好程序员
从平凡到卓越，为梦想而拼搏
GoodProgrammer.org

2.5.1 Hive是什么

###2.5.1.1 为什么有Hive

```
Hive最初是应Facebook每天产生的海量新兴社会网络数据进行管理和机器学习的需求而产生和发展的。
数据量大，处理的效率比较低
mapreduce学习成本比较高
类sql的用法，工作效率高
入门比较简单
```

2.5.1.2 Hive是什么

```
hive是一个数据仓库。
hive是一个基于hadoop的数据仓库。
hive是一个基于hadoop的数据仓库，可以通过类sql的方式来对数据提供读、写以及管理（元数据）的功能。
总结：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数据库表，并提供类SQL查询功能。
主要用途：用来做离线数据分析，比直接用mapreduce开发效率更高
看官网，定义如下：
The Apache Hive data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Structure can be projected onto data already in storage. A command line tool and JDBC driver are provided to connect users to Hive.
```

2.5.1.3 为什么使用Hive

```
直接使用hadoop所面临的问题
    人员学习成本太高
    项目周期要求太短
    MapReduce实现复杂查询逻辑开发难度太大
为什么要使用Hive
    操作接口采用类SQL语法，提供快速开发的能力。
    避免了去写MapReduce，减少开发人员的学习成本。
    功能扩展很方便。
```

##2.5.2 Hive的特点

千锋好程序大数据学院

可扩展

Hive可以自由的扩展集群的规模，一般情况下不需要重启服务。

延展性

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。

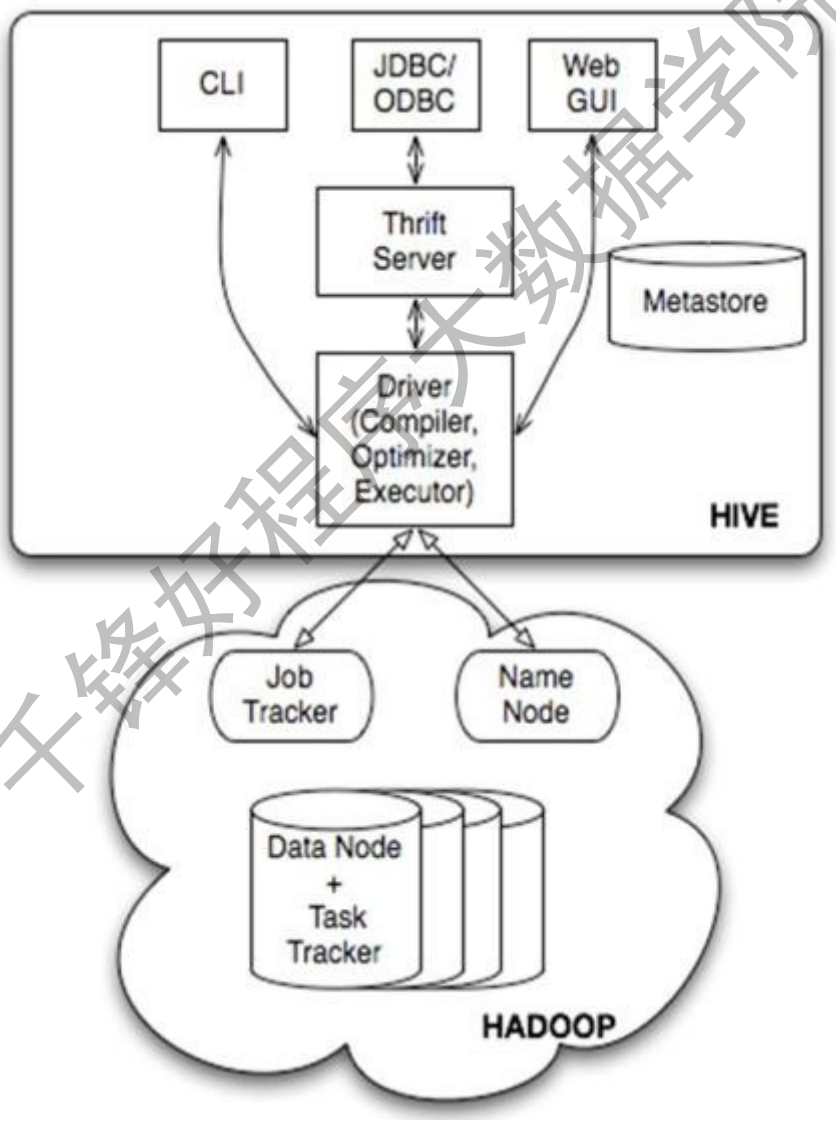
容错

良好的容错性，节点出现问题SQL仍可完成执行。

看官网，介绍如下：

```
1、Tools to enable easy access to data via SQL, thus enabling data warehousing tasks such as extract/transform/load (ETL), reporting, and data analysis.
2、A mechanism to impose structure on a variety of data formats
3、Access to files stored either directly in Apache HDFS? or in other data storage systems such as Apache HBase?
4、Query execution via Apache Tez?, Apache Spark?, or MapReduce
5、Procedural language with HPL-SQL
6、Sub-second query retrieval via Hive LLAP, Apache YARN and Apache Slider.
```

##2.5.3 Hive架构简述



千锋好程序大数据学院

用户连接客户端：cli、jdbc/odbc 、web gui
thriftserver:第三方服务
metastore：存储hive的元数据（库名、表名、字段名、字段类型、分区、分桶、创建时间、创建人等）。
解析器：将hql抽象成表达式树
编译器：对hql语句进行词法、语法、语义的编译(需要跟元数据关联)，编译完成后会生成一个有向无环的执行计划。hive上就是编译成mapreduce的job。
优化器：将执行计划进行优化，减少不必要的列、使用分区、使用索引等。优化job。
执行器：将优化后的执行计划提交给hadoop的yarn上执行。提交job。

2.5.4 Hive和Hadoop的关系

hive是基于hadoop的。
hive本身其实没有多少功能，hive就相当于在hadoop上面包了一个壳子，就是对hadoop进行了一次封装。
hive的存储是基于hdfs/hbase的，hive的计算是基于mapreduce。

##2.5.5 Hive的安装部署

Hive常用的安装分三种(注意：Hive会自动监测Hadoop的环境变量，如有就必须启动Hadoop)

本地模式(多用户模式)：

使用hive自带默认元数据库derby来进行存储，通常用于测试
优点：使用简单，不用进行配置
缺点：只支持单session。

安装步骤：
1、解压hive，并配置环境变量
vi /etc/profile
source /etc/profile
2、配置hive-env.sh
export JAVA_HOME=
3、启动hive：bin/hive
1.1.2版本有依赖冲突：jline版本冲突,1.2.1之后版本无此冲突
/usr/local/hive-1.2.1/lib和/usr/local/hadoop-2.6.4/share/hadoop/yarn/lib中都包含jline的jar包，导致版本冲突
解决方案：
1、cp /usr/local/hive-1.2.1/lib/jline-2.12.jar /usr/local/hadoop-2.6.4/share/hadoop/yarn/lib/
2、rm -rf jline-0.9.94.jar
特点：元数据库文件会在启动hive命令的目录下生成。（在不同目录下测试启动；在相同目录下多次启动hive）

##2.5.6 配置Hive的远程模式

###2.5.6.1 远程模式（将元数据库放在该台机器上）(多用户模式)

通常使用关系型数据库来进行元数据存储（mysql、oracle等执行带jdbc驱动的数据库）
优点：支持多session
缺点：需要配置、还需要安装mysql等关系型数据库
步骤：
1、解压并配置环境变量
2、配置hive的配置文件
cp hive-env.sh.template hive-env.sh
vi hive-env.sh（可以配置jdk、hive的conf路径）
3、配置hive的自定义配置文件

千锋好程序大数据学院

```
vi hive-site.xml
<configuration>
<!--配置mysql的连接字符串-->
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=true</value>
<description>JDBC connect string for a JDBC metastore</description>
</property>
<!--配置mysql的连接驱动-->
<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>com.mysql.jdbc.Driver</value>
<description>Driver class name for a JDBC metastore</description>
</property>
<!--配置登录mysql的用户-->
<property>
<name>javax.jdo.option.ConnectionUserName</name>
<value>root</value>
<description>username to use against metastore database</description>
</property>
<!--配置登录mysql的密码-->
<property>
<name>javax.jdo.option.ConnectionPassword</name>
<value>root</value>
<description>password to use against metastore database</description>
</property>
</configuration>

注意：hive的元数据在mysql库里创建的数据库hive的编码最好设置成latin1.
4、将mysql的驱动包上传到$HIVE_HOME/lib下
5、启动hive
```

###2.5.6.2 远程模式（将元数据库放在其他机器上）(多用户模式)

```
和1差不多，只是将元数据放在别的服务器上，这种的就是咱们常说的集群模式。
可以有一个hive的server和多个hive的client。
```

##2.5.7 数据库基本操作-库、表

```
注释内容：
//
/**
*/
<!--anc-->
#

语法规则：
hive的数据库名、表名都不区分大小写

命名规则：
1、名字不能使用数字开头
2、不能使用关键字
3、尽量不使用特殊符号
```

千锋好程序大数据学院

```
hive有一个默认的数据库default，如果不明确的说明要使用哪个库，则使用默认数据库。
create database qf1704;
create database if not exists qf1704;
create database if not exists qfdb comment 'this is a database of qf';
```

```
创建库的本质：在hive的warehouse目录下创建一个目录（库名.db命名的目录）
切换库：
use qf1704;
create table t_user(id int,name string);
使用库的另一种方式：
create table qfdb.t_user(id int,name string);
```

###2.5.7.1 加载数据

1、将文件直接上传到表目录下。

```
mkdir /hivedata
cd /hivedata
vi user.txt
#数据从班级名单来
1,廉德枫
2,刘浩（小）
3,王鑫
4,司翔

hdfs dfs -put /hivedata/user.txt 表目录
```

###2.5.7.2 建表语法

CREATE [EXTERNAL] TABLE [IF NOT EXISTS] TABLENAME
[COLUMNNAME COLUMNTYPE [COMMENT 'COLUMN COMMENT'],...]
[COMMENT 'TABLE COMMENT']
[PARTITIONED BY (COLUMNNAME COLUMNTYPE [COMMENT 'COLUMN COMMENT'],...)]
[CLUSTERED BY (COLUMNNAME COLUMNTYPE [COMMENT 'COLUMN COMMENT'],...) [SORTED BY (COLUMNNAME [ASC|DESC])...] INTO NUM_BUCKETS BUCKETS]
[ROW FORMAT ROW_FORMAT]
[STORED AS FILEFORMAT]
[LOCATION HDFS_PATH];

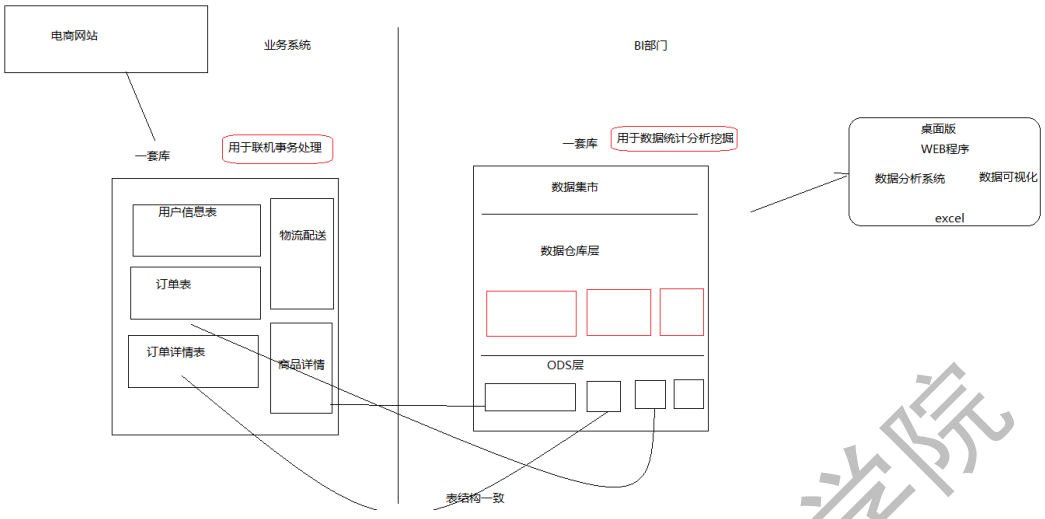
##2.5.8 表类型详解

```
1、内部表 表目录会创建在hdfs的/user/hive/warehouse/下的相应的库对应的目录中。
2、外部表 外部表会根据创建表时LOCATION指定的路径来创建目录，如果没有指定LOCATION，则位置跟内部表相同

内部表和外部表在创建时的差别：就差两个关键字，EXTERNAL LOCATION
举例：
内部表 -- CRAATE TABLE T_INNER(ID INT);
外部表 -- CREATE EXTERNAL TABLE T_OUTER(ID  INT) LOCATION 'HDFS:///AA/BB/XX';

Hive表创建时要做的两件事：
1、在hdfs下创建表目录
2、在元数据库mysql创建相应表的描述数据（元数据）
内部表和外部表在drop时有不同的特性：
1、drop时，元数据都会被清除
2、drop时，内部表的表目录会被删除，但是外部表的表目录不会被删除。
```

