**hive笔记**

目录

[一、Hive安装部署 6](#_Toc103850893)

[1、规划 6](#_Toc103850894)

[2、Hive安装模式 6](#_Toc103850895)

[3、CentOS7下安装部署MySQL5 6](#_Toc103850896)

[4、配置MySQL5 7](#_Toc103850897)

[5、Hive3.1.3安装部署 10](#_Toc103850898)

[6、初始化MySQL数据库，提供元数据服务 16](#_Toc103850899)

[7、启动hive 17](#_Toc103850900)

[8、通过进程与浏览器查看 17](#_Toc103850901)

[9、客户端工具测试 18](#_Toc103850902)

[10、停止hive 23](#_Toc103850903)

[11、虚拟机创建快照，方便恢复 24](#_Toc103850904)

[二、Hive概述 25](#_Toc103850905)

[1、Hive版本—词频统计 25](#_Toc103850906)

[2、业务分析 27](#_Toc103850907)

[3、Hive概述 27](#_Toc103850908)

[4、大数据生态圈中的hive 28](#_Toc103850909)

[5、hive优点与缺点 29](#_Toc103850910)

[6、Hive应用场景 30](#_Toc103850911)

[7、Facebook的hive应用 30](#_Toc103850912)

[8、Hive架构 31](#_Toc103850913)

[9、Hive运行机制 32](#_Toc103850914)

[10、hive和hadoop关系 33](#_Toc103850915)

[11、Hive和RDBMS比较 34](#_Toc103850916)

[12、hive设计特征 34](#_Toc103850917)

[13、Hive运行模式 36](#_Toc103850918)

[14、hive元数据 36](#_Toc103850919)

[15、hive原语 36](#_Toc103850920)

[16、hive数据存储 37](#_Toc103850921)

[17、Hive访问 38](#_Toc103850922)

[三、Hive数据定义语言DDL 39](#_Toc103850923)

[官网参考链接 39](#_Toc103850924)

[1、HQL分类 39](#_Toc103850925)

[2、数据库操作 40](#_Toc103850926)

[3、创建表 40](#_Toc103850927)

[4、数据类型 43](#_Toc103850928)

[5、表的维护 48](#_Toc103850929)

[6、控制字符 49](#_Toc103850930)

[7、数据加载 50](#_Toc103850931)

[8、约束 52](#_Toc103850932)

[9、事务表 53](#_Toc103850933)

[10、外部表 56](#_Toc103850934)

[11、分区表 57](#_Toc103850935)

[12、桶表 60](#_Toc103850936)

[13、显示命令 64](#_Toc103850937)

[14、插入数据 64](#_Toc103850938)

[15、导出数据 66](#_Toc103850939)

[16、清空表数据 67](#_Toc103850940)

[四、 67](#_Toc103850941)

[1、 67](#_Toc103850942)

[2、 67](#_Toc103850943)

[3、 68](#_Toc103850944)

[4、 68](#_Toc103850945)

[5、 68](#_Toc103850946)

[6、 68](#_Toc103850947)

[7、 69](#_Toc103850948)

[8、 69](#_Toc103850949)

[9、 69](#_Toc103850950)

[10、 69](#_Toc103850951)

[五、 70](#_Toc103850952)

[1、 70](#_Toc103850953)

[2、 70](#_Toc103850954)

[3、 70](#_Toc103850955)

[4、 70](#_Toc103850956)

[5、 71](#_Toc103850957)

[6、 71](#_Toc103850958)

[7、 71](#_Toc103850959)

[8、 71](#_Toc103850960)

[9、 71](#_Toc103850961)

[10、 72](#_Toc103850962)

[六、 72](#_Toc103850963)

[1、 72](#_Toc103850964)

[2、 72](#_Toc103850965)

[3、 73](#_Toc103850966)

[4、 73](#_Toc103850967)

[5、 73](#_Toc103850968)

[6、 73](#_Toc103850969)

[7、 73](#_Toc103850970)

[8、 74](#_Toc103850971)

[9、 74](#_Toc103850972)

[10、 74](#_Toc103850973)

[七、 74](#_Toc103850974)

[1、 74](#_Toc103850975)

[2、 75](#_Toc103850976)

[3、 75](#_Toc103850977)

[4、 75](#_Toc103850978)

[5、 75](#_Toc103850979)

[6、 76](#_Toc103850980)

[7、 76](#_Toc103850981)

[8、 76](#_Toc103850982)

[9、 76](#_Toc103850983)

[10、 76](#_Toc103850984)

[八、 77](#_Toc103850985)

[1、 77](#_Toc103850986)

[2、 77](#_Toc103850987)

[3、 77](#_Toc103850988)

[4、 77](#_Toc103850989)

[5、 78](#_Toc103850990)

[6、 78](#_Toc103850991)

[7、 78](#_Toc103850992)

[8、 78](#_Toc103850993)

[9、 79](#_Toc103850994)

[10、 79](#_Toc103850995)

[九、 79](#_Toc103850996)

[1、 79](#_Toc103850997)

[2、 79](#_Toc103850998)

[3、 80](#_Toc103850999)

[4、 80](#_Toc103851000)

[5、 80](#_Toc103851001)

[6、 80](#_Toc103851002)

[7、 80](#_Toc103851003)

[8、 81](#_Toc103851004)

[9、 81](#_Toc103851005)

[10、 81](#_Toc103851006)

[十、 81](#_Toc103851007)

[1、 81](#_Toc103851008)

[2、 82](#_Toc103851009)

[3、 82](#_Toc103851010)

[4、 82](#_Toc103851011)

[5、 82](#_Toc103851012)

[6、 83](#_Toc103851013)

[7、 83](#_Toc103851014)

[8、 83](#_Toc103851015)

[9、 83](#_Toc103851016)

[10、 83](#_Toc103851017)

[十一、 84](#_Toc103851018)

[1、 84](#_Toc103851019)

[2、 84](#_Toc103851020)

[3、 84](#_Toc103851021)

[4、 84](#_Toc103851022)

[5、 85](#_Toc103851023)

[6、 85](#_Toc103851024)

[7、 85](#_Toc103851025)

[8、 85](#_Toc103851026)

[9、 86](#_Toc103851027)

[10、 86](#_Toc103851028)

[十二、 86](#_Toc103851029)

[1、 86](#_Toc103851030)

[2、 86](#_Toc103851031)

[3、 87](#_Toc103851032)

[4、 87](#_Toc103851033)

[5、 87](#_Toc103851034)

[6、 87](#_Toc103851035)

[7、 87](#_Toc103851036)

[8、 88](#_Toc103851037)

[9、 88](#_Toc103851038)

[10、 88](#_Toc103851039)

[十三、 88](#_Toc103851040)

[1、 88](#_Toc103851041)

[2、 89](#_Toc103851042)

[3、 89](#_Toc103851043)

[4、 89](#_Toc103851044)

[5、 89](#_Toc103851045)

[6、 90](#_Toc103851046)

[7、 90](#_Toc103851047)

[8、 90](#_Toc103851048)

[9、 90](#_Toc103851049)

[10、 90](#_Toc103851050)

[十四、 91](#_Toc103851051)

[1、 91](#_Toc103851052)

[2、 91](#_Toc103851053)

[3、 91](#_Toc103851054)

[4、 91](#_Toc103851055)

[5、 92](#_Toc103851056)

[6、 92](#_Toc103851057)

[7、 92](#_Toc103851058)

[8、 92](#_Toc103851059)

[9、 93](#_Toc103851060)

[10、 93](#_Toc103851061)

[十五、 93](#_Toc103851062)

[1、 93](#_Toc103851063)

[2、 93](#_Toc103851064)

[3、 94](#_Toc103851065)

[4、 94](#_Toc103851066)

[5、 94](#_Toc103851067)

[6、 94](#_Toc103851068)

[7、 95](#_Toc103851069)

[8、 95](#_Toc103851070)

[9、 95](#_Toc103851071)

[10、 95](#_Toc103851072)

# 一、Hive安装部署

## 1、规划

|  |  |  |
| --- | --- | --- |
| IP | hostname | actor |
| 192.168.245.151 | master | NameNode / HMaster / ZookKeeper / RDBMS / Hive |
| 192.168.245.152 | slave1 | DataNode / HRegionServer / ZookKeeper / Hive |
| 192.168.245.153 | slave2 | DataNode / HRegionServer / ZookKeeper / Hive |

## 2、Hive安装模式

Derby

hive将数据存储在HDFS中

元数据默认存在在hive自带的数据库derby中

Derby不支持多用户并发访问

用于测试

Local

使用MySQL或其它数据库替换Derby来存储元数据

支持多用户并发访问

仅部署一个节点服务器

用于开发模式

Remote

基于Local方式

MySQL数据所在节点提供metastore service服务，其它节点连接该服务获取元数据

支持多用户并发访问

所有节点同时支持多用户并发访问，简单说就是支持N个大数据分析师同时使用

用于生产环境

## 3、CentOS7下安装部署MySQL5

更新协议

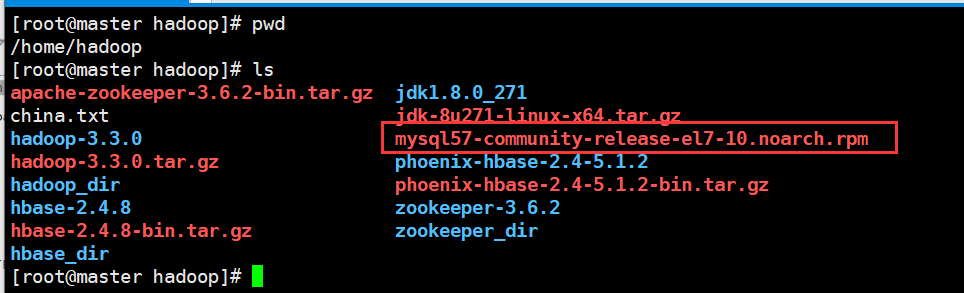
rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022



下载并安装MySQL5

wget -i -c <http://dev.mysql.com/get/mysql57-community-release-el7-10.noarch.rpm>



