尚硅谷大数据技术之Hive

(作者：尚硅谷大数据研发部)

版本：V1.3

# 第1章 Hive基本概念

## 1.1 什么是Hive

Hive：由Facebook开源用于解决海量结构化日志的数据统计。

Hive是基于Hadoop的一个数据仓库工具，可以将结构化的数据文件映射为一张表，并提供类SQL查询功能。

本质是：将HQL转化成MapReduce程序

 1）Hive处理的数据存储在HDFS

2）Hive分析数据底层的实现是MapReduce

3）执行程序运行在Yarn上

## 1.2 Hive的优缺点

### 1.2.1 优点

1. 操作接口采用类SQL语法，提供快速开发的能力（简单、容易上手）。
2. 避免了去写MapReduce，减少开发人员的学习成本。
3. Hive的执行延迟比较高，因此Hive常用于数据分析，对实时性要求不高的场合。
4. Hive优势在于处理大数据，对于处理小数据没有优势，因为Hive的执行延迟比较高。
5. Hive支持用户自定义函数，用户可以根据自己的需求来实现自己的函数。

### 1.2.2 缺点

1．Hive的HQL表达能力有限

（1）迭代式算法无法表达

（2）数据挖掘方面不擅长

2．Hive的效率比较低

（1）Hive自动生成的MapReduce作业，通常情况下不够智能化

（2）Hive调优比较困难，粒度较粗

## 1.3 Hive架构原理



图6-1 Hive架构原理

1．用户接口：Client

CLI（hive shell）、JDBC/ODBC(java访问hive)、WEBUI（浏览器访问hive）

2．元数据：Metastore

元数据包括：表名、表所属的数据库（默认是default）、表的拥有者、列/分区字段、表的类型（是否是外部表）、表的数据所在目录等；

默认存储在自带的derby数据库中，推荐使用MySQL存储Metastore

3．Hadoop

使用HDFS进行存储，使用MapReduce进行计算。

4．驱动器：Driver

（1）解析器（SQL Parser）：将SQL字符串转换成抽象语法树AST，这一步一般都用第三方工具库完成，比如antlr；对AST进行语法分析，比如表是否存在、字段是否存在、SQL语义是否有误。

（2）编译器（Physical Plan）：将AST编译生成逻辑执行计划。

（3）优化器（Query Optimizer）：对逻辑执行计划进行优化。

（4）执行器（Execution）：把逻辑执行计划转换成可以运行的物理计划。对于Hive来说，就是MR/Spark。



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。

## 1.4 Hive和数据库比较

由于 Hive 采用了类似SQL 的查询语言 HQL(Hive Query Language)，因此很容易将 Hive 理解为数据库。其实从结构上来看，Hive 和数据库除了拥有类似的查询语言，再无类似之处。本文将从多个方面来阐述 Hive 和数据库的差异。数据库可以用在 Online 的应用中，但是Hive 是为数据仓库而设计的，清楚这一点，有助于从应用角度理解 Hive 的特性。

### 1.4.1 查询语言

由于SQL被广泛的应用在数据仓库中，因此，专门针对Hive的特性设计了类SQL的查询语言HQL。熟悉SQL开发的开发者可以很方便的使用Hive进行开发。

### 1.4.2 数据存储位置

Hive 是建立在 Hadoop 之上的，所有 Hive 的数据都是存储在 HDFS 中的。而数据库则可以将数据保存在块设备或者本地文件系统中。

### 1.4.3 数据更新

由于Hive是针对数据仓库应用设计的，而数据仓库的内容是读多写少的。因此，Hive中不建议对数据的改写，所有的数据都是在加载的时候确定好的。而数据库中的数据通常是需要经常进行修改的，因此可以使用 INSERT INTO …  VALUES 添加数据，使用 UPDATE … SET修改数据。

### 1.4.4 索引

Hive在加载数据的过程中不会对数据进行任何处理，甚至不会对数据进行扫描，因此也没有对数据中的某些Key建立索引。Hive要访问数据中满足条件的特定值时，需要暴力扫描整个数据，因此访问延迟较高。由于 MapReduce 的引入， Hive 可以并行访问数据，因此即使没有索引，对于[大数据](http://lib.csdn.net/base/hadoop)量的访问，Hive 仍然可以体现出优势。数据库中，通常会针对一个或者几个列建立索引，因此对于少量的特定条件的数据的访问，数据库可以有很高的效率，较低的延迟。由于数据的访问延迟较高，决定了 Hive 不适合在线数据查询。

### 1.4.5 执行

Hive中大多数查询的执行是通过 Hadoop 提供的 MapReduce 来实现的。而数据库通常有自己的执行引擎。

### 1.4.6 执行延迟

Hive 在查询数据的时候，由于没有索引，需要扫描整个表，因此延迟较高。另外一个导致 Hive 执行延迟高的因素是 MapReduce框架。由于MapReduce 本身具有较高的延迟，因此在利用MapReduce 执行Hive查询时，也会有较高的延迟。相对的，数据库的执行延迟较低。当然，这个低是有条件的，即数据规模较小，当数据规模大到超过数据库的处理能力的时候，Hive的并行计算显然能体现出优势。

### 1.4.7 可扩展性

由于Hive是建立在Hadoop之上的，因此Hive的可扩展性是和Hadoop的可扩展性是一致的（世界上最大的Hadoop 集群在 Yahoo!，2009年的规模在4000 台节点左右）。而数据库由于 ACID 语义的严格限制，扩展行非常有限。目前最先进的并行数据库 [Oracle](http://lib.csdn.net/base/oracle) 在理论上的扩展能力也只有100台左右。