千锋扣丁程序大数据学院

外部表的使用场景：使用后数据不想被删除的情况使用外部表（推荐使用）
所以，整个数据仓库的最底层的表使用外部表。

2.5.9 数据仓库概念




```
create table if not exists t2(
id int,
name string
)
row format delimited fields terminated by ','
lines terminated by '\n'
stored as textfile
;
```

创建表的本质：在使用的库目录下创建一个目录（目录名就是表名）

2.5.9.1 加载数据

常用方式：

```
    load data [local] inpath '/hivedata/user.txt' [overwrite] into table t_user;
```

加载数据的本质：
1、如果数据在本地，加载数据的本质就是将数据copy到hdfs上的表目录下。
2、如果数据在hdfs上，加载数据的本质是将数据移动到hdfs的表目录下。

```
load data local inpath '/hivedata/user.txt' into table t2;
```

hive使用的是严格的读时模式：加载数据时不检查数据的完整性，读时发现数据不对则使用NULL来代替。
而mysql使用的是写时模式：

```
create table t3(
id int,
name string
) comment 'this is a table'
row format delimited fields terminated by ','
lines terminated by '\n'
stored as textfile
;
```

千锋好程序大数据学院

###2.5.9.2 insert into 方式灌入数据

```
insert into t3
select * from t2 where id < 3
;
```

###2.5.9.3 克隆表

####2.5.9.3.1 不带数据，只克隆表的结构

```
create table if not exists t4 like t3;
```

####2.5.9.3.2 克隆表并带数据

```
create table if not exists t5 like t3 location '/user/hive/warehouse/qf1704.db/t3';
```

```
create table if not exists t6(
id int,
name string
) comment 'this is a table'
row format delimited fields terminated by ','
lines terminated by '\n'
stored as textfile
location '/user/hive/warehouse/qf1704.db/t3'
;

后面接的一定是hdfs中的目录，不是文件
```

####2.5.9.3.3 克隆表并带数据

```
更灵活的方式:跟创建表的方式一样，元数据和目录都会创建。
create table t7
as
select * from t3
;
```

```
查看库描述:
    desc/describe database [extended] qf1704;
查看表描述:
    desc/describe [extended] t7;
desc formatted t7;
show create table t7; 比较全
```

千锋好程序大数据学院

####2.5.9.3.4 示例

```
CREATE TABLE log1(
id          string COMMENT 'this is id column',
phonenumber     bigint,
mac           string,
ip            string,
url           string,
status1         string,
status2         string,
upflow          int,
downflow        int,
status3    string,
dt string
)
COMMENT 'this is log table'
```

ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
LINES TERMINATED BY '\n'
stored as textfile;

```
加载数据：
    load data local inpath '/root/data.log.txt' into table log;

1、统计每个电话号码的总流量(M)
    select l.phonenumber,
    round(sum(l.upflow + l.downflow) / 1024.0,2) as total
    from log1 l
    group by l.phonenumber
    ;

2、第二个需求，求访问次数排名前3的url：
    select l.url url,
    count(l.url) as urlcount
    from log1 l
    group by l.url
    order by urlcount desc
    limit 3
    ;

3、模拟收费：
```

####2.5.9.3.5 表属性修改

```
1、修改表名  rename to ???是否comment
    alter table t7 rename to a1;

2、修改列名：change column
    alter table a1 change column name name1 string;

3、修改列的位置
    alter table log1 change column ip ip string after status3;
    alter table log1 change column ip ip string first;

4、修改字段类型：
    alter table a1 change column name1 name int;

5、增加字段：add columns
    alter table a1 add columns (
    sex int,...)
    ;

6、删除字段：replace columns
    alter table a1 replace columns(
    id int,
    name int,
    size int,
    pic string
    );

7、内部表和外部表转换：
```

千锋好程序大数据学院

```
    alter table a1 set tblproperties('EXTERNAL'='TRUE'); ###内部表转外部表，true一
定要大写;
    alter table a1 set tblproperties('EXTERNAL'='false');##false大小写都没有关系

8、显示当前库：
    <property>
        <name>hive.cli.print.current.db</name>
        <value>false</value>
        <description>Whether to include the current database in the Hive prompt.
</description>
    </property>

当前session里设置该参数：
hive > set hive.cli.print.current.db=true;
查看当前设置的值：
hive > set hive.cli.print.current.db;

9、删除库：
drop database qfdb;             ####这个只能删除空库
drop database qfdb cascade;     ####如果不是空库，则可以加cascade强制删除
```

2.5.10 分区表概念

###2.5.10.1 为什么分区

```
随着系统运行的时间越来越长，表的数据量越来越大，而hive查询通常是使用全表扫描，
这样会导致大量不必要的数据扫描，从而大大降低了查询的效率。
从而引进分区技术，使用分区技术，能避免hive做全表扫描，从而提交查询效率。
可以将用户的整个表在存储上分成多个子目录（子目录以分区变量的值来命名）
```

###2.5.10.2 怎么分区

```
根据业务需求，通常以年、月、日、小时、地区等进行分区。
```

###2.5.10.3 分区的技术

```
[PARTITIONED BY (COLUMNNAME COLUMNTYPE [COMMENT 'COLUMN COMMENT'],...)]
1、hive的分区名区分大小写
2、hive的分区字段是一个伪字段，但是可以用来进行操作
3、一张表可以有一个或者多个分区，并且分区下面也可以有一个或者多个分区。
```

###2.5.10.4 分区的意义

```
可以让用户在做数据统计的时候缩小数据扫描的范围，因为可以在select是指定要统计哪个分区
```

2.5.10.5 分区本质

```
在表的目录或者是分区的目录下在创建目录，分区的目录名为指定字段=值
```

2.5.11 分区表案例

###2.5.11.1 一级分区

千锋好程序大数据学院

创建分区表：

```
create table if not exists part1(
id int,
name string
)
partitioned by (dt string)
row format delimited fields terminated by ','
;
```

加载数据

```
load data local inpath '/hivedata/user.txt' into table part1 partition(dt='2018-03-21');
load data local inpath '/hivedata/user.txt' into table part1 partition(dt='2018-03-20');
```

###2.5.11.2 二级分区

```
create table if not exists part2(
id int,
name string
)
partitioned by (year string,month string)
row format delimited fields terminated by ','
;

load data local inpath '/hivedata/user.txt' into table part2
partition(year='2018',month='03');
load data local inpath '/hivedata/user.txt' into table part2
partition(year='2018',month=02);
```

###2.5.11.3 三级分区

```
create table if not exists part3(
id int,
name string
)
partitioned by (year string,month string,day string)
row format delimited fields terminated by ','
;

load data local inpath '/hivedata/user.txt' into table part3
partition(year='2018',month='03',day='21'); month=03

load data local inpath '/hivedata/user.txt' into table part3
partition(year='2018',month='02',day='20');  month=2

create table if not exists part4(
id int,
name string
)
partitioned by (year string,month string,DAY string)
row format delimited fields terminated by ','
;

load data local inpath '/hivedata/user.txt' into table part4
partition(year='2018',month='03',day='21');
```

千锋好程序大数据学院

###2.5.11.4 测试是否区分大小写

```
load data local inpath '/hivedata/user.txt' into table part4
partition(year='2018',month='03',day='AA');
```

###2.5.11.5 查看分区:

```
show partitions part4;
```

###2.5.11.6 修改分区：

```
1、修改分区名字？？？（没有命令，暴力修改）
增加分区：
alter table part5 add partition(dt='2018-03-21');
alter table part5 add partition(dt='2018-03-20') partition(dt='2018-03-17');
```

###2.5.11.7 增加分区并设置数据

```
alter table part5 add partition(dt='2018-03-27') location
'/user/hive/warehouse/qf1704.db/part1/dt=2018-03-20';

alter table part5 add partition(dt='2018-03-26') location
'/user/hive/warehouse/qf1704.db/part1/dt=2018-03-20'
partition(dt='2018-03-24') location
'/user/hive/warehouse/qf1704.db/part1/dt=2018-03-21';
```

###2.5.11.8 修改分区的hdfs的路径

（注意：location后接的hdfs路径需要写成完全路径）

```
alter table part5 partition(dt='2018-03-21') set location
'/user/hive/warehouse/qf1704.db/part1/dt=2018-03-21';    --错误使用
alter table part5 partition(dt='2018-03-21') set location
'hdfs://hdp01:9000/user/hive/warehouse/qf1704.db/part1/dt=2018-03-21';
```

###2.5.11.9 删除分区

```
alter table part5 drop partition(dt='2018-03-21');
```

###2.5.11.10 删除多个分区

```
alter table part5 drop partition(dt='2018-03-24'),partition(dt='2018-03-26');
```

##2.5.12 分区类型详解

```
静态分区：加载数据到指定分区的值。
动态分区：数据未知，根据分区的值来确定需要创建的分区。
混合分区：静态和动态都有。
```

2.5.13 动态分区属性设置及示例

千锋教育程序大数据学院

```
动态分区的属性：
hive.exec.dynamic.partition=true
hive.exec.dynamic.partition.mode=strict/nonstrict
hive.exec.max.dynamic.partitions=1000
hive.exec.max.dynamic.partitions.pernode=100
```

###2.5.13.1 创建动态分区表

```
create table dy_part1(
id int,
name string
)
partitioned by (dt string)
row format delimited fields terminated by ','
;
```

###2.5.13.2 动态分区加载数据不能使用load方式加载

```
load data local inpath '/hivedata/user.txt' into table dy_part1 partition(dt);
```

###2.5.13.3 动态分区使用insert into的方式加载数据

```
先创建临时表：
create table temp_part(
id int,
name string,
dt string
)
row format delimited fields terminated by ','
;

#数据从班级名单来
1,廉德枫,2019-06-25
2,刘浩（小）,2019-06-25
3,王鑫,2019-06-25
4,司翔,2019-06-25

导入数据：
load data local inpath '/hivedata/t4.txt' into table temp_part;

insert into dy_part1 partition(dt)
select id,name,dt from temp_part;
```

##2.5.14 混合分区示例

(注意列的个数匹配)

```
create table dy_part2(
id int,
name string
)
partitioned by (year string,month string,day string)
row format delimited fields terminated by ','
;
```

千锋好程序大数据学院

###2.5.14.1 创建临时表

```
create table temp_part2(
id int,
name string,
year string,
month string,
day string
)
row format delimited fields terminated by ','
;

#数据从班级名单来
1,廉德枫,2019,06,25
2,刘浩（小）,2019,06,25
3,王鑫,2019,06,25
4,司翔,2019,06,25
```

###2.5.14.2 导入数据到分区表

(注意：字段的对应)

```
错误用法：
insert into dy_part2 partition (year='2018',month,day)
select * from temp_part2;
正确用法：
insert into dy_part2 partition (year='2018',month,day)
select id,name,month,day from temp_part2;
```

###2.5.14.3 设置hive为严格模式执行

```
set hive.mapred.mode=nonstrict/strict;
<name>hive.mapred.mode</name>
<value>nonstrict</value>
<description>
  The mode in which the Hive operations are being performed.
  In strict mode, some risky queries are not allowed to run. They include:
    Cartesian Product.
    No partition being picked up for a query.
    Comparing bigints and strings.
    Comparing bigints and doubles.
    Orderby without limit.
</description>
```

##2.5.15 分区表注意事项

```
1、hive的分区使用的是表外字段，分区字段是一个伪列，但是分区字段是可以做查询过滤。
2、分区字段不建议使用中文
3、一般不建议使用动态分区，因为动态分区会使用mapreduce来进行查询数据，如果分区数据过多，导致namenode和resourcemanager的性能瓶颈。所以建议在使用动态分区前尽可能预知分区数量。
4、分区属性的修改都可以使用修改元数据和hdfs数据内容。
```