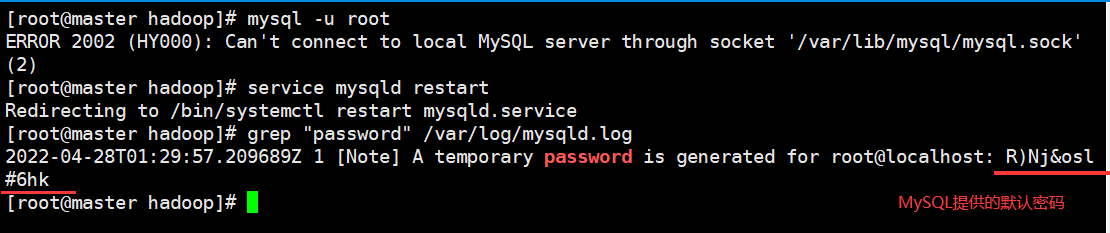






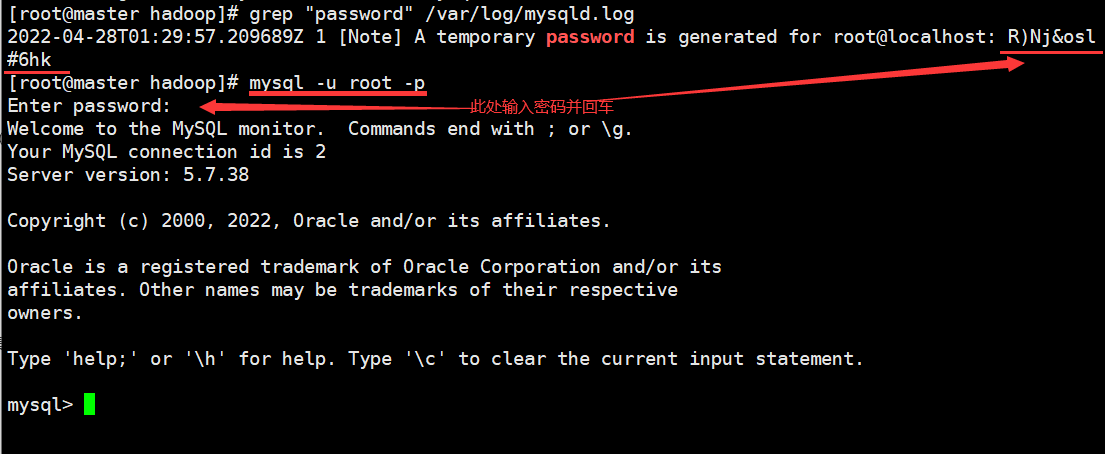
## 4、配置MySQL5

查看默认密码



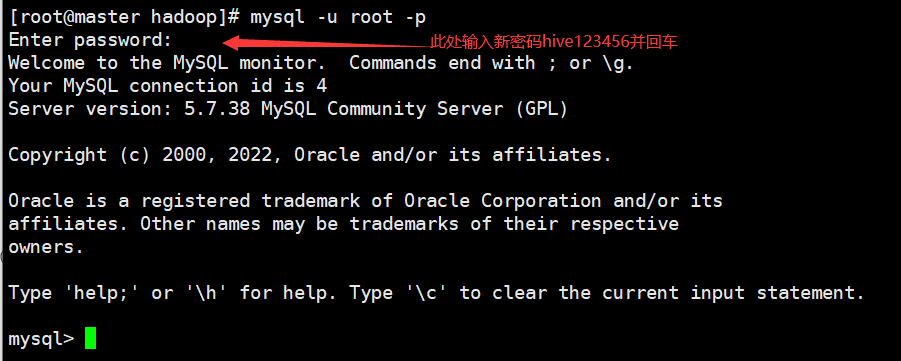
R)Nj&osl#6hk

修改密码

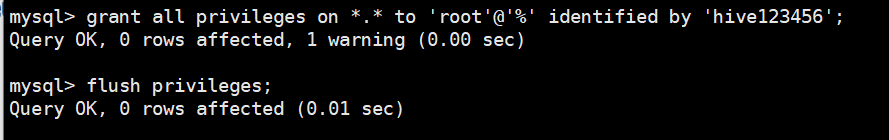




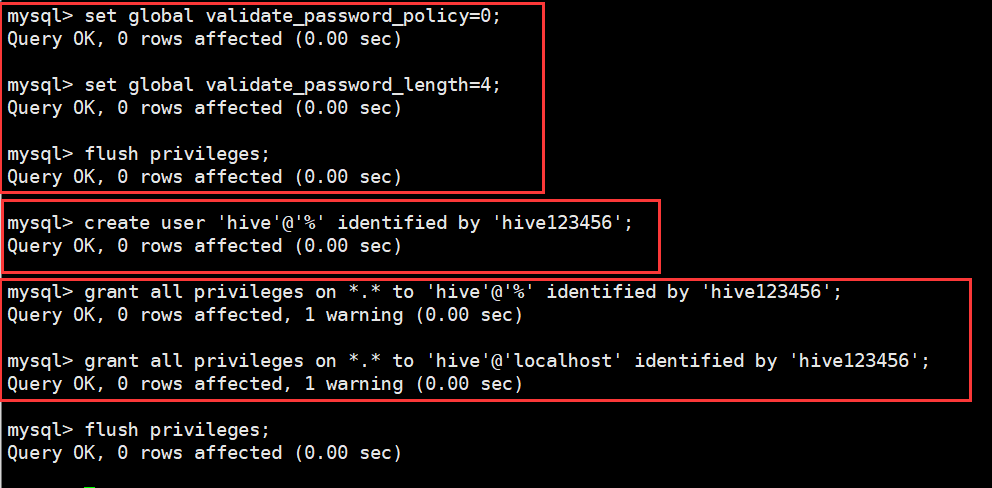
使用新密码重新登录



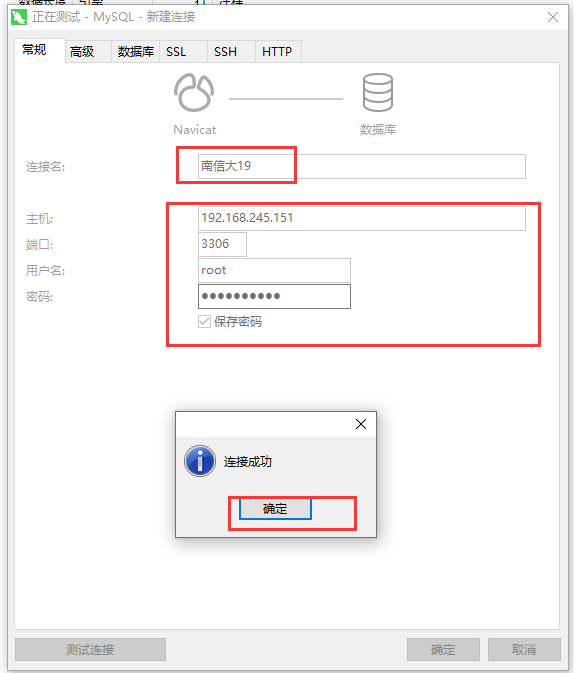
设置允许远程登录

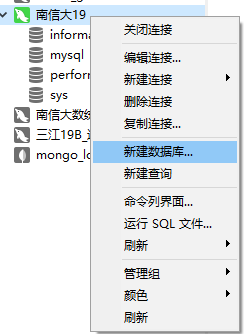


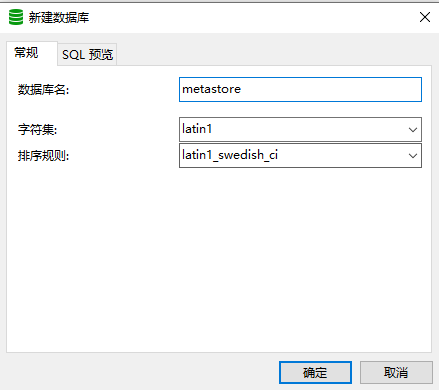
创建用户hive并允许远程登录



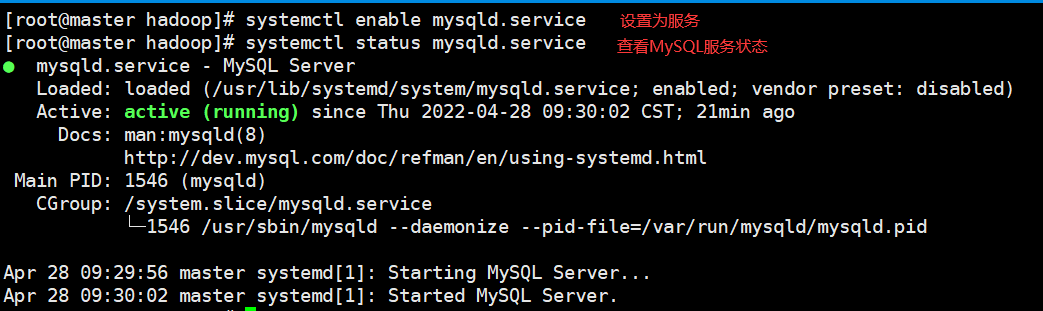
win10连接MySQL并创建新的数据库







查看服务并设置MySQL为服务



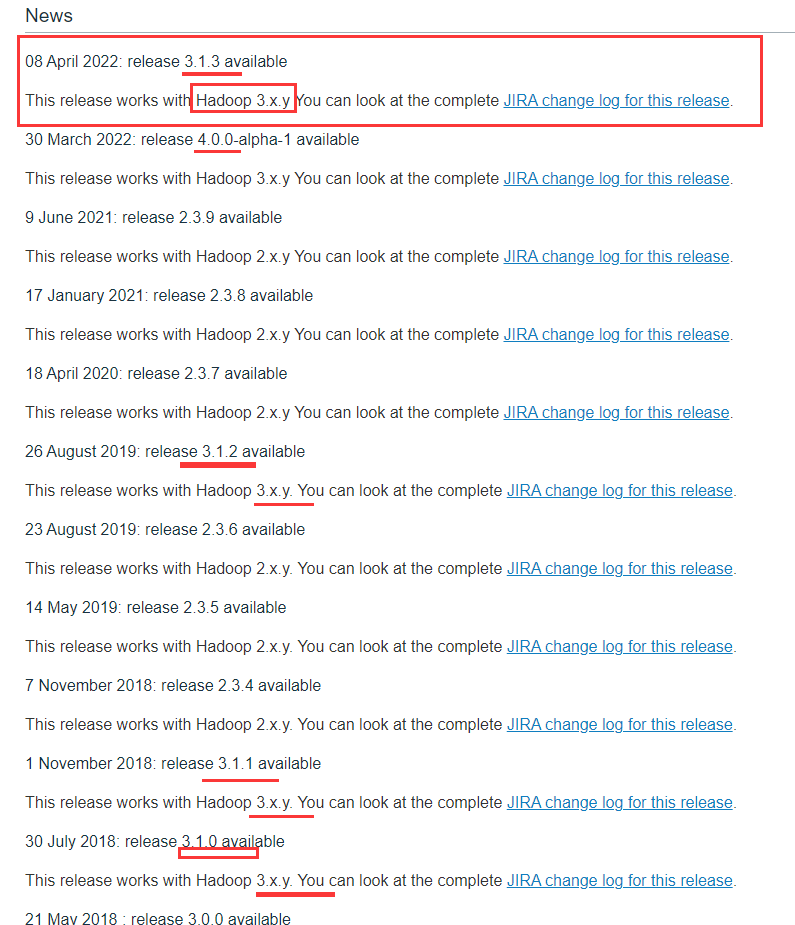
## 5、Hive3.1.3安装部署

下载

apache-hive-3.1.3-bin.tar.gz

官网

<https://hive.apache.org/downloads.html>



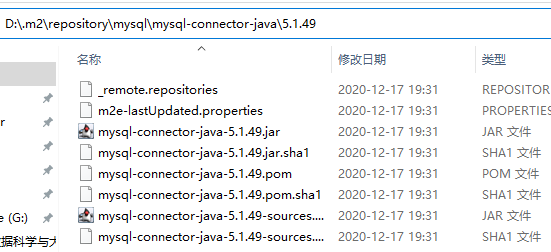
下载MySQL的JDBC驱动

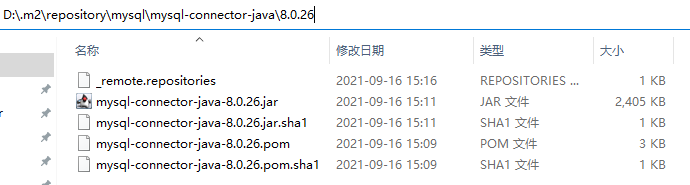
mysql-connector-java-5.1.49.jar

官网

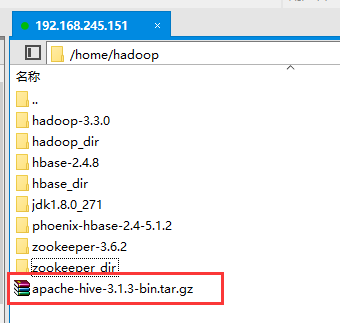
<https://www.mysql.com/cn/products/connector/>

maven本地仓库



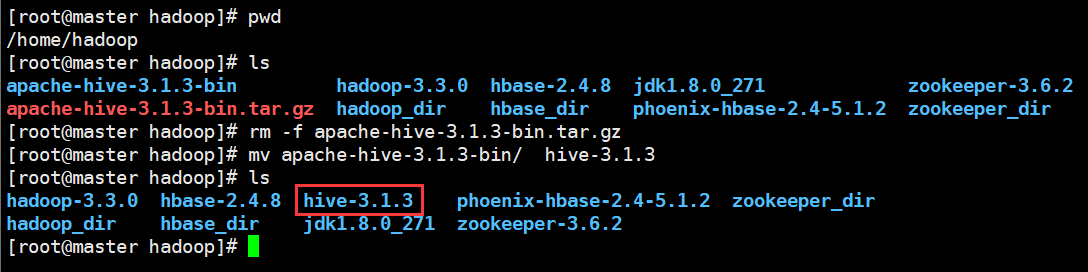


上传hive包



解压并重命名

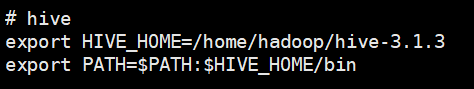




配置环境变量



添加内容如下

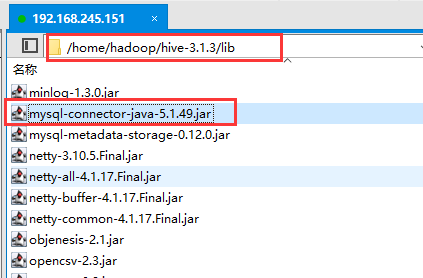


保存并退出

刷新环境变量并验证



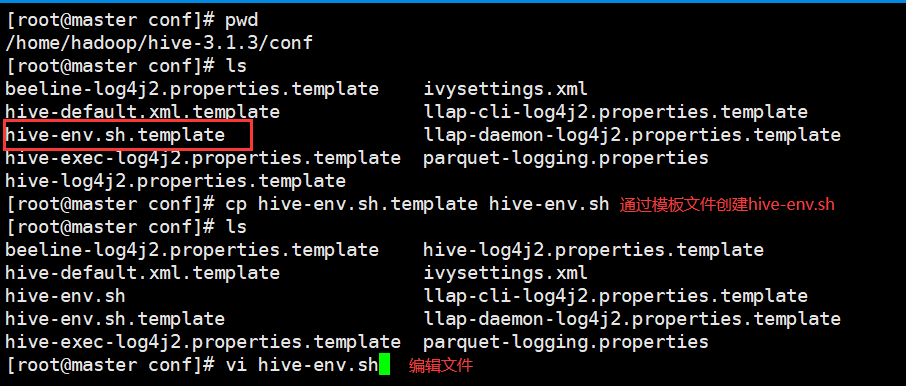
上传JDBC驱动



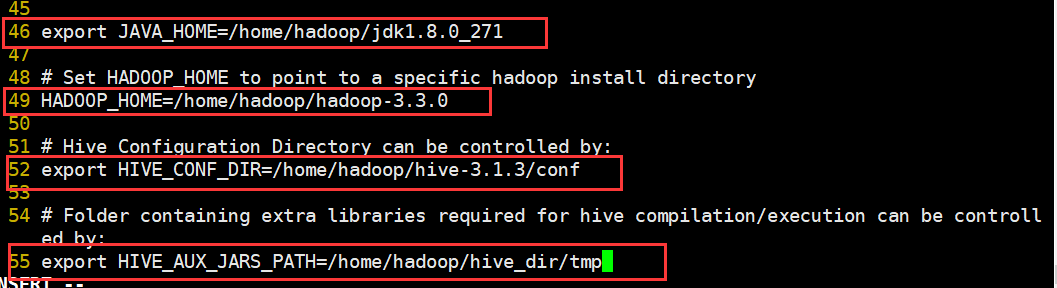
创建工作目录



配置hive-env.sh

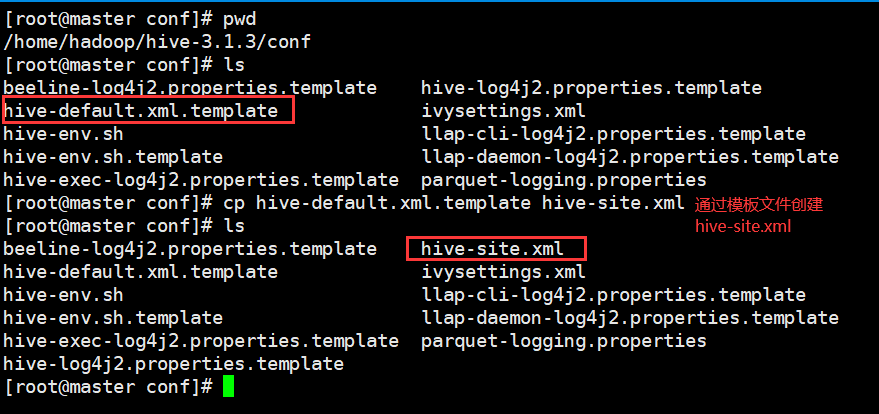


修改内容如下

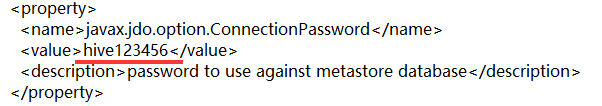


退出并保存

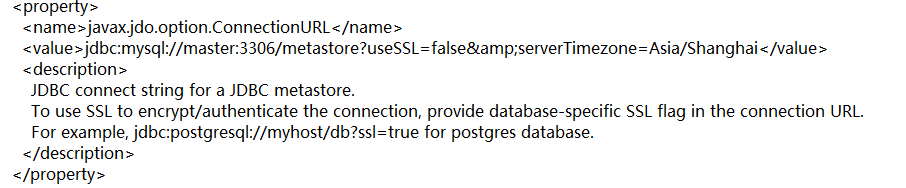
配置hive-site.xml

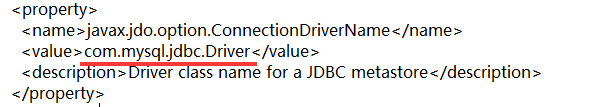


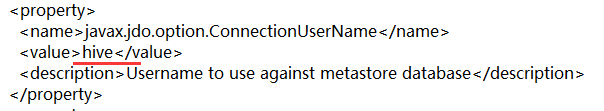
修改如下

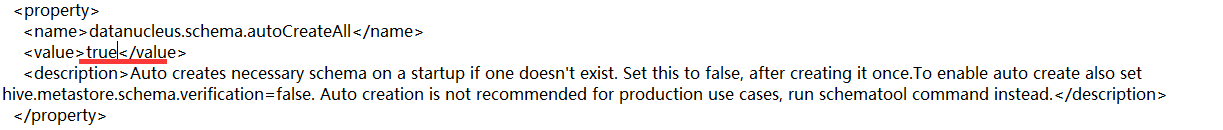
、

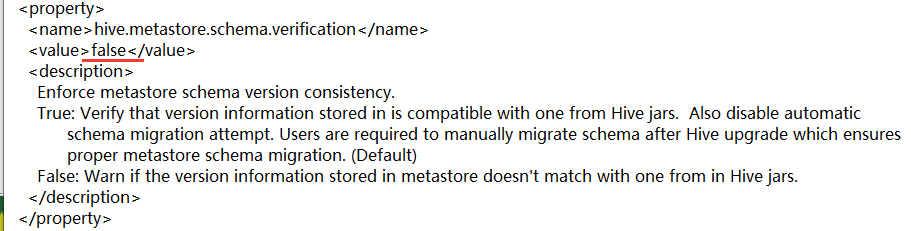
jdbc:mysql://master:3306/metastore?useSSL=false&amp;serverTimezone=Asia/Shanghai

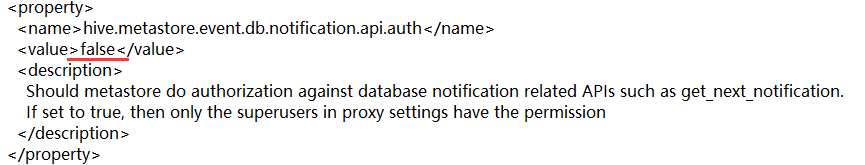


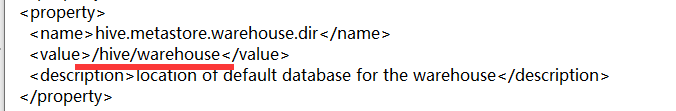


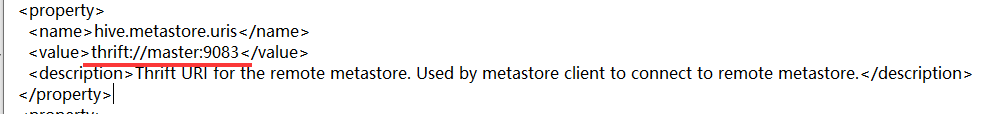


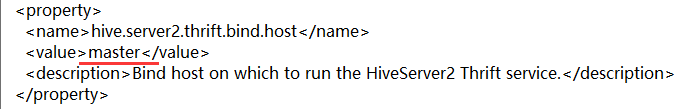


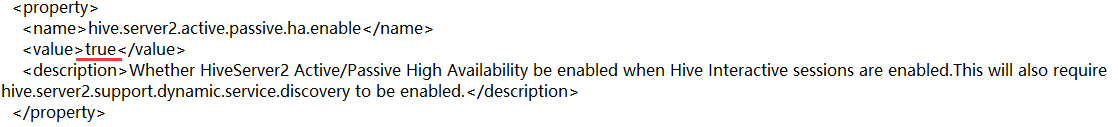












替换目标

${system:java.io.tmpdir}/${system:user.name}

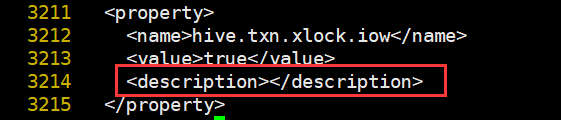
${system:java.io.tmpdir}/${hive.session.id}\_resources

