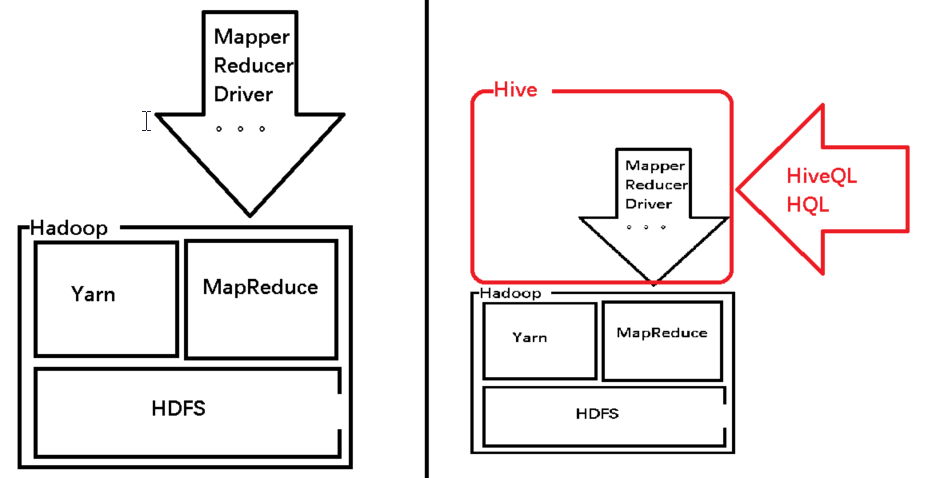
# Hive

## Hadoop在分布式数据处理中遇到的问题

* MR开发调试复杂，不适合要求快速得出结果的场景。
* Hadoop由Java开发，对JAVA支持最好，对其他语言的使用者不够友好。
* 需要对Hadoop底层具有一定的了解，并且熟悉API才能开发出优秀的MR程序。

## 概述

Hive是一个建立在Hadoop基础之上的数据仓库工具，以HiveQL（类SQL）的操作方式让我们能够轻松的实现分布式的海量离线数据处理。而不必去编写调试繁琐的MR程序。

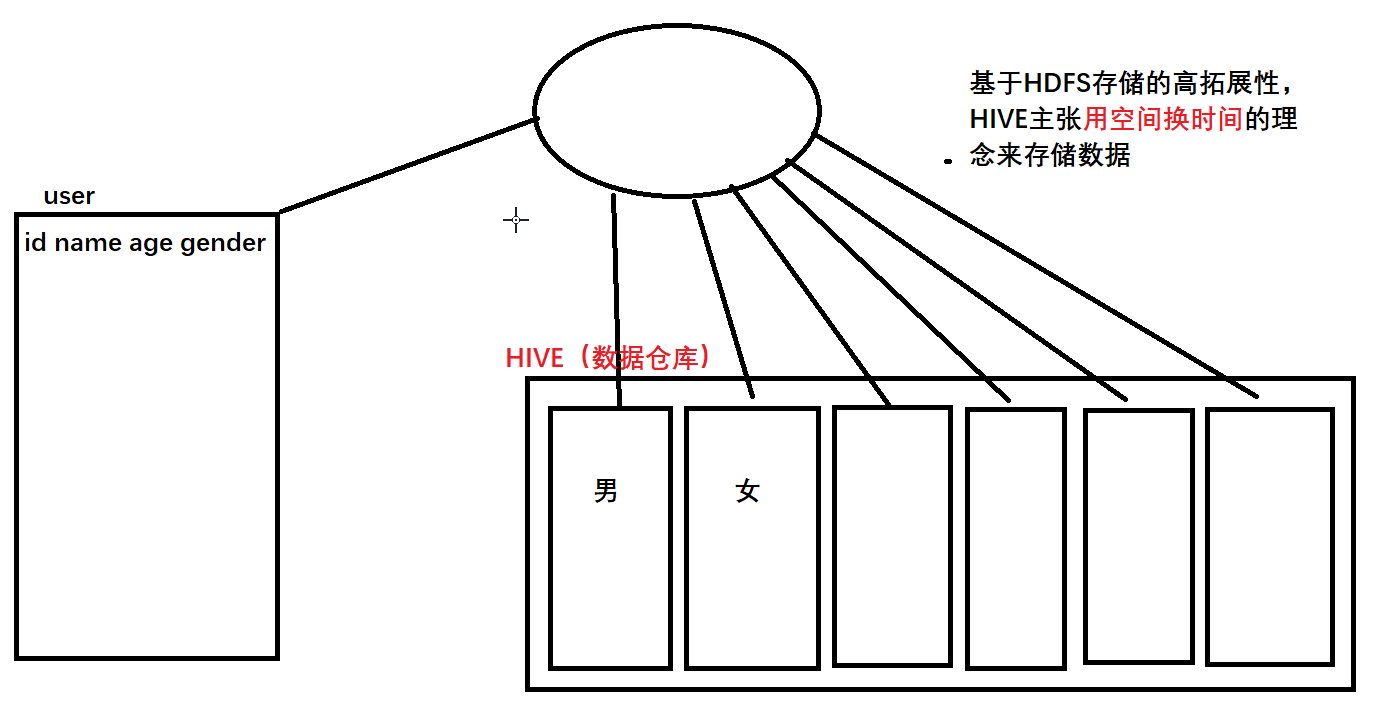


优点：

* 避免了MR繁琐的开发调试过程，Hive自动将我们输入的HQL编译为MR运行
* HQL这种类SQL语言对于任何开发语言的程序员来说都比较友好。
* 我们不需要对Hadoop底层有太多的理解，也不用记忆大量的API就能实现分布式数据的处理
* Hive也提供了自定义函数的方式来补充自身函数库可能存在的不足。即编写JAVA代码来实现复杂的逻辑并封装为UDF（自定义函数）来供我们重复使用。

### 数据仓库

数据仓库是一个面向主题的，稳定的，反应历史数据的数据存储工具，他主要支持管理者的决策分析。

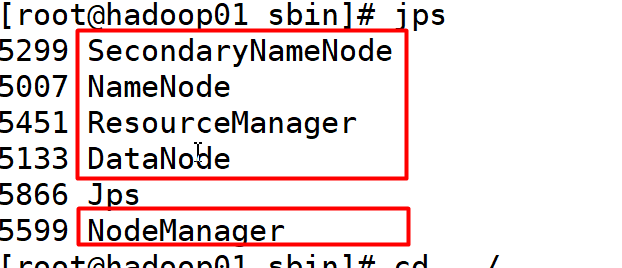


本节总结：

Hive其实就是在Hadoop外边套了一层壳，底层其实就是Hadoop，数据存储用HDFS，数据处理用MR，不同的是我们在数据处理中不需要再去编写复杂的MR而是用简单的类SQL（HiveQL）就能实现海量离线数据的分析。

