ClickHouse

数据压缩是怎么做的，意义是什么

扫描数据，重复的数据不要了，用一些offset来替代，网络传输的时候数据量也会比较小

helloworld和low ，low就是helloworld\_(7,3),意思就行，往前7个单位，可以匹配上3个重复的字符，压缩速度很快，压缩比例就会很低，有些算法压缩很慢，但是压缩比例很高，

行式存储的数据，重复的部分不多，所以压缩效率不高，列式存储，一个文件里就一个字段的值，很多是重复的，此时压缩效率会比较高

cpu的向量化执行

cpu在处理数据的时候，从内存里读写的性能大致100ns，1ms=1000us，1us=1000ns，1ns=1000ps

clickhouse，用上了SIMD指令

Multi--Master多主架构和数据分片

并不是master-slave架构，es、redis、mysql、hdfs、hbase都是master-salve架构，

大多数的分布式系统，其实都是master-salve架构的

hdfs，namenode->多个datanode

hbase，hmaster->多个hregionserver

es，多个节点选举出一个master->集群其他节点

redis，mysql

clickhouse是多主架构的，有很多个主节点，多机器上部署的节点，每个都是master，跟eureka，peer-to-peer，每个节点扮演的角色都是一样的，集群里并不存在主控节点。

栗子：

hdfs，分片，把一个大文件拆分为很多的小块，拆分小块的过程就是一个分片的过程，不同的小块放在不同的服务器上存储，数据分片+分布式存储

hbase，底层就是基于hdfs的，region的拆分其实就是一个数据分片的过程，不同的region放不同的服务器上管理，分布式管理的过程，region已经是一个数据分片了，粗粒度的，region的数据hfile，也是放在hdfs上的，本身就是拆分为很多block小块。分布式存储在多台服务器上的

ES，索引，shard，不同的索引可以拆分为很多的shard，就是一个数据分片的过程，不同的shard就放在不同的服务器上

REDIS，redis cluster，redis多实例集群，把数据分散在redis各个服务器上，也是一种数据分片

Mysql，数据分片就是分库分表的一个过程，不同的服务器上的库里和不同的表里

以上，一个服务器可以放很多数据分片

clickhouse，一个分片就对应了一个节点，有多少个节点就有多少个分片，一开始可以就是用一个本地表和一个分片，后续可以增加分片

几百毫秒，实时灌入，实时分析的过程，sparkssql基本上都是分钟级别，

设置副本的概念，

内核级数据模型分析

DataType，用于进行序列化和反序列化

底层依赖Column和Field，clickhouse擅长的就是对表里面某几个字段进行聚合分析，不太擅长单行的

OLTP，本质就是要对高效的对单行数据做CRUD，事务，还要支持快速的根据多个事件检索数据，而不是跑聚合函数来分析

MergeTree引擎表的创建和存储过程分析

order by 指定的是排序建，是必填的 ，指定在一个数据片段里根据什么来排序，一般情况下把逐渐

setings

如果有分区建，那么每个分区是一个目录，分区目录里面，默认会包含.mrk文件，如果有分区，就会还有额外的partittion.dat,mi.idx,如果建立了二级索引

checksums.txt，是放校验和的文件，

columns.txt,是放所有的字段信息的

primary.idx,是基于主键建立的一级索引

.bin,是列式存储的数据文件

.mrk,标记文件，记录了.bin文件里的各个压缩块的偏移量

partition..dat是放的分区字段值，

skp两个文件，，，都是二级索引文件

你往mergetree里灌入数据的时候，会根据分区建去组织数据，但是其实一个分区是会有多个目录的，最后一个0是合并层级，实际上默认每隔一段时间，会对一个分区的多个目录做合并

合并完可能会有202001010——1——2——1·这样的东西，

写入过程

进行一个向上的归并，对每个数据区间排序过后，name.

第二数据区间，字段A.bin,一合并发现超过64k，合并成一个压缩块

字段A的数据，放在字段A.bin压缩文件中

后面的每个数据区间对应的都是这样的思路

字段A的数据，压缩块1，夸夸夸各种跳转

\_MergeTree的数据查询过程分析

单机clickhouse查询怎么查，分区，一定要根据查询条件建立一些联合 索引，针对查询where条件的字段都建好索引 ，直接根据数据结构裤裤筛，建好了索引之后，每个数据区间里面，

先定位出来数据区间，

然后定位要对哪些字段进行数据分析，找到字段对应的压缩块所在的位置

再到.bin文件对应的位置压缩块进行解压缩，拿到内存再进行分组，聚合，

然后就是纯内存的计算，

数据写入以及查询原理回顾

分成一个一个的数据区间，然后构成主键索引，数据区间凑满64k写到一个压缩块里去，再对某个字段在哪个压缩块里进行标记，磁盘上存的是稀疏索引，然后构建一个索引的树，字段A

数据TTl机制的支持介绍

MergeTree这个家族的引擎都 讲一下，分别有什么特点和机制，

Replaceing MergeTree

基于order by后面的字段进行去重的，如果根据多个 字段进行去重的，

summingMergeTree：为数据统计而生的引擎

在合并的时候，按照预定义的order by表达式进行合并汇总，把多行数据合并为一行，后续查询的时候，数据量会很小，速度会很快

AggreatingMergeTree：更为强大的预聚合引擎

summingMergeTree只能预先进行sum累加，不够强大，AggreatingMergeTree可以预先进行自定义函数的聚合，基本就跟kylin的思路类似了

优化视图，物化视图，

clickhous的分布式存储+SQL查询=HDFS+HBase+Phoneix

collapsingMergeTree：支持行级修改和删除

collapsingMergeTree可以支持行级力度的修改和删除，实时数据分析，可以用这个定义底表，实时数据流进行预聚合

合并的时候会触发啊上述的数据删除和重新插入

自己去拓展学习clickhouse

索引，数据存储的原理
其他的一些表引擎，查询语法，运维管理，分布式架构
挑选合适的表引擎，建表
​

华润万家

mysql 里left join，会创建临时表

一个数据块有8092个数据，建立索引1，然后下一个数据块，建立索引2.

分布式能力依赖mergetree实现的

同样满足两种特性怎做？监听节点，实时的监听binlog，会在proxy这里转发到click'house这一层。数据一致性，这张表会有一个隐藏的id类型，

列数据库

mergetree 引擎表的创建和存储结构

建表语句

partition by指定的是分区键，每一天的数据都是在一个分区里面

orderby

primary key

如果建立了二级索引，分区目录里，还有二级索引的.mrk文件

标记条目1：压缩块0在.bin数据文件中的偏移量

索引条目1：第0行数据的主键值

mergetree写入过程分析

在内存里划分一个个数据区间，对各个字段的值提出来，判断是否大于64k，判断对应.bin里的哪个压缩块

凑够指定的markrange之后，就会向上进行归并和合并

每隔一段时间，同一个分区的目录可能会被合并，数据被压缩

《126\_授人以渔：教你自己去拓展学习Clickhouse（2）》

其他的一些表引擎，查询语法，运维管理，分布式架构

你如果要用好clickhouse，了解清楚他的各种表引擎的原理和特性，MergeTree家族引擎，原理大家都已经知道了，对表引擎的工作原理熟悉的基础之上去用他，你大概知道用这个表引擎建了表之后，查询的时候是如何工作的

如何去建立索引，如何分区，如何做优化，如何让他的性能是最高的

《124\_CollapsingMergeTree：支持行级修改和删除》

CollapsingMergeTree可以支持行级粒度的修改和删除，不过这个一般在离线批量分析型场景里用的很少，一般离线批量分析型场景，都是每天或者定时把一批数据灌入clickhouse，然后就可以用AggregatingMergeTree物化视图定义好预聚合，再查询分析就可以了

但是如果要用clickhouse做实时数据分析，那么可以用这个CollapsingMergeTree来定义底表，也就是明细表，实时数据流对clickhouse做实时的增删改，然后定义AggregatingMergeTree来进行预聚合，可以做实时数据分析

CREATE TABLE t1(

id String,

c1 Int32,

sign Int8,

gmt\_created DateTime

) ENGINE = CollapsingMergeTree(sign)

PARTITION BY toYYYYMM(gmt\_created)

ORDER BY id

插入数据：INSERT INTO TABLE t1 VALUES(‘001’, 100, 1, ‘2020-01-01 01:00:00’)

删除数据：INSERT INTO TABLE t1 VALUES(‘001’, 100, -1, ‘2020-01-01 01:00:00’)

插入修改后的数据：INSERT INTO TABLE t1 VALUES(‘001’, 150, 1, ‘2020-01-01 01:00:00’)

在合并的时候就会触发上述的根据sign的数据删除和重新插入

查询改写为：SELECT id, SUM(c1 \* sign) FROM t1 GROUP BY id HAVING SUM(sign) > 0

《099\_动手操作一下Clickhouse的安装和部署（1）》

先按照以前的方式去装一台linux虚拟机，接着可以执行下面的安装操作

Clickhouse的rpm安装包下载

https://packagecloud.io/altinity/clickhouse

clickhouse-client-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-common-static-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-common-19.17.4.11-1.el7.x86\_64.rpm

检查linux虚拟机的cpu是否支持SSE 4.2指令集，那是向量化引擎需要的

grep -q sse4\_2 /proc/cpuinfo && echo "SSE 4.2 supported" || echo "SSE 4.2 not supported"

设置FQDN：hostnamectl –static set-hostname clickhouse-01

验证是否生效：hostname -f

配置hosts文件：vi /etc/hosts，本机ip地址 clickhouse-01

上传4个rpm安装包到/usr/local目录下去

rpm -ivh ./\*.rpm，全部安装，可能会出现一些依赖包缺失，此时缺失什么就百度一下，然后先安装依赖就可以了，yum install -y libicu.x86\_64

安装完毕之后，clickhouse会在服务器上构建一套目录结构

/etc/clickhouse-server：配置文件，config.xml全局配置和users.xml用户配置

/var/lib/clickhouse：数据存储目录，可以修改

/var/log/clickhouse-server：日志目录，可以修改

/etc/security/limits.d/clickhouse.conf：文件句柄数量配置，默认是262144，可以修改文件句柄，在config.xml里可以通过max\_open\_files来指定

/etc/cron.d/clickhouse-server：crontab定时任务配置，主要是用来恢复异常中断的clickhouse进程的，每隔10s会执行一次定时任务，尝试启动clickhouse进程，如果进程在运行就算了，否则就自动启动

/usr/bin目录下有一些可执行命令：clickhouse、clickhouse-client、clickhouse-server，等等，分别是服务端和客户端

修改config.xml，<path>、<tmp\_path>、<user\_files\_path>，都可以修改一下

接着把你修改的目录设置为clickhouse有权限，chown clickhouse.clickhouse /usr/local/xxx -R

接着启动clickhouse：service clickhouse-server start，默认读取的是/etc/clickhouse-server/config.xml

clickhouse-client，进入客户端，show databases;

客户端支持TCP和HTTP，TCP一般是内部通信以及cli客户端，端口号是9000，HTTP是面向系统的，端口号是8123，cli客户端主要用于调试、测试以及运维，进入之后，就可以DDL、DML以及查询了

但是系统访问一般是用JDBC去使用的

<dependency>

<groupId>ru.yandex.clickhouse</groupId>

<artifactId>clickhouse-jdbc</artifactId>

<version>0.2.4</version>

</dependency>

Class.forNmae(“ru.yandex.clickhouse.clickHouseDriver”);

String url = “jdbc:clickhouse://clickhouse.zhss.com:8123/default”;

String user = “default”;

String password = “”;

Connection connection = DriverManager.getConnection(url, username, password);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executQuery(“SELECT xxx”);

resultSet.next();

resultSet.getInt(1);

数据类型：int、float、decimal、string、fexedstring、uuid、DateTime、DateTime64、Date、Array、Tuple、Enum、Nested、Nullable、Domain

CREATE DATABASE [IF NOT EXISTS] db\_name [ENGINE = engine]

引擎有5种：ordinary（默认）、dictionary、memory、lazy、mysql

创建数据库之后，就会在数据目录下有一个文件夹，metadata目录下也有恢复数据库的SQL文件，show databases可以查看到你创建的数据库

USE db\_name，DROP DATABASE [IF EXISTS] db\_name

其实说白了，这个都跟mysql一样的，不同的业务最好是建自己的库，起码逻辑上和物理上都是可以分开的

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name (

name1 [type] [DEFAULT|MATERALIZED|ALIAS expr],

name2 [type] [DEFAULT|MATERALIZED|ALIAS expr]

) ENGINE = engine

CREATE TABLE test\_table\_1 (

title String,

content String,

publish\_time DateTime

) ENGINE = Memory

CREATE TABLE [IF NOT EXISTS] [db\_name1.]table\_name AS [db\_name2.]table\_name2 [ENGINE = engine]，仅仅复制表结构

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name ENGINE = engine AS SELECT，复制表结构，同时数据都会复制一份

DESC table\_name，可以查看表结构

DROP TABLE [IF EXISTS] [db\_name.]table\_name

DEFAULT可以定义默认值

临时表：CREATE TEMPORARY TABLE，别的都一样，仅仅支持memory引擎，会话级别的，会话结束，表就销毁了，说白了，假设你系统需要用这个，就创建一个临时表，数据都在内存，等你程序运行完毕，表就没了，一般我们不怎么用

ALTER TABLE table\_name ADD COLUMN [IF NOT EXISTS] name [type] [default] [AFTER name\_after]，增加字段

ALTER TABLE table\_name MODIFY COLUMN [IF EXISTS] name [type] [default]，修改字段

ALTER TABLE table\_name DROP COLUMN [IF EXISTS] name，删除字段

RENAME TABLE [db\_name1.]table\_name1 TO [db\_name2.]table\_name2, ….，重命名

TRUNCATE TABLE [IF EXISTS] [db\_name.]table\_name，清空表数据

分区表：PARTITION BY toYYYYMM(publish\_time)

然后你插入数据，随着你不同的时间，他会把你数据放入不同的分区里去，不同分区的数据放在各自的目录里，一般在分析型数据库里，分区是很有用的，因为查询的时候指定分区，就可以仅仅扫描分区目录内的数据了

ALTER TABLE table\_name DROP PARTITION partition，删除某个分区

ALTER TABLE table\_name REPLACE PARTITION partition FROM table\_name\_b，复制别的表的分区数据，两个表的分区键相同，表结构相同，其实一般不怎么用这个

ALTER TABLE table\_anme CLEAR COLUMN column\_name IN PARTITION partition，重置分区数据

ALTER TABLE table\_name DETACH/ATTACH PARTITION partition，卸载和装载

视图，简单来说，分为逻辑视图和物理视图，逻辑视图就是一个逻辑上的概念而已，方便你查询用的，物理视图，就是一个物理上独立存储的数据

CREATE VIEW [IF NOT EXISTS] [db\_name.]view\_name AS SELECT

CREATE MATERALIZED VIEW [IF NOT EXISTS] [db\_name.]view\_name [ENGINE = engine] AS SELECT

INSERT INTO [db.]table [(c1,c2,c3…)] VALUES(v1,v2,v3…)，插入数据

数据可以删除和修改，但是在clickhouse这种分析型数据库里，是极为重的操作，他是异步的，而且无法回滚，一般不建议针对单条数据做，最好是批量删除和批量修改

ALTER TABLE [db\_name].table\_name DELETE WHERE filter\_expr，批量删除数据，他会重新弄一个新的目录，里面放入删除后的数据，老的目录还不会立即删除，所以他很重！

ALTER TABLE [db\_name.]table\_name UPDATE col1=val1，修改也是同样的

《132\_Clickhouse的集群架构与副本机制介绍（4）》

之前讲的都是单节点的，不过单节点的理解了，其实多节点的集群、副本、分片，也都是差不多的，因为基本都这么回事儿

然后首先是可以部署多个clickhous节点组成集群，这个是没问题的

其次就是先讲副本的概念，副本的话，主要是通过ReplicatedMergeTree引擎实现的，说着说，之前讲的各种引擎加上Replicated关键字都可以转化为支持副本的表引擎，就这么个意思

这个副本机制是通过zk来实现的，不过个人认为这是一个败笔，因为我认为一个优秀的分布式系统，不应该随意依赖外部，除非你自己就是觉得麻烦，一定要依赖，否则一般我们都是自行开发分布式系统的协调机制的

ReplicatedMergeTree表引擎会通过zk去协同，把数据同步到其他节点上去，形成一个副本，在表级别可以对副本进行配置，比如副本的数量以及在集群里的分布位置，而且Clickhouse支持副本的多主架构，任何一个副本都可以执行增删改，各个副本会互相同步

要玩儿clickhouse集群架构和副本设置，就得先启动多个clickhouse节点，每个节点都配置一下zk，在/etc/clickhouse-server/config.d目录下，创建一个metrika.xml：

<?xml version=”1.0”?>

<yandex>

<zookeeper-servers>

<node index=”1”>

<host>xx.xx.xx </host>

<port>2181</port>

</node>

</zookeeper-servers>

</yandex>

然后在config.xml里加入一段配置：

<include\_form>/etc/clickhouse-server/config.d/metrika.xml</inclucde\_form>

<zookeeper incl=”zookeeper-servers” optional=”false” />

加入这个配置之后，启动clickhouse集群就可以了，集群之间会通过同一个zk进行协调，而且clickhouse提供了zk内置表，select \* from system.zookeeper where path=’/’，就可以查到zk里的指定目录下的数据

select name,value,czxid,mzxid from system.zookeeper where path=’/clickhouse’

然后就可以创建支持副本的表了

ENGINE = ReplicatedMergeTree(‘zk\_path’, ‘replica\_name’)

zk\_path指的是这个表在zk里的路径，一般是/clickhouse/tables/{shard}/table\_name，这里shard是你的分片编号，比如01，02之类的，然后就是表名了，刚开始你可以建一个分片的表

后面是有一个replica\_name就是副本的名称

举个例子，那你在clickhouse一个节点上可以创建一个表，引擎的话，那就是zk\_path就是定义的是这个表的哪个分片了，可以就一个分片，那么分片编号都相同，那不同机器上可以定义不同的副本名称了

副本名称一般用的是机器的hostname

ENGINE = ReplicatedMergeTree(‘/clickhouse/tables/01/test\_table’, ’192.168.31.155’)

ENGINE = ReplicatedMergeTree(‘/clickhouse/tables/01/test\_table’, ’192.168.31.273’)

create table table\_name

ENGINE = ReplicatedMergeTree(‘01’, ’01’)

/clickhouse/tables/table\_name/01

/clickhouse/tables/table\_name/02

/clickhouse/tables/table\_name/03

第一个副本的节点的值，192.168.31.155

如果要真的是采用了zk来作为一个集中化的集群元数据管理的地方，完全没必要自己手工去建表指定各种分片和副本

你可以提供一个客户端的工具，连接到clickhouse任意一个节点就可以了，然后执行一个建表的命令：

create table table\_name

ENGINE = ReplicatedMergeTree(3, 1)

《122\_AggregatingMergeTree：更为强大的预聚合引擎（1）》

SummingMergeTree只能预先进行sum累加，不够强大，AggregatingMergeTree可以预先进行自定义函数的聚合，基本就跟kylin的思路类似了

CREATE TABLE t1(

id String,

c1 String,

c2 AggregateFunction(uniq, String),

c3 AggregateFunction(sum, UInt32),

gmt\_created DateTime

)

ENGINE = AggregatingMergeTree()

PARTITION BY toYYYYMM(gmt\_created)

ORDER BY (id, c1)

PRIMARY KEY id

上面那种搞法就会在合并的时候，根据id和c1分组聚合，对c2和c3按照你指定的函数聚合了，一个是uniq唯一函数，一个是sum累加函数

INSERT INTO TABLE t1

SELECT ‘001’,’c1’, uniqState(‘c2’), sumState(toUInt32(300)), ‘2020-01-01 01:00:00’，插入必须用这种select语法

SESLECT id, c1, uniqMerge(c2), sumMerge(c3) FROM t1 GROUP BY id,c1，查询必须用这种语法就可以了

《116\_MergeTree的数据写入过程分析（2）》

写入数据，先确定分区，当然一个分区可能有多个目录了，分区的多个目录可能会合并，写入的时候先在内存里，按照8192这个粒度划分为多个数据区间，每个数据区间都是8192条数据，区间都有start和end的数据offset，比如0~8192，8192~16384

然后每获取到一个数据区间，就会在primary.idx一级索引里写入一条稀疏索引数据，比如说第一个区间就是索引条目0，索引值就是第0行数据，第二个区间对应的是索引条目1，索引值是第8192行数据的值，第三个区间对应的是索引条目2，索引值是第16384行数据的值，以此类推

而且每次凑够一个数据区间，除了在primary.idx里写入一条稀疏索引，还得在各个字段的.mrk标记文件里写入一条标记数据，这个一个数据区间对应一条稀疏索引，也对应各个字段的.mrk标记文件里的一条标记数据，稀疏索引和标记数据是一一对应的

.mrk标记文件里，实际上写入的是当前这个区间的数据在这个字段的.bin数据压缩文件里的压缩块的偏移量

比如说，第一个数据区间的所有字段A的数据，没有凑够64KB，那么他铁定是属于.bin数据压缩文件里的压缩快0了，那么这个数据区间对应的字段A的.mrk文件的标记条目0记录的值，其实就是字段A的.bin文件里的压缩块0的偏移量了

