方法来提取 `Field` 值，或根据特定的 `IColumn` 实现的数据内存布局的知识，以一种通用但不高效的方式实现。为此，函数将会转换为特定的 `IColumn` 类型并直接处理内部表示。比如，`ColumnUInt64` 具有 `getData` 方法，该方法返回一个指向列的内部数组的引用，然后一个单独的例程可以直接读写或填充该数组。实际上，《抽象漏洞 (leaky abstractions)》允许我们以更高效的方式来实现各种特定的例程。

数据类型

`IDataType` 负责序列化和反序列化：读写二进制或文本形式的列或单个值构成的块。`IDataType` 直接与表的数据类型相对应。比如，有 `DataTypeUInt32`、`DataTypeDateTime`、`DataTypeString` 等数据类型。

`IDataType` 与 `IColumn` 之间的关联并不大。不同的数据类型在内存中能够用相同的 `IColumn` 实现来表示。比如，`DataTypeUInt32` 和 `DataTypeDateTime` 都是用 `ColumnUInt32` 或 `ColumnConstUInt32` 来表示的。另外，相同的数据类型也可以用不同的 `IColumn` 实现来表示。比如，`DataTypeUInt8` 既可以使用 `ColumnUInt8` 来表示，也可以使用过 `ColumnConstUInt8` 来表示。

`IDataType` 仅存储元数据。比如，`DataTypeUInt8` 不存储任何东西（除了 `vptr`）；`DataTypeFixedString` 仅存储 `N`（固定长度字符串的串长度）。

`IDataType` 具有针对性各种数据格式的辅助函数。比如如下一些辅助函数：序列化一个值并加上可能的引号；序列化一个值用于 JSON 格式；序列化一个值作为 XML 格式的一部分。辅助函数与数据格式并没有直接的对应。比如，两种不同的数据格式 `Pretty` 和 `TabSeparated` 均可以使用 `IDataType` 接口提供的 `serializeTextEscaped` 这一辅助函数。

块 (Block)

`Block` 是表示内存中表的子集 (chunk) 的容器，是由三元组：`(IColumn, IDatatype, 列名)` 构成的集合。在查询执行期间，数据是按 `Block` 进行处理的。如果我们有一个 `Block`，那么就有了数据（在 `IColumn` 对象中），有了数据的类型信息告诉我们如何处理该列，同时也有了列名（来自表的原始列名，或人为指定的用于临时计算结果的名字）。

当我们遍历一个块中的列进行某些函数计算时，会把结果列加入到块中，但不会更改函数参数中的列，因为操作是不可变的。之后，不需要的列可以从块中删除，但不是修改。这对于消除公共子表达式非常方便。

`Block` 用于处理数据块。注意，对于相同类型的计算，列名和类型对不同的块保持相同，仅列数据不同。最好把块数据（block data）和块头（block header）分离开来，因为小块大小会因复制共享指针和列名而带来很高的临时字符串开销。

块流 (Block Streams)

块流用于处理数据。我们可以使用块流从某个地方读取数据，执行数据转换，或将数据写到某个地方。`IBlockInputStream` 具有 `read` 方法，其能够在数据可用时获取下一个块。`IBlockOutputStream` 具有 `write` 方法，其能够将块写到某处。

块流负责：

1. 读或写一个表。表仅返回一个流用于读写块。
2. 完成数据格式化。比如，如果你打算将数据以 `Pretty` 格式输出到终端，你可以创建一个块输出流，将块写入该流中，然后进行格式化。
3. 执行数据转换。假设你现在有 `IBlockInputStream` 并且打算创建一个过滤流，那么你可以创建一个 `FilterBlockInputStream` 并用 `IBlockInputStream` 进行初始化。之后，当你从 `FilterBlockInputStream` 中拉取块时，会从你的流中提取一个块，对其进行过滤，然后将过滤后的块返回给你。查询执行流水线就是以这种方式表示的。

还有一些更复杂的转换。比如，当你从 `AggregatingBlockInputStream` 拉取数据时，会从数据源读取全部数据进行聚集，然后将聚集后的数据流返回给你。另一个例子：`UnionBlockInputStream` 的构造函数接受多个输入源和多个线程，其能够启动多线程从多个输入源并行读取数据。

块流使用 «pull» 方法来控制流：当你从第一个流中拉取块时，它会接着从嵌套的流中拉取所需的块，然后整个执行流水线开始工作。»pull« 和 «push» 都不是最好的方案，因为控制流不是明确的，这限制了各种功能的实现，比如多个查询同步执行（多个流水线合并到一起）。这个限制可以通过协程或直接运行互相等待的线程来解决。如果控制流明确，那么我们会有更多的可能性：如果我们定位了数据从一个计算单元传递到那些外部的计算单元中其中一个计算单元的逻辑。阅读这篇文章来获取更多的想法。

我们需要注意，查询执行流水线在每一步都会创建临时数据。我们要尽量使块的大小足够小，从而 CPU 缓存能够容纳下临时数据。在这个假设下，与其他计算相比，读写临时数据几乎没有任何开销的。我们也可以考虑一种替代方案：将流水线中的多个操作融合在一起，使流水线尽可能短，并删除大量临时数据。这可能是一个优点，但同时也有缺点。比如，拆分流水线使得中间数据缓存、获取同时运行的类似查询的中间数据以及相似查询的流水线合并等功能很容易实现。

格式 (Formats)

数据格式同块流一起实现。既有仅用于向客户端输出数据的展示格式，如 `IBlockOutputStream` 提供的 `Pretty` 格式，也有其它输入输出格式，比如 `TabSeparated` 或 `JSONEachRow`。

此外还有行流：`IRowInputStream` 和 `IRowOutputStream`。它们允许你按行 pull/push 数据，而不是按块。行流只需要简单地面向行格式实现。包装器 `BlockInputStreamFromRowInputStream` 和 `BlockOutputStreamFromRowOutputStream` 允许你将面向行的流转换为正常的面向块的流。

I/O

对于面向字节的输入输出，有 `ReadBuffer` 和 `WriteBuffer` 这两个抽象类。它们用来替代 C++ 的 `iostream`。不用担心：每个成熟的 C++ 项目都会有充分的理由使用某些东西来代替 `iostream`。

`ReadBuffer` 和 `WriteBuffer` 由一个连续的缓冲区和指向缓冲区中某个位置的一个指针组成。实现中，缓冲区可能拥有内存，也可能不拥有内存。有一个虚方法会使用随后的数据来填充缓冲区（针对 `ReadBuffer`）或刷新缓冲区（针对 `WriteBuffer`），该虚方法很少被调用。

`ReadBuffer` 和 `WriteBuffer` 的实现用于处理文件、文件描述符和网络套接字（socket），也用于实现压缩（`CompressedWriteBuffer` 在写入数据前需要先用一个 `WriteBuffer` 进行初始化并进行压缩）和其它用途。`ConcatReadBuffer`、`LimitReadBuffer` 和 `HashingWriteBuffer` 的用途正如其名字所描述的一样。

`ReadBuffer` 和 `WriteBuffer` 仅处理字节。为了实现格式化输入和输出（比如以十进制格式写一个数字），`ReadHelpers` 和 `WriteHelpers` 头文件中有一些辅助函数可用。

让我们来看一下，当你把一个结果集以 `JSON` 格式写到标准输出 (`stdout`) 时会发生什么。你已经准备好从 `IBlockInputStream` 获取结果集，然后创建 `WriteBufferFromFileDescriptor(STDOUT_FILENO)` 用于写字节到标准输出，创建 `JSONRowOutputStream` 并用 `WriteBuffer` 初始化，用于将行以 `JSON` 格式写到标准输出，你还可以在其上创建 `BlockOutputStreamFromRowOutputStream`，将其表示为 `IBlockOutputStream`。然后调用 `copyData` 将数据从 `IBlockInputStream` 传输到 `IBlockOutputStream`，一切工作正常。在内部，`JSONRowOutputStream` 会写入 `JSON` 分隔符，并以指向 `IColumn` 的引用和行数作为参数调用 `IDataType::serializeTextJSON` 函数。随后，`IDataType::serializeTextJSON` 将会调用 `WriteHelpers.h` 中的一个方法：比如，`writeText` 用于数值类型，`writeJSONString` 用于 `DataTimeString`。

表 (Tables)

表由 `IStorage` 接口表示。该接口的不同实现对应不同的表引擎。比如 `StorageMergeTree`、`StorageMemory` 等。这些类的实例就是表。

`IStorage` 中最重要的方法是 `read` 和 `write`，除此之外还有 `alter`、`rename` 和 `drop` 等方法。`read` 方法接受如下参数：需要从表中读取的列集，需要执行的 `AST` 查询，以及所需返回的流的数量。`read` 方法的返回值是一个或多个 `IBlockInputStream` 对象，以及在查询执行期间在一个表引擎内完成的关于数据处理阶段的信息。

在大多数情况下，`read` 方法仅负责从表中读取指定的列，而不会进行进一步的数据处理。进一步的数据处理均由查询解释器完成，不由 `IStorage` 负责。

但是也有值得注意的例外：

- `AST` 查询被传递给 `read` 方法，表引擎可以使用它来判断是否能够使用索引，从而从表中读取更少的数据。
- 有时候，表引擎能够将数据处理到一个特定阶段。比如，`StorageDistributed` 可以向远程服务器发送查询，要求它们将来自不同的远程服务器能够合并的数据处理到某个阶段，并返回预处理后的数据，然后查询解释器完成后续的数据处理。

表的 `read` 方法能够返回多个 `IBlockInputStream` 对象以允许并行处理数据。多个块输入流能够从一个表中并行读取。然后你可以通过不同的转换对这些流进行装饰（比如表达式求值或过滤），转换过程能够独立计算，并在其上创建一个 `UnionBlockInputStream`，以并行读取多个流。

另外也有 `TableFunction`。`TableFunction` 能够在查询的 `FROM` 字句中返回一个临时的 `IStorage` 以供使用。

要快速了解如何实现自己的表引擎，可以查看一些简单的表引擎，比如 `StorageMemory` 或 `StorageTinyLog`。

作为 `read` 方法的结果，`IStorage` 返回 `QueryProcessingStage` - 关于 `storage` 里哪部分查询已经被计算的信息。当前我们仅有非常粗粒度的信息。`Storage` 无法告诉我们«对于这个范围的数据，我已经处理完了 WHERE 字句里的这部分表达式»。我们需要在这个地方继续努力。

解析器 (Parsers)

查询由一个手写递归下降解析器解析。比如，`ParserSelectQuery` 只是针对查询的不同部分递归地调用下层解析器。解析器创建 `AST`。`AST` 由节点表示，节点是 `IAST` 的实例。

由于历史原因，未使用解析器生成器。

解释器 (Interpreters)

解释器负责从 `AST` 创建查询执行流水线。既有一些简单的解释器，如 `InterpreterExistsQuery` 和 `InterpreterDropQuery`，也有更复杂的解释器，如 `InterpreterSelectQuery`。查询执行流水线由块输入或输出流组成。比如，`SELECT` 查询的解释结果是从 `FROM` 字句的结果集中读取数据的 `IBlockInputStream`；`INSERT` 查询的结果是写入需要插入的数据的 `IBlockOutputStream`；`SELECT INSERT` 查询的解释结果是 `IBlockInputStream`，它在第一次读取时返回一个空结果集，同时将数据从 `SELECT` 复制到 `INSERT`。

`InterpreterSelectQuery` 使用 `ExpressionAnalyzer` 和 `ExpressionActions` 机制来进行查询分析和转换。这是大多数基于规则的查询优化完成的地方。`ExpressionAnalyzer` 非常混乱，应该进行重写：不同的查询转换和优化应该被提取出来并划分成不同的类，从而允许模块化转换或查询。

函数 (Functions)

函数既有普通函数，也有聚合函数。对于聚合函数，请看下一节。

普通函数不会改变行数 - 它们的执行看起来就像是独立地处理每一行数据。实际上，函数不会作用于一个单独的行上，而是作用在以 `Block` 为单位的数据上，以实现向量查询执行。

还有一些杂项函数，比如 `块大小`、`rowNumberInBlock`，以及 `跑累积`，它们对块进行处理，并且不遵从行的独立性。

`ClickHouse` 具有强类型，因此隐式类型转换不会发生。如果函数不支持某个特定的类型组合，则会抛出异常。但函数可以通过重载以支持许多不同的类型组合。比如，`plus` 函数（用于实现 `+` 运算符）支持任意数字类型的组合：`UInt8 + Float32`, `UInt16 + Int8` 等。同时，一些可变参数的函数能够级接收任意数目的参数，比如 `concat` 函数。

实现函数可能有些不方便，因为函数的实现需要包含所有支持该操作的数据类型和 `IColumn` 类型。比如，`plus` 函数能够利用 C++ 模板针对不同的数字类型组合、常量以及非常量的左值和右值进行代码生成。

这是一个实现动态代码生成的好地方，从而能够避免模板代码膨胀。同样，运行时代码生成也使得实现融合函数成为可能，比如融合《乘-加》，或者在单层循环迭代中进行多重比较。

由于向量查询执行，函数不会«短路»。比如，如果你写 `WHERE f(x) AND g(y)`，两边都会进行计算，即使是对于 `f(x)` 为 0 的行（除非 `f(x)` 是零常量表达式）。但是如果 `f(x)` 的选择条件很高，并且计算 `f(x)` 比计算 `g(y)` 要划算得多，那么最好进行多遍计算：首先计算 `f(x)`，根据计算结果对列数据进行过滤，然后计算 `g(y)`，之后只需对较小数量的数据进行过滤。

聚合函数

聚合函数是状态函数。它们将传入的值激活到某个状态，并允许你从该状态获取结果。聚合函数使用 `IAggregateFunction` 接口进行管理。状态可以非常简单（`AggregateFunctionCount` 的状态只是一个单一的 `UInt64` 值），也可以非常复杂（`AggregateFunctionUniqCombined` 的状态是由一个线性数组、一个散列表和一个 `HyperLogLog` 概率数据结构组合而成的）。

为了能够在执行一个基数很大的 `GROUP BY` 查询时处理多个聚合状态，需要在 `Arena`（一个内存池）或任何合适的内存块中分配状态。状态可以有一个非平凡的构造器和析构器：比如，复杂的聚合状态能够自己分配额外的内存。这需要注意状态的创建和销毁并恰当地传递状态的所有权，以跟踪谁将何时销毁状态。

聚合状态可以被序列化和反序列化，以在分布式查询执行期间通过网络传递或者在内存不够的时候将其写到硬盘。聚合状态甚至可以通过 `DataTypeAggregateFunction` 存储到一个表中，以允许数据的增量聚合。

聚合函数状态的序列化数据格式目前尚未版本化。如果只是临时存储聚合状态，这样是可以的。但是我们有 `AggregatingMergeTree` 表引擎用于增量聚合，并且人们已经在生产中使用它。这就是为什么在未来当我们更改任何聚合函数的序列化格式时需要增加向后兼容的支持。

服务器 (Server)

服务器实现了多个不同的接口：

- 一个用于任何外部客户端的 HTTP 接口。
- 一个用于本机 ClickHouse 客户端以及在分布式查询执行中跨服务器通信的 TCP 接口。
- 一个用于传输数据以进行拷贝的接口。

在内部，它只是一个没有协程、纤程等的基础多线程服务器。服务器不是为处理高速率的简单查询设计的，而是为处理相对低速率的复杂查询设计的，每一个复杂查询能够对大量的数据进行处理分析。

服务器使用必要的查询执行需要的环境初始化 `Context` 类：可用数据库列表、用户和访问权限、设置、集群、进程列表和查询日志等。这些环境被解释器使用。

我们维护了服务器 TCP 协议的完全向后向前兼容性：旧客户端可以和新服务器通信，新客户端也可以和旧服务器通信。但是我们并不想永久维护它，我们将在大约一年后删除对旧版本的支持。

对于所有的外部应用，我们推荐使用 HTTP 接口，因为该接口很简单，容易使用。TCP 接口与内部数据结构的联系更加紧密：它使用内部格式传递数据块，并使用自定义帧来压缩数据。我们没有发布该协议的 C 库，因为它需要链接大部分的 ClickHouse 代码库，这是不切实际的。

分布式查询执行

集群设置中的服务器大多是独立的。你可以在一个集群中的一个或多个服务器上创建一个 `Distributed` 表。`Distributed` 表本身并不存储数据，它只为集群的多个节点上的所有本地表提供一个«视图（view）»。当从 `Distributed` 表中进行 `SELECT` 时，它会重写该查询，根据负载平衡设置来选择远程节点，并将查询发送给节点。`Distributed` 表请求远程服务器处理查询，直到可以合并来自不同服务器的中间结果的阶段。然后它接收中间结果并进行合并。分布式表会尝试将尽可能多的工作分配给远程服务器，并且不会通过网络发送太多的中间数据。

当 `IN` 或 `JOIN` 子句中包含子查询并且每个子查询都使用分布式表时，事情会变得更加复杂。我们有不同的策略来执行这些查询。

分布式查询执行没有全局查询计划。每个节点都有针对自己的工作部分的本地查询计划。我们仅有简单的一次性分布式查询执行：将查询发送给远程节点，然后合并结果。但是对于具有高基数的 `GROUP BY` 或具有大量临时数据的 `JOIN` 这样困难的查询来说，这是不可行的：在这种情况下，我们需要在服务器之间«改组»数据，这需要额外的协调。ClickHouse 不支持这类查询执行，我们需要在这方面进行努力。

合并树

`MergeTree` 是一系列支持按主键索引的存储引擎。主键可以是一个任意的列或表达式的元组。`MergeTree` 表中的数据存储于«分块»中。每一个分块以主键序存储数据（数据按主键元组的字典序排序）。表的所有列都存储在这些«分块»中分离的 `column.bin` 文件中。`column.bin` 文件由压缩块组成，每一个块通常是 64 KB 到 1 MB 大小的未压缩数据，具体取决于平均值大小。这些块由一个接一个连续放置的列值组成。每一列的列值顺序相同（顺序由主键定义），因此当你按多列进行迭代时，你能够得到相应列的值。

主键本身是«稀疏»的。它并不是索引单一的行，而是索引某个范围内的数据。一个单独的 `primary.idx` 文件具有每个第 N 行的主键值，其中 N 称为 `index_granularity`（通常，N = 8192）。同时，对于每一列，都有带有标记的 `column.mrk` 文件，该文件记录的是每个第 N 行在数据文件中的偏移量。每个标记是一个 `pair`：文件中的偏移量到压缩块的起始，以及解压缩块中的偏移量到数据的起始。通常，压缩块根据标记对齐，并且解压缩块中的偏移量为 0。`primary.idx` 的数据始终驻留在内存，同时 `column.mrk` 的数据被缓存。