### 1.4.8 数据规模

由于Hive建立在集群上并可以利用MapReduce进行并行计算，因此可以支持很大规模的数据；对应的，数据库可以支持的数据规模较小。

# 第2章 Hive安装

## 2.1 Hive安装地址

1．Hive官网地址

http://hive.apache.org/

2．文档查看地址

https://cwiki.apache.org/confluence/display/Hive/GettingStarted

3．下载地址

http://archive.apache.org/dist/hive/

4．github地址

https://github.com/apache/hive

## 2.2 Hive安装部署

1．Hive安装及配置

（1）把apache-hive-1.2.1-bin.tar.gz上传到linux的/opt/software目录下

（2）解压apache-hive-1.2.1-bin.tar.gz到/opt/module/目录下面

[atguigu@hadoop102 software]$ tar -zxvf apache-hive-1.2.1-bin.tar.gz -C /opt/module/

（3）修改apache-hive-1.2.1-bin.tar.gz的名称为hive

[atguigu@hadoop102 module]$ mv apache-hive-1.2.1-bin/ hive

（4）修改/opt/module/hive/conf目录下的hive-env.sh.template名称为hive-env.sh

[atguigu@hadoop102 conf]$ mv hive-env.sh.template hive-env.sh

（5）配置hive-env.sh文件

（a）配置HADOOP\_HOME路径

export HADOOP\_HOME=/opt/module/hadoop-2.7.2

（b）配置HIVE\_CONF\_DIR路径

export HIVE\_CONF\_DIR=/opt/module/hive/conf

2．Hadoop集群配置

（1）必须启动hdfs和yarn

[atguigu@hadoop102 hadoop-2.7.2]$ sbin/start-dfs.sh

[atguigu@hadoop103 hadoop-2.7.2]$ sbin/start-yarn.sh

（2）在HDFS上创建/tmp和/user/hive/warehouse两个目录并修改他们的同组权限可写

[atguigu@hadoop102 hadoop-2.7.2]$ bin/hadoop fs -mkdir /tmp

[atguigu@hadoop102 hadoop-2.7.2]$ bin/hadoop fs -mkdir -p /user/hive/warehouse

[atguigu@hadoop102 hadoop-2.7.2]$ bin/hadoop fs -chmod g+w /tmp

[atguigu@hadoop102 hadoop-2.7.2]$ bin/hadoop fs -chmod g+w /user/hive/warehouse

3．Hive基本操作

（1）启动hive

[atguigu@hadoop102 hive]$ bin/hive

（2）查看数据库

hive> show databases;

（3）打开默认数据库

hive> use default;

（4）显示default数据库中的表

hive> show tables;

（5）创建一张表

hive> create table student(id int, name string);

（6）显示数据库中有几张表

hive> show tables;

（7）查看表的结构

hive> desc student;

（8）向表中插入数据

hive> insert into student values(1000,"ss");

（9）查询表中数据

hive> select \* from student;

（10）退出hive

hive> quit;

## 2.3 将本地文件导入Hive案例

需求

将本地/opt/module/datas/student.txt这个目录下的数据导入到hive的student(id int, name string)表中。

1．数据准备

在/opt/module/datas这个目录下准备数据

（1）在/opt/module/目录下创建datas

[atguigu@hadoop102 module]$ mkdir datas

（2）在/opt/module/datas/目录下创建student.txt文件并添加数据

[atguigu@hadoop102 datas]$ touch student.txt

[atguigu@hadoop102 datas]$ vi student.txt

1001 zhangshan

1002 lishi

1003 zhaoliu

注意以tab键间隔。

2．Hive实际操作

（1）启动hive

[atguigu@hadoop102 hive]$ bin/hive

（2）显示数据库

hive> show databases;

（3）使用default数据库

hive> use default;

（4）显示default数据库中的表

hive> show tables;

（5）删除已创建的student表

hive> drop table student;

（6）创建student表, 并声明文件分隔符’\t’

hive> create table student(id int, name string) ROW FORMAT DELIMITED FIELDS TERMINATED

BY '\t';

（7）加载/opt/module/datas/student.txt 文件到student数据库表中。

hive> load data local inpath '/opt/module/datas/student.txt' into table student;

（8）Hive查询结果

hive> select \* from student;

OK

1001 zhangshan

1002 lishi

1003 zhaoliu

Time taken: 0.266 seconds, Fetched: 3 row(s)

3．遇到的问题

再打开一个客户端窗口启动hive，会产生java.sql.SQLException异常。

Exception in thread "main" java.lang.RuntimeException: java.lang.RuntimeException:

Unable to instantiate

org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClient

at org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:522)

at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:677)

at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:621)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:606)

at org.apache.hadoop.util.RunJar.run(RunJar.java:221)

at org.apache.hadoop.util.RunJar.main(RunJar.java:136)

Caused by: java.lang.RuntimeException: Unable to instantiate org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClient

at org.apache.hadoop.hive.metastore.MetaStoreUtils.newInstance(MetaStoreUtils.java:1523)

at org.apache.hadoop.hive.metastore.RetryingMetaStoreClient.<init>(RetryingMetaStoreClient.java:86)

at org.apache.hadoop.hive.metastore.RetryingMetaStoreClient.getProxy(RetryingMetaStoreClient.java:132)

at org.apache.hadoop.hive.metastore.RetryingMetaStoreClient.getProxy(RetryingMetaStoreClient.java:104)

at org.apache.hadoop.hive.ql.metadata.Hive.createMetaStoreClient(Hive.java:3005)

at org.apache.hadoop.hive.ql.metadata.Hive.getMSC(Hive.java:3024)

at org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:503)

... 8 more

原因是，Metastore默认存储在自带的derby数据库中，推荐使用MySQL存储Metastore;