接着第二个数据区间会在primary.idx里写一条稀疏索引条目，索引值就是第二个区间起始数据行（8192行）的主键值，同时会在各个字段的.mrk标记文件里，去写入一个标记条目1，那么标记条目1的值是什么呢？

现在假设两个数据区间内的字段A的数据量已经超过了64kb，小于1mb，此时就会把这波数据写为一个压缩块，压缩块放入字段A的.bin文件里去，所以第二个数据区间对应的字段A的.mrk文件里的标记条目1的值，也是字段A的.bin文件里的压缩块0的偏移量

后续的数据区间都是以此类推的

首先要在内存里凑够一个8192条数据的数据区间，然后就是写primary.idx里的一级索引条目的值，做一个稀疏索引，接着各个字段都得写.mrk标记文件，此时这个数据区间一定在各个字段的.mrk标记文件里会有一个标记条目

标记条目的值，得判断一下，就是这个数据区间的各个字段的值大小，是在各个字段的.bin文件里的第几个压缩块里，把那个压缩块的偏移量作为那些字段的标记条目的值

《121\_SummingMergeTree：为数据统计而生的引擎》

如果用clickhouse主要是进行group by后的统计分析，那其实就没必要每次都基于MergeTree来跑，因为每次楼出来一大批数据，然后内存group by之类的，也是很耗时的，数据量大的话，搞到秒级也有可能

所以完全可以用SummingMergeTree来跑分析查询，他会在合并的时候，按照预定义的order by表达式进行数据合并汇总，把多行数据合并为一行，那么后续你实际查询的时候，数据量会很小，速度会很快

CREATE TABLE t1(

id String,

c1 String,

c2 UInt32,

c3 Float64,

gmt\_created DateTime

)

ENGINE = SummingMergeTree()

PARTITION BY toYYYYMM(gmt\_created)

ORDER BY (id, c1)

PRIMARY KEY (id)

在合并的时候，就会在同一个分区内根据ORDER BY的组合进行分组聚合了，会把c2和c3字段默认进行累加聚合操作

《120\_ReplacingMergeTree：依据主键进行数据去重》

普通的MergeTree引擎，是允许主键数据重复的，所以如果你要实现一个主键数据不重复的效果，就得用ReplacingMergeTree引擎，这都是MergeTree的变种引擎，核心思路是差不多的，但是有一些变种

CREATE TABLE t1(

id String,

gmt\_created DateTime

)

ENGINE = ReplacingMergeTree()

PARTITION BY toYYYYMM(gmt\_created)

ORDER BY id

PRIMARY KEY id

这个会基于ORDER BY后面的字段进行去重的，如果你要根据多个字段进行去重，就可以ORDER BY(c1, c2)这样子

这样你如果往表里灌入很多主键重复的数据，他在触发分区目录合并的时候就会去重了，也可以用optimize命令强制合并看效果，但是他仅仅是对相同的分区目录合并的时候会去重，但是如果是跨分区的重复数据是不会去重的

《133\_Clickhouse副本机制底层原理揭秘（1）》

首先，在两个机器上创建互为副本的表，语法都讲过了，表名和分片一致，但是副本名是各自的hostname，第一台机器，先会在zk的/replicas下注册自己的副本节点，其实就是自己的hostname那个节点，然后启动监听任务，监听/log节点下的变动，最后就是往/leader\_election下插入子节点选举主副本，如果是第一个就是主副本

第二台机器，也会干一样的事儿，只不过往/leader\_election下插入子节点，发现自己不是主副本罢了

接着在第一个机器上insert进去数据，此时先会执行之前讲过的逻辑和机制，在机器本地写入数据，包括数据分区，数据压缩块，索引构建，数据标记，等等吧，接着会在zk的/blocks节点下写入数据分区的block\_d

此时有一个重要的参数，就是insert\_quorum，默认是0，也就是一个副本写入成功就ok了，如果你设置为1，那么他必须要等待另外一个副本也写成功才行，如果要让另外一台机器写入副本，核心是在zk的/log节点下写入一个LogEntry，里面记录了另外一台机器需要拉取的数据是哪些

然后第二个机器因为一直在监听zk的/log节点下的变动，所以此时会发现有变动，就会发现有新的数据要拉取，他就会把这个数据拉取任务房到队列里去，异步进行拉取，他会遍历zk的/replicas下的各个副本节点，从里面选一个数据最新的节点

接着就从那个节点去下载自己需要的数据就可以了

不过这里吐槽一下，个人认为这种副本机制基于zk来搞，弄的过于复杂，一点都不好，还有异步执行之类的时效问题

其他的一些操作基本大致都是差不多的，他说白了就是通过这种方式去进行副本数据的同步罢了

《135\_Clickhouse副本机制底层原理揭秘（3）》

首先，在两个机器上创建互为副本的表，语法都讲过了，表名和分片一致，但是副本名是各自的hostname，第一台机器，先会在zk的/replicas下注册自己的副本节点，其实就是自己的hostname那个节点，然后启动监听任务，监听/log节点下的变动，最后就是往/leader\_election下插入子节点选举主副本，如果是第一个就是主副本

第二台机器，也会干一样的事儿，只不过往/leader\_election下插入子节点，发现自己不是主副本罢了

接着在第一个机器上insert进去数据，此时先会执行之前讲过的逻辑和机制，在机器本地写入数据，包括数据分区，数据压缩块，索引构建，数据标记，等等吧，接着会在zk的/blocks节点下写入数据分区的block\_d

此时有一个重要的参数，就是insert\_quorum，默认是0，也就是一个副本写入成功就ok了，如果你设置为1，那么他必须要等待另外一个副本也写成功才行，如果要让另外一台机器写入副本，核心是在zk的/log节点下写入一个LogEntry，里面记录了另外一台机器需要拉取的数据是哪些

然后第二个机器因为一直在监听zk的/log节点下的变动，所以此时会发现有变动，就会发现有新的数据要拉取，他就会把这个数据拉取任务房到队列里去，异步进行拉取，他会遍历zk的/replicas下的各个副本节点，从里面选一个数据最新的节点

接着就从那个节点去下载自己需要的数据就可以了

不过这里吐槽一下，个人认为这种副本机制基于zk来搞，弄的过于复杂，一点都不好，还有异步执行之类的时效问题

其他的一些操作基本大致都是差不多的，他说白了就是通过这种方式去进行副本数据的同步罢了

《129\_Clickhouse的集群架构与副本机制介绍（1）》

之前讲的都是单节点的，不过单节点的理解了，其实多节点的集群、副本、分片，也都是差不多的，因为基本都这么回事儿

然后首先是可以部署多个clickhous节点组成集群，这个是没问题的

其次就是先讲副本的概念，副本的话，主要是通过ReplicatedMergeTree引擎实现的，说着说，之前讲的各种引擎加上Replicated关键字都可以转化为支持副本的表引擎，就这么个意思

这个副本机制是通过zk来实现的，不过个人认为这是一个败笔，因为我认为一个优秀的分布式系统，不应该随意依赖外部，除非你自己就是觉得麻烦，一定要依赖，否则一般我们都是自行开发分布式系统的协调机制的

ReplicatedMergeTree表引擎会通过zk去协同，把数据同步到其他节点上去，形成一个副本，在表级别可以对副本进行配置，比如副本的数量以及在集群里的分布位置，而且Clickhouse支持副本的多主架构，任何一个副本都可以执行增删改，各个副本会互相同步

要玩儿clickhouse集群架构和副本设置，就得先启动多个clickhouse节点，每个节点都配置一下zk，在/etc/clickhouse-server/config.d目录下，创建一个metrika.xml：

<?xml version=”1.0”?>

<yandex>

<zookeeper-servers>

<node index=”1”>

<host>xx.xx.xx </host>

<port>2181</port>

</node>

</zookeeper-servers>

</yandex>

然后在config.xml里加入一段配置：

<include\_form>/etc/clickhouse-server/config.d/metrika.xml</inclucde\_form>

<zookeeper incl=”zookeeper-servers” optional=”false” />

加入这个配置之后，启动clickhouse集群就可以了，集群之间会通过同一个zk进行协调，而且clickhouse提供了zk内置表，select \* from system.zookeeper where path=’/’，就可以查到zk里的指定目录下的数据

select name,value,czxid,mzxid from system.zookeeper where path=’/clickhouse’

然后就可以创建支持副本的表了

ENGINE = ReplicatedMergeTree(‘zk\_path’, ‘replica\_name’)

zk\_path指的是这个表在zk里的路径，一般是/clickhouse/tables/{shard}/table\_name，这里shard是你的分片编号，比如01，02之类的，然后就是表名了，刚开始你可以建一个分片的表

后面是有一个replica\_name就是副本的名称

举个例子，那你在clickhouse一个节点上可以创建一个表，引擎的话，那就是zk\_path就是定义的是这个表的哪个分片了，可以就一个分片，那么分片编号都相同，那不同机器上可以定义不同的副本名称了

副本名称一般用的是机器的hostname

《123\_AggregatingMergeTree：更为强大的预聚合引擎（2）》

SummingMergeTree只能预先进行sum累加，不够强大，AggregatingMergeTree可以预先进行自定义函数的聚合，基本就跟kylin的思路类似了

clickhouse的实时数据批量写入 + 明细数据增删改 = HDFS + HBase

clickhouse的分布式存储 + SQL查询 -> 实时数据即席查询 = HDFS + HBase + Phoenix = HDFS + Hive + Presto

clickhouse的预聚合引擎 -> OLAP分析 = HDFS + HBase + MapReduce + Kylin

CREATE TABLE t1(

id String,

c1 String,

c2 AggregateFunction(uniq, String),

c3 AggregateFunction(sum, UInt32),

gmt\_created DateTime

)

ENGINE = AggregatingMergeTree()

PARTITION BY toYYYYMM(gmt\_created)

ORDER BY (id, c1)

PRIMARY KEY id

上面那种搞法就会在合并的时候，根据id和c1分组聚合，对c2和c3按照你指定的函数聚合了，一个是uniq唯一函数，一个是sum累加函数

INSERT INTO TABLE t1

SELECT ‘001’,’c1’, uniqState(‘c2’), sumState(toUInt32(300)), ‘2020-01-01 01:00:00’，插入必须用这种select语法

SESLECT id, c1, uniqMerge(c2), sumMerge(c3) FROM t1 GROUP BY id,c1，查询必须用这种语法就可以了

不过一般用起来都是基于物化视图的，把这种汇总表创建为底层明细表的物化视图

CREATE TABLE t1\_detail(

id String,

c1 String,

c2 String,

c3 UInt32,

gmt\_created DateTime

) ENGINE = MergeTree()

PARTITION BY toYYYYMM(gmt\_created)

ORDER BY (id, c1)

CREATE MATERIALIZED VIEW t1\_aggr\_view

ENGINE = AggregatingMergeTree()

PARTITION BY toYYYYMM(gmt\_created)

ORDER BY (id, c1)

AS SELECT

id,

city,

uniqState(c2) AS c2,

sumState(c3) AS c3

FROM t1\_detail

GROUP BY id, c1

插入数据直接在明细表插入就可以了，然后数据会同步到物化视图里去，接着按照引擎定义的聚合查询语句进行聚合，然后查询的时候，就直接从物化视图里查询就可以了：SELECT id,c1,uniqMerge(c2), sumMerge(c3) FROM t1\_aggr\_view GROUP BY id, c1

《101\_动手操作一下Clickhouse的安装和部署（3）》

先按照以前的方式去装一台linux虚拟机，接着可以执行下面的安装操作

Clickhouse的rpm安装包下载

https://packagecloud.io/altinity/clickhouse

clickhouse-client-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-common-static-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-common-19.17.4.11-1.el7.x86\_64.rpm

检查linux虚拟机的cpu是否支持SSE 4.2指令集，那是向量化引擎需要的

grep -q sse4\_2 /proc/cpuinfo && echo "SSE 4.2 supported" || echo "SSE 4.2 not supported"

设置FQDN：hostnamectl –static set-hostname clickhouse-01

验证是否生效：hostname -f

配置hosts文件：vi /etc/hosts，本机ip地址 clickhouse-01

上传4个rpm安装包到/usr/local目录下去

rpm -ivh ./\*.rpm，全部安装，可能会出现一些依赖包缺失，此时缺失什么就百度一下，然后先安装依赖就可以了，yum install -y libicu.x86\_64

安装完毕之后，clickhouse会在服务器上构建一套目录结构

/etc/clickhouse-server：配置文件，config.xml全局配置和users.xml用户配置

/var/lib/clickhouse：数据存储目录，可以修改

/var/log/clickhouse-server：日志目录，可以修改

/etc/security/limits.d/clickhouse.conf：文件句柄数量配置，默认是262144，可以修改文件句柄，在config.xml里可以通过max\_open\_files来指定，linux里一切皆文件

/etc/cron.d/clickhouse-server：crontab定时任务配置，主要是用来恢复异常中断的clickhouse进程的，每隔10s会执行一次定时任务，尝试启动clickhouse进程，如果进程在运行就算了，否则就自动启动

/usr/bin目录下有一些可执行命令：clickhouse、clickhouse-client、clickhouse-server，等等，分别是服务端和客户端

修改config.xml，<path>、<tmp\_path>、<user\_files\_path>，都可以修改一下，接着把你修改的目录设置为clickhouse有权限，chown clickhouse.clickhouse /usr/local/xxx -R

接着启动clickhouse：service clickhouse-server start，默认读取的是/etc/clickhouse-server/config.xml

clickhouse-client，进入客户端，show databases;

客户端支持TCP和HTTP，TCP一般是内部通信以及cli客户端，端口号是9000，HTTP是面向系统的，端口号是8123，cli客户端主要用于调试、测试以及运维，进入之后，就可以DDL、DML以及查询了

但是系统访问一般是用JDBC去使用的

<dependency>

<groupId>ru.yandex.clickhouse</groupId>

<artifactId>clickhouse-jdbc</artifactId>

<version>0.2.4</version>

</dependency>

Class.forNmae(“ru.yandex.clickhouse.clickHouseDriver”);

String url = “jdbc:clickhouse://clickhouse.zhss.com:8123/default”;

String user = “default”;

String password = “”;

Connection connection = DriverManager.getConnection(url, username, password);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executQuery(“SELECT xxx”);

resultSet.next();

resultSet.getInt(1);

数据类型：int、float、decimal、string、fexedstring、uuid、DateTime、DateTime64、Date、Array、Tuple、Enum、Nested、Nullable、Domain

CREATE DATABASE [IF NOT EXISTS] db\_name [ENGINE = engine]

引擎有5种：ordinary（默认）、dictionary、memory、lazy、mysql

创建数据库之后，就会在数据目录下有一个文件夹，metadata目录下也有恢复数据库的SQL文件，show databases可以查看到你创建的数据库

USE db\_name，DROP DATABASE [IF EXISTS] db\_name

其实说白了，这个都跟mysql一样的，不同的业务最好是建自己的库，起码逻辑上和物理上都是可以分开的

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name (

name1 [type] [DEFAULT|MATERALIZED|ALIAS expr],

name2 [type] [DEFAULT|MATERALIZED|ALIAS expr]

) ENGINE = engine

CREATE TABLE test\_table\_1 (

title String,

content String,

publish\_time DateTime

) ENGINE = Memory

CREATE TABLE [IF NOT EXISTS] [db\_name1.]table\_name AS [db\_name2.]table\_name2 [ENGINE = engine]，仅仅复制表结构

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name ENGINE = engine AS SELECT，复制表结构，同时数据都会复制一份

DESC table\_name，可以查看表结构

DROP TABLE [IF EXISTS] [db\_name.]table\_name

DEFAULT可以定义默认值

临时表：CREATE TEMPORARY TABLE，别的都一样，仅仅支持memory引擎，会话级别的，会话结束，表就销毁了，说白了，假设你系统需要用这个，就创建一个临时表，数据都在内存，等你程序运行完毕，表就没了，一般我们不怎么用

ALTER TABLE table\_name ADD COLUMN [IF NOT EXISTS] name [type] [default] [AFTER name\_after]，增加字段

ALTER TABLE table\_name MODIFY COLUMN [IF EXISTS] name [type] [default]，修改字段

ALTER TABLE table\_name DROP COLUMN [IF EXISTS] name，删除字段

RENAME TABLE [db\_name1.]table\_name1 TO [db\_name2.]table\_name2, ….，重命名

TRUNCATE TABLE [IF EXISTS] [db\_name.]table\_name，清空表数据

分区表：PARTITION BY toYYYYMM(publish\_time)

然后你插入数据，随着你不同的时间，他会把你数据放入不同的分区里去，不同分区的数据放在各自的目录里，一般在分析型数据库里，分区是很有用的，因为查询的时候指定分区，就可以仅仅扫描分区目录内的数据了

ALTER TABLE table\_name DROP PARTITION partition，删除某个分区

ALTER TABLE table\_name REPLACE PARTITION partition FROM table\_name\_b，复制别的表的分区数据，两个表的分区键相同，表结构相同，其实一般不怎么用这个

ALTER TABLE table\_anme CLEAR COLUMN column\_name IN PARTITION partition，重置分区数据

ALTER TABLE table\_name DETACH/ATTACH PARTITION partition，卸载和装载

视图，简单来说，分为逻辑视图和物理视图，逻辑视图就是一个逻辑上的概念而已，方便你查询用的，物理视图，就是一个物理上独立存储的数据

CREATE VIEW [IF NOT EXISTS] [db\_name.]view\_name AS SELECT

CREATE MATERALIZED VIEW [IF NOT EXISTS] [db\_name.]view\_name [ENGINE = engine] AS SELECT

INSERT INTO [db.]table [(c1,c2,c3…)] VALUES(v1,v2,v3…)，插入数据

数据可以删除和修改，但是在clickhouse这种分析型数据库里，是极为重的操作，他是异步的，而且无法回滚，一般不建议针对单条数据做，最好是批量删除和批量修改

ALTER TABLE [db\_name].table\_name DELETE WHERE filter\_expr，批量删除数据，他会重新弄一个新的目录，里面放入删除后的数据，老的目录还不会立即删除，所以他很重！

ALTER TABLE [db\_name.]table\_name UPDATE col1=val1，修改也是同样的

《100\_动手操作一下Clickhouse的安装和部署（2）》

先按照以前的方式去装一台linux虚拟机，接着可以执行下面的安装操作

Clickhouse的rpm安装包下载

https://packagecloud.io/altinity/clickhouse

clickhouse-client-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-common-static-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-common-19.17.4.11-1.el7.x86\_64.rpm

检查linux虚拟机的cpu是否支持SSE 4.2指令集，那是向量化引擎需要的

grep -q sse4\_2 /proc/cpuinfo && echo "SSE 4.2 supported" || echo "SSE 4.2 not supported"

设置FQDN：hostnamectl –static set-hostname clickhouse-01

验证是否生效：hostname -f

配置hosts文件：vi /etc/hosts，本机ip地址 clickhouse-01

上传4个rpm安装包到/usr/local目录下去

rpm -ivh ./\*.rpm，全部安装，可能会出现一些依赖包缺失，此时缺失什么就百度一下，然后先安装依赖就可以了，yum install -y libicu.x86\_64

安装完毕之后，clickhouse会在服务器上构建一套目录结构

/etc/clickhouse-server：配置文件，config.xml全局配置和users.xml用户配置

/var/lib/clickhouse：数据存储目录，可以修改

/var/log/clickhouse-server：日志目录，可以修改

/etc/security/limits.d/clickhouse.conf：文件句柄数量配置，默认是262144，可以修改文件句柄，在config.xml里可以通过max\_open\_files来指定，linux里一切皆文件

/etc/cron.d/clickhouse-server：crontab定时任务配置，主要是用来恢复异常中断的clickhouse进程的，每隔10s会执行一次定时任务，尝试启动clickhouse进程，如果进程在运行就算了，否则就自动启动

/usr/bin目录下有一些可执行命令：clickhouse、clickhouse-client、clickhouse-server，等等，分别是服务端和客户端

修改config.xml，<path>、<tmp\_path>、<user\_files\_path>，都可以修改一下

接着把你修改的目录设置为clickhouse有权限，chown clickhouse.clickhouse /usr/local/xxx -R

接着启动clickhouse：service clickhouse-server start，默认读取的是/etc/clickhouse-server/config.xml

clickhouse-client，进入客户端，show databases;

客户端支持TCP和HTTP，TCP一般是内部通信以及cli客户端，端口号是9000，HTTP是面向系统的，端口号是8123，cli客户端主要用于调试、测试以及运维，进入之后，就可以DDL、DML以及查询了

但是系统访问一般是用JDBC去使用的

<dependency>

<groupId>ru.yandex.clickhouse</groupId>

<artifactId>clickhouse-jdbc</artifactId>

<version>0.2.4</version>

</dependency>

Class.forNmae(“ru.yandex.clickhouse.clickHouseDriver”);

String url = “jdbc:clickhouse://clickhouse.zhss.com:8123/default”;

String user = “default”;

String password = “”;

Connection connection = DriverManager.getConnection(url, username, password);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executQuery(“SELECT xxx”);

resultSet.next();

resultSet.getInt(1);