2.5.16 Hive分区和Mysql分区的区别

千锋好程序大数据学院

```
mysql分区字段用的是表内字段；而hive分区字段采用表外字段。
```

2.5.17 分桶

###2.5.17.1 为什么要分桶

```
当单个的分区或者表的数据量过大，分区不能更细粒度的划分数据，就需要使用分桶技术将数据划分成更细的粒度。
```

###2.5.17.2 分桶技术

```
[CLUSTERED BY (COLUMNNAME COLUMNTYPE [COMMENT 'COLUMN COMMENT'],...)
[SORTED BY (COLUMNNAME [ASC|DESC])...] INTO NUM_BUCKETS BUCKETS]
```

###2.5.17.3 关键字

```
bucket
```

###2.5.17.4 分桶的意义

为了保存分桶查询的分桶结构（数据按照分桶字段进行保存hash散列）
分桶表的数据进行抽样和JOIN时可以提高查询效率 sample

##2.5.18 分桶表的使用示例

首先要创建带分桶定义的表（分桶表）
然后创建一个临时表（普通表）
从临时表中使用分桶查询将查询到的数据插入到分桶表中

###2.5.18.1 创建分桶表

```
create table if not exists buc1(
id int,
name string,
age int
)
clustered by (id) into 4 buckets
row format delimited fields terminated by ','
;

数据：
id,name,age
1,aa1,18
2,aa2,19
3,aa3,20
4,aa4,21
5,aa5,22
6,aa6,23
7,aa7,24
8,aa8,25
9,aa9,26
```

###2.5.18.2 创建临时表

千锋好程序大数据学院

```
create table if not exists temp_buc1(
id int,
name string,
age int
)
row format delimited fields terminated by ','
;
```

###2.5.18.3 分桶使用load方式加载数据不能体现分桶

```
load data local inpath '/home/hivedata/buc1.txt' into table buc1;
```

###2.5.18.4 加载数据到临时表

```
load data local inpath '/home/hivedata/buc1.txt' into table temp_buc1;
```

###2.5.18.5 使用分桶查询将数据导入到分桶表

```
insert overwrite table buc2
select id,name,age from temp_buc1
cluster by (id)
;
```

###2.5.18.6 设置强制分桶的属性

```
set hive.enforce.bucketing=false/true
<name>hive.enforce.bucketing</name>
<value>false</value>
<description>Whether bucketing is enforced. If true, while inserting into the
table, bucketing is enforced.</description>
```

###2.5.18.7 如果设置了reduces的个数和总桶数不一样，请手动设置:

```
set mapreduce.job.reduces=?
```

###2.5.18.8 创建指定排序字段的分桶表

```
create table if not exists buc4(
id int,
name string,
age int
)
clustered by (id)
sorted by (id desc) into 8 buckets
row format delimited fields terminated by ','
;
```

###2.5.18.9 导入数据

千锋好程序大数据学院

```
insert overwrite table buc3
select id,name,age from temp_buc1
distribute by (id) sort by (id asc)
;
和下面的语句效果一样
insert overwrite table buc4
select id,name,age from temp_buc1
cluster by (id)
;
```

##2.5.19 分桶表查询案例

```
select * from buc3;
select * from buc3 tablesample(bucket 1 out of 1 on id);
```

###2.5.19.1 查询第1桶的数据

```
select * from buc3 tablesample(bucket 1 out of 4 on id);
select * from buc3 tablesample(bucket 2 out of 4 on id);
select * from buc3 tablesample(bucket 1 out of 2 on id);

tablesample(bucket x out of y on id)
x:代表从第几桶开始查询
y:查询的总桶数，y可以是总桶数的倍数或者是因子。x不能大于y。

不压缩不拉伸：

1 1+4

压缩：

1 1+4/2 1+4/2+4/2
2 2+4/2 2+4/2+4/2

select * from buc3 tablesample(bucket 1 out of 8 on id);
select * from buc3 tablesample(bucket 2 out of 8 on id);
```

###2.5.19.2 查询

（注意：tablesample一定是放在from后面）

查询id为基数：

```
select
*
from buc2 tablesample(bucket 2 out of 2 on uid)
where uname = "aa1"
;

select
*
from buc2
where uname = "aa1"
;

查询：
```

千锋好程序大数据学院

```
select * from temp_buc1 limit 3;
select * from temp_buc1 tablesample(3 rows);
select * from temp_buc1 tablesample(30 percent);
select * from temp_buc1 tablesample(6B);B k M G T P

SELECT * from temp_buc1 order by rand() limit 3;
```

##2.5.20 分区分桶联合案例

需求：按照性别分区（1男2女），在分区中按照id的奇偶进行分桶：

```
id,name,sex
1,bb1,1
2,bb2,2
3,bb3,2
4,bb4,1
5,bb5,1
6,bb6,1
7,bb7,2
8,bb8,1
9,bb9,2
```

建表：

```
create table if not exists stu(
id int,
name string
)
partitioned by (sex string)
clustered by (id) into 2 buckets
row format delimited fields terminated by ','
;
```

建临时表：

```
create table if not exists stu_temp(
id int,
name string,
sex string
)
row format delimited fields terminated by ','
;
```

加载临时表的数据

```
load data local inpath '/home/hivedata/stu.txt' into table stu_temp;
```

将数据导入到分区分桶表

```
insert overwrite table stu partition(sex)
select id,name,sex from stu_temp cluster by id ;
```

查询性别为女，学号为奇数的学生

```
select * from stu tablesample(bucket 2 out of 2 on id)
where sex = '2';
```

千锋好程序大数据学院

2.5.21 分桶表总结和注意事项

2.5.21.1 分桶总结

```
1、定义
    clustered by (id)          ---指定分桶的字段
    sorted by (id asc|desc)    ---指定数据的排序规则，表示咱们预期的数据是以这种规则进行的排序

2、导入数据
    cluster by (id)   ---指定getPartition以哪个字段来进行hash，并且排序字段也是指定的字段，排序是以asc排列
    distribute by (id)    ---- 指定getPartition以哪个字段来进行hash
    sort by (name asc | desc) ---指定排序字段

    区别：distribute by 这种方式可以分别指定getPartition和sort的字段

导数据时：
    insert overwrite table buc3
    select id,name,age from temp_buc1
    distribute by (id) sort by (id asc)
    ;
    和下面的语句效果一样
    insert overwrite table buc4
    select id,name,age from temp_buc1
    cluster by (id)
    ;
```

2.5.21.2 注意事项

```
分区使用的是表外字段，分桶使用的是表内字段
分桶跟家细粒度的管理数据，更多的是使用来做抽样、join
```

2.5.22 查询语句基本语法

```
select
from
join on
where
group by
having
order by
sort by
limit
union/union all
;
```

sql语句的执行顺序

```
FROM
<left_table>
ON
<join_condition>
<join_type>
 JOIN
<right_table>
```

千锋好程序大数据学院

```
WHERE
<where_condition>
GROUP BY
<group_by_list>
HAVING
<having_condition>
SELECT
DISTINCT
<select_list>
ORDER BY
<order_by_condition>
LIMIT
<limit_number>
```

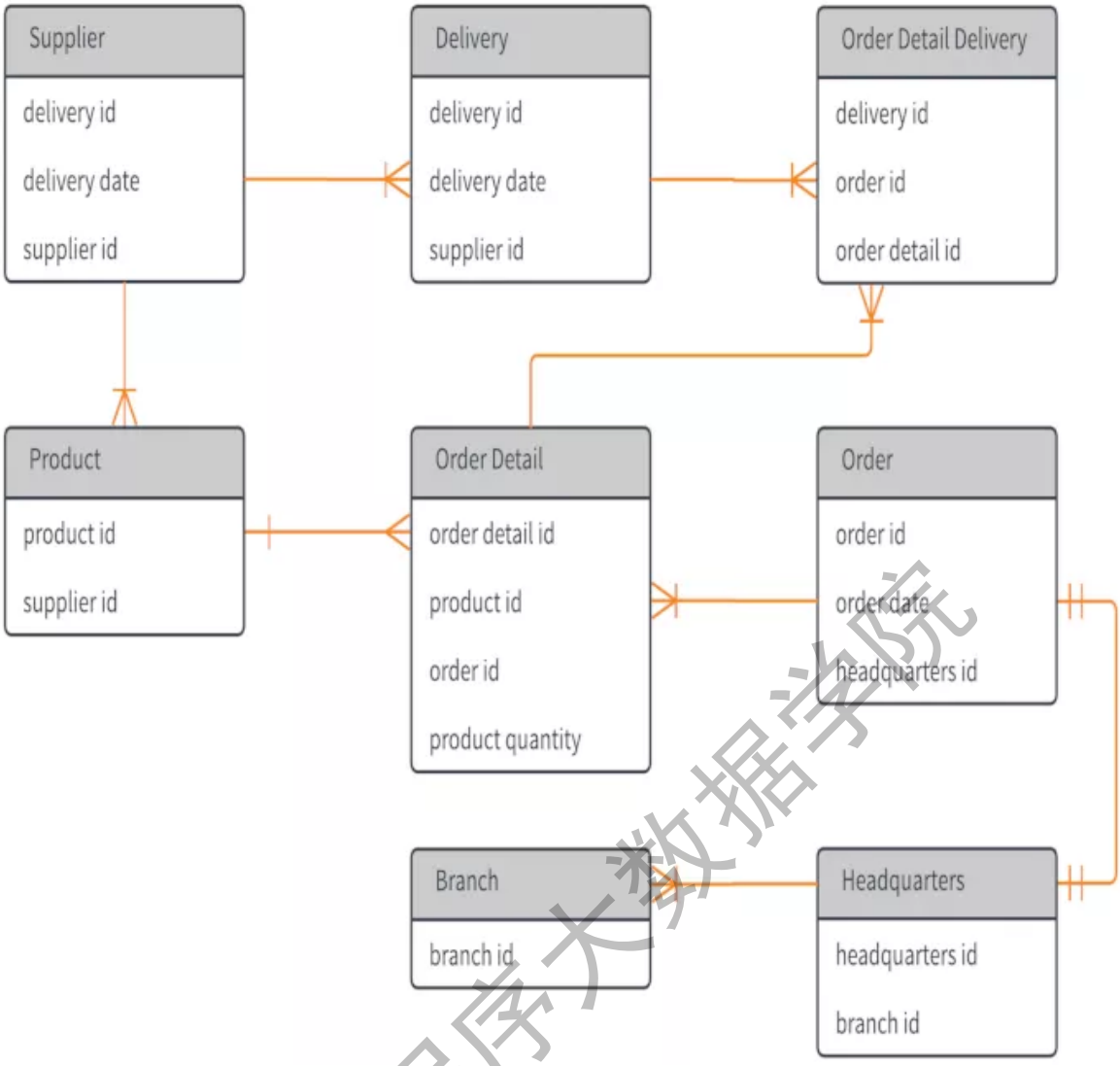
```
尽量不要使用子查询、尽量不要使用 in not in
select * from aa1
where id in (select id from bb);
查询尽量避免join连接查询，但是这种操作咱们是永远避免不了的。
查询永远是小表驱动大表（永远是小结果集驱动大结果集）
```