当我们从 `MergeTree` 的一个分块中读取部分内容时，我们会查看 `primary.idx` 数据并查找可能包含所请求数据的范围，然后查看 `column.mrk` 并计算偏移量从而得知从哪里开始读取些范围的数据。由于稀疏性，可能会读取额外的数据。`ClickHouse` 不适用于高负载的简单点查询，因为对于每一个键，整个 `index_granularity` 范围的行的数据都需要读取，并且对于每一列需要解压缩整个压缩块。我们使索引稀疏，是因为每一个单一的服务器需要在索引没有明显内存消耗的情况下，维护数万亿行的数据。另外，由于主键是稀疏的，导致其不是唯一的：无法在 `INSERT` 时检查一个键在表中是否存在。你可以在一个表中使用同一个键创建多个行。

当你向 `MergeTree` 中插入一堆数据时，数据按主键排序并形成一个新的分块。为了保证分块的数量相对较少，有后台线程定期选择一些分块并将它们合并成一个有序的分块，这就是 `MergeTree` 的名称来源。当然，合并会导致«写入放大»。所有的分块都是不可变的：它们仅会被创建和删除，不会被修改。当运行 `SELECT` 查询时，`MergeTree` 会保存一个表的快照（分块集合）。合并之后，还会保留旧的分块一段时间，以便发生故障后更容易恢复，因此如果我们发现某些合并后的分块可能已损坏，我们可以将其替换为原分块。

`MergeTree` 不是 `LSM` 树，因为它不包含»memtable«和»log«：插入的数据直接写入文件系统。这使得它仅适用于批量插入数据，而不适用于非常频繁地一行一行插入 - 大约每秒一次是没问题的，但是每秒一千次就会有问题。我们这样做是为了简单起见，因为我们已经在我们的应用中批量插入数据。

`MergeTree` 表只能有一个（主）索引：没有任何辅助索引。在一个逻辑表下，允许有多个物理表示，比如，可以以多个物理顺序存储数据，或者同时表示预聚合数据和原始数据。

有些 `MergeTree` 引擎会在后台合并期间做一些额外工作，比如 `CollapsingMergeTree` 和 `AggregatingMergeTree`。这可以视为对更新的特殊支持。请记住这些不是真正的更新，因为用户通常无法控制后台合并将会执行的时间，并且 `MergeTree` 中的数据几乎总是存储在多个分块中，而不是完全合并的形式。

复制（Replication）

`ClickHouse` 中的复制是基于表实现的。你可以在同一个服务器上有一些可复制的表和不可复制的表。你也可以以不同的方式进行表的复制，比如一个表进行双因子复制，另一个进行三因子复制。

复制是在 `ReplicatedMergeTree` 存储引擎中实现的。`ZooKeeper` 中的路径被指定为存储引擎的参数。`ZooKeeper` 中所有具有相同路径的表互为副本：它们同步数据并保持一致性。只需创建或删除表，就可以实现动态添加或删除副本。

复制使用异步多主机方案。你可以将数据插入到与 `ZooKeeper` 进行会话的任意副本中，并将数据复制到所有其它副本中。由于 `ClickHouse` 不支持 `UPDATEs`，因此复制是无冲突的。由于没有对插入的仲裁确认，如果一个节点发生故障，刚刚插入的数据可能会丢失。

用于复制的元数据存储在 `ZooKeeper` 中。其中一个复制日志列出了要执行的操作。操作包括：获取分块、合并分块和删除分区等。每一个副本将复制日志复制到其队列中，然后执行队列中的操作。比如，在插入时，在复制日志中创建«获取分块»这一操作，然后每一个副本都会去下载该分块。所有副本之间会协调进行合并以获得相同字节的结果。所有的分块在所有的副本上以相同的方式合并。为实现该目的，其中一个副本被选为领导者，该副本首先进行合并，并把«合并分块»操作写到日志中。

复制是物理的：只有压缩的分块会在节点之间传输，查询则不会。为了降低网络成本（避免网络放大），大多数情况下，会在每一个副本上独立地处理合并。只有在存在显著的合并延迟的情况下，才会通过网络发送大块的合并分块。

另外，每一个副本将其状态作为分块和校验和组成的集合存储在 `ZooKeeper` 中。当本地文件系统中的状态与 `ZooKeeper` 中引用的状态不同时，该副本会通过从其它副本下载缺失和损坏的分块来恢复其一致性。当本地文件系统中出现一些意外或损坏的数据时，`ClickHouse` 不会将其删除，而是将其移动到一个单独的目录下并忘记它。

ClickHouse 集群由独立的分片组成，每一个分片由多个副本组成。集群不是弹性的，因此在添加新的分片后，数据不会自动在分片之间重新平衡。相反，集群负载将变得不均衡。该实现为你提供了更多控制，对于相对较小的集群，例如只有数十个节点的集群来说是很好的。但是对于我们在生产中使用的具有数百个节点的集群来说，这种方法成为一个重大缺陷。我们应该实现一个表引擎，使得该引擎能够跨集群扩展数据，同时具有动态复制的区域，这些区域能够在集群之间自动拆分和平衡。

ClickHouse支持Linux, FreeBSD 及 Mac OS X 系统。

Windows使用指引

如果您的系统是Windows，则需要创建Ubuntu虚拟机。可以安装VirtualBox来构建虚拟机。Ubuntu的下载链接为：<https://www.ubuntu.com/#download>。请使用下载好的镜像创建一个虚拟机（请确保虚拟机有至少4GB的内存容量）。在Ubuntu中使用«terminal»程序（gnome-terminal，konsole等）运行命令行终端，或使用快捷键Ctrl+Alt+T。

在GitHub上创建源码库

您需要(申请)一个GitHub账户来使用ClickHouse。

如果没有账户，请在<https://github.com>上注册一个。如果没有SSH密钥，请在本地创建密钥并将公钥上传到GitHub上。这有助于你提交更新代码。并且在不同的SSH服务端，你也可以使用相同的SSH密钥。

要创建ClickHouse源码库的分支，请在<https://github.com/ClickHouse/ClickHouse>页面上点击右上角的«fork»按钮。它会在本账户上创建您个人的ClickHouse/ClickHouse分支。

若要参与开发，首先请在ClickHouse的分支中提交您期望的变更，然后创建一个«pull请求»，以便这些变更能够被(ClickHouse/ClickHouse)主库接受。

请先安装git来使用git源码库。

请在Ubuntu终端上使用下列的指令来安装git:

```
sudo apt update  
sudo apt install git
```

在<https://education.github.com/git-cheat-sheet-education.pdf>中找到有关使用Git的简易手册。有关Git的详细手册，请参见: <https://git-scm.com/book/ru/v2>。

拷贝源码库到开发机

接下来，请将源码下载到开发机上。这步操作被称为«拷贝源码库»，是因为它在您的开发机上创建了源码库的本地副本。

在终端命令行输入下列指令：

```
git clone --recursive git@github.com:your_github_username/ClickHouse.git  
cd ClickHouse
```

请注意，您需要将`your_github_username`替换成实际使用的账户名！

这个指令将创建一个包含项目副本的ClickHouse工作目录。

重要的是，工作目录的路径中不应包含空格，因为这可能会导致运行构建系统时出现问题。

请注意，ClickHouse源码库使用了submodules。这是对其他库的引用（即项目所依赖的外部库）。即在拷贝源码库时，需要如上述指令中那样指定`--recursive`。如果在拷贝源码库时没有包含子模块，需要执行使用下列的指令：

```
git submodule init  
git submodule update
```

可以通过 `git submodule status` 来检查子模块的状态。

如果提示下列的错误信息：

```
Permission denied (publickey).  
fatal: Could not read from remote repository.  
  
Please make sure you have the correct access rights  
and the repository exists.
```

这通常表示缺少用于连接GitHub的SSH密钥。这些密钥一般都在`~/.ssh`中。要接受SSH密钥，请在GitHub UI的设置页面中上传它们。

您还可以通过https协议来拷贝源码库：

```
git clone https://github.com/ClickHouse/ClickHouse.git
```

但是，这无法将变更提交到服务器上。您仍然可以暂时使用，并后续再添加SSH密钥，用`git remote`命令替换源码库的远程地址。

还可以将原始ClickHouse库的地址添加到本地库中，以便从那里获取更新：

```
git remote add upstream git@github.com:ClickHouse/ClickHouse.git
```

命令执行成功后，可以通过执行`git pull upstream master`，从ClickHouse的主分支中拉去更新。

使用子模块

在git中使用子模块可能会很痛苦。接下来的命令将有助于管理它：

```
# ! each command accepts --recursive  
# 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
```

接下来的命令将帮助您将所有子模块重置为初始状态（！华林！ -里面的任何chnges将被删除）：

```
# Synchronizes submodules' remote URL with .gitmodules  
git submodule sync --recursive  
# Update the registered submodules with initialize not yet initialized  
git submodule update --init --recursive  
# 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 --recursive  
git submodule foreach git submodule update --init --recursive  
git submodule foreach git submodule foreach git reset --hard  
git submodule foreach git submodule foreach git clean -xfd
```

构建系统

ClickHouse使用 CMake 和 Ninja 来构建系统。

CMake - 一个可以生成Ninja文件的元构建系统（构建任务）。

Ninja - 一个轻量级的构建系统，专注于速度，用于执行这些cmake生成的任务。

在Ubuntu,Debian或者Mint系统上执行sudo apt install cmake ninja-build来安装ninja。

在CentOS,RedHat系统上执行sudo yum install cmake ninja-build。

如果您曾经使用过Arch或Gentoo，那么也许知道如何安装CMake。

若要在Mac OS X上安装CMake和Ninja，请先安装Homebrew，然后再通过brew安装其他内容：

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"  
brew install cmake ninja
```

接下来，检查CMake的版本：cmake --version。如果版本低于3.3，则需要从以下网站安装更新版本：<https://cmake.org/download/>。

可供选择的外部库

ClickHouse使用多个外部库进行构建。大多数外部库不需要单独安装，而是和ClickHouse一起在子模块中构建。可以查看contrib中罗列的清单。

C++ 编译器

We support clang starting from version 11.

On Ubuntu/Debian you can use the automatic installation script (check [official webpage](#))

```
sudo bash -c "$(wget -O - https://apt.llvm.org/llvm.sh)"
```

构建的过程

如果当前已经准备好构建ClickHouse，我们建议您在ClickHouse中创建一个单独的目录build，其中包含所有构建组件：

```
mkdir build  
cd build
```

您也可以有多个不同类型的构建目录（例如，build_release, build_debug等等）。

在build目录下，通过运行CMake配置构建。在第一次运行之前，请定义用于指定编译器的环境变量（本示例中为gcc 9 编译器）。

```
export CC=clang CXX=clang++  
cmake ..
```

CC变量指代C的编译器（C Compiler的缩写），而CXX变量指代要使用哪个C++编译器进行编译。

为了更快的构建，请使用debug构建类型-不含优化的构建。为此提供以下的参数-D CMAKE_BUILD_TYPE=Debug:

```
cmake -D CMAKE_BUILD_TYPE=Debug ..
```

您可以通过在**build**目录中运行此命令来更改构建类型。

运行ninja进行构建：

```
ninja clickhouse-server clickhouse-client
```

在此示例中，仅将构建所需的二进制文件。

如果您需要构建所有的二进制文件（**utilities**和**tests**），请运行不带参数的**ninja**：

```
ninja
```

全量构建需要大约30GB的可用磁盘空间或15GB的空间来构建主要的二进制文件。

当构建的机器上有大量内存时，可考虑设置与-j参数并行运行的构建任务数量：

```
ninja -j 1 clickhouse-server clickhouse-client
```

在拥有4GB内存的机器上，建议设置成1，在拥有8GB内存的机器上，建议按-j 2设置。

如果您收到以下消息：

```
ninja : error : loading'build.ninja' : No such file or directory
```

则表示生成构建配置失败，请检查上述消息。

成功启动构建过程后，您将看到构建进度-已处理任务的数量和任务总数。

在libhdfs2库中生成有关protobuf文件的消息时，可能会显示诸如libprotobuf WARNING。它们没有影响，可以忽略不计。

成功构建后，会得到一个可执行文件ClickHouse/<build_dir>/programs/clickhouse:

```
ls -l programs/clickhouse
```

运行ClickHouse可执行文件

要以当前的用户身份运行服务，请进入到ClickHouse/programs/server/ 目录（在**build**文件夹外）并运行：

```
../../build/programs/clickhouse server
```

在这种情况下，ClickHouse将使用位于当前目录中的配置文件。您可以从任何目录运行**Clickhouse server**，并将配置文件--config-file的路径指定为命令行参数。

在另外一个终端上连接ClickHouse的clickhouse-client客户端，请进入到ClickHouse/build/programs/ 并运行./clickhouse client。

如果您在Mac OS X 或者 FreeBSD上收到Connection refused的消息，请尝试指定主机地址为127.0.0.1：

```
clickhouse client --host 127.0.0.1
```

您可以使用自定义构建的ClickHouse二进制文件替换系统中安装的ClickHouse二进制文件的生成版本。为此，请参照官方网站上的说明在计算机上安装ClickHouse。接下来，运行以下命令：

```
sudo service clickhouse-server stop  
sudo cp ClickHouse/build/programs/clickhouse /usr/bin/  
sudo service clickhouse-server start
```

请注意，`clickhouse-client`，`clickhouse-server`和其他服务通常共享`clickhouse`二进制文件的符号链接。

您还可以使用系统上安装的ClickHouse软件包中的配置文件运行自定义构建的ClickHouse二进制文件：

```
sudo service clickhouse-server stop  
sudo -u clickhouse ClickHouse/build/programs/clickhouse server --config-file /etc/clickhouse-server/config.xml
```

IDE (集成开发环境)

如果您还不知道使用哪款IDE，我们推荐使用CLion。CLion是一款商业软件，但能够有30天的免费使用时间。它同时也对学生免费。CLion可以在Linux和Mac OS X上使用。

KDevelop和QTCreator是另外两款适合开发ClickHouse的替代IDE。尽管不太稳定，但KDevelop还是作为一款非常便捷的IDE。如果KDevelop在打开项目后不久崩溃，则您应该在打开项目文件列表后立即单击《全部停止》按钮。按此处理后，KDevelop可以正常使用。

作为简易的代码编辑器，您可以使用Sublime Text或Visual Studio Code或Kate（在Linux上都可用）。

值得一提的是CLion会创建自己的build路径，它还会自行选择debug作为构建类型。对于配置，它使用CLion中定义的CMake版本，而不是您安装的版本。最后，CLion会使用make而不是ninja去构建任务。这属于正常的现象，请记住这一点，以免造成混淆。

编写代码

ClickHouse的架构描述可以在此处查看：<https://clickhouse.com/docs/en/development/architecture/>

代码风格指引：<https://clickhouse.com/docs/en/development/style/>

编写测试用例：<https://clickhouse.com/docs/en/development/tests/>

任务列表：<https://github.com/ClickHouse/ClickHouse/issues?q=is%3Aopen+is%3Aissue+label%3A%22easy+task%22>

测试数据

开发ClickHouse通常需要加载现实的数据集，尤其是在性能测试的场景。我们可以从Yandex.Metrica获取一组特别准备的匿名数据。这些数据需要额外使用3GB的空闲磁盘空间。请注意，完成大多数开发任务并不需要此数据。

```
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 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
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.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, ClickPhraselD 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, ClickTargetPhraselD 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.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
FixedSize(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.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(Int64), `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

```

创建拉取请求

进入到GitHub 用户界面中的fork库。如果您已经在某个分支中进行开发，则需要选择该分支。在屏幕中有一个《拉取请求》的按钮。实际上这等价于《创建一个请求以接受对主库的变更》。

即使工作尚未完成，也可以创建拉取请求。在这种情况下，请在标题的开头加上«WIP»（正在进行中），以便后续更改。这对于协同审查和讨论更改以及运行所有可用测试用例很有用。提供有关变更的简短描述很重要，这将在后续用于生成重新发布变更日志。

Yandex成员一旦在您的拉取请求上贴上«可以测试»标签，就会开始测试。一些初始检查项（例如，代码类型）的结果会在几分钟内反馈。构建的检查结果将在半小时内完成。而主要的测试用例集结果将在一小时内报告给您。

系统将分别为您的拉取请求准备ClickHouse二进制版本。若要检索这些构建信息，请在检查列表中单击« ClickHouse构建检查»旁边的«详细信息»链接。在这里，您会找到指向ClickHouse的.deb软件包的直接链接，此外，甚至可以将其部署在生产服务器上（如果您不担心）。

某些构建项很可能会在首次构建时失败。这是因为我们同时检查了基于gcc和clang的构建，几乎所有现有的被clang启用的警告（总是带有-Werror标志）。在同一页面上，您可以找到所有构建的日志，因此不必以所有可能的方式构建ClickHouse。

使用的三方库

图书馆	许可
base64	BSD2-条款许可
升压	提升软件许可证1.0
brotli	MIT
capnproto	MIT
cctz	Apache许可证2.0
双转换	BSD3-条款许可
FastMemcpy	MIT
googletest	BSD3-条款许可
超扫描	BSD3-条款许可
libcxxabi	BSD + MIT
libdivide	Zlib许可证
libgsasl	LGPL v2.1
libhdfs3	Apache许可证2.0
libmetrohash	Apache许可证2.0
libpcg-随机	Apache许可证2.0
libressl	OpenSSL许可证
librdkafka	BSD2-条款许可

libwidechar_width	CC0 1.0通用
llvm	BSD3-条款许可
lz4	BSD2-条款许可
mariadb-连接器-c	LGPL v2.1
murmurhash	公共领域
pdqsort	Zlib许可证
poco	提升软件许可证-1.0版
protobuf	BSD3-条款许可
re2	BSD3-条款许可
UnixODBC	LGPL v2.1
zlib-ng	Zlib许可证
zstd	BSD3-条款许可

在 Mac OS X 中编译 ClickHouse

ClickHouse 支持在 Mac OS X 10.12 版本中编译。若您在用更早的操作系统版本，可以尝试在指令中使用 Gentoo Prefix 和 clang sl。

通过适当的更改，它应该可以适用于任何其他的 Linux 发行版。

安装 Homebrew

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

安装编译器，工具库

```
$ brew install cmake ninja libtool gettext
```

拉取 ClickHouse 源码

```
git clone --recursive git@github.com:ClickHouse/ClickHouse.git
## or: git clone --recursive https://github.com/ClickHouse/ClickHouse.git
cd ClickHouse
```

编译 ClickHouse