## 安装

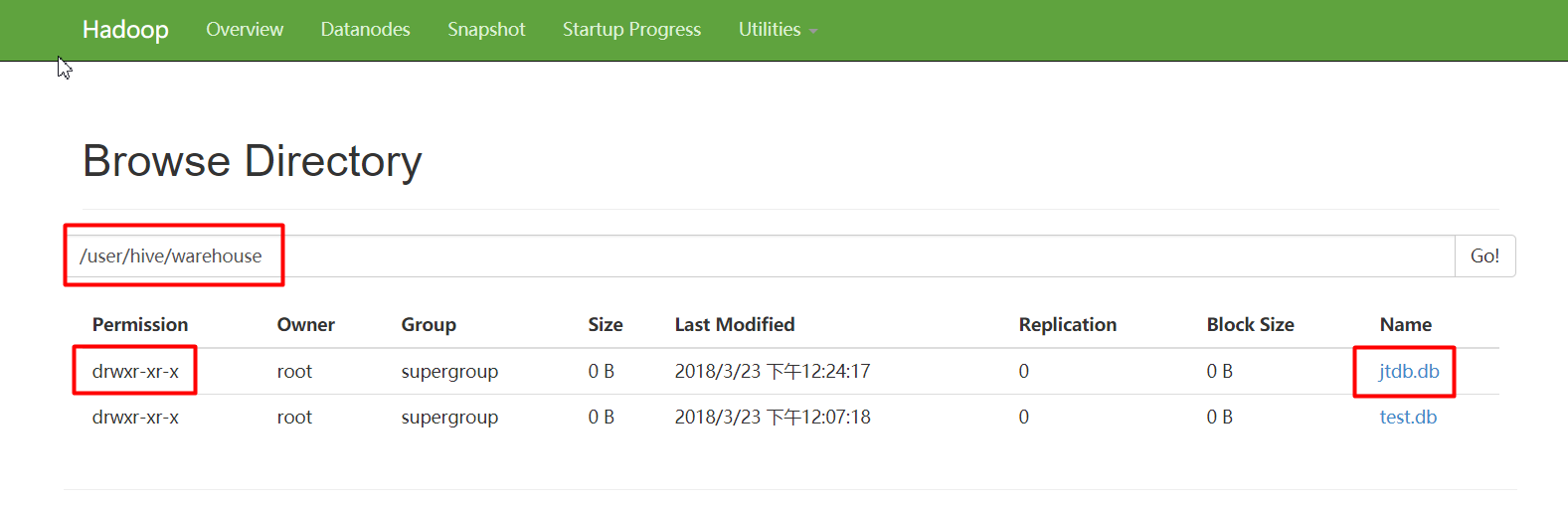
Hive安装非常简单，解压之后即可直接运行，不需要太多配置，前提是要配置JAVA\_HOME和HADOOP\_HOME。并且Hadoop要全量启动（五个进程）



## 基础操作

### 创建数据库

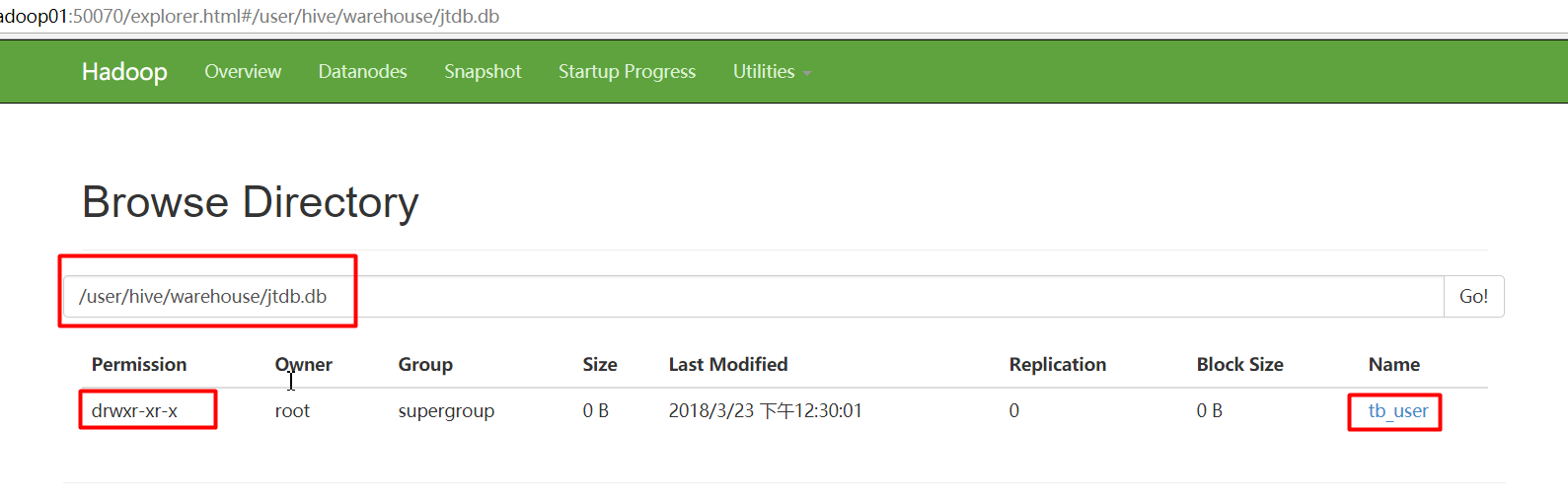
create database jtdb;



结论1：Hive中的数据库，其实就是HDFS中/user/hive/warehouse目录下的以.db结尾的文件夹。

### 创建表

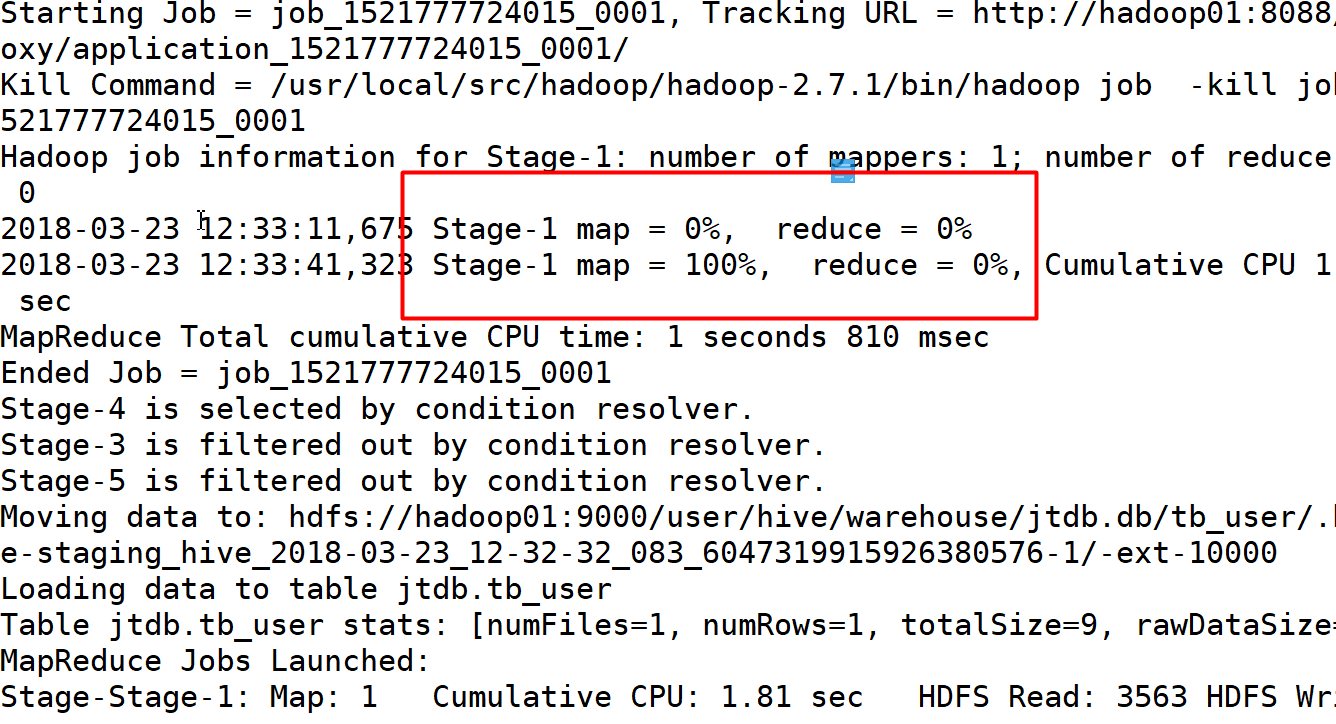
create table tb\_user(id int,name string);



