Hive教案

|  |  |  |  |
| --- | --- | --- | --- |
| 作者 |  | 当前版本： | **1.0** |
| 评审人 |  | 前一版本： | 版本号/无 |
| 状态： | 草稿/定稿 | 发布日期： |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **作者** | **修改内容** |
| 2015-12-10 | 1.0 |  | 初始版本 |
| 2016-04-20 | 1.1 |  | 部分修改 |
|  |  |  |  |

# 1.Hive介绍

## 1.1 Hive是怎么出现的,？

1.在大数据技术出现之前，传统的数据分析大多是用什么工具来实现的？

答案：sql 关系型数据库

2.这种工具相对mapreduce这中方式来说有什么优点？

答案：不用进行思维转换，同时也更加接近人类语言，使用人群较大

3.能不能把这种工具通过某种方式跟hadoop结合起来，实现用原来处理问题的思维来处理海量数据？

答案：能，这个东西就是hive

## 1.2 Hive是什么

Hive是基于Hadoop的数据仓库解决方案。由于Hadoop本身在数据存储和计算方面有很好的可扩展性和高容错性，因此使用Hive构建的数据仓库也秉承了这些特性。

简单来说，Hive就是在Hadoop上架了一层SQL接口，可以将SQL翻译成MapReduce去Hadoop上执行，这样就使得数据开发和分析人员很方便的使用SQL来完成海量数据的统计和分析，而不必使用编程语言开发MapReduce那么麻烦。

Hive就是我们所说的使用传统的sql来分析海量数据的工具。

可以把Hive当成一个“数据库”，它也具备传统数据库的数据单元，数据库（Database/Schema）和表（Table）.但实际上只是Hadoop上的一个工具。

**所以从技术角度来看：**

hive就是一个翻译器，将传统的sql转化成mapreduce。

或者说：hive是基于mapreduce 的一个上层数据分析框架

## 1.3 Hive能用来干什么

Hive的两个组成部分：

1.基于mapreduce

2.使用传统sql的编程思想

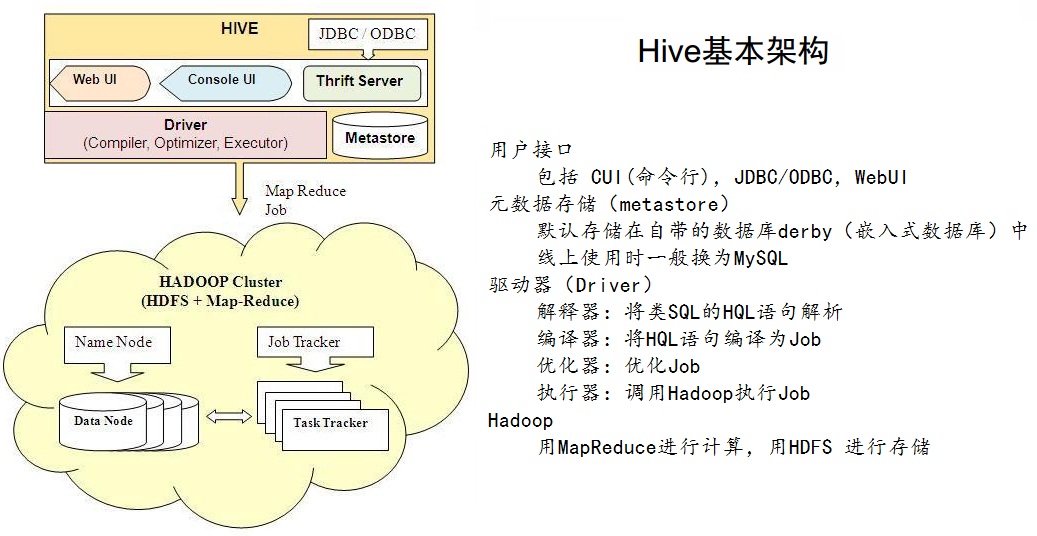
所以结合hive的两大主要组件来看，hive的用处如下：

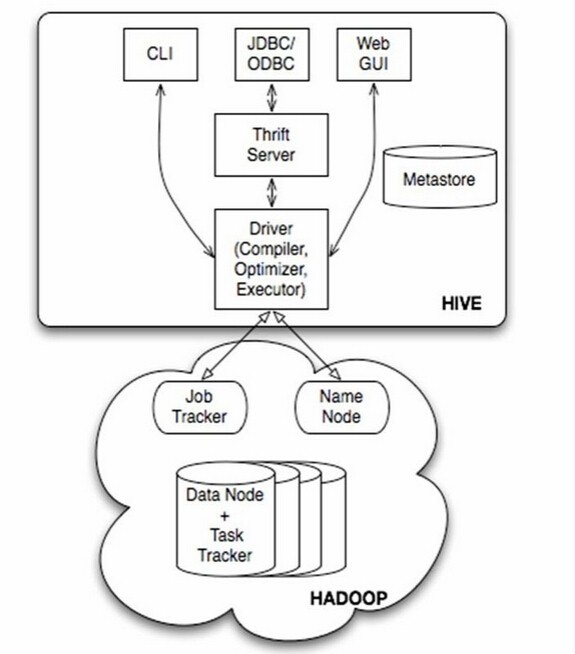
hive能够用来进行海量数据的存储、查询和离线分析。

# Hive原理

## 2.1 Hive运行原理

### 2.1.1Hive架构





 Hive通过给用户提供的一系列交互接口，接收到用户的指令(SQL)，使用自己的Driver，结合元数据(MetaStore)，将这些指令翻译成MapReduce，提交到Hadoop中执行，最后，将执行返回的结果输出到用户交互接口。