替换为

/home/hadoop/hive\_dir/tmp

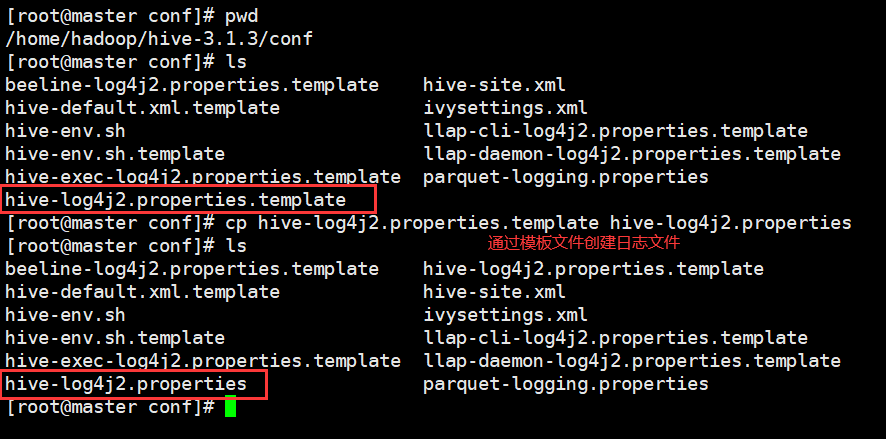
保存并退出

删除hive-site.xml中3215行中的特殊字符



配置节内容清空，保存退出

配置日志

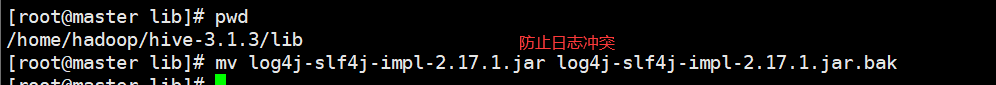


修改如下



保存并退出

包的问题





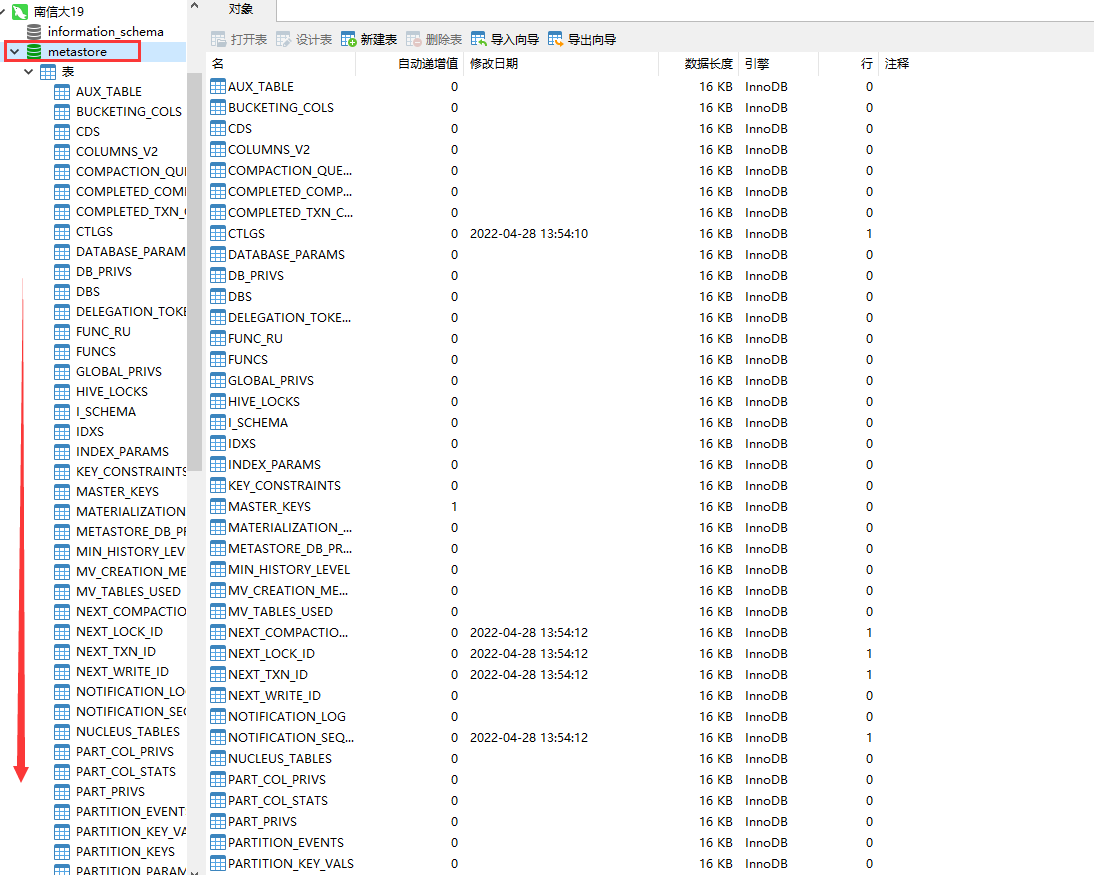
## 6、初始化MySQL数据库，提供元数据服务

初始化MySQL数据库



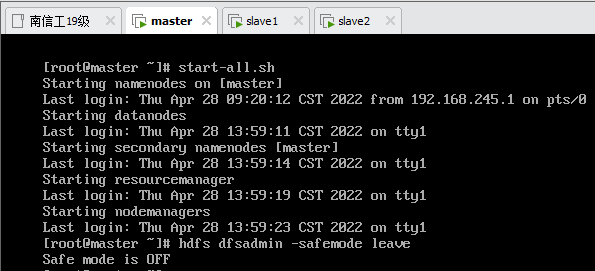
查看初始化后信息





## 7、启动hive

先启动hadoop集群

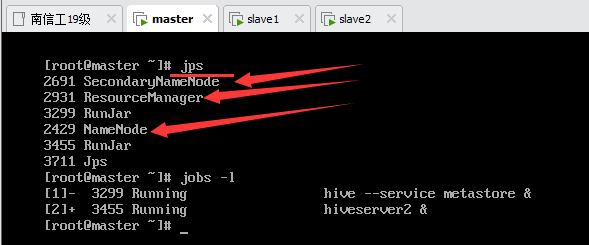


再启动hive



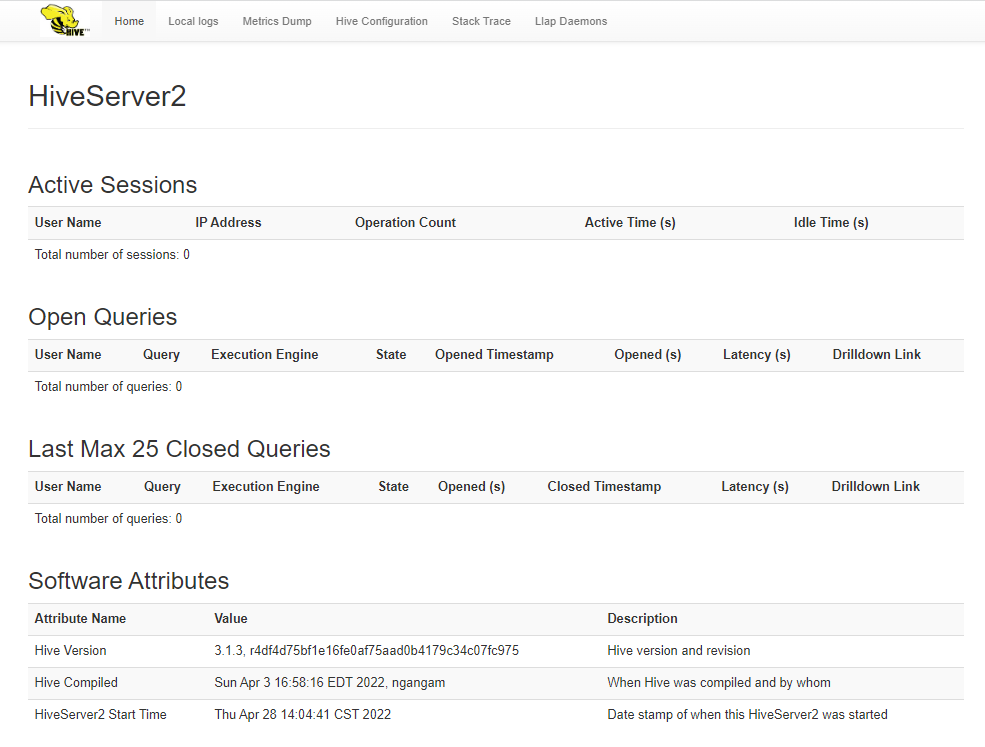
## 8、通过进程与浏览器查看

进程



浏览器查看

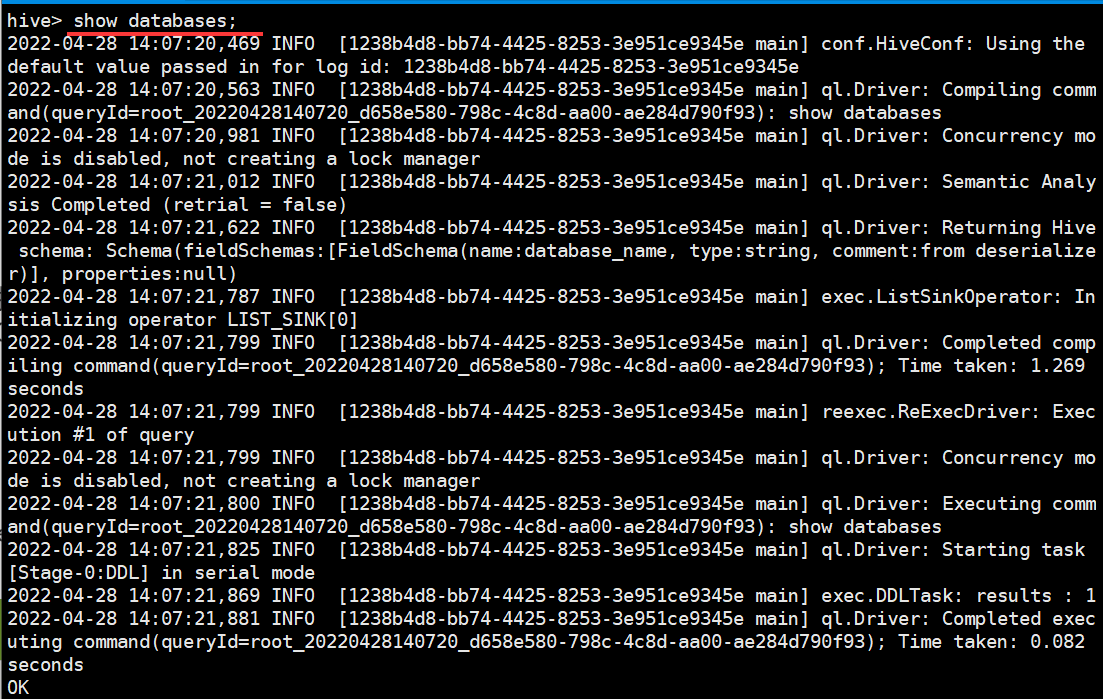
<http://master:10002/>

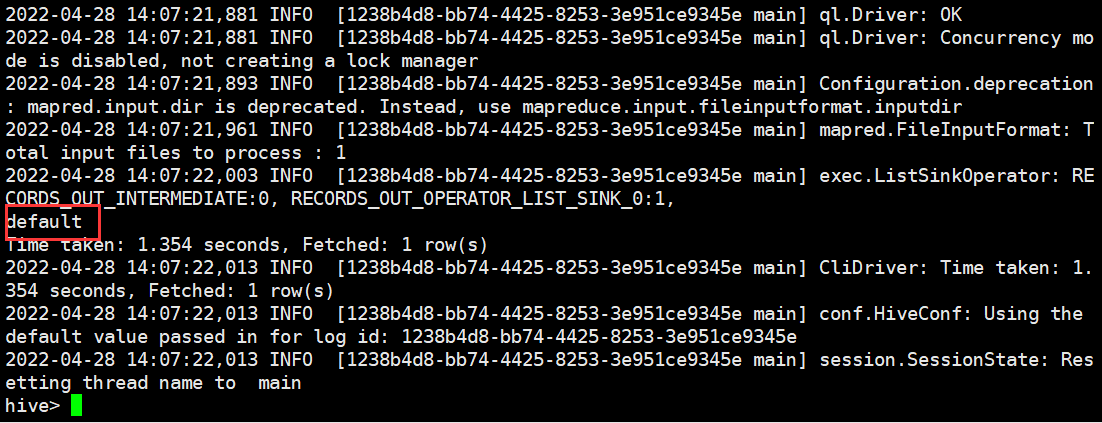


## 9、客户端工具测试

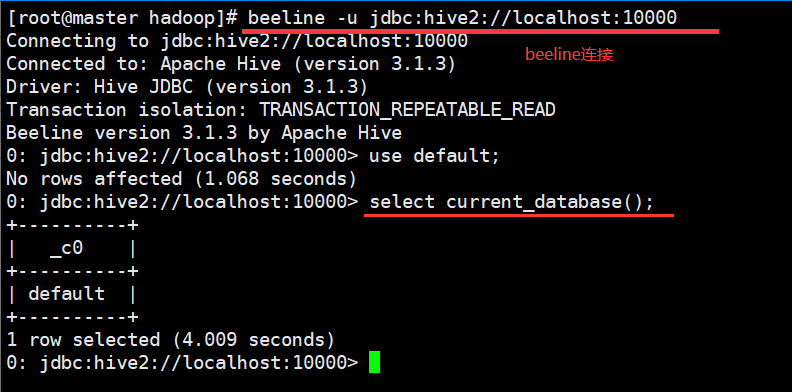
hive shell



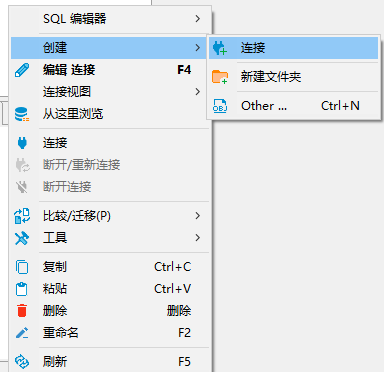


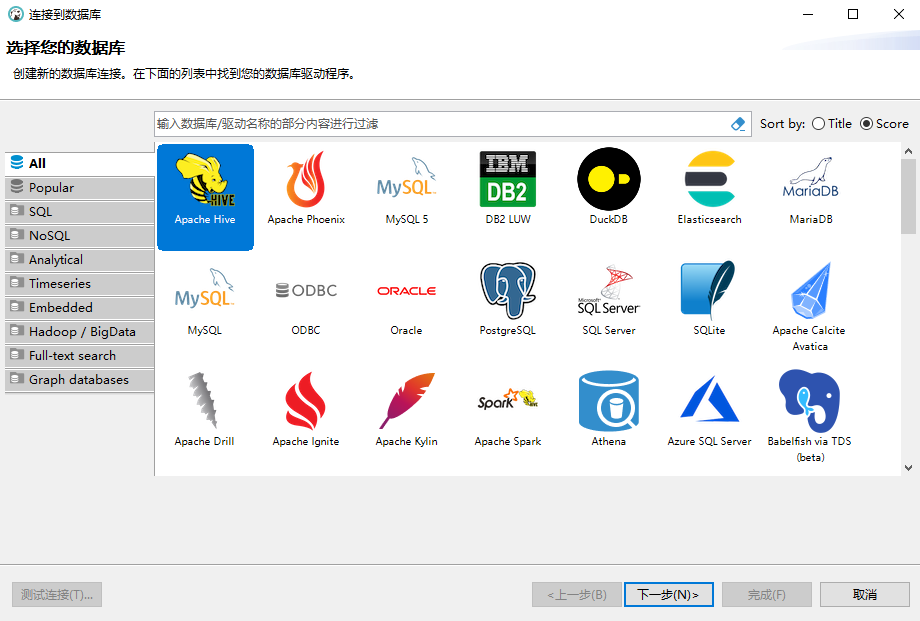


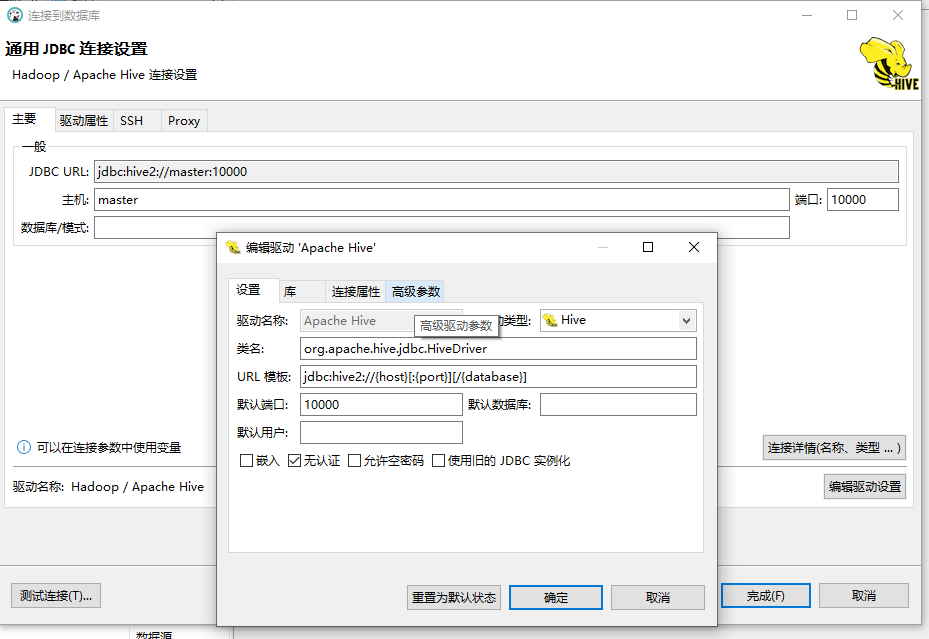
beelin工具

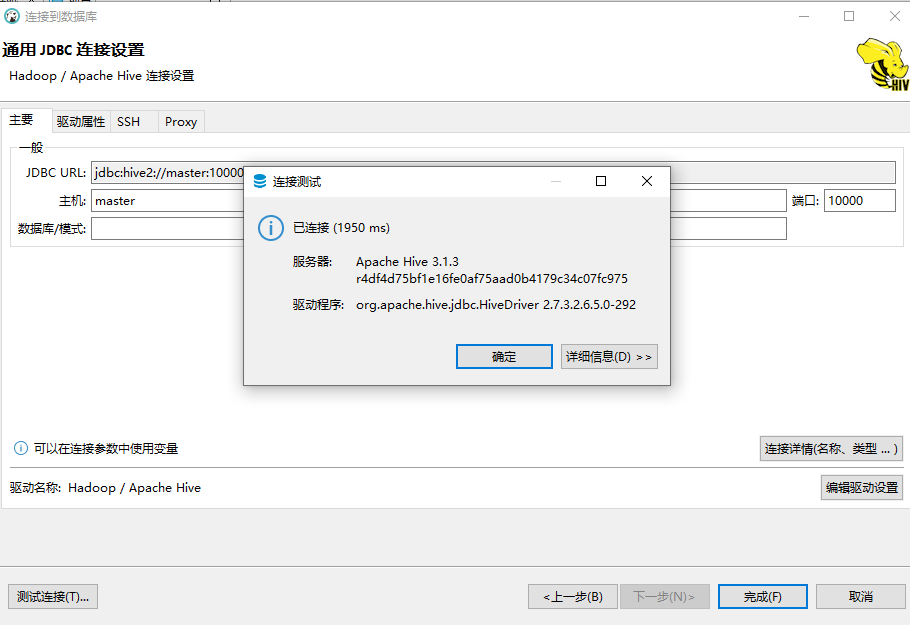


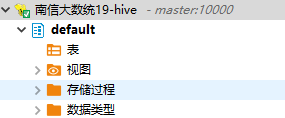
DBeaver工具

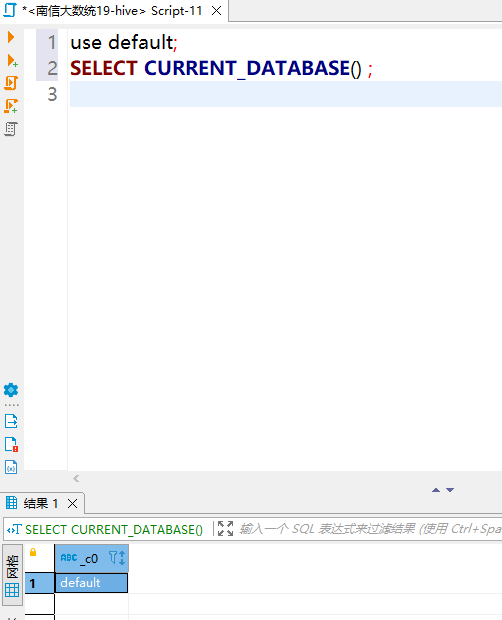




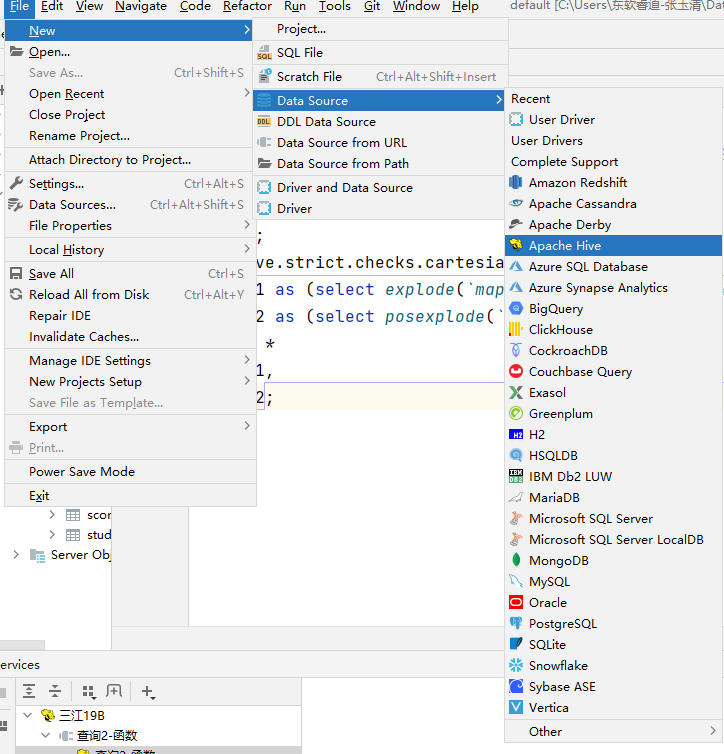


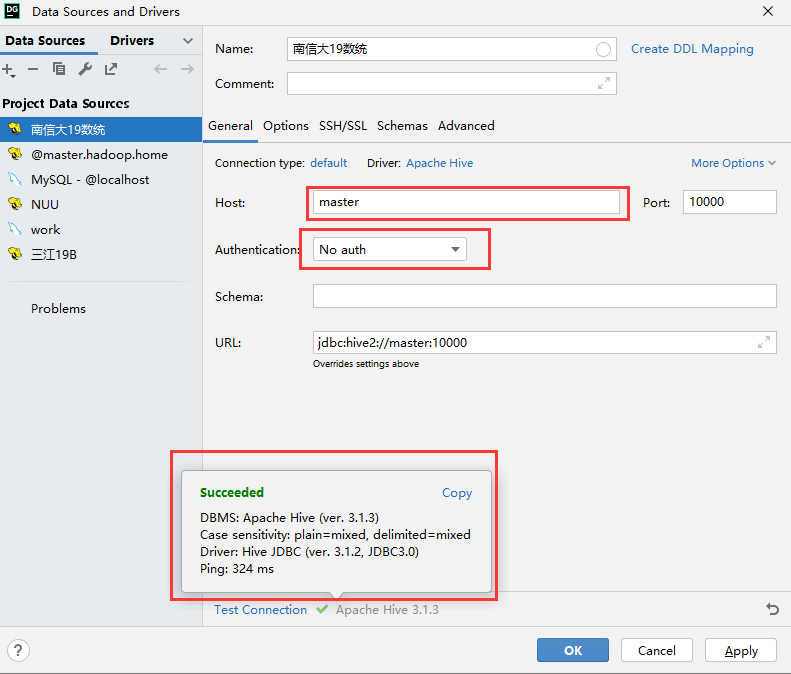


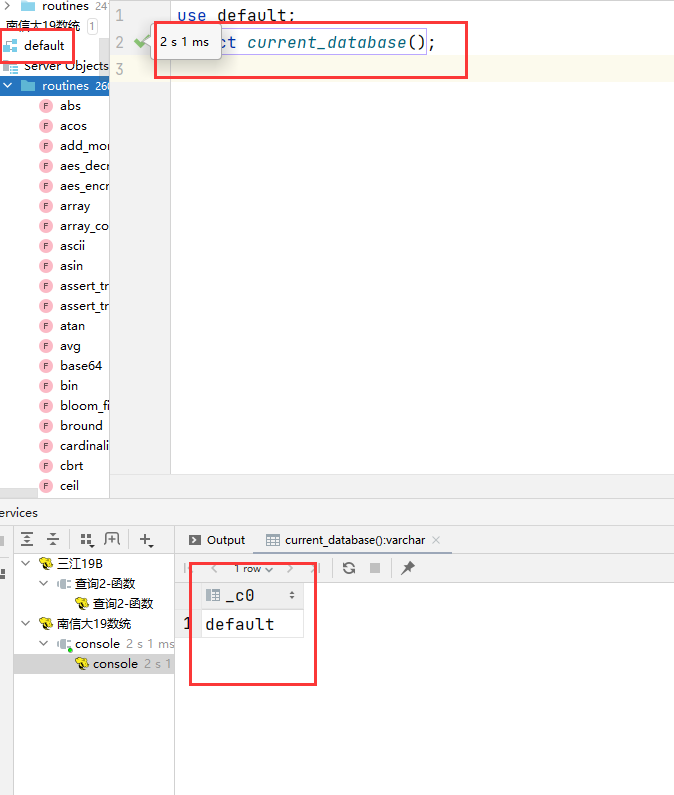




DataGrip工具





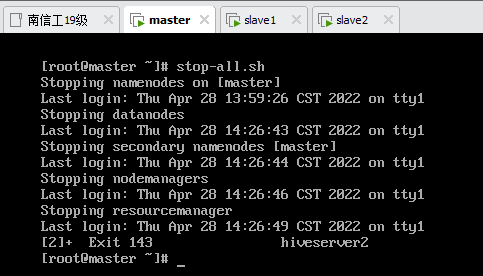


## 10、停止hive

先停止hive



再停止hadoop

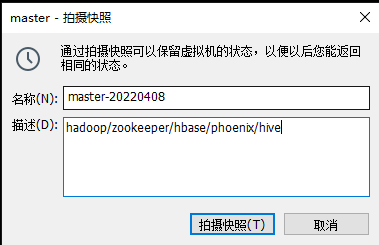