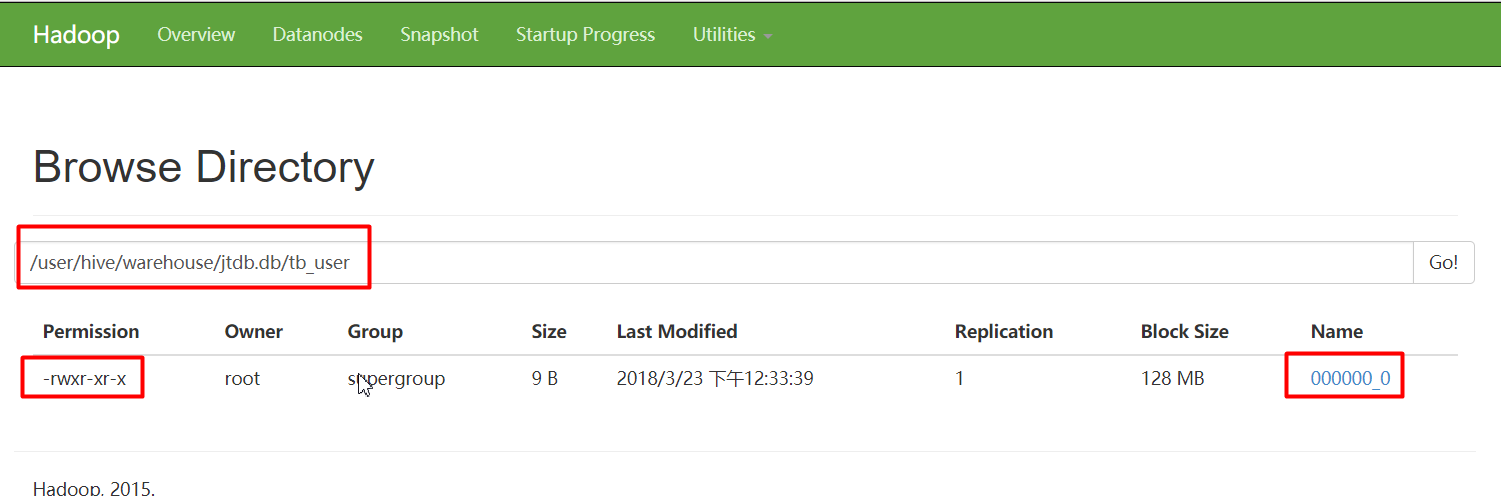
结论2：Hive中的表其实就是HDFS中对应数据库文件夹下的一级文件夹。

### 插入数据

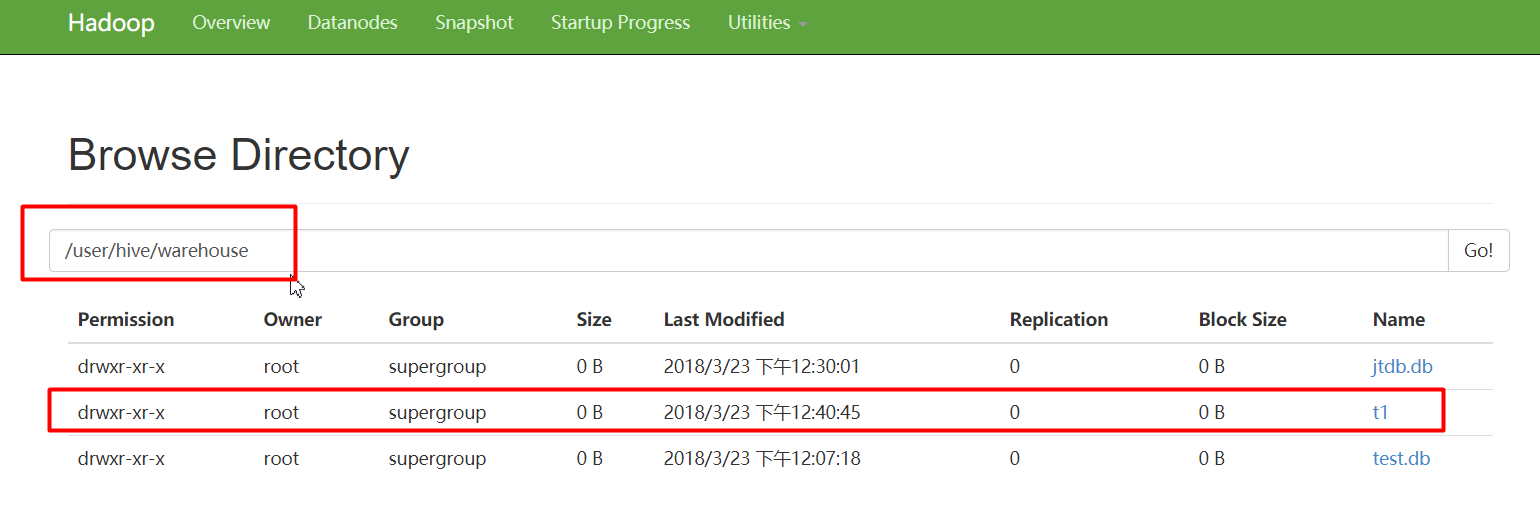
insert into table tb\_user values(1,"dongcc");



结论3：Hive在必要的时候会将HQL编译为MR来执行，上图展示了变异后的MR执行的过程。



结论4：hive中的数据在HDFS中以文件的形式存储在其所在表的文件夹中。



结论5：默认数据库default没有自己单独的文件夹，其下的表直接存放在/user/hive/warehouse下。

### 加载数据

Load data local inpath ‘/usr/local/src/hive/data/book.txt’ into table table\_name;