## 2.4 MySql安装

### 2.4.1 安装包准备

1．查看mysql是否安装，如果安装了，卸载mysql

（1）查看

[root@hadoop102 桌面]# rpm -qa|grep mysql

mysql-libs-5.1.73-7.el6.x86\_64

（2）卸载

[root@hadoop102 桌面]# rpm -e --nodeps mysql-libs-5.1.73-7.el6.x86\_64

2．解压mysql-libs.zip文件到当前目录

[root@hadoop102 software]# unzip mysql-libs.zip

[root@hadoop102 software]# ls

mysql-libs.zip

mysql-libs

3．进入到mysql-libs文件夹下

[root@hadoop102 mysql-libs]# ll

总用量 76048

-rw-r--r--. 1 root root 18509960 3月 26 2015 MySQL-client-5.6.24-1.el6.x86\_64.rpm

-rw-r--r--. 1 root root 3575135 12月 1 2013 mysql-connector-java-5.1.27.tar.gz

-rw-r--r--. 1 root root 55782196 3月 26 2015 MySQL-server-5.6.24-1.el6.x86\_64.rpm

### 2.4.2 安装MySql服务器

1．安装mysql服务端

[root@hadoop102 mysql-libs]# rpm -ivh MySQL-server-5.6.24-1.el6.x86\_64.rpm

2．查看产生的随机密码

[root@hadoop102 mysql-libs]# cat /root/.mysql\_secret

OEXaQuS8IWkG19Xs

3．查看mysql状态

[root@hadoop102 mysql-libs]# service mysql status

4．启动mysql

[root@hadoop102 mysql-libs]# service mysql start

### 2.4.3 安装MySql客户端

1．安装mysql客户端

[root@hadoop102 mysql-libs]# rpm -ivh MySQL-client-5.6.24-1.el6.x86\_64.rpm

2．链接mysql

[root@hadoop102 mysql-libs]# mysql -uroot -pOEXaQuS8IWkG19Xs

3．修改密码

mysql>SET PASSWORD=PASSWORD('000000');

4．退出mysql

mysql>exit

### 2.4.4 MySql中user表中主机配置

配置只要是root用户+密码，在任何主机上都能登录MySQL数据库。

1．进入mysql

[root@hadoop102 mysql-libs]# mysql -uroot -p000000

2．显示数据库

mysql>show databases;

3．使用mysql数据库

mysql>use mysql;

4．展示mysql数据库中的所有表

mysql>show tables;

5．展示user表的结构

mysql>desc user;

6．查询user表

mysql>select User, Host, Password from user;

7．修改user表，把Host表内容修改为%

mysql>update user set host='%' where host='localhost';

8．删除root用户的其他host

mysql>delete from user where Host='hadoop102';

mysql>delete from user where Host='127.0.0.1';

mysql>delete from user where Host='::1';

9．刷新

mysql>flush privileges;

10．退出

mysql>quit;

## 2.5 Hive元数据配置到MySql

### 2.5.1 驱动拷贝

1．在/opt/software/mysql-libs目录下解压mysql-connector-java-5.1.27.tar.gz驱动包

[root@hadoop102 mysql-libs]# tar -zxvf mysql-connector-java-5.1.27.tar.gz

2．拷贝/opt/software/mysql-libs/mysql-connector-java-5.1.27目录下的mysql-connector-java-5.1.27-bin.jar到/opt/module/hive/lib/

[root@hadoop102 mysql-connector-java-5.1.27]# cp mysql-connector-java-5.1.27-bin.jar

/opt/module/hive/lib/

### 2.5.2 配置Metastore到MySql

1．在/opt/module/hive/conf目录下创建一个hive-site.xml

[atguigu@hadoop102 conf]$ touch hive-site.xml

[atguigu@hadoop102 conf]$ vi hive-site.xml

2．根据官方文档配置参数，拷贝数据到hive-site.xml文件中

<https://cwiki.apache.org/confluence/display/Hive/AdminManual+MetastoreAdmin>

<?xml version="1.0"?>

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

<configuration>

<property>

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

<value>jdbc:mysql://hadoop102:3306/metastore?createDatabaseIfNotExist=true</value>

<description>JDBC connect string for a JDBC metastore</description>

</property>

<property>

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

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

<description>Driver class name for a JDBC metastore</description>

</property>

<property>

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

<value>root</value>

<description>username to use against metastore database</description>

</property>

<property>

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

<value>000000</value>

<description>password to use against metastore database</description>

</property>

</configuration>

3．配置完毕后，如果启动hive异常，可以重新启动虚拟机。（重启后，别忘了启动hadoop集群）

### 2.5.3 多窗口启动Hive测试

1．先启动MySQL

[atguigu@hadoop102 mysql-libs]$ mysql -uroot -p000000

查看有几个数据库

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| mysql |

| performance\_schema |

| test |

+--------------------+

2．再次打开多个窗口，分别启动hive

[atguigu@hadoop102 hive]$ bin/hive

3．启动hive后，回到MySQL窗口查看数据库，显示增加了metastore数据库

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| metastore |

| mysql |

| performance\_schema |

| test |

+--------------------+

## 2.6 Hive常用交互命令

|  |
| --- |
| [atguigu@hadoop102 hive]$ bin/hive -help  usage: hive  -d,--define <key=value> Variable subsitution to apply to hive  commands. e.g. -d A=B or --define A=B  --database <databasename> Specify the database to use  -e <quoted-query-string> SQL from command line  -f <filename> SQL from files  -H,--help Print help information  --hiveconf <property=value> Use value for given property  --hivevar <key=value> Variable subsitution to apply to hive  commands. e.g. --hivevar A=B  -i <filename> Initialization SQL file  -S,--silent Silent mode in interactive shell  -v,--verbose Verbose mode (echo executed SQL to the console) |

1．“-e”不进入hive的交互窗口执行sql语句

[atguigu@hadoop102 hive]$ bin/hive -e "select id from student;"

2．“-f”执行脚本中sql语句

（1）在/opt/module/datas目录下创建hivef.sql文件

[atguigu@hadoop102 datas]$ touch hivef.sql

文件中写入正确的sql语句

select \*from student;