在使用过程中，至需要将Hive看做是一个数据库就行，本身Hive也具备了数据库的很多特性和功能。

### 2.1.2 Hive执行流程

编译器将一个Hive Query Language转换操作符，操作符是Hive的最小的处理单元，每个操作符代表HDFS的一个操作或者一道MapReduce作业。Hive定义一个处理过程为Operator。

|  |  |
| --- | --- |
| 操作符 | 描述 |
| TableScanOperator | 扫描hive表数据 |
| ReduceSinkOperator | 创建将发送到Reducer端的<Key,Value>对 |
| JoinOperator | Join两份数据 |
| SelectOperator | 选择输出列 |
| FileSinkOperator | 建立结果数据,输出至文件 |
| FilterOperator | 过滤输入数据 |
| GroupByOperator | GroupBy语句 |
| MapJoinOperator | /\*+mapjoin(t) \*/ |
| LimitOperator | Limit语句 |
| UnionOperator | Union语句 |

Hive通过ExecMapper和ExecReducer执行MapReduce任务。

## 2.2 Hive数据库

Hive中的数据库概念，本质上仅仅是表的一个目录结构或命名空间。

## 2.2 Hive表

Hive中的表概念，本质上仅仅是数据的一个目录结构或命名空间。

2.2.1Hive的数据单元

Databases：数据库。概念等同于关系型数据库的Schema，不多解释；

Tables：表。概念等同于关系型数据库的表，不多解释；

Partitions：分区。概念类似于关系型数据库的表分区，没有那么多分区类型，只支持固定分区，将同一组数据存放至一个固定的分区中。

Buckets (or Clusters):分桶。同一个分区内的数据还可以细分，将相同的KEY再划分至一个桶中，这个有点类似于HASH分区，只不过这里是HASH分桶，也有点类似子分区吧。

### 2.2.2管理表（内部表）

Hive会控制表中数据的生命周期，当删除一张管理表时，同时也会删除表中的数据。管理表中的数据不与其他工具共享。

### 2.2.3外部表

Hive只控制元数据，当删除一张外部表时，只会删除元数据，并不会删除真实数据。外部表中的数据与其他工具共享。

**内、外部表的区别：**

删除表时：

数据是否共享：

一些HiveQL语法不试用于外部表。

### 2.2.4分区管理表、分区外部表

Hive中的分区表类型并不复杂，通常以表中的某一个或多个列为分区依据，并创建文件夹，将表中的其他列中的数据放到该文件夹下的数据文件中。

**创建分区表的最主要的目的是：快速查询！**

分区还可以分为静态分区和动态分区，静态分区的分区字段不能增加，动态分区相反。默认情况下的分区表是静态的。

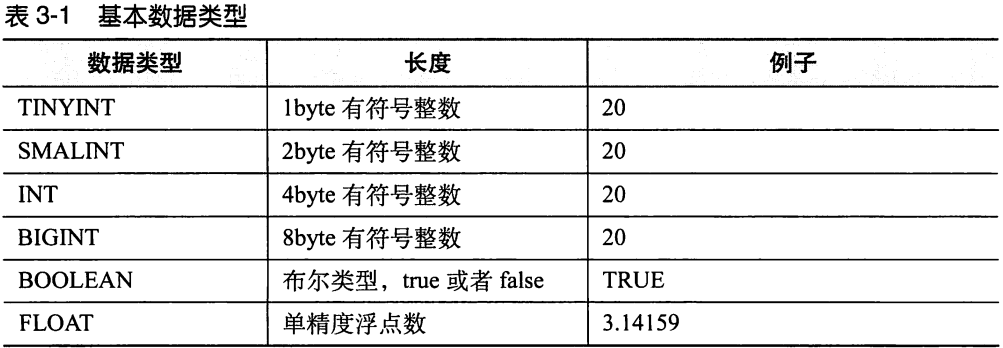
## 2.3 Hive分桶

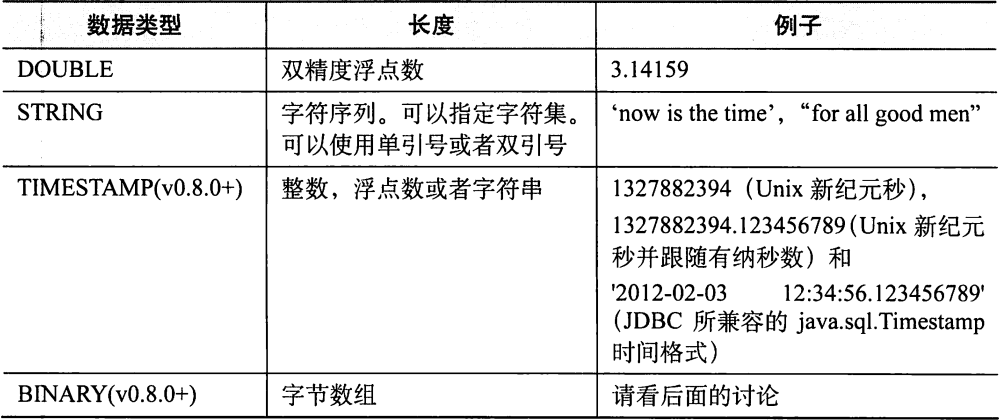
Hive以表中的某一列作为分桶的依据，桶的个数有用户设置，这里以用户表中的id字段来划分桶，划分4个桶。Hive会计算分桶列的hash值再以桶的个数取模来计算某条记录属于那个桶。