数据类型：int、float、decimal、string、fexedstring、uuid、DateTime、DateTime64、Date、Array、Tuple、Enum、Nested、Nullable、Domain

CREATE DATABASE [IF NOT EXISTS] db\_name [ENGINE = engine]

引擎有5种：ordinary（默认）、dictionary、memory、lazy、mysql

创建数据库之后，就会在数据目录下有一个文件夹，metadata目录下也有恢复数据库的SQL文件，show databases可以查看到你创建的数据库

USE db\_name，DROP DATABASE [IF EXISTS] db\_name

其实说白了，这个都跟mysql一样的，不同的业务最好是建自己的库，起码逻辑上和物理上都是可以分开的

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name (

name1 [type] [DEFAULT|MATERALIZED|ALIAS expr],

name2 [type] [DEFAULT|MATERALIZED|ALIAS expr]

) ENGINE = engine

CREATE TABLE test\_table\_1 (

title String,

content String,

publish\_time DateTime

) ENGINE = Memory

CREATE TABLE [IF NOT EXISTS] [db\_name1.]table\_name AS [db\_name2.]table\_name2 [ENGINE = engine]，仅仅复制表结构

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name ENGINE = engine AS SELECT，复制表结构，同时数据都会复制一份

DESC table\_name，可以查看表结构

DROP TABLE [IF EXISTS] [db\_name.]table\_name

DEFAULT可以定义默认值

临时表：CREATE TEMPORARY TABLE，别的都一样，仅仅支持memory引擎，会话级别的，会话结束，表就销毁了，说白了，假设你系统需要用这个，就创建一个临时表，数据都在内存，等你程序运行完毕，表就没了，一般我们不怎么用

ALTER TABLE table\_name ADD COLUMN [IF NOT EXISTS] name [type] [default] [AFTER name\_after]，增加字段

ALTER TABLE table\_name MODIFY COLUMN [IF EXISTS] name [type] [default]，修改字段

ALTER TABLE table\_name DROP COLUMN [IF EXISTS] name，删除字段

RENAME TABLE [db\_name1.]table\_name1 TO [db\_name2.]table\_name2, ….，重命名

TRUNCATE TABLE [IF EXISTS] [db\_name.]table\_name，清空表数据

分区表：PARTITION BY toYYYYMM(publish\_time)

然后你插入数据，随着你不同的时间，他会把你数据放入不同的分区里去，不同分区的数据放在各自的目录里，一般在分析型数据库里，分区是很有用的，因为查询的时候指定分区，就可以仅仅扫描分区目录内的数据了

ALTER TABLE table\_name DROP PARTITION partition，删除某个分区

ALTER TABLE table\_name REPLACE PARTITION partition FROM table\_name\_b，复制别的表的分区数据，两个表的分区键相同，表结构相同，其实一般不怎么用这个

ALTER TABLE table\_anme CLEAR COLUMN column\_name IN PARTITION partition，重置分区数据

ALTER TABLE table\_name DETACH/ATTACH PARTITION partition，卸载和装载

视图，简单来说，分为逻辑视图和物理视图，逻辑视图就是一个逻辑上的概念而已，方便你查询用的，物理视图，就是一个物理上独立存储的数据

CREATE VIEW [IF NOT EXISTS] [db\_name.]view\_name AS SELECT

CREATE MATERALIZED VIEW [IF NOT EXISTS] [db\_name.]view\_name [ENGINE = engine] AS SELECT

INSERT INTO [db.]table [(c1,c2,c3…)] VALUES(v1,v2,v3…)，插入数据

数据可以删除和修改，但是在clickhouse这种分析型数据库里，是极为重的操作，他是异步的，而且无法回滚，一般不建议针对单条数据做，最好是批量删除和批量修改

ALTER TABLE [db\_name].table\_name DELETE WHERE filter\_expr，批量删除数据，他会重新弄一个新的目录，里面放入删除后的数据，老的目录还不会立即删除，所以他很重！

ALTER TABLE [db\_name.]table\_name UPDATE col1=val1，修改也是同样的

《117\_MergeTree的数据查询过程分析》

《113\_MergeTree中的列式数据存储以及数据标记（1）》

数据写入列式存储文件的时候，先对数据按照order by进行排序，为什么要排序呢？之前不是讲过索引了么，索引本身就是得按照顺序去生成的，而且还会生成多层级的数据范围，最后都是按照有序数据范围去扫描的

然后会用LZ4进行压缩，压缩过后，就会把压缩数据块写入.bin文件里去

首先在内存里，得按照8192这么个数据粒度对数据进行组织，组织成一个一个的数据范围，接着就看，一个范围的数据<=64KB，就会等待，一直等待到数据凑够64KB了再说，主要累及的数据在64KB到1MB之间，就会生成一个压缩块

如果一个数据区间大于1MB，就对他生成多个压缩块，所以.bin里就是这样放入各个压缩块的，在处理数据的过程中，也会去生成对应的一级索引和二级索引

除了这个.bin数据文件之外，其实还有.mrk标记文件，里面记录的是一个数据区间在压缩后的数据块内的偏移量

《110\_MergeTree引擎表的创建和存储结构（2）》

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name (

name1 [type] [DEFAULT|MATERIALIZED|ALIAS expr],

name2 [type] [DEFAULT|MATERIALIZED|ALIAS expr]

) ENGINE = MergeTree()

PARTITION BY expr

ORDER BY expr

PRIMARY KEY expr

SAMPLE BY expr

SETTINGS name=value,name=value

PARTITION BY指定的是分区键，选填，如果指定了，那么就会进行数据分区，如果不指定，就只有一个数据分区，其实一般都是建议根据业务特点对数据进行分区的，因为分区后确实对性能有好处

ORDER BY指定的是排序键，这是必填的，指定在一个数据片段里根据什么来排序，一般情况下把主键和排序键指定为一样的，但是也可以指定特殊的排序键，PRIMARY KEY指定主键，选填，如果没填就跟ORDER BY排序键一样，主键会生成一级索引用于查询

如果有分区键，那么每个分区是一个目录

分区目录里面，默认会包含checksums.txt、columns.txt、count.txt、primary.idx、[Column].bin、[Column].mrk、[Column].mrk2，这些文件，如果是有分区键的，那么就会还有额外的partition.dat、minmax\_[Column].idx，如果建立了二级索引，还会有skp\_idx\_[Column].idx和skp\_idx\_[Column].mrk

checksums.txt，是放校验和的文件，主要保存其他各个文件的size以及哈希值，可以用于校验各个文件是否破损

columns.txt，是放所有的字段信息的

count.txt，是放数据分区下的总行数的

primary.idx，是基于主键建立的一级索引

[Column].bin，是列式存储的数据文件，存放一列的数据，用lz4压缩，.mrk文件记录了.bin文件里各个数据的偏移量，查询的时候通过一级索引先找到.mrk里的便宜量，再根据便宜量可以去.bin里读取数据，.mrk2是自适应大小的索引

partition.dat是放的分区字段值，minmax\_[Column].idx是分区下的数据最小值和最大值

skp两个文件，都是二级索引文件

简单来说呢，你往MergeTree的表里灌入数据的时候，会根据分区键去组织数据，但是其实一个分区是会有多个目录的，就是可能同一个分区会有20200101\_1\_1\_0，20200101\_2\_2\_0，那个1或者2之类的，其实是分区内的最小和最大数据块的编号，最后一个0是合并层级，实际上他默认每隔一段时间，比如10分钟，会对一个分区的多个目录做合并

合并完之后可能会有20200101\_1\_2\_1这样的一个东西，然后旧的目录会保留，默认也会每隔8分钟后台做一个清理的

这个事儿，大家心里有个数就行了

《134\_Clickhouse副本机制底层原理揭秘（2）》

首先，在两个机器上创建互为副本的表，语法都讲过了，表名和分片一致，但是副本名是各自的hostname，第一台机器，先会在zk的/replicas下注册自己的副本节点，其实就是自己的hostname那个节点，然后启动监听任务，监听/log节点下的变动，最后就是往/leader\_election下插入子节点选举主副本，如果是第一个就是主副本

第二台机器，也会干一样的事儿，只不过往/leader\_election下插入子节点，发现自己不是主副本罢了

接着在第一个机器上insert进去数据，此时先会执行之前讲过的逻辑和机制，在机器本地写入数据，包括数据分区，数据压缩块，索引构建，数据标记，等等吧，接着会在zk的/blocks节点下写入数据分区的block\_d

此时有一个重要的参数，就是insert\_quorum，默认是0，也就是一个副本写入成功就ok了，如果你设置为1，那么他必须要等待另外一个副本也写成功才行，如果要让另外一台机器写入副本，核心是在zk的/log节点下写入一个LogEntry，里面记录了另外一台机器需要拉取的数据是哪些

然后第二个机器因为一直在监听zk的/log节点下的变动，所以此时会发现有变动，就会发现有新的数据要拉取，他就会把这个数据拉取任务房到队列里去，异步进行拉取，他会遍历zk的/replicas下的各个副本节点，从里面选一个数据最新的节点

接着就从那个节点去下载自己需要的数据就可以了

不过这里吐槽一下，个人认为这种副本机制基于zk来搞，弄的过于复杂，一点都不好，还有异步执行之类的时效问题

其他的一些操作基本大致都是差不多的，他说白了就是通过这种方式去进行副本数据的同步罢了

《111\_MergeTree的主键索引和二级索引是如何运作的？（1）》

index\_granularity，8192

第8192条数据的主键的值，写入到主键索引里去，第16384条数据，写一条索引，就是16384条数据的主键的值

[0][11546][25568]

其实很简单的，MergeTree的主键索引也就是一级索引，他是稀疏索引，默认是每隔8192行数据会写入一条索引，也就是primary.idx里面去，每次写入的就是主键的那个值，每个值都紧凑的挨在一起，这就是一个稀疏索引的概念

稠密索引，mysql用的就是稠密索引，b树来存放索引，你的每条数据在b树里都有一个索引条目，组织成了一个树，在树里跳转来查找

稀疏索引还是很好的，因为他可以避免索引数据过大，一般上亿数据只要1万条索引就够了，这就是一个好处

clickhouse对主键索引/一级索引就是这样的一个写法，在primary.idx里，每隔8192条数据写一条索引，存储的时候就是紧凑的让索引的值挨在一起，索引很小，10亿条数据对应10万条索引，几万条而已

索引都是直接加载到内存里去使用的，建立一套内存里的索引数据结构用于加速查询

而且这里有一个MarkRange的概念，就是说针对每个数据片段都是一个MarkRange，然后当根据主键 查询的时候，就会根据主键范围区间去匹配MarkRange，最后得到你需要的数据范围是什么

首先，比如说你有一个MarkRange是涵盖全部数据的，他下面是拆分为8个数据片段，你就在8个数据片段里找就行了，定位到一个片段，继续往下拆分为8个数据分片，然后再次定位，最终就能定位到一个数据片段了

CREATE语句里，INDEX index\_name expr TYPE index\_type(…) GRANULARITY granularity

这里有很关键的参数区分，一个是index\_granularity，默认是8192，意思就是按照这个粒度去生成稀疏索引，然后granularity假设是3的话，就是把3个数据分区的内容拿出来做一个汇总，按照之前说的，依次向上生成完整的数据区间

二级索引有4种类型，minmax、set、ngrambf\_v1、tokenbf\_v1

（1）INDEX a ID TYPE minmax GRANULARITY 5

这个其实就跟之前讲的差不多的，说白了就是对每5个数据区间，计算最小和最大的极值，然后扫描的时候就可以快速扫描了，一般都是用这种索引

（2）INDEX b (length(ID) \* 8) TYPE set(100) GRANULARITY 5

这个意思是说，会记录ID长度\*8的值，然后在每个数据区间内，最多记录100条

（3）另外两种索引不常用，都是记录布隆过滤器的

《114\_MergeTree中的列式数据存储以及数据标记（2）》

数据写入列式存储文件的时候，先对数据按照order by进行排序，为什么要排序呢？之前不是讲过索引了么，索引本身就是得按照顺序去生成的，而且还会生成多层级的数据范围，最后都是按照有序数据范围去扫描的

然后会用LZ4进行压缩，压缩过后，就会把压缩数据块写入.bin文件里去

首先在内存里，得按照8192这么个数据粒度对数据进行组织，组织成一个一个的数据范围，接着就看，一个范围的数据<=64KB，就会等待，一直等待到数据凑够64KB了再说，主要累及的数据在64KB到1MB之间，就会生成一个压缩块

如果一个数据区间大于1MB，就对他生成多个压缩块，所以.bin里就是这样放入各个压缩块的，在处理数据的过程中，也会去生成对应的一级索引和二级索引

除了这个.bin数据文件之外，其实还有.mrk标记文件，里面记录的是一个数据区间在压缩后的数据块内的偏移量

《112\_MergeTree的主键索引和二级索引是如何运作的？（2）》

index\_granularity，8192

第8192条数据的主键的值，写入到主键索引里去，第16384条数据，写一条索引，就是16384条数据的主键的值

[0][11546][25568]

其实很简单的，MergeTree的主键索引也就是一级索引，他是稀疏索引，默认是每隔8192行数据会写入一条索引，也就是primary.idx里面去，每次写入的就是主键的那个值，每个值都紧凑的挨在一起，这就是一个稀疏索引的概念

稠密索引，mysql用的就是稠密索引，b树来存放索引，你的每条数据在b树里都有一个索引条目，组织成了一个树，在树里跳转来查找

稀疏索引还是很好的，因为他可以避免索引数据过大，一般上亿数据只要1万条索引就够了，这就是一个好处

clickhouse对主键索引/一级索引就是这样的一个写法，在primary.idx里，每隔8192条数据写一条索引，存储的时候就是紧凑的让索引的值挨在一起，索引很小，10亿条数据对应10万条索引，几万条而已

索引都是直接加载到内存里去使用的，建立一套内存里的索引数据结构用于加速查询

而且这里有一个MarkRange的概念，就是说针对每个数据片段都是一个MarkRange，然后当根据主键 查询的时候，就会根据主键范围区间去匹配MarkRange，最后得到你需要的数据范围是什么

首先，比如说你有一个MarkRange是涵盖全部数据的，他下面是拆分为8个数据片段，你就在8个数据片段里找就行了，定位到一个片段，继续往下拆分为8个数据分片，然后再次定位，最终就能定位到一个数据片段了

CREATE语句里，INDEX index\_name expr TYPE index\_type(…) GRANULARITY granularity

这里有很关键的参数区分，一个是index\_granularity，默认是8192，意思就是按照这个粒度去生成稀疏索引，然后granularity假设是3的话，就是把3个数据分区的内容拿出来做一个汇总，按照之前说的，依次向上生成完整的数据区间

二级索引有4种类型，minmax、set、ngrambf\_v1、tokenbf\_v1

（1）INDEX a ID TYPE minmax GRANULARITY 5

这个其实就跟之前讲的差不多的，说白了就是对每5个数据区间，计算最小和最大的极值，然后扫描的时候就可以快速扫描了，一般都是用这种索引

（2）INDEX b (length(ID) \* 8) TYPE set(100) GRANULARITY 5

这个意思是说，会记录ID长度\*8的值，然后在每个数据区间内，最多记录100条

（3）另外两种索引不常用，都是记录布隆过滤器的

《103\_动手操作一下Clickhouse的安装和部署（5）》

先按照以前的方式去装一台linux虚拟机，接着可以执行下面的安装操作

Clickhouse的rpm安装包下载

https://packagecloud.io/altinity/clickhouse

clickhouse-client-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-common-static-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-common-19.17.4.11-1.el7.x86\_64.rpm

检查linux虚拟机的cpu是否支持SSE 4.2指令集，那是向量化引擎需要的

grep -q sse4\_2 /proc/cpuinfo && echo "SSE 4.2 supported" || echo "SSE 4.2 not supported"

设置FQDN：hostnamectl –static set-hostname clickhouse-01

验证是否生效：hostname -f

配置hosts文件：vi /etc/hosts，本机ip地址 clickhouse-01

上传4个rpm安装包到/usr/local目录下去

rpm -ivh ./\*.rpm，全部安装，可能会出现一些依赖包缺失，此时缺失什么就百度一下，然后先安装依赖就可以了，yum install -y libicu.x86\_64

安装完毕之后，clickhouse会在服务器上构建一套目录结构

/etc/clickhouse-server：配置文件，config.xml全局配置和users.xml用户配置

/var/lib/clickhouse：数据存储目录，可以修改

/var/log/clickhouse-server：日志目录，可以修改

/etc/security/limits.d/clickhouse.conf：文件句柄数量配置，默认是262144，可以修改文件句柄，在config.xml里可以通过max\_open\_files来指定，linux里一切皆文件

/etc/cron.d/clickhouse-server：crontab定时任务配置，主要是用来恢复异常中断的clickhouse进程的，每隔10s会执行一次定时任务，尝试启动clickhouse进程，如果进程在运行就算了，否则就自动启动

/usr/bin目录下有一些可执行命令：clickhouse、clickhouse-client、clickhouse-server，等等，分别是服务端和客户端

修改config.xml，<path>、<tmp\_path>、<user\_files\_path>，都可以修改一下，接着把你修改的目录设置为clickhouse有权限，chown clickhouse.clickhouse /usr/local/xxx -R

接着启动clickhouse：service clickhouse-server start，默认读取的是/etc/clickhouse-server/config.xml

clickhouse-client，进入客户端，show databases;

客户端支持TCP和HTTP，TCP一般是内部通信以及cli客户端，端口号是9000，HTTP是面向系统的，端口号是8123，cli客户端主要用于调试、测试以及运维，进入之后，就可以DDL、DML以及查询了

但是系统访问一般是用JDBC去使用的

<dependency>

<groupId>ru.yandex.clickhouse</groupId>

<artifactId>clickhouse-jdbc</artifactId>

<version>0.2.4</version>

</dependency>

Class.forNmae(“ru.yandex.clickhouse.clickHouseDriver”);

String url = “jdbc:clickhouse://clickhouse.zhss.com:8123/default”;

String user = “default”;

String password = “”;

Connection connection = DriverManager.getConnection(url, username, password);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executQuery(“SELECT xxx”);

resultSet.next();

resultSet.getInt(1);

数据类型：int、float、decimal、string、fexedstring、uuid、DateTime、DateTime64、Date、Array、Tuple、Enum、Nested、Nullable、Domain

CREATE DATABASE [IF NOT EXISTS] db\_name [ENGINE = engine]

引擎有5种：ordinary（默认）、dictionary、memory、lazy、mysql

创建数据库之后，就会在数据目录下有一个文件夹，metadata目录下也有恢复数据库的SQL文件，show databases可以查看到你创建的数据库

USE db\_name，DROP DATABASE [IF EXISTS] db\_name

其实说白了，这个都跟mysql一样的，不同的业务最好是建自己的库，起码逻辑上和物理上都是可以分开的

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name (

name1 [type] [DEFAULT|MATERALIZED|ALIAS expr],

name2 [type] [DEFAULT|MATERALIZED|ALIAS expr]

) ENGINE = engine

CREATE TABLE test\_table\_1 (

title String,

content String,

publish\_time DateTime

) ENGINE = Memory

CREATE TABLE [IF NOT EXISTS] [db\_name1.]table\_name AS [db\_name2.]table\_name2 [ENGINE = engine]，仅仅复制表结构

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name ENGINE = engine AS SELECT，复制表结构，同时数据都会复制一份

DESC table\_name，可以查看表结构

DROP TABLE [IF EXISTS] [db\_name.]table\_name

DEFAULT可以定义默认值

临时表：CREATE TEMPORARY TABLE，别的都一样，仅仅支持memory引擎，会话级别的，会话结束，表就销毁了，说白了，假设你系统需要用这个，就创建一个临时表，数据都在内存，等你程序运行完毕，表就没了，一般我们不怎么用

ALTER TABLE table\_name ADD COLUMN [IF NOT EXISTS] name [type] [default] [AFTER name\_after]，增加字段

ALTER TABLE table\_name MODIFY COLUMN [IF EXISTS] name [type] [default]，修改字段

ALTER TABLE table\_name DROP COLUMN [IF EXISTS] name，删除字段

RENAME TABLE [db\_name1.]table\_name1 TO [db\_name2.]table\_name2, ….，重命名

TRUNCATE TABLE [IF EXISTS] [db\_name.]table\_name，清空表数据

分区表：PARTITION BY toYYYYMM(publish\_time)

然后你插入数据，随着你不同的时间，他会把你数据放入不同的分区里去，不同分区的数据放在各自的目录里，一般在分析型数据库里，分区是很有用的，因为查询的时候指定分区，就可以仅仅扫描分区目录内的数据了

ALTER TABLE table\_name DROP PARTITION partition，删除某个分区

ALTER TABLE table\_name REPLACE PARTITION partition FROM table\_name\_b，复制别的表的分区数据，两个表的分区键相同，表结构相同，其实一般不怎么用这个

ALTER TABLE table\_anme CLEAR COLUMN column\_name IN PARTITION partition，重置分区数据

ALTER TABLE table\_name DETACH/ATTACH PARTITION partition，卸载和装载

视图，简单来说，分为逻辑视图和物理视图，逻辑视图就是一个逻辑上的概念而已，方便你查询用的，物理视图，就是一个物理上独立存储的数据

CREATE VIEW [IF NOT EXISTS] [db\_name.]view\_name AS SELECT

CREATE MATERALIZED VIEW [IF NOT EXISTS] [db\_name.]view\_name [ENGINE = engine] AS SELECT