（2）执行文件中的sql语句

[atguigu@hadoop102 hive]$ bin/hive -f /opt/module/datas/hivef.sql

（3）执行文件中的sql语句并将结果写入文件中

[atguigu@hadoop102 hive]$ bin/hive -f /opt/module/datas/hivef.sql > /opt/module/datas/hive\_result.txt

## 2.7 Hive其他命令操作

1．退出hive窗口：

hive(default)>exit;

hive(default)>quit;

在新版的hive中没区别了，在以前的版本是有的：

exit:先隐性提交数据，再退出；

quit:不提交数据，退出；

2．在hive cli命令窗口中如何查看hdfs文件系统

hive(default)>dfs -ls /;

3．在hive cli命令窗口中如何查看本地文件系统

hive(default)>! ls /opt/module/datas;

4．查看在hive中输入的所有历史命令

（1）进入到当前用户的根目录/root或/home/atguigu

（2）查看. hivehistory文件

[atguigu@hadoop102 ~]$ cat .hivehistory

## 2.8 Hive常见属性配置

### 2.8.1 Hive数据仓库位置配置

1）Default数据仓库的最原始位置是在hdfs上的：/user/hive/warehouse路径下。

2）在仓库目录下，没有对默认的数据库default创建文件夹。如果某张表属于default数据库，直接在数据仓库目录下创建一个文件夹。

3）修改default数据仓库原始位置（将hive-default.xml.template如下配置信息拷贝到hive-site.xml文件中）。

|  |
| --- |
| <property>  <name>hive.metastore.warehouse.dir</name>  <value>/user/hive/warehouse</value>  <description>location of default database for the warehouse</description>  </property> |

配置同组用户有执行权限

bin/hdfs dfs -chmod g+w /user/hive/warehouse

### 2.8.2 查询后信息显示配置

1）在hive-site.xml文件中添加如下配置信息，就可以实现显示当前数据库，以及查询表的头信息配置。

<property>

<name>hive.cli.print.header</name>

<value>true</value>

</property>

<property>

<name>hive.cli.print.current.db</name>

<value>true</value>

</property>

2）重新启动hive，对比配置前后差异。

（1）配置前，如图6-2所示

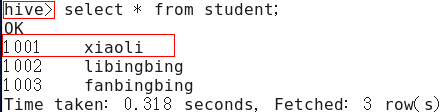


图6-2 配置前

（2）配置后，如图6-3所示

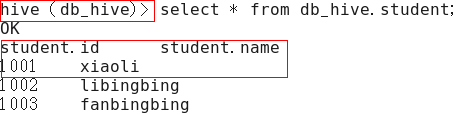


图6-3 配置后

### 2.8.3 Hive运行日志信息配置

1．Hive的log默认存放在/tmp/atguigu/hive.log目录下（当前用户名下）

2．修改hive的log存放日志到/opt/module/hive/logs

（1）修改/opt/module/hive/conf/hive-log4j.properties.template文件名称为

hive-log4j.properties

[atguigu@hadoop102 conf]$ pwd

/opt/module/hive/conf

[atguigu@hadoop102 conf]$ mv hive-log4j.properties.template hive-log4j.properties

（2）在hive-log4j.properties文件中修改log存放位置

hive.log.dir=/opt/module/hive/logs

### 2.8.4 参数配置方式

1．查看当前所有的配置信息

hive>set;

2．参数的配置三种方式

（1）配置文件方式

默认配置文件：hive-default.xml

用户自定义配置文件：hive-site.xml

注意：用户自定义配置会覆盖默认配置。另外，Hive也会读入Hadoop的配置，因为Hive是作为Hadoop的客户端启动的，Hive的配置会覆盖Hadoop的配置。配置文件的设定对本机启动的所有Hive进程都有效。

（2）命令行参数方式

启动Hive时，可以在命令行添加-hiveconf param=value来设定参数。

例如：

[atguigu@hadoop103 hive]$ bin/hive -hiveconf mapred.reduce.tasks=10;

注意：仅对本次hive启动有效

查看参数设置：

hive (default)> set mapred.reduce.tasks;

（3）参数声明方式

可以在HQL中使用SET关键字设定参数

例如：

hive (default)> set mapred.reduce.tasks=100;

注意：仅对本次hive启动有效。

查看参数设置

hive (default)> set mapred.reduce.tasks;

上述三种设定方式的优先级依次递增。即配置文件<命令行参数<参数声明。注意某些系统级的参数，例如log4j相关的设定，必须用前两种方式设定，因为那些参数的读取在会话建立以前已经完成了。

# 第3章 Hive数据类型

## 3.1 基本数据类型

表6-1

|  |  |  |  |
| --- | --- | --- | --- |
| Hive数据类型 | Java数据类型 | 长度 | 例子 |
| TINYINT | byte | 1byte有符号整数 | 20 |
| SMALINT | short | 2byte有符号整数 | 20 |
| INT | int | 4byte有符号整数 | 20 |
| BIGINT | long | 8byte有符号整数 | 20 |
| BOOLEAN | boolean | 布尔类型，true或者false | TRUE FALSE |
| FLOAT | float | 单精度浮点数 | 3.14159 |
| DOUBLE | double | 双精度浮点数 | 3.14159 |
| STRING | string | 字符系列。可以指定字符集。可以使用单引号或者双引号。 | ‘now is the time’ “for all good men” |
| TIMESTAMP |  | 时间类型 |  |
| BINARY |  | 字节数组 |  |

对于Hive的String类型相当于数据库的varchar类型，该类型是一个可变的字符串，不过它不能声明其中最多能存储多少个字符，理论上它可以存储2GB的字符数。