分桶的优点在于，将数据大致平均的、随机的放入多个桶中，这样方便对海量的数据做抽样调查、分析。

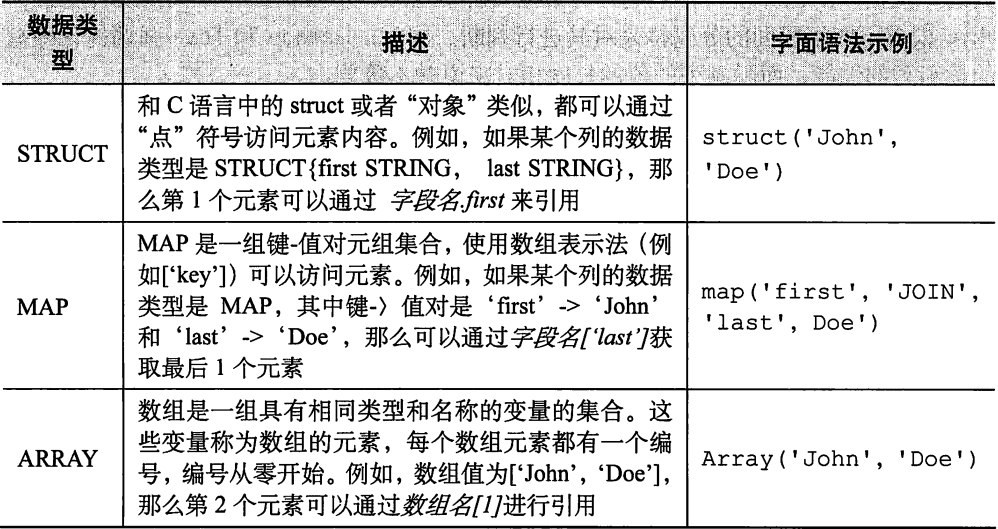
## 2.4 Hive数据类型

### 2.4.1基本数据类型

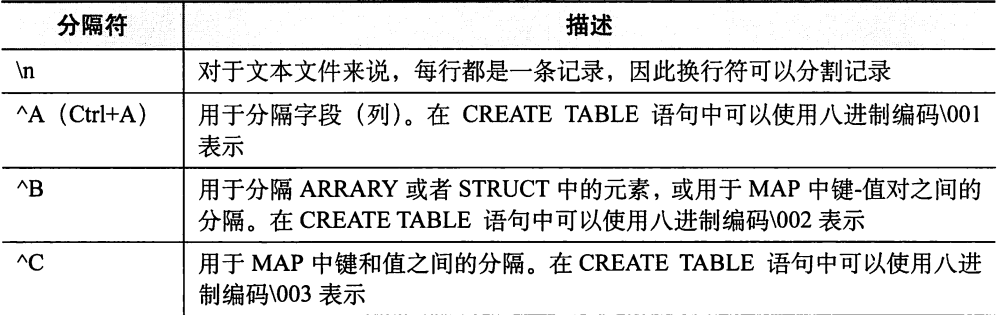




### 2.4.2复杂数据类型



### 2.4.3文本文件数据编码（分隔符）



## 2.5 Hive在HDFS上的文件结构

|  |  |  |
| --- | --- | --- |
|  | **HDFS** | **mysql数据库** |
| 文件结构实例 | hdfs:///sourcedb/log/2015-03-01.log | soucedb数据库中有log表 |
| 文件结构 | 文件夹1/文件夹2/文件 | 库/表/表中的数据 |

|  |  |
| --- | --- |
| **HDFS** | **mysql数据库** |
| 文件夹1 | 库 |
| 文件夹2 | 表 |
| 文件 | 表中的数据 |

Hive元数据的本质：

Hive库表和HDFS上的文件结构的对应关系

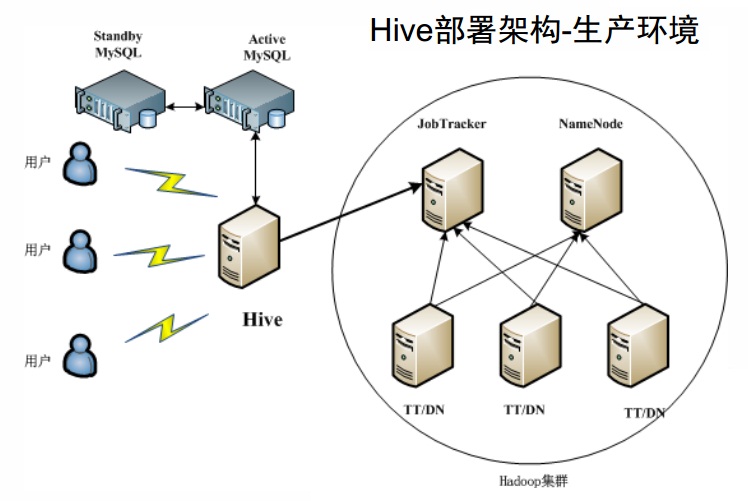
不配置的话默认在HDFS上的以下目录：hdfs:///user/hive/warehouse/