INSERT INTO [db.]table [(c1,c2,c3…)] VALUES(v1,v2,v3…)，插入数据

数据可以删除和修改，但是在clickhouse这种分析型数据库里，是极为重的操作，他是异步的，而且无法回滚，一般不建议针对单条数据做，最好是批量删除和批量修改

ALTER TABLE [db\_name].table\_name DELETE WHERE filter\_expr，批量删除数据，他会重新弄一个新的目录，里面放入删除后的数据，老的目录还不会立即删除，所以他很重！

ALTER TABLE [db\_name.]table\_name UPDATE col1=val1，修改也是同样的

《102\_动手操作一下Clickhouse的安装和部署（4）》

先按照以前的方式去装一台linux虚拟机，接着可以执行下面的安装操作

Clickhouse的rpm安装包下载

https://packagecloud.io/altinity/clickhouse

clickhouse-client-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-common-static-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-common-19.17.4.11-1.el7.x86\_64.rpm

检查linux虚拟机的cpu是否支持SSE 4.2指令集，那是向量化引擎需要的

grep -q sse4\_2 /proc/cpuinfo && echo "SSE 4.2 supported" || echo "SSE 4.2 not supported"

设置FQDN：hostnamectl –static set-hostname clickhouse-01

验证是否生效：hostname -f

配置hosts文件：vi /etc/hosts，本机ip地址 clickhouse-01

上传4个rpm安装包到/usr/local目录下去

rpm -ivh ./\*.rpm，全部安装，可能会出现一些依赖包缺失，此时缺失什么就百度一下，然后先安装依赖就可以了，yum install -y libicu.x86\_64

安装完毕之后，clickhouse会在服务器上构建一套目录结构

/etc/clickhouse-server：配置文件，config.xml全局配置和users.xml用户配置

/var/lib/clickhouse：数据存储目录，可以修改

/var/log/clickhouse-server：日志目录，可以修改

/etc/security/limits.d/clickhouse.conf：文件句柄数量配置，默认是262144，可以修改文件句柄，在config.xml里可以通过max\_open\_files来指定，linux里一切皆文件

/etc/cron.d/clickhouse-server：crontab定时任务配置，主要是用来恢复异常中断的clickhouse进程的，每隔10s会执行一次定时任务，尝试启动clickhouse进程，如果进程在运行就算了，否则就自动启动

/usr/bin目录下有一些可执行命令：clickhouse、clickhouse-client、clickhouse-server，等等，分别是服务端和客户端

修改config.xml，<path>、<tmp\_path>、<user\_files\_path>，都可以修改一下，接着把你修改的目录设置为clickhouse有权限，chown clickhouse.clickhouse /usr/local/xxx -R

接着启动clickhouse：service clickhouse-server start，默认读取的是/etc/clickhouse-server/config.xml

clickhouse-client，进入客户端，show databases;

客户端支持TCP和HTTP，TCP一般是内部通信以及cli客户端，端口号是9000，HTTP是面向系统的，端口号是8123，cli客户端主要用于调试、测试以及运维，进入之后，就可以DDL、DML以及查询了

但是系统访问一般是用JDBC去使用的

<dependency>

<groupId>ru.yandex.clickhouse</groupId>

<artifactId>clickhouse-jdbc</artifactId>

<version>0.2.4</version>

</dependency>

Class.forNmae(“ru.yandex.clickhouse.clickHouseDriver”);

String url = “jdbc:clickhouse://clickhouse.zhss.com:8123/default”;

String user = “default”;

String password = “”;

Connection connection = DriverManager.getConnection(url, username, password);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executQuery(“SELECT xxx”);

resultSet.next();

resultSet.getInt(1);

数据类型：int、float、decimal、string、fexedstring、uuid、DateTime、DateTime64、Date、Array、Tuple、Enum、Nested、Nullable、Domain

CREATE DATABASE [IF NOT EXISTS] db\_name [ENGINE = engine]

引擎有5种：ordinary（默认）、dictionary、memory、lazy、mysql

创建数据库之后，就会在数据目录下有一个文件夹，metadata目录下也有恢复数据库的SQL文件，show databases可以查看到你创建的数据库

USE db\_name，DROP DATABASE [IF EXISTS] db\_name

其实说白了，这个都跟mysql一样的，不同的业务最好是建自己的库，起码逻辑上和物理上都是可以分开的

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name (

name1 [type] [DEFAULT|MATERALIZED|ALIAS expr],

name2 [type] [DEFAULT|MATERALIZED|ALIAS expr]

) ENGINE = engine

CREATE TABLE test\_table\_1 (

title String,

content String,

publish\_time DateTime

) ENGINE = Memory

CREATE TABLE [IF NOT EXISTS] [db\_name1.]table\_name AS [db\_name2.]table\_name2 [ENGINE = engine]，仅仅复制表结构

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name ENGINE = engine AS SELECT，复制表结构，同时数据都会复制一份

DESC table\_name，可以查看表结构

DROP TABLE [IF EXISTS] [db\_name.]table\_name

DEFAULT可以定义默认值

临时表：CREATE TEMPORARY TABLE，别的都一样，仅仅支持memory引擎，会话级别的，会话结束，表就销毁了，说白了，假设你系统需要用这个，就创建一个临时表，数据都在内存，等你程序运行完毕，表就没了，一般我们不怎么用

ALTER TABLE table\_name ADD COLUMN [IF NOT EXISTS] name [type] [default] [AFTER name\_after]，增加字段

ALTER TABLE table\_name MODIFY COLUMN [IF EXISTS] name [type] [default]，修改字段

ALTER TABLE table\_name DROP COLUMN [IF EXISTS] name，删除字段

RENAME TABLE [db\_name1.]table\_name1 TO [db\_name2.]table\_name2, ….，重命名

TRUNCATE TABLE [IF EXISTS] [db\_name.]table\_name，清空表数据

分区表：PARTITION BY toYYYYMM(publish\_time)

然后你插入数据，随着你不同的时间，他会把你数据放入不同的分区里去，不同分区的数据放在各自的目录里，一般在分析型数据库里，分区是很有用的，因为查询的时候指定分区，就可以仅仅扫描分区目录内的数据了

ALTER TABLE table\_name DROP PARTITION partition，删除某个分区

ALTER TABLE table\_name REPLACE PARTITION partition FROM table\_name\_b，复制别的表的分区数据，两个表的分区键相同，表结构相同，其实一般不怎么用这个

ALTER TABLE table\_anme CLEAR COLUMN column\_name IN PARTITION partition，重置分区数据

ALTER TABLE table\_name DETACH/ATTACH PARTITION partition，卸载和装载

视图，简单来说，分为逻辑视图和物理视图，逻辑视图就是一个逻辑上的概念而已，方便你查询用的，物理视图，就是一个物理上独立存储的数据

CREATE VIEW [IF NOT EXISTS] [db\_name.]view\_name AS SELECT

CREATE MATERALIZED VIEW [IF NOT EXISTS] [db\_name.]view\_name [ENGINE = engine] AS SELECT

INSERT INTO [db.]table [(c1,c2,c3…)] VALUES(v1,v2,v3…)，插入数据

数据可以删除和修改，但是在clickhouse这种分析型数据库里，是极为重的操作，他是异步的，而且无法回滚，一般不建议针对单条数据做，最好是批量删除和批量修改

ALTER TABLE [db\_name].table\_name DELETE WHERE filter\_expr，批量删除数据，他会重新弄一个新的目录，里面放入删除后的数据，老的目录还不会立即删除，所以他很重！

ALTER TABLE [db\_name.]table\_name UPDATE col1=val1，修改也是同样的

《127\_授人以渔：教你自己去拓展学习Clickhouse（3）》

其他的一些表引擎，查询语法，运维管理，分布式架构

你如果要用好clickhouse，了解清楚他的各种表引擎的原理和特性，MergeTree家族引擎，原理大家都已经知道了，对表引擎的工作原理熟悉的基础之上去用他，你大概知道用这个表引擎建了表之后，查询的时候是如何工作的

如何去建立索引，如何分区，如何做优化，如何让他的性能是最高的

《108\_熟悉Clickhouse数据库的DDL语法（5）》

先按照以前的方式去装一台linux虚拟机，接着可以执行下面的安装操作

Clickhouse的rpm安装包下载

https://packagecloud.io/altinity/clickhouse

clickhouse-client-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-common-static-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-common-19.17.4.11-1.el7.x86\_64.rpm

检查linux虚拟机的cpu是否支持SSE 4.2指令集，那是向量化引擎需要的

grep -q sse4\_2 /proc/cpuinfo && echo "SSE 4.2 supported" || echo "SSE 4.2 not supported"

设置FQDN：hostnamectl –static set-hostname clickhouse-01

验证是否生效：hostname -f

配置hosts文件：vi /etc/hosts，本机ip地址 clickhouse-01

上传4个rpm安装包到/usr/local目录下去

rpm -ivh ./\*.rpm，全部安装，可能会出现一些依赖包缺失，此时缺失什么就百度一下，然后先安装依赖就可以了，yum install -y libicu.x86\_64

安装完毕之后，clickhouse会在服务器上构建一套目录结构

/etc/clickhouse-server：配置文件，config.xml全局配置和users.xml用户配置

/var/lib/clickhouse：数据存储目录，可以修改

/var/log/clickhouse-server：日志目录，可以修改

/etc/security/limits.d/clickhouse.conf：文件句柄数量配置，默认是262144，可以修改文件句柄，在config.xml里可以通过max\_open\_files来指定，linux里一切皆文件

/etc/cron.d/clickhouse-server：crontab定时任务配置，主要是用来恢复异常中断的clickhouse进程的，每隔10s会执行一次定时任务，尝试启动clickhouse进程，如果进程在运行就算了，否则就自动启动

/usr/bin目录下有一些可执行命令：clickhouse、clickhouse-client、clickhouse-server，等等，分别是服务端和客户端

修改config.xml，<path>、<tmp\_path>、<user\_files\_path>，都可以修改一下，接着把你修改的目录设置为clickhouse有权限，chown clickhouse.clickhouse /usr/local/xxx -R

接着启动clickhouse：service clickhouse-server start，默认读取的是/etc/clickhouse-server/config.xml

clickhouse-client，进入客户端，show databases;

客户端支持TCP和HTTP，TCP一般是内部通信以及cli客户端，端口号是9000，HTTP是面向系统的，端口号是8123，cli客户端主要用于调试、测试以及运维，进入之后，就可以DDL、DML以及查询了

但是系统访问一般是用JDBC去使用的

<dependency>

<groupId>ru.yandex.clickhouse</groupId>

<artifactId>clickhouse-jdbc</artifactId>

<version>0.2.4</version>

</dependency>

Class.forNmae(“ru.yandex.clickhouse.clickHouseDriver”);

String url = “jdbc:clickhouse://clickhouse.zhss.com:8123/default”;

String user = “default”;

String password = “”;

Connection connection = DriverManager.getConnection(url, username, password);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executQuery(“SELECT xxx”);

resultSet.next();

resultSet.getInt(1);

CREATE DATABASE [IF NOT EXISTS] db\_name [ENGINE = engine]

引擎有5种：ordinary（默认）、dictionary、memory、lazy、mysql

创建数据库之后，就会在数据目录下有一个文件夹，metadata目录下也有恢复数据库的SQL文件，show databases可以查看到你创建的数据库

USE db\_name，DROP DATABASE [IF EXISTS] db\_name

其实说白了，这个都跟mysql一样的，不同的业务最好是建自己的库，起码逻辑上和物理上都是可以分开的

数据类型：Int、Float、Decimal、String、FixedString、Uuid、DateTime、DateTime64、Date、Array、Tuple、Enum、Nested、Nullable、Domain

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name (

name1 [type] [DEFAULT|MATERALIZED|ALIAS expr],

name2 [type] [DEFAULT|MATERALIZED|ALIAS expr]

) ENGINE = engine

CREATE TABLE test\_table\_1 (

title String,

content String,

publish\_time DateTime

)

CREATE TABLE [IF NOT EXISTS] [db\_name1.]table\_name AS [db\_name2.]table\_name2 [ENGINE = engine]，仅仅复制表结构

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name ENGINE = engine AS SELECT，复制表结构，同时数据都会复制一份

DESC table\_name，可以查看表结构

DROP TABLE [IF EXISTS] [db\_name.]table\_name

DEFAULT可以定义默认值

*临时表：CREATE TEMPORARY TABLE，别的都一样，仅仅支持memory引擎，会话级别的，会话结束，表就销毁了，说白了，假设你系统需要用这个，就创建一个临时表，数据都在内存，等你程序运行完毕，表就没了，一般我们不怎么用*

ALTER TABLE table\_name ADD COLUMN [IF NOT EXISTS] name [type] [default] [AFTER name\_after]，增加字段

ALTER TABLE table\_name MODIFY COLUMN [IF EXISTS] name [type] [default]，修改字段

ALTER TABLE table\_name DROP COLUMN [IF EXISTS] name，删除字段

RENAME TABLE [db\_name1.]table\_name1 TO [db\_name2.]table\_name2, ….，重命名

TRUNCATE TABLE [IF EXISTS] [db\_name.]table\_name，清空表数据

MergeTree，重点剖析他内部的原理，有很多种存储引擎，存储引擎内部的原理之后，我们再来学习他不同的表如何指定存储引擎，在内部他会如何运行

分区表：PARTITION BY toYYYYMM(publish\_time)

create table test\_table (id Int, title String, content String, publish\_time Date) engine=MergeTree() partition by toYYYYMM(publish\_time) order by id

然后你插入数据，随着你不同的时间，他会把你数据放入不同的分区里去，不同分区的数据放在各自的目录里，一般在分析型数据库里，分区是很有用的，因为查询的时候指定分区，就可以仅仅扫描分区目录内的数据了

ALTER TABLE table\_name DROP PARTITION partition，删除某个分区

ALTER TABLE table\_name REPLACE PARTITION partition FROM table\_name\_b，复制别的表的分区数据，两个表的分区键相同，表结构相同，其实一般不怎么用这个

ALTER TABLE table\_anme CLEAR COLUMN column\_name IN PARTITION partition，重置分区数据

ALTER TABLE table\_name DETACH/ATTACH PARTITION partition，卸载和装载

视图，简单来说，分为逻辑视图和物理视图，逻辑视图就是一个逻辑上的概念而已，方便你查询用的，物理视图，就是一个物理上独立存储的数据

CREATE VIEW [IF NOT EXISTS] [db\_name.]view\_name AS SELECT

CREATE MATERALIZED VIEW [IF NOT EXISTS] [db\_name.]view\_name [ENGINE = engine] AS SELECT

INSERT INTO [db.]table [(c1,c2,c3…)] VALUES(v1,v2,v3…)，插入数据

insert into test\_table(id, title, content, publish\_time) values(

数据可以删除和修改，但是在clickhouse这种分析型数据库里，是极为重的操作，他是异步的，而且无法回滚，一般不建议针对单条数据做，最好是批量删除和批量修改

ALTER TABLE [db\_name].table\_name DELETE WHERE filter\_expr，批量删除数据，他会重新弄一个新的目录，里面放入删除后的数据，老的目录还不会立即删除，所以他很重！

ALTER TABLE [db\_name.]table\_name UPDATE col1=val1，修改也是同样的

《115\_MergeTree的数据写入过程分析（1）》

写入数据，先确定分区，当然一个分区可能有多个目录了，分区的多个目录可能会合并，写入的时候先在内存里，按照8192这个粒度划分为多个数据区间，每个数据区间都是8192条数据，区间都有start和end的数据offset，比如0~8192，8192~16384

然后每获取到一个数据区间，就会在primary.idx一级索引里写入一条稀疏索引数据，比如说第一个区间就是索引条目0，索引值就是第0行数据，第二个区间对应的是索引条目1，索引值是第8192行数据的值，第三个区间对应的是索引条目2，索引值是第16384行数据的值，以此类推

而且每次凑够一个数据区间，除了在primary.idx里写入一条稀疏索引，还得在各个字段的.mrk标记文件里写入一条标记数据，这个一个数据区间对应一条稀疏索引，也对应各个字段的.mrk标记文件里的一条标记数据，稀疏索引和标记数据是一一对应的

.mrk标记文件里，实际上写入的是当前这个区间的数据在这个字段的.bin数据压缩文件里的压缩块的偏移量

比如说，第一个数据区间的所有字段A的数据，没有凑够64KB，那么他铁定是属于.bin数据压缩文件里的压缩快0了，那么这个数据区间对应的字段A的.mrk文件的标记条目0记录的值，其实就是字段A的.bin文件里的压缩块0的偏移量了

接着第二个数据区间会在primary.idx里写一条稀疏索引条目，索引值就是第二个区间起始数据行（8192行）的主键值，同时会在各个字段的.mrk标记文件里，去写入一个标记条目1，那么标记条目1的值是什么呢？

现在假设两个数据区间内的字段A的数据量已经超过了64kb，小于1mb，此时就会把这波数据写为一个压缩块，压缩块放入字段A的.bin文件里去，所以第二个数据区间对应的字段A的.mrk文件里的标记条目1的值，也是字段A的.bin文件里的压缩块0的偏移量

后续的数据区间都是以此类推的

首先要在内存里凑够一个8192条数据的数据区间，然后就是写primary.idx里的一级索引条目的值，做一个稀疏索引，接着各个字段都得写.mrk标记文件，此时这个数据区间一定在各个字段的.mrk标记文件里会有一个标记条目

标记条目的值，得判断一下，就是这个数据区间的各个字段的值大小，是在各个字段的.bin文件里的第几个压缩块里，把那个压缩块的偏移量作为那些字段的标记条目的值

《105\_熟悉Clickhouse数据库的DDL语法（2）》

先按照以前的方式去装一台linux虚拟机，接着可以执行下面的安装操作

Clickhouse的rpm安装包下载

https://packagecloud.io/altinity/clickhouse

clickhouse-client-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-common-static-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-common-19.17.4.11-1.el7.x86\_64.rpm

检查linux虚拟机的cpu是否支持SSE 4.2指令集，那是向量化引擎需要的

grep -q sse4\_2 /proc/cpuinfo && echo "SSE 4.2 supported" || echo "SSE 4.2 not supported"

设置FQDN：hostnamectl –static set-hostname clickhouse-01

验证是否生效：hostname -f

配置hosts文件：vi /etc/hosts，本机ip地址 clickhouse-01

上传4个rpm安装包到/usr/local目录下去

rpm -ivh ./\*.rpm，全部安装，可能会出现一些依赖包缺失，此时缺失什么就百度一下，然后先安装依赖就可以了，yum install -y libicu.x86\_64

安装完毕之后，clickhouse会在服务器上构建一套目录结构

/etc/clickhouse-server：配置文件，config.xml全局配置和users.xml用户配置

/var/lib/clickhouse：数据存储目录，可以修改

/var/log/clickhouse-server：日志目录，可以修改

/etc/security/limits.d/clickhouse.conf：文件句柄数量配置，默认是262144，可以修改文件句柄，在config.xml里可以通过max\_open\_files来指定，linux里一切皆文件

/etc/cron.d/clickhouse-server：crontab定时任务配置，主要是用来恢复异常中断的clickhouse进程的，每隔10s会执行一次定时任务，尝试启动clickhouse进程，如果进程在运行就算了，否则就自动启动

/usr/bin目录下有一些可执行命令：clickhouse、clickhouse-client、clickhouse-server，等等，分别是服务端和客户端

修改config.xml，<path>、<tmp\_path>、<user\_files\_path>，都可以修改一下，接着把你修改的目录设置为clickhouse有权限，chown clickhouse.clickhouse /usr/local/xxx -R

接着启动clickhouse：service clickhouse-server start，默认读取的是/etc/clickhouse-server/config.xml

clickhouse-client，进入客户端，show databases;

客户端支持TCP和HTTP，TCP一般是内部通信以及cli客户端，端口号是9000，HTTP是面向系统的，端口号是8123，cli客户端主要用于调试、测试以及运维，进入之后，就可以DDL、DML以及查询了

但是系统访问一般是用JDBC去使用的

<dependency>

<groupId>ru.yandex.clickhouse</groupId>

<artifactId>clickhouse-jdbc</artifactId>

<version>0.2.4</version>

</dependency>

Class.forNmae(“ru.yandex.clickhouse.clickHouseDriver”);

String url = “jdbc:clickhouse://clickhouse.zhss.com:8123/default”;

String user = “default”;

String password = “”;

Connection connection = DriverManager.getConnection(url, username, password);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executQuery(“SELECT xxx”);

resultSet.next();

resultSet.getInt(1);

CREATE DATABASE [IF NOT EXISTS] db\_name [ENGINE = engine]

引擎有5种：ordinary（默认）、dictionary、memory、lazy、mysql

创建数据库之后，就会在数据目录下有一个文件夹，metadata目录下也有恢复数据库的SQL文件，show databases可以查看到你创建的数据库

USE db\_name，DROP DATABASE [IF EXISTS] db\_name

其实说白了，这个都跟mysql一样的，不同的业务最好是建自己的库，起码逻辑上和物理上都是可以分开的

数据类型：Int、Float、Decimal、String、FixedString、Uuid、DateTime、DateTime64、Date、Array、Tuple、Enum、Nested、Nullable、Domain