关系型数据库最难的地方，就是建模（model）。

错综复杂的数据，需要建立模型，才能储存在数据库。所谓"模型"就是两样东西：实体（entity）+ 关系（relationship）。

实体指的是那些实际的对象，带有自己的属性，可以理解成一组相关属性的容器。关系就是实体之间的联系，通常可以分成"一对一"、"一对多"和"多对多"等类型。

千锋好程序大数据学院



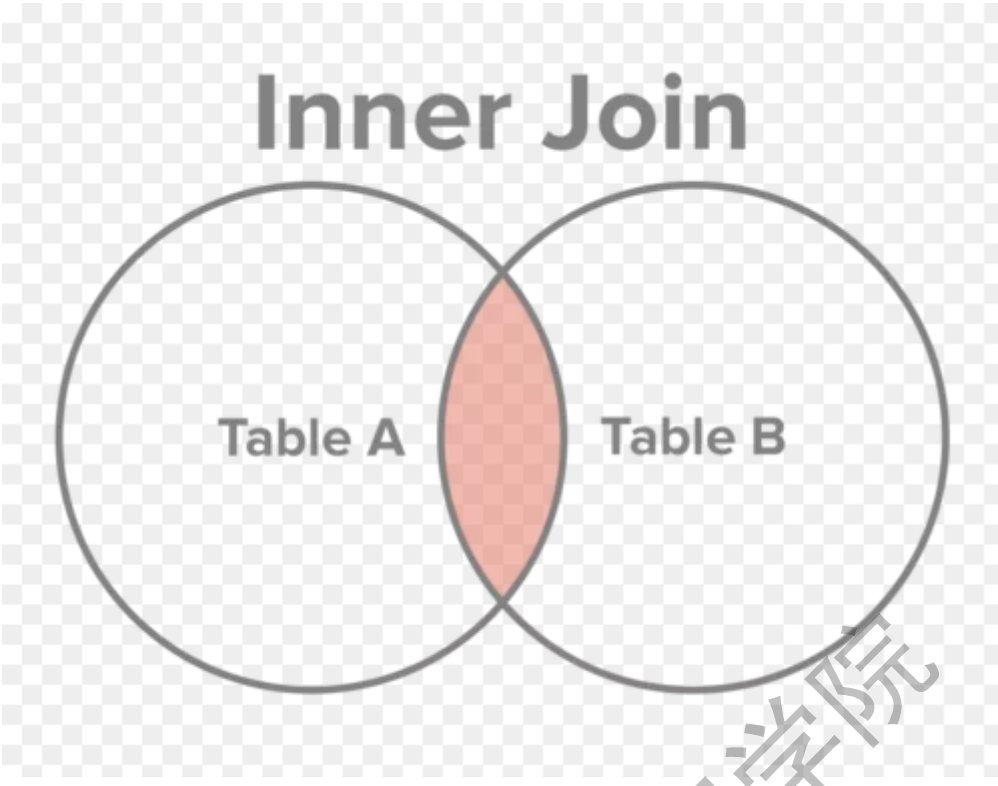
千锋好程序大数据学院

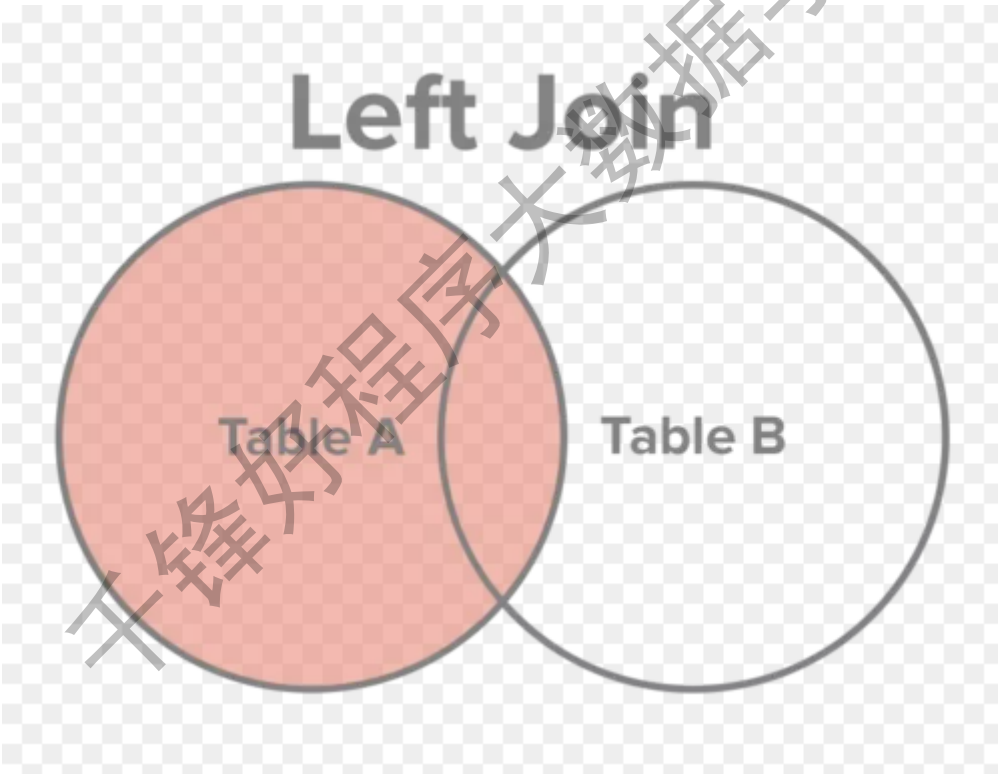
2.5.23 Join的语法与特点

在关系型数据库里面，每个实体有自己的一张表（table），所有属性都是这张表的字段（field），表与表之间根据关联字段"连接"（join）在一起。所以，表的连接是关系型数据库的核心问题。

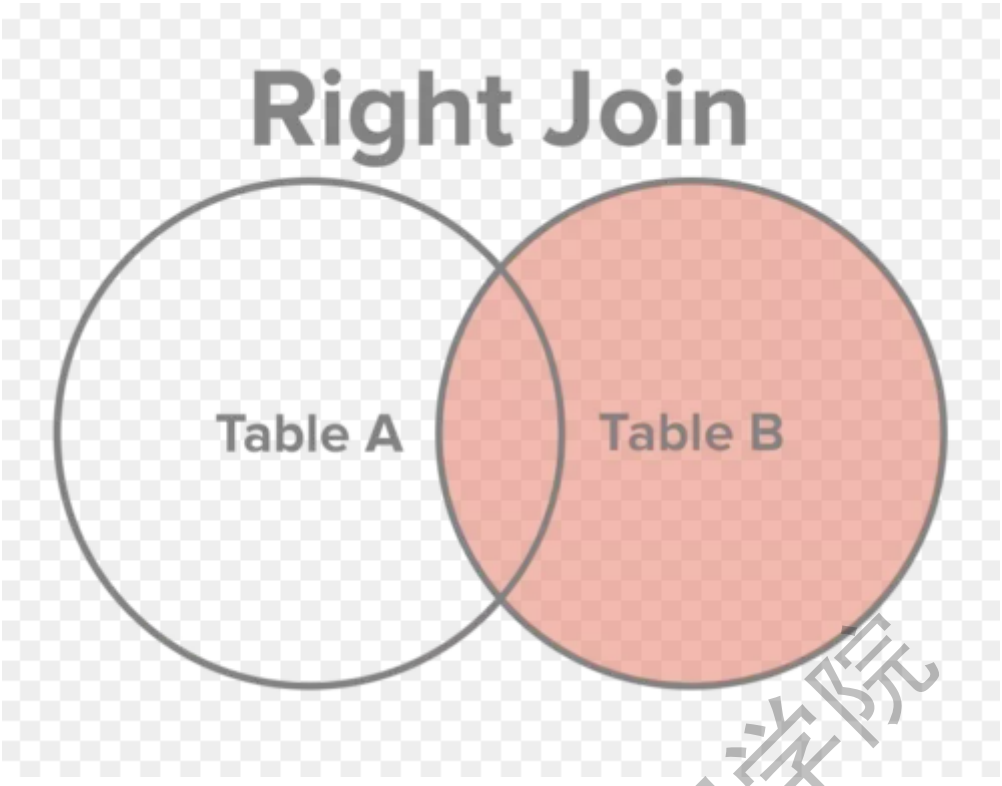
表的连接分成好几种类型。
内连接（inner join）
外连接（outer join）
左连接（left join）
右连接（right join）
全连接（full join）

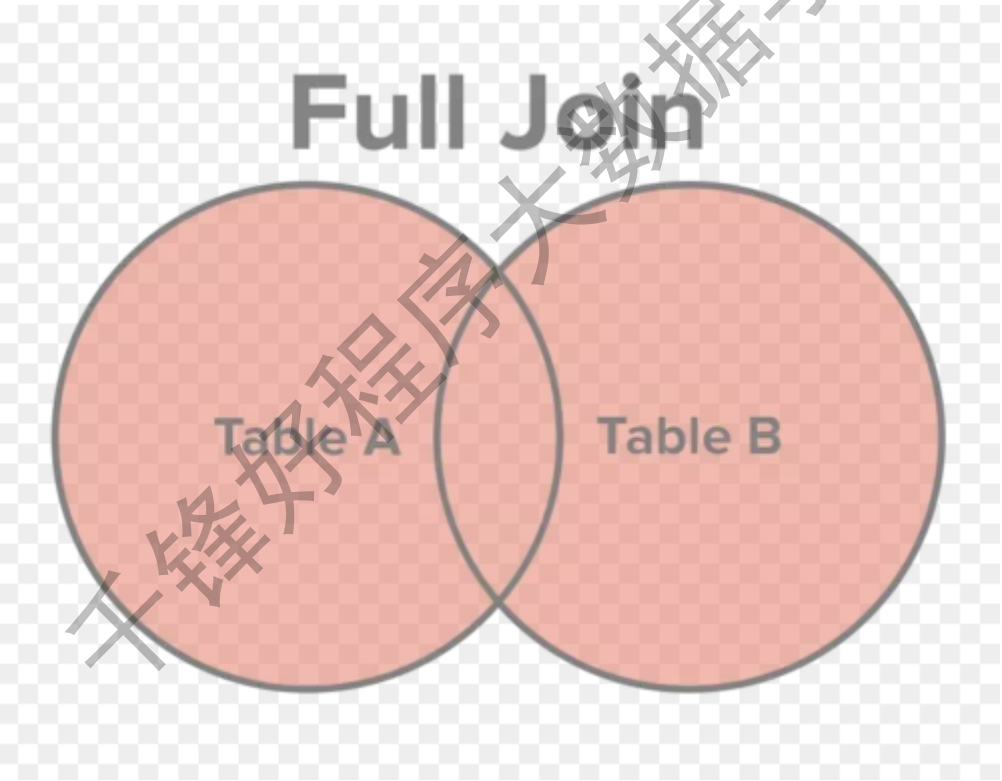
以前，很多文章采用维恩图（两个圆的集合运算），解释不同连接的差异。





千锋好程序大数据学院





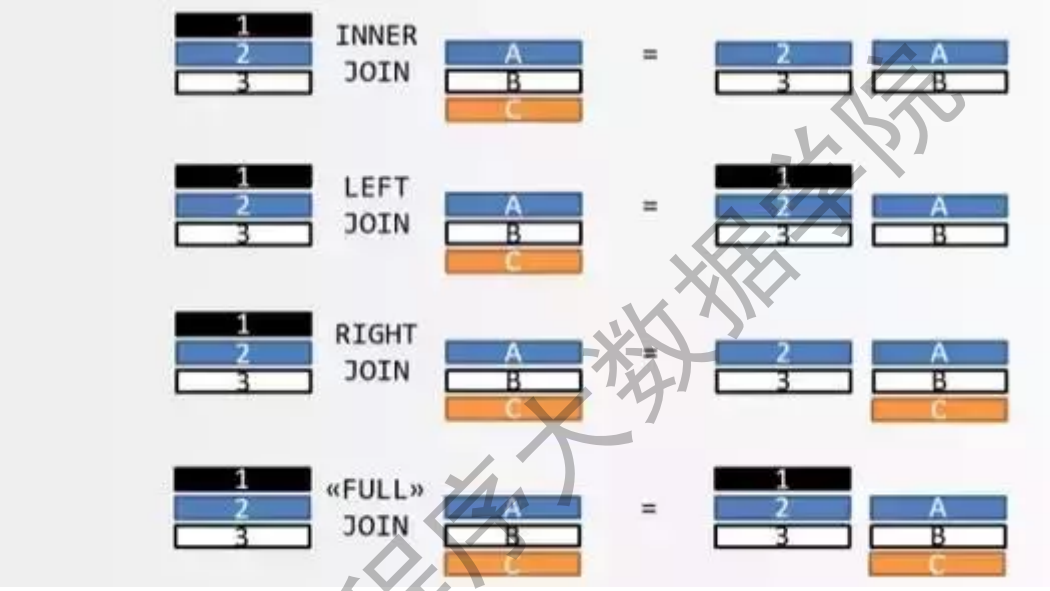
千锋好程序大数据学院

上周，我读到一篇文章，认为还有比维恩图更好的解释方式。我发现确实如此，换一个角度解释，更容易懂。

所谓"连接"，就是两张表根据关联字段，组合成一个数据集。问题是，两张表的关联字段的值往往是不一致的，如果关联字段不匹配，怎么处理？比如，表 A 包含张三和李四，表 B 包含李四和王五，匹配的只有李四这一条记录。

很容易看出，一共有四种处理方法。
只返回两张表匹配的记录，这叫内连接（inner join）。
返回匹配的记录，以及表 A 多余的记录，这叫左连接（left join）。
返回匹配的记录，以及表 B 多余的记录，这叫右连接（right join）。
返回匹配的记录，以及表 A 和表 B 各自的多余记录，这叫全连接（full join）。