## 11、虚拟机创建快照，方便恢复

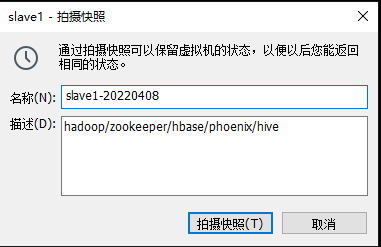
先清理一下

。。。。

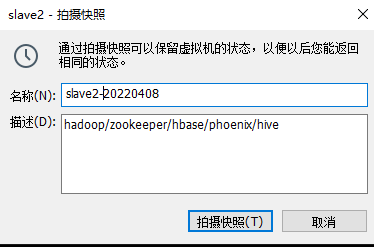
master创建快照



slave1创建快照



slave1创建快照



# 二、Hive概述

## 1、Hive版本—词频统计

回顾以前的版本

hadoop版本—词频统计

hadoop版本—面向对象词频统计

hbase版本-词频统计（hdfs + hbase）

hbase版本-词频统计(hbase)

hadoop缺点

过于底层

不灵活，代码量高，可维护性差，效率差

缺乏高级抽象

完整实现过程，并不是关注实现什么，导致调试更多的无用细节

**特别提醒：授权**



hive版本—词频

1）创建表

***-- 表book存储图书信息*create table book(  
 line string  
);  
*-- 表wordcount存储词频统计结果*create table wordcount(  
 word string,  
 totalcount int  
);**

2）表加载数据

*-- 表book从HDFS中加载数据：剪切动作*load data inpath 'hdfs://master:9000/books/pg20417.txt'  
 overwrite into table book;  
*-- 查看表book数据*select line  
from book;

3）统计词频并写入表wordcount中

***-- 统计词频并写入表wordcount中*select *split*(line, ' ')  
from book;  
  
select *explode*(*split*(line, ' '))  
from book;  
  
select word  
from (select *explode*(*split*(line, ' ')) as word  
 from book) t;  
  
select word, *count*(word)  
from (select *explode*(*split*(line, ' ')) as word  
 from book) t  
group by word  
order by *count*(word) desc;  
  
insert into table wordcount  
select word, *count*(word) as totalcount  
from (select *explode*(*split*(line, ' ')) as word  
 from book) t  
group by word  
order by *count*(word) desc;**