```
$ mkdir build  
$ cd build  
$ cmake .. -DCMAKE_CXX_COMPILER=`which clang++` -DCMAKE_C_COMPILER=`which clang`  
$ ninja  
$ cd ..
```

注意事项

若你想运行 clickhouse-server，请先确保增加系统的最大文件数配置。

注意

可能需要用 sudo

为此，请创建以下文件：

/资源库/LaunchDaemons/limit.maxfiles.plist:

```
<?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>
```

执行以下命令：

```
$ sudo chown root:wheel /Library/LaunchDaemons/limit.maxfiles.plist
```

然后重启。

可以通过 `ulimit -n` 命令来检查是否生效。

如何在Linux中编译Mac OS X ClickHouse

Linux机器也可以编译运行在OS X系统的clickhouse二进制包，这可以用于在Linux上跑持续集成测试。如果要在Mac OS X上直接构建ClickHouse，请参考另外一篇指南：https://clickhouse.com/docs/zh/development/build_osx/

Mac OS X的交叉编译基于以下构建说明，请首先遵循它们。

安装Clang-8

按照<https://apt.llvm.org/>中的说明进行Ubuntu或Debian安装。

例如，安装Bionic的命令如下：

```
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
```

安装交叉编译工具集

我们假设安装 `cctools` 在 `${CCTOOLS}` 路径下

```
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

cd ${CCTOOLS}
wget https://github.com/phracker/MacOSX-SDKs/releases/download/10.15/MacOSX10.15.sdk.tar.xz
tar xJf MacOSX10.15.sdk.tar.xz
```

编译 ClickHouse

```
cd ClickHouse
mkdir build-osx
CC=clang-8 CXX=clang++-8 cmake . -Bbuild-osx -DCMAKE_SYSTEM_NAME=Darwin \
-DCMAKE_AR:FILEPATH=${CCTOOLS}/bin/x86_64-apple-darwin-ar \
-DCMAKE_RANLIB:FILEPATH=${CCTOOLS}/bin/x86_64-apple-darwin-ranlib \
-DLINKER_NAME=${CCTOOLS}/bin/x86_64-apple-darwin-ld \
-DSDK_PATH=${CCTOOLS}/MacOSX10.15.sdk
ninja -C build-osx
```

生成的二进制文件将具有Mach-O可执行格式，并且不能在Linux上运行。

如何将测试查询添加到 ClickHouse CI

ClickHouse有数百个（甚至数千个）功能。每个提交都由包含数千个测试用例的一组复杂测试进行检查。

核心功能经过了很多的测试，但是ClickHouse CI可以发现一些极端情况和不同的功能组合。

我们看到的大多数错误/回归都发生在测试覆盖率较差的灰色区域中。

我们非常有兴趣通过测试涵盖实现生活中使用的大多数可能的场景和功能组合。

为什么要添加测试

为什么/何时应将测试用例添加到ClickHouse代码中：

- 1) 您使用了一些复杂的场景/功能组合/您有一些可能未被广泛使用的情况
- 2) 您会看到更改日志中没有通知的版本之间的某些行为发生了变化
- 3) 您只是想帮助提高ClickHouse的质量并确保您使用的功能在未来的版本中不会被破坏
- 4) 一旦测试被添加/接受，您可以确保您检查的角落案例永远不会被意外损坏。
- 5) 你将成为伟大的开源社区的一份子
- 6) 您的名字将出现在`system.contributors`表中！
- 7) 你会让世界变得更好。

要做的步骤

先决条件

我假设你运行一些Linux机器（你可以在其他操作系统上使用 docker/虚拟机）和任何现代浏览器/互联网连接，并且你有一些基本的Linux和SQL技能。

不需要任何高度专业化的内容（因此您不需要了解 C++ 或了解 ClickHouse CI 的工作原理）。

准备

1) [create GitHub account](#) (如果你还没有)

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](#) - 打开 <https://github.com/ClickHouse/ClickHouse> 和 press fork button in the top right corner:



4) 例如，将代码fork克隆到PC上的某个文件夹，[~/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
```

测试的新分支

1) 从最新的clickhouse master创建一个新分支

```
cd ~/workspace/ClickHouse
git fetch upstream
git checkout -b name_for_a_branch_with_my_test upstream/master
```

安装并运行 clickhouse

1) 安装clickhouse-server ([参考离线文档](#))

2) 安装测试配置（它将使用Zookeeper模拟实现并调整一些设置）

```
cd ~/workspace/ClickHouse/tests/config
sudo ./install.sh
```

3) 运行clickhouse-server

```
sudo systemctl restart clickhouse-server
```

创建测试文件

1) 找到测试的编号 - 在[tests/queries/0_stateless/](#)中找到编号最大的文件

```
$ cd ~/workspace/ClickHouse  
$ ls tests/queries/0_stateless/[0-9]*.reference | tail -n 1  
tests/queries/0_stateless/01520_client_print_query_id.reference
```

目前，测试的最后一个数字是01520，所以我的测试将有数字01521

2) 使用您测试的功能的下一个编号和名称创建一个SQL文件

```
touch tests/queries/0_stateless/01521_dummy_test.sql
```

3) 使用您最喜欢的编辑器编辑SQL文件（请参阅下面的创建测试提示）

```
vim tests/queries/0_stateless/01521_dummy_test.sql
```

4) 运行测试，并将其结果放入参考文件中：

```
clickhouse-client -nmT < tests/queries/0_stateless/01521_dummy_test.sql | tee  
tests/queries/0_stateless/01521_dummy_test.reference
```

5) 确保一切正确，如果测试输出不正确（例如由于某些错误），请使用文本编辑器调整参考文件。

如何创建一个好的测试

- 测试应该是
 - 最小 - 仅创建与测试功能相关的表，删除不相关的列和部分查询
 - 快速 - 不应超过几秒钟（更好的亚秒）
 - 正确 - 失败则功能不起作用
 - 确定性的
 - 隔离/无状态
 - 不要依赖一些环境的东西
 - 尽可能不要依赖时间
- 尝试覆盖极端情况(zeros / Nulls / empty sets / throwing exceptions)
- 要测试该查询返回错误，您可以在查询后添加特殊注释：-- { serverError 60 } 或-- { clientError 20 }
- 不要切换数据库（除非必要）
- 如果需要，您可以在同一节点上创建多个表副本
- 您可以在需要时使用测试集群定义之一（请参阅 system.clusters）
- 使用 number / numbers_mt / zeros / zeros_mt 和类似的查询要在适用时初始化数据
- 在测试之后和测试之前清理创建的对象（DROP IF EXISTS） - 在有一些脏状态的情况下
- 优先选择同步操作模式 (mutations, merges)
- 以0_stateless文件夹中的其他SQL文件为例
- 确保您想要测试的特性/特性组合尚未被现有测试覆盖

测试命名规则

正确地命名测试非常重要，因此可以在clickhouse-test调用中关闭一些测试子集。

Tester flag	测试名称中应该包含什么	什么时候应该添加标志
--[no-]zookeeper	"zookeeper"或"replica"	测试使用来自ReplicatedMergeTree家族的表
--[no-]shard	"shard"或"distributed"或"global"	使用到127.0.0.2或类似的连接进行测试
--[no-]long	"long"或"deadlock"或"race"	测试运行时间超过60秒

Commit / push / 创建PR.

1) commit & push您的修改

```
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) 使用一个在推送过程中显示的链接，创建一个到master的PR
3) 调整PR标题和内容，在Changelog category (leave one)中保留
Build/Testing/Packaging Improvement，如果需要，请填写其余字段。

如何构建 ClickHouse 发布包

安装 Git 和 Pbuilder

```
sudo apt-get update  
sudo apt-get install git pbuilder debhelper lsb-release fakeroot sudo debian-archive-keyring debian-keyring
```

拉取 ClickHouse 源码

```
git clone --recursive https://github.com/ClickHouse/ClickHouse.git  
cd ClickHouse
```

运行发布脚本

```
./release
```

如何在开发过程中编译 ClickHouse

以下教程是在 Ubuntu Linux 中进行编译的示例。
通过适当的更改，它应该可以适用于任何其他的 Linux 发行版。
仅支持具有 x86_64、AArch64。对 Power9 的支持是实验性的。

安装 Git 和 CMake 和 Ninja

```
sudo apt-get install git cmake ninja-build
```

或cmake3而不是旧系统上的cmake。
或者在早期版本的系统中用 cmake3 替代 cmake

安装 Clang

On Ubuntu/Debian you can use the automatic installation script (check [official webpage](#))

```
sudo bash -c "$(wget -O - https://apt.llvm.org/llvm.sh)"
```

拉取 ClickHouse 源码

```
git clone --recursive git@github.com:ClickHouse/ClickHouse.git
## or: git clone --recursive https://github.com/ClickHouse/ClickHouse.git
cd ClickHouse
```

编译 ClickHouse

```
mkdir build
cd build
cmake ..
ninja
cd ..
```

若要创建一个执行文件，执行 `ninja clickhouse`。

这个命令会使得 `programs clickhouse` 文件可执行，您可以使用 `client` 或 `server` 参数运行。

如何编写 C++ 代码

一般建议

1. 以下是建议，而不是要求。
2. 如果你在修改代码，遵守已有的风格是有意义的。
3. 代码的风格需保持一致。一致的风格有利于阅读代码，并且方便检索代码。
4. 许多规则没有逻辑原因；它们是由既定的做法决定的。

格式化

1. 大多数格式化可以用 `clang-format` 自动完成。
2. 缩进是4个空格。配置开发环境，使得 TAB 代表添加四个空格。
3. 左右花括号需在单独的行。

```
inline void readBoolText(bool & x, ReadBuffer & buf)
{
    char tmp = '0';
    readChar(tmp, buf);
    x = tmp != '0';
}
```

4. 若整个方法体仅有一行 描述，则可以放到单独的行上。在花括号周围放置空格（除了行尾的空格）。

```
inline size_t mask() const { return buf_size() - 1; }
inline size_t place(HashValue x) const { return x & mask(); }
```

5. 对于函数。不要在括号周围放置空格。

```
void reinsert(const Value & x)
```

```
memcpy(&buf[place_value], &x, sizeof(x));
```

6. 在 `if`, `for`, `while` 和其他表达式中，在开括号前面插入一个空格（与函数声明相反）。

```
for (size_t i = 0; i < rows; i += storage.index_granularity)
```

7. 在二元运算符 (`+`, `-`, `*`, `/`, `%`, ...) 和三元运算符 `?:` 周围添加空格。

```
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. 若有换行，新行应该以运算符开头，并且增加对应的缩进。

```
if (elapsed_ns)
    message << "("
        << rows_read_on_server * 1000000000 / elapsed_ns << " rows/s., "
        << bytes_read_on_server * 1000.0 / elapsed_ns << " MB/s.) ";
```

9. 如果需要，可以在一行内使用空格来对齐。

```
dst.ClickLogID      = click.LogID;
dst.ClickEventID    = click.EventID;
dst.ClickGoodEvent  = click.GoodEvent;
```

10. 不要在 `.`, `->` 周围加入空格

如有必要，运算符可以包裹到下一行。在这种情况下，它前面的偏移量增加。

11. 不要使用空格来分开一元运算符 (`--`, `++`, `*`, `&`, ...) 和参数。

12. 在逗号后面加一个空格，而不是在之前。同样的规则也适合 `for` 循环中的分号。

13. 不要用空格分开 `[]` 运算符。

14. 在 `template <...>` 表达式中，在 `template` 和 `<` 中加入一个空格，在 `<` 后面或在 `>` 前面都不要有空格。

```
template <typename TKey, typename TValue>
struct AggregatedStatElement
{}
```

15. 在类和结构体中，`public`, `private` 以及 `protected` 同 `class/struct` 无需缩进，其他代码须缩进。

```
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. 如果对整个文件使用相同的 `namespace`，并且没有其他重要的东西，则 `namespace` 中不需要偏移量。

17. 在 `if`, `for`, `while` 中包裹的代码块中，若代码是一个单行的 `statement`，那么大括号是可选的。可以将 `statement` 放到一行中。这个规则同样适用于嵌套的 `if`, `for`, `while`, ...

但是如果内部 `statement` 包含大括号或 `else`，则外部块应该用大括号括起来。

```
/// Finish write.  
for (auto & stream : streams)  
    stream.second->finalize();
```

18. 行的末尾不应该包含空格。

19. 源文件应该用 UTF-8 编码。

20. 非ASCII字符可用于字符串文字。

```
<< ", " << (timer.elapsed() / chunks_stats.hits) << " µsec(hit.>";
```

21 不要在一行中写入多个表达式。

22. 将函数内部的代码段分组，并将它们与不超过一行的空行分开。

23. 将 函数，类用一个或两个空行分开。

24. `const` 必须写在类型名称之前。

```
//correct  
const char * pos  
const std::string & s  
//incorrect  
char const * pos
```

25. 声明指针或引用时，`*` 和 `&` 符号两边应该都用空格分隔。

```
//correct  
const char * pos  
//incorrect  
const char* pos  
const char *pos
```

26. 使用模板类型时，使用 `using` 关键字对它们进行别名（最简单的情况除外）。

换句话说，模板参数仅在 `using` 中指定，并且不在代码中重复。

`using`可以在本地声明，例如在函数内部。

```
//correct  
using FileStreams = std::map<std::string, std::shared_ptr<Stream>>;  
FileStreams streams;  
//incorrect  
std::map<std::string, std::shared_ptr<Stream>> streams;
```

27. 不要在一个语句中声明不同类型的多个变量。

```
//incorrect  
int x, *y;
```

28. 不要使用C风格的类型转换。

```
//incorrect  
std::cerr << (int)c << std::endl;  
//correct  
std::cerr << static_cast<int>(c) << std::endl;
```

29. 在类和结构中，组成员和函数分别在每个可见范围内。

30. 对于小类和结构，没有必要将方法声明与实现分开。

对于任何类或结构中的小方法也是如此。

对于模板化类和结构，不要将方法声明与实现分开（因为否则它们必须在同一个转换单元中定义）

31. 您可以将换行规则定在140个字符，而不是80个字符。

32. 如果不需要 postfix，请始终使用前缀增量/减量运算符。

```
for (Names::const_iterator it = column_names.begin(); it != column_names.end(); ++it)
```

评论

1. 请务必为所有非常重要的代码部分添加注释。

这是非常重要的。 编写注释可能会帮助您意识到代码不是必需的，或者设计错误。

```
/** 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. 注释可以尽可能详细。

3. 在他们描述的代码之前放置注释。 在极少数情况下，注释可以在代码之后，在同一行上。

```
/** 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. 注释应该只用英文撰写。

5. 如果您正在编写库，请在主头文件中包含解释它的详细注释。

6. 请勿添加无效的注释。 特别是，不要留下像这样的空注释：

```
/*
 * 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:
 */
```

这个示例来源于 <http://home.tamk.fi/~jaalto/course/coding-style/doc/unmaintainable-code/>。

7. 不要在每个文件的开头写入垃圾注释（作者，创建日期...）。

8. 单行注释用三个斜杆：///，多行注释以/**开始。这些注释会当做文档。

注意：您可以使用 Doxygen 从这些注释中生成文档。但是通常不使用 Doxygen，因为在 IDE 中导航代码更方便。

9. 多行注释的开头和结尾不得有空行（关闭多行注释的行除外）。

10. 要注释掉代码，请使用基本注释，而不是“文档”注释。

11. 在提交之前删除代码的无效注释部分。

12. 不要在注释或代码中使用亵渎语言。

13. 不要使用大写字母。不要使用过多的标点符号。

```
/// WHAT THE FAIL???
```

14. 不要使用注释来制作分隔符。

```
///*****
```

15. 不要在注释中开始讨论。

```
/// Why did you do this stuff?
```

16. 没有必要在块的末尾写一条注释来描述它的含义。

```
/// for
```

姓名

1. 在变量和类成员的名称中使用带下划线的小写字母。

```
size_t max_block_size;
```

2. 对于函数（方法）的名称，请使用以小写字母开头的驼峰标识。

```
std::string getName() const override { return "Memory"; }
```

3. 对于类（结构）的名称，使用以大写字母开头的驼峰标识。接口名称用 I 前缀。

```
class StorageMemory : public IStorage
```

4. using 的命名方式与类相同，或者以 `_t` 命名。

5. 模板类型参数的名称：在简单的情况下，使用 `T; T1, T2`。

对于更复杂的情况，要么遵循类名规则，要么添加前缀 `T`。

```
template <typename TKey, typename TValue>
struct AggregatedStatElement
```

6. 模板常量参数的名称：遵循变量名称的规则，或者在简单的情况下使用 `N`。

```
template <bool without_www>
struct ExtractDomain
```

7. 对于抽象类（接口），用 I 前缀。

```
class IBlockInputStream
```

8. 如果在本地使用变量，则可以使用短名称。

在所有其他情况下，请使用能描述含义的名称。

```
bool info_successfully_loaded = false;
```

9. define 和全局常量的名称使用全大写带下划线的形式，如 `ALL_CAPS`。

```
##define MAX_SRC_TABLE_NAMES_TO_STORE 1000
```

10. 文件名应使用与其内容相同的样式。

如果文件包含单个类，则以与该类名称相同的方式命名该文件（CamelCase）。

如果文件包含单个函数，则以与函数名称相同的方式命名文件（camelCase）。

11. 如果名称包含缩写，则：

- 对于变量名，缩写应使用小写字母 `mysql_connection`（不是 `mySQL_connection`）。
- 对于类和函数的名称，请将大写字母保留在缩写 `MySQLConnection`（不是 `MySqlConnection`）。

12. 仅用于初始化类成员的构造方法参数的命名方式应与类成员相同，但最后使用下划线。

```
FileQueueProcessor(
    const std::string & path_,
    const std::string & prefix_,
    std::shared_ptr<FileHandler> handler_
) : path(path_),
prefix(prefix_),
handler(handler_),
log(&Logger::get("FileQueueProcessor"))
{ }
```

如果构造函数体中未使用该参数，则可以省略下划线后缀。

13. 局部变量和类成员的名称没有区别（不需要前缀）。

```
timer (not m_timer)
```

14. 对于 `enum` 中的常量，请使用带大写字母的驼峰标识。`ALL_CAPS` 也可以接受。如果 `enum` 是非本地的，请使用 `enum class`。

```
enum class CompressionMethod
{
    QuickLZ = 0,
    LZ4      = 1,
};
```

15. 所有名字必须是英文。不允许音译俄语单词。