## 3.2 集合数据类型

表6-2

|  |  |  |
| --- | --- | --- |
| 数据类型 | 描述 | 语法示例 |
| STRUCT | 和c语言中的struct类似，都可以通过“点”符号访问元素内容。例如，如果某个列的数据类型是STRUCT{first STRING, last STRING},那么第1个元素可以通过字段.first来引用。 | struct() |
| MAP | MAP是一组键-值对元组集合，使用数组表示法可以访问数据。例如，如果某个列的数据类型是MAP，其中键->值对是’first’->’John’和’last’->’Doe’，那么可以通过字段名[‘last’]获取最后一个元素 | map() |
| ARRAY | 数组是一组具有相同类型和名称的变量的集合。这些变量称为数组的元素，每个数组元素都有一个编号，编号从零开始。例如，数组值为[‘John’, ‘Doe’]，那么第2个元素可以通过数组名[1]进行引用。 | Array() |

Hive有三种复杂数据类型ARRAY、MAP 和 STRUCT。ARRAY和MAP与Java中的Array和Map类似，而STRUCT与C语言中的Struct类似，它封装了一个命名字段集合，复杂数据类型允许任意层次的嵌套。

案例实操

1. 假设某表有如下一行，我们用JSON格式来表示其数据结构。在Hive下访问的格式为

|  |
| --- |
| {  "name": "songsong",  "friends": ["bingbing" , "lili"] , //列表Array,  "children": { //键值Map,  "xiao song": 18 ,  "xiaoxiao song": 19  }  "address": { //结构Struct,  "street": "hui long guan" ,  "city": "beijing"  }  } |

2）基于上述数据结构，我们在Hive里创建对应的表，并导入数据。

创建本地测试文件test.txt

|  |
| --- |
| songsong,bingbing\_lili,xiao song:18\_xiaoxiao song:19,hui long guan\_beijing  yangyang,caicai\_susu,xiao yang:18\_xiaoxiao yang:19,chao yang\_beijing |

注意：MAP，STRUCT和ARRAY里的元素间关系都可以用同一个字符表示，这里用“\_”。

3）Hive上创建测试表test

|  |
| --- |
| create table test(  name string,  friends array<string>,  children map<string, int>,  address struct<street:string, city:string>  )  row format delimited fields terminated by ','  collection items terminated by '\_'  map keys terminated by ':'  lines terminated by '\n'; |

字段解释：

row format delimited fields terminated by ',' -- 列分隔符

collection items terminated by '\_' --MAP STRUCT 和 ARRAY 的分隔符(数据分割符号)

map keys terminated by ':' -- MAP中的key与value的分隔符

lines terminated by '\n'; -- 行分隔符

4）导入文本数据到测试表

hive (default)> load data local inpath ‘/opt/module/datas/test.txt’into table test

5）访问三种集合列里的数据，以下分别是ARRAY，MAP，STRUCT的访问方式

|  |
| --- |
| hive (default)> select friends[1],children['xiao song'],address.city from test  where name="songsong";  OK  \_c0 \_c1 city  lili 18 beijing  Time taken: 0.076 seconds, Fetched: 1 row(s) |

## 3.3 类型转化

Hive的原子数据类型是可以进行隐式转换的，类似于Java的类型转换，例如某表达式使用INT类型，TINYINT会自动转换为INT类型，但是Hive不会进行反向转化，例如，某表达式使用TINYINT类型，INT不会自动转换为TINYINT类型，它会返回错误，除非使用CAST操作。

1．隐式类型转换规则如下

（1）任何整数类型都可以隐式地转换为一个范围更广的类型，如TINYINT可以转换成INT，INT可以转换成BIGINT。

（2）所有整数类型、FLOAT和STRING类型都可以隐式地转换成DOUBLE。

（3）TINYINT、SMALLINT、INT都可以转换为FLOAT。

（4）BOOLEAN类型不可以转换为任何其它的类型。

2．可以使用CAST操作显示进行数据类型转换

例如CAST('1' AS INT)将把字符串'1' 转换成整数1；如果强制类型转换失败，如执行CAST('X' AS INT)，表达式返回空值 NULL。

# 第4章 DDL数据定义

## 4.1 创建数据库

1）创建一个数据库，数据库在HDFS上的默认存储路径是/user/hive/warehouse/\*.db。

hive (default)> create database db\_hive;

2）避免要创建的数据库已经存在错误，增加if not exists判断。（标准写法）

|  |
| --- |
| hive (default)> create database db\_hive;  FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. Database db\_hive already exists  hive (default)> create database if not exists db\_hive; |

3）创建一个数据库，指定数据库在HDFS上存放的位置

hive (default)> create database db\_hive2 location '/db\_hive2.db';

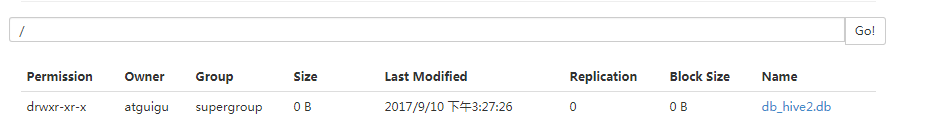


图6-4 数据库存放位置

## 4.2 查询数据库

### 4.2.1 显示数据库

1．显示数据库

hive> show databases;

2．过滤显示查询的数据库

hive> show databases like 'db\_hive\*';

OK

db\_hive

db\_hive\_1

### 4.2.2 查看数据库详情

1．显示数据库信息

hive> desc database db\_hive;

OK

db\_hive hdfs://hadoop102:9000/user/hive/warehouse/db\_hive.db atguiguUSER

2．显示数据库详细信息，extended

hive> desc database extended db\_hive;

OK

db\_hive hdfs://hadoop102:9000/user/hive/warehouse/db\_hive.db atguiguUSER

40.3.3 切换当前数据库

hive (default)> use db\_hive;

### 4.3.3 切换当前数据库

hive (default)> use db\_hive;

## 4.3 修改数据库

用户可以使用ALTER DATABASE命令为某个数据库的DBPROPERTIES设置键-值对属性值，来描述这个数据库的属性信息。数据库的其他元数据信息都是不可更改的，包括数据库名和数据库所在的目录位置。

hive (default)> alter database db\_hive set dbproperties('createtime'='20170830');