4）查看词频结果

***-- 查看词频结果*select word, totalcount  
from wordcount;**

结论：

掌握SQL的人员也可以做大数据分析，并不需要我们掌握Java等高阶知识

降低企业成本

## 2、业务分析

数据（转换、综合分析）

通过类似SQL简化操作转换步骤，简单说，就是生成中间结果

存储常见的模式，如排序、分组、连接、子查询、窗口、聚合……

关注点聚集至统计分析上

提供重用性

高层工具（将SQL转换为MapReduce任务）

hive

🡪类似SQL的HQL

🡪hive 3.x.y版本，兼容SQL 2016标准

pig🡪PingLatin

## 3、Hive概述

hive：（基于hadoop的）数据仓库

facebook

开发

开源

ETL（提取、转化、加载）

存储

查询

分析

定义类似SQL的查询语言，称为HQL

hivey设计目的：让SQL技能良好分析师也可以进行大数据分析

hive是HQL解析引擎，将HQL语句转换为MapReduce任务

hive是语言翻译器

数据存储使用HDFS

数据计算使用MapReduce🡪支持多个计算引擎mr / tez / spark

mr是默认支持的计算引擎

hive缺点

HQL表达能力有限

🡪SQL 2016：比较强大，可以胜任大部分工作

🡪自定义用户函数，进一为提升hive的功能

复杂计算还需要编写MapReduce任务

hive不是完整数据库

基于HDFS，限制了hive

不支持记录级别的更新或删除操作

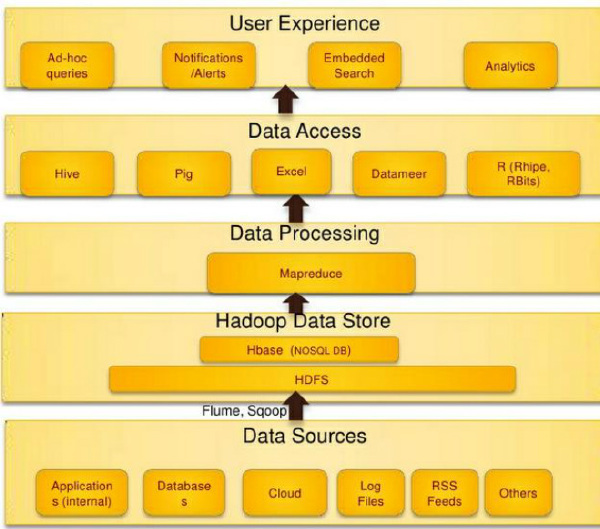
模拟：通过查询生成新结果，再将新结果插入至原表中覆盖

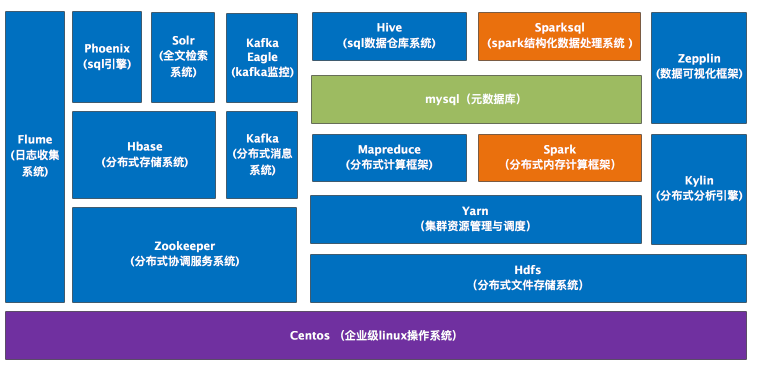
hadoop面向批处理系统

执行效率差，时间长，任务延时比较严重

hive查询延时比较严重，不合适作为及时查询使用

## 4、大数据生态圈中的hive





## 5、hive优点与缺点

优点

HQL：类似于SQL，3.x.y版本兼容SQL2016标准，降低学习成本

JDBC/ODBC接口：开发人员更容易使用，无学习成本

使用HDFS存储，使用MapReduce作为计算引擎，解决了大数据处理的瓶颈

RDBMS海量数据处理存在瓶颈

计算引擎可以切换mr / tez / spark

统一的元数据管理：存储在derby或MySQL等数据库中，且与pig、presto等共享

并行计算：使用集群的所有资源

缺点

HQL表达能力有限

复杂计算需要编写MapReduce任务

查询延时比较严重

离线查询

调优困难，精度较粗

可控性差

推荐拥抱3.0时代

hadoop 3.x.y

hbase 3.x.y

hive 3.x.y

spark 3.x.y

## 6、Hive应用场景

主要场景

用于静态的结构

经常分析

使用场景

日志分析：PV、UV

多维度数据分析

海量数据存储处理

数据挖掘

离线数据分析

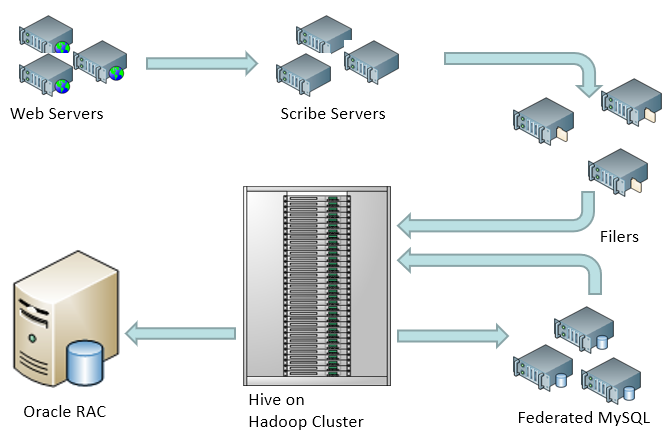
不适合场景

复杂的机器学习算法

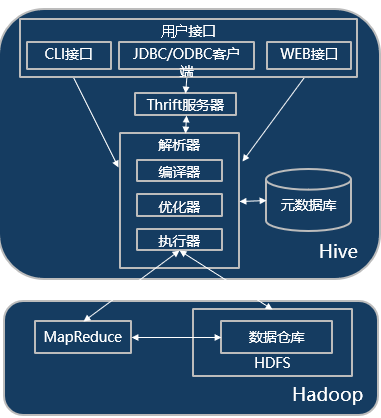
复杂的科学计算

联机交互式实时查询

## 7、Facebook的hive应用



## 8、Hive架构



用户接口

CLI：命令行交互式界面

JDBC / ODBC：标准的JDBC操作

WebUI：从2.x.y取消

元数据数据库

存储hive数据库的基础信息

存储在RDBMS表中

元数据，包括数据库信息、表信息、列、分区……

hive需要启动元数据服务

hive –service metastore &

解析器

类似于JVM

编译器

优化器

执行器

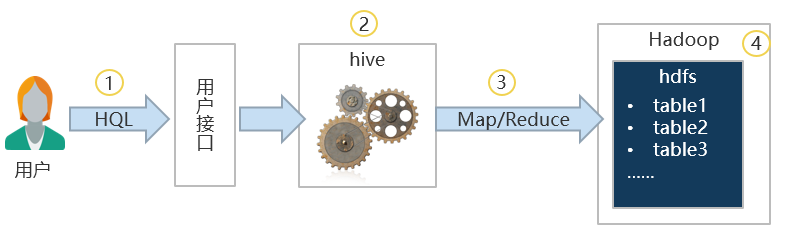
可以参考RDBMS优化，生成查询计划

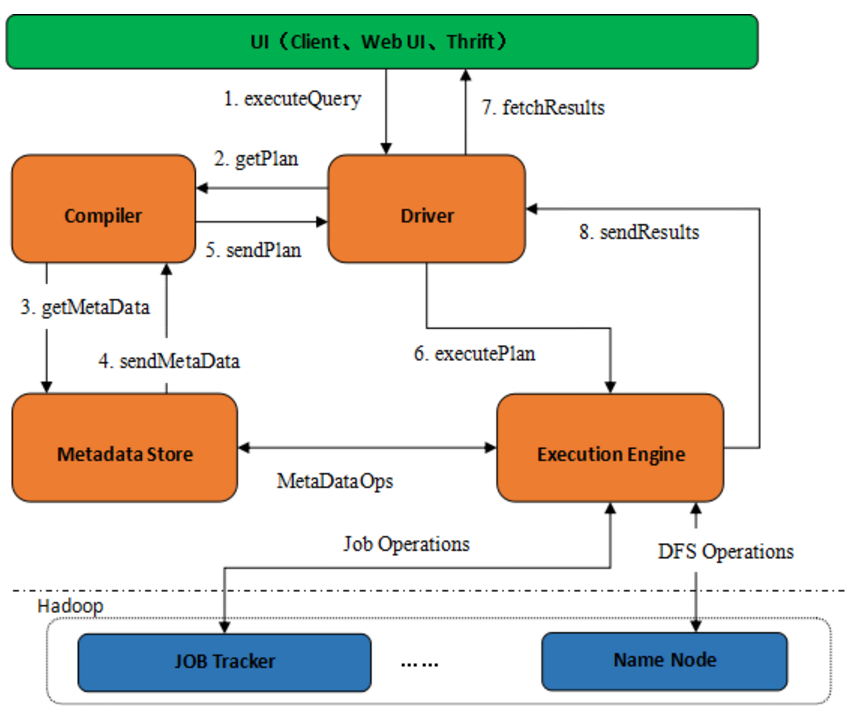
hadoop

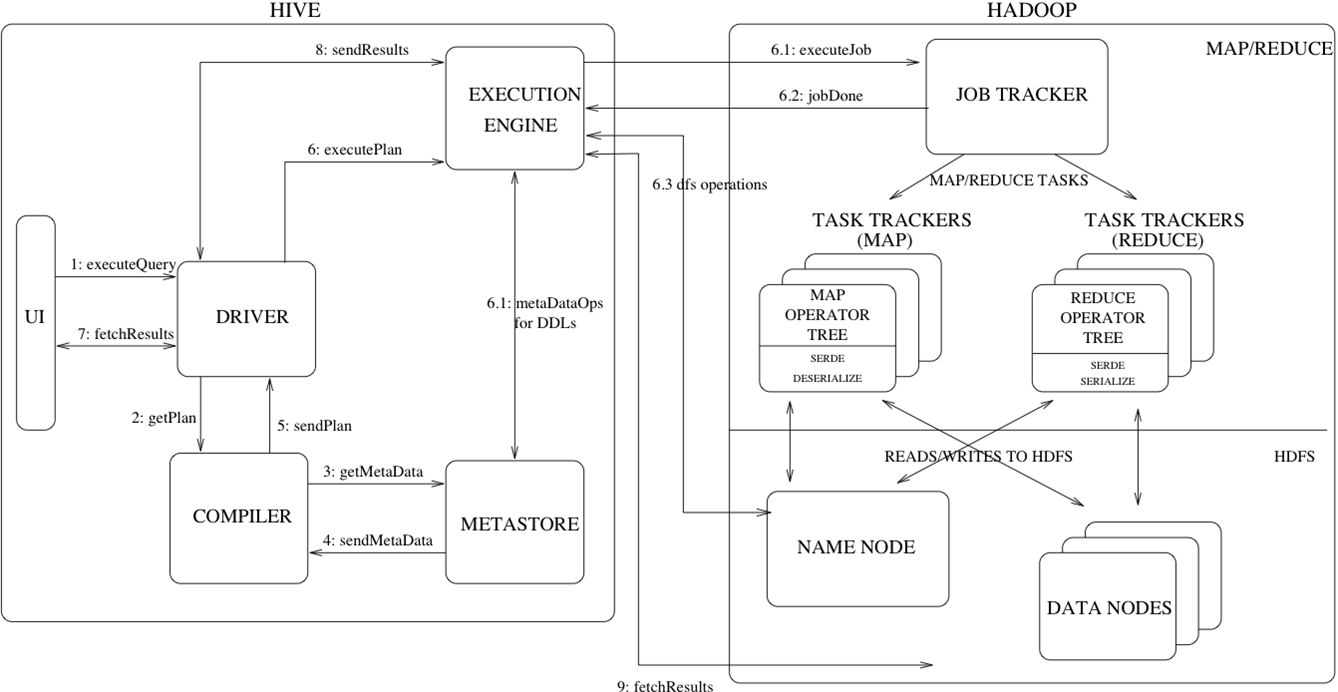
HDFS存储

MapReduce离线并行计算

## 9、Hive运行机制







## 10、hive和hadoop关系

hive构建在hadoop之上

HQL由hive完成（编译器、优化器、执行器）生成查询计划

查询计划被转化为MapReduce任务，交给Hadoop执行

hive的所有数据均存储在HDFS上

hadoop和hive均采用UTF-8编码

## 11、Hive和RDBMS比较



## 12、hive设计特征

支持索引，加快数据查询

hive 3.x.y正式取消索引，优秀到，无需使用索引

不同的文件格式

行式存储

TextFile

SequenceFile

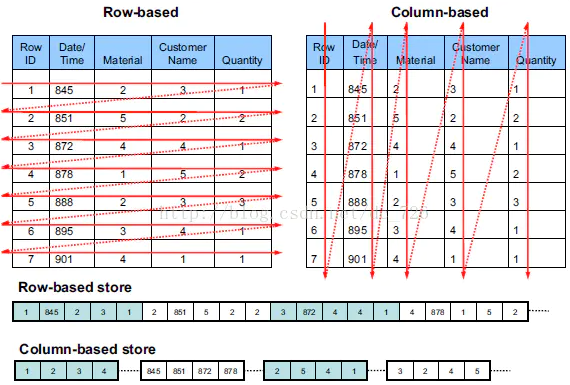
Avro

列式存储

RCFile

ORCFile

Parquet



列式存储优点

数据在磁盘上是按字段顺序连续存储

插入操作和更新操作时效率高

列多存储缺点

从源表中读取一二个字段，也要读取完整的记录

列式存储的优点就是行式存储的缺点，列式存储的表只会返回想要的字段，投影操作比较高效

行式存储的适用场景包括：

1、适合随机的增删改查操作;

2、需要在行中选取所有属性的查询操作;

3、需要频繁插入或更新的操作。

列式存储引擎的适用场景包括：

1、查询过程中，可针对各列的运算并发执行(SMP)，最后在内存中聚合完整记录集，最大可能降低查询响应时间;

2、可在数据列中高效查找数据，无需维护索引(任何列都能作为索引)，查询过程中能够尽量减少无关IO，避免全表扫描;