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name (

name1 [type] [DEFAULT|MATERALIZED|ALIAS expr],

name2 [type] [DEFAULT|MATERALIZED|ALIAS expr]

) ENGINE = engine

CREATE TABLE test\_table\_1 (

title String,

content String,

publish\_time DateTime

)

CREATE TABLE [IF NOT EXISTS] [db\_name1.]table\_name AS [db\_name2.]table\_name2 [ENGINE = engine]，仅仅复制表结构

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name ENGINE = engine AS SELECT，复制表结构，同时数据都会复制一份

DESC table\_name，可以查看表结构

DROP TABLE [IF EXISTS] [db\_name.]table\_name

DEFAULT可以定义默认值

临时表：CREATE TEMPORARY TABLE，别的都一样，仅仅支持memory引擎，会话级别的，会话结束，表就销毁了，说白了，假设你系统需要用这个，就创建一个临时表，数据都在内存，等你程序运行完毕，表就没了，一般我们不怎么用

ALTER TABLE table\_name ADD COLUMN [IF NOT EXISTS] name [type] [default] [AFTER name\_after]，增加字段

ALTER TABLE table\_name MODIFY COLUMN [IF EXISTS] name [type] [default]，修改字段

ALTER TABLE table\_name DROP COLUMN [IF EXISTS] name，删除字段

RENAME TABLE [db\_name1.]table\_name1 TO [db\_name2.]table\_name2, ….，重命名

TRUNCATE TABLE [IF EXISTS] [db\_name.]table\_name，清空表数据

分区表：PARTITION BY toYYYYMM(publish\_time)

然后你插入数据，随着你不同的时间，他会把你数据放入不同的分区里去，不同分区的数据放在各自的目录里，一般在分析型数据库里，分区是很有用的，因为查询的时候指定分区，就可以仅仅扫描分区目录内的数据了

ALTER TABLE table\_name DROP PARTITION partition，删除某个分区

ALTER TABLE table\_name REPLACE PARTITION partition FROM table\_name\_b，复制别的表的分区数据，两个表的分区键相同，表结构相同，其实一般不怎么用这个

ALTER TABLE table\_anme CLEAR COLUMN column\_name IN PARTITION partition，重置分区数据

ALTER TABLE table\_name DETACH/ATTACH PARTITION partition，卸载和装载

视图，简单来说，分为逻辑视图和物理视图，逻辑视图就是一个逻辑上的概念而已，方便你查询用的，物理视图，就是一个物理上独立存储的数据

CREATE VIEW [IF NOT EXISTS] [db\_name.]view\_name AS SELECT

CREATE MATERALIZED VIEW [IF NOT EXISTS] [db\_name.]view\_name [ENGINE = engine] AS SELECT

INSERT INTO [db.]table [(c1,c2,c3…)] VALUES(v1,v2,v3…)，插入数据

数据可以删除和修改，但是在clickhouse这种分析型数据库里，是极为重的操作，他是异步的，而且无法回滚，一般不建议针对单条数据做，最好是批量删除和批量修改

ALTER TABLE [db\_name].table\_name DELETE WHERE filter\_expr，批量删除数据，他会重新弄一个新的目录，里面放入删除后的数据，老的目录还不会立即删除，所以他很重！

ALTER TABLE [db\_name.]table\_name UPDATE col1=val1，修改也是同样的

《104\_熟悉Clickhouse数据库的DDL语法（1）》

先按照以前的方式去装一台linux虚拟机，接着可以执行下面的安装操作

Clickhouse的rpm安装包下载

https://packagecloud.io/altinity/clickhouse

clickhouse-client-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-common-static-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-common-19.17.4.11-1.el7.x86\_64.rpm

检查linux虚拟机的cpu是否支持SSE 4.2指令集，那是向量化引擎需要的

grep -q sse4\_2 /proc/cpuinfo && echo "SSE 4.2 supported" || echo "SSE 4.2 not supported"

设置FQDN：hostnamectl –static set-hostname clickhouse-01

验证是否生效：hostname -f

配置hosts文件：vi /etc/hosts，本机ip地址 clickhouse-01

上传4个rpm安装包到/usr/local目录下去

rpm -ivh ./\*.rpm，全部安装，可能会出现一些依赖包缺失，此时缺失什么就百度一下，然后先安装依赖就可以了，yum install -y libicu.x86\_64

安装完毕之后，clickhouse会在服务器上构建一套目录结构

/etc/clickhouse-server：配置文件，config.xml全局配置和users.xml用户配置

/var/lib/clickhouse：数据存储目录，可以修改

/var/log/clickhouse-server：日志目录，可以修改

/etc/security/limits.d/clickhouse.conf：文件句柄数量配置，默认是262144，可以修改文件句柄，在config.xml里可以通过max\_open\_files来指定，linux里一切皆文件

/etc/cron.d/clickhouse-server：crontab定时任务配置，主要是用来恢复异常中断的clickhouse进程的，每隔10s会执行一次定时任务，尝试启动clickhouse进程，如果进程在运行就算了，否则就自动启动

/usr/bin目录下有一些可执行命令：clickhouse、clickhouse-client、clickhouse-server，等等，分别是服务端和客户端

修改config.xml，<path>、<tmp\_path>、<user\_files\_path>，都可以修改一下，接着把你修改的目录设置为clickhouse有权限，chown clickhouse.clickhouse /usr/local/xxx -R

接着启动clickhouse：service clickhouse-server start，默认读取的是/etc/clickhouse-server/config.xml

clickhouse-client，进入客户端，show databases;

客户端支持TCP和HTTP，TCP一般是内部通信以及cli客户端，端口号是9000，HTTP是面向系统的，端口号是8123，cli客户端主要用于调试、测试以及运维，进入之后，就可以DDL、DML以及查询了

但是系统访问一般是用JDBC去使用的

<dependency>

<groupId>ru.yandex.clickhouse</groupId>

<artifactId>clickhouse-jdbc</artifactId>

<version>0.2.4</version>

</dependency>

Class.forNmae(“ru.yandex.clickhouse.clickHouseDriver”);

String url = “jdbc:clickhouse://clickhouse.zhss.com:8123/default”;

String user = “default”;

String password = “”;

Connection connection = DriverManager.getConnection(url, username, password);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executQuery(“SELECT xxx”);

resultSet.next();

resultSet.getInt(1);

CREATE DATABASE [IF NOT EXISTS] db\_name [ENGINE = engine]

引擎有5种：ordinary（默认）、dictionary、memory、lazy、mysql

创建数据库之后，就会在数据目录下有一个文件夹，metadata目录下也有恢复数据库的SQL文件，show databases可以查看到你创建的数据库

USE db\_name，DROP DATABASE [IF EXISTS] db\_name

其实说白了，这个都跟mysql一样的，不同的业务最好是建自己的库，起码逻辑上和物理上都是可以分开的

数据类型：Int、Float、Decimal、String、FixedString、Uuid、DateTime、DateTime64、Date、Array、Tuple、Enum、Nested、Nullable、Domain

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name (

name1 [type] [DEFAULT|MATERALIZED|ALIAS expr],

name2 [type] [DEFAULT|MATERALIZED|ALIAS expr]

) ENGINE = engine

CREATE TABLE test\_table\_1 (

title String,

content String,

publish\_time DateTime

) ENGINE = Memory

CREATE TABLE [IF NOT EXISTS] [db\_name1.]table\_name AS [db\_name2.]table\_name2 [ENGINE = engine]，仅仅复制表结构

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name ENGINE = engine AS SELECT，复制表结构，同时数据都会复制一份

DESC table\_name，可以查看表结构

DROP TABLE [IF EXISTS] [db\_name.]table\_name

DEFAULT可以定义默认值

临时表：CREATE TEMPORARY TABLE，别的都一样，仅仅支持memory引擎，会话级别的，会话结束，表就销毁了，说白了，假设你系统需要用这个，就创建一个临时表，数据都在内存，等你程序运行完毕，表就没了，一般我们不怎么用

ALTER TABLE table\_name ADD COLUMN [IF NOT EXISTS] name [type] [default] [AFTER name\_after]，增加字段

ALTER TABLE table\_name MODIFY COLUMN [IF EXISTS] name [type] [default]，修改字段

ALTER TABLE table\_name DROP COLUMN [IF EXISTS] name，删除字段

RENAME TABLE [db\_name1.]table\_name1 TO [db\_name2.]table\_name2, ….，重命名

TRUNCATE TABLE [IF EXISTS] [db\_name.]table\_name，清空表数据

分区表：PARTITION BY toYYYYMM(publish\_time)

然后你插入数据，随着你不同的时间，他会把你数据放入不同的分区里去，不同分区的数据放在各自的目录里，一般在分析型数据库里，分区是很有用的，因为查询的时候指定分区，就可以仅仅扫描分区目录内的数据了

ALTER TABLE table\_name DROP PARTITION partition，删除某个分区

ALTER TABLE table\_name REPLACE PARTITION partition FROM table\_name\_b，复制别的表的分区数据，两个表的分区键相同，表结构相同，其实一般不怎么用这个

ALTER TABLE table\_anme CLEAR COLUMN column\_name IN PARTITION partition，重置分区数据

ALTER TABLE table\_name DETACH/ATTACH PARTITION partition，卸载和装载

视图，简单来说，分为逻辑视图和物理视图，逻辑视图就是一个逻辑上的概念而已，方便你查询用的，物理视图，就是一个物理上独立存储的数据

CREATE VIEW [IF NOT EXISTS] [db\_name.]view\_name AS SELECT

CREATE MATERALIZED VIEW [IF NOT EXISTS] [db\_name.]view\_name [ENGINE = engine] AS SELECT

INSERT INTO [db.]table [(c1,c2,c3…)] VALUES(v1,v2,v3…)，插入数据

数据可以删除和修改，但是在clickhouse这种分析型数据库里，是极为重的操作，他是异步的，而且无法回滚，一般不建议针对单条数据做，最好是批量删除和批量修改

ALTER TABLE [db\_name].table\_name DELETE WHERE filter\_expr，批量删除数据，他会重新弄一个新的目录，里面放入删除后的数据，老的目录还不会立即删除，所以他很重！

ALTER TABLE [db\_name.]table\_name UPDATE col1=val1，修改也是同样的

《130\_Clickhouse的集群架构与副本机制介绍（2）》

之前讲的都是单节点的，不过单节点的理解了，其实多节点的集群、副本、分片，也都是差不多的，因为基本都这么回事儿

然后首先是可以部署多个clickhous节点组成集群，这个是没问题的

其次就是先讲副本的概念，副本的话，主要是通过ReplicatedMergeTree引擎实现的，说着说，之前讲的各种引擎加上Replicated关键字都可以转化为支持副本的表引擎，就这么个意思

这个副本机制是通过zk来实现的，不过个人认为这是一个败笔，因为我认为一个优秀的分布式系统，不应该随意依赖外部，除非你自己就是觉得麻烦，一定要依赖，否则一般我们都是自行开发分布式系统的协调机制的

ReplicatedMergeTree表引擎会通过zk去协同，把数据同步到其他节点上去，形成一个副本，在表级别可以对副本进行配置，比如副本的数量以及在集群里的分布位置，而且Clickhouse支持副本的多主架构，任何一个副本都可以执行增删改，各个副本会互相同步

要玩儿clickhouse集群架构和副本设置，就得先启动多个clickhouse节点，每个节点都配置一下zk，在/etc/clickhouse-server/config.d目录下，创建一个metrika.xml：

<?xml version=”1.0”?>

<yandex>

<zookeeper-servers>

<node index=”1”>

<host>xx.xx.xx </host>

<port>2181</port>

</node>

</zookeeper-servers>

</yandex>

然后在config.xml里加入一段配置：

<include\_form>/etc/clickhouse-server/config.d/metrika.xml</inclucde\_form>

<zookeeper incl=”zookeeper-servers” optional=”false” />

加入这个配置之后，启动clickhouse集群就可以了，集群之间会通过同一个zk进行协调，而且clickhouse提供了zk内置表，select \* from system.zookeeper where path=’/’，就可以查到zk里的指定目录下的数据

select name,value,czxid,mzxid from system.zookeeper where path=’/clickhouse’

然后就可以创建支持副本的表了

ENGINE = ReplicatedMergeTree(‘zk\_path’, ‘replica\_name’)

zk\_path指的是这个表在zk里的路径，一般是/clickhouse/tables/{shard}/table\_name，这里shard是你的分片编号，比如01，02之类的，然后就是表名了，刚开始你可以建一个分片的表

后面是有一个replica\_name就是副本的名称

举个例子，那你在clickhouse一个节点上可以创建一个表，引擎的话，那就是zk\_path就是定义的是这个表的哪个分片了，可以就一个分片，那么分片编号都相同，那不同机器上可以定义不同的副本名称了

副本名称一般用的是机器的hostname

《131\_Clickhouse的集群架构与副本机制介绍（3）》

之前讲的都是单节点的，不过单节点的理解了，其实多节点的集群、副本、分片，也都是差不多的，因为基本都这么回事儿

然后首先是可以部署多个clickhous节点组成集群，这个是没问题的

其次就是先讲副本的概念，副本的话，主要是通过ReplicatedMergeTree引擎实现的，说着说，之前讲的各种引擎加上Replicated关键字都可以转化为支持副本的表引擎，就这么个意思

这个副本机制是通过zk来实现的，不过个人认为这是一个败笔，因为我认为一个优秀的分布式系统，不应该随意依赖外部，除非你自己就是觉得麻烦，一定要依赖，否则一般我们都是自行开发分布式系统的协调机制的

ReplicatedMergeTree表引擎会通过zk去协同，把数据同步到其他节点上去，形成一个副本，在表级别可以对副本进行配置，比如副本的数量以及在集群里的分布位置，而且Clickhouse支持副本的多主架构，任何一个副本都可以执行增删改，各个副本会互相同步

要玩儿clickhouse集群架构和副本设置，就得先启动多个clickhouse节点，每个节点都配置一下zk，在/etc/clickhouse-server/config.d目录下，创建一个metrika.xml：

<?xml version=”1.0”?>

<yandex>

<zookeeper-servers>

<node index=”1”>

<host>xx.xx.xx </host>

<port>2181</port>

</node>

</zookeeper-servers>

</yandex>

然后在config.xml里加入一段配置：

<include\_form>/etc/clickhouse-server/config.d/metrika.xml</inclucde\_form>

<zookeeper incl=”zookeeper-servers” optional=”false” />

加入这个配置之后，启动clickhouse集群就可以了，集群之间会通过同一个zk进行协调，而且clickhouse提供了zk内置表，select \* from system.zookeeper where path=’/’，就可以查到zk里的指定目录下的数据

select name,value,czxid,mzxid from system.zookeeper where path=’/clickhouse’

然后就可以创建支持副本的表了

ENGINE = ReplicatedMergeTree(‘zk\_path’, ‘replica\_name’)

zk\_path指的是这个表在zk里的路径，一般是/clickhouse/tables/{shard}/table\_name，这里shard是你的分片编号，比如01，02之类的，然后就是表名了，刚开始你可以建一个分片的表

后面是有一个replica\_name就是副本的名称

举个例子，那你在clickhouse一个节点上可以创建一个表，引擎的话，那就是zk\_path就是定义的是这个表的哪个分片了，可以就一个分片，那么分片编号都相同，那不同机器上可以定义不同的副本名称了

副本名称一般用的是机器的hostname

ENGINE = ReplicatedMergeTree(‘’, ‘replica\_name’)

《118\_Clickhouse的数据写入以及查询原理回顾》

《125\_授人以渔：教你自己去拓展学习Clickhouse（1）》

其他的一些表引擎，查询语法，运维管理，分布式架构

《107\_熟悉Clickhouse数据库的DDL语法（4）》

先按照以前的方式去装一台linux虚拟机，接着可以执行下面的安装操作

Clickhouse的rpm安装包下载

https://packagecloud.io/altinity/clickhouse

clickhouse-client-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-common-static-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-common-19.17.4.11-1.el7.x86\_64.rpm

检查linux虚拟机的cpu是否支持SSE 4.2指令集，那是向量化引擎需要的

grep -q sse4\_2 /proc/cpuinfo && echo "SSE 4.2 supported" || echo "SSE 4.2 not supported"

设置FQDN：hostnamectl –static set-hostname clickhouse-01

验证是否生效：hostname -f

配置hosts文件：vi /etc/hosts，本机ip地址 clickhouse-01

上传4个rpm安装包到/usr/local目录下去

rpm -ivh ./\*.rpm，全部安装，可能会出现一些依赖包缺失，此时缺失什么就百度一下，然后先安装依赖就可以了，yum install -y libicu.x86\_64

安装完毕之后，clickhouse会在服务器上构建一套目录结构

/etc/clickhouse-server：配置文件，config.xml全局配置和users.xml用户配置

/var/lib/clickhouse：数据存储目录，可以修改

/var/log/clickhouse-server：日志目录，可以修改

/etc/security/limits.d/clickhouse.conf：文件句柄数量配置，默认是262144，可以修改文件句柄，在config.xml里可以通过max\_open\_files来指定，linux里一切皆文件

/etc/cron.d/clickhouse-server：crontab定时任务配置，主要是用来恢复异常中断的clickhouse进程的，每隔10s会执行一次定时任务，尝试启动clickhouse进程，如果进程在运行就算了，否则就自动启动

/usr/bin目录下有一些可执行命令：clickhouse、clickhouse-client、clickhouse-server，等等，分别是服务端和客户端

修改config.xml，<path>、<tmp\_path>、<user\_files\_path>，都可以修改一下，接着把你修改的目录设置为clickhouse有权限，chown clickhouse.clickhouse /usr/local/xxx -R

接着启动clickhouse：service clickhouse-server start，默认读取的是/etc/clickhouse-server/config.xml

clickhouse-client，进入客户端，show databases;

客户端支持TCP和HTTP，TCP一般是内部通信以及cli客户端，端口号是9000，HTTP是面向系统的，端口号是8123，cli客户端主要用于调试、测试以及运维，进入之后，就可以DDL、DML以及查询了

但是系统访问一般是用JDBC去使用的

<dependency>

<groupId>ru.yandex.clickhouse</groupId>

<artifactId>clickhouse-jdbc</artifactId>

<version>0.2.4</version>

</dependency>

Class.forNmae(“ru.yandex.clickhouse.clickHouseDriver”);

String url = “jdbc:clickhouse://clickhouse.zhss.com:8123/default”;

String user = “default”;

String password = “”;

Connection connection = DriverManager.getConnection(url, username, password);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executQuery(“SELECT xxx”);

resultSet.next();

resultSet.getInt(1);

CREATE DATABASE [IF NOT EXISTS] db\_name [ENGINE = engine]

引擎有5种：ordinary（默认）、dictionary、memory、lazy、mysql

创建数据库之后，就会在数据目录下有一个文件夹，metadata目录下也有恢复数据库的SQL文件，show databases可以查看到你创建的数据库

USE db\_name，DROP DATABASE [IF EXISTS] db\_name

其实说白了，这个都跟mysql一样的，不同的业务最好是建自己的库，起码逻辑上和物理上都是可以分开的

数据类型：Int、Float、Decimal、String、FixedString、Uuid、DateTime、DateTime64、Date、Array、Tuple、Enum、Nested、Nullable、Domain

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name (

name1 [type] [DEFAULT|MATERALIZED|ALIAS expr],

name2 [type] [DEFAULT|MATERALIZED|ALIAS expr]

) ENGINE = engine

CREATE TABLE test\_table\_1 (

title String,

content String,

publish\_time DateTime

)

CREATE TABLE [IF NOT EXISTS] [db\_name1.]table\_name AS [db\_name2.]table\_name2 [ENGINE = engine]，仅仅复制表结构

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name ENGINE = engine AS SELECT，复制表结构，同时数据都会复制一份

DESC table\_name，可以查看表结构

DROP TABLE [IF EXISTS] [db\_name.]table\_name

DEFAULT可以定义默认值

*临时表：CREATE TEMPORARY TABLE，别的都一样，仅仅支持memory引擎，会话级别的，会话结束，表就销毁了，说白了，假设你系统需要用这个，就创建一个临时表，数据都在内存，等你程序运行完毕，表就没了，一般我们不怎么用*

ALTER TABLE table\_name ADD COLUMN [IF NOT EXISTS] name [type] [default] [AFTER name\_after]，增加字段

ALTER TABLE table\_name MODIFY COLUMN [IF EXISTS] name [type] [default]，修改字段

ALTER TABLE table\_name DROP COLUMN [IF EXISTS] name，删除字段

RENAME TABLE [db\_name1.]table\_name1 TO [db\_name2.]table\_name2, ….，重命名

TRUNCATE TABLE [IF EXISTS] [db\_name.]table\_name，清空表数据

MergeTree，重点剖析他内部的原理，有很多种存储引擎，存储引擎内部的原理之后，我们再来学习他不同的表如何指定存储引擎，在内部他会如何运行

分区表：PARTITION BY toYYYYMM(publish\_time)

create table test\_table (id Int, title String, content String, publish\_time Date) engine=MergeTree() partition by toYYYYMM(publish\_time) order by id