Hive元数据的存储：

存储在关系型数据库中。默认存Derby 内置数据库 2mb，内嵌在Hive中。

实际工作中使用的最多是MySQL。

## 2.6Hive的生产环境架构



中间件

# 3. Hive元数据库安装和配置（以mysql为例）

## 3.1 Mysql的安装（即mysql服务的安装）

如果操作系统能够联网（或者有本地yum仓库）且安装有yum，可直接使用yum进行安装，比较方便，命令如下：

# >yum install mysql-server

如果操作系统无法联网（也没有本地yum仓库）则可以使用rpm包的方式进行安装，需要注意的是，由于操作系统上所含有的软件多少不确定，所以安装过程中可能需要解决包依赖问题，在此不再赘述。

## 3.2 Mysql服务启动

安装完成后需要启动mysql的服务才能进行给超级用户设置密码等操作，注意:mysql服务的名字是mysqld。启动命令如下：

# >service mysqld start

查看mysql服务的状态命令如下：

# >service mysqld status

## 3.3给root用户添加密码和任意主机访问的权限

首先进入mysql：

# >mysql -u root

然后给root用户添加密码（现在已经进入mysql控制台，直接执行下列sql即可）：

mysql >update mysql.user set password=PASSWORD ('root') where User='root';

添加完成后须刷新权限才能生效：

mysql >flush privileges;

查询用户所拥有的访问权限信息的命令如下：

mysql >select host,user,password from mysql.user;

可以看到，初始情况下：root用户只能通过mysql服务所在的机器本身来进行访问

有时我们因为种种原因无法直接登录到mysql服务器上操作，所以我们可以通过修改配置以达到root用户可以使用任意主机访问数据库，这样做的好处是我们可以远程进行数据操作，还可以使用可视化工具提高操作效率。命令如下：

mysql >update mysql.user set host = '%' where user = 'root' and host = '127.0.0.1';

操作完成后须刷新权限才能生效：

mysql >flush privileges;

## 3.4自己创建hive库并设置编码为latin1(默认编码)

命令如下：

mysql >create database hive DEFAULT CHARACTER SETlatin1;

或直接使用如下简化命令：

mysql >create database hive;

# 4. Hive安装配置

## 4.1 Hive和hadoop之间的联系方式

hive通过/etc/profile文件中配置的HADOOP\_HOME跟hadoop进行沟通

所以，hive的配置文件中跟hadoop没有联系

## 4.2 Hive安装的位置

可以在任意配置了HADOOP\_HOME的机器上安装

## 4.3解压Hive压缩包

例如我们将从Apache hive官网上下载下来的匹配操作系统和hadoop版本的gz压缩包放置在/home/bigdata目录下。

我们首先进入该目录：

# >cd /home/bigdata

将压缩文件进行解压：

# >tar –xzvf apache-hive-1.1.0-bin.tar.gz

## 4.4设置HIVE\_HOME

配置HIVE\_HOME，其好处在于我们在任何目录都可以很方便的启动hiveCLI：

在/etc/profile 文件中添加如下信息：

export HIVE\_HOME=/home/bigdata/hive

export PATH=$PATH:$HIVE\_HOME/bin:

hive通过/etc/profile文件中配置的HADOOP\_HOME跟hadoop进行沟通

所以，hive的配置文件中跟hadoop没有联系

## 4.5Hive配置文件

进入配置文件目录：

# >cd /home/bigdata/hive/conf

将hive-default.xml.template文件拷贝并重命名成hive-site.xml：

# >mv hive-default.xml.template hive-site.xml

清空文件中<configuration></configuration>之间的内容并加入下列内容：

<property>

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

<value>jdbc:mysql://slave1.com:3306/hive?characterEncoding=UTF-8</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>

将mysql的驱动jar包添加到hive的lib目录下

## 4.6 Hive多种配置方式

### 4.6.1 配置文件

hive-site.xml

Hadoop中的配置文件

### 4.6.2 命令行参数

启动Hive时，可以在命令行添加**-hiveconf param=value**，但这一设定只对本次启动的Session有效

### 4.6.3参数声明

在HiveQL前用set关键字进行参数设定，这一设定的作用域也是Session级的，也是单次有效

**注意：以上三种设定方式的优先级依次递增。即参数声明覆盖命令行参数，命令行参数覆盖配置文件设定**

## 4.6 启动Hive

直接在命令行中输入hive即可：

# >hive

注意：如果Hive版本比较新，hadoop版本稍旧的话可能会报jline相关的错，只需要将

**\*如果启动Hive报jline错误，需要Hive中jline.jar与Hadoop中的jline.jar版本不同，启动hive出错。如果启动hive不出错可不用做。**

**错误信息：**

“Found class jline.Terminal, but interface was expected”

Hive的lib目录下的jline-2.12.jar包覆盖YARN下的jline包即可

Hive的jline包地址 ：

/home/bigdata/hive/lib/jline-2.12.jar

YARN的jline包地址：

/home/bigdata/hadoop/share/hadoop/yarn/lib/jline-xx.jar

# >rmr –rf /home/bigdata/hadoop/share/hadoop/yarn/lib/jline-xx.jar

# >cp /home/bigdata/hive/lib/jline-2.12.jar /home/bigdata/hadoop/share/hadoop/yarn/lib

## 4.7 Hive在HDFS上的默认存储路径

Hive的数据都是存储在HDFS上的，默认有一个根目录，在hive-site.xml中，由参数hive.metastore.warehouse.dir指定。默认值为/user/hive/warehouse. （一般不需要）

# Hive Query Lanaguage语法

## 5.1HQL基本语法

### 5.1.1Hive脚本注释

hive >--注释内容

### 5.1.2 Hive的语法规则

Hql语法和数据库表的名字都不区分大小写

Mysql中的表名区分大小写

### 5.1.3 查看所有库

hive >SHOW DATABASES;