3、各列独立存储，且数据类型已知，可以针对该列的数据类型、数据量大小等因素动态选择压缩算法，以提高物理存储利用率。

元数据保存在RDBMS中

RDBMS查询速度快

减少查询中执行语义检查时间

直接使用HDFS数据

内置大量用户函数

支持用户扩展函数UDF

HQL

将HQL转换为MapReduce任务，并在hadoop集群上执行

## 13、Hive运行模式

Derby

hive将数据存储在HDFS中

元数据默认存在在hive自带的数据库derby中

Derby不支持多用户并发访问

用于测试

Local

使用MySQL或其它数据库替换Derby来存储元数据

支持多用户并发访问

仅部署一个节点服务器

用于开发模式

Remote

基于Local方式

MySQL数据所在节点提供metastore service服务，其它节点连接该服务获取元数据

支持多用户并发访问

所有节点同时支持多用户并发访问，简单说就是支持N个大数据分析师同时使用

用于生产环境

## 14、hive元数据

元数据metastore

数据库信息

表信息

……

数据量不高，相对而言，查询速度飞快

存储在RDBMS中

作用

启动元数据服务

客户端连接元数据服务，元数据服务连接RDBMS来存取元数据

## 15、hive原语

hive数据模型

表组成

列组成

数据类型

简单数据类型

复合数据类型

表

表 / 内部表

数据存储在hive中（最终是存储在HDFS）

删除表时，hive与HDFS中数据同时删除

外部表

数据存储在HDFS中，hive中保持一个引用

删除表时，删除hive中的引用，HDFS中数据保留

分区表

数据按照一定的规则进行划分分区

所谓的分区就是一个目录

同一个分区的数据放在同一个目录内

粗料度的划分

提高查询效率

桶表

在一定范围内的数据按照hash的方式进行划分

细料度划分

## 16、hive数据存储

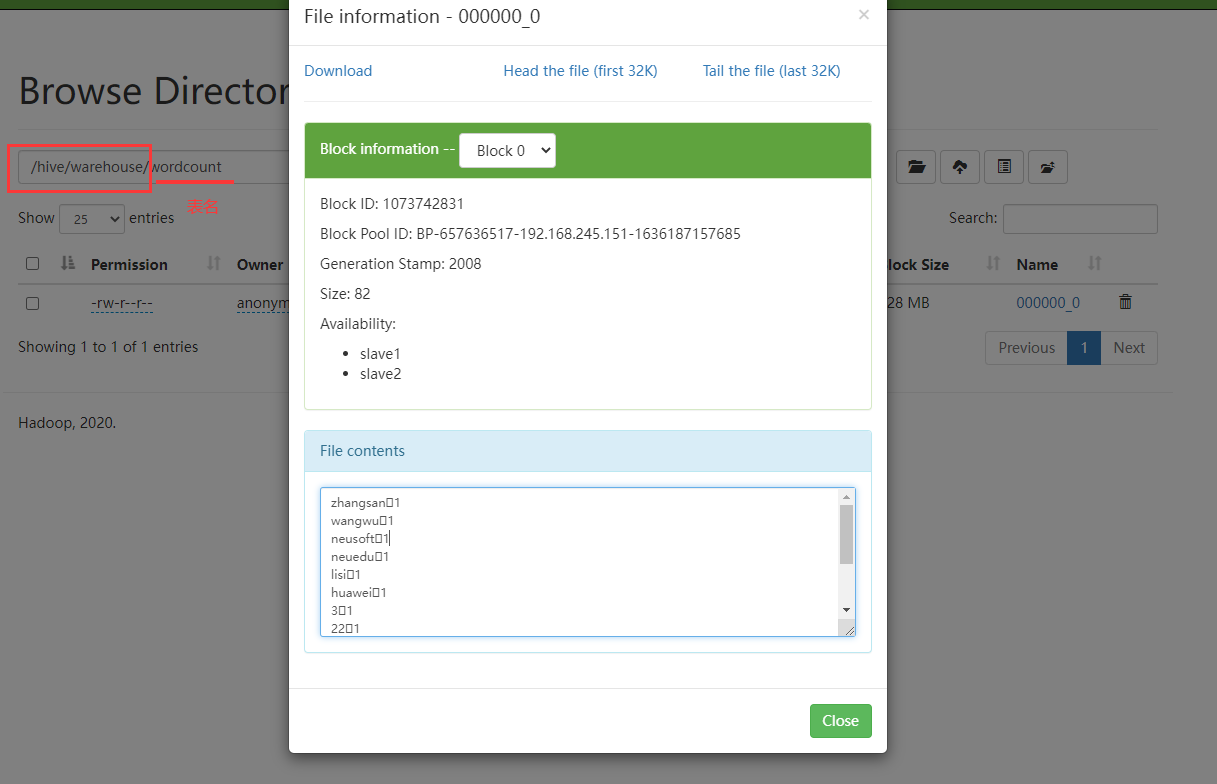
存储在HDFS

hive-site.xml

hive.metastore.warehouse.dir

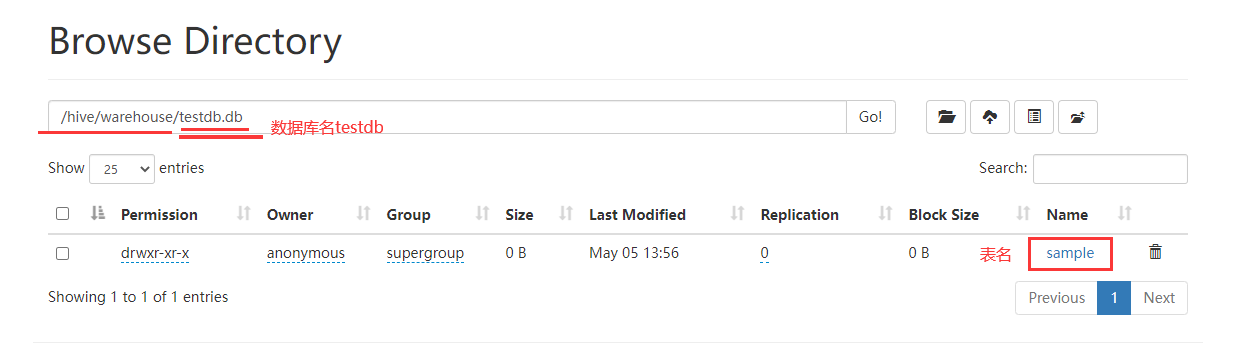
默认数据库default

/hive/warehouse/表名



指定数据库testdb

/hive/warehouse/testdb.db/表名



## 17、Hive访问

Hive CLI（Hive Command Line Interface）

**hive 3.x.y不再推荐**

hive cli交互式操作



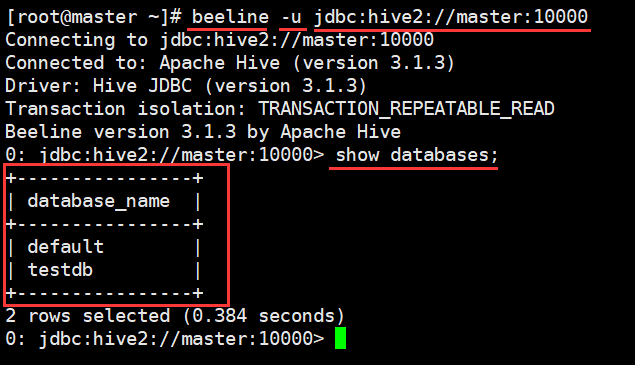




Hive Server

beeline交互式操作

**hive 3.x.y隆重推荐**



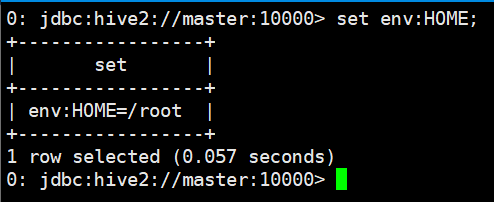
可视化工具

DBeaver

DataGrip

其它产品

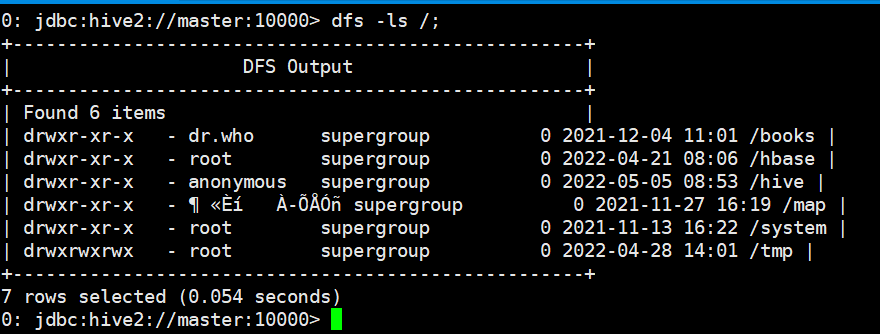
显示变量值



显示所有变量



hdfs操作



# 三、Hive数据定义语言DDL

## 官网参考链接

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual>

## 1、HQL分类

DDL

数据定义语言

create/alter/drop……

DML

数据操纵语言

select/insert/load/update/delete

hive3.x.y兼容SQL2016标准

支持事务ACID，但是不支持事务操纵，如开始事务、提交事务、回滚事务等

update/delete仅在事务表的情况下支持

## 2、数据库操作

数据库

hive中本质上是一个命名空间

HDFS中本质上是一个目录

***-- 显示所有数据库列表*show databases ;  
*-- 创建数据库testdb*create database if not exists testdb;  
*-- 切换数据库*use testdb;  
*-- 仅删除数据库，若存在表时，则异常*drop database if exists testdb ;  
*-- 删除数据库，同时删除数据库中的表*drop database if exists testdb cascade ;**

## 3、创建表

语法

CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db\_name.]table\_name    -- (Note: TEMPORARY available in Hive 0.14.0 and later)

  [(col\_name data\_type [column\_constraint\_specification] [COMMENT col\_comment], ... [constraint\_specification])]

  [COMMENT table\_comment]

  [PARTITIONED BY (col\_name data\_type [COMMENT col\_comment], ...)]

  [CLUSTERED BY (col\_name, col\_name, ...) [SORTED BY (col\_name [ASC|DESC], ...)] INTO num\_buckets BUCKETS]

  [SKEWED BY (col\_name, col\_name, ...)                  -- (Note: Available in Hive 0.10.0 and later)]

     ON ((col\_value, col\_value, ...), (col\_value, col\_value, ...), ...)

     [STORED AS DIRECTORIES]

  [

   [ROW FORMAT row\_format]

   [STORED AS file\_format]

     | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)]  -- (Note: Available in Hive 0.6.0 and later)

  ]

  [LOCATION hdfs\_path]

  [TBLPROPERTIES (property\_name=property\_value, ...)]   -- (Note: Available in Hive 0.6.0 and later)

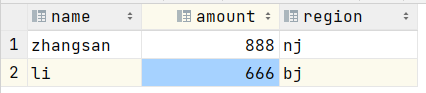
  [AS select\_statement];   -- (Note: Available in Hive 0.5.0 and later; not supported for external tables)

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

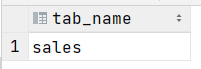
  LIKE existing\_table\_or\_view\_name

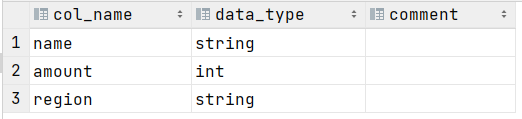
  [LOCATION hdfs\_path];

***-- 切换数据库*use testdb;  
*-- 创建表*create table if not exists sales  
(  
 name string,  
 amount integer,  
 region string  
)  
 row format delimited fields terminated by ',';  
*-- 插入数据*insert into table sales  
values ('zhangsan', 888, 'nj');  
insert into table sales  
values ('li', 666, 'bj');  
*-- 查看数据*select name, amount, region  
from sales;**



***-- 显示所有的数据表*show tables ;**

 ***-- 查看表结构*desc sales;  
describe sales;**

 ***-- 查看表的扩展信息*desc extended sales;**



## 4、数据类型

基本数据类型

整型🡪tinyint / smallint / int / integer / bigint

浮点🡪float / double

十进制🡪decimal / numeric3.0.0

二进制🡪binary

字符串🡪char / varchar / string

布尔🡪boolean

日期与时间🡪date / timestamp / interval

复合数据类型（通过基本数据类型组合而成的新类型）

数组🡪array<数据类型>

键值对🡪map<键类型,值类型>

结构🡪struct<属性1:值类型,属性2:值类型……>

联合🡪uniontype<类型1,类型2……>

100Y / 100S / 100L

100.5F

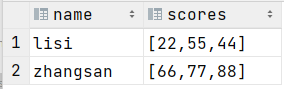
true / false

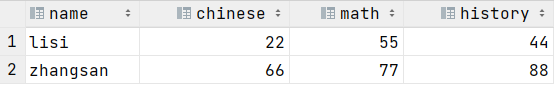
**数组array**

相同数据类型

通过下标来访问

***-- 切换数据库*use testdb;  
*-- 创建表：包含数组array类型*create table if not exists score1  
(  
 name string,  
 scores array<int>  
)  
 row format delimited fields terminated by ','  
 collection items terminated by ':'  
 stored as textfile;  
*-- 插入数据*insert into score1  
select 'zhangsan', *`array`*(66, 77, 88)  
union  
select 'lisi', *`array`*(22, 55, 44);  
*-- 查看数据*select name, scores  
from score1;**

 ***-- 查看详细成绩信息*select name, scores[0] as chinese, scores[1] as math, scores[2] as history  
from score1;**

 ***-- 查看历史及格的信息*select name, scores[0] as chinese, scores[1] as math, scores[2] as history  
from score1  
where scores[2] >= 60;**

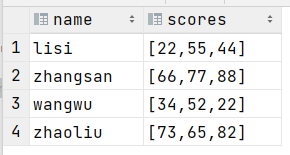


***-- 加载数据：模拟大数据场景***

***-- 存储格式与表存储格式相同：列与列之间使用逗号分隔，数组项之间使得冒号分隔***



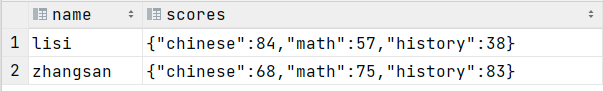
**load data local inpath '/root/score1.txt'  
 into table score1;  
*-- 查看数据*select name, scores  
from score1;**



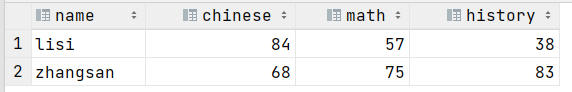
**键值对map**

包含key 🡪 value，通过key来访问value

***-- 切换数据库*use testdb;  
*-- 创建表：包含map类型*create table if not exists score2  
(  
 name string,  
 scores map<string,int>  
)  
 row format delimited fields terminated by ','  
 collection items terminated by '\_'  
 map keys terminated by ':'  
 stored as textfile;  
*-- 插入数据*insert into score2  
select 'zhangsan', *`map`*('chinese', 68, 'math', 75, 'history', 83)  
union  
select 'lisi', *`map`*('chinese', 84, 'math', 57, 'history', 38);  
*-- 查看数据*select name, scores  
from score2;**



***-- 查看详细成绩信息*select name,  
 scores['chinese'] as chinese,  
 scores['math'] as math,  
 scores['history'] as history  
from score2;**



***-- 查看历史及格的详细信息*select name,  
 scores['chinese'] as chinese,  
 scores['math'] as math,  
 scores['history'] as history  
from score2  
where scores['history'] >= 60;**

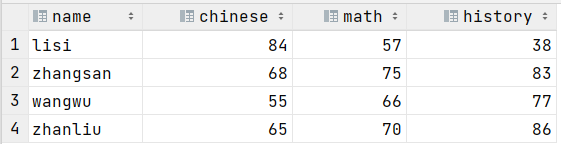


***-- 加载数据：模拟大数据场景  
-- 存储格式与表存储格式相同：列与列之间使用逗号分隔，项与项之间使得下划线分隔,key与value之前使用冒号分隔***





**load data local inpath '/root/score2.txt'  
 into table score2;  
*-- 查看数据*select name,  
 scores['chinese'] as chinese,  
 scores['math'] as math,  
 scores['history'] as history  
from score2;**

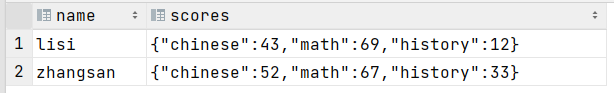


**结构struct**

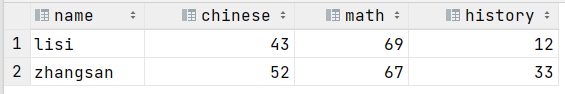
整体概念（与类相似），包含不同数据类型的元素

通过点语法访问所需要数据

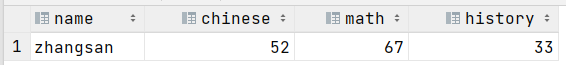
***-- 切换数据库*use testdb;  
*-- 创建表：包含struct类型*create table if not exists score3  
(  
 name string,  
 scores struct<chinese:int,math:int,history:int>  
)  
 row format delimited fields terminated by ','  
 collection items terminated by ':'  
 stored as textfile;  
*-- 插入数据*insert into score3  
select 'zhangsan', *named\_struct*('chinese', 52, 'math', 67, 'history', 33)  
union  
select 'lisi', *named\_struct*('chinese', 43, 'math', 69, 'history', 12);  
*-- 查看数据：数据表达形式与map非常相似*select name, scores  
from score3;**



***-- 显示详细的成绩信息*select name, scores.chinese, scores.math, scores.history  
from score3;**



***-- 显示历史成绩大于20分的信息*select name, scores.chinese, scores.math, scores.history  
from score3  
where scores.history > 20;**

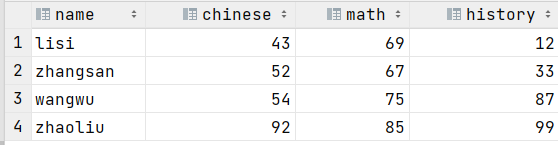


***-- 加载数据：模拟大数据场景  
-- 存储格式与表存储格式相同：列与列之间使用逗号分隔，项与项之间使得冒号分隔***



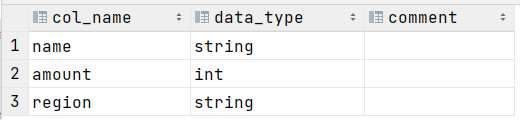


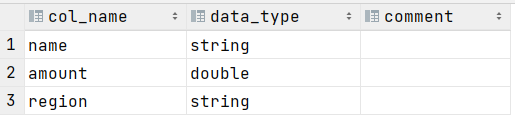
**load data local inpath '/root/score3.txt'  
 into table score3;  
*-- 显示详细的成绩信息*select name, scores.chinese, scores.math, scores.history  
from score3;**



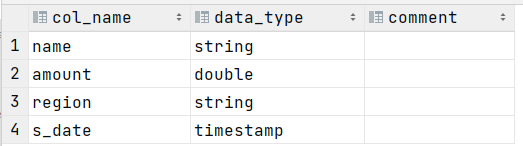
## 5、表的维护

***-- 切换数据库*use testdb;  
*-- 查看表结构*desc sales;**

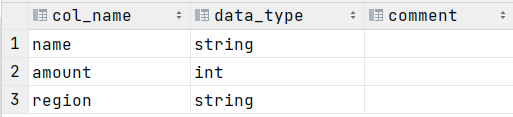
*-- --修改表*alter table sales  
 change amount amount double;  
desc sales;



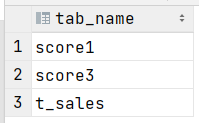
***-- 增加列*alter table sales  
 add columns (s\_date timestamp);  
desc sales;**



***-- 删除列：列的类型与原表中需要保持一致***alter table sales  
 replace columns (  
 name string,  
 amount double,  
 region string  
 );  
desc sales;

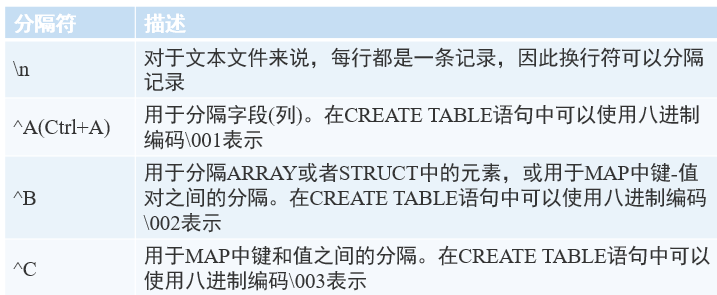


***-- 重命名表*alter table sales  
 rename to t\_sales;  
show tables ;**



***-- 删除表：删除表及其数据，若是外部表则删除表保留数据*drop table t\_sales;**

## 6、控制字符



## 7、数据加载

语法

LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)]

LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename [PARTITION (partcol1=val1, partcol2=val2 ...)] [INPUTFORMAT 'inputformat' SERDE 'serde'] (3.0 or later)

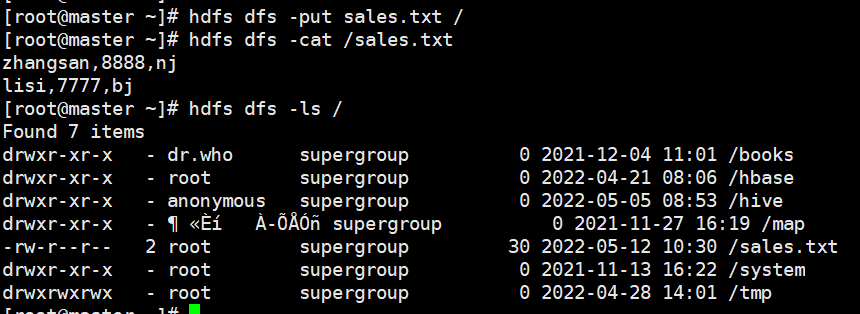
local是本地文件，将文件上传至hive存储目录



无local是代表hdfs文件，将hdfs文件移到至hive存储目录

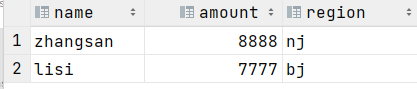


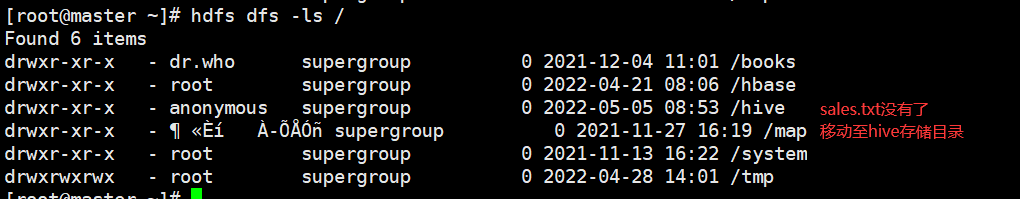


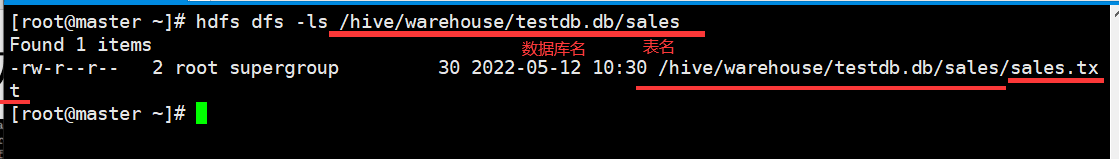


***-- 切换数据库*use testdb;  
*-- 创建表*create table if not exists sales  
(  
 name string,  
 amount integer,  
 region string  
)  
 row format delimited fields terminated by ',';  
*-- 加载数据*load data inpath 'hdfs://master:9000/sales.txt'  
 overwrite into table sales;**