```
not Stroka
```

16. 缩写须是众所周知的（当您可以在维基百科或搜索引擎中轻松找到缩写的含义时）。

```
`AST`, `SQL`.  
Not `NVDH` (some random letters)
```

如果缩短版本是常用的，则可以接受不完整的单词。

如果旁边有注释包含全名，您也可以使用缩写。

17. C++ 源码文件名称必须为 `.cpp` 拓展名。头文件必须为 `.h` 拓展名。

如何编写代码

1. 内存管理。

手动内存释放 (`delete`) 只能在库代码中使用。

在库代码中，`delete` 运算符只能在析构函数中使用。

在应用程序代码中，内存必须由拥有它的对象释放。

示例：

- 最简单的方法是将对象放在堆栈上，或使其成为另一个类的成员。
- 对于大量小对象，请使用容器。
- 对于自动释放少量在堆中的对象，可以用 `shared_ptr/unique_ptr`。

2. 资源管理。

使用 `RAII` 以及查看以上说明。

3. 错误处理。

在大多数情况下，您只需要抛出一个异常，而不需要捕获它（因为 `RAII`）。

在离线数据处理应用程序中，通常可以接受不捕获异常。

在处理用户请求的服务器中，捕获连接处理程序顶层的异常通常就足够了。

在线程函数中，你应该在 `join` 之后捕获并保留所有异常以在主线程中重新抛出它们。

```
/// 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();
```

不处理就不要隐藏异常。永远不要盲目地把所有异常都记录到日志中。

```
//Not correct
catch (...) {}
```

如果您需要忽略某些异常，请仅针对特定异常执行此操作并重新抛出其余异常。

```
catch (const DB::Exception & e)
{
    if (e.code() == ErrorCodes::UNKNOWN_AGGREGATE_FUNCTION)
        return nullptr;
    else
        throw;
}
```

当使用具有返回码或 `errno` 的函数时，请始终检查结果并在出现错误时抛出异常。

```
if (0 != close(fd))
    throwErrorFromErrno("Cannot close file " + file_name, ErrorCodes::CANNOT_CLOSE_FILE);
```

不要使用断言。

4. 异常类型。

不需要在应用程序代码中使用复杂的异常层次结构。系统管理员应该可以理解异常文本。

5. 从析构函数中抛出异常。

不建议这样做，但允许这样做。

按照以下选项：

- 创建一个函数（`done()` 或 `finalize()`），它将提前完成所有可能导致异常的工作。如果调用了该函数，则稍后在析构函数中应该没有异常。
- 过于复杂的任务（例如通过网络发送消息）可以放在单独的方法中，类用户必须在销毁之前调用它们。

- 如果析构函数中存在异常，则最好记录它而不是隐藏它（如果 `logger` 可用）。
- 在简单的应用程序中，依赖于 `std::terminate`（对于 C++ 11 中默认情况下为 `noexcept` 的情况）来处理异常是可以接受的。

6. 匿名代码块。

您可以在单个函数内创建单独的代码块，以使某些变量成为局部变量，以便在退出块时调用析构函数。

```
Block block = data.in->read();  
  
{  
    std::lock_guard<std::mutex> lock(mutex);  
    data.ready = true;  
    data.block = block;  
}  
  
ready_any.set();
```

7. 多线程。

在离线数据处理程序中：

- 尝试在单个 CPU 核心上获得最佳性能。然后，您可以根据需要并行化代码。

在服务端应用中：

- 使用线程池来处理请求。此时，我们还没有任何需要用户空间上下文切换的任务。

`Fork` 不用于并行化。

8. 同步线程。

通常可以使不同的线程使用不同的存储单元（甚至更好：不同的缓存线），并且不使用任何线程同步（除了 `joinAll`）。

如果需要同步，在大多数情况下，在 `lock_guard` 下使用互斥量就足够了。

在其他情况下，使用系统同步原语。不要使用忙等待。

仅在最简单的情况下才应使用原子操作。

除非是您的主要专业领域，否则不要尝试实施无锁数据结构。

9. 指针和引用。

大部分情况下，请用引用。

10. 常量。

使用 `const` 引用、指针，指向常量、`const_iterator` 和 `const` 方法。

将 `const` 视为默认值，仅在必要时使用非 `const`。

当按值传递变量时，使用 `const` 通常没有意义。

11. 无符号。

必要时使用 `unsigned`。

12. 数值类型。

使用 `UInt8`, `UInt16`, `UInt32`, `UInt64`, `Int8`, `Int16`, `Int32` 和 `Int64`，同样还有 `size_t`, `ssize_t` 和 `ptrdiff_t`。

不要使用这些类型：`signed / unsigned long`, `long long`, `short`, `signed / unsigned char`, `char`。

13. 参数传递。

通过引用传递复杂类型（包括 `std::string`）。

如果函数中传递堆中创建的对象，则使参数类型为 `shared_ptr` 或者 `unique_ptr`。

14. 返回值

大部分情况下使用 `return`。不要使用 `return std::move(res)`。

如果函数在堆上分配对象并返回它，请使用 `shared_ptr` 或 `unique_ptr`。

在极少数情况下，您可能需要通过参数返回值。在这种情况下，参数应该是引用传递的。

```
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. 命名空间。

没有必要为应用程序代码使用单独的 `namespace`。

小型库也不需要这个。

对于中大型库，须将所有代码放在 `namespace` 中。

在库的 `.h` 文件中，您可以使用 `namespace detail` 来隐藏应用程序代码不需要的实现细节。

在 `.cpp` 文件中，您可以使用 `static` 或匿名命名空间来隐藏符号。

同样 `namespace` 可用于 `enum` 以防止相应的名称落入外部 `namespace`（但最好使用 `enum class`）。

16. 延迟初始化。

如果初始化需要参数，那么通常不应该编写默认构造函数。

如果稍后您需要延迟初始化，则可以添加将创建无效对象的默认构造函数。或者，对于少量对象，您可以使用 `shared_ptr` / `unique_ptr`。

```
Loader(DB::Connection * connection_, const std::string & query, size_t max_block_size_);  
  
/// For deferred initialization  
Loader() {}
```

17. 虚函数。

如果该类不是用于多态使用，则不需要将函数设置为虚拟。这也适用于析构函数。

18. 编码。

在所有情况下使用 UTF-8 编码。使用 `std::string` 和 `char*`。不要使用 `std::wstring` 和 `wchar_t`。

19. 日志。

请参阅代码中的示例。

在提交之前，删除所有无意义和调试日志记录，以及任何其他类型的调试输出。

应该避免循环记录日志，即使在 Trace 级别也是如此。

日志必须在任何日志记录级别都可读。

在大多数情况下，只应在应用程序代码中使用日志记录。

日志消息必须用英文写成。

对于系统管理员来说，日志最好是可以理解的。

不要在日志中使用亵渎语言。

在日志中使用UTF-8编码。在极少数情况下，您可以在日志中使用非ASCII字符。

20. 输入-输出。

不要使用 `iostreams` 在对应用程序性能至关重要的内部循环中（并且永远不要使用 `stringstream`）。

使用 `DB/IO` 库替代。

21. 日期和时间。

参考 `DateLUT` 库。

22. 引入头文件。

一直用 `#pragma once` 而不是其他宏。

23. using 语法

`using namespace` 不会被使用。您可以使用特定的 `using`。但是在类或函数中使它成为局部的。

24. 不要使用 trailing return type 为必要的功能。

```
auto f() -> void
```

25. 声明和初始化变量。

```
//right way
std::string s = "Hello";
std::string s{"Hello"};

//wrong way
auto s = std::string{"Hello"};
```

26. 对于虚函数，在基类中编写 `virtual`，但在后代类中写 `override` 而不是 `virtual`。

没有用到的 C++ 特性。

1. 不使用虚拟继承。

2. 不使用 C++03 中的异常标准。

平台

1. 我们为特定平台编写代码。

但在其他条件相同的情况下，首选跨平台或可移植代码。

2. 语言：C++20.

3. 编译器：`clang`。此时（2021年03月），代码使用11版编译。（它也可以使用`gcc`编译 but it is not suitable for production）

使用标准库（`libc++`）。

4. 操作系统：Linux Ubuntu，不比 Precise 早。

5. 代码是为x86_64 CPU架构编写的。

CPU指令集是我们服务器中支持的最小集合。目前，它是SSE 4.2。

6. 使用 `-Wall -Wextra -Werror` 编译参数。

7. 对所有库使用静态链接，除了那些难以静态连接的库（参见 `ldd` 命令的输出）。

8. 使用发布的设置来开发和调试代码。

工具

1. KDevelop 是一个好的 IDE.

2. 调试可以使用 `gdb`，`valgrind (memcheck)`，`strace`，`-fsanitize=...`，或 `tcmalloc_minimal_debug`.

3. 对于性能分析，使用 `Linux Perf`，`valgrind (callgrind)`，或者 `strace -cf`。

4. 源代码用 `Git` 作版本控制。

5. 使用 `CMake` 构建。

6. 程序的发布使用 `deb` 安装包。

7. 提交到 master 分支的代码不能破坏编译。

虽然只有选定的修订被认为是可行的。

8. 尽可能经常地进行提交，即使代码只是部分准备好了。

为了这种目的可以创建分支。

如果您的代码在 `master` 分支中尚不可构建，在 `push` 之前需要将其从构建中排除。您需要在几天内完成或删除它。

9. 对于非一般的更改，请使用分支并在服务器上发布它们。

10. 未使用的代码将从 `repo` 中删除。

库

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.

一般建议

1. 尽可能精简代码。

2. 尝试用最简单的方式实现。

3. 在你知道代码是如何工作以及内部循环如何运作之前，不要编写代码。

4. 在最简单的情况下，使用 `using` 而不是类或结构。

5. 如果可能，不要编写复制构造函数，赋值运算符，析构函数（虚拟函数除外，如果类包含至少一个虚函数），移动构造函数或移动赋值运算符。换句话说，编译器生成的函数必须正常工作。您可以使用 `default`。

6. 鼓励简化代码。尽可能减小代码的大小。

其他建议

1. 从 `stddef.h` 明确指定 `std::` 的类型。

不推荐。换句话说，我们建议写 `size_t` 而不是 `std::size_t`，因为它更短。

也接受添加 `std::`。

2. 为标准C库中的函数明确指定 `std::`

不推荐。换句话说，写 `memcpy` 而不是 `std::memcpy`。

原因是有一些类似的非标准功能，例如 `memmem`。我们偶尔会使用这些功能。`namespace std` 中不存在这些函数。

如果你到处都写 `std::memcpy` 而不是 `memcpy`，那么没有 `std::` 的 `memmem` 会显得很奇怪。

不过，如果您愿意，仍然可以使用 `std::`。

3. 当标准C++库中提供相同的函数时，使用C中的函数。

如果它更高效，这是可以接受的。

例如，使用 `memcpy` 而不是 `std::copy` 来复制大块内存。

4. 函数的多行参数。

允许以下任何包装样式：

```
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)
```

碌莽碌release拢.0755-88888888

ClickHouse版本v20.3.4.10,2020-03-20

错误修复

- 此版本还包含20.1.8.41的所有错误修复
- 修复丢失 `rows_before_limit_at_least` 用于通过http进行查询（使用处理器管道）。这修复 #9730. #9757 (尼古拉*科切托夫)

ClickHouse释放v20.3.3.6,2020-03-17

错误修复

- 此版本还包含20.1.7.38的所有错误修复
- 修复复制中的错误，如果用户在以前的版本上执行了突变，则不允许复制工作。这修复 #9645. #9652 (阿利沙平)。它使版本20.3再次向后兼容。
- 添加设置 `use_compact_format_in_distributed_parts_names` 它允许写文件 `INSERT` 查询到 `Distributed` 表格格式更紧凑。这修复 #9647. #9653 (阿利沙平)。它使版本20.3再次向后兼容。

ClickHouse版本v20.3.2.1,2020-03-12

向后不兼容的更改

- 修正了这个问题 `file name too long` 当发送数据 `Distributed` 大量副本的表。修复了服务器日志中显示副本凭据的问题。磁盘上的目录名格式已更改为 `[shard{shard_index}][_replica{replica_index}]]`。#8911 (米哈伊尔*科罗托夫) 升级到新版本后，您将无法在没有人工干预的情况下降级，因为旧的服务器版本无法识别新的目录格式。如果要降级，则必须手动将相应的目录重命名为旧格式。仅当您使用了异步时，此更改才相关 `INSERTS` 到 `Distributed` 桌子 在版本20.3.3中，我们将介绍一个设置，让您逐渐启用新格式。
- 更改了mutation命令的复制日志条目的格式。在安装新版本之前，您必须等待旧的突变处理。
- 实现简单的内存分析器，将堆栈跟踪转储到 `system.trace_log` 超过软分配限制的每N个字节 #8765 (伊万) #9472 (阿列克谢-米洛维多夫) 列 `system.trace_log` 从改名 `timer_type` 到 `trace_type`。这将需要改变第三方性能分析和flamegraph处理工具。
- 在任何地方使用操作系统线程id，而不是内部线程编号。这修复 #7477 老 `clickhouse-client` 无法接收从服务器发送的日志，当设置 `send_logs_level` 已启用，因为结构化日志消息的名称和类型已更改。另一方面，不同的服务器版本可以相互发送不同类型的日志。当你不使用 `send_logs_level` 设置，你不应该关心。#8954 (阿列克谢-米洛维多夫)
- 删除 `indexHint` 功能 #9542 (阿列克谢-米洛维多夫)
- 删除 `findClusterIndex`, `findClusterValue` 功能。这修复 #8641. 如果您正在使用这些功能，请发送电子邮件至 `clickhouse-feedback@yandex-team.com` #9543 (阿列克谢-米洛维多夫)
- 现在不允许创建列或添加列 `SELECT` 子查询作为默认表达式。#9481 (阿利沙平)
- 需要联接中的子查询的别名。#9274 (Artem Zuikov)
- 改进 `ALTER MODIFY/ADD` 查询逻辑。现在你不能 `ADD` 不带类型的列, `MODIFY` 默认表达式不改变列的类型和 `MODIFY type` 不会丢失默认表达式值。修复 #8669. #9227 (阿利沙平)
- 要求重新启动服务器以应用日志记录配置中的更改。这是一种临时解决方法，可以避免服务器将日志记录到已删除的日志文件中的错误（请参阅 #8696). #8707 (Alexander Kuzmenkov)
- 设置 `experimental_use_processors` 默认情况下启用。此设置允许使用新的查询管道。这是内部重构，我们期望没有明显的变化。如果您将看到任何问题，请将其设置为返回零。#8768 (阿列克谢-米洛维多夫)

新功能

- 添加 Avro 和 AvroConfluent 输入/输出格式 #8571 (安德鲁Onyshchuk) #8957 (安德鲁Onyshchuk) #8717 (阿列克谢-米洛维多夫)
- 过期密钥的多线程和非阻塞更新 cache 字典 (可选的权限读取旧的) 。 #8303 (尼基塔*米哈伊洛夫)
- 添加查询 ALTER ... MATERIALIZE TTL 它运行突变，强制通过TTL删除过期的数据，并重新计算所有部分有关ttl的元信息。 #8775 (安东*波波夫)
- 如果需要，从HashJoin切换到MergeJoin (在磁盘上 #9082 (Artem Zuikov))
- 已添加 MOVE PARTITION 命令 ALTER TABLE #4729 #6168 (纪尧姆*塔瑟里)
- 动态地从配置文件重新加载存储配置。 #8594 (Vladimir Chebotarev)
- 允许更改 storage_policy 为了不那么富有的人。 #8107 (Vladimir Chebotarev)
- 增加了对s3存储和表功能的glob/通配符的支持。 #8851 (Vladimir Chebotarev)
- 执行 bitAnd, bitOr, bitXor, bitNot 为 FixedString(N) 数据类型。 #9091 (纪尧姆*塔瑟里)
- 添加功能 bitCount. 这修复 #8702. #8708 (阿列克谢-米洛维多夫) #8749 (ikopylov)
- 添加 generateRandom 表函数生成具有给定模式的随机行。允许用数据填充任意测试表。 #8994 (Ilya Yatsishin)
- JSONEachRowFormat : 当对象包含在顶层数组中时，支持特殊情况。 #8860 (克鲁格洛夫*帕维尔)
- 现在可以创建一个列 DEFAULT 取决于默认列的表达式 ALIAS 表达。 #9489 (阿利沙平)
- 允许指定 --limit 超过源数据大小 clickhouse-obfuscator. 数据将以不同的随机种子重复。 #9155 (阿列克谢-米洛维多夫)
- 已添加 groupArraySample 功能 (类似于 groupArray) 与 reservior采样算法。 #8286 (阿莫斯鸟)
- 现在，您可以监视更新队列的大小 cache/complex_key_cache 通过系统指标字典。 #9413 (尼基塔*米哈伊洛夫)
- 允许使用CRLF作为CSV输出格式的行分隔符与设置 output_format_csv_crlf_end_of_line 设置为1 #8934 #8935 #8963 (米哈伊尔*科罗托夫)
- 实现的更多功能 H3 API: h3GetBaseCell, h3HexAreaM2, h3IndexesAreNeighbors, h3ToChildren, h3ToString 和 stringToH3 #8938 (Nico Mandery)
- 引入新设置: max_parser_depth 控制最大堆栈大小并允许大型复杂查询。这修复 #6681 和 #7668. #8647 (马克西姆 *斯米尔诺夫)
- 添加设置 force_optimize_skip_unused_shards 如果无法跳过未使用的分片，则设置为抛出 #8805 (Azat Khuzhin)
- 允许配置多个磁盘/卷用于存储数据发送 Distributed 发动机 #8756 (Azat Khuzhin)
- 支持存储策略 (<tmp_policy>) 用于存储临时数据。 #8750 (Azat Khuzhin)
- 已添加 X-ClickHouse-Exception-Code 如果在发送数据之前引发异常，则设置的HTTP头。这实现了 #4971. #8786 (米哈伊尔*科罗托夫)
- 添加功能 ifNotFinite. 这只是一个句法糖: ifNotFinite(x, y) = isFinite(x) ? x : y. #8710 (阿列克谢-米洛维多夫)
- 已添加 last_successful_update_time 列中 system.dictionaries 表 #9394 (尼基塔*米哈伊洛夫)
- 添加 blockSerializedSize 功能 (磁盘大小不压缩) #8952 (Azat Khuzhin)
- 添加功能 moduloOrZero #9358 (hczi)

- 添加系统表 `system.zeros` 和 `system.zeros_mt` 以及故事功能 `zeros()` 和 `zeros_mt()`. 表 (和表函数) 包含具有名称的单列 `zero` 和类型 `UInt8`. 此列包含零。为了测试目的，需要它作为生成许多行的最快方法。这修复 #6604 #9593 (尼古拉*科切托夫)

实验特点

- 添加新的紧凑格式的部件 MergeTree-家庭表中的所有列都存储在一个文件中。它有助于提高小型和频繁插入的性能。旧的格式 (每列一个文件) 现在被称为 `wide`。数据存储格式由设置控制 `min_bytes_for_wide_part` 和 `min_rows_for_wide_part`. #8290 (安东*波波夫)
- 支持S3存储 `Log`, `TinyLog` 和 `StripeLog` 桌子 #8862 (帕维尔*科瓦连科)