### 可能遇到的问题

* Hive启动不了

检查JAVA\_HOME和HADOOP\_HOME是否配置成功。如果没有问题并报错：Cannot find hadoop installation: $HADOOP\_HOME or $HADOOP\_PREFIX must be set or hadoop must be in the path

**解决办法：**

指定HADOOP\_HOME路径

cd /usr/local/src/hive/apache-hive-1.2.0-bin/conf

cp hive-env.sh.template hive-env.sh

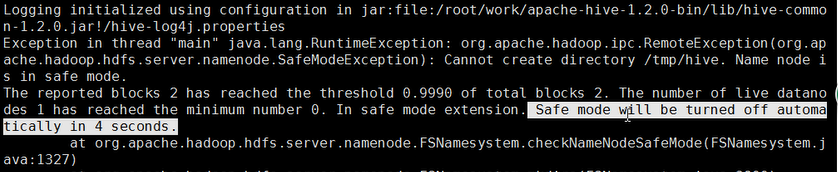
vim hive-env.sh

增加HADOOP\_HOME



HADOOP\_HOME=/usr/local/src/hadoop/hadoop-2.7.1

#### Hive启动报错Safe mode



Hadoop在启动时有个安全模式，其在启动时有些工作要做，元数据的处理，DataNode的等待等过程。需要一段时间，遇到时需要等一段时间，耐心稍微等一会。过会自动就会好。

如果长时间还报错，还在安全模式。可以手工设置退出安全模式。

[root@hadoop01 bin]# pwd

/usr/local/src/hadoop/hadoop-2.7.1/bin

[root@hadoop01 bin]# ./hadoop dfsadmin -safemode leave

DEPRECATED: Use of this script to execute hdfs command is deprecated.

Instead use the hdfs command for it.

Safe mode is OFF

[root@hadoop01 bin]#

参数value的说明如下：

* enter - 进入安全模式
* leave - 强制NameNode离开安全模式
* get - 返回安全模式是否开启的信息
* wait - 等待，一直到安全模式结束

本节总结：

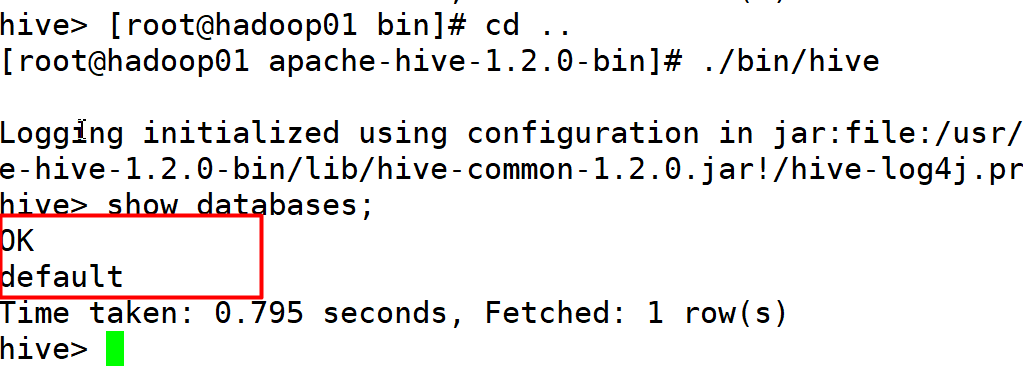
Hive的设计其实非常简单，其中数据库和表都是HDFS中的文件夹，而数据则是文件夹中的文件。Hive只是来维护这些文件夹和文件的关系。

## 元数据库的替换

Hive没有将描述数据库、表、数据之间关系的元数据直接存放在HDFS中，而是存放在了传统的关系型数据库中，这样保证了元数据可以快速的进行增删改查。

Hive原生的将元数据保存在了内置的Derby数据库中。

Derby存在的问题：我们经过试验发现，当我们更换了hive的启动发起目录后，原来的数据都读取不到了。但是HDFS中数据都在，这是为什么？



我们发现，Derby以当前目录为基础，创建matestore\_db文件来存放数据，当我们更换了启动目录后，Derby重新创建了这个文件来存储元数据，这是原始元数据读取不到，导致我们无法读到Hive中存放的数据。

这种情况我们是无法忍受的，实际开发中不会使用Derby来做Hive的元数据库。所以我们要将他替换掉。以mysql为例。

1. 修改配置文件hive-site.xml

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://192.168.65.161:3306/hive?createDatabaseIfNotExist=true</value>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

</property>

<property>

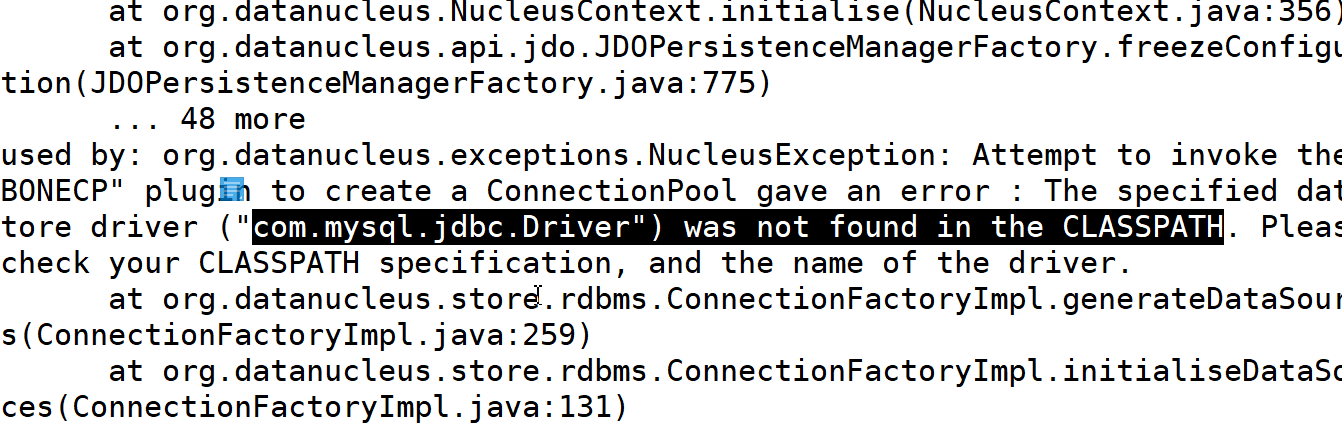
<name>javax.jdo.option.ConnectionPassword</name>