在hive中查看修改结果

hive> desc database extended db\_hive;

db\_name comment location owner\_name owner\_type parameters

db\_hive hdfs://hadoop102:8020/user/hive/warehouse/db\_hive.db atguigu USER {createtime=20170830}

## 4.4 删除数据库

1．删除空数据库

hive>drop database db\_hive2;

2．如果删除的数据库不存在，最好采用 if exists判断数据库是否存在

hive> drop database db\_hive;

FAILED: SemanticException [Error 10072]: Database does not exist: db\_hive

hive> drop database if exists db\_hive2;

3．如果数据库不为空，可以采用cascade命令，强制删除

hive> drop database db\_hive;

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

hive> drop database db\_hive cascade;

## 4.5 创建表

1．建表语法

CREATE [EXTERNAL] TABLE [IF NOT EXISTS] table\_name

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

[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]

[ROW FORMAT row\_format]

[STORED AS file\_format]

[LOCATION hdfs\_path]

2．字段解释说明

（1）CREATE TABLE 创建一个指定名字的表。如果相同名字的表已经存在，则抛出异常；用户可以用 IF NOT EXISTS 选项来忽略这个异常。

（2）EXTERNAL关键字可以让用户创建一个外部表，在建表的同时指定一个指向实际数据的路径（LOCATION），Hive创建内部表时，会将数据移动到数据仓库指向的路径；若创建外部表，仅记录数据所在的路径，不对数据的位置做任何改变。在删除表的时候，内部表的元数据和数据会被一起删除，而外部表只删除元数据，不删除数据。

（3）COMMENT：为表和列添加注释。

（4）PARTITIONED BY创建分区表

（5）CLUSTERED BY创建分桶表

（6）SORTED BY不常用

（7）ROW FORMAT

DELIMITED [FIELDS TERMINATED BY char] [COLLECTION ITEMS TERMINATED BY char]

[MAP KEYS TERMINATED BY char] [LINES TERMINATED BY char]

| SERDE serde\_name [WITH SERDEPROPERTIES (property\_name=property\_value, property\_name=property\_value, ...)]

用户在建表的时候可以自定义SerDe或者使用自带的SerDe。如果没有指定ROW FORMAT 或者ROW FORMAT DELIMITED，将会使用自带的SerDe。在建表的时候，用户还需要为表指定列，用户在指定表的列的同时也会指定自定义的SerDe，Hive通过SerDe确定表的具体的列的数据。

SerDe是Serialize/Deserilize的简称，目的是用于序列化和反序列化。

（8）STORED AS指定存储文件类型

常用的存储文件类型：SEQUENCEFILE（二进制序列文件）、TEXTFILE（文本）、RCFILE（列式存储格式文件）

如果文件数据是纯文本，可以使用STORED AS TEXTFILE。如果数据需要压缩，使用 STORED AS SEQUENCEFILE。

（9）LOCATION ：指定表在HDFS上的存储位置。

（10）LIKE允许用户复制现有的表结构，但是不复制数据。

### 4.5.1 管理表

1．理论

默认创建的表都是所谓的管理表，有时也被称为内部表。因为这种表，Hive会（或多或少地）控制着数据的生命周期。Hive默认情况下会将这些表的数据存储在由配置项hive.metastore.warehouse.dir(例如，/user/hive/warehouse)所定义的目录的子目录下。 当我们删除一个管理表时，Hive也会删除这个表中数据。管理表不适合和其他工具共享数据。

2．案例实操

（1）普通创建表

|  |
| --- |
| create table if not exists student2(  id int, name string  )  row format delimited fields terminated by '\t'  stored as textfile  location '/user/hive/warehouse/student2'; |

（2）根据查询结果创建表（查询的结果会添加到新创建的表中）

|  |
| --- |
| create table if not exists student3 as select id, name from student; |

（3）根据已经存在的表结构创建表

|  |
| --- |
| create table if not exists student4 like student; |

（4）查询表的类型

hive (default)> desc formatted student2;

Table Type: MANAGED\_TABLE

### 4.5.2 外部表

1．理论

因为表是外部表，所以Hive并非认为其完全拥有这份数据。删除该表并不会删除掉这份数据，不过描述表的元数据信息会被删除掉。

2．管理表和外部表的使用场景

每天将收集到的网站日志定期流入HDFS文本文件。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。

3．案例实操

分别创建部门和员工外部表，并向表中导入数据。

（1）原始数据

（2）建表语句

创建部门表

|  |
| --- |
| create external table if not exists default.dept(  deptno int,  dname string,  loc int  )  row format delimited fields terminated by '\t'; |

创建员工表

|  |
| --- |
| create external table if not exists default.emp(  empno int,  ename string,  job string,  mgr int,  hiredate string,  sal double,  comm double,  deptno int)  row format delimited fields terminated by '\t'; |

（3）查看创建的表

hive (default)> show tables;

OK

tab\_name

dept

emp

（4）向外部表中导入数据

导入数据

hive (default)> load data local inpath '/opt/module/datas/dept.txt' into table default.dept;

hive (default)> load data local inpath '/opt/module/datas/emp.txt' into table default.emp;

查询结果

hive (default)> select \* from emp;

hive (default)> select \* from dept;

（5）查看表格式化数据

hive (default)> desc formatted dept;

Table Type: EXTERNAL\_TABLE

### 4.5.3 管理表与外部表的互相转换

（1）查询表的类型

hive (default)> desc formatted student2;

Table Type: MANAGED\_TABLE

（2）修改内部表student2为外部表

alter table student2 set tblproperties('EXTERNAL'='TRUE');

（3）查询表的类型

hive (default)> desc formatted student2;