*-- 查看数据*select name, amount, region  
from sales;







overwrite覆盖，简单说，就是相当于是清空原表中数据，再加载数据

## 8、约束

约束

primary key主键约束

unique惟一约束

not null非空约束

default默认约束

check检查约束

foreign key references外键约束

列后语法

column\_constraint\_specification:

  : [ PRIMARY KEY|UNIQUE|NOT NULL|DEFAULT [default\_value]|CHECK  [check\_expression] ENABLE|DISABLE NOVALIDATE RELY/NORELY ]

所有列后语法

constraint\_specification:

  : [, PRIMARY KEY (col\_name, ...) DISABLE NOVALIDATE RELY/NORELY ]

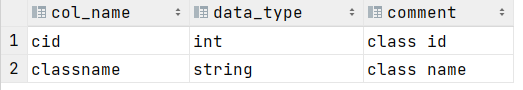
    [, PRIMARY KEY (col\_name, ...) DISABLE NOVALIDATE RELY/NORELY ]

    [, CONSTRAINT constraint\_name FOREIGN KEY (col\_name, ...) REFERENCES table\_name(col\_name, ...) DISABLE NOVALIDATE

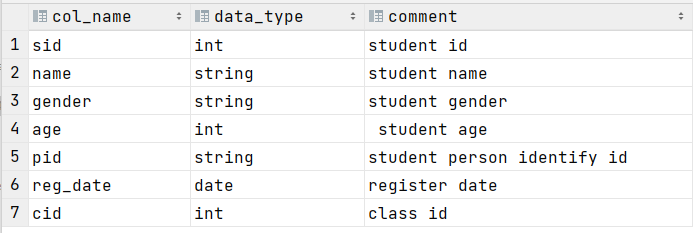
    [, CONSTRAINT constraint\_name UNIQUE (col\_name, ...) DISABLE NOVALIDATE RELY/NORELY ]

    [, CONSTRAINT constraint\_name CHECK [check\_expression] ENABLE|DISABLE NOVALIDATE RELY/NORELY ]

***-- 切换数据库*use testdb;  
*-- 创建班级表*create table if not exists class  
(  
 cid int not null comment 'class id',  
 classname string not null comment 'class name',  
 primary key (cid) disable novalidate  
);  
*-- 查询表结构*desc class;**



***-- 创建学生表*create table if not exists student  
(  
 sid int not null comment 'student id',  
 name string not null comment 'student name',  
 gender string comment 'student gender',  
 age int comment ' student age',  
 pid string unique disable novalidate comment 'student person identify id',  
 reg\_date date default *`current\_date`*() disable novalidate comment 'register date',  
 cid int comment 'class id',  
 primary key (sid) disable novalidate,  
 constraint fk foreign key (cid) references class (cid) disable novalidate,  
 constraint ck check (age > 18) disable novalidate norely  
);  
*-- 查看表结构*desc student;**



## 9、事务表

事务是由一系列逻辑工作单元组成的一个执行单元，要么全部执行，要么全部失败。

转帐（是一个事务）

A-100

B+100

不可分割

ACID四个属性

Atomacity原子性

不可分割

要么全部执行，要么全部失败

Consistency一致性

事务完成后，数据保持一致状态，与成功与否无关

转帐：

A(1000),B(5000)，银行总额6000元

A转账100给B，不管转账是否成功，银行总额依然是6000元

Isolation隔离性

并发操作

一个事务执行时，对另一个事务的影响，称之为事务隔离级别

已提交读、未提交读、可重复读、可串行读

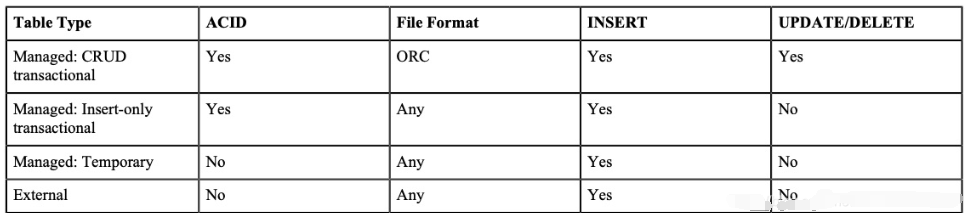
脏读、幻像读……

引出新的话题：锁机制（表级锁、行级锁、排它锁、共享锁……）

Durability持久性

事务成功后，数据写入磁盘，永久保存

事务失败后，数据会回滚至操作前的初始状态，保证数据一致，相当于数据无变化



hive-site.xml添加

<property>

<name>hive.support.concurrency</name>

<value>true</value>

</property>

<property>

<name>hive.enforce.bucketing</name>

<value>true</value>

</property>

<property>

<name>hive.exec.dynamic.partition.mode</name>

<value>nonstrict</value>

</property>

<property>

<name>hive.txn.manager</name>

<value>org.apache.hadoop.hive.ql.lockmgr.DbTxnManager</value>

</property>

<property>

<name>hive.compactor.initiator.on</name>

<value>true</value>

</property>

<property>

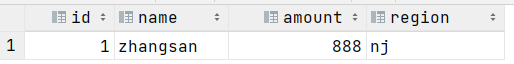
<name>hive.compactor.worker.threads</name>

<value>1</value>

</property>

必须是ORC格式

***-- 创建事务表*create table trans\_sales  
(  
 id int not null,  
 name string,  
 amount integer,  
 region string  
)  
 stored as orc  
 tblproperties ("transactional" = "true");  
*-- 插入数据*insert into table trans\_sales  
values (1, 'zhangsan', 888, 'nj');  
*-- 查看数据*select id, name, amount, region  
from trans\_sales;**



***-- 更新表*update trans\_sales  
set amount=1888  
where id = 1;  
*-- 查看数据*select id, name, amount, region  
from trans\_sales;**



不支持以下语句（也就是开始事务、提交事务、回滚事务）

begin trasn、commit trans 、rollback trans

与事务相关的操作命令

***-- 事务相关命令*show transactions ;  
show compactions ;  
show locks;  
abort transactions 1001;**

## 10、外部表

外部表

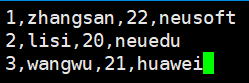
原始数据不存储在hive目录中

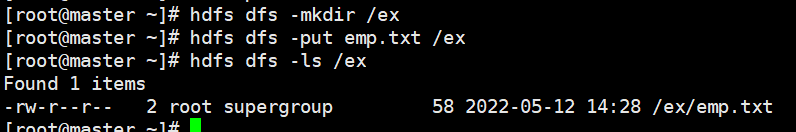
hive中仅保留一个原始数据的一个引用

删除外部表时，原始数据不会被删除

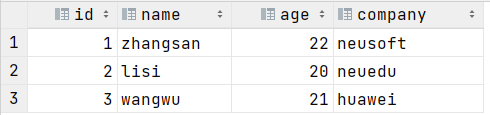
***-- 切换数据库*use testdb;  
*-- 准备数据***



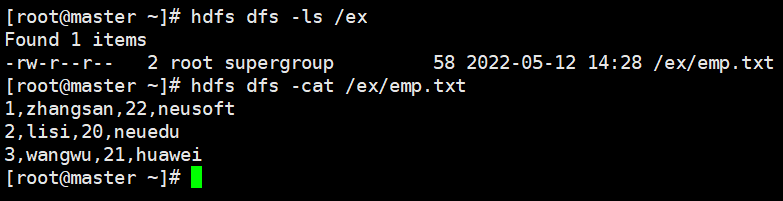




***-- 创建外部表*create external table emp  
(  
 id int,  
 name string,  
 age int,  
 company string  
) row format delimited fields terminated by ','  
 location '/ex';  
*-- 查看数据*select id, name, age, company  
from emp;**



***-- 删除外部表*drop table if exists emp;**



## 11、分区表

hive 3.x.y

通过索引机制，提升查询效率🡪取消索引(底层设计原因，提升不明显)

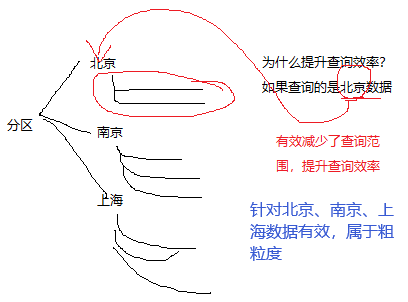
分区

根据分区列的值对表进行粗略划分的机制🡪粗粒度

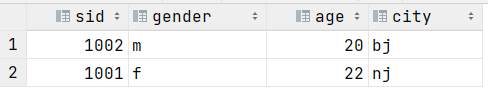
hive中每个分区对应分区列的一个索引

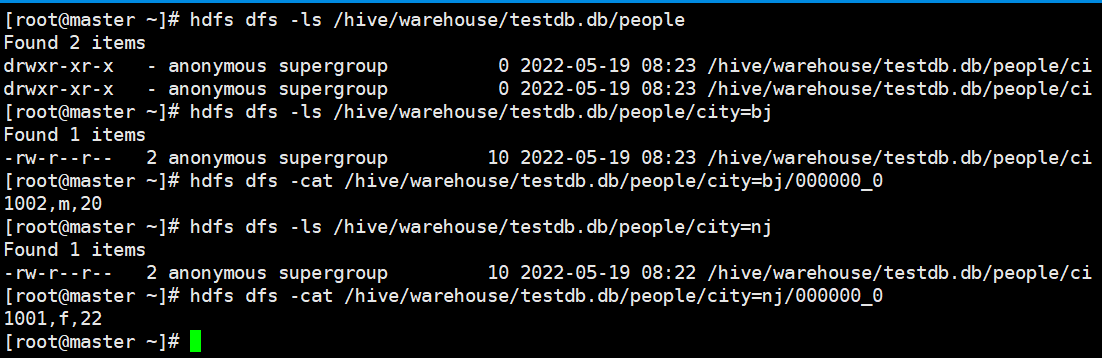
每个分区对应着表下的一个目录

HDFS是以子目录的形式存在



***-- 切换数据库*use testdb;  
*-- 创建分区表：确定哪个列用于分区*create table if not exists people  
(  
 sid int not null,  
 gender string,  
 age int  
)  
 partitioned by (city string)  
 row format delimited fields terminated by ',';  
*-- 插入数据*insert into table people  
 partition (city = 'nj')  
values (1001, 'f', 22);  
insert into table people  
 partition (city = 'bj')  
values (1002, 'm', 20);  
*-- 显示所有数据*select sid,  
 gender,  
 age,  
 city  
from people;**

  
***-- 查看HDFS存储结构***

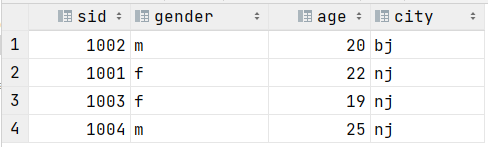


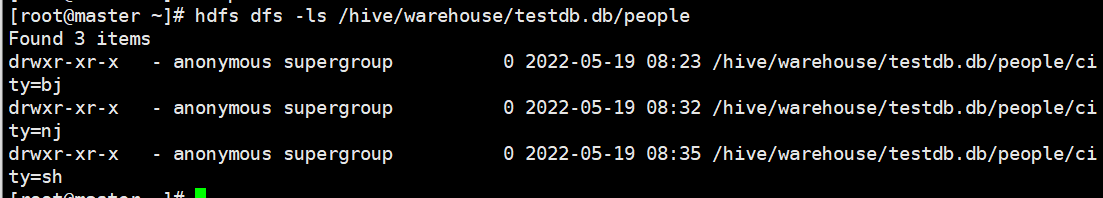
*-- 加载数据：编写数据文件且没有分区列，导入数据时需要提供分区*

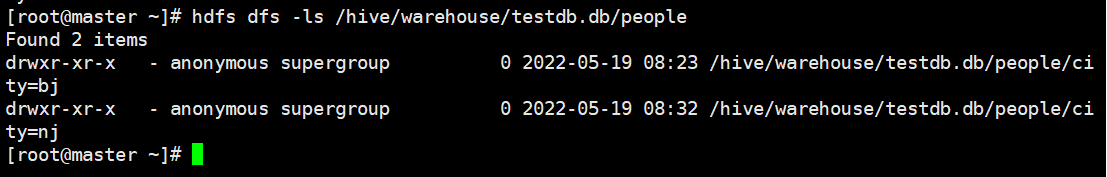




**load data local inpath '/root/p.txt'  
 into table people partition (city = 'nj');  
*-- 显示所有数据*select sid,  
 gender,  
 age,  
 city  
from people;**



***-- 增加分区：其实底层就是增加了一个子目录*alter table people  
 add partition (city = 'sh');  
  
*-- 删除分区：其实底层就是删除了一个子目录*alter table people  
 drop if exists partition (city = 'sh');**