### 5.1.4 使用（进入）某个库

hive >use 库名;

### 5.1.5 查看某个库下的所有表

首先进入到某个库中，然后执行下列命令

# hive >SHOW TABLES;

## 5.2Hive数据库相关语法

### 5.2.1 创建

CREATE DATABASE [IF NOT EXISTS] 数据库名称 [COMMENT ‘注释’];

hive >CREATE DATABASE IF NOT EXISTS log;

hive >CREATE DATABASE result COMMENT ‘this is result db’；

注意：这里COMMENT语句两边加上中括号表示这是一个可选项，即注释可以加也可不加

同样的一个表或者表中的一个列也可以以同样的方式进行添加注释，但是同样都是可选的

注意：同样IF NOT EXISTS关键字也是可选的，但是添加之后如果要创建的表已经存在了就不再创建，也不会报错，

但是如果不加的话，会报如下错误信息：

Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. Database log already exists

### 5.2.2查询数据库信息

DESCRIBE DATABASE [EXTENDED] 数据库名称；

Desc database [EXTENDED] 数据库名称；

注意：添加EXTENDED关键字可以查看更多信息

### 5.2.3 创建数据库的本质

HDFS上Hive的根目录下新建一个 数据库名.db 的文件夹

注意：默认的default库没有名字，就在根目录下

### 5.2.4 记录Hive库信息的元数据表

Hive元数据库中的DBS表

### 5.2.5Hive不同数据库之间的切换

hive > use 库名；

### 5.2.6Hive数据库删除

drop database if exists 库名；

注意：加上if exists 可以防止因库不存在而抛出警告信息，错误信息如下：

FAILED: SemanticException [Error 10072]: Database does not exist: d

注意：默认情况下Hive不允许用户删除一个包含有表（空表也算有表）的数据库，这时候我们需要在结尾加上关键字**cascade**。即：

hive >drop database if exists 库名 cascade；

如果删除一含有表的库报错信息如下：

FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. InvalidOperationException(message:Database log is not empty. One or more tables exist.)

## 5.3Hive表相关语法

### 5.3.1 创表示例如下

hive >CREATE TABLE log(

id string COMMENT 'this is id column',

phonenumber bigint,

mac string,

ip string,

url string,

tiele string,

colum1 string,

colum2 string,

colum3 string,

upflow int,

downflow int

)

COMMENT 'this is log table'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

LINES TERMINATED BY '\n'

stored as textfile;

**注意：行与行之间的分隔符目前只支持换行符”\n”，所以一般都省略不写。**

### 5.3.3创建表的本质：

在表所在的库的目录下新建一个以表名为名字的文件夹

### 5.3.4记录Hive中的表的元数据表

Hive元数据库中的TBLS表、COLUMNS\_V2、SDS等表

### 5.3.5 Hive表三种添加数据方式

* 1.从本地往Hive的log表中添加数据:

hive >LOAD DATA LOCAL INPATH ‘/home/data/flow.log’ INTO TABLE log1;

* 2.从HDFS上往Hive的log表中覆盖数据:

hive >LOAD DATA INPATH ‘/input/flow.log’OVERWRITE INTO TABLE log2\_tmp;

* 3.向log表中添加结果集数据：

hive >INSERT INTO TABLE log3

select \* from log2\_tmp;

注意：如果通过insert的方式覆盖，需要把into直接换成overwrite，跟load方式不同

* 向Hive表中追加或覆盖数据语法

LOAD DATA [LOCAL] INPATH ‘本地或HDFS地址’[OVERWRITE] INTO TABLE ‘表名’;

**LOCAL关键字**

表示使用本地文件系统(会将数据拷贝一份到hive目录中)，不写表示使用HDFS（会从HDFS上移动到hive的目录中）

**OVERWRITE关键字**

表示本次添加将覆盖表中以前的数据，不写表示将进行数据追加

**思考？**

**既然Hive从HDFS上加载数据就是一个转移文件的过程，那么我们创建表的时候可以直接指定一个HDFS上的目录吗？**

Hive创建日志表1（指定数据在HDFS上的目录）

注意：需要提前在HDFS上的 /source 目录下准备好一份日志数据

hive >CREATE TABLE IF NOT EXISTS **log1**(

id string,

phonenumber bigint,

mac string,

ip string,

url string,

tiele string,

colum1 string,

colum2 string,

colum3 string,

upflow int,

downflow int)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

stored as textfile

**location '/source'**;

此时查询表中的内容

hive >select \* from **log1;**

总结

创建表的时候可以直接指定一个HDFS上的目录

注意：这里的IF NOT EXISTS 有个陷阱，就算新创建的表的结构跟原来同名的表结构不一样，也不新建表

**思考？**

**Hive加载HDFS上的数据可以看到表中有内容，直接指定一个HDFS目录也可看到内容，那么假如我们指定一个没有数据的目录，或者加载一个不同格式的文件，会产生什么结果呢？**

### 5.3.6Hive的数据管理模式

关系型数据库的数据管理模式：**写时模式**，即添加数据的时候就检查数据格式，格式错误无法插入。

Hive的数据管理模式：**读时模式**，即读取数据的时候才检查数据格式，不匹配返回null。

即：Hive加载数据的时候并不操作或处理数据

所以：Hive可以在建表的时候就指定数据所在的目录

本质：Hive的建表语句就是对文件中的数据的结构的一种描述

原因：Hive的作用是**分析海量**数据（不需要精确管理）

### 5.3.7 Hive删除表

hive >drop table log;