错误修复

- 修正了日志消息中不一致的空格。 #9322 (阿列克谢-米洛维多夫)
- 修复在创建表时将未命名元组数组展平为嵌套结构的错误。 #8866 (achulkov2)
- 修复了以下问题 “`Too many open files`” 如果有太多的文件匹配 `glob` 模式可能会发生错误 `File` 表或 `file` 表功能。现在文件懒洋洋地打开。这修复 #8857 #8861 (阿列克谢-米洛维多夫)
- 删除临时表现在只删除临时表。 #8907 (维塔利*巴拉诺夫)
- 当我们关闭服务器或分离/附加表时删除过时的分区。 #8602 (纪尧姆*塔瑟里)
- 默认磁盘如何计算可用空间 `data` 子目录。修复了可用空间量计算不正确的问题，如果 `data` 目录被安装到一个单独的设备 (罕见的情况)。这修复 #7441 #9257 (米哈伊尔*科罗托夫)
- 允许逗号 (交叉) 与 `IN ()` 内部连接。 #9251 (Artem Zuikov)
- 如果在 `WHERE` 部分中有 `[NOT]LIKE` 运算符，则允许将 `CROSS` 重写为 `INNER JOIN`。 #9229 (Artem Zuikov)
- 修复后可能不正确的结果 `GROUP BY` 启用设置 `distributed_aggregation_memory_efficient`. 修复 #9134. #9289 (尼古拉*科切托夫)
- 找到的键在缓存字典的指标中被计为错过。 #9411 (尼基塔*米哈伊洛夫)
- 修复引入的复制协议不兼容 #8598. #9412 (阿利沙平)
- 在固定的竞争条件 `queue_task_handle` 在启动 `ReplicatedMergeTree` 桌子 #9552 (阿列克谢-米洛维多夫)
- 令牌 `NOT` 没有工作 `SHOW TABLES NOT LIKE` 查询 #8727 #8940 (阿列克谢-米洛维多夫)
- 添加范围检查功能 `h3EdgeLengthM`. 如果没有这个检查，缓冲区溢出是可能的。 #8945 (阿列克谢-米洛维多夫)
- 修复了多个参数 (超过10) 的三元逻辑运算批量计算中的错误。 #8718 (亚历山大*卡扎科夫)
- 修复PREWHERE优化的错误，这可能导致段错误或 `Inconsistent number of columns got from MergeTreeRangeReader` 例外。 #9024 (安东*波波夫)
- 修复意外 `Timeout exceeded while reading from socket` 异常，在实际超时之前以及启用查询探查器时，在安全连接上随机发生。还添加 `connect_timeout_with_failover_secure_ms` 设置 (默认100ms)，这是类似于 `connect_timeout_with_failover_ms`，但用于安全连接 (因为SSL握手比普通TCP连接慢) #9026 (tavplubix)
- 修复突变最终确定的错误，当突变可能处于以下状态时 `parts_to_do=0` 和 `is_done=0`. #9022 (阿利沙平)
- 使用新的任何连接逻辑 `partial_merge_join` 设置。有可能使 `ANY|ALL|SEMI LEFT` 和 `ALL INNER` 加入与 `partial_merge_join=1` 现在 #8932 (Artem Zuikov)
- Shard现在将从发起者获得的设置夹到 shard 的 `constraints`，而不是抛出异常。此修补程序允许将查询发送到具有另一个约束的分片。 #9447 (维塔利*巴拉诺夫)
- 修正了内存管理问题 `MergeTreeReadPool`. #8791 (Vladimir Chebotarev)

- 修复 `toDecimal*OrNull()` 使用字符串调用时的函数系列 e. 修复 #8312 #8764 (Artem Zuikov)
- 请确保 `FORMAT Null` 不向客户端发送数据。 #8767 (Alexander Kuzmenkov)
- 修复时间戳中的错误 `LiveViewBlockInputStream` 不会更新。 `LIVE VIEW` 是一个实验特征。 #8644 (vxider) #8625 (vxider)
- 固定 `ALTER MODIFY TTL` 不允许删除旧ttl表达式的错误行为。 #8422 (Vladimir Chebotarev)
- 修复了MergeTreeIndexSet中的UBSan报告。这修复 #9250 #9365 (阿列克谢-米洛维多夫)
- 固定的行为 `match` 和 `extract` 当干草堆有零字节的函数。当干草堆不变时，这种行为是错误的。这修复 #9160 #9163 (阿列克谢-米洛维多夫) #9345 (阿列克谢-米洛维多夫)
- 避免从apache Avro第三方库中的析构函数抛出。 #9066 (安德鲁Onyshchuk)
- 不要提交从轮询的批次 `Kafka` 部分，因为它可能会导致数据漏洞。 #8876 (filimonov)
- 修复 `joinGet` 使用可为空的返回类型。 <https://github.com/ClickHouse/ClickHouse/issues/8919> #9014 (阿莫斯鸟)
- 修复压缩时的数据不兼容 `T64` 编解ec #9016 (Artem Zuikov) 修复数据类型id `T64` 在受影响的版本中导致错误(de)压缩的压缩编解ec。 #9033 (Artem Zuikov)
- 添加设置 `enable_early_constant_folding` 并禁用它在某些情况下，导致错误。 #9010 (Artem Zuikov)
- 使用VIEW修复下推谓词优化器并启用测试 #9011 (张冬)
- 修复段错误 `Merge` 表，从读取时可能发生 `File` 储存 #9387 (tavplubix)
- 添加了对存储策略的检查 `ATTACH PARTITION FROM`, `REPLACE PARTITION`, `MOVE TO TABLE`. 否则，它可以使部分数据重新启动后无法访问，并阻止ClickHouse启动。 #9383 (Vladimir Chebotarev)
- 修复改变，如果有TTL设置表。 #8800 (安东*波波夫)
- 修复在以下情况下可能发生的竞争条件 `SYSTEM RELOAD ALL DICTIONARIES` 在某些字典被修改/添加/删除时执行。 #8801 (维塔利*巴拉诺夫)
- 在以前的版本 `Memory` 数据库引擎使用空数据路径，因此在以下位置创建表 `path directory` (e.g. `/var/lib/clickhouse/`), `not in data directory of database` (e.g. `/var/lib/clickhouse/db_name`). #8753 (tavplubix)
- 修复了关于缺少默认磁盘或策略的错误日志消息。 #9530 (Vladimir Chebotarev)
- 修复数组类型的**bloom_filter**索引的not(has())。 #9407 (achimbab)
- 允许表中的第一列 `Log` 引擎是别名 #9231 (伊万)
- 从读取时修复范围的顺序 `MergeTree` 表中的一个线程。它可能会导致例外 `MergeTreeRangeReader` 或错误的查询结果。 #9050 (安东*波波夫)
- 赖眉露>> `reinterpretAsFixedString` 返回 `FixedString` 而不是 `String`. #9052 (安德鲁Onyshchuk)
- 避免极少数情况下，当用户可以得到错误的错误消息 (`Success` 而不是详细的错误描述)。 #9457 (阿列克谢-米洛维多夫)
- 使用时不要崩溃 `Template` 使用空行模板格式化。 #8785 (Alexander Kuzmenkov)
- 系统表的元数据文件可能在错误的位置创建 #8653 (tavplubix) 修复 #8581.
- 修复缓存字典中`exception_ptr`上的数据竞赛 #8303. #9379 (尼基塔*米哈伊洛夫)
- 不要为查询引发异常 `ATTACH TABLE IF NOT EXISTS`. 以前它是抛出，如果表已经存在，尽管 `IF NOT EXISTS` 条款 #8967 (安东*波波夫)

- 修复了异常消息中丢失的关闭paren。 #8811 (阿列克谢-米洛维多夫)
- 避免消息 Possible deadlock avoided 在clickhouse客户端在交互模式下启动。 #9455 (阿列克谢-米洛维多夫)
- 修复了base64编码值末尾填充格式错误的问题。更新base64库。这修复 #9491，关闭 #9492 #9500 (阿列克谢-米洛维多夫)
- 防止丢失数据 Kafka 在极少数情况下，在读取后缀之后但在提交之前发生异常。修复 #9378 #9507 (filimonov)
- 在固定的异常 DROP TABLE IF EXISTS #8663 (尼基塔*瓦西列夫)
- 修复当用户尝试崩溃 ALTER MODIFY SETTING 对于老格式化 MergeTree 表引擎家族. #9435 (阿利沙平)
- 支持在JSON相关函数中不适合Int64的UInt64号码。更新SIMDJSON掌握。这修复 #9209 #9344 (阿列克谢-米洛维多夫)
- 当使用非严格单调函数索引时，固定执行反转谓词。 #9223 (亚历山大*卡扎科夫)
- 不要试图折叠 IN 常量在 GROUP BY #8868 (阿莫斯鸟)
- 修复bug ALTER DELETE 突变导致索引损坏。这修复 #9019 和 #8982. 另外修复极其罕见的竞争条件 ReplicatedMergeTree ALTER 查询。 #9048 (阿利沙平)
- 当设置 compile_expressions 被启用，你可以得到 unexpected column 在 LLVMExecutableFunction 当我们使用 Nullable 类型 #8910 (纪尧姆*塔瑟里)
- 多个修复 Kafka 引擎：1) 修复在消费者组重新平衡期间出现的重复项。2)修复罕见 ‘holes’ 当数据从一个轮询的几个分区轮询并部分提交时出现（现在我们总是处理/提交整个轮询的消息块）。3) 通过块大小修复刷新（在此之前，只有超时刷新才能正常工作）。4) 更好的订阅程序（与分配反馈）。5) 使测试工作得更快（默认时间间隔和超时）。由于数据之前没有被块大小刷新（根据文档），pr可能会导致默认设置的性能下降（由于更频繁和更小的刷新不太理想）。如果您在更改后遇到性能问题-请增加 kafka_max_block_size 在表中的更大的值（例如 CREATE TABLE ...Engine=Kafka ... SETTINGS ... kafka_max_block_size=524288). 修复 #7259 #8917 (filimonov)
- 修复 Parameter out of bound 在PREWHERE优化之后的某些查询中出现异常。 #8914 (Baudouin Giard)
- 修正了函数参数混合常量的情况 arrayZip. #8705 (阿列克谢-米洛维多夫)
- 执行时 CREATE 查询，在存储引擎参数中折叠常量表达式。将空数据库名称替换为当前数据库。修复 #6508, #3492 #9262 (tavplubix)
- 现在不可能创建或添加具有简单循环别名的列，如 a DEFAULT b, b DEFAULT a. #9603 (阿利沙平)
- 修正了双重移动可能会损坏原始部分的错误。这是相关的，如果你使用 ALTER TABLE MOVE #8680 (Vladimir Chebotarev)
- 允许 interval 用于正确解析的标识符，而无需反引号。当一个查询不能被执行，即使固定的问题 interval 标识符用反引号或双引号括起来。这修复 #9124. #9142 (阿列克谢-米洛维多夫)
- 修正了模糊测试和不正确的行为 bitTestAll/bitTestAny 功能。 #9143 (阿列克谢-米洛维多夫)
- 修复可能的崩溃/错误的行数 LIMIT n WITH TIES 当有很多行等于第n行时。 #9464 (tavplubix)
- 使用enabled编写的部件修复突变 insert_quorum. #9463 (阿利沙平)
- 修复数据竞赛破坏 Poco::HTTPServer. 当服务器启动并立即关闭时，可能会发生这种情况。 #9468 (安东*波波夫)
- 修复运行时显示误导性错误消息的错误 SHOW CREATE TABLE a_table_that_does_not_exist #8899 (achulkov2)
- 固定 Parameters are out of bound 例外在一些罕见的情况下，当我们在一个常数 SELECT 条款时，我们有一个 ORDER BY 和一个 LIMIT 条款 #8892 (纪尧姆*塔瑟里)
- 修复突变定稿，当已经完成突变可以有状态 is_done=0. #9217 (阿利沙平)

- 防止执行 `ALTER ADD INDEX` 对于旧语法的MergeTree表，因为它不起作用。 #8822 (米哈伊尔*科罗托夫)
- 在服务器启动时不要访问表，这 `LIVE VIEW` 取决于，所以服务器将能够启动。也删除 `LIVE VIEW` 分离时的依赖关系 `LIVE VIEW`. `LIVE VIEW` 是一个实验特征。 #8824 (tavplubix)
- 修复可能的段错误 `MergeTreeRangeReader`，同时执行 `PREWHERE`. #9106 (安东*波波夫)
- 修复与列TTL可能不匹配的校验和。 #9451 (安东*波波夫)
- 修正了一个错误，当部分没有被移动的情况下，只有一个卷的TTL规则在后台。 #8672 (Vladimir Chebotarev)
- 修正了这个问题 `Method createColumn() is not implemented for data type Set` 这修复 #7799. #8674 (阿列克谢-米洛维多夫)
- 现在我们将尝试更频繁地完成突变。 #9427 (阿利沙平)
- 修复 `intDiv` 减一个常数 #9351 (hcz)
- 修复可能的竞争条件 `BlockIO`. #9356 (尼古拉*科切托夫)
- 修复尝试使用/删除时导致服务器终止的错误 `Kafka` 使用错误的参数创建的表。 #9513 (filimonov)
- 增加了解决方法，如果操作系统返回错误的结果 `timer_create` 功能。 #8837 (阿列克谢-米洛维多夫)
- 在使用固定错误 `min_marks_for_seek` 参数。修复了分布式表中没有分片键时的错误消息，并且我们尝试跳过未使用的分片。 #8908 (Azat Khuzhin)

改进

- 执行 `ALTER MODIFY/DROP` 对突变的顶部查询 `ReplicatedMergeTree*` 引擎家族。现在 `ALTERS` 仅在元数据更新阶段阻止，之后不阻止。 #8701 (阿利沙平)
- 添加重写交叉到内部连接的能力 `WHERE` 包含未编译名称的部分。 #9512 (Artem Zuikov)
- 起眉露>> `SHOW TABLES` 和 `SHOW DATABASES` 查询支持 `WHERE` 表达式和 `FROM/IN` #9076 (sundyli)
- 添加了一个设置 `deduplicate_blocks_in_dependent_materialized_views`. #9070 (urykhy)
- 在最近的变化之后，MySQL客户端开始以十六进制打印二进制字符串，从而使它们不可读 (#9032). ClickHouse中的解决方法是将字符串列标记为UTF-8，这并不总是如此，但通常是这种情况。 #9079 (尤里*巴拉诺夫)
- 添加对字符串和`FixedString`键的支持 `sumMap` #8903 (Baudouin Giard)
- 支持`SummingMergeTree`地图中的字符串键 #8933 (Baudouin Giard)
- 即使线程已抛出异常，也向线程池发送线程终止信号 #8736 (丁香飞)
- 允许设置 `query_id` 在 `clickhouse-benchmark` #9416 (安东*波波夫)
- 不要让奇怪的表达 `ALTER TABLE ... PARTITION partition` 查询。这个地址 #7192 #8835 (阿列克谢-米洛维多夫)
- 表 `system.table_engines` 现在提供有关功能支持的信息（如 `supports_ttl` 或 `supports_sort_order`）. #8830 (Max Akhmedov)
- 启用 `system.metric_log` 默认情况下。它将包含具有`ProfileEvents`值的行，`CurrentMetrics`收集与“`collect_interval_milliseconds`”间隔（默认情况下为一秒）。该表非常小（通常以兆字节为单位），默认情况下收集此数据是合理的。 #9225 (阿列克谢-米洛维多夫)
- Initialize query profiler for all threads in a group, e.g. it allows to fully profile insert-queries. Fixes #6964 #8874 (伊万)
- 现在是暂时的 `LIVE VIEW` 创建者 `CREATE LIVE VIEW name WITH TIMEOUT [42] ...` 而不是 `CREATE TEMPORARY LIVE VIEW ...`，因为以前的语法不符合 `CREATE TEMPORARY TABLE ...` #9131 (tavplubix)

- 添加`text_log`。级别配置参数，以限制进入`system.text_log`表 #8809 (Azat Khuzhin)
- 允许根据TTL规则将下载的部分放入磁盘/卷 #8598 (Vladimir Chebotarev)
- 对于外部MySQL字典，允许将MySQL连接池共同化为“share”他们在字典中。此选项显着减少到MySQL服务器的连接数。 #9409 (Clément Rodriguez)
- 显示分位数的最近查询执行时间`clickhouse-benchmark`输出而不是插值值。最好显示与某些查询的执行时间相对应的值。 #8712 (阿列克谢-米洛维多夫)
- 可以在将数据插入到Kafka时为消息添加密钥和时间戳。修复 #7198 #8969 (filimonov)
- 如果服务器从终端运行，请按颜色突出显示线程号，查询id和日志优先级。这是为了提高开发人员相关日志消息的可读性。 #8961 (阿列克谢-米洛维多夫)
- 更好的异常消息，同时加载表`Ordinary`数据库。#9527 (阿列克谢-米洛维多夫)
- 执行`arraySlice`对于具有聚合函数状态的数组。这修复 #9388 #9391 (阿列克谢-米洛维多夫)
- 允许在`in`运算符的右侧使用常量函数和常量数组。#8813 (安东*波波夫)
- 如果在获取系统数据时发生了zookeeper异常。副本，将其显示在单独的列中。这实现了 #9137 #9138 (阿列克谢-米洛维多夫)
- 原子删除`destroy`上的MergeTree数据部分。#8402 (Vladimir Chebotarev)
- 支持分布式表的行级安全性。#8926 (伊万)
- Now we recognize suffix (like KB, KiB...) in settings values. #8072 (米哈伊尔*科罗托夫)
- 在构建大型连接的结果时防止内存不足。#8637 (Artem Zuikov)
- 在交互模式下为建议添加群集名称`clickhouse-client`. #8709 (阿列克谢-米洛维多夫)
- Initialize query profiler for all threads in a group, e.g. it allows to fully profile insert-queries #8820 (伊万)
- 添加列`exception_code`在`system.query_log`桌子 #8770 (米哈伊尔*科罗托夫)
- 在端口上启用MySQL兼容服务器 9004 在默认服务器配置文件中。在配置的例子固定密码生成命令。#8771 (尤里*巴拉诺夫)
- 如果文件系统是只读的，请防止在关闭时中止。这修复 #9094 #9100 (阿列克谢-米洛维多夫)
- 当HTTP POST查询中需要长度时，更好的异常消息。#9453 (阿列克谢-米洛维多夫)
- 添加`_path`和`_file`虚拟列`HDFS`和`File`发动机和`hdfs`和`file`表函数 #8489 (Olga Khvostikova)
- 修复错误`Cannot find column`同时插入到`MATERIALIZED VIEW`在情况下，如果新列被添加到视图的内部表。#8766 #8788 (vzakaznikov) #8788 #8806 (尼古拉*科切托夫) #8803 (尼古拉*科切托夫)
- 通过最终更新后发送进度（如日志）修复本机客户端-服务器协议的进度。这可能仅与使用本机协议的某些第三方工具相关。#9495 (Azat Khuzhin)
- 添加系统指标跟踪使用MySQL协议的客户端连接数 (#9013). #9015 (尤金*克里莫夫)
- 从现在开始，HTTP响应将有`X-ClickHouse-Timezone`标题设置为相同的时区值`SELECT timezone()`会报告。#9493 (Denis Glazachev)