<value>root</value>

</property>

</configuration>

将配置文件修改后并不能成功启动hive，报错如下：

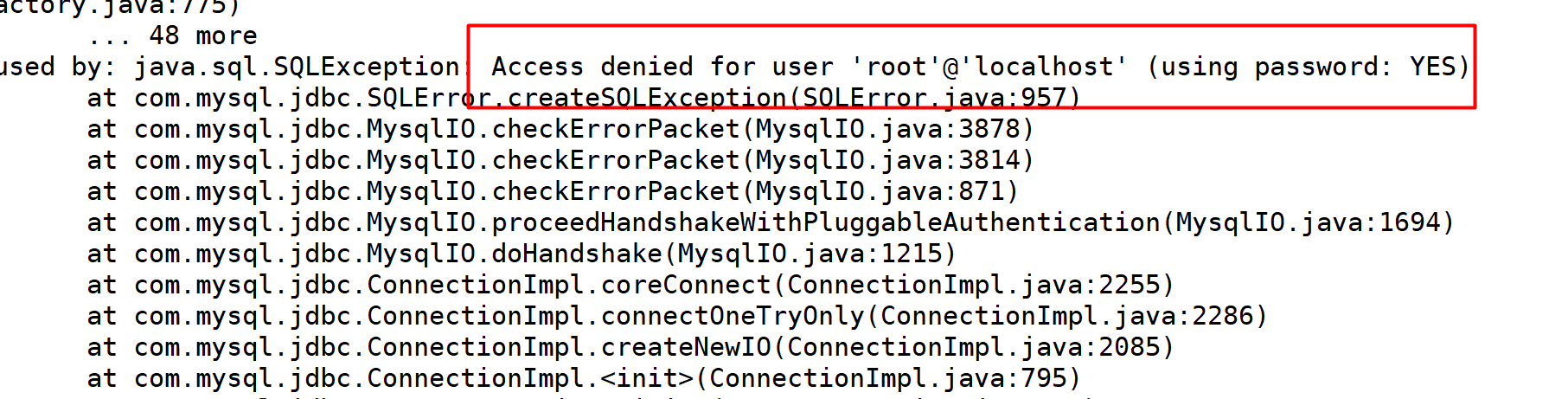


原因，我们并没有给hive相关的驱动包。

2.导入驱动包

直接将mysql驱动包上传到hive的lib目录中即可

但是再次尝试启动仍然报错，



没有访问权限，解决办法：开启mysql外部访问权限。

grant all privileges on \*.\* to 'root'@'%' identified by 'root' with grant option;

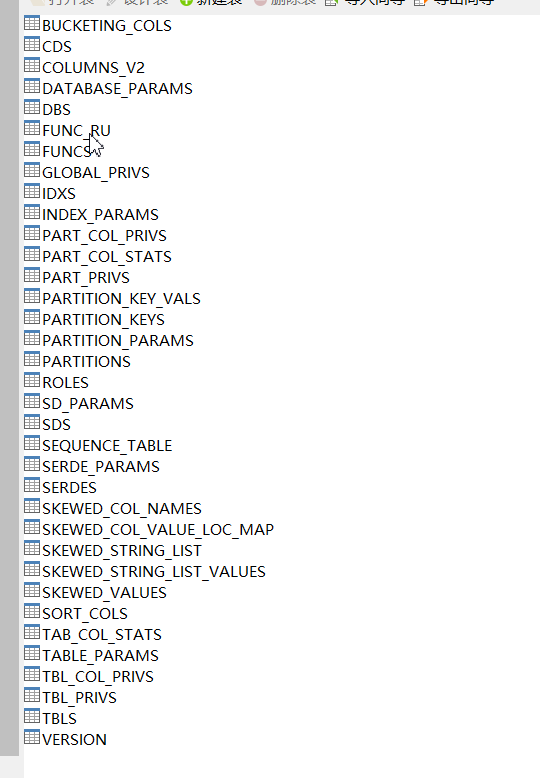
flush privileges; #更新权限

如果上面修改后还提示权限错误，修改指定机器

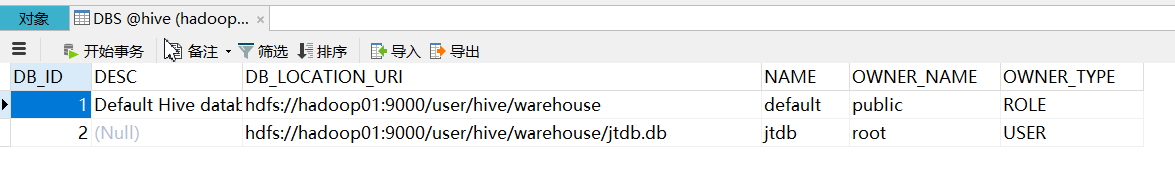
grant all privileges on \*.\* to 'root'@'hadoop01' identified by 'root' with grant option;

flush privileges;

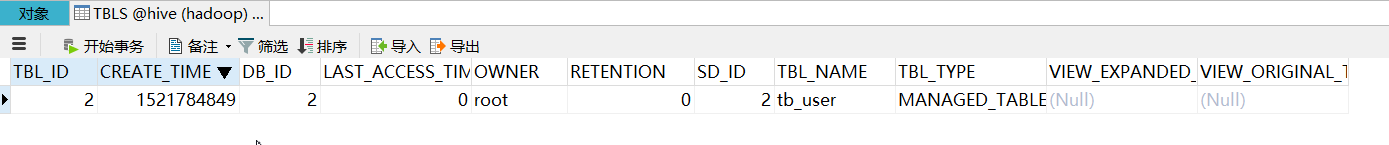
### 元数据信息



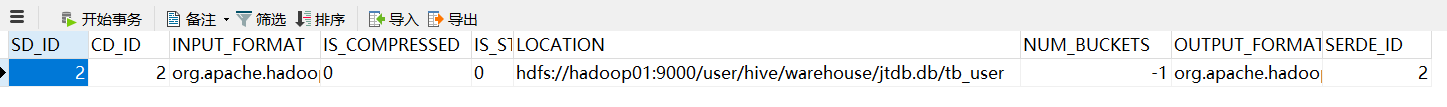
DBS：数据库信息



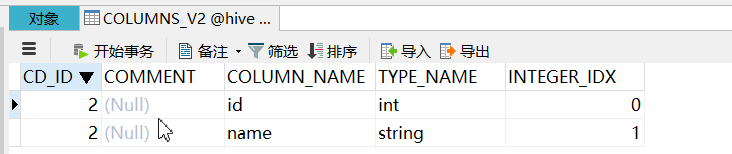
TBLS：表信息



SDS：表详细信息



COLUMNS\_V2：列信息



本节总结

Hive原生的使用Derby作为元数据库，但是Derby具有天生的缺点，不能用作生产环境，所以我们需要将它替换成mysql来作为我们的元数据库。注意！！！元数据库中hive表的字符集必须是latin1！！！否则在执行Hql过程中会报错！

Hive为什么要用传统关系型数据库来作为他的元数据库而不用HDFS来存储元数据？关系型数据库支持实时的增删改查！

## 内部表和外部表

### 内部表（托管表）：MANAGED\_TABLE

Hive中先创建表后插入或上传数据的表称之为内部表。

创建语句：

create table table\_name (id int,name string) row format delimited fields terminated by ‘\t’;

加载数据：

load data local inpath ‘文件路径’ into table table\_name;