注：数据表删除后，元数据也被删除，HDFS上的需要分析的数据和表所在目录也被删除

### 5.3.8 EXTERNAL类型日志表extlog删除后效果

1. 创表语句

注意：需要提前在HDFS上的 /source 目录下准备好一份日志数据

hive >CREATE **EXTERNAL** TABLE IF NOT EXISTS **extlog**(

id string,

phonenumber bigint,

mac string,

ip string,

url string,

tiele string,

colum1 string,

colum2 string,

colum3 string,

upflow int,

downflow int

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

stored as textfile

**location '/input/source'**;

2.删除extlog表命令

hive >drop table extlog;

注：元数据被删除掉，但是HDFS上的需要分析的数据被保留了

3.总结

这种建表时添加EXTERNAL关键字的表称之为外部表，相应的，没有添加关键字的表为内部表或者管理表（因为Hive元数据中记录的内部表的类型为MANAGED\_TABLE）

外部表的特点是删除时只删除元数据，内部表将元数据和需要分析的HDFS上的数据也删除了。

### 5.3.9Hive表类型总结

|  |  |  |
| --- | --- | --- |
|  | **外部表** | **内部表（管理表）** |
| 元数据库中的类型 | EXTERNAL \_TABLE | MANAGED\_TABLE |
| 关键字 | EXTERNAL | 无 |
| 建表示例 | create exteranl table t ; | create table t ; |
| 删除数据时元数据 | 被删除 | 被删除 |
| 删除数据时表中的数据 | 被保留 | 被删除 |
| 适用场景 | 与其他应用程序共享数据 | 自己管理数据的生命周期 |
| 示例 | 待分析日志表 源数据日志 | 日志分析结果表 中间表 |

**注意：**

**内部表建表时如果需要指定位置请确保指定的位置不被其他人使用，因为内部表删除的时候会将目录和数据同时删除，一般使用内部表时都不指定数据位置。**

内部表和外部表最大的区别

内部表DROP时候会删除HDFS上的数据;

外部表DROP时候不会删除HDFS上的数据;

内部表适用场景：

Hive中间表、结果表、一般不需要从外部（如本地文件、HDFS上load数据）的情况。

外部表适用场景：

源表，需要定期将外部数据映射到表中。

我们的使用场景：

每天将收集到的网站日志定期流入HDFS文本文件，一天一个目录；

在Hive中建立外部表作为源表，通过添加分区的方式，将每天HDFS上的原始日志映射到外部表的天分区中；

在外部表（原始日志表）的基础上做大量的统计分析，用到的中间表、结果表使用内部表存储，数据通过SELECT+INSERT进入内部表。

### 5.3.10Hive内部表和外部表示例

--准备工作：确保hdfs:///source目录下有日志文件

--下面使用内部表外部表进行分析处理

--创建日志库（日志层）

CREATE DATABASE IF NOT EXISTS log;

--创建日志表

--使用外部表同时指定数据位置（即：数据存放在hdfs:///source）

--优点：不用加载数据（即不用移动数据），数据还可以被其他人员或应用程序使用

hive >CREATE EXTERNAL TABLE IF NOT EXISTS log.log(

id string,

phonenumber bigint,

mac string,

ip string,

url string,

tiele string,

colum1 string,

colum2 string,

colum3 string,

upflow int,

downflow int

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

stored as textfile

location '/source';

--创建结果库（结果层）

CREATE DATABASE IF NOT EXISTS result;

--创建流量统计表

--使用内部表，不指定数据位置（即默认存hive根目录下的result库下的flowsum表中hdfs:///user/hive/warehouse/result.db/）

--优点：Hive能够完全控制数据

hive >CREATE TABLE IF NOT EXISTS result.flowsum(

phonenumber bigint,

flowsum bigint

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

stored as textfile;

--将查询结果插入到结果表中（在原有结果的基础上追加）

INSERT INTO TABLE result.flowsum

select phonenumber,sum(upflow)+sum(downflow) as flowsum

from log.log

group by phonenumber;

### 5.3.11 Hive内部表分区

1. 先看一个实例

hive >CREATE TABLE IF NOT EXISTS part(

id string,

phonenumber bigint,

mac string,

ip string,

url string,

tiele string,

colum1 string,

colum2 string,

colum3 string,

upflow int,

downflow int

)

PARTITIONED BY (year int,month int,day int)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

stored as textfile;

2.向表中加载数据并进入HDFS查看现象

hive >load data local inpath '/root/Desktop/flow.log'

into table part

partition (year=2015,month=5,day=24);

3. 表分区

一个所谓分区的本质就是一个确定的目录

如果我们的日志文件因为时间的累积产生了多个，分别存放在不同的文件夹下，那么查询的时候如果可以指定文件夹查询效率就会很高，这就是hive表分区所要做的事情

4. 文件夹命名规则

分区字段=分区值 例如：date=2015

5. 查看表中的分区

SHOW PARTITIONS 表名;

### 5.3.12 Hive外部表分区

hive >CREATE EXTERNAL TABLE IF NOT EXISTS extpart(

id string,

phonenumber bigint,

mac string,

ip string,

url string,

tiele string,

colum1 string,

colum2 string,

colum3 string,

upflow int,

downflow int

)

PARTITIONED BY (year int,month int,day int)

location '/extpartition';