Table Type: EXTERNAL\_TABLE

（4）修改外部表student2为内部表

alter table student2 set tblproperties('EXTERNAL'='FALSE');

（5）查询表的类型

hive (default)> desc formatted student2;

Table Type: MANAGED\_TABLE

注意：('EXTERNAL'='TRUE')和('EXTERNAL'='FALSE')为固定写法，区分大小写！

## 4.6 分区表

分区表实际上就是对应一个HDFS文件系统上的独立的文件夹，该文件夹下是该分区所有的数据文件。Hive中的分区就是分目录，把一个大的数据集根据业务需要分割成小的数据集。在查询时通过WHERE子句中的表达式选择查询所需要的指定的分区，这样的查询效率会提高很多。

### 4.6.1 分区表基本操作

1．引入分区表（需要根据日期对日志进行管理）

/user/hive/warehouse/log\_partition/20170702/20170702.log

/user/hive/warehouse/log\_partition/20170703/20170703.log

/user/hive/warehouse/log\_partition/20170704/20170704.log

2．创建分区表语法

|  |
| --- |
| hive (default)> create table dept\_partition(  deptno int, dname string, loc string  )  partitioned by (month string)  row format delimited fields terminated by '\t'; |

3．加载数据到分区表中

hive (default)> load data local inpath '/opt/module/datas/dept.txt' into table default.dept\_partition partition(month='201709');

hive (default)> load data local inpath '/opt/module/datas/dept.txt' into table default.dept\_partition partition(month='201708');

hive (default)> load data local inpath '/opt/module/datas/dept.txt' into table default.dept\_partition partition(month='201707’);

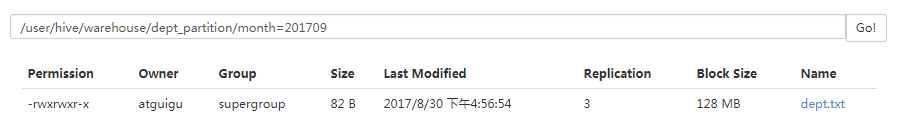


图6-5 加载数据到分区表

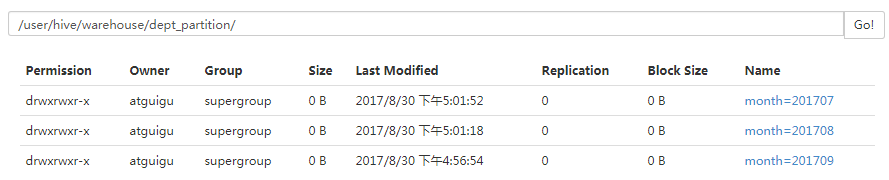


图6-6 分区表

4．查询分区表中数据

单分区查询

hive (default)> select \* from dept\_partition where month='201709';

多分区联合查询

hive (default)> select \* from dept\_partition where month='201709'

union

select \* from dept\_partition where month='201708'

union

select \* from dept\_partition where month='201707';

\_u3.deptno \_u3.dname \_u3.loc \_u3.month

10 ACCOUNTING NEW YORK 201707

10 ACCOUNTING NEW YORK 201708

10 ACCOUNTING NEW YORK 201709

20 RESEARCH DALLAS 201707

20 RESEARCH DALLAS 201708

20 RESEARCH DALLAS 201709

30 SALES CHICAGO 201707

30 SALES CHICAGO 201708

30 SALES CHICAGO 201709

40 OPERATIONS BOSTON 201707

40 OPERATIONS BOSTON 201708

40 OPERATIONS BOSTON 201709

5．增加分区

创建单个分区

hive (default)> alter table dept\_partition add partition(month='201706') ;

同时创建多个分区

hive (default)> alter table dept\_partition add partition(month='201705') partition(month='201704');

6．删除分区

删除单个分区

hive (default)> alter table dept\_partition drop partition (month='201704');

同时删除多个分区

hive (default)> alter table dept\_partition drop partition (month='201705'), partition (month='201706');

7．查看分区表有多少分区

hive> show partitions dept\_partition;

8．查看分区表结构

hive> desc formatted dept\_partition;

# Partition Information

# col\_name data\_type comment

month string

### 4.6.2 分区表注意事项

1．创建二级分区表

|  |
| --- |
| hive (default)> create table dept\_partition2(  deptno int, dname string, loc string  )  partitioned by (month string, day string)  row format delimited fields terminated by '\t'; |

2．正常的加载数据

（1）加载数据到二级分区表中

hive (default)> load data local inpath '/opt/module/datas/dept.txt' into table

default.dept\_partition2 partition(month='201709', day='13');

（2）查询分区数据

hive (default)> select \* from dept\_partition2 where month='201709' and day='13';

3．把数据直接上传到分区目录上，让分区表和数据产生关联的三种方式

（1）方式一：上传数据后修复

上传数据

hive (default)> dfs -mkdir -p

/user/hive/warehouse/dept\_partition2/month=201709/day=12;

hive (default)> dfs -put /opt/module/datas/dept.txt /user/hive/warehouse/dept\_partition2/month=201709/day=12;

查询数据（查询不到刚上传的数据）

hive (default)> select \* from dept\_partition2 where month='201709' and day='12';

执行修复命令

hive> msck repair table dept\_partition2;

再次查询数据

hive (default)> select \* from dept\_partition2 where month='201709' and day='12';

（2）方式二：上传数据后添加分区

上传数据

hive (default)> dfs -mkdir -p

/user/hive/warehouse/dept\_partition2/month=201709/day=11;

hive (default)> dfs -put /opt/module/datas/dept.txt /user/hive/warehouse/dept\_partition2/month=201709/day=11;

执行添加分区

hive (default)> alter table dept\_partition2 add partition(month='201709',

day='11');

查询数据

hive (default)> select \* from dept\_partition2 where month='201709' and day='11';

（3）方式三：上传数据后load数据到分区

创建目录

hive (default)> dfs -mkdir -p

/user/hive/warehouse/dept\_partition2/month=201709/day=10;