然后你插入数据，随着你不同的时间，他会把你数据放入不同的分区里去，不同分区的数据放在各自的目录里，一般在分析型数据库里，分区是很有用的，因为查询的时候指定分区，就可以仅仅扫描分区目录内的数据了

ALTER TABLE table\_name DROP PARTITION partition，删除某个分区

ALTER TABLE table\_name REPLACE PARTITION partition FROM table\_name\_b，复制别的表的分区数据，两个表的分区键相同，表结构相同，其实一般不怎么用这个

ALTER TABLE table\_anme CLEAR COLUMN column\_name IN PARTITION partition，重置分区数据

ALTER TABLE table\_name DETACH/ATTACH PARTITION partition，卸载和装载

视图，简单来说，分为逻辑视图和物理视图，逻辑视图就是一个逻辑上的概念而已，方便你查询用的，物理视图，就是一个物理上独立存储的数据

CREATE VIEW [IF NOT EXISTS] [db\_name.]view\_name AS SELECT

CREATE MATERALIZED VIEW [IF NOT EXISTS] [db\_name.]view\_name [ENGINE = engine] AS SELECT

INSERT INTO [db.]table [(c1,c2,c3…)] VALUES(v1,v2,v3…)，插入数据

insert into test\_table(id, title, content, publish\_time) values(

数据可以删除和修改，但是在clickhouse这种分析型数据库里，是极为重的操作，他是异步的，而且无法回滚，一般不建议针对单条数据做，最好是批量删除和批量修改

ALTER TABLE [db\_name].table\_name DELETE WHERE filter\_expr，批量删除数据，他会重新弄一个新的目录，里面放入删除后的数据，老的目录还不会立即删除，所以他很重！

ALTER TABLE [db\_name.]table\_name UPDATE col1=val1，修改也是同样的

《109\_MergeTree引擎表的创建和存储结构（1）》

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name (

name1 [type] [DEFAULT|MATERIALIZED|ALIAS expr],

name2 [type] [DEFAULT|MATERIALIZED|ALIAS expr]

) ENGINE = MergeTree()

PARTITION BY expr

ORDER BY expr

PRIMARY KEY expr

SAMPLE BY expr

SETTINGS name=value,name=value

PARTITION BY指定的是分区键，选填，如果指定了，那么就会进行数据分区，如果不指定，就只有一个数据分区，其实一般都是建议根据业务特点对数据进行分区的，因为分区后确实对性能有好处

ORDER BY指定的是排序键，这是必填的，指定在一个数据片段里根据什么来排序，一般情况下把主键和排序键指定为一样的，但是也可以指定特殊的排序键，PRIMARY KEY指定主键，选填，如果没填就跟ORDER BY排序键一样，主键会生成一级索引用于查询

如果有分区键，那么每个分区是一个目录

分区目录里面，默认会包含checksums.txt、columns.txt、count.txt、primary.idx、[Column].bin、[Column].mrk、[Column].mrk2，这些文件，如果是有分区键的，那么就会还有额外的partition.dat、minmax\_[Column].idx，如果建立了二级索引，还会有skp\_idx\_[Column].idx和skp\_idx\_[Column].mrk

checksums.txt，是放校验和的文件，主要保存其他各个文件的size以及哈希值，可以用于校验各个文件是否破损

columns.txt，是放所有的字段信息的

count.txt，是放数据分区下的总行数的

primary.idx，是基于主键建立的一级索引

[Column].bin，是列式存储的数据文件，存放一列的数据，用lz4压缩，.mrk文件记录了.bin文件里各个数据的偏移量，查询的时候通过一级索引先找到.mrk里的便宜量，再根据便宜量可以去.bin里读取数据，.mrk2是自适应大小的索引

partition.dat是放的分区字段值，minmax\_[Column].idx是分区下的数据最小值和最大值

skp两个文件，都是二级索引文件

简单来说呢，你往MergeTree的表里灌入数据的时候，会根据分区键去组织数据，但是其实一个分区是会有多个目录的，就是可能同一个分区会有20200101\_1\_1\_0，20200101\_2\_2\_0，那个1或者2之类的，其实是分区内的最小和最大数据块的编号，最后一个0是合并层级，实际上他默认每隔一段时间，比如10分钟，会对一个分区的多个目录做合并

合并完之后可能会有20200101\_1\_2\_1这样的一个东西，然后旧的目录会保留，默认也会每隔8分钟后台做一个清理的

这个事儿，大家心里有个数就行了

《128\_授人以渔：教你自己去拓展学习Clickhouse（4）》

其他的一些表引擎，查询语法，运维管理，分布式架构

你如果要用好clickhouse，了解清楚他的各种表引擎的原理和特性，MergeTree家族引擎，原理大家都已经知道了，对表引擎的工作原理熟悉的基础之上去用他，你大概知道用这个表引擎建了表之后，查询的时候是如何工作的

如何去建立索引，如何分区，如何做优化，如何让他的性能是最高的

《106\_熟悉Clickhouse数据库的DDL语法（3）》

先按照以前的方式去装一台linux虚拟机，接着可以执行下面的安装操作

Clickhouse的rpm安装包下载

https://packagecloud.io/altinity/clickhouse

clickhouse-client-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-common-static-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-19.17.4.11-1.el7.x86\_64.rpm

clickhouse-server-common-19.17.4.11-1.el7.x86\_64.rpm

检查linux虚拟机的cpu是否支持SSE 4.2指令集，那是向量化引擎需要的

grep -q sse4\_2 /proc/cpuinfo && echo "SSE 4.2 supported" || echo "SSE 4.2 not supported"

设置FQDN：hostnamectl –static set-hostname clickhouse-01

验证是否生效：hostname -f

配置hosts文件：vi /etc/hosts，本机ip地址 clickhouse-01

上传4个rpm安装包到/usr/local目录下去

rpm -ivh ./\*.rpm，全部安装，可能会出现一些依赖包缺失，此时缺失什么就百度一下，然后先安装依赖就可以了，yum install -y libicu.x86\_64

安装完毕之后，clickhouse会在服务器上构建一套目录结构

/etc/clickhouse-server：配置文件，config.xml全局配置和users.xml用户配置

/var/lib/clickhouse：数据存储目录，可以修改

/var/log/clickhouse-server：日志目录，可以修改

/etc/security/limits.d/clickhouse.conf：文件句柄数量配置，默认是262144，可以修改文件句柄，在config.xml里可以通过max\_open\_files来指定，linux里一切皆文件

/etc/cron.d/clickhouse-server：crontab定时任务配置，主要是用来恢复异常中断的clickhouse进程的，每隔10s会执行一次定时任务，尝试启动clickhouse进程，如果进程在运行就算了，否则就自动启动

/usr/bin目录下有一些可执行命令：clickhouse、clickhouse-client、clickhouse-server，等等，分别是服务端和客户端

修改config.xml，<path>、<tmp\_path>、<user\_files\_path>，都可以修改一下，接着把你修改的目录设置为clickhouse有权限，chown clickhouse.clickhouse /usr/local/xxx -R

接着启动clickhouse：service clickhouse-server start，默认读取的是/etc/clickhouse-server/config.xml

clickhouse-client，进入客户端，show databases;

客户端支持TCP和HTTP，TCP一般是内部通信以及cli客户端，端口号是9000，HTTP是面向系统的，端口号是8123，cli客户端主要用于调试、测试以及运维，进入之后，就可以DDL、DML以及查询了

但是系统访问一般是用JDBC去使用的

<dependency>

<groupId>ru.yandex.clickhouse</groupId>

<artifactId>clickhouse-jdbc</artifactId>

<version>0.2.4</version>

</dependency>

Class.forNmae(“ru.yandex.clickhouse.clickHouseDriver”);

String url = “jdbc:clickhouse://clickhouse.zhss.com:8123/default”;

String user = “default”;

String password = “”;

Connection connection = DriverManager.getConnection(url, username, password);

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executQuery(“SELECT xxx”);

resultSet.next();

resultSet.getInt(1);

CREATE DATABASE [IF NOT EXISTS] db\_name [ENGINE = engine]

引擎有5种：ordinary（默认）、dictionary、memory、lazy、mysql

创建数据库之后，就会在数据目录下有一个文件夹，metadata目录下也有恢复数据库的SQL文件，show databases可以查看到你创建的数据库

USE db\_name，DROP DATABASE [IF EXISTS] db\_name

其实说白了，这个都跟mysql一样的，不同的业务最好是建自己的库，起码逻辑上和物理上都是可以分开的

数据类型：Int、Float、Decimal、String、FixedString、Uuid、DateTime、DateTime64、Date、Array、Tuple、Enum、Nested、Nullable、Domain

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name (

name1 [type] [DEFAULT|MATERALIZED|ALIAS expr],

name2 [type] [DEFAULT|MATERALIZED|ALIAS expr]

) ENGINE = engine

CREATE TABLE test\_table\_1 (

title String,

content String,

publish\_time DateTime

)

CREATE TABLE [IF NOT EXISTS] [db\_name1.]table\_name AS [db\_name2.]table\_name2 [ENGINE = engine]，仅仅复制表结构

CREATE TABLE [IF NOT EXISTS] [db\_name.]table\_name ENGINE = engine AS SELECT，复制表结构，同时数据都会复制一份

DESC table\_name，可以查看表结构

DROP TABLE [IF EXISTS] [db\_name.]table\_name

DEFAULT可以定义默认值

*临时表：CREATE TEMPORARY TABLE，别的都一样，仅仅支持memory引擎，会话级别的，会话结束，表就销毁了，说白了，假设你系统需要用这个，就创建一个临时表，数据都在内存，等你程序运行完毕，表就没了，一般我们不怎么用*

ALTER TABLE table\_name ADD COLUMN [IF NOT EXISTS] name [type] [default] [AFTER name\_after]，增加字段

ALTER TABLE table\_name MODIFY COLUMN [IF EXISTS] name [type] [default]，修改字段

ALTER TABLE table\_name DROP COLUMN [IF EXISTS] name，删除字段

RENAME TABLE [db\_name1.]table\_name1 TO [db\_name2.]table\_name2, ….，重命名

TRUNCATE TABLE [IF EXISTS] [db\_name.]table\_name，清空表数据

MergeTree，重点剖析他内部的原理，有很多种存储引擎，存储引擎内部的原理之后，我们再来学习他不同的表如何指定存储引擎，在内部他会如何运行

分区表：PARTITION BY toYYYYMM(publish\_time)

然后你插入数据，随着你不同的时间，他会把你数据放入不同的分区里去，不同分区的数据放在各自的目录里，一般在分析型数据库里，分区是很有用的，因为查询的时候指定分区，就可以仅仅扫描分区目录内的数据了

ALTER TABLE table\_name DROP PARTITION partition，删除某个分区

ALTER TABLE table\_name REPLACE PARTITION partition FROM table\_name\_b，复制别的表的分区数据，两个表的分区键相同，表结构相同，其实一般不怎么用这个

ALTER TABLE table\_anme CLEAR COLUMN column\_name IN PARTITION partition，重置分区数据

ALTER TABLE table\_name DETACH/ATTACH PARTITION partition，卸载和装载

视图，简单来说，分为逻辑视图和物理视图，逻辑视图就是一个逻辑上的概念而已，方便你查询用的，物理视图，就是一个物理上独立存储的数据

CREATE VIEW [IF NOT EXISTS] [db\_name.]view\_name AS SELECT

CREATE MATERALIZED VIEW [IF NOT EXISTS] [db\_name.]view\_name [ENGINE = engine] AS SELECT

INSERT INTO [db.]table [(c1,c2,c3…)] VALUES(v1,v2,v3…)，插入数据

数据可以删除和修改，但是在clickhouse这种分析型数据库里，是极为重的操作，他是异步的，而且无法回滚，一般不建议针对单条数据做，最好是批量删除和批量修改

ALTER TABLE [db\_name].table\_name DELETE WHERE filter\_expr，批量删除数据，他会重新弄一个新的目录，里面放入删除后的数据，老的目录还不会立即删除，所以他很重！

ALTER TABLE [db\_name.]table\_name UPDATE col1=val1，修改也是同样的

《119\_Clickhouse对数据TTL机制的支持介绍》

MergeTree这个家族的引擎都讲一下，分别有什么特点和机制，分别适用于什么场景下；其他的引擎；分布式架构原理和机制，副本高可用机制；正式开始做第一个子项目，ES + HBase + Clickhouse，来做一下

互联网保险项目，着手组建兼职的开发团队，打造完全真实的生产级的数据云平台，所有的集群都是真实的，机器都是高配置的，数据都可以达到亿级的模拟，互联网保险系统会从0开始打造一个真实的

真实的数据如何流转到你的大数据平台里去

写代码，做项目，都可以放在真实的环境里跑，积累生产经验

CREATE TABLE t1(

id String,

gmt\_created DateTime,

c1 String TTL gmt\_created + INTERVAL 10 SECOND

)

ENGINE = MergeTree

PARTITION BY toYYYYMM(gmt\_created)

ORDER BY id

INSERT INTO TABLE t1 VALUES(‘110’, now(), ‘C1’)

SELECT \* FROM t1，可以看到数据

10秒过后，OPTIMIZE TABLE t1 FINAL，强制执行数据清理，查询表，SELECT \* FROM t1，就可以看到这条数据没了

上面是列级别的，还有表级别的，在建表的时候带上：

TTL gmt\_created + INTERVAL 1 DAY，那么表里的数据在指定时间过后就会自动清理掉了，不过说实话，这个功能并没有那么的实用，因为一般数据最好都是要长期驻留提供查询的，短期临时的那种数据查询，除非是什么系统里用clickhouse来放中间临时数据

《093\_一个非常高大上的概念：CPU寄存器的向量化执行》

clickhouse用了一个非常高大上的概念，就是向量化执行，这是CPU寄存器的一个特性，也就是说，常规情况下，比如你要处理一波数据，可能会写出来for循环，for(data : datas) { process(data) }，就这样的一种代码

这种代码属于线性重复循环处理多条数据，对于cpu层面来说，此时就会一条指令接着一条执行的执行和处理，依次处理这一波数据，那这个效率当然是有点慢的了，而且的话呢，这个处理的时候，还会每次都从内存里读取数据

Java，并发、线程，JVM专栏，前提是说你已经对Java架构的部分内容，以及儒猿技术窝里的mysql、jvm之类的专栏，都已经学习过了

内存读写的性能大致100ns，1毫秒=1000微秒，1微秒=1000纳秒，1纳秒=1000皮秒

然后呢，这里如果你可以用比较偏向于底层的编程语言，比如说c++一类的，clickhouse就是这种，那么你可以尝试用上cpu的一种特殊指令去处理一波数据，也就是SIMD指令，全称是single instruction multiple data，一条指令处理多条数据

一旦使用了这种指令，那么他就会把多条数据一次性加载到CPU内部的寄存器里去，接着在寄存器层面，用一条指令并行处理多条数据，多条数据的处理就是并行的了，而且寄存器内部的数据读写是300ps级别的

这就是在利用CPU处理数据的时候，他是如何把性能提升到极致的

这种多条数据同时并行处理的方式，就是所谓的向量化执行

除此之外，他还会用多线程并发的处理，因为现在都是多核cpu了，所以完全可以开启多线程分散在多个cpu核上，并行运行，每个线程在单个cpu内存，又可以使用SIMD进行向量化数据处理，所以性能是优化到极致的

《080\_OLAP数据分析到底指的是什么呢？》

clickhouse这个技术，2014年火的是hadoop、storm；2015~2016年火的是spark；2017年开始，kylin、druid、es，很多技术，都火了，还有其他的一些东西；2018年，flink开始就比较火了，数据中台；2019年开始，clickhouse，数据湖

数据仓库和OLAP有一定关系，但是不是完全关联起来的

数据仓库，其实说白了重在对数据进行组织，包括分层，建模，ETL，预处理，主题，集市，预聚合~~、报表~~，等等，就是把数据规范化的准备好，可以使用

在数据仓库之上，是有应用的，比如说BI，也就是基于数仓里的数据跑各种报表，做商业智能决策；也有比如OLAP，这个OLAP和BI也可能是有一定关系的，BI范围更大，BI可能包含了OLAP，但是除了OLAP可能还有别的，比如数据挖掘、机器学习，或者定制化报表，都可能在里面

关于数据仓库的一些基础性的概念，请大家看一下大数据面试突击之离线数仓

OLTP：联机在线事务处理系统

OLAP：联机在线数据分析系统

OLAP是数据应用的一种，就是基于规范组织好的数据，让你进行多维度数据分析，一般都是提供一个产品，让各种人都可以托拉拽的去分析数据，就是比如说，有一份公司销售数据，包含了：部门、销售人员、商品、日期、销量，这么几个字段，这是规范化处理好的一份数据

然后OLAP就是可以对部门、销售人员、商品、日期这么几个维度进行分析

各个部门在每一天的总销量汇总

每个部门的每个销售人员在每一天的业绩汇总

每个部门在每一天对每个商品的销量业绩汇总

对一份规范化组织的数据，任意选取一个或者多个维度，按照这些维度进行分组和聚合，看到每一个多维度分组的聚合指标

OLAP、报表、BI之间的区别和关系是什么呢？

这个OLAP分析的时候，一般会引入一个数据立方体的概念，这个数据立方体是什么意思呢？比如说，现在要基于OLAP方法来分析各个部门在每个日期的每个商品的销量，可以有一个数据立方体的概念

上卷、下钻、切片、切块，旋转

当然，我觉得数据立方体的思维有点奇怪，所以一般我都不采用这种思维来思考OLAP，其实OLAP很简单，就是一份数据，多个维度，各种维度上卷、下钻去分析，所谓的切片和切块，就是某个维度或者某几个维度是固定值，然后分析其他的维度罢了

《095\_Clickhouse的Multi-Master多主架构和数据分片（2）》

clickhouse有一个特殊的地方，就是他并不是master-slave架构，其实大家思考一下，无论是HDFS、HBase、Elasticsearch、Redis、MySQL，等等吧，都是master-slave架构的，HDFS是NameNode作为主控节点，HBase是HMaster作为主控节点，ES也会选举出一个节点作为Master角色，Redis也是Master-Slave主从同步架构，MySQL也是Master-Slave主从同步架构

见到的大多数的分布式系统，其实都是master-slave架构的

hdfs，namenode -> 多个datanode

hbase，hmaster -> 多个hregionserver

ES，多个节点选举出一个master -> 集群其他节点

redis，master + slave的主从架构

mysql，master + slave的主从架构

其实你见到的这些技术，大多都是主从同步架构，但是Clickhouse是非常特殊的，他采用了Multi-Master多主架构，也就是说，他多机器上部署的节点，每个都是Master，你请求到每个节点上的效果都是一样的，这很特别

spring cloud netflix技术栈，源码剖析的课程，eureka，peer-to-peer，每个节点扮演的角色都是一样的，集群里并没有一个所谓的主控节点

HDFS，分片，把一个大文件拆分为很多的小块，拆分小块的过程就是一个分片的过程，不同的小块放在不同的服务器上存储，数据分片+分布式存储

HBase，底层就是基于HDFS的，分布式的，HRegionServer（内存级的数据结构，每台机器分布式处理不同的数据，管理不同的数据部分），Region的拆分其实就是一个数据分片的过程，不同的Region放不同的服务器上管理，分布式管理的过程，Region已经是一个数据分片了，粗粒度的

Region的数据，HFile，也是放在HDFS上的，本身就是拆分为很多block小块，分布式存储在多台服务器上的

ES，索引，shard，不同的索引可以拆分为很多的shard，就是一个数据分片的过程，不同的shard就放在不同的服务器上

redis，redis cluster，redis多实例集群，把数据分散在redis各个服务器上，这也是一种数据分片；mysql，分库分表，不同的服务器上的库里和不同的表里

但是他也能实现数据分片的效果，在他这里的话，一个分片就对应了一个节点，你有多少个节点就是多少个分片，他有本地表和分布式表的概念，本地表就是一个分片，对应的数据就在一个节点上，然后分布式表属于一个路由表的概念，包含的是多个数据分片的信息，有点类似于分库分表里的中间件干的事儿

刚开始你可以就是用一个本地表和一个分片，后续可以增加分片，迁移数据到多个分片的，很灵活，然后可以进行多分片分布式查询

他的查询性能极高，这个我们说过了，SparkSQL/Hive本质是离线查询，分钟级

ES适合的是搜索，而不是聚合分析，他天生就不是用来跑数据指标统计的

MySQL不适合海量数据，即使分库分表，现在也很难进行支撑；分布式数据库，TiDB，适合的是海量数据场景下的OLTP系统，不擅长海量数据分析场景；Kylin+HBase，太重，效果也达不到特别理想

clickhouse，万亿级数据，弄几百台服务器，做好分片，分布式查询，90%查询在1s内，没有任何人可以做到，他未来当之无愧是实时OLAP分析领域的王者

另外他也有副本的概念，就是分片的话，可能就是一个逻辑上的概念，一个节点就对应一个分片，然后副本的话，意思就是如果你设置副本数量是1，那么这个分片就对应一个副本，如果你设置副本数量为2，那么就会有2个副本，在不同的机器上