外部表添加分区信息(可以更加方便的分析我们熟悉的文件结构）

hive >ALTER TABLE extpart

hive >ADD PARTITION (year=2015, month=05,day=26)

hive >LOCATION 'hdfs:///extpartition/2015/05/26';

### 5.3.13 Hive表动态和静态分区

* **静态分区插入**

即：插入的时候就指定分区

静态分区示例

hive >load data local inpath '/root/Desktop/flow.log'

hive >into table part

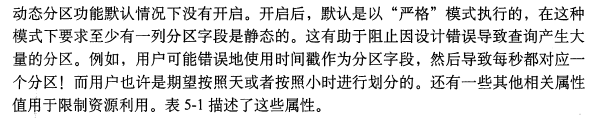
hive >partition (year=2015,month=5,day=24);

* **动态分区插入**

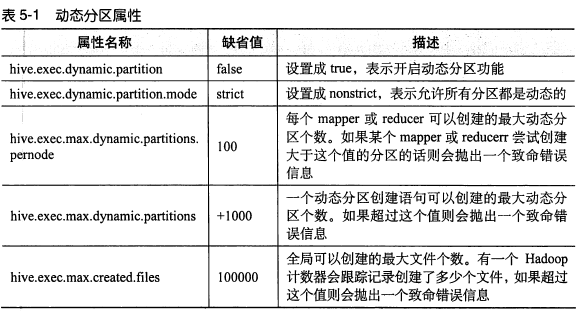
即：查询的结果集插入到分区表中，结果集中包含分区字段

动态分区示例

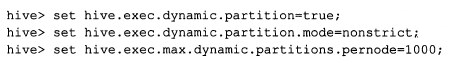
动态分区插入前的准备



动态分区属性



设置分区属性保证分区可用



动态分区插入示例

--创建pv表

hive >create table pv(pv int,year int,month int,day int)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

location '/pv';

--创建带有分区的pvpart表

hive >create table pvpart(pv int)

PARTITIONED BY (year int,month int,day int)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

--设置Hive动态分区属性

hive >set hive.exec.dynamic.partition=true;

hive >set hive.exec.dynamic.partition.mode=nonstrict;

hive >set hive.exec.max.dynamic.partitions.pernode=100;

--动态分区插入

hive >insert into table pvpart

hive >partition (year,month,day)

hive >select pv,year,month,day

hive >from pv;

### 5.3.14 表修改

1. ALTER关键字

Hive表信息修改所使用的是alter，注意不是js中的alert，alert是警告的意思，alter是修改的意思

注意：这里修改的都是元数据信息、语法部分使用大写表示

2. 修改表名

**首先创建一个测试用表**

hive >create table test (

Id int,

name string,

class string);

### 5.3.15查看表信息

hive >DESC [EXTENDED] test;

注意:EXTENDED关键字的本意是扩展的，所以添加EXTENDED关键字可获得更多表的信息

### 5.3.16修改表名

hive >ALTER TABLE test RENAME TO test1;

注意：RENAME后面有TO关键字，不要忘了写

**修改列id 为myid 且类型更改为string且列位置位于class之后**

hive >ALTER TABLE test1

hive >CHANGE COLUMN id myid string COMMENT 'change id to myid'

hive >AFTER class;

注意： COLUMN关键字可以不写，同样的COMMENT为可选项，就算只改列的数据类型也需要先写旧列名然后再写一遍旧列名

注意：如果想让myid位于首位，只需要将**AFTER class语句**换成FIRST关键字即可，也就是说FIRST关键字之后不需要跟列名了

### 5.3.17增加列

hive >ALTER TABLE test1 ADD COLUMN**S**(

score int COMMENT 'this is score',

sex string);

注意：这里的关键字是COLUMNS（不能省略），表示可以插入多个列，同样的COMMENT也是可选项

### 5.3.17 删除并添加列

hive >ALTER TABLE test1 REPLACE COLUMNS(

score int,

height int);

注意：此操作会删除原来表中所有的列并将替换语句中所有的列添加到表中，

所以建议先在文本编辑器里整体编辑并仔细检查后再执行

**注意：这里的操作不会影响到分区列**

### 5.3.19 修改表属性

--将内部表转为外部表

查看test1表属性,查看是否是内部表或外部表

hive >DESC EXTENDED test1;

--将内部表转为外部表

hive >ALTER TABLE test1 SET TBLPROPERTIES(

'EXTERNAL' = 'TRUE');

--将外部表转为内部表

hive >ALTER TABLE test1 SET TBLPROPERTIES(

'EXTERNAL' = 'false');

注意： EXTERNAL关键字必须大写

### 5.3.20 显示表中存在的所有分区

Hive>show partitions table;

### 5.3.21 表总结

CREATE EXTERNAL TABLE t\_lxw1234 (

id INT,

ip STRING COMMENT ‘访问者IP’,

avg\_view\_depth DECIMAL(5,1),

bounce\_rate DECIMAL(6,5)

) COMMENT ‘lxw的大数据田地-lxw1234.com’

PARTITIONED BY (day STRING)

ROW FORMAT DELIMITED ‘\n’

FIELDS TERMINATED BY ‘,’

STORED AS textfile

LOCATION ‘hdfs://cdh5/tmp/lxw1234/';