删除表：

Drop table table\_name；

删除表时，连同数据一起被删除。

再次加载数据一样可以被管理到。

### 外部表：EXTERNAL\_TABLE

Hive中先有数据后创建表来管理数据的表叫做外部表。

创建表：在创建之前先要有数据。

Create external table table\_name (id int ,name string) row format delimited fields terminated by ‘\t’ location ‘HDFS中文件所在路径’

删除表：

Drop table table\_name；

外部表在删除时，只会将元数据信息删掉，而不会删除数据本身。

再次加载数据一样可以被管理。

### 内外部表的区别：

内部表在被drop的时候，元数据信息会被删除，表数据也会被删除。

外部表在被drop的时候，元数据信息会被删除，原来的表数据不会被删除

本节总结：  
 Hive中最基本的表分类为内部表和外部表，内部表是现有表再有数据，外部表是先有数据后有表，在表被删除时，内部表中元数据和表数据都会被删除，而外部表只会把元数据删除而保留原本的表数据。

## 分区表

在hive中数据的面向主题存储就是由表的分区来实现的。

### 单级分区

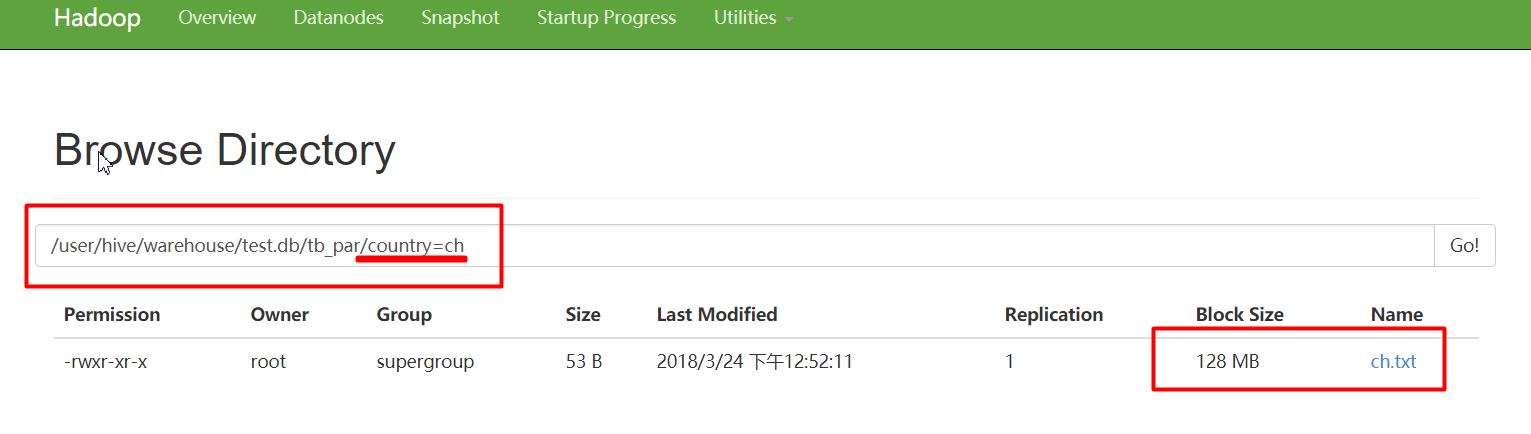
创建分区表：

create table table\_name (id int,name string) partitioned by (country string) row format delimited fields terminated by ‘\t’;

与普通表相比只是多了partitioned by (country string)

为分区表中载入数据：

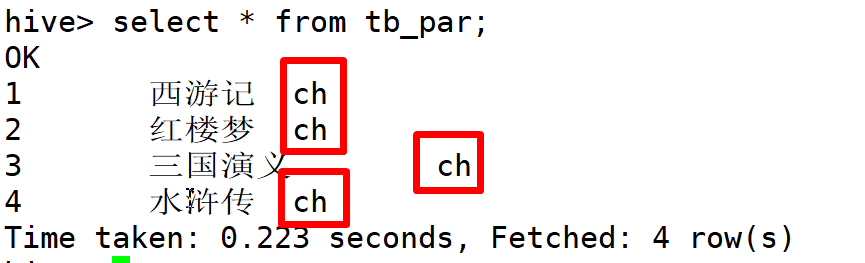
load data local inpath ‘文件位置’ overwrite into table table\_name partition (country=’ch’);



经过查看HDFS中存储发现：

分区在表所在目录中以文件夹形式存在，数据落在分区目录中，并且分区文件夹以分区类型=分区名定义如上图country=ch；

另外在select \* from tb\_par；时发现：



分区信息查询结果中展示，但是并不是给数据增加了一列。而是单独的信息绑定。

### 再次加载数据不能直接被管理

在分区表中仍然支持手动直接上传文件到HDFS中，但是需要我们按照对应的目录标准创建分区目录，并且需要

alter table table\_name add partition （country=’jp’）location ‘自定义的分区目录’；

这样元数据信息中才能有该分区的信息，hive才能读取到这些我们自己上传的数据。

### 多级分区

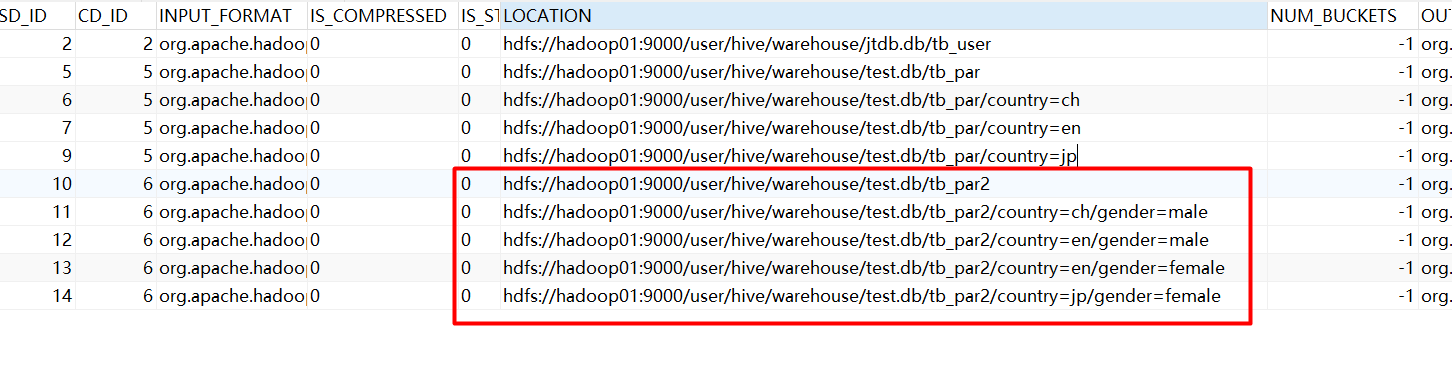
创建多级分区表：

create table table\_name (id int,name string) partitioned by (country string，gender string) row format delimited fields terminated by ‘\t’;