《089\_Clickhouse是数据库管理系统：这到底有多牛？》

OLAP，clickhouse非常适合海量数据的实时分析，实时OLAP，实时其他分析也是可以的，OLAP指的主要是基于多种维度组合起来进行聚合和分析，加一点条件在里面，其他分析，Clickhouse也可以做到的

数据实时的灌入到clickhouse里去，接下来直接就可以对数据进行分析了

Clickhouse首先是一个数据库管理系统，也就是一个DBMS，也就是说，用他可以增删改数据库，有表的概念，甚至有视图的概念，还可以执行DML操作，可以对数据进行增删改查，是不是感觉太牛了，这都跟MySQL差不多了？

仅仅这一点，他就可以说是一个OLAP数据库了！不像HBase说白了就是一个分布式KV存储，或者HDFS这种就是分布式文件存储系统，Hive那种是数据仓库的一个概念，因为Hive没法做灵活的增删改查

他还有权限控制的功能，控制用户对数据的访问权限，当然，这个比较简单

也有数据备份和恢复机制

当然，还有分布式集群管理的模式，相当于管理了多个数据库

另外他直接就支持关系型数据模型，也就是我们平常在MySQL里见到的那种，还支持较为完备的SQL语法，这样的话，让我们基于他直接进行OLAP数据模型的设计和查询语句的编写，都非常的轻松和简单了

另外，Clickhouse还借鉴了MySQL的存储引擎的概念，他这里也有表引擎的概念，其实都是一个意思，说白了，所谓的存储引擎/表引擎，就是说，当你在执行增删改查的时候，他底层是怎么做的，用什么机制来做

MySQL就有innodb、myisam、memory等多种经典的存储引擎，分别适用于不同的场景；clickhouse也有20多种表引擎，不同的表引擎，对你的数据增删改查的执行都有一定的区别，没有最好，只有最合适

《087\_基于HBase+Kylin实现的OLAP分析架构（3）》

早期那套基于大数据生态的数据仓库+Hive/SparkSQL+MySQL的OLAP分析架构其实跑着还算凑合，但是这里的核心就是在于轻度聚合以后的数据会放到MySQL里去，这个数据量不能太大，毕竟我们是要对MySQL里的数据跑各种维度group by以后的分析语句的，就算是建好了索引，但是单表在百万级的数量已经是极限了

增量的明细数据有1亿， 3个维度，维度1的取值有100种，维度2的取值有1000值，维度3的取值有10000种，100 \* 1000 \* 10000 = 10亿

所以后来慢慢发现这个数据仓库里轻度聚合后的数据量越来越大，经常聚合后的数据量有日增几十万，全表都千万级了，基于MySQL跑的OLAP分析语句越来越困难，都跑不动了，这个时候这套架构就有点问题了

儒猿技术窝的《从0开始带你成为MySQL实战高手》，MySQL的B树索引原理，索引里的数据很多，内存放不下太多的缓存页，索引深度很深，每次就算基于索引搜索，也要在磁盘里大量的磁盘IO才可以

维度1、维度2、维度3

维度1 -> 聚合

维度1 + 维度2 -> 聚合

维度2 + 维度3 -> 聚合

维度1 + 维度2 + 维度3 -> 聚合

聚合好的数据，都是一些key-value对，hbase的架构原理，kv数据，v是面向列族的，列族存储的，列是动态可变的，本质还是kv数据

维度1 -> a、b、c

维度2 -> x、z、y

维度3 -> d、w、s

a+x -> 聚合后的值

a+z -> 聚合后的值

所以后来开始涌现HBase+Kylin的这套架构，Kylin本身就是依赖于HBase的，这套架构的核心就是依然会对数据进行轻度聚合和 汇总，但是预聚合以后的数据放到HBase里去，HBase毕竟是大数据存储，所以放很多数据问题是不太大的，哪怕是聚合后的数据有千万级，亿级的规模，都是ok的

hbase底层是基于hdfs的，所以放海量数据，完全没问题，hbase构建在hdfs之上的数据存储系统，负责对hdfs里的数据进行索引的组织、写入的缓冲、内存数据组织、内存缓存、数据分布式分片、数据负载均衡

然后Kylin会针对预聚合以后的数据提供OLAP分析能力给你，架构和性能上做了大量的优化，海量数据场景下的OLAP分析能力还是不错的，只要你自己封装一个可以托拉拽的界面，底层基于kylin这套架构，就可以搞自助式的OLAP分析了

但是这套架构有一个缺点

就是早期的Kylin仅仅支持离线OLAP，什么意思呢？就是说，都是凌晨跑数据，把昨天的数据预处理好了以后放HBase，然后你分析的数据都是昨天的，也就是分析的是离线数据，你想对今天的数据做实时OLAP，还做不到

当然，Kylin后来也推出了实时OLAP功能，包括什么Druid也是做实时OLAP的，所以实时OLAP后来也出现了，有一段时间，各个公司做OLAP这块，是用Kylin做离线OLAP，Druid做实时OLAP，组合起来用

但是实时OLAP这块概念还没怎么火起来呢，直接clickhouse就横空出世了

《090\_Clickhouse是数据库管理系统：这到底有多牛？（2）》

OLAP，clickhouse非常适合海量数据的实时分析，实时OLAP，实时其他分析也是可以的，OLAP指的主要是基于多种维度组合起来进行聚合和分析，加一点条件在里面，其他分析，Clickhouse也可以做到的

数据实时的灌入到clickhouse里去，接下来直接就可以对数据进行分析了

Clickhouse首先是一个数据库管理系统，也就是一个DBMS，也就是说，用他可以增删改数据库，有表的概念，甚至有视图的概念，还可以执行DML操作，可以对数据进行增删改查，是不是感觉太牛了，这都跟MySQL差不多了？

仅仅这一点，他就可以说是一个OLAP数据库了！不像HBase说白了就是一个分布式KV存储，或者HDFS这种就是分布式文件存储系统，Hive那种是数据仓库的一个概念，因为Hive没法做灵活的增删改查

在大数据技术栈里，面向实时数据的，实时灌入，实时查询的，Hbase、ES、Druid，能够做到像clickhouse这样，能面向实时数据，还能提供DBMS的DDL、DML完备功能的，其实几乎是没有

他还有权限控制的功能，控制用户对数据的访问权限，当然，这个比较简单

也有数据备份和恢复机制

当然，还有分布式集群管理的模式，相当于管理了多个数据库

另外他直接就支持关系型数据模型，也就是我们平常在MySQL里见到的那种，还支持较为完备的SQL语法，这样的话，让我们基于他直接进行OLAP数据模型的设计和查询语句的编写，都非常的轻松和简单了

另外，Clickhouse还借鉴了MySQL的存储引擎的概念，他这里也有表引擎的概念，其实都是一个意思，说白了，所谓的存储引擎/表引擎，就是说，当你在执行增删改查的时候，他底层是怎么做的，用什么机制来做

MySQL就有innodb、myisam、memory等多种经典的存储引擎，分别适用于不同的场景；clickhouse也有20多种表引擎，不同的表引擎，对你的数据增删改查的执行都有一定的区别，没有最好，只有最合适

《088\_横空出世的Clickhouse：实时OLAP领域的主流》

数据仓库+Hive/SparkSQL+MySQL，这个方案基本可以放进历史垃圾堆了，基本是淘汰型的方案；HBase+Kylin，太重了，依赖太多，早期弄离线数据，现在有实时OLAP，用是可以用的，就是嫌弃他太重了，组件太多；Druid，用也是可以用的，但是一直没成主流

kylin的这套方案，最大的痛点，离线OLAP，还是次要的，推实时数据接入，依赖了hbase，上kylin的OLAP方案，hbase -> hdfs，依赖太多了，对很多公司而言，维护成本挺高的，很重

Clickhouse，没任何依赖，开箱即用，跟es一样，而且可以实时灌入数据，不需要预聚合，支持完善SQL，直接可以查，还有完善的DB管理功能，高可用之类的架构都有，性能超高，大数据场景下，比Hive、MySQL之类的快上百倍

俄罗斯的Yandex公司开源的，那是俄罗斯本土搜索引擎，就是相当于国内的百度吧，clickhouse最早是他们自研了用来提供广告流量分析的，后来经过了几次架构迭代和升级，达到了现在的成熟阶段

Yandex自己用clickhouse存了万亿级的数据，90%的查询都是1s内返回

OLAP这个领域，事实上的标准，实时OLAP分析

clickhouse的缺点，不支持事务，他主要是擅长的针对明细数据，按多维度来汇总聚合分析，不擅长根据id去查询单行，不擅长按行删除数据，他是有一些局限性的

Clickhouse架构原理：1周

Clickhouse部署以及使用：1周

Clickhouse内核原理：4周

Clickhouse查询语法：1周

Clickhouse管理运维：1周

数据采集 + kafka + hbase + es + clickhouse，保单查询系统

《098\_Clickhouse对性能极致优化追求的匠心精神》

hadoop、spark、hbase、zookeeper、kafka，优秀，尊重和认可；clickhouse，我是真的发自内心的佩服，俄罗斯这帮搞clickhouse的人，都是世界级的顶尖水平的，他们做这个技术，选用了c系列语言，大量的对os、硬件层面的东西做了优化

眼睛里的，都是Java、Scala这种编程语言级别的特性、OS级别的特性、分布式架构、常规的优化手段，我的技术水平，在某个领域写出来类似这样水准和级别的开源分布式系统，代码写的比他们还漂亮

clickhouse作者，眼睛里看到的是硬件，顶尖，把计算机硬件级别的东西玩儿的溜熟了

在我研究过了clickhouse之后，确实觉得clickhouse是为了追求性能的极致优化，有匠心精神的，所以大家之前也看到了，比如说数据压缩，就是为了提升性能，向量化执行，也是硬件级别的性能提升和优化，更不用说写了大量的代码，在内存里执行group by之类的操作去提升性能，同时大量使用CPU级别的缓存，而不是内存

另外对于各种数据结构和算法，都是大量的测试以及选择性能最优的

所以clickhouse的性能提升是在各种细节上，针对硬件采用低层编程语言和OS级别的机制、硬件级别的特性去提升的，同时在写代码的时候，尽量多的使用高速缓存，算法上也是大幅度的优化，最终让他拥有极致的性能

这帮人的技术水平，甚至可以达到那种国家级的军工级别的水准

总体来说，俄罗斯人还是有他的可敬之处的，别的不说，当年的前苏联在重工业、军事工业是多么强调，包括航空卫星之类的，这个民族应该说当时在前苏联是达到高光时刻的，现在确实是没落了，但是里面民族里的一些基因，还是让他们可以做出让世界眼前一亮的事情，这点确实是我们应该学习的地方

clickhouse，有些同学公司也在用，clickhouse也有这样那样的问题

《084\_基于Hive/SparkSQL实现的OLAP分析架构（2）》

ROLAP，就是关系型OLAP分析，最早就是在Oracle和MySQL里跑OLAP的，用MyISAM之类的专门面向查询的存储引擎，但是基本就是最多亿级数据量，再多就扛不住了，跑一个报表要很长时间

后来就演化到大数据，其实核心就是利用Hive/SparkSQL去进行数仓建模，建立好数据仓库，然后用Hive/SparkSQL去跑OLAP分析，建数据仓库，建到主题这 一层，数据都是明细粒度的，hive表里面

OLAP界面化的应用系统，用户选择几个维度，以及聚合的指标，去跑OLAP分析，用HiveSQL/SparkSQL，到hive表里去跑，基于明细粒度的数据，把你的OLAP分析的结果跑出来

不过说实话，依托数据仓库的OLAP分析，一般不能说直接在页面上用户选择几个维度，然后即席跑一个Hive/SparkSQL作业去针对明细数据进行OLAP分析，再返回结果的，这一般需要几十秒到几分钟，甚至几十分钟不等

我们确实做过这种项目，最早spark项目里的用户分析，就是用这种方式做的，不过当时不是做OLAP，是跑用户分析任务

但是这种方式做OLAP很不靠谱，所以一般来说，当时都是在数仓里最后一层聚合层，做轻度聚合和汇总，你还得确保轻度聚合之后的数据量比较小，什么意思呢？聚合可能会导致数据量大幅度减小，也可能会导致数据量大幅度膨胀

比如一个明细表1000万数据，有3个维度，维度1有10种取值，维度2有20种取值，维度3有100种取值，这个时候基于3个维度轻度聚合，最后表里的数据是10 \* 20 \* 100 = 2w条数据，这就是大幅度减小了数据量

一般如果能大幅度减小数据量，那么轻度聚合后的数据一般可以放到MySQL之类的数据库里去，用MyISAM存储引擎执行查询，做一个OLAP分析系统，用户选择不同的维度条件，直接在MySQL的轻度聚合表里查了，这个速度一般还是ok的，1s内返回结果

这也是早期大数据技术架构里做OLAP分析惯用的方法

而且这种预先把维度进行预聚合的方式，称为MOLAP，就是多维度预聚合的OLAP，在早期Oracle、MySQL时代，都得用物化视图之类的东西放预聚合数据，但是后来在大数据时代，就是在数据仓库的最后一层放预聚合后的数据

《079\_数据仓库和OLAP之间的关系是什么？》

clickhouse这个技术，2014年火的是hadoop、storm；2015~2016年火的是spark；2017年开始，kylin、druid、es，很多技术，都火了，还有其他的一些东西；2018年，flink开始就比较火了，数据中台；2019年开始，clickhouse，数据湖

数据仓库和OLAP有一定关系，但是不是完全关联起来的

数据仓库，其实说白了重在对数据进行组织，包括分层，建模，ETL，预处理，主题，集市，预聚合、报表，等等，就是把数据规范化的准备好，可以使用

在数据仓库之上，是有应用的，比如说BI，也就是基于数仓里的数据跑各种报表，做商业智能决策；也有比如OLAP，这个OLAP和BI也可能是有一定关系的，BI范围更大，BI可能包含了OLAP，但是除了OLAP可能还有别的，比如数据挖掘、机器学习，或者定制化报表，都可能在里面

OLAP是数据应用的一种，就是基于规范组织好的数据，让你进行多维度数据分析，一般都是提供一个产品，让各种人都可以托拉拽的去分析数据，就是比如说，有一份公司销售数据，包含了：部门、销售人员、商品、日期、销量，这么几个字段，这是规范化处理好的一份数据

然后OLAP就是可以对部门、销售人员、商品、日期这么几个维度进行分析

这个OLAP分析的时候，一般会引入一个数据立方体的概念，这个数据立方体是什么意思呢？比如说，现在要基于OLAP方法来分析各个部门在每个日期的每个商品的销量，可以有一个数据立方体的概念

上卷、下钻、切片、切块，旋转

当然，我觉得数据立方体的思维有点奇怪，所以一般我都不采用这种思维来思考OLAP，其实OLAP很简单，就是一份数据，多个维度，各种维度上卷、下钻去分析，所谓的切片和切块，就是某个维度或者某几个维度是固定值，然后分析其他的维度罢了

《085\_基于HBase+Kylin实现的OLAP分析架构（1）》

早期那套基于大数据生态的数据仓库+Hive/SparkSQL+MySQL的OLAP分析架构其实跑着还算凑合，但是这里的核心就是在于轻度聚合以后的数据会放到MySQL里去，这个数据量不能太大，毕竟我们是要对MySQL里的数据跑各种维度group by以后的分析语句的，就算是建好了索引，但是单表在百万级的数量已经是极限了

增量的明细数据有1亿， 3个维度，维度1的取值有100种，维度2的取值有1000值，维度3的取值有10000种，100 \* 1000 \* 10000 = 10亿

所以后来慢慢发现这个数据仓库里轻度聚合后的数据量越来越大，经常聚合后的数据量有日增几十万，全表都千万级了，基于MySQL跑的OLAP分析语句越来越困难，都跑不动了，这个时候这套架构就有点问题了

儒猿技术窝的《从0开始带你成为MySQL实战高手》，MySQL的B树索引原理，索引里的数据很多，内存放不下太多的缓存页，索引深度很深，每次就算基于索引搜索，也要在磁盘里大量的磁盘IO才可以

所以后来开始涌现HBase+Kylin的这套架构，Kylin本身就是依赖于HBase的，这套架构的核心就是依然会对数据进行轻度聚合和 汇总，但是预聚合以后的数据放到HBase里去，HBase毕竟是大数据存储，所以放很多数据问题是不太大的，哪怕是聚合后的数据有千万级，亿级的规模，都是ok的

然后Kylin会针对预聚合以后的数据提供OLAP分析能力给你，架构和性能上做了大量的优化，海量数据场景下的OLAP分析能力还是不错的，只要你自己封装一个可以托拉拽的界面，底层基于kylin这套架构，就可以搞自助式的OLAP分析了

但是这套架构有一个缺点

就是早期的Kylin仅仅支持离线OLAP，什么意思呢？就是说，都是凌晨跑数据，把昨天的数据预处理好了以后放HBase，然后你分析的数据都是昨天的，也就是分析的是离线数据，你想对今天的数据做实时OLAP，还做不到

当然，Kylin后来也推出了实时OLAP功能，包括什么Druid也是做实时OLAP的，所以实时OLAP后来也出现了，有一段时间，各个公司做OLAP这块，是用Kylin做离线OLAP，Druid做实时OLAP，组合起来用

但是实时OLAP这块概念还没怎么火起来呢，直接clickhouse就横空出世了

《097\_Clickhouse的内核级数据模型分析》

DataType，用于进行序列化和反序列化

OLTP的核心，本质就是要高效的对单行数据做CRUD，事务，还要支持快速的根据多个条件检索数据，而不是跑聚合函数来分析

Kylin：OLAP查询引擎 -> MapReduce/Spark离线批处理作业，对数据进行预聚合 -> HBase去进行预聚合数据的存储 -> HDFS进行底层分布式存储

MyCat、Sharding-Sphere，数据库中间件，他们自己本身不提供数据库的功能，可以帮助你对数据库集群进行路由和管理

DataType底层依赖Column和Field，简单来说Column映射的是表里的一列数据，就是这一列的全部数据，如果是列里的某一行的数据，那就是Field来代表的，但是他们也仅仅是映射而已，属于映射型的数据模型，就是代表了某种数据

Block概念，这个Block才是真正执行数据读写的东西呢，他里面负责执行IO输入输出，使用流进行操作

Storage概念：上面那些都是一些基础的东西，其实具体怎么做数据的增删改查，都是有表引擎来决定的，相对应配合的是Parser和Intercepter，就是用来解析SQL以及解释SQL的，这很重要

Function概念：对你的数据执行聚合

《083\_基于Hive/SparkSQL实现的OLAP分析架构（1）》

ROLAP，就是关系型OLAP分析，最早就是在Oracle和MySQL里跑OLAP的，用MyISAM之类的专门面向查询的存储引擎，但是基本就是最多亿级数据量，再多就扛不住了，跑一个报表要很长时间

后来就演化到大数据，其实核心就是利用Hive/SparkSQL去进行数仓建模，建立好数据仓库，然后用Hive/SparkSQL去跑OLAP分析，建数据仓库，建到主题这 一层，数据都是明细粒度的，hive表里面

OLAP界面化的应用系统，用户选择几个维度，以及聚合的指标，去跑OLAP分析，用HiveSQL/SparkSQL，到hive表里去跑，基于明细粒度的数据，把你的OLAP分析的结果跑出来

不过说实话，依托数据仓库的OLAP分析，一般不能说直接在页面上用户选择几个维度，然后即席跑一个Hive/SparkSQL作业去针对明细数据进行OLAP分析，再返回结果的，这一般需要几十秒到几分钟，甚至几十分钟不等

我们确实做过这种项目，最早spark项目里的用户分析，就是用这种方式做的，不过当时不是做OLAP，是跑用户分析任务

但是这种方式做OLAP很不靠谱，所以一般来说，当时都是在数仓里最后一层聚合层，做轻度聚合和汇总，你还得确保轻度聚合之后的数据量比较小，什么意思呢？聚合可能会导致数据量大幅度减小，也可能会导致数据量大幅度膨胀

比如一个明细表1000万数据，有3个维度，维度1有10种取值，维度2有20种取值，维度3有100种取值，这个时候基于3个维度轻度聚合，最后表里的数据是10 \* 20 \* 100 = 2w条数据，这就是大幅度减小了数据量

一般如果能大幅度减小数据量，那么轻度聚合后的数据一般可以放到MySQL之类的数据库里去，用MyISAM存储引擎执行查询，做一个OLAP分析系统，用户选择不同的维度条件，直接在MySQL的轻度聚合表里查了，这个速度一般还是ok的，1s内返回结果

这也是早期大数据技术架构里做OLAP分析惯用的方法

而且这种预先把维度进行预聚合的方式，称为MOLAP，就是多维度预聚合的OLAP，在早期Oracle、MySQL时代，都得用物化视图之类的东西放预聚合数据，但是后来在大数据时代，就是在数据仓库的最后一层放预聚合后的数据

《094\_Clickhouse的Multi-Master多主架构和数据分片（1）》

clickhouse有一个特殊的地方，就是他并不是master-slave架构，其实大家思考一下，无论是HDFS、HBase、Elasticsearch、Redis、MySQL，等等吧，都是master-slave架构的，HDFS是NameNode作为主控节点，HBase是HMaster作为主控节点，ES也会选举出一个节点作为Master角色，Redis也是Master-Slave主从同步架构，MySQL也是Master-Slave主从同步架构