关键字EXTERNAL：

表示该表为外部表，如果不指定EXTERNAL关键字，则表示内部表

关键字COMMENT

为表和列添加注释

关键字PARTITIONED BY

表示该表为分区表，分区字段为day,类型为string

关键字ROW FORMAT DELIMITED

指定表的分隔符，通常后面要与以下关键字连用：

LINES TERMINATED BY ‘\n’ //指定行分隔符

FIELDS TERMINATED BY ‘,’ //指定每行中字段分隔符为逗号

COLLECTION ITEMS TERMINATED BY ‘,’ //指定集合中元素之间的分隔符

MAP KEYS TERMINATED BY ‘:’ //指定数据中Map类型的Key与Value之间的分隔符

举个例子：

create table score(name string, score map<string,int>)

ROW FORMAT DELIMITED ‘\n’

FIELDS TERMINATED BY ‘\t’

COLLECTION ITEMS TERMINATED BY ‘,’

MAP KEYS TERMINATED BY ‘:';

要加载的文本数据为：

biansutao ‘数学':80,’ 语文':89,’英语':95

jobs ‘语文':60,’数学':80,’英语':99

关键字STORED AS

指定表在HDFS上的文件存储格式，可选的文件存储格式有：

TEXTFILE //文本，默认值

SEQUENCEFILE // 二进制序列文件

RCFILE //列式存储格式文件 Hive0.6以后开始支持

ORC //列式存储格式文件，比RCFILE有更高的压缩比和读写效率，Hive0.11以后开始支持

PARQUET //列出存储格式文件，Hive0.13以后开始支持

关键词LOCATION

指定表在HDFS上的存储位置。

### 5.4 Hive所支持的表操作类型

Hive是一个海量数据分析工具，所以Hive支持以下数据操作：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **增** | **删** | **改** | **查** |
| **支持** | 1.直接往表中加载一个文件中的所有数据  2.往表中加载一个结果集 | 删除整个表 | 表结构 | 支持（最主要） |
| **不支持** | Insert into values 方式添加数据 | 按条件删除 | 数据 | 无 |

### 5.5 Hive的日志

Hive的日志一般存放在操作系统本地的/tmp/${user.name}/hive.log中，该文件路径由

$HIVE\_HOME/conf/hive-log4j.properties中指定，可自己修改该路径。

# 6. Hive的数据管理模式

## 6.1关系型数据库的数据管理模式

写时模式（添加数据的时候就检查数据格式，格式错误无法插入）

## 6.2 Hive的数据管理模式

读时模式（读取数据的时候才检查数据格式，不匹配返回null）

即：hive加载数据的时候并不操作或处理数据

所以：hive可以在建表的时候就指定数据所在的目录

本质：hive的建表语句就是对文件中的数据的结构的一种描述

原因：hive的作用是**分析海量**数据（不需要精确管理）

# 7 Hive高级配置总结

# ------------------配置相关--------------------------

0. 配置存储

在$HOME目录下的.hiverc文件中

1. 在提示符中显示数据库名

set hive.cli.print.current.db=true;

2. 优先使用本地模式执行

set hive.exec.mode.loacl.auto=true;

3. 打印列名

set hive.cli.print.header=true;

4. （非）严格模式

set hive.mapred.mode=strict;

set hive.mapred.mode=nonstrict;

5. 开启动态分区

set hive.exec.dynamic.partition=true;

# --------------动态分区调整--------------------

-- 设置动态分区模式

set hive.exec.dynamic.partition.mode=strict

-- 设置总的动态分区个数

set hive.exec.max.dynamic.partitions=300000

-- 设置每个节点上动态分区个数

set hive.exec.max.dynamic.partitions.pernode=10000

9. 设置全局可以产生文件的个数

set hive.exec.max.created.files=100000;

10.开启map site join

set hive.auto.convert.join=true;

11.设置小表大小（字节）

set hive.mapjoin.smalltable.filesize=25000000

12.强制将数据按照桶结构定义来插入数据

set hive.enforce.bucketing=true

13.limit优化

-- 启用limit优化

set hive.limit.optimize.enable=true;

-- 限制从最大多少条数据中进行limit

set hive.limit.row.max.size=10000;

-- 限制最多遍历的文件个数

set hive.limit.optimize.limit.file=10;

14. 压缩

--开启中间压缩（即map到reduce之间的数据压缩）

set hive.exec.compress.intermediate=true;

-- 开启hadoop中间压缩（即map到reduce之间的数据压缩）

set mapred.compress.map.output=true;

-- 开启hive最终压缩（即reduce输出的数据压缩）

set hive.exec.compress.output=true;

4. 数据仓库的存储地址

hive-default.xml中，不一般不同

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/user/hive/warehouse</value>

<description>location of default database for the warehouse</description>

</property>

# ------------------变量定义-------------------------

1、定义

hive --hivevar myname=wufan;

2、获取变量值

${hivevar:myname}

3、显示变量

set myname;

set hivevar:myname;

4、修改（或赋值）变量

set hivevar:myname=shawn;

# ------------------使用变量-------------------------

# 方式一

-- my.sh

#!/bin/bash

tablename="student"

limitcount="8"

hive -S -e "use test; select \* from ${tablename} limit ${limitcount};"

# 方式二

-- my.sh

#! /bin/bash

hive --hivevar tableName=T1 --hivevar tableField1=F1 --hivevar tableField2=F2 -f my.hql

-- my.hql

use mydb;

create table if not exists ${hivevar:tableName}

(

${hivevar:tableField1} string,

${hivevar:tableField2} string

);

# ------------------执行相关-------------------------

1. 执行外部的脚本

hive -f myscript.hql

2. 直接执行脚本

hive -e "select name from user"

3. 静默执行

hive -S myscript.hql