性能改进

- 使用IN提高分析指标的性能 #9261 (安东*波波夫)
- 逻辑函数+代码清理更简单，更有效的代码。跟进到 #8718 #8728 (亚历山大*卡扎科夫)

- 整体性能改善（范围为5%。通过确保使用C++20功能进行更严格的别名处理，对于受影响的查询来说，这是200%）。#9304 (阿莫斯鸟)
- 比较函数的内部循环更严格的别名。#9327 (阿列克谢-米洛维多夫)
- 对于算术函数的内部循环更严格的别名。#9325 (阿列克谢-米洛维多夫)
- ColumnVector::replicate()的实现速度快约3倍，通过该实现ColumnConst::convertToFullColumn()。在实现常数时，也将在测试中有用。#9293 (亚历山大*卡扎科夫)
- 另一个小小的性能改进 ColumnVector::replicate()（这加快了 materialize 函数和高阶函数），甚至进一步改进 #9293 #9442 (亚历山大*卡扎科夫)
- 改进的性能 stochasticLinearRegression 聚合函数。此补丁由英特尔贡献。#8652 (阿列克谢-米洛维多夫)
- 提高性能 reinterpretAsFixedString 功能。#9342 (阿列克谢-米洛维多夫)
- 不要向客户端发送块 Null 处理器管道中的格式。#8797 (尼古拉*科切托夫) #8767 (Alexander Kuzmenkov)

构建/测试/包装改进

- 异常处理现在可以在适用于Linux的Windows子系统上正常工作。看<https://github.com/ClickHouse-Extras/libunwind/pull/3> 这修复 #6480 #9564 (sobolevsv)
- 替换 readline 与 replxx 对于在交互式线编辑 clickhouse-client #8416 (伊万)
- 在FunctionsComparison中更好的构建时间和更少的模板实例化。#9324 (阿列克谢-米洛维多夫)
- 增加了与集成 clang-tidy 在线人 另请参阅 #6044 #9566 (阿列克谢-米洛维多夫)
- 现在我们使用CI链接ClickHouse lld 即使是 gcc. #9049 (阿利沙平)
- 允许随机线程调度和插入毛刺时 THREAD_FUZZER_* 设置环境变量。这有助于测试。#9459 (阿列克谢-米洛维多夫)
- 在无状态测试中启用安全套接字 #9288 (tavplubix)
- 使SPLIT_SHARED_LIBRARIES=OFF更强大 #9156 (Azat Khuzhin)
- 赖眉露>> “performance_introspection_and_logging” 测试可靠的随机服务器卡住。这可能发生在CI环境中。另请参阅 #9515 #9528 (阿列克谢-米洛维多夫)
- 在样式检查中验证XML。#9550 (阿列克谢-米洛维多夫)
- 修正了测试中的竞争条件 00738_lock_for_inner_table. 这个测试依赖于睡眠。#9555 (阿列克谢-米洛维多夫)
- 删除类型的性能测试 once. 这是在统计比较模式下运行所有性能测试（更可靠）所需的。#9557 (阿列克谢-米洛维多夫)
- 增加了算术函数的性能测试。#9326 (阿列克谢-米洛维多夫)
- 增加了性能测试 sumMap 和 sumMapWithOverflow 聚合函数。后续行动 #8933 #8947 (阿列克谢-米洛维多夫)
- 通过样式检查确保错误代码的样式。#9370 (阿列克谢-米洛维多夫)
- 为测试历史添加脚本。#8796 (阿利沙平)
- 添加GCC警告 -Wsuggest-override 找到并修复所有地方 override 必须使用关键字。#8760 (kreuzerkrieg)
- 在Mac OS X下忽略弱符号，因为它必须被定义 #9538 (已删除用户)
- 规范性能测试中某些查询的运行时间。这是在准备在比较模式下运行所有性能测试时完成的。#9565 (阿列克谢-米洛维多夫)
- 修复一些测试，以支持pytest与查询测试 #9062 (伊万)

- 使用MSan在生成中启用SSL，因此在运行无状态测试时，服务器不会在启动时失败 #9531 (tavplubix)
- 修复测试结果中的数据库替换 #9384 (Ilya Yatsishin)
- 针对其他平台构建修复程序 #9381 (proller) #8755 (proller) #8631 (proller)
- 将磁盘部分添加到无状态复盖率测试docker映像 #9213 (帕维尔*科瓦连科)
- 使用GRPC构建时，摆脱源代码树中的文件 #9588 (阿莫斯鸟)
- 通过从上下文中删除SessionCleaner来缩短构建时间。让SessionCleaner的代码更简单。#9232 (阿列克谢-米洛维多夫)
- 更新了clickhouse-test脚本中挂起查询的检查 #8858 (亚历山大*卡扎科夫)
- 从存储库中删除了一些无用的文件。#8843 (阿列克谢-米洛维多夫)
- 更改类型的数学perftests从 once 到 loop. #8783 (尼古拉*科切托夫)
- 添加码头镜像，它允许为我们的代码库构建交互式代码浏览器HTML报告。#8781 (阿利沙平) 见 Woboq代码浏览器
- 抑制MSan下的一些测试失败。#8780 (Alexander Kuzmenkov)
- 加速“exception while insert”测试 此测试通常在具有复盖率的调试版本中超时。#8711 (阿列克谢-米洛维多夫)
- 更新 libcxx 和 libcxxabi 为了主人在准备 #9304 #9308 (阿列克谢-米洛维多夫)
- 修复flacky测试 00910_zookeeper_test_alter_compression_codecs. #9525 (阿列克谢-米洛维多夫)
- 清理重复的链接器标志。确保链接器不会查找意想不到的符号。#9433 (阿莫斯鸟)
- 添加 clickhouse-odbc 驱动程序进入测试图像。这允许通过自己的ODBC驱动程序测试ClickHouse与ClickHouse的交互。#9348 (filimonov)
- 修复单元测试中的几个错误。#9047 (阿利沙平)
- 启用 -Wmissing/include-dirs GCC警告消除所有不存在的包括-主要是由于CMake脚本错误 #8704 (kreuzerkrieg)
- 描述查询探查器无法工作的原因。这是用于 #9049 #9144 (阿列克谢-米洛维多夫)
- 将OpenSSL更新到上游主机。修复了TLS连接可能会失败并显示消息的问题 OpenSSL SSL_read: error:14094438:SSL routines:ssl3_read_bytes:tls1 alert internal error 和 SSL Exception: error:2400006E:random number generator::error retrieving entropy. 该问题出现在版本20.1中。#8956 (阿列克谢-米洛维多夫)
- 更新服务器的Dockerfile #8893 (Ilya Mazaev)
- Build-gcc-from-sources脚本中的小修复 #8774 (Michael Nacharov)
- 替换 numbers 到 zeros 在perftests其中 number 不使用列。这将导致更干净的测试结果。#9600 (尼古拉*科切托夫)
- 修复列构造函数中使用initializer_list时堆栈溢出问题。#9367 (已删除用户)
- 将librdkafka升级到v1.3.0。启用bund绑 rdkafka 和 gsasl mac OS X上的库 #9000 (安德鲁Onyshchuk)
- 在GCC9.2.0上构建修复程序 #9306 (vxider)

碌莽碌.拢.0755-88888888

ClickHouse版本v20.1.8.41,2020-03-20

错误修复

- 修复可能的永久性 Cannot schedule a task 错误（由于未处理的异常 ParallelAggregatingBlockInputStream::Handler::onFinish/onFinishThread）。这修复 #6833. #9154 (Azat Khuzhin)
- 修复过多的内存消耗 ALTER 查询（突变）。这修复 #9533 和 #9670. #9754 (阿利沙平)
- 修复外部字典DDL中反引用的错误。这修复 #9619. #9734 (阿利沙平)

ClickHouse释放v20.1.7.38,2020-03-18

错误修复

- 修正了不正确的内部函数名称 sumKahan 和 sumWithOverflow. 在远程查询中使用此函数时，我会导致异常。#9636 (Azat Khuzhin). 这个问题是在所有ClickHouse版本。
- 允许 ALTER ON CLUSTER 的 Distributed 具有内部复制的表。这修复 #3268. #9617 (shinoi2). 这个问题是在所有ClickHouse版本。
- 修复可能的异常 Size of filter doesn't match size of column 和 Invalid number of rows in Chunk 在 MergeTreeRangeReader. 它们可能在执行时出现 PREWHERE 在某些情况下。修复 #9132. #9612 (安东*波波夫)
- 修复了这个问题：如果你编写一个简单的算术表达式，则不会保留时区 time + 1 （与像这样的表达形成对比 time + INTERVAL 1 SECOND). 这修复 #5743. #9323 (阿列克谢-米洛维多夫). 这个问题是在所有ClickHouse版本。
- 现在不可能创建或添加具有简单循环别名的列，如 a DEFAULT b, b DEFAULT a. #9603 (阿利沙平)
- 修复了base64编码值末尾填充格式错误的问题。更新base64库。这修复 #9491，关闭 #9492 #9500 (阿列克谢-米洛维多夫)
- 修复数据竞赛破坏 Poco::HTTPServer. 当服务器启动并立即关闭时，可能会发生这种情况。#9468 (安东*波波夫)
- 修复可能的崩溃/错误的行数 LIMIT n WITH TIES 当有很多行等于第n行时。#9464 (tavplubix)
- 修复与列Ttl可能不匹配的校验和。#9451 (安东*波波夫)
- 修复当用户尝试崩溃 ALTER MODIFY SETTING 对于老格式化 MergeTree 表引擎家族。#9435 (阿利沙平)
- 现在我们将尝试更频繁地完成突变。#9427 (阿利沙平)
- 修复引入的复制协议不兼容 #8598. #9412 (阿利沙平)
- 修复数组类型的bloom_filter索引的not(has())。#9407 (achimbab)
- 固定的行为 match 和 extract 当干草堆有零字节的函数。当干草堆不变时，这种行为是错误的。这修复 #9160 #9163 (阿列克谢-米洛维多夫) #9345 (阿列克谢-米洛维多夫)

构建/测试/包装改进

- 异常处理现在可以在适用于Linux的Windows子系统上正常工作。看<https://github.com/ClickHouse-Extras/libunwind/pull/3> 这修复 #6480 #9564 (sobolevsv)

ClickHouse释放v20.1.6.30,2020-03-05

错误修复

- 修复压缩时的数据不兼容 T64 编解ec
#9039 (abyss7)
- 在一个线程中从MergeTree表中读取时修复范围顺序。修复 #8964.
#9050 (Curtizj))
- 修复可能的段错误 MergeTreeRangeReader，同时执行 PREWHERE. 修复 #9064.
#9106 (Curtizj))

- 修复 `reinterpretAsFixedString` 返回 `FixedString` 而不是 `String`.
[#9052 \(oandrew\)](#)
- 修复 `joinGet` 使用可为空的返回类型。修复 [#8919](#)
[#9014 \(amosbird\)](#)
- 修复 `bittestall/bitTestAny` 函数的模糊测试和不正确的行为。
[#9143 \(阿列克谢-米洛维多夫\)](#)
- 修复当干草堆有零字节时匹配和提取函数的行为。当干草堆不变时，这种行为是错误的。修复 [#9160](#)
[#9163 \(阿列克谢-米洛维多夫\)](#)
- 当使用非严格单调函数索引时，固定执行反转谓词。修复 [#9034](#)
[#9223 \(Akazz\)](#)
- 允许重写 `CROSS` 到 `INNER JOIN` 如果有 `[NOT] LIKE` 操作员在 `WHERE` 科。修复 [#9191](#)
[#9229 \(4ertus2\)](#)
- 允许使用日志引擎的表中的第一列成为别名。
[#9231 \(abyss7\)](#)
- 允许逗号加入 `IN()` 进去 修复 [#7314](#).
[#9251 \(4ertus2\)](#)
- 改进 `ALTER MODIFY/ADD` 查询逻辑。现在你不能 `ADD` 不带类型的列, `MODIFY` 默认表达式不改变列的类型和 `MODIFY type` 不会丢失默认表达式值。修复 [#8669](#).
[#9227 \(alesapin\)](#)
- 修复突变最终确定，当已经完成突变时可以具有状态 `is_done=0`。
[#9217 \(alesapin\)](#)
- 碜莽禄Support: “Processors” 管道系统.数字和系统.`numbers_mt` 这也修复了错误时 `max_execution_time` 不被尊重。
[#7796 \(KochetovNicolai\)](#)
- 修复错误的计数 `DictCacheKeysRequestedFound` 公制。
[#9411 \(nikitamikhaylov\)](#)
- 添加了对存储策略的检查 `ATTACH PARTITION FROM`, `REPLACE PARTITION`, `MOVE TO TABLE` 否则可能使部分数据在重新启动后无法访问，并阻止ClickHouse启动。
[#9383 \(excitoon\)](#)
- 在固定的瑞银报告 `MergeTreeIndexSet`. 这修复 [#9250](#)
[#9365 \(阿列克谢-米洛维多夫\)](#)
- 在 `BlockIO` 中修复可能的数据集。
[#9356 \(KochetovNicolai\)](#)
- 支持 `UInt64` 在 `JSON` 相关函数中不适合 `Int64` 的数字。更新 `SIMDJSON` 为了主人 这修复 [#9209](#)
[#9344 \(阿列克谢-米洛维多夫\)](#)
- 如果将数据目录挂载到单独的设备，则修复可用空间量计算不正确时的问题。对于默认磁盘，计算数据子目录的可用空间。这修复 [#7441](#)
[#9257 \(米尔布\)](#)
- 修复 TLS 连接可能会失败并显示消息时的问题 `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.`. 将 OpenSSL 更新到上游主机。
[#8956 \(阿列克谢-米洛维多夫\)](#)

- 执行时 `CREATE` 查询，在存储引擎参数中折叠常量表达式。将空数据库名称替换为当前数据库。修复 #6508, #3492。还修复了 `ClickHouseDictionarySource` 中检查本地地址。
#9262 (tabplubix)
- 修复段错误 `StorageMerge`，从 `StorageFile` 读取时可能发生。
#9387 (tabplubix)
- 防止丢失数据 `Kafka` 在极少数情况下，在读取后缀之后但在提交之前发生异常。修复 #9378。相关: #7175
#9507 (菲利蒙诺夫)
- 修复尝试使用/删除时导致服务器终止的错误 `Kafka` 使用错误的参数创建的表。修复 #9494。结合 #9507。
#9513 (菲利蒙诺夫)

新功能

- 添加 `deduplicate_blocks_in_dependent_materialized_views` 用于控制具有实例化视图的表中幂等插入的行为的选项。这个新功能是由 Altinity 的特殊要求添加到错误修正版本中的。
#9070 (urykhy)

ClickHouse 版本 v20.1.2.4, 2020-01-22

向后不兼容的更改

- 使设置 `merge_tree_uniform_read_distribution` 过时了。服务器仍可识别此设置，但无效。#8308 (阿列克谢-米洛维多夫)
- 更改函数的返回类型 `greatCircleDistance` 到 `Float32` 因为现在计算的结果是 `Float32`. #7993 (阿列克谢-米洛维多夫)
- 现在预计查询参数表示为 “escaped” 格式。例如，要传递字符串 `a<tab>b` 你必须写 `a\ltb` 或 `a\<tab>b` 并分别，`a%5Ctb` 或 `a%5C%09b` 在 URL 中。这是需要添加传递 `NULL` 作为的可能性 `\N`。这修复 #7488. #8517 (阿列克谢-米洛维多夫)
- 启用 `use_minimalistic_part_header_in_zookeeper` 设置 `ReplicatedMergeTree` 默认情况下。这将显着减少存储在 ZooKeeper 中的数据量。自 19.1 版本以来支持此设置，我们已经在多个服务的生产中使用它，半年以上没有任何问题。如果您有机会降级到 19.1 以前的版本，请禁用此设置。#6850 (阿列克谢-米洛维多夫)
- 数据跳过索引已准备就绪并默认启用。设置 `allow_experimental_data_skipping_indices`, `allow_experimental_cross_to_join_conversion` 和 `allow_experimental_multiple_joins_emulation` 现在已经过时，什么也不做。#7974 (阿列克谢-米洛维多夫)
- 添加新建 ANY JOIN 逻辑 `StorageJoin` 符合 `JOIN` 操作。要在不改变行为的情况下进行升级，您需要添加 SETTINGS `any_join_distinct_right_table_keys = 1` 引擎联接表元数据或在升级后重新创建这些表。#8400 (Artem Zuikov)
- 要求重新启动服务器以应用日志记录配置中的更改。这是一种临时解决方法，可以避免服务器将日志记录到已删除的日志文件中的错误（请参阅 #8696）。#8707 (Alexander Kuzmenkov)

新功能

- 添加了有关部件路径的信息 `system.merges`. #8043 (Vladimir Chebotarev)
- 添加执行能力 `SYSTEM RELOAD DICTIONARY` 查询中 `ON CLUSTER` 模式 #8288 (纪尧姆*塔瑟里)
- 添加执行能力 `CREATE DICTIONARY` 查询中 `ON CLUSTER` 模式 #8163 (阿利沙平)
- 现在用户的个人资料 `users.xml` 可以继承多个配置文件。#8343 (Mikhail f. Shiryaev)
- 已添加 `system.stack_trace` 允许查看所有服务器线程的堆栈跟踪的表。这对于开发人员反省服务器状态非常有用。这修复 #7576. #8344 (阿列克谢-米洛维多夫)
- 添加 `DateTime64` 具有可配置子秒精度的数据类型。#7170 (瓦西里*内姆科夫)
- 添加表函数 `clusterAllReplicas` 这允许查询集群中的所有节点。#8493 (kiran sunkari)