上传数据

hive (default)> load data local inpath '/opt/module/datas/dept.txt' into table

dept\_partition2 partition(month='201709',day='10');

查询数据

hive (default)> select \* from dept\_partition2 where month='201709' and day='10';

## 4.7 修改表

### 4.7.1 重命名表

1．语法

ALTER TABLE table\_name RENAME TO new\_table\_name

2．实操案例

hive (default)> alter table dept\_partition2 rename to dept\_partition3;

### 4.7.2 增加、修改和删除表分区

详见4.6.1分区表基本操作。

### 4.7.3 增加/修改/替换列信息

1．语法

更新列

ALTER TABLE table\_name CHANGE [COLUMN] col\_old\_name col\_new\_name column\_type [COMMENT col\_comment] [FIRST|AFTER column\_name]

增加和替换列

ALTER TABLE table\_name ADD|REPLACE COLUMNS (col\_name data\_type [COMMENT col\_comment], ...)

注：ADD是代表新增一字段，字段位置在所有列后面(partition列前)，REPLACE则是表示替换表中所有字段。

2．实操案例

（1）查询表结构

hive> desc dept\_partition;

（2）添加列

hive (default)> alter table dept\_partition add columns(deptdesc string);

（3）查询表结构

hive> desc dept\_partition;

（4）更新列

hive (default)> alter table dept\_partition change column deptdesc desc int;

（5）查询表结构

hive> desc dept\_partition;

（6）替换列

hive (default)> alter table dept\_partition replace columns(deptno string, dname

string, loc string);

（7）查询表结构

hive> desc dept\_partition;

## 4.8 删除表

hive (default)> drop table dept\_partition;

# 第5章 DML数据操作

## 5.1 数据导入

### 5.1.1 向表中装载数据（Load）

1．语法

hive> load data [local] inpath '/opt/module/datas/student.txt' [overwrite] into table student [partition (partcol1=val1,…)];

（1）load data:表示加载数据

（2）local:表示从本地加载数据到hive表；否则从HDFS加载数据到hive表

（3）inpath:表示加载数据的路径

（4）overwrite:表示覆盖表中已有数据，否则表示追加

（5）into table:表示加载到哪张表

（6）student:表示具体的表

（7）partition:表示上传到指定分区

2．实操案例

（0）创建一张表

hive (default)> create table student(id string, name string) row format delimited fields terminated by '\t';

（1）加载本地文件到hive

hive (default)> load data local inpath '/opt/module/datas/student.txt' into table default.student;

（2）加载HDFS文件到hive中

上传文件到HDFS

hive (default)> dfs -put /opt/module/datas/student.txt /user/atguigu/hive;

加载HDFS上数据

hive (default)> load data inpath '/user/atguigu/hive/student.txt' into table default.student;

（3）加载数据覆盖表中已有的数据

上传文件到HDFS

hive (default)> dfs -put /opt/module/datas/student.txt /user/atguigu/hive;

加载数据覆盖表中已有的数据

hive (default)> load data inpath '/user/atguigu/hive/student.txt' overwrite into table default.student;

### 5.1.2 通过查询语句向表中插入数据（Insert）

1．创建一张分区表

hive (default)> create table student(id int, name string) partitioned by (month string) row format delimited fields terminated by '\t';

2．基本插入数据

hive (default)> insert into table student partition(month='201709') values(1,'wangwu');

3．基本模式插入（根据单张表查询结果）

hive (default)> insert overwrite table student partition(month='201708')

select id, name from student where month='201709';

4．多插入模式（根据多张表查询结果）

hive (default)> from student

insert overwrite table student partition(month='201707')

select id, name where month='201709'

insert overwrite table student partition(month='201706')

select id, name where month='201709';

### 5.1.3 查询语句中创建表并加载数据（As Select）

详见4.5.1章创建表。

根据查询结果创建表（查询的结果会添加到新创建的表中）

create table if not exists student3

as select id, name from student;

### 5.1.4 创建表时通过Location指定加载数据路径

1．创建表，并指定在hdfs上的位置

hive (default)> create table if not exists student5(

id int, name string

)

row format delimited fields terminated by '\t'

location '/user/hive/warehouse/student5';

2．上传数据到hdfs上

hive (default)> dfs -put /opt/module/datas/student.txt

/user/hive/warehouse/student5;

3．查询数据

hive (default)> select \* from student5;

### 5.1.5 Import数据到指定Hive表中

注意：先用export导出后，再将数据导入。

hive (default)> import table student2 partition(month='201709') from

'/user/hive/warehouse/export/student';

## 5.2 数据导出

### 5.2.1 Insert导出

1．将查询的结果导出到本地

hive (default)> insert overwrite local directory '/opt/module/datas/export/student'

select \* from student;

2．将查询的结果格式化导出到本地

hive(default)>insert overwrite local directory '/opt/module/datas/export/student1'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' select \* from student;

3．将查询的结果导出到HDFS上(没有local)

hive (default)> insert overwrite directory '/user/atguigu/student2'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

select \* from student;

### 5.2.2 Hadoop命令导出到本地

hive (default)> dfs -get /user/hive/warehouse/student/month=201709/000000\_0

/opt/module/datas/export/student3.txt;

### 5.2.3 Hive Shell 命令导出

基本语法：（hive -f/-e 执行语句或者脚本 > file）

[atguigu@hadoop102 hive]$ bin/hive -e 'select \* from default.student;' >

/opt/module/datas/export/student4.txt;

### 5.2.4 Export导出到HDFS上

(defahiveult)>export table default.student to

'/user/hive/warehouse/export/student';

### 5.2.5 Sqoop导出

后续课程专门讲。

## 5.3 清除表中数据（Truncate）

注意：Truncate只能删除管理表，不能删除外部表中数据

hive (default)> truncate table student;