下图就是四种连接的图示。我觉得，这张图比维恩图更易懂，也更准确。

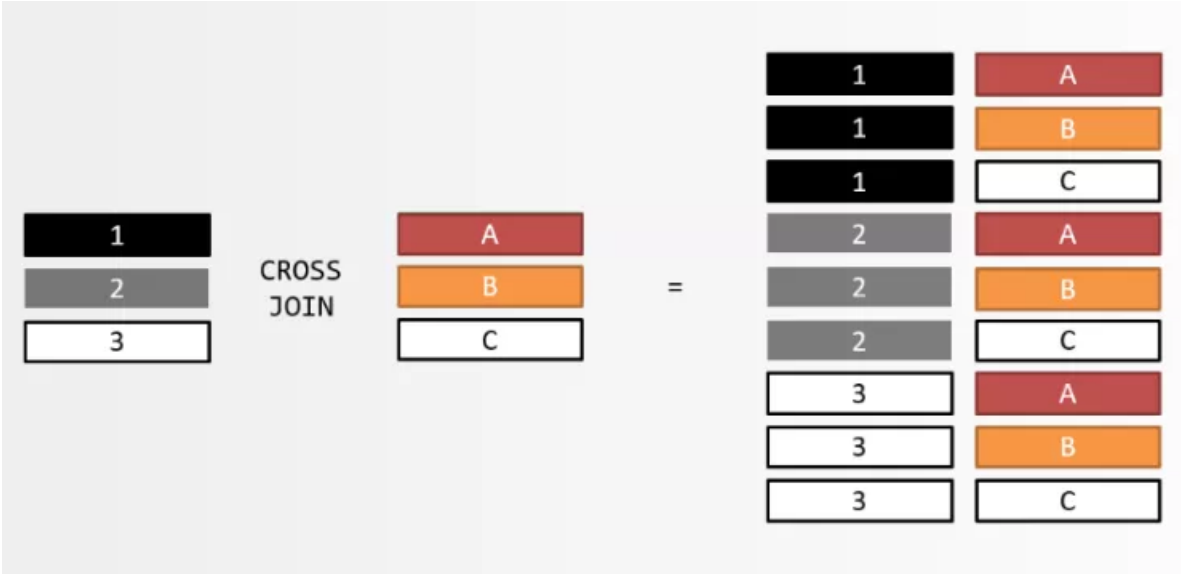


上图中，表 A 的记录是 123，表 B 的记录是 ABC，颜色表示匹配关系。返回结果中，如果另一张表没有匹配的记录，则用 null 填充。

这四种连接，又可以分成两大类：内连接（inner join）表示只包含匹配的记录，外连接（outer join）表示还包含不匹配的记录。所以，左连接、右连接、全连接都属于外连接。

此外，还存在一种特殊的连接，叫做"交叉连接"（cross join），指的是表 A 和表 B 不存在关联字段，这时表 A（共有 n 条记录）与表 B （共有 m 条记录）连接后，会产生一张包含 n x m 条记录的新表（见下图）。

干锋好程序大数据学院



##2.5.24 Join案例-语句特点演示

```
类型：(left join\left outer join) \ right join \right outer join \inner join \
full outer join

特殊类型：
left semi join
```

2/例子：

```
准备数据
1,a
2,b
3,c
4,d
7,y
8,u

2,bb
3,cc
7,yy
9,pp
```

建表：

```
create table if not exists u1(
id int,
name string
)
row format delimited fields terminated by ','
;

create table if not exists u2(
id int,
name string
)
row format delimited fields terminated by ','
;
```

千锋好程序大数据学院

```
load data local inpath '/home/hivedata/u1.txt' into table u1;
load data local inpath '/home/hivedata/u2.txt' into table u2;
```

join 、 inner join 、 多表用逗号分开 ：内连接，三者差不太多，相互连接上才能出来结果。

join : 不加任何的on 或者是where过滤条件时，称之为笛卡尔积。

```
select *
from u1
join u2 on u1.id =u2.id;

select *
from u1
inner join u2 on u1.id =u2.id;

select *
from u1,u2
where u1.id=u2.id;
```

left join/left outer join/left semi join :
数据以左表的数据为准，左表存在的数据都查询出，右表的数据关联上就出来，关联不上以NULL代替

```
select *
from u1
left join u2 on u1.id =u2.id;

select *
from u1
left outer join u2 on u1.id =u2.id;
```

上面的 SQL 语句还可以加上where条件从句，对记录进行筛选，比如只返回表 A 里面不匹配表 B 的记录。

left join 从hive0.8版本开始有
left join 和 left outer join 几乎差不多

right join /right outer join

```
select *
from u1
right join u2 on u1.id =u2.id;

select *
from u1
right outer join u2 on u1.id =u2.id;
```

以右表为准，来匹配左表信息，如果匹配不上，使用NULL来代替。
hive不支持right semi join：
这两个得到的结果也是一样的。

千锋好程序大数据学院

```
full outer join
(相互进行连接，如果有一张表的数据连接不上来使用NULL来代替)
select *
from u1
full outer join u2 on u1.id =u2.id;

另一个例子，返回表 A 或表 B 所有不匹配的记录。
```

2.5.25 Hive专有Join特点

```
select *
from u1
left semi join u2 on u1.id =u2.id;

select u1.* ,u2.*
from u1
left semi join u2 on u1.id =u2.id;

select u1.*
from u1
where exists (select 1 from u2 where u2.id =u1.id);

left semi join 叫做半开连接，通常是left join的一种优化，只能查询左表的信息，然后主要解决hive中存在不不存在的问题。
```

2.5.25.1 hive小表标识

```
hive提供小表标识：使用的是STREAMTABLE(小表别名)
select
/+STREAMTABLE(d)/
d.name,
e.name
from u1 e
join u2 d
on d.id = e.id
;
```

2.5.25.2 子查询

```
hive对子查询支持不是很友好，特别是 "="问题较多
select
e.*
from emp e
where e.deptno = (
select
d.deptno
from dept d
limit 1
)
;

select
e.*
from emp e
```

千锋好程序大数据学院

```
where e.deptno in (
select
d.deptno
from dept d
)
;
```

```
inner join 和outer join的区别:
分区字段对outer join 中的on条件是无效，对inner join 中的on条件有效

有inner join 但是没有full inner join
有full outer join但是没有outer join
所有join连接，只支持等值连接(= 和 and )。不支持 != 、 < 、> 、 <> 、>=、 <= 、or
```

2.5.25.3 map-side join

```
map-side join:
如果所有的表中有小表，将会把小表缓存内存中，然后在map端进行连接关系查找。hive在map端
查找时将减小查询量，从内存中读取缓存小表数据，效率较快，还省去大量数据传输和shuffle耗时。

注意看该属性:
set hive.auto.convert.join=true
select
e.*
from u1 d
join u2 e
on d.id = e.id
;
```

```
以前的老版本，需要添加(/+MAPJOIN(小表名)/)来标识该join为map端的join。hive 0.7以后hive已经
废弃，但是仍然管用:
????需要再测试看看是否有效???
select
/+MAPJOIN(d)/
e.*
from u1 d
join u2 e
on d.id = e.id
;
到底小表多大才会被转换为map-side join:
set hive.mapjoin.smalltable.filesize=25000000    约23.8MB
```

```
on ：  所有on只支持等值连接。

where ：  where后面通常是表达式 、还可以是非聚合函数表达式(但是不能是聚合函数表达式)

select
d.*
from dept d
where  length(d.dname) > 5
;
```

###2.5.25.4 表达式别名

千锋好程序大数据学院

```
select
d.dname,
length(d.dname) as ds
from dept d
;
```

2.5.26 Where语句特点

where后不能跟聚合函数

```
select
e.deptno,
count(e.deptno) ct
from emp e
where  count(e.deptno) > 3
group by e.deptno
;
```

2.5.27 Group By语句特点

```
group by ：  分组，通常和聚合函数搭配使用
查询的字段要么出现在group by 后面，要么出现在聚合函数里面
select
e.deptno,
count(e.ename) ct
from emp e
group by e.deptno
;
```

2.5.28 Having语句特点

```
对分组以后的结果集进行过滤。
select
e.deptno,
count(e.deptno) ct
from emp e
group by e.deptno
having ct > 3
;
```

2.5.29 Limit语句特点

```
limit ：  从结果集中取数据的条数
将set hive.limit.optimize.enable=true 时，limit限制数据时就不会全盘扫描，而是根据限制的数量进行抽样。

同时还有两个配置项需要注意：
hive.limit.row.max.size         这个是控制最大的抽样数量
hive.limit.optimize.limit.file 这个是抽样的最大文件数量
```

千锋好程序大数据学院

2.5.30 Cluster By分桶查询

cluster by ：兼有distribute by以及sort by的升序功能。
排序只能是升序排序（默认排序规则），不能指定排序规则为asc 或者desc。

2.5.31 分区排序Distribute By

distribute by ： 根据by后的字段和reducer个数，决定map的输出去往那个reducer。
默认使用查询的第一列的hash值来决定map的输出去往那个reducer。如果reducer的个数为1时没有任何体现。

sort by:局部排序，只保证单个reducer有顺序。
order by：全局排序，保证所有reducer中的数据都是有顺序。
如果reduser个数只有一个，两者都差不多。
两者都通常和 desc 、 asc 搭配。默认使用升序asc。

order by的缺点：
由于是全局排序，所以所有的数据会通过一个Reducer 进行处理，当数据结果较大的时候，一个Reducer进行处理十分影响性能。
注意事项：
当开启MR 严格模式的时候ORDER BY 必须要设置 LIMIT 子句 ，否则会报错

手动设置reducer个数：

```
set mapreduce.job.reduces=3;
select
e.empno
from emp e
order by e.empno desc
;
```

只要使用order by ，reducer的个数将是1个。

如果sort by 和 distribute by 同时出现：那个在前面？？

如果sort by 和 distribute by 同时出现，并且后面的字段一样、sort by使用升序时 <==> cluster by 字段

union ：将多个结果集合并，去重，排序
union all ：将多个结果集合并，不去重，不排序。

```
select
d.deptno as deptno,
d.dname as dname
from dept d
union
select
e.deptno as deptno,
e.ename as dname
from emp e
;

select
d.deptno as deptno,
d.dname as dname
from dept d
```

千锋好程序大数据学院

```
union all
select
d.dname as dname,
d.deptno as deptno
from dept d
;

单个union 语句不支持：orderBy、clusterBy、distributeBy、sortBy、limit
单个union语句字段的个数要求相同，字段的顺序要求相同。

distinct : 去重
```

##2.5.32 数据类型

hive的数据类型分为基础数据类型和复杂数据类型：

2.5.32.1 基础数据类型

```
tinyint     1   -128~127
smallint    2   -2的15 ~ 2的15-1
int         4
bigint      8
float       4
double      8
boolean     1
string
binary          字节
timestamp       2017-06-02 11:36:22

java中有的而hive中没有的：
long
char
short
byte

create table if not exists bs1(
id1 tinyint,
id2 smallint,
id3 int,
id4 bigint,
sla float,
sla1 double,
isok boolean,
content binary,
dt timestamp
)
row format delimited fields terminated by '\t'
;

233 12 342523 455345345 30000 60000 nihao helloworld 2017-06-02
126 13 342526 455345346 80000 100000 true helloworld1 2017-06-02 11:41:30

load data local inpath '/home/hivedata/bs' into table bs1;
```

2.5.32.2 复杂的数据类型

千锋好程序大数据学院

```
array : col array<基本类型> ,下标从0开始，越界不报错，以NULL代替
map   : column map<string,string>
struct: col struct
```

2.5.32.2.1 array示例

```
zhangsan    78,89,92,96
lisi    67,75,83,94

create table if not exists arr1(
name string,
score array<String>
)
row format delimited fields terminated by '\t'
;

注意terminated顺序？？
create table if not exists arr2(
name string,
score array<String>
)
row format delimited fields terminated by '\t'
collection items terminated by ','
;

导入数据
load data local inpath '/home/hivedata/arr1.txt' into table arr1;

查询：
select * from arr1;
select name,score[1] from arr2 where size(score) > 3;

扩展查询：
zhangsan        90
zhangsan        87
zhangsan        63
zhangsan        76

内嵌查询：
explode:展开
select explode(score) score from arr1;

lateral view : 虚拟表
select name,cj from arr1 lateral view explode(score) score as cj;

统计每个学生的总成绩：
select name,sum(cj) as totalscore from arr1 lateral view explode(score) score as
cj group by name;

如何往array字段写入数据：
准备数据：
create table arr_temp
as
select name,cj from arr1 lateral view explode(score) score as cj;

1、collect_set函数：
```