加载数据：

load data local inpath '/usr/local/src/hive/data/jp\_female.txt' overwrite into table tb\_par2 partition (country='jp',gender='female');

查看元数据：发现分区就是多了几级目录。



查询多级分区目录：

select \* from tb\_par2 where gender = 'female' and country = 'jp';

本节总结：

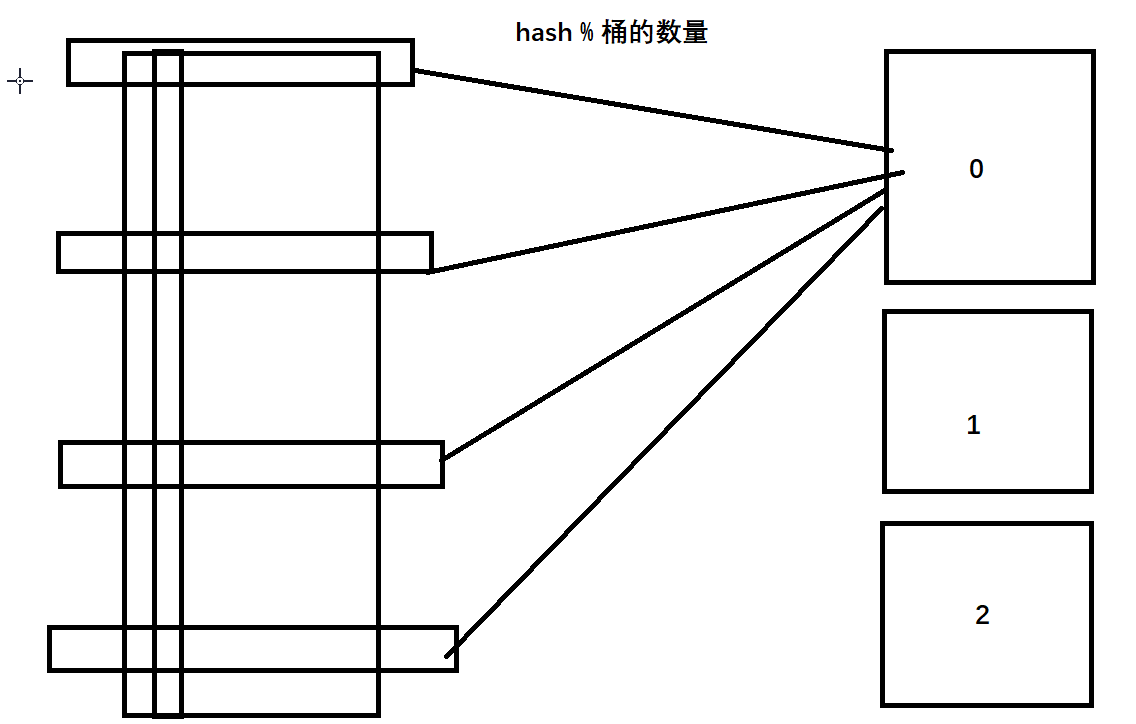
Hive中的分区表，实现了hive对数据的面向主题存储，在条件查询过程中加快了查询速度，本质就是把同主题的数据放在同一个文件夹下。并且支持多级分区。需要注意的是，直接在HDFS中上传数据需要通过alter语句为元数据信息中增加分区信息才能是hive正确管理这些直接上传的数据。

注意：

在实际的生产中分区不是随意设置的，也不是越多越好，而是将经常查询的维度设置为分区即可！

## 分桶表

在测试过程中，每次都将全表数据加载进来会花费非常多的时间，而随意截取部分数据进行测试又不具有代表性，这时我们可以使用分桶的方法，来减少测试数据量，并使结果具有较高的代表性。



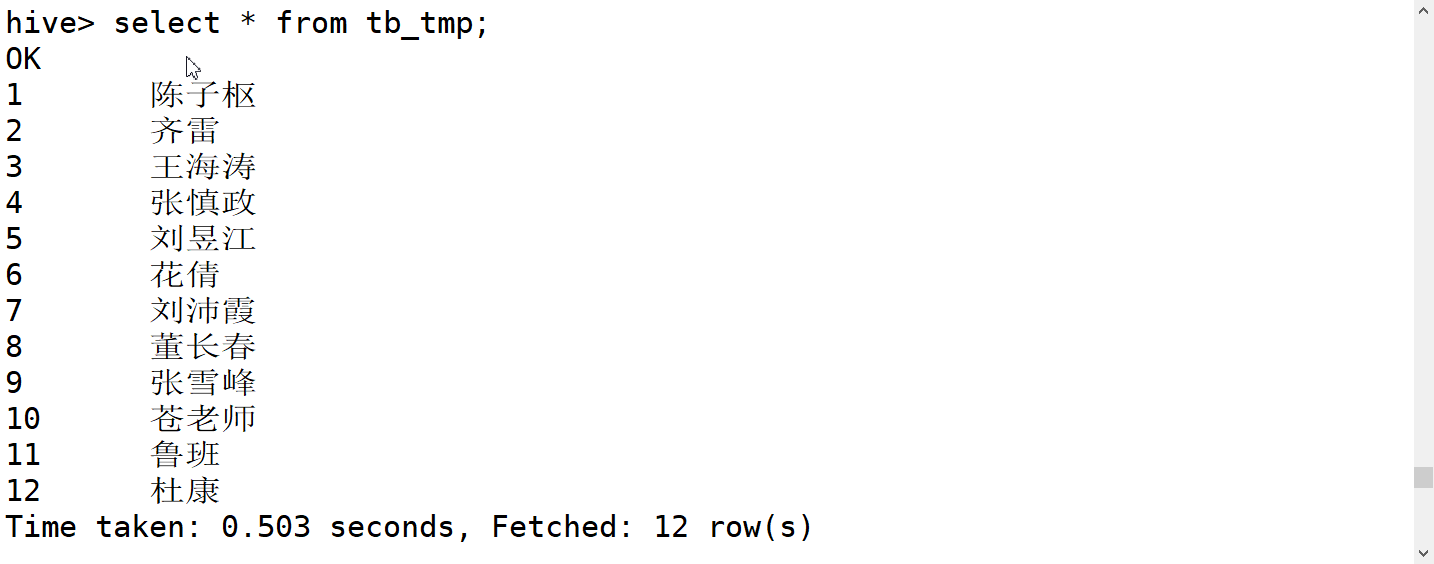
### 创建分桶表：

Create table table\_name (id int,name string) clustered by (id) into 4 buckets row format delimited fields terminated by ‘,’;