- 添加聚合函数 `categoricalInformationValue` 其计算出离散特征的信息值。 #8117 (hcz)
- 加快数据文件的解析 `CSV`, `TSV` 和 `JSONEachRow` 通过并行进行格式化。 #7780 (Alexander Kuzmenkov)
- 添加功能 `bankerRound` 它执行银行家的四舍五入。 #8112 (hcz)
- 支持区域名称的嵌入式字典中的更多语言: 'ru', 'en', 'ua', 'uk', 'by', 'kz', 'tr', 'de', 'uz', 'lv', 'lt', 'et', 'pt', 'he', 'vi'. #8189 (阿列克谢-米洛维多夫)
- 改进的一致性 `ANY JOIN` 逻辑 现在 `t1 ANY LEFT JOIN t2` 等于 `t2 ANY RIGHT JOIN t1`. #7665 (Artem Zuikov)
- 添加设置 `any_join_distinct_right_table_keys` 这使旧的行为 `ANY INNER JOIN`. #7665 (Artem Zuikov)
- 添加新建 `SEMI` 和 `ANTI JOIN`. 老 `ANY INNER JOIN` 行为现在可作为 `SEMI LEFT JOIN`. #7665 (Artem Zuikov)
- 已添加 `Distributed` 格式 `File` 发动机和 `file` 表函数, 它允许从读 `.bin` 通过异步插入生成的文件 `Distributed` 桌子 #8535 (尼古拉*科切托夫)
- 添加可选的重置列参数 `runningAccumulate` 这允许为每个新的键值重置聚合结果。 #8326 (谢尔盖*科诺年科)
- 添加使用 `ClickHouse` 作为普罗米修斯端点的能力。 #7900 (vdimir)
- 添加部分 `<remote_url_allow_hosts>` 在 `config.xml` 这将限制允许的主机用于远程表引擎和表函数 `URL`, `S3`, `HDFS`. #7154 (米哈伊尔*科罗托夫)
- 添加功能 `greatCircleAngle` 它计算球体上的距离 (以度为单位) 。 #8105 (阿列克谢-米洛维多夫)
- 改变地球半径与 `h3` 库一致。 #8105 (阿列克谢-米洛维多夫)
- 已添加 `JSONCompactEachRow` 和 `JSONCompactEachRowWithNamesAndTypes` 输入和输出格式。 #7841 (米哈伊尔*科罗托夫)
- 增加了与文件相关的表引擎和表函数的功能 (`File`, `S3`, `URL`, `HDFS`) 它允许读取和写入 `gzip` 基于附加引擎参数或文件扩展名的文件。 #7840 (安德烈*博德罗夫)
- 添加了 `randomASCII(length)` 函数, 生成一个字符串与一个随机集 `ASCII` 可打印字符。 #8401 (刺刀)
- 添加功能 `JSONExtractArrayRaw` 它返回从未解析的 json 数组元素上的数组 `JSON` 字符串。 #8081 (Oleg Matrokhin)
- 添加 `arrayZip` 函数允许将多个长度相等的数据组合成一个元组数组。 #8149 (张冬)
- 添加根据配置的磁盘之间移动数据的能力 `TTL`-表达式为 *MergeTree 表引擎家族. #8140 (Vladimir Chebotarev)
- 增加了新的聚合功能 `avgWeighted` 其允许计算加权平均值。 #7898 (安德烈*博德罗夫)
- 现在并行解析默认启用 `TSV`, `TSKV`, `CSV` 和 `JSONEachRow` 格式。 #7894 (尼基塔*米哈伊洛夫)
- 从添加几个地理功能 `H3` 图书馆: `h3GetResolution`, `h3EdgeAngle`, `h3EdgeLength`, `h3IsValid` 和 `h3kRing`. #8034 (Konstantin Malanchev)
- 增加了对 `brotli` 的支持 (`br`) 压缩文件相关的存储和表函数。这修复 #8156. #8526 (阿列克谢-米洛维多夫)
- 添加 `groupBit*` 功能的 `SimpleAggregationFunction` 类型。 #8485 (纪尧姆*塔瑟里)

错误修复

- 修复重命名表 `Distributed` 引擎 修复问题 #7868. #8306 (tavplubix)
- 现在字典支持 `EXPRESSION` 对于非 `ClickHouse` SQL 方言中任意字符串中的属性。 #8098 (阿利沙平)
- 修复损坏 `INSERT SELECT FROM mysql(...)` 查询。这修复 #8070 和 #7960. #8234 (tavplubix)
- 修复错误 “Mismatch column sizes” 插入默认值时 `Tuple` 从 `JSONEachRow`. 这修复 #5653. #8606 (tavplubix)

- 现在将在使用的情况下抛出一个异常 `WITH TIES` 旁边的 `LIMIT BY`. 还增加了使用能力 `TOP` 与 `LIMIT BY`. 这修复 #7472. #7637 (尼基塔*米哈伊洛夫)
- 从新鲜的glibc版本中修复unintended依赖关系 `clickhouse-odbc-bridge` 二进制 #8046 (阿莫斯鸟)
- 修正错误的检查功能 *MergeTree 引擎家族. 现在, 当我们在最后一个颗粒和最后一个标记 (非最终) 中有相同数量的行时, 它不会失败。 #8047 (阿利沙平)
- 修复插入 `Enum*` 列后 `ALTER` 查询, 当基础数值类型等于表指定类型时。这修复 #7836. #7908 (安东*波波夫)
- 允许非常数负 “size” 函数的参数 `substring`. 这是不允许的错误。这修复 #4832. #7703 (阿列克谢-米洛维多夫)
- 修复当错误数量的参数传递到解析错误 (OJ)DBC 表引擎。#7709 (阿利沙平)
- 将日志发送到 `syslog` 时使用正在运行的 `clickhouse` 进程的命令名。在以前的版本中, 使用空字符串而不是命令名称。#8460 (Michael Nacharov)
- 修复检查允许的主机 `localhost`. 这个公关修复了在提供的解决方案 #8241. #8342 (维塔利*巴拉诺夫)
- 修复罕见的崩溃 `argMin` 和 `argMax` 长字符串参数的函数, 当结果被用于 `runningAccumulate` 功能。这修复 #8325 #8341 (恐龙)
- 修复表的内存过度使用 `Buffer` 引擎 #8345 (Azat Khuzhin)
- 修正了可以采取的功能中的潜在错误 `NULL` 作为参数之一, 并返回非NULL。#8196 (阿列克谢-米洛维多夫)
- 在线程池中更好地计算后台进程的指标 `MergeTree` 表引擎. #8194 (Vladimir Chebotarev)
- 修复功能 `IN` 里面 `WHERE` 存在行级表筛选器时的语句。修复 #6687 #8357 (伊万)
- 现在, 如果整数值没有完全解析设置值, 则会引发异常。#7678 (米哈伊尔*科罗托夫)
- 修复当聚合函数用于查询具有两个以上本地分片的分布式表时出现的异常。#8164 (小路)
- 现在, bloom filter 可以处理零长度数组, 并且不执行冗余计算。#8242 (achimbab)
- 修正了通过匹配客户端主机来检查客户端主机是否允许 `host_regex` 在指定 `users.xml`. #8241 (维塔利*巴拉诺夫)
- 放松不明确的列检查, 导致多个误报 `JOIN ON` 科。#8385 (Artem Zuikov)
- 修正了可能的服务器崩溃 (`std::terminate`) 当服务器不能发送或写入数据 `JSON` 或 `XML` 格式与值 `String` 数据类型 (需要 `UTF-8` 验证) 或使用 `Brotli` 算法或其他一些罕见情况下压缩结果数据时。这修复 #7603 #8384 (阿列克谢-米洛维多夫)
- 修复竞争条件 `StorageDistributedDirectoryMonitor` 被线人发现 这修复 #8364. #8383 (尼古拉*科切托夫)
- 现在背景合并 *MergeTree 表引擎家族更准确地保留存储策略卷顺序。#8549 (Vladimir Chebotarev)
- 现在表引擎 `Kafka` 与正常工作 `Native` 格式。这修复 #6731 #7337 #8003. #8016 (filimonov)
- 固定格式与标题 (如 `CSVWithNames`) 这是抛出关于 EOF 表引擎的异常 `Kafka`. #8016 (filimonov)
- 修复了从子查询右侧部分制作 `set` 的错误 `IN` 科。这修复 #5767 和 #2542. #7755 (尼基塔*米哈伊洛夫)
- 从存储读取时修复可能的崩溃 `File`. #7756 (尼古拉*科切托夫)
- 在固定的文件读取 `Parquet` 包含类型列的格式 `list`. #8334 (马苏兰)
- 修复错误 `Not found column` 对于分布式查询 `PREWHERE` 条件取决于采样键 if `max_parallel_replicas > 1`. #7913 (尼古拉*科切托夫)
- 修复错误 `Not found column` 如果使用查询 `PREWHERE` 依赖于表的别名, 结果集由于主键条件而为空。#7911 (尼古拉*科切托夫)

- 函数的固定返回类型 `rand` 和 `randConstant` 在情况下 `Nullable` 争论。现在函数总是返回 `UInt32` 而且从来没有 `Nullable(UInt32)`. #8204 (尼古拉*科切托夫)
- 禁用谓词下推 `WITH FILL` 表达。这修复 #7784. #7789 (张冬)
- 修正错误 `count()` 结果 `SummingMergeTree` 当 `FINAL` 部分被使用。#3280 #7786 (尼基塔*米哈伊洛夫)
- 修复来自远程服务器的常量函数可能不正确的结果。它发生在具有以下功能的查询中 `version()`, `uptime()` 等。它为不同的服务器返回不同的常量值。这修复 #7666. #7689 (尼古拉*科切托夫)
- 修复下推谓词优化中导致错误结果的复杂错误。这解决了下推谓词优化的很多问题。#8503 (张冬)
- 修复崩溃 `CREATE TABLE .. AS dictionary` 查询。#8508 (Azat Khuzhin)
- 一些改进ClickHouse语法 .g4 文件 #8294 (太阳里)
- 修复导致崩溃的错误 `JOINS`与表与发动机 `Join`. 这修复 #7556 #8254 #7915 #8100. #8298 (Artem Zuikov)
- 修复冗余字典重新加载 `CREATE DATABASE`. #7916 (Azat Khuzhin)
- 限制从读取流的最大数量 `StorageFile` 和 `StorageHDFS`. 修复
<https://github.com/ClickHouse/ClickHouse/issues/7650>. #7981 (阿利沙平)
- 修复bug `ALTER ... MODIFY ... CODEC` 查询，当用户同时指定默认表达式和编解`ec`。修复 8593. #8614 (阿利沙平)
- 修复列的后台合并错误 `SimpleAggregateFunction(LowCardinality)` 类型。#8613 (尼古拉*科切托夫)
- 固定类型签入功能 `toDateTime64`. #8375 (瓦西里*内姆科夫)
- 现在服务器不崩溃 `LEFT` 或 `FULL JOIN` 与和加入引擎和不支持 `join_use_nulls` 设置。#8479 (Artem Zuikov)
- 现在 `DROP DICTIONARY IF EXISTS db.dict` 查询不会抛出异常，如果 `db` 根本不存在 #8185 (维塔利*巴拉诺夫)
- 修复表函数中可能出现的崩溃 (`file`, `mysql`, `remote`) 引用删除引起的 `IStorage` 对象。修复插入表函数时指定的列的不正确解析。#7762 (tavplubix)
- 确保网络启动前 `clickhouse-server`. 这修复 #7507. #8570 (余志昌)
- 修复安全连接的超时处理，因此查询不会无限挂起。这修复 #8126. #8128 (阿列克谢-米洛维多夫)
- 修复 `clickhouse-copier`并发工人之间的冗余争用。#7816 (丁香飞)
- 现在突变不会跳过附加的部分，即使它们的突变版本比当前的突变版本大。#7812 (余志昌) #8250 (阿利沙平)
- 忽略冗余副本 *MergeTree 数据部分移动到另一个磁盘和服务器重新启动后。#7810 (Vladimir Chebotarev)
- 修复崩溃 `FULL JOIN` 与 `LowCardinality` 在 `JOIN` 钥匙 #8252 (Artem Zuikov)
- 禁止在插入查询中多次使用列名，如 `INSERT INTO tbl (x, y, x)`. 这修复 #5465, #7681. #7685 (阿利沙平)
- 增加了回退，用于检测未知Cpu的物理CPU内核数量（使用逻辑CPU内核数量）。这修复 #5239. #7726 (阿列克谢-米洛维多夫)
- 修复 `There's no column` 实例化列和别名列错。#8210 (Artem Zuikov)
- 固定切断崩溃时 `EXISTS` 查询没有使用 `TABLE` 或 `DICTIONARY` 预选赛 就像 `EXISTS t`. 这修复 #8172. 此错误在版本 19.17 中引入。#8213 (阿列克谢-米洛维多夫)
- 修复罕见错误 "Sizes of columns doesn't match" 使用时可能会出现 `SimpleAggregateFunction` 列。#7790 (Boris Granveaud)
- 修正错误，其中用户空 `allow_databases` 可以访问所有数据库（和相同的 `allow_dictionaries`). #7793 (DeifyTheGod)

- 修复客户端崩溃时，服务器已经从客户端断开连接。 #8071 (Azat Khuzhin)
- 修复 ORDER BY 在按主键前缀和非主键后缀排序的情况下行为。 #7759 (安东*波波夫)
- 检查表中是否存在合格列。这修复 #6836. #7758 (Artem Zuikov)
- 固定行为 ALTER MOVE 合并完成后立即运行移动指定的超部分。修复 #8103. #8104 (Vladimir Chebotarev)
- 使用时修复可能的服务器崩溃 UNION 具有不同数量的列。修复 #7279. #7929 (尼古拉*科切托夫)
- 修复函数结果子字符串的大小 substr 负大小。#8589 (尼古拉*科切托夫)
- 现在服务器不执行部分突变 MergeTree 如果后台池中没有足够的可用线程。#8588 (tavplubix)
- 修复格式化时的小错字 UNION ALL AST. #7999 (litao91)
- 修正了负数不正确的布隆过滤结果。这修复 #8317. #8566 (张冬)
- 在解压缩固定潜在的缓冲区溢出。恶意用户可以传递捏造的压缩数据，这将导致缓冲区后读取。这个问题是由Yandex 信息安全部队的Eldar Zaitov发现的。#8404 (阿列克谢-米洛维多夫)
- 修复因整数溢出而导致的错误结果 arrayIntersect. #7777 (尼古拉*科切托夫)
- 现在 OPTIMIZE TABLE query 不会等待脱机副本执行该操作。#8314 (javi santana)
- 固定 ALTER TTL 解析器 Replicated*MergeTree 桌子 #8318 (Vladimir Chebotarev)
- 修复服务器和客户端之间的通信，以便服务器在查询失败后读取临时表信息。#8084 (Azat Khuzhin)
- 修复 bitmapAnd 在聚合位图和标量位图相交时出现函数错误。#8082 (黄月)
- 完善的定义 ZXid 根据动物园管理员的程序员指南，它修复了错误 clickhouse-cluster-copier. #8088 (丁香飞)
- odbc 表函数现在尊重 external_table_functions_use_nulls 设置。#7506 (瓦西里*内姆科夫)
- 修正了导致罕见的数据竞赛的错误。#8143 (亚历山大*卡扎科夫)
- 现在 SYSTEM RELOAD DICTIONARY 完全重新加载字典，忽略 update_field. 这修复 #7440. #8037 (维塔利*巴拉诺夫)
- 添加检查字典是否存在于创建查询的能力。#8032 (阿利沙平)
- 修复 Float* 解析中 Values 格式。这修复 #7817. #7870 (tavplubix)
- 修复崩溃时，我们不能在一些后台操作保留空间 *MergeTree 表引擎家族. #7873 (Vladimir Chebotarev)
- 修复表包含合并操作时的崩溃 SimpleAggregateFunction(LowCardinality) 列。这修复 #8515. #8522 (Azat Khuzhin)
- 恢复对所有ICU区域设置的支持，并添加对常量表达式应用排序规则的功能。还添加语言名称 system.collations 桌子 #8051 (阿利沙平)
- 修正错误时，外部字典与零最小寿命 (LIFETIME(MIN 0 MAX N), LIFETIME(N)) 不要在后台更新。#7983 (阿利沙平)
- 修复当clickhouse源外部字典在查询中有子查询时崩溃。#8351 (尼古拉*科切托夫)
- 修复文件扩展名不正确的解析表与引擎 URL. 这修复 #8157. #8419 (安德烈*博德罗夫)
- 修复 CHECK TABLE 查询为 *MergeTree 表没有关键. 修复 #7543. #7979 (阿利沙平)
- 固定转换 Float64 到MySQL类型。#8079 (尤里*巴拉诺夫)
- 现在，如果表没有完全删除，因为服务器崩溃，服务器将尝试恢复并加载它。#8176 (tavplubix)
- 修复了表函数中的崩溃 file 同时插入到不存在的文件。现在在这种情况下，文件将被创建，然后插入将被处理。#8177 (Olga Khvostikova)

- 修复罕见的死锁时，可能发生 `trace_log` 处于启用状态。 #7838 (filimonov)
- 添加能力与不同类型的工作，除了 `Date` 在 `RangeHashed` 从DDL查询创建的外部字典。修复 7899. #8275 (阿利沙平)
- 修复崩溃时 `now64()` 用另一个函数的结果调用。 #8270 (瓦西里*内姆科夫)
- 修正了通过mysql有线协议检测客户端IP连接的错误。 #7743 (Dmitry Muzyka)
- 修复空阵列处理 `arraySplit` 功能。这修复 #7708. #7747 (hczi)
- 修复了以下问题 `pid-file` 另一个运行 `clickhouse-server` 可能会被删除。 #8487 (徐伟清)
- 修复字典重新加载，如果它有 `invalidate_query`，停止更新，并在以前的更新尝试一些异常。 #8029 (阿利沙平)
- 修正了功能错误 `arrayReduce` 这可能会导致 “double free” 和聚合函数组合器中的错误 `Resample` 这可能会导致内存泄漏。添加聚合功能 `aggThrow`. 此功能可用于测试目的。 #8446 (阿列克谢-米洛维多夫)