千锋好程序大数据学院

```
create table if not exists arr2(
name string,
score array<int>
)
row format delimited fields terminated by ' '
collection items terminated by ','
;

将数据写成array格式：
insert into arr2
select name,collect_set(cj) from arr_temp group by name;

2、array
课后练习，表结构如下：
name chinesescore mathscore englishscore
将数据转换成array格式：
select array(chinesescore,mathscore,englishscore) from table1;
```

2.5.32.2.2 map示例

```
zhangsan chinese:90,math:87,english:63,nature:76
lisi chinese:60,math:30,english:78,nature:0
wangwu chinese:89,math:25,english:81,nature:9

create table if not exists map2(
name string,
score map<string,int>
)
row format delimited fields terminated by ' '
collection items terminated by ','
map keys terminated by ':'
;

加载数据

load data local inpath '/home/hivedata/map1.txt' into table map2;

查询：
查询数学大于35分的学生的英语和自然成绩：
select
m.name,
m.score['english'] ,
m.score['nature']
from map2 m
where m.score['math'] > 35
;

展开数据：
explode
select explode(score) as (m_class,m_score) from map2;

使用lateral view explode结合查询：
select name,m_class,m_score from map2 lateral view explode(score) score as
m_class,m_score;

怎么将数据写入到map字段：
str_to_map:
```

千锋好程序大数据学院

```
map:

select province,concat(m_name,':',m_score) as score from map_temp;

select province,str_to_map(concat(m_name,':',m_score),',',':') as score from
map_temp;

create table map_temp(
name string,
score1 int,
score2 int,
score3 int
)
row format delimited fields terminated by ','
;

用的arr的数据修改版
create table if not exists map3(
name string,
score map<string,int>
)
row format delimited fields terminated by ' '
collection items terminated by ','
map keys terminated by ':'
;

导入数据：
insert into map3
select name,map('chinese',score1,'math',score2,'english',score3) from map_temp;
```

####2.5.32.2.3 struct

```
create table if not exists str1(
name string,
score structchinese:int,math:int,english:int
)
row format delimited fields terminated by ' '
collection items terminated by ','
;

导入数据：
load data local inpath '/home/hivedata/arr1.txt' into table str1;

查询数据：
查询数学大于35分的学生的英语和语文成绩：
select name,
score.english,
score.chinese
from str1
where score.math > 35
;
```

####2.5.32.2.4 复杂数据类型案例

```
uid uname belong tax addr
1 xdd ll,lw,lg,lm wuxian:300,gongjijin:1200,shebao:300 北京,西城区,中南海
```

千锋好程序大数据学院

```
2 lkq lg,lm,lw,ll,mm wuxian:200,gongjijin:1000,shebao:200 河北,石家庄,中山路

查询：下属个数大于4个，公积金小于1200，省份在河北的数据

create table if not exists ss(
id int,
name string,
belong array<string>,
tax map<string,double>,
addr struct<province:string,city:string,road:string>
)
row format delimited fields terminated by ' '
collection items terminated by ','
map keys terminated by ':'
stored as textfile
;

导入数据

load data local inpath '/home/hivedata/ss.txt' into table ss;

查询：下属个数大于4个，公积金小于1200，省份在河北的数据
select ss.id,
ss.name,
ss.belong[0],
ss.belong[1],
ss.tax['wuxian'],
ss.tax['shebao'],
ss.addr.road
from ss
where size(ss.belong) > 4 and
ss.tax['gongjijin'] < 1200 and
ss.addr.province = '河北'
;
```

2.5.32.2.5 嵌套数据类型

```
嵌套??
所有元素分割符自己调（搜：hive的map类型处理）

hive共支持8个层级的分隔符，依次是：
\001,\002,\003,...\008
map嵌套使用
uid uname belong tax addr
1   xdd wuxian:(300,300),gongjijin:1200,1500,shebao:300,
2   lkq wuxian:200,300,gongjijin:1000,1200,shebao:200

create table qt(
id int,
name string,
addr map<string,array<string>>
)
row format delimited fields terminated by '\t'
collection items terminated by ','
map keys terminated by ':'
;
```

千锋好程序大数据学院

##2.5.33 系统内置函数介绍
查看

```
show functions;
desc function array;
```

##2.5.34 排名函数（窗口函数）详解

2.5.34.1 排名函数

```
row_number()：没有并列，相同名次依顺序排
rank()：有并列，相同名次空位
dense_rank()：有并列，相同名次不空位
```

准备数据

```
多次的考试成绩
```

测试

```
create table if not exists stu_score(
dt string,
name string,
score int
)
row format delimited
fields terminated by '\t'
;

load data local inpath '/hivedata/stu_score.txt' into table stu_score;
```

```
需求：对每次考试按照考试成绩倒序
select *
from stu_score
order by dt,score desc
;

需求：获取每次考试的排名情况
over开窗子句：
dt  name    score   rn
select
dt,
name,
score,
row_number() over(distribute by dt sort by score desc) rn
from stu_score
;

需求：求每次考试的前三名
select *
from (
select
dt,
name,
```

千锋好程序大数据学院

```
score,
row_number() over(distribute by dt sort by score desc) rn
from stu_score
) a
where a.rn < 4
;

select *
from (
select
dt,
name,
score,
row_number() over(distribute by dt sort by score desc) rn,
rank() over(distribute by dt sort by score desc) rk,
dense_rank() over(distribute by dt sort by score desc) drk
from stu_score
) a
where a.rk < 6
;

select *
from (
select
dt,
name,
score,
row_number() over(partition by dt order by score desc) rn,
rank() over(partition by dt order by score desc) rk,
dense_rank() over(partition by dt order by score desc) drk
from stu_score
) a
where a.rk < 6
;
```

2.5.34.2 first_value和last_value

```
需求：求每次考试的最高分、最低分
select *
from stu_score a
join (
select
dt,
max(score) as maxscore,
min(score) as minscore
from stu_score
group by dt
) b on b.dt =a.dt
;

需求：求每次考试每位学员的成绩与最好成绩的差值
select
dt,
name,
score,
max(score) over(distribute by dt sort by score desc) maxscore
from stu_score
```

千锋好程序大数据学院

```
;

第一个版本、最近一个版本
first_value:
last_value:
需求：求每位学员的第一次考试成绩和最后一次考试成绩
求每次考试每位学员的成绩与最好成绩的差值
select
dt,
name,
score,
first_value(score ignore nulls) over(distribute by dt sort by score desc)
maxscore,
last_value(score ignore nulls) over(distribute by dt sort by score desc rows
between unbounded preceding and unbounded following) minscore
from stu_score
;
window子句
last_value默认行的范围从第一行到当前行，如果要让其取值范围是所有分组数据，则需要使用rows
between unbounded preceding and unbounded following
window子句：
rows between unbounded preceding and unbounded following

需求：求第一个非空的版本
在first_value和last_value中加上ignore nulls
```

2.5.34.3 lag和lead

```
需求：求每位学员的每次考试的成绩与上一次的成绩对比
lag(score,2)   -- 取出前n行的数据
lead(score,2)  -- 取出后n行的数据
select
dt,
name,
score,
lag(score,1) over(distribute by name sort by dt asc) as upscore
from stu_score
;
```

2.5.34.4 练习

```
黑名单
数据：
dt  id  url
2019-04-03 11:09:11 1 www.sina.com
2019-04-03 11:10:11 1 www.baidu.com
.....

需求：求出5分钟内访问次数到达100的用户（求黑名单）

解决：使用lag
lag(dt,100) over(distribute by id sort dt asc)
```

##2.5.35 自定义函数概念

千锋好程序大数据学院

###2.5.35.1 为什么需要自定义函数

```
hive的内置函数满足不了所有的业务需求。
hive提供很多的模块可以自定义功能，比如：自定义函数、serde、输入输出格式等。
```

###2.5.35.2 常见自定义函数有哪些

```
udf：用户自定义函数，user defined function。一对一的输入输出。（最常用的）。
udaf：用户自定义聚合函数。user defined aggregate function。多对一的输入输出。
udtf：用户自定义表生成函数。user defined table-generate function.一对多的输入输出。
```

2.5.35.3 编写udf的方式

```
1、继承UDF，重写evaluate()，允许重载。（常用）
2、继承genericUDF，重写initlizer（）、getdisplay（）、evaluate()。
```

2.5.36 自定义函数第一个案例

```
public class firstUDF extends UDF {
    public String evaluate(String str){
        String upper = null;
        //1、检查输入参数
        if (StringUtils.isEmpty(str)){

        } else {
            upper = str.toUpperCase();
        }

        return upper;
    }

    //调试自定义函数
    public static void main(String[] args){
        System.out.println(new firstUDF().evaluate("jiajingwen"));
    }
}
```

千锋好程序大数据学院

##2.5.37 UDF使用方式介绍

###2.5.37.1 第一种（本session有效）

1、将编写的udf的jar包上传到服务器上，并且将jar包添加到hive的class path中

add jar /root/1701Demo-0.0.1.jar;

```
<name>hive.aux.jars.path</name>
<value>$HIVE_HOME/auxlib</value>
<description>The location of the plugin jars that contain implementations of
user defined functions and serdes.</description>

2、创建一个临时函数名：
create temporary function tolow as 'edu.qianfeng.GP1704.hiveUDF.FirstUDF';

3、检查函数是否创建成功
```

```
show functions;

4、测试功能
小技巧：
    create table dual(id string);
    insert into dual values (' ');

设置hive是否跑本地模式
      <name>hive.exec.mode.local.auto</name>
      <value>false</value>
      <description>Let Hive determine whether to run in local mode
automatically</description>

5、测试代码：
    select tolow('GAOYUANYUAN') from dual;
6、使用完毕，确定没有再调用该函数，可以注销(小心)
drop temporary function if exists tolow;
```

###2.5.37.2 第二种(也是在本session有效，临时函数)

1、将编写的udf的jar包上传到服务器上

2、创建配置文件

vi ./hive-init

add jar /root/1701Demo-0.0.1.jar;

create temporary function tolow as 'edu.qianfeng.GP1704.hiveUDF.FirstUDF';

```
3、启动hive的时候带上初始化文件：
hive -i ./hive-init
```

###2.5.37.3 第三种

1、将编写的udf的jar包上传到服务器上

2、在hive的安装目录的bin目录下创建一个配置文件，文件名：.hiverc

vi ./bin/.hiverc

add jar /root/1701Demo-0.0.1.jar;

create temporary function tolow as 'edu.qianfeng.GP1704.hiveUDF.FirstUDF';

3、启动hive

hive

###2.5.37.4 编译源码(费劲)

1)将写好的Jave文件拷贝到~/install/hive-0.8.1/src/ql/src/java/org/apache/hadoop/hive/ql/udf/

cd ~/install/hive-0.8.1/src/ql/src/java/org/apache/hadoop/hive/ql/udf/

ls -lhgt |head

2)修改~/install/hive-0.8.1/src/ql/src/java/org/apache/hadoop/hive/ql/exec/FunctionRegistry.java，增加import和RegisterUDF

千锋好程序大数据学院

```
import com.meilishuo.hive.udf.UDFIp2Long;   //添加import
registerUDF("ip2long", UDFIp2Long.class, false); //添加register

3)在~/install/hive-0.8.1/src下运行ant -Dhadoop.version=1.0.1 package
cd ~/install/hive-0.8.1/src
ant -Dhadoop.version=1.0.1 package

4)替换exec的jar包，新生成的包在/hive-0.8.1/src/build/ql目录下,替换链接
cp hive-exec-0.8.1.jar /hadoop/hive/lib/hive-exec-0.8.1.jar.0628
rm hive-exec-0.8.1.jar
ln -s hive-exec-0.8.1.jar.0628 hive-exec-0.8.1.jar
5)重启进行测试
```

2.5.38 生日转换成年龄

传入参数：2000-7-12
输出参数：17

代码如下：

```
public class birthday2Age extends UDF {
    public int evaluate(String birth){
        //1、判断参数
        if (StringUtils.isEmpty(birth)){
            return -1;
        }

        //拆分生日，获取年月日
        String[] birthdays = birth.split("-");

        //得到生日年月日
        int birthYear = Integer.parseInt(birthdays[0]);
        int birthMonth = Integer.parseInt(birthdays[1]);
        int birthDay = Integer.parseInt(birthdays[2]);

        //获取当前时间
        Calendar calendar = Calendar.getInstance();

        int nowYear = calendar.get(Calendar.YEAR);
        int nowMonth = calendar.get(Calendar.MONTH) + 1;
        int nowDay = calendar.get(Calendar.DAY_OF_MONTH);

        //计算年龄
        int age = nowYear - birthYear;

        //判断月份和日期
        if(nowMonth < birthMonth){
            age -=1;
        } else if (nowMonth == birthMonth && nowDay < birthDay){
            age-=1;
        }
        return age;
    }

    public static void main(String[] args){
        System.out.println(new birthday2Age().evaluate("1980-03-31"));
    }
```