### 加载数据：

分桶表不支持直接加载数据，需要从其他表中导入。（符合实际生产情况，只有测试阶段需要分桶）

创建普通外部表，并将数据载入。然后导出到分桶表中。

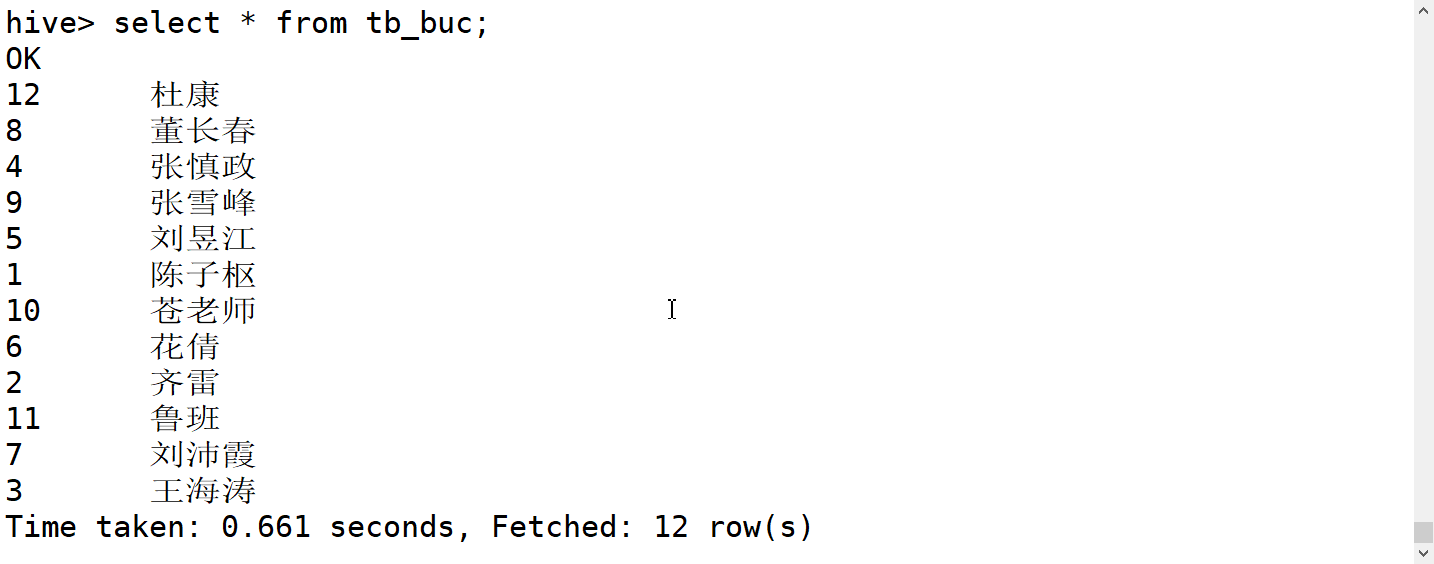


Hive默认未开启多个reduce，分桶需要多个reduce同时工作，所以这里要开启分桶模式：

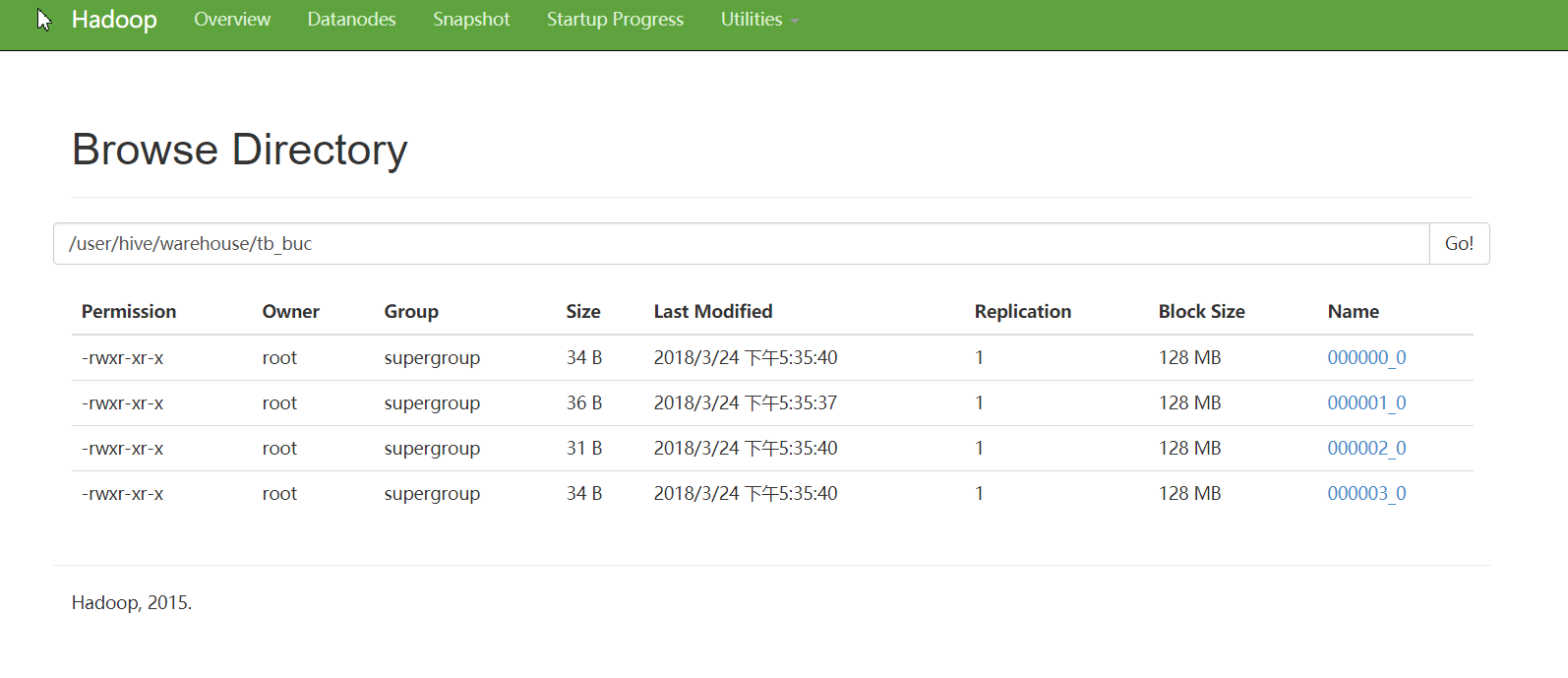
hive> set hive.enforce.bucketing=true;

然后进行导入：这个过程时间较长，桶越多时间越长。

Insert overwrite table table\_name select \* from table\_tmp;



数据顺序发生了改变，查看HDFS存储发现整个文件被拆分成四部分，自动命名0~3；



### 数据取样

Select \* from table\_name tablesample （bucket x out of y on id）;

其中x为第几个桶，不能大于总桶数， y为选取数据的步长(几个桶取一次数据)。Y应为桶数的因数或倍数，当y大于桶数时，选取结果非整桶抽取，而是抽取每桶中的y除以桶数分之一，即：1/(y/桶数)

按百分比取样：整表数据额百分之三十

Select \* from table\_name tablesample(30 percent);

本节总结：

Hive中分桶表的目的是为了测试方便，不至于频繁使用整表数据进行测试，拖慢测试速度，而是选取部分具有代表性的数据进行测试，hash值模余分桶的方式保证了抽样数据的代表性。（hash的散列）