改进

- 改进了使用时的日志记录 `S3` 表引擎。 #8251 (Grigory Pervakov)
- 在调用时未传递任何参数时打印帮助消息 `clickhouse-local`. 这修复 #5335. #8230 (安德烈*纳戈尔尼)
- 添加设置 `mutations_sync` 这允许等待 `ALTER UPDATE/DELETE` 同步查询。 #8237 (阿利沙平)
- 允许设置相对 `user_files_path` 在 `config.xml` (在类似的方式 `format_schema_path`). #7632 (hczi)
- 为转换函数添加非法类型的异常 `-OrZero` 后缀 #7880 (安德烈*科尼利亚耶夫)
- 简化在分布式查询中发送到分片的数据头的格式。 #8044 (维塔利*巴拉诺夫)
- `Live View` 表引擎重构。 #8519 (vzakaznikov)
- 为从DDL查询创建的外部字典添加额外的检查。 #8127 (阿利沙平)
- 修复错误 `Column ... already exists` 使用时 `FINAL` 和 `SAMPLE` together, e.g. `select count() from table final sample 1/2.` 修复 #5186. #7907 (尼古拉*科切托夫)
- 现在表的第一个参数 `joinGet` 函数可以是表标识符。 #7707 (阿莫斯鸟)
- 允许使用 `MaterializedView` 与上面的子查询 `Kafka` 桌子 #8197 (filimonov)
- 现在后台在磁盘之间移动，运行它的seprate线程池。 #7670 (Vladimir Chebotarev)
- `SYSTEM RELOAD DICTIONARY` 现在同步执行。 #8240 (维塔利*巴拉诺夫)
- 堆栈跟踪现在显示物理地址 (对象文件中的偏移量)，而不是虚拟内存地址 (加载对象文件的位置)。这允许使用 `addr2line` 当二进制独立于位置并且ASLR处于活动状态时。这修复 #8360. #8387 (阿列克谢-米洛维多夫)
- 支持行级安全筛选器的新语法: `<table name='table_name'>...</table>`. 修复 #5779. #8381 (伊万)
- 现在 `cityHash` 功能可以与工作 `Decimal` 和 `UUID` 类型。修复 #5184. #7693 (米哈伊尔*科罗托夫)
- 从系统日志中删除了固定的索引粒度 (它是1024)，因为它在实现自适应粒度之后已经过时。 #7698 (阿列克谢-米洛维多夫)
- 当ClickHouse在没有SSL的情况下编译时，启用MySQL兼容服务器。 #7852 (尤里*巴拉诺夫)
- 现在服务器校验和分布式批处理，这在批处理中损坏数据的情况下提供了更多详细的错误。 #7914 (Azat Khuzhin)
- 碟莽禄Support: `DROP DATABASE`, `DETACH TABLE`, `DROP TABLE` 和 `ATTACH TABLE` 为 MySQL 数据库引擎。 #8202 (张冬)
- 在S3表功能和表引擎中添加身份验证。 #7623 (Vladimir Chebotarev)

- 增加了检查额外的部分 MergeTree 在不同的磁盘上，为了不允许错过未定义磁盘上的数据部分。 #8118 (Vladimir Chebotarev)
- 启用 Mac 客户端和服务器的 SSL 支持。 #8297 (伊万)
- 现在 ClickHouse 可以作为 MySQL 联合服务器（参见 <https://dev.mysql.com/doc/refman/5.7/en/federated-create-server.html>）。 #7717 (Maxim Fedotov)
- clickhouse-client 现在只能启用 bracketed-paste 当多查询处于打开状态且多行处于关闭状态时。这修复 (#7757) [<https://github.com/ClickHouse/ClickHouse/issues/7757>]。 #7761 (阿莫斯鸟)
- 碟莽禄 Support: Array(Decimal) 在 if 功能。 #7721 (Artem Zuikov)
- 支持小数 arrayDifference, arrayCumSum 和 arrayCumSumNegative 功能。 #7724 (Artem Zuikov)
- 已添加 lifetime 列到 system.dictionaries 桌子 #6820 #7727 (kekekekule)
- 改进了检查不同磁盘上的现有部件 *MergeTree 表引擎。地址 #7660. #8440 (Vladimir Chebotarev)
- 集成与 AWS SDK 为 S3 交互允许使用开箱即用的所有 S3 功能。 #8011 (帕维尔*科瓦连科)
- 增加了对子查询的支持 Live View 桌子 #7792 (vzakaznikov)
- 检查使用 Date 或 DateTime 从列 TTL 表达式已删除。 #7920 (Vladimir Chebotarev)
- 有关磁盘的信息已添加到 system.detached_parts 桌子 #7833 (Vladimir Chebotarev)
- 现在设置 max_(table|partition)_size_to_drop 无需重新启动即可更改。 #7779 (Grigory Pervakov)
- 错误消息的可用性略好。要求用户不要删除下面的行 Stack trace:.. #7897 (阿列克谢-米洛维多夫)
- 更好地阅读消息 Kafka 引擎在各种格式后 #7935. #8035 (伊万)
- 与不支持 MySQL 客户端更好的兼容性 sha2_password 验证插件。 #8036 (尤里*巴拉诺夫)
- 支持 MySQL 兼容性服务器中的更多列类型。 #7975 (尤里*巴拉诺夫)
- 执行 ORDER BY 优化 Merge, Buffer 和 Materialized View 存储与底层 MergeTree 桌子 #8130 (安东*波波夫)
- 现在我们总是使用 POSIX 实现 getrandom 与旧内核更好的兼容性 (\<3.17) 。 #7940 (阿莫斯鸟)
- 更好地检查移动 ttl 规则中的有效目标。 #8410 (Vladimir Chebotarev)
- 更好地检查损坏的刀片批次 Distributed 表引擎。 #7933 (Azat Khuzhin)
- 添加带有部件名称数组的列，这些部件将来必须处理突变 system.mutations 桌子 #8179 (阿利沙平)
- 处理器的并行合并排序优化。 #8552 (尼古拉*科切托夫)
- 设置 mark_cache_min_lifetime 现在已经过时了，什么也不做。在以前的版本中，标记缓存可以在内存中增长大于 mark_cache_size 以容纳内的数据 mark_cache_min_lifetime 秒。这导致了混乱和比预期更高的内存使用率，这在内存受限的系统上尤其糟糕。如果您在安装此版本后会看到性能下降，则应增加 mark_cache_size. #8484 (阿列克谢-米洛维多夫)
- 准备使用 tid 到处都是 这是必要的 #7477. #8276 (阿列克谢-米洛维多夫)

性能改进

- 处理器管道中的性能优化。 #7988 (尼古拉*科切托夫)
- 缓存字典中过期密钥的非阻塞更新（具有读取旧密钥的权限）。 #8303 (尼基塔*米哈伊洛夫)
- 没有编译 ClickHouse -fno-omit-frame-pointer 在全球范围内多余一个寄存器。 #8097 (阿莫斯鸟)

- 加速 `greatCircleDistance` 功能，并为它添加性能测试。 #7307 (Olga Khvostikova)
- 改进的功能性能 `roundDown`. #8465 (阿列克谢-米洛维多夫)
- 改进的性能 `max`, `min`, `argMin`, `argMax` 为 `DateTime64` 数据类型。 #8199 (瓦西里*内姆科夫)
- 改进了无限制或大限制和外部排序的排序性能。 #8545 (阿列克谢-米洛维多夫)
- 改进的性能格式化浮点数高达6倍。 #8542 (阿列克谢-米洛维多夫)
- 改进的性能 `modulo` 功能。 #7750 (阿莫斯鸟)
- 优化 `ORDER BY` 并与单列键合并。 #8335 (阿列克谢-米洛维多夫)
- 更好地实施 `arrayReduce`, `-Array` 和 `-State` 组合子 #7710 (阿莫斯鸟)
- 现在 `PREWHERE` 应优化为至少一样高效 `WHERE`. #7769 (阿莫斯鸟)
- 改进方式 `round` 和 `roundBankers` 处理负数。 #8229 (hcZ)
- 改进的解码性能 `DoubleDelta` 和 `Gorilla` 编解码器大约30-40%。 这修复 #7082. #8019 (瓦西里*内姆科夫)
- 改进的性能 `base64` 相关功能。 #8444 (阿列克谢-米洛维多夫)
- 增加了一个功能 `geoDistance`. 它类似于 `greatCircleDistance` 但使用近似于WGS-84椭球模型。 两个功能的性能几乎相同。 #8086 (阿列克谢-米洛维多夫)
- 更快 `min` 和 `max` 聚合函数 `Decimal` 数据类型。 #8144 (Artem Zuikov)
- 矢量化处理 `arrayReduce`. #7608 (阿莫斯鸟)
- `if` 链现在优化为 `multilf`. #8355 (kamalov-ruslan)
- 修复性能回归 `Kafka` 表引擎在19.15中引入。 这修复 #7261. #7935 (filimonov)
- 已删除“pie”代码生成 `gcc` 从Debian软件包偶尔带来默认情况下。 #8483 (阿列克谢-米洛维多夫)
- 并行解析数据格式 #6553 (尼基塔*米哈伊洛夫)
- 启用优化的解析器 `Values` 默认使用表达式 (`input_format_values_deduce_templates_of_expressions=1`). #8231 (tavplubix)

构建/测试/包装改进

- 构建修复 `ARM` 而在最小模式。 #8304 (proller)
- 添加复盖文件刷新 `clickhouse-server` 当不调用`std::atexit`时。 还略微改进了无状态测试的复盖率日志记录。 #8267 (阿利沙平)
- 更新contrib中的LLVM库。 避免从操作系统包中使用LLVM。 #8258 (阿列克谢-米洛维多夫)
- 使bund绑 `curl` 建立完全安静。 #8232 #8203 (帕维尔*科瓦连科)
- 修复一些 `MemorySanitizer` 警告。 #8235 (Alexander Kuzmenkov)
- 使用 `add_warning` 和 `no_warning` 宏 `CMakeLists.txt`. #8604 (伊万)
- 添加对Minio S3兼容对象的支持 (<https://min.io/>)为了更好的集成测试。 #7863 #7875 (帕维尔*科瓦连科)
- 导入 `libc` 标题到contrib。 它允许在各种系统中使构建更加一致 (仅适用于 `x86_64-linux-gnu`). #5773 (阿列克谢-米洛维多夫)
- 删除 `-fPIC` 从一些图书馆。 #8464 (阿列克谢-米洛维多夫)

- 清洁 CMakeLists.txt 对于卷曲。看[#8459](https://github.com/ClickHouse/ClickHouse/pull/8011#issuecomment-569478910) (阿列克谢-米洛维多夫)
- 无声警告 CapNProto 图书馆. [#8220](#) (阿列克谢-米洛维多夫)
- 为短字符串优化哈希表添加性能测试。 [#7679](#) (阿莫斯鸟)
- 现在 ClickHouse 将建立在 AArch64 即使 MADV_FREE 不可用。这修复 [#8027](#). [#8243](#) (阿莫斯鸟)
- 更新 zlib-ng 来解决记忆消毒的问题 [#7182](#) [#8206](#) (Alexander Kuzmenkov)
- 在非Linux系统上启用内部MySQL库，因为操作系统包的使用非常脆弱，通常根本不起作用。这修复 [#5765](#). [#8426](#) (阿列克谢-米洛维多夫)
- 修复了启用后在某些系统上构建的问题 libc++. 这取代了 [#8374](#). [#8380](#) (阿列克谢-米洛维多夫)
- 赖眉露>> Field 方法更类型安全，以找到更多的错误。 [#7386](#) [#8209](#) (Alexander Kuzmenkov)
- 添加丢失的文件到 libc-headers 子模块。 [#8507](#) (阿列克谢-米洛维多夫)
- 修复错误 JSON 引用性能测试输出。 [#8497](#) (尼古拉*科切托夫)
- 现在堆栈跟踪显示 std::exception 和 Poco::Exception. 在以前的版本中，它仅适用于 DB::Exception. 这改进了诊断。 [#8501](#) (阿列克谢-米洛维多夫)
- 移植 clock_gettime 和 clock_nanosleep 对于新鲜的glibc版本。 [#8054](#) (阿莫斯鸟)
- 启用 part_log 在示例配置开发人员。 [#8609](#) (阿列克谢-米洛维多夫)
- 修复重新加载的异步性质 01036_no_superfluous_dict_reload_on_create_database*. [#8111](#) (Azat Khuzhin)
- 固定编解码器性能测试。 [#8615](#) (瓦西里*内姆科夫)
- 添加安装脚本 .tgz 为他们构建和文档。 [#8612](#) [#8591](#) (阿利沙平)
- 删除旧 ZSTD 测试（它是在2016年创建的，以重现zstd1.0版本之前的错误）。这修复 [#8618](#). [#8619](#) (阿列克谢-米洛维多夫)
- 固定构建在Mac OS卡特琳娜。 [#8600](#) (meo)
- 增加编解码器性能测试中的行数，以使结果显着。 [#8574](#) (瓦西里*内姆科夫)
- 在调试版本中，处理 LOGICAL_ERROR 异常作为断言失败，使得它们更容易被注意到。 [#8475](#) (Alexander Kuzmenkov)
- 使与格式相关的性能测试更具确定性。 [#8477](#) (阿列克谢-米洛维多夫)
- 更新 lz4 来修复记忆消毒器的故障 [#8181](#) (Alexander Kuzmenkov)
- 在异常处理中抑制已知MemorySanitizer误报。 [#8182](#) (Alexander Kuzmenkov)
- 更新 gcc 和 g++ 到版本9在 build/docker/build.sh [#7766](#) (TLightSky)
- 添加性能测试用例来测试 PREWHERE 比 WHERE. [#7768](#) (阿莫斯鸟)
- 在修复一个笨拙的测试方面取得了进展。 [#8621](#) (阿列克谢-米洛维多夫)
- 避免从MemorySanitizer报告数据 libunwind. [#8539](#) (阿列克谢-米洛维多夫)
- 更新 libc++ 到最新版本。 [#8324](#) (阿列克谢-米洛维多夫)
- 从源头构建ICU库。这修复 [#6460](#). [#8219](#) (阿列克谢-米洛维多夫)

- 从切换 `libressl` 到 `openssl`. ClickHouse应在此更改后支持TLS1.3和SNI。这修复 #8171. #8218 (阿列克谢-米洛维多夫)
- 使用时固定的UBSan报告 `chacha20_poly1305` 从SSL (发生在连接到`https://yandex.ru/`)。#8214 (阿列克谢-米洛维多夫)
- 修复默认密码文件的模式 `.deb linux`发行版。#8075 (proller)
- 改进的表达式获取 `clickhouse-server` PID输入 `clickhouse-test`. #8063 (亚历山大*卡扎科夫)
- 更新contrib/gtest到v1.10.0。#8587 (Alexander Burmak)
- 修复了ThreadSanitizer报告 `base64` 图书馆. 还将此库更新到最新版本，但无关紧要。这修复 #8397. #8403 (阿列克谢-米洛维多夫)
- 修复 `00600_replace_running_query` 对于处理器。#8272 (尼古拉*科切托夫)
- 删除支持 `tcmalloc` 为了使 `CMakeLists.txt` 更简单 #8310 (阿列克谢-米洛维多夫)
- 发布海湾合作委员会构建现在使用 `libc++` 而不是 `libstdc++`. 最近 `libc++` 只与叮当一起使用。这将提高构建配置的一致性和可移植性。#8311 (阿列克谢-米洛维多夫)
- 使用MemorySanitizer启用ICU库进行构建。#8222 (阿列克谢-米洛维多夫)
- 禁止从警告 `CapNProto` 图书馆. #8224 (阿列克谢-米洛维多夫)
- 删除代码的特殊情况 `tcmalloc`，因为它不再受支持。#8225 (阿列克谢-米洛维多夫)
- 在CI coverage任务中，优雅地终止服务器以允许它保存coverage报告。这修复了我们最近看到的不完整的复盖率报告。#8142 (阿利沙平)
- 针对所有编解码器的性能测试 `Float64` 和 `UInt64` 值。#8349 (瓦西里*内姆科夫)
- `termcap` 非常不推荐使用，并导致各种问题 (f.g.missing “up” 帽和呼应 `^J` 而不是多行)。帮个忙 `terminfo` 或bund 绑 `ncurses`. #7737 (阿莫斯鸟)
- 修复 `test_storage_s3` 集成测试。#7734 (尼古拉*科切托夫)
- 碌莽禄Support: `StorageFile(<format>, null)` 将块插入给定格式的文件而不实际写入磁盘。这是性能测试所必需的。#8455 (阿莫斯鸟)
- 添加参数 `--print-time` 功能测试打印每个测试的执行时间。#8001 (尼古拉*科切托夫)
- 添加断言 `KeyCondition` 同时评估RPN。这将修复来自gcc-9的警告。#8279 (阿列克谢-米洛维多夫)
- 在CI构建中转储cmake选项。#8273 (Alexander Kuzmenkov)
- 不要为某些fat库生成调试信息。#8271 (阿列克谢-米洛维多夫)
- 赖眉露>> `log_to_console.xml` 始终登录到stderr，无论它是否交互。#8395 (Alexander Kuzmenkov)
- 删除了一些未使用的功能 `clickhouse-performance-test` 工具 #8555 (阿列克谢-米洛维多夫)
- 现在我们也将搜索 `lld-X` 与相应的 `clang-X` 版本。#8092 (阿利沙平)
- 实木复合地板建设改善。#8421 (马苏兰)
- 更多海湾合作委员会警告 #8221 (kreuzerkrieg)
- Arch Linux的软件包现在允许运行ClickHouse服务器，而不仅仅是客户端。#8534 (Vladimir Chebotarev)
- 修复与处理器的测试。微小的性能修复。#7672 (尼古拉*科切托夫)

- 更新contrib/protobuf。 #8256 (Matwey V.Kornilov)
- 在准备切换到c++20作为新年庆祝活动。“May the C++ force be with ClickHouse.” #8447 (阿莫斯鸟)

实验特点

- 增加了实验设置 `min_bytes_to_use_mmap_io`. 它允许读取大文件，而无需将数据从内核复制到用户空间。默认情况下禁用该设置。建议的阈值大约是64MB，因为mmap/munmap很慢。#8520 (阿列克谢-米洛维多夫)
- 返工配额作为访问控制系统的一部分。增加了新表 `system.quotas`，新功能 `currentQuota`, `currentQuotaKey`，新的SQL语法 `CREATE QUOTA`, `ALTER QUOTA`, `DROP QUOTA`, `SHOW QUOTA`. #7257 (维塔利*巴拉诺夫)
- 允许跳过带有警告的未知设置，而不是引发异常。#7653 (维塔利*巴拉诺夫)
- 重新设计的行策略作为访问控制系统的一部分。增加了新表 `system.row_policies`，新功能 `currentRowPolicies()`，新的SQL语法 `CREATE POLICY`, `ALTER POLICY`, `DROP POLICY`, `SHOW CREATE POLICY`, `SHOW POLICIES`. #7808 (维塔利*巴拉诺夫)

安全修复

- 修正了读取目录结构中的表的可能性 `File` 表引擎。这修复 #8536. #8537 (阿列克谢-米洛维多夫)

更新日志2019

Was this content helpful? RATING_STARS	Rating: RATING_VALUE - RATING_COUNT votes
---	--

©2016-2021 ClickHouse, Inc.
Built from 269a58373a