千锋好程序大数据学院

```
}
```

打包并使用，也可直接在开发环境测试

```
add jar /root/1701Demo-0.0.1.jar;
create temporary function birthday2Age as
'edu.qianfeng.GP1704.hiveUDF.Birthday2Age';
```

2.5.39 根据Key查找Value

```
数据如：sex=1&hight=180&weight=130&sal=28000
输入：key值 sex
输出：返回value值 sex对应的1
json:
{'sex':1,'hight':180}
JSONobject
```

代码如下

```
public class key2Value extends UDF {
    public String evaluate(String str,String key) throws JSONException {
        //1、判断参数
        if (StringUtils.isEmpty(str) || StringUtils.isEmpty(key)){
            return null;
        }

        //将str转换成json格式
        //sex=1&hight=180&weight=130&sal=28000
        //{'sex':1,'hight':180,'weight':130,'sal':28000}
        String s1 = str.replace("&",",");
        String s2 = s1.replace("=",":");
        String s3 = "{" +s2 + "}";

        //使用尽送对象解析json串
        JSONObject jo = new JSONObject(s3);
        return jo.get(key).toString();
    }

    public static void main(String[] args) throws JSONException {
        System.out.println(new
key2Value().evaluate("sex=1&hight=180&weight=130&sal=28000&faceId=189","faceId")
);
    }
}
```

2.5.40 正则表达式解析日志

千锋好程序大数据学院

```
解析前：
220.181.108.151 - - [31/Jan/2012:00:02:32 +0800] "GET /home.php?
mod=space&uid=158&do=album&view=me&from=space HTTP/1.1" 200 8784 "-"
"Mozilla/5.0 (compatible; Baiduspider/2.0;
+http://www.baidu.com/search/spider.html)"
解析后：
220.181.108.151 20120131 120232 GET /home.php?
mod=space&uid=158&do=album&view=me&from=space HTTP    200 Mozilla
```

代码如下

```
public class logParser extends UDF {
    public String evaluate(String log) throws ParseException {
        //1
        if(StringUtils.isEmpty(log)){
            return null;
        }

        //220.181.108.151 - - [31/Jan/2012:00:02:32 +0800] \"GET /home.php?
mod=space&uid=158&do=album&view=me&from=space HTTP/1.1\" 200 8784 \"-\"
\"Mozilla/5.0 (compatible; Baiduspider/2.0;
+http://www.baidu.com/search/spider.html)\"
        //220.181.108.151   20120131 120232 GET /home.php?
mod=space&uid=158&do=album&view=me&from=space HTTP    200 Mozilla

        //定义一个正则表达式
        String reg = "^([0-9.]+\\d+) - - \\[(.* \\+\\d+)\\] .+(GET|POST) (.+)
(HTTP)\\S+ (\\d+) .+\\\"(\\w+).+$";

        //获取一个模式匹配器
        Pattern pattern = Pattern.compile(reg);

        //匹配结果
        Matcher matcher = pattern.matcher(log);

        //
        StringBuffer sb = new StringBuffer();

        //判断数据是否匹配上
        if (matcher.find()){
            //先获取匹配的段数
            int count = matcher.groupCount();
            //循环获取每段的内容，并且将内容拼接起来
            for (int i = 1; i <= count; i++) {
                //判断字段是否是时间字段，如果是，则做时间格式转换
                if (i == 2){
                    //定义一个时间格式来解析当前时间
                    Date d = new SimpleDateFormat("dd/MMM/yyyy:HH:mm:ss Z",
Locale.ENGLISH).parse(matcher.group(i));

                    //定义输出的时间格式
                    SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd
hhmmss");

                    //将时间格式转换成输出格式，并输出
                    sb.append(sdf.format(d) + "\t");
```

千锋好程序大数据学院

```
                } else {
                    sb.append(matcher.group(i) + "\t");
                }
            }
        }
        return sb.toString();
    }

    public static void main(String[] args) throws ParseException {
        System.out.println(new logParser().evaluate("220.181.108.151 - -
[31/Jan/2012:00:02:32 +0800] \\\"GET /home.php?
mod=space&uid=158&do=album&view=me&from=space HTTP/1.1\\\" 200 8784 \\\"-\\\"
\\\"Mozilla/5.0 (compatible; Baiduspider/2.0;
+http://www.baidu.com/search/spider.html)\\\""));
    }
}
```

2.5.41 Json数据解析UDF开发

```
有原始json数据如下：
{"movie":"1193","rate":"5","timeStamp":"978300760","uid":"1"}
{"movie":"661","rate":"3","timeStamp":"978302109","uid":"1"}
{"movie":"914","rate":"3","timeStamp":"978301968","uid":"1"}
{"movie":"3408","rate":"4","timeStamp":"978300275","uid":"1"}
{"movie":"2355","rate":"5","timeStamp":"978824291","uid":"1"}
{"movie":"1197","rate":"3","timeStamp":"978302268","uid":"1"}
{"movie":"1287","rate":"5","timeStamp":"978302039","uid":"1"}

需要将数据导入到hive数据仓库中
我不管你中间用几个表，最终我要得到一个结果表：
movie   rate    timestamp   uid
1197    3   978302268   1
```

步骤

1、将原始数据导入到hive库，先创建一个单字段的表

```
create table if not exists t_json(json string);
load data local inpath '/hivedata/rating.json' into table t_json;
```

2、写一个自定义函数，利用自定义函数解析json，将json串解析一个以\t分割的字符串，并上传使用

```
public class jsonParser extends UDF{
    private Logger logger = Logger.getLogger(jsonParser.class);

    public String evaluate(String json) {
        //判断传入参数
        if(Strings.isNullOrEmpty(json)){
            return null;
        }

        ObjectMapper objectMapper = new ObjectMapper();

        try {
```

千锋好程序员教育学院

```
            MovieRateBean bean = objectMapper.readValue(json,
MovieRateBean.class);
            return bean.toString();
        } catch (IOException e) {
            logger.error("解析json串失败！！",e);
        }

        return null;
    }

    public static void main(String[] args) {
        System.out.println(new jsonParser().evaluate("
{\"movie\":\"1193\",\"rate\":\"5\",\"timeStamp\":\"978300760\",\"uid\":\"1\"}"))
;
    }
}
```

3、insert into select func()将数据插入临时表

```
create table if not exists t_json_tmp
as
select jsonParser(json) as jsonLine from t_json
;
```

4、用split函数从临时表中解析出字符串中各个字段存入到最终结果表

```
create table if not exists t_movieRate
as
select split(jsonLine,'\t')[0] as movie,split(jsonLine,'\t')[1] as
rate,split(jsonLine,'\t')[2] as times,split(jsonLine,'\t')[3] as uid
from t_json_tmp
;

内置的json解析函数：
select get_json_object(json,'$.movie') as movie from t_json limit 10;
```

2.5.42 Transform实现UDF功能

```
Hive的 TRANSFORM 关键字提供了在SQL中调用自写脚本的功能
适合实现Hive中没有的功能又不想写UDF的情况

1、需求：将上述数据中的timestamp类型的数据转换成weekday

2、写一个python脚本，实现时间转换功能
vi  /root/weekday.py

!/bin/python
import sys
import datetime

for line in sys.stdin:
  line = line.strip()
  movie, rate, times,uid = line.split('\t')
  weekday = datetime.datetime.fromtimestamp(float(times)).isoweekday()
  print '\t'.join([movie, rate, str(weekday),uid])
```

千锋好程序大数据学院

```
3、将python脚本加入hive的classpath
add file /root/weekday.py;

4、定义一个结果表用户接收转换的结果
  CREATE TABLE u_data_new (
  movieid INT,
  rating INT,
  weekday INT,
  userid INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t';

5、使用transform+python查询转换
insert into u_data_new
select TRANSFORM(movie,rate,times,uid)
USING 'python weekday.py'
AS (movieid,rating,weekday,userid)
from t_movieRate
;
```

2.5.43 函数实战：级联求和

```
级联求和accumulate
create table t_access_times(username string,month string,salary int)
row format delimited fields terminated by ',';

load data local inpath '/home/hivedata/access.dat' into table t_access_times;

A,2015-01,5
A,2015-01,15
B,2015-01,5
A,2015-01,8
B,2015-01,25
A,2015-01,5
A,2015-02,4
A,2015-02,6
B,2015-02,10
B,2015-02,5

最终要得出的结果集：
访客   月份   月访问总计    累计访问总计
A   2015-01 33  33
A   2015-02 10  43
……. ……. ……. …….
B   2015-01 30  30
B   2015-02 15  45
……. ……. ……. …….

思路1：表自连接
第一步：先统计每个用户每个月的总的访问次数
create table if not exists t_access_total
as
select
username,
month,
```

千锋好程序大数据学院

```
sum(salary) as totalcount
from t_access_times
group by username,month
;
```

第二步：表本身关联，以username做为关联条件

```
select a.username as aname,
a.month as amonth,
a.totalcount as acount,
b.username as bname,
b.month as bmonth,
b.totalcount as bcount
from t_access_total a,
t_access_total b
where a.username = b.username
;
```

结果集：

```
A       2015-01 33      A       2015-01 33
A       2015-02 10      A       2015-01 33
A       2015-01 33      A       2015-02 10
A       2015-02 10      A       2015-02 10
B       2015-01 30      B       2015-01 30
B       2015-02 15      B       2015-01 30
B       2015-01 30      B       2015-02 15
B       2015-02 15      B       2015-02 15
```

第三步：根据上面结果分析出，要得出最后的结果集，应该以username相等，并且前面的日期小于等于后面日期，然后前面的数量累加

```
select a.bname,
a.bmonth,
a.bcount,
sum(a.acount) as totalcount
 from (select a.username as aname,
a.month as amonth,
a.totalcount as acount,
b.username as bname,
b.month as bmonth,
b.totalcount as bcount
from t_access_total a,
t_access_total b
where a.username = b.username) a
where a.aname=a.bname and
a.amonth <=a.bmonth
group by a.bname,a.bmonth,a.bcount
;
```

思路2、窗口函数

```
sum(amount) over(distribute by fangke sort by dt asc rows between unbounded
preceding and current row)
```

千锋好程序大数据学院

2.5.44 数据导入

1、从本地文件系统中导入hive表
2、从hdfs文件系统中导入hive表
3、从hive的一个表中导入到另一个表

```
4、直接将数据copy到hive表目录
5、location
6、克隆带数据
7、多表导入
8、CTAS

hive不允许局部数据操作(增、删、改)。

CREATE TABLE text1(
uid int,
uname string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
;

create table text2 (
uid int,
uname string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
;

create table text3 (
uname string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
;

7、多表导入:一次扫描源表，多次导入
from text1
insert into text2
select uid,uname
where uid < 2
insert into text3
select uname
;

8、CTAS:
create table if not exists text5
as
select uid from text1 where uid > 3
;
```

千锋好程序大数据学院

2.5.45 数据导出

```
hive数据导出:
1、从hive表中导出本地文件系统中(目录、文件)
2、从hive表中导出hdfs文件系统中
3、hive表中导出到其它hive表中

关键字:insert overwrite directory

1、
insert overwrite local directory '/home/hivedata/out/00'
select * from t_access_times;
```

直接导入到本地文件系统的文件中：

```
hive -help
hive -e 'select * from text1' >> /home/hivedata/out/02;

2、
insert overwrite directory '/home/hivedata/out/00'
select * from text1;

修改导出后的列与列之间的格式：
insert overwrite local directory '/home/hivedata/out/01'
row format delimited fields terminated by '\t'
select * from text1;
```

##2.5.46 文件读取/解析的方式指定ROW FORMAT

```
[ROW FORMAT row_format]
```

总述：HIVE用了两个类去读数据

```
一个类用于从文件中读取一条一条的记录（可能是一行，可能是xml文件中的一个完整标签）
一个类用于从上面读到的记录中切分出一个一个的字段（可能简单地按照分隔符切，也可以对复杂结构进行自定义切）
```

ROW FORMAT示例：

```
ROW FORMAT :用什么INPUTFORMAT去读数据
DELIMITED ：用普通的org.apache.hadoop.mapred.TextInputFormat去读数据行，以回车符作为行分割
FIELDS TERMINATED BY ',' ： 表示用什么SerDe类去解析一行中的数据，默认用org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
```