见到的大多数的分布式系统，其实都是master-slave架构的

hdfs，namenode -> 多个datanode

hbase，hmaster -> 多个hregionserver

ES，多个节点选举出一个master -> 集群其他节点

redis，master + slave的主从架构

mysql，master + slave的主从架构

其实你见到的这些技术，大多都是主从同步架构，但是Clickhouse是非常特殊的，他采用了Multi-Master多主架构，也就是说，他多机器上部署的节点，每个都是Master，你请求到每个节点上的效果都是一样的，这很特别

spring cloud netflix技术栈，源码剖析的课程，eureka，peer-to-peer，每个节点扮演的角色都是一样的，集群里并没有一个所谓的主控节点

但是他也能实现数据分片的效果，在他这里的话，一个分片就对应了一个节点，你有多少个节点就是多少个分片，他有本地表和分布式表的概念，本地表就是一个分片，对应的数据就在一个节点上，然后分布式表属于一个路由表的概念，包含的是多个数据分片的信息，有点类似于分库分表里的中间件干的事儿

刚开始你可以就是用一个本地表和一个分片，后续可以增加分片，迁移数据到多个分片的，很灵活，然后可以进行多分片分布式查询

他的查询性能极高，这个我们说过了，SparkSQL/Hive本质是离线查询，分钟级

ES适合的是搜索，而不是聚合分析，他天生就不是用来跑数据指标统计的

MySQL不适合海量数据，即使分库分表，现在也很难进行支撑；分布式数据库，TiDB，适合的是海量数据场景下的OLTP系统，不擅长海量数据分析场景；Kylin+HBase，太重，效果也达不到特别理想

clickhouse，万亿级数据，弄几百台服务器，做好分片，分布式查询，90%查询在1s内，没有任何人可以做到，他未来当之无愧是实时OLAP分析领域的王者

另外他也有副本的概念，就是分片的话，可能就是一个逻辑上的概念，一个节点就对应一个分片，然后副本的话，意思就是如果你设置副本数量是1，那么这个分片就对应一个副本，如果你设置副本数量为2，那么就会有2个副本，在不同的机器上

《086\_基于HBase+Kylin实现的OLAP分析架构（2）》

早期那套基于大数据生态的数据仓库+Hive/SparkSQL+MySQL的OLAP分析架构其实跑着还算凑合，但是这里的核心就是在于轻度聚合以后的数据会放到MySQL里去，这个数据量不能太大，毕竟我们是要对MySQL里的数据跑各种维度group by以后的分析语句的，就算是建好了索引，但是单表在百万级的数量已经是极限了

增量的明细数据有1亿， 3个维度，维度1的取值有100种，维度2的取值有1000值，维度3的取值有10000种，100 \* 1000 \* 10000 = 10亿

所以后来慢慢发现这个数据仓库里轻度聚合后的数据量越来越大，经常聚合后的数据量有日增几十万，全表都千万级了，基于MySQL跑的OLAP分析语句越来越困难，都跑不动了，这个时候这套架构就有点问题了

儒猿技术窝的《从0开始带你成为MySQL实战高手》，MySQL的B树索引原理，索引里的数据很多，内存放不下太多的缓存页，索引深度很深，每次就算基于索引搜索，也要在磁盘里大量的磁盘IO才可以

维度1、维度2、维度3

维度1 -> 聚合

维度1 + 维度2 -> 聚合

维度2 + 维度3 -> 聚合

维度1 + 维度2 + 维度3 -> 聚合

聚合好的数据，都是一些key-value对，hbase的架构原理，kv数据，v是面向列族的，列族存储的，列是动态可变的，本质还是kv数据

维度1 -> a、b、c

维度2 -> x、z、y

维度3 -> d、w、s

a+x -> 聚合后的值

a+z -> 聚合后的值

所以后来开始涌现HBase+Kylin的这套架构，Kylin本身就是依赖于HBase的，这套架构的核心就是依然会对数据进行轻度聚合和 汇总，但是预聚合以后的数据放到HBase里去，HBase毕竟是大数据存储，所以放很多数据问题是不太大的，哪怕是聚合后的数据有千万级，亿级的规模，都是ok的

hbase底层是基于hdfs的，所以放海量数据，完全没问题，hbase构建在hdfs之上的数据存储系统，负责对hdfs里的数据进行索引的组织、写入的缓冲、内存数据组织、内存缓存、数据分布式分片、数据负载均衡

然后Kylin会针对预聚合以后的数据提供OLAP分析能力给你，架构和性能上做了大量的优化，海量数据场景下的OLAP分析能力还是不错的，只要你自己封装一个可以托拉拽的界面，底层基于kylin这套架构，就可以搞自助式的OLAP分析了

但是这套架构有一个缺点

就是早期的Kylin仅仅支持离线OLAP，什么意思呢？就是说，都是凌晨跑数据，把昨天的数据预处理好了以后放HBase，然后你分析的数据都是昨天的，也就是分析的是离线数据，你想对今天的数据做实时OLAP，还做不到

当然，Kylin后来也推出了实时OLAP功能，包括什么Druid也是做实时OLAP的，所以实时OLAP后来也出现了，有一段时间，各个公司做OLAP这块，是用Kylin做离线OLAP，Druid做实时OLAP，组合起来用

但是实时OLAP这块概念还没怎么火起来呢，直接clickhouse就横空出世了

《091\_Clickhouse的列式存储指的到底是什么意思？》

Clickhouse首先是一个数据库管理系统，也就是一个DBMS，也就是说，用他可以增删改数据库，有表的概念，甚至有视图的概念，还可以执行DML操作，可以对数据进行增删改查，是不是感觉太牛了，这都跟MySQL差不多了？

仅仅这一点，他就可以说是一个OLAP数据库了！不像HBase说白了就是一个KV存储，或者HDFS这种就是分布式文件存储，Hive那种是数据仓库的一个概念，因为Hive没法做灵活的增删改查

他还有权限控制的功能，控制用户对数据的访问权限，当然，这个比较简单

也有数据备份和恢复机制

当然，还有分布式集群管理的模式

hbase课程，好好的看过了，理解列族存储是什么意思，会作为一个单独的存储单元，进行存储，region（按行切割），每个region在存储的时候，底层是拆分为多个存储单元，每个存储单元是放一个列族的数据，会在底层以HFile的形式放在hdfs上

列式存储，这是一个极为重要的概念，如果你是行式存储，就是MySQL那样，每一行数据都放在一个文件里，一个文件有N多行数据，如果你查询的时候只要部分字段，这个时候就得扫描一行数据，然后从里面提取部分字段，这个效率不高

在磁盘IO的时候，要把每一行的20~30个字段都读取出来，提取2~3个字段

列式存储，就是说按列来进行存储，不同列的数据存储在不同的文件里，读取部分字段，只要直接读取部分字段的文件就可以了，而且一般列式存储还会进行数据压缩，让数据更小，读取性能更高

这个数据压缩，本质上其实就是扫描数据集，把里面重复的部分省略，用一些offset来替代，避免存储重复的数据，尽可能把一份数据用最下的空间来存储，比如说helloworld，和low，如果放在一起存储，那么可能会存储为

helloworld\_(7,3)，意思就是，往前7个单位，然后可以匹配上3个重复的字符，其实真实的压缩算法要复杂很多的，压缩过后往往会大幅度降低数据大小，但是格式也会大幅度的改变，没法直接读取的

所以行式存储的数据，往往他重复的部分不多，所以压缩效率不高；列式存储，一个文件里就一个字段的值，很多是重复的，此时压缩效率会比较好，所以列式存储之后，一般都会进行数据压缩

这样读取数据的时候性能高，网络传输速度高

clickhouse就是列式存储，每个列的数据放不同的文件，压缩算法是lz4，很多时候，压缩比可以达到N:1，比如压缩前大小是7，压缩后大小是1，就这个意思，压缩之后可以大幅度的减少数据存储，提升磁盘IO和网络传输的速度

《082\_BI系统与OLAP分析之间的关系是什么？》

clickhouse这个技术，2014年火的是hadoop、storm；2015~2016年火的是spark；2017年开始，kylin、druid、es，很多技术，都火了，还有其他的一些东西；2018年，flink开始就比较火了，数据中台；2019年开始，clickhouse，数据湖

数据仓库和OLAP有一定关系，但是不是完全关联起来的

数据仓库，其实说白了重在对数据进行组织，包括分层，建模，ETL，预处理，主题，集市，预聚合~~、报表~~，等等，就是把数据规范化的准备好，可以使用

在数据仓库之上，是有应用的，比如说BI，也就是基于数仓里的数据跑各种报表，做商业智能决策；也有比如OLAP，这个OLAP和BI也可能是有一定关系的，BI范围更大，BI可能包含了OLAP，但是除了OLAP可能还有别的，比如数据挖掘、机器学习，或者定制化报表，都可能在里面

关于数据仓库的一些基础性的概念，请大家看一下大数据面试突击之离线数仓

OLTP：联机在线事务处理系统

OLAP：联机在线数据分析系统

OLAP是数据应用的一种，就是基于规范组织好的数据，让你进行多维度数据分析，一般都是提供一个产品，让各种人都可以托拉拽的去分析数据，就是比如说，有一份公司销售数据，包含了：部门、销售人员、商品、日期、销量，这么几个字段，这是规范化处理好的一份数据

然后OLAP就是可以对部门、销售人员、商品、日期这么几个维度进行分析

各个部门在每一天的总销量汇总

每个部门的每个销售人员在每一天的业绩汇总

每个部门在每一天对每个商品的销量业绩汇总

对一份规范化组织的数据，任意选取一个或者多个维度，按照这些维度进行分组和聚合，看到每一个多维度分组的聚合指标

OLAP、报表、BI之间的区别和关系是什么呢？

数据平台（开源技术的运维/二次开发 + Java自研）、数据仓库（SQL、数仓建模、数仓治理、开源技术的使用）、数据应用（开源技术的使用 + Java开发）

OLAP分析产品，一般是这样子的，一般都会提供对应的界面给用户，托拉拽，可以自由选择一些维度，选择某一份数据，拖拉选择里面的某几个维度，然后选择聚合函数或者方法，然后OLAP分析的产品，在底层就会针对那份数据，自动按照你选择的这些维度进行分组和聚合

这个OLAP分析的时候，一般会引入一个数据立方体的概念，这个数据立方体是什么意思呢？比如说，现在要基于OLAP方法来分析各个部门在每个日期的每个商品的销量，可以有一个数据立方体的概念

上卷、下钻、切片、切块，旋转

当然，我觉得数据立方体的思维有点奇怪，所以一般我都不采用这种思维来思考OLAP，其实OLAP很简单，就是一份数据，多个维度，各种维度上卷、下钻去分析，所谓的切片和切块，就是某个维度或者某几个维度是固定值，然后分析其他的维度罢了

数据仓库，专门做好数据，依托于数据平台提供的开源技术和一些工具系统，去把数仓做好，数据都在开源技术平台里，OLAP分析应用，一般面向公司里所有对数据有需求的人，都可以自助式的去看数据

BI，商业智能，对数据进行深入的分析和挖掘，产出很多有价值的信息，提供给公司的中层和高层，了解公司运转的情况，辅助他们进行决策，这是BI系统，范围其实很大，做很多的报表，很多报表业务逻辑非常复杂，就不是简单的找几个维度分组聚合一下就可以的，报表需要写非常复杂的SQL

机器学习来做数据挖掘，运用机器学习这种手段，去深度的挖掘数据，从里面挖掘出来有价值的信息

BI系统，都会基于SQL写报表 + 机器学习搞数据挖掘，两种方式，去进行大量的复杂的分析，拿到很多的结果，通过界面提供各种数据报表以及报告，给公司的中高层去看，这里的数据分析任务都很复杂，必须有研发人员去进行开发

通过机器学习的平台跑机器学习算法，针对的是数据仓库里的数据，跑出来结果

OLAP，自助式的，公司里各种想看数据的中低层的员工，都可以自助式的使用OLAP去分析公司里的数据，得到自己想要的结论

BI系统里，有可能也会涉及到基于OLAP技术，基于clickhouse去开发一些分析功能，提供一些类似于OLAP分析的功能

自助式、探索式的BI系统，提供给你大量的界面化的组件，你可以自由的选择、托拉拽，组合出来一套极为复杂的你想执行的分析，他在底层可能会自动生成复杂SQL去跑你想要的数据，甚至自动去跑机器学习算法，拿你想要的数据

保单日常数据分析

保单数据本身也是一份数据，有一些维度，clickhouse做一些OLAP分析，可以在界面上，提供一些维度选择 ，你可以自由的选择一些维度，拿到不同维度组合下的聚合指标，保单的日常指标

《081\_如何用数据立方体思维来理解OLAP分析？》

clickhouse这个技术，2014年火的是hadoop、storm；2015~2016年火的是spark；2017年开始，kylin、druid、es，很多技术，都火了，还有其他的一些东西；2018年，flink开始就比较火了，数据中台；2019年开始，clickhouse，数据湖

数据仓库和OLAP有一定关系，但是不是完全关联起来的

数据仓库，其实说白了重在对数据进行组织，包括分层，建模，ETL，预处理，主题，集市，预聚合~~、报表~~，等等，就是把数据规范化的准备好，可以使用

在数据仓库之上，是有应用的，比如说BI，也就是基于数仓里的数据跑各种报表，做商业智能决策；也有比如OLAP，这个OLAP和BI也可能是有一定关系的，BI范围更大，BI可能包含了OLAP，但是除了OLAP可能还有别的，比如数据挖掘、机器学习，或者定制化报表，都可能在里面

关于数据仓库的一些基础性的概念，请大家看一下大数据面试突击之离线数仓

OLTP：联机在线事务处理系统

OLAP：联机在线数据分析系统

OLAP是数据应用的一种，就是基于规范组织好的数据，让你进行多维度数据分析，一般都是提供一个产品，让各种人都可以托拉拽的去分析数据，就是比如说，有一份公司销售数据，包含了：部门、销售人员、商品、日期、销量，这么几个字段，这是规范化处理好的一份数据

然后OLAP就是可以对部门、销售人员、商品、日期这么几个维度进行分析

各个部门在每一天的总销量汇总

每个部门的每个销售人员在每一天的业绩汇总

每个部门在每一天对每个商品的销量业绩汇总

对一份规范化组织的数据，任意选取一个或者多个维度，按照这些维度进行分组和聚合，看到每一个多维度分组的聚合指标

OLAP、报表、BI之间的区别和关系是什么呢？

数据平台（开源技术的运维/二次开发 + Java自研）、数据仓库（SQL、数仓建模、数仓治理、开源技术的使用）、数据应用（开源技术的使用 + Java开发）

OLAP分析产品，一般是这样子的，一般都会提供对应的界面给用户，托拉拽，可以自由选择一些维度，选择某一份数据，拖拉选择里面的某几个维度，然后选择聚合函数或者方法，然后OLAP分析的产品，在底层就会针对那份数据，自动按照你选择的这些维度进行分组和聚合

这个OLAP分析的时候，一般会引入一个数据立方体的概念，这个数据立方体是什么意思呢？比如说，现在要基于OLAP方法来分析各个部门在每个日期的每个商品的销量，可以有一个数据立方体的概念

上卷、下钻、切片、切块，旋转

当然，我觉得数据立方体的思维有点奇怪，所以一般我都不采用这种思维来思考OLAP，其实OLAP很简单，就是一份数据，多个维度，各种维度上卷、下钻去分析，所谓的切片和切块，就是某个维度或者某几个维度是固定值，然后分析其他的维度罢了

《092\_Clickhouse的数据压缩是怎么做的，意义是什么？》

Clickhouse首先是一个数据库管理系统，也就是一个DBMS，也就是说，用他可以增删改数据库，有表的概念，甚至有视图的概念，还可以执行DML操作，可以对数据进行增删改查，是不是感觉太牛了，这都跟MySQL差不多了？

仅仅这一点，他就可以说是一个OLAP数据库了！不像HBase说白了就是一个KV存储，或者HDFS这种就是分布式文件存储，Hive那种是数据仓库的一个概念，因为Hive没法做灵活的增删改查

他还有权限控制的功能，控制用户对数据的访问权限，当然，这个比较简单

也有数据备份和恢复机制

当然，还有分布式集群管理的模式

列式存储，这是一个极为重要的概念，如果你是行式存储，就是MySQL那样，每一行数据都放在一个文件里，一个文件有N多行数据，如果你查询的时候只要部分字段，这个时候就得扫描一行数据，然后从里面提取部分字段，这个效率不高

列式存储，就是说按列来进行存储，不同列的数据存储在不同的文件里，读取部分字段，只要直接读取部分字段的文件就可以了，而且一般列式存储还会进行数据压缩，让数据更小，读取性能更高

这个数据压缩，本质上其实就是扫描数据集，把里面重复的部分省略，用一些offset来替代，避免存储重复的数据，尽可能把一份数据用最下的空间来存储，比如说helloworld，和low，如果放在一起存储，那么可能会存储为

helloworld\_(7,3)，意思就是，往前7个单位，然后可以匹配上3个重复的字符，其实真实的压缩算法要复杂很多的，压缩过后往往会大幅度降低数据大小，但是格式也会大幅度的改变，没法直接读取的

所以行式存储的数据，往往他重复的部分不多，所以压缩效率不高；列式存储，一个文件里就一个字段的值，很多是重复的，此时压缩效率会比较好，所以列式存储之后，一般都会进行数据压缩

这样读取数据的时候性能高，网络传输速度高

clickhouse就是列式存储，每个列的数据放不同的文件，压缩算法是lz4，很多时候，压缩比可以达到N:1，比如压缩前大小是7，压缩后大小是1，就这个意思，压缩之后可以大幅度的减少数据存储，提升磁盘IO和网络传输的速度

《096\_Clickhouse的Multi-Master多主架构和数据分片（3）》

clickhouse有一个特殊的地方，就是他并不是master-slave架构，其实大家思考一下，无论是HDFS、HBase、Elasticsearch、Redis、MySQL，等等吧，都是master-slave架构的，HDFS是NameNode作为主控节点，HBase是HMaster作为主控节点，ES也会选举出一个节点作为Master角色，Redis也是Master-Slave主从同步架构，MySQL也是Master-Slave主从同步架构

见到的大多数的分布式系统，其实都是master-slave架构的

hdfs，namenode -> 多个datanode

hbase，hmaster -> 多个hregionserver

ES，多个节点选举出一个master -> 集群其他节点

redis，master + slave的主从架构

mysql，master + slave的主从架构

其实你见到的这些技术，大多都是主从同步架构，但是Clickhouse是非常特殊的，他采用了Multi-Master多主架构，也就是说，他多机器上部署的节点，每个都是Master，你请求到每个节点上的效果都是一样的，这很特别

spring cloud netflix技术栈，源码剖析的课程，eureka，peer-to-peer，每个节点扮演的角色都是一样的，集群里并没有一个所谓的主控节点

HDFS，分片，把一个大文件拆分为很多的小块，拆分小块的过程就是一个分片的过程，不同的小块放在不同的服务器上存储，数据分片+分布式存储

HBase，底层就是基于HDFS的，分布式的，HRegionServer（内存级的数据结构，每台机器分布式处理不同的数据，管理不同的数据部分），Region的拆分其实就是一个数据分片的过程，不同的Region放不同的服务器上管理，分布式管理的过程，Region已经是一个数据分片了，粗粒度的

Region的数据，HFile，也是放在HDFS上的，本身就是拆分为很多block小块，分布式存储在多台服务器上的

ES，索引，shard，不同的索引可以拆分为很多的shard，就是一个数据分片的过程，不同的shard就放在不同的服务器上

redis，redis cluster，redis多实例集群，把数据分散在redis各个服务器上，这也是一种数据分片；mysql，分库分表，不同的服务器上的库里和不同的表里

但是他也能实现数据分片的效果，在他这里的话，一个分片就对应了一个节点，你有多少个节点就是多少个分片，他有本地表和分布式表的概念，本地表就是一个分片，对应的数据就在一个节点上，然后分布式表属于一个路由表的概念，包含的是多个数据分片的信息，有点类似于分库分表里的中间件干的事儿

刚开始你可以就是用一个本地表和一个分片，后续可以增加分片，迁移数据到多个分片的，很灵活，然后可以进行多分片分布式查询

他的查询性能极高，这个我们说过了，SparkSQL/Hive本质是离线查询，分钟级

ES适合的是搜索，而不是聚合分析，他天生就不是用来跑数据指标统计的

MySQL不适合海量数据，即使分库分表，现在也很难进行支撑；分布式数据库，TiDB，适合的是海量数据场景下的OLTP系统，不擅长海量数据分析场景；Kylin+HBase+HDFS，太重，效果也达不到特别理想

clickhouse，万亿级数据，弄几百台服务器，做好分片，分布式查询，90%查询在1s内，没有任何人可以做到，他未来当之无愧是实时OLAP分析领域的王者

另外他也有副本的概念，就是分片的话，可能就是一个逻辑上的概念，一个节点就对应一个分片，然后副本的话，意思就是如果你设置副本数量是1，那么这个分片就对应一个副本，如果你设置副本数量为2，那么就会有2个副本，在不同的机器上