***-- 分区表，何时使用时，才能提升查询效率：条件语句中必须包含分区表*select sid,  
 gender,  
 age,  
 city  
from people  
where city = 'bj';**



## 12、桶表

桶表

对于每张表或分区表，针对某些列进行桶的组织，实现细粒度的数据范围划分

存储在哪个桶中计算公式：列值哈希 / 桶的个数求余

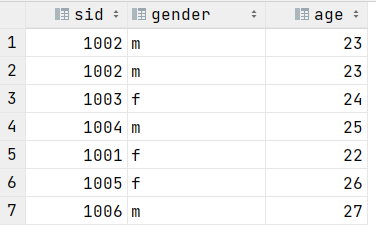
显著提升查询效率

提高取样率（抽检）：大数据，数据量高，显示所有数据不现实

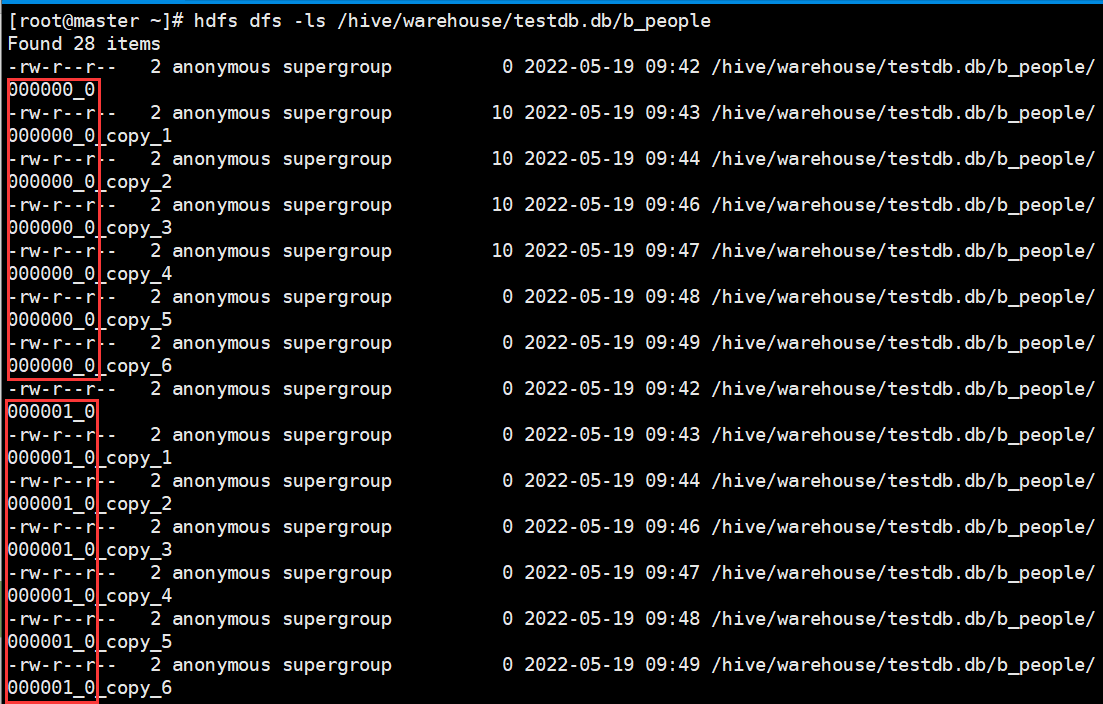
桶在物理上实际就是文件系统中的表目录中的一个文件

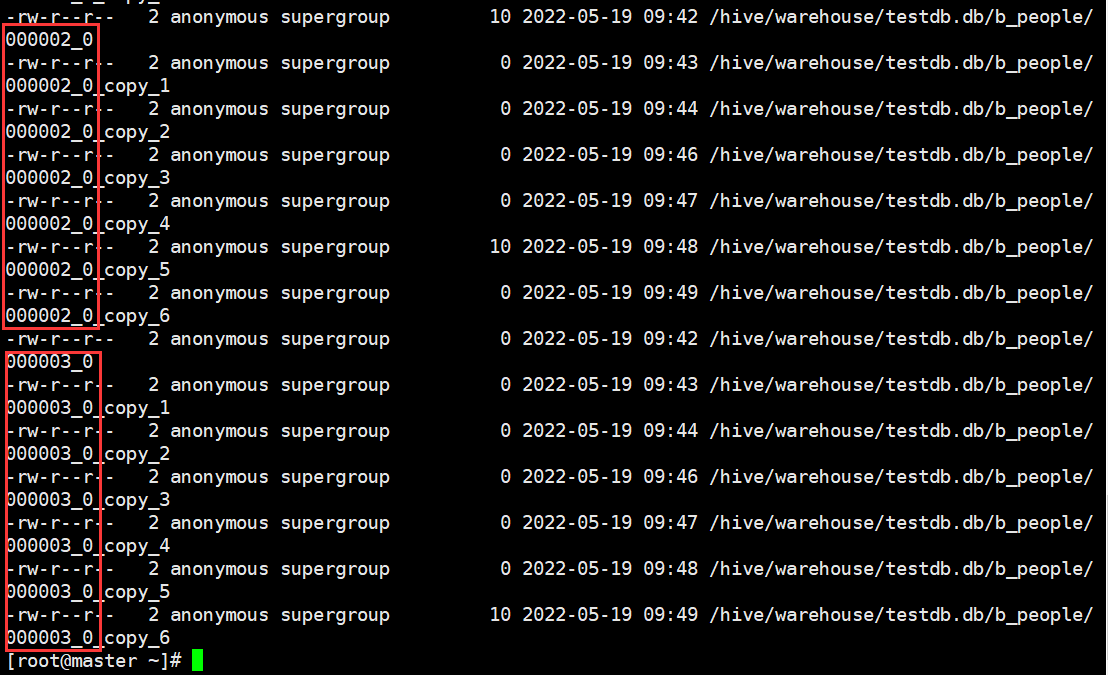
桶中的数据可以根据一个或多个列进行排序🡪归并排序，提升map端连接查询效率

***-- 切换数据库*use testdb;  
*-- 开启分桶*set hive.enforce.bucketing=true;  
*-- 设置分桶数量：其实就是reduce并发数量*set mapreduce.job.reduces=4;  
*-- 创建桶表：明确哪个列用于计算哈希值，确定桶的数量*create table if not exists b\_people  
(  
 sid int not null,  
 gender string,  
 age int  
)  
 clustered by (sid) into 4 buckets  
 row format delimited fields terminated by ',';  
*-- 插入数据*insert into table b\_people  
values (1001, 'f', 22);  
insert into table b\_people  
values (1002, 'm', 23);  
insert into table b\_people  
values (1003, 'f', 24);  
insert into table b\_people  
values (1004, 'm', 25);  
insert into table b\_people  
values (1005, 'f', 26);  
insert into table b\_people  
values (1006, 'm', 27);  
*-- 显示所有数据*select sid,  
 gender,  
 age  
from b\_people;**



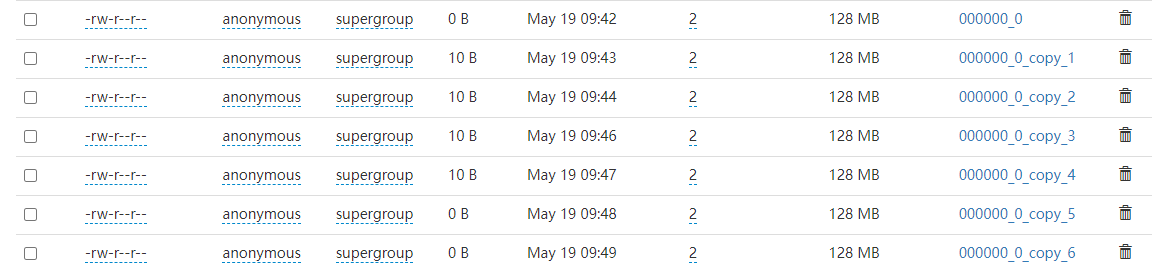
*-- 查看HDFS结构*

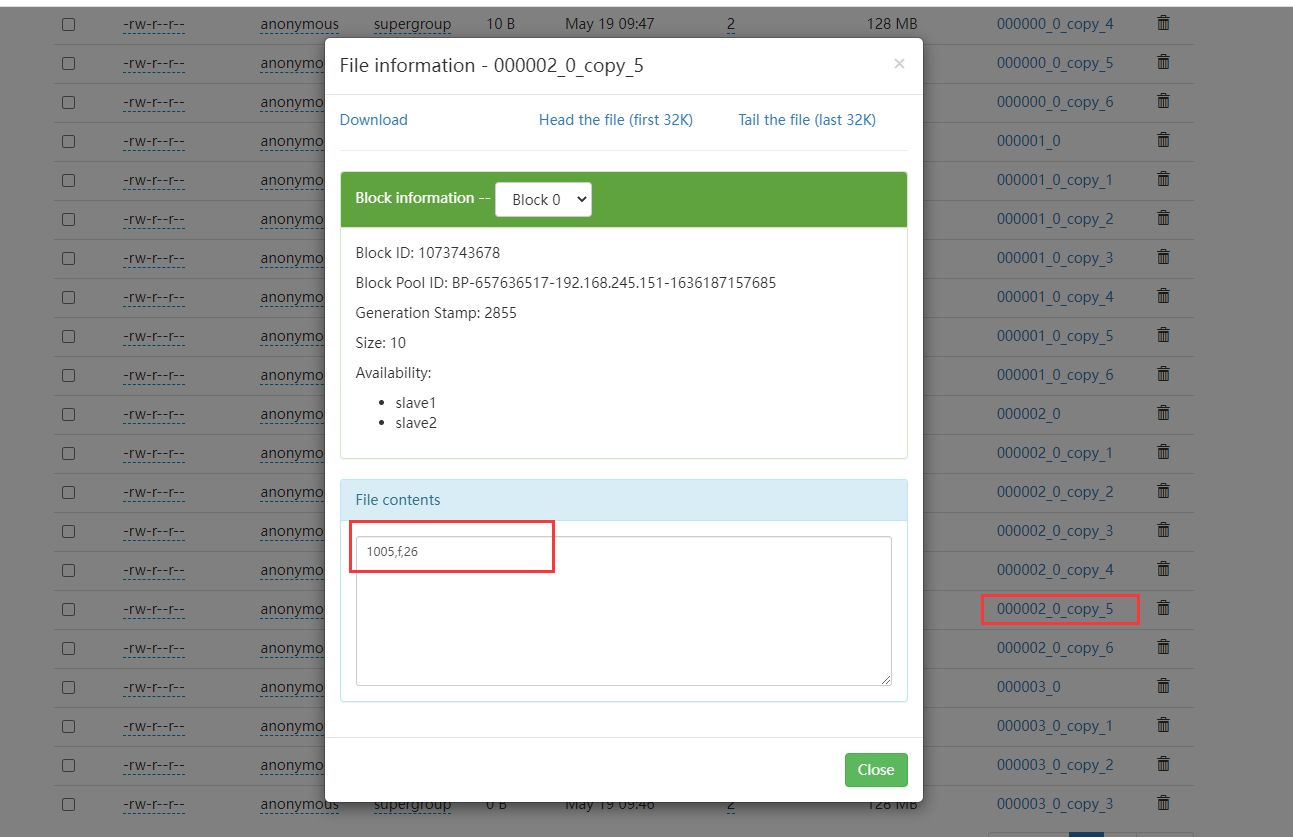
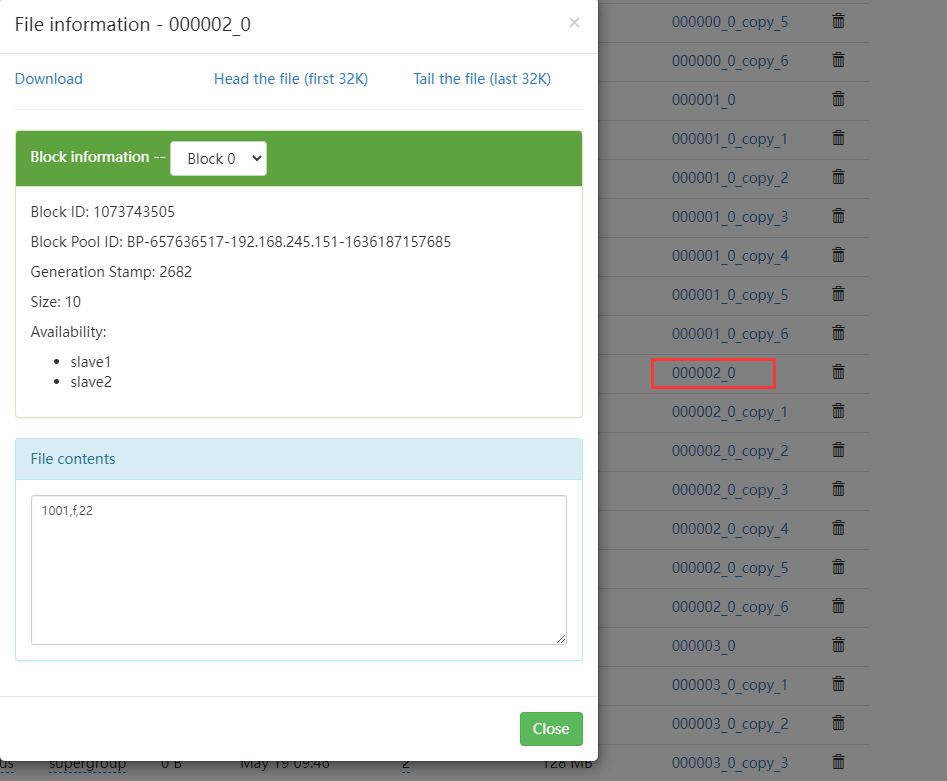




共计4个

000000、000001、000002、000003





**-- 创建桶表：明确哪个列用于计算哈希值，确定桶的数量，根据一列或多列进行排序  
create table if not exists b\_people  
(  
 sid int not null,  
 gender string,  
 age int  
)  
 clustered by (sid) sorted by (sid asc) into 4 buckets  
 row format delimited fields terminated by ',';**

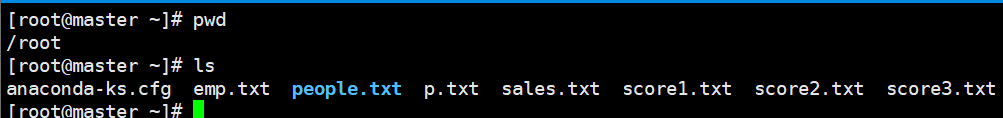
## 13、显示命令

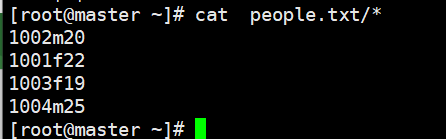
***-- 显示命令  
-- 显示所有数据库*show databases ;  
*-- 显示当前数据库的表，可以先切换数据库*show tables ;  
*-- 显示所有分区*show partitions testdb.people;  
*-- 显示所有函数*show functions ;  
*-- 查看表结构*desc testdb.people;  
desc extended testdb.people;  
desc formatted testdb.people;**

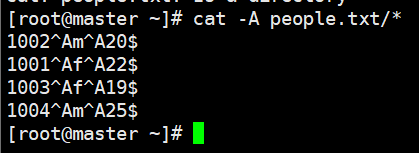
## 14、插入数据

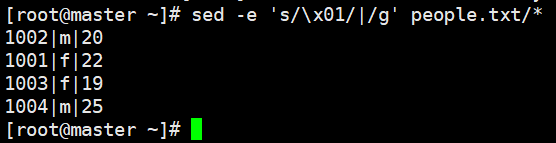
***-- 插入数据的几种形式  
-- 切换数据数据*use testdb;  
*-- 插入数据：针对分区表*insert into table people  
 partition (city = 'bj')  
values (8, 'f', 44);  
*-- 显示b\_people所有数据*select sid,  
 gender,  
 age  
from b\_people;  
*-- 插入数据：针对普通表、桶表*insert into table b\_people  
values (1009, 'f', 33);  
*-- 插入数据：覆盖形式,相当于先清空原表中数据，再插入新数据*insert overwrite table b\_people  
values (1009, 'f', 33);  
*-- 插入数据：模拟删除1009的数据*insert overwrite table b\_people  
select sid, gender, age  
from b\_people  
where sid != 1009;  
*-- 插入数据：模拟清空表操作*insert overwrite table b\_people  
select sid, gender, age  
from b\_people  
where 1 = 2;  
*-- 插入数据：多条数据插入*insert into table b\_people  
select 1008, 'm', 31  
union  
select 1009, 'f', 33;  
*-- 插入数据：插入多张表*from people  
insert  
into table b\_people  
select sid, gender, age  
where city = 'bj'  
insert  
into table p\_people  
select sid, gender, age  
where city = 'nj';  
*-- 使用with ... as ...：可以出现在任何语句中，相当于是中间结果  
-- with as 必须是第1条语句，后续可以重复使用*with tmp as (select sid, gender, age  
 from people)  
insert  
into table b\_people  
select sid, gender, age  
from tmp;**

## 15、导出数据

***-- 导出数据  
-- 本地文件系统*insert overwrite local directory '/root/people.txt'  
select sid, gender, age from people;  
*-- 查看本地文件***





 ***-- HDFS文件系统*insert overwrite directory '/people.txt'  
select sid, gender, age from people;  
*-- 同时导出至多个文件*from people  
insert  
overwrite  
local  
directory  
'/root/people.txt'  
select sid, gender, age  
insert  
overwrite  
directory  
'/people.txt'  
select sid, gender, age;**

## 16、清空表数据

模拟删除

insert overwrite table t\_table1 select \* from t\_table1 where 1=0;

truncate 表名;

清空表中所有数据

不能删除外部表

删除数据操作是不可逆的

删除表

drop table 表名;

drop table 表名 purge;🡪永久删除，不准备恢复

# 四、

## 1、

## 2、

## 3、

## 4、

## 5、

## 6、

## 7、

## 8、

## 9、

## 10、

# 五、

## 1、

## 2、

## 3、

## 4、

## 5、

## 6、

## 7、

## 8、

## 9、

## 10、

# 六、

## 1、

## 2、

## 3、

## 4、

## 5、

## 6、

## 7、

## 8、

## 9、

## 10、

# 七、

## 1、

## 2、

## 3、

## 4、

## 5、

## 6、

## 7、

## 8、

## 9、

## 10、

# 八、

## 1、

## 2、

## 3、

## 4、

## 5、

## 6、

## 7、

## 8、

## 9、

## 10、

# 九、

## 1、

## 2、

## 3、

## 4、

## 5、

## 6、

## 7、

## 8、

## 9、

## 10、

# 十、

## 1、

## 2、

## 3、

## 4、

## 5、

## 6、

## 7、

## 8、

## 9、

## 10、

# 十一、

## 1、

## 2、

## 3、

## 4、

## 5、

## 6、

## 7、

## 8、

## 9、

## 10、

# 十二、

## 1、

## 2、

## 3、

## 4、

## 5、

## 6、

## 7、

## 8、

## 9、

## 10、

# 十三、

## 1、

## 2、

## 3、

## 4、

## 5、

## 6、

## 7、

## 8、

## 9、

## 10、

# 十四、

## 1、

## 2、

## 3、

## 4、

## 5、

## 6、

## 7、

## 8、

## 9、

## 10、

# 十五、

## 1、

## 2、

## 3、

## 4、

## 5、

## 6、

## 7、

## 8、

## 9、

## 10、