##2.5.47 序列化和反序列化

```
serialize:序列化(写数据)
deserialize:反序列化(读数据)
```

###2.5.47.1 常用Serder

```
csv 、tsv 、json serder 、 regexp serder 等。
csv ： 逗号分隔值
tsv ： tab 分隔值
json ： json格式的数据
regexp ： 数据需要复合正则表达式
```

###2.5.47.2 csv
创建表：

```
create table if not exists csv1(
uid int,
uname string,
age int
)
row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
```

千锋好程序大数据学院

```
stored as textfile
;

load data local inpath '/hivedata/csv1' into table csv1;

create table if not exists csv3(
uid int,
uname string,
age int
)
row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
with serdeproperties(
"separatorChar"=",",
"qutoeChar"="'",
"escapeChar"="\\"
)
stored as textfile
;
```

###2.5.47.3 Json Serde

如果是第三方jar包或者是自己写的，就必须要先加载jar包：

hive>add jar /home/json-serde-1.3-jar-with-dependencies.jar;

```
create table if not exists js1(
pid int,
content string
)
row format serde "org.openx.data.jsonserde.JsonSerDe"
;

{"pid":1,"content":"this is pid of 1 content"}
{"pid":2,"content":"this is pid of 2 content"}

load data local inpath '/home/hivedata/js1' into table js1;
```

复杂数据类型嵌套案例：

```
create table if not exists complex(
uid int,
uname string,
belong array<String>,
tax map<String,array<double>>
)
row format serde "org.openx.data.jsonserde.JsonSerDe"
;

{"uid":1,"uname":"zs","belong":["zs1","zs2","zs3"],"tax":{"shebao":
[220,280,300],"gongjijin":[600,1200,2400]}}
{"uid":2,"uname":"ls","belong":["ls1","ls2","ls3"],"tax":{"shebao":
[260,300,360],"gongjijin":[800,1600,2600]}}

load data local inpath '/home/hivedata/complex' into table complex;

select
c.*
from complex c
```

千锋好程序大数据学院

```
where size(c.belong) = 3
and c.tax["gongjijin"][1] > 1800
;
```

###2.5.47.4 Regex Serde

hive对文件中字段的分隔符默认情况下只支持单字节分隔符，如果数据中的分隔符是多字节的，则hive默认是处理不了的。

```
01||gaoyuanyuan||18
02||gaogao||26
create table if not exists bi(
id int,
name string,
age int
)
row format delimited
fields terminated by '%'
stored as 'customInputFormat'
;

create table if not exists t_bi_reg(
id string,
name string,
age int
)
row format serde 'org.apache.hadoop.hive.serde2.RegexSerDe'
with serdeproperties(
'input.regex'='(.*)\\|\\|(.*)\\|\\|(.*)',
'output.format.string'='%1$s %2$s %3$s'
)
stored as textfile
;

load data local inpath '/hivedata/bi.txt' into table t_bi_reg;
```

缺点：字段多的情况，要指定的内容也多

通过自定义InputFormat来解决特殊字符分割的问题。
原理是在InputFormat读取行时将数据中的"多字符分隔符"替换为hive的默认的单字符分隔符，以便hive在执行serde操作时可以按照默认的分隔符的格式来对字段进行分割。

然后将代码打包，并copy到hive安装目录下的lib文件夹中，并重启hive
还需要在hive中添加jar包，add jar path，才能在执行hql查询该表时将自定义的 jar包传递给maptask
建表：

```
create table if not exists t_bi_reg(
id string,
name string,
age int
)
row format delimited
fields terminated by '|'
stored as inputformat
'com.qfedu.bigdata.hiveUDF.inputformat.BiDelimiterInputFormat'
stored as outputformat
'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
;
```

千锋好程序大数据学院

###2.5.47.5 hive的分隔符

```
hive默认的列与列之间的分隔符是：\001,注意不是tab
通常分隔符：
tab
,
" "
|
\n
\001    ^A (\u0001,注意不是\0001也不是\01)
\002    ^B
\003    ^C
```

##2.5.48 hive文件存储

```
hive默认的数据文件存储格式为：textfile
textfile：普通的文本文件存储，不压缩。
sequencefile:hive为用户提供的二进制存储，本身就压缩。不能用load方式加载数据
rcfile:hive提供行列混合存储，hive在该格式下，将会尽量把附近的行和列的块尽量存储到一起。仍然压缩，查询效率较高。
orc
```

###2.5.48.1 属性设置

hive.default.fileformat

TextFile

Expects one of [textfile, sequencefile, rcfile, orc].

Default file format for CREATE TABLE statement. Users can explicitly override it by CREATE TABLE ... STORED AS [FORMAT]

```
create table if not exists text1(
uid int,
uname string
)
row format delimited fields terminated by ' '
;

load data local inpath '/hivedata/seq1' into table seq1;
```

###2.5.48.2 创建sequencefile

```
create table if not exists seq1(
uid int,
uname string
)
row format delimited fields terminated by ','
stored as sequencefile
;

该方式不行：
load data local inpath '/hivedata/seq1' into table seq1;

使用以下方式：
```

千锋好程序大数据学院

```
insert into table seq1
select uid,uname from text1;
```

###2.5.48.3 创建rcfile

```
create table if not exists rc1(
uid int,
uname string
)
row format delimited fields terminated by ' '
stored as rcfile
;

该方式不行：
load data local inpath '/hivedata/seq1' into table rc1;

使用inset into方式：
insert into table rc1
select uid,uname from text1;

create table seq2(
movie string,
rate string,
times string,
uid string
)
row format delimited fields terminated by ','
stored as sequencefile
;

create table if not exists rc1(
movie string,
rate string,
times string,
uid string
)
row format delimited fields terminated by ','
stored as rcfile
;

from t_movieRate
insert into table seq2
select *
insert into table rc1
select *
;

综合效率：是defaultCodec+rcfile较好
```

###2.5.48.4 自定义存储格式

```
数据：
seq_yd元数据文件：
aGVsbG8gemhhbmdoYW8=
aGVsbG8gZmVpZmVpLGdvb2QgZ29vZCBzdHVkeSxkYXkgZGF5IHVw
seq_yd文件为base64编码后的内容，decode后数据为：
```

千锋好程序大数据学院

```
hello zhanghao
hello feifei,good good study,day day up

create table cus(str STRING)
stored as
inputformat
'org.apache.hadoop.hive.contrib.fileformat.base64.Base64TextInputFormat'
outputformat
'org.apache.hadoop.hive.contrib.fileformat.base64.Base64TextOutputFormat';

LOAD DATA LOCAL INPATH '/home/hivedata/cus' INTO TABLE cus;
```

##2.5.49 Hive视图

```
hive的视图简单理解为逻辑上的表
hive现目前只支持逻辑视图，不支持物化视图。

hive的视图意义：

1、对数据进行局部暴露(涉及隐私数据不暴露)。
2、简化复杂查询。

创建视图：

create view if not exists tab_v1
as
select uid from tab1 where uid < 10;

查看视图：

show tables;
show create table tab_v1;
desc tab_v1;

视图是否可以克隆：(hive-1.2.1暂时不支持)

create view tab_v2 like tab_v1;

？？view 暂没有结构的修改？？？？？(直接修改元数据可行)

ALTER VIEW view_name SET TBLPROPERTIES table_properties
table_properties:
  : ("TBL_NAME" = "tbv1")
;

create view if not exists v1 as select * from text1;

ALTER VIEW v1 SET TBLPROPERTIES("TBL_NAME" = "v11");  ???任然有问题
create view if not exists v1 as select * from text1;

删除视图：
drop view if exists tab_v2;

注意：
1、切忌先删除视图对应的表后再查询视图。
2、视图是不能用insert into 或者load 方式来加载数据。
```

千锋好程序大数据学院

```
3、视图是只读，不能修改其结构、表相关属性。
```

##2.5.50 Hive的日志

```
hive的系统日志：
默认目录：/tmp/{user.name}
hive.log.dir={java.io.tmpdir}/{user.name}
hive.log.file=hive.log
hive的查询日志：
<name>hive.querylog.location</name>
<value>{system:java.io.tmpdir}/${system:user.name}</value>
<description>Location of Hive run time structured log file</description>
```

##2.5.51 Hive的压缩

```
hive的map阶段压缩：
set hive.exec.compress.output=false;
set hive.exec.compress.intermediate=false;
hive.intermediate.compression.codec
hive.intermediate.compression.type

hive的reduce阶段压缩：
set hive.exec.compress.output=false;
set hive.exec.compress.intermediate=false;
hive.intermediate.compression.codec
hive.intermediate.compression.type
```

##2.5.52 Hive的运行方式

###2.5.52.1 Hive的属性设置

```
1、hive-site.xml
2、hive通过命令行参数设置
3、hive通过cli端set设置
```

###2.5.52.2 三种设置方式的区别

```
1、属性优先级别从上往下一次升高。
2、hive-site.xml是全局和永久的，其它两是临时和局部。
3、hive-site.xml适合所有属性配置，而后两个对于系统级别的属性不能配置。
比如启动所需的元数据库url、log配置等。
```

###2.5.52.3 Hive有四类属性

```
hiveconf：可读可写
hivevar：自定义临时变量，可读可写
system：可读可写
env：可读不可写

--hiveconf <property=value>   Use value for given property
--hivevar <key=value>         Variable subsitution to apply to hive
                                  commands. e.g. --hivevar A=B
hive -e
hive -f
hive -S     静音模式
```

千锋好程序大数据学院

```
hive -i
hive --database

hive>set -v;  ##查看hive相关的环境变量
hive -e "set" | grep current   ##单个查找
```

2.5.52.4 Hive三种运行方式

```
在hive的cli端运行：(开发测试，以及临时跑作业)
通过命令行 hive -e 'sql query';
通过命令行 hive -f /hql文件   (生产线)

hive --database qf1701 -e 'select * from text1';
hive --database qf1701 --hivevar ls=2 --hiveconf tn=text1 -e 'select * from
${hiveconf:tn} limit ${hivevar:ls}';

hive -S --hivevar mapoutputdir=/home/hivedata/out/05 --hivevar
textoutdir=/home/hivedata/out/06 --hivevar limit=1 -f ./hql

注意：
1、一个--hivevar 或者 --hiveconf 只能带一个参数
2、--hiveconf 或者 --hivevar 可以混合使用
3、--hiveconf 或 --hivevar 定义参数不能取消
```

##2.5.53 Hive的远程模式使用

```
和1差不多，只是将元数据放在别的服务器上，这种的就是咱们常说的集群模式。
可以有一个hive的server和多个hive的client。

hive也可以启动为一个服务器，来对外提供

启动方式，（假如是在hadoop01上）：
启动为前台：bin/hiveserver2
启动为后台：nohup bin/hiveserver2 1>/var/log/hiveserver.log
2>/var/log/hiveserver.err &

启动成功后，可以在别的节点上用beeline去连接
方式（1）
hive/bin/beeline  回车，进入beeline的命令界面
输入命令连接hiveserver2
beeline> !connect jdbc:hive2://hdp01:10000
（hdp01是hiveserver2所启动的那台主机名，端口默认是10000）
方式（2）
或者启动就连接：
bin/beeline -u jdbc:hive2://hdp01:10000 -n hadoop

接下来就可以做正常sql查询了
```

2.5.54 Hive企业级调优

```
1、环境方面：服务器的配置、容器的配置、环境搭建
2、具体软件配置参数：
3、代码级别的优化：
```

千锋好程序大数据学院

1、explain 和 explain extended ：

```
explain select * from text1;
explain extended select * from text1;
explain extended
select
d.deptno as deptno,
d.dname as dname
from dept d
union all
select
d.dname as dname,
d.deptno as deptno
from dept d
;
```

explain ： 只有对hql语句的解释。
explain extended：对hql语句的解释，以及抽象表达式树的生成。

stage 相当于一个job，一个stage可以是limit、也可以是一个子查询、也可以是group by等。
hive默认一次只执行一个stage，但是如果stage之间没有相互依赖，将可以并行执行。
任务越复杂，hql代码越复杂，stage越多，运行的时间一般越长。

2、join
hive的查询永远是小表(结果集)驱动大表(结果集)
hive中的on的条件只能是等值连接
注意hive是否配置普通join转换成map端join、以及mapjoin小表文件大小的阀值

3、limit的优化：

```
hive.limit.row.max.size=100000
hive.limit.optimize.limit.file=10
hive.limit.optimize.enable=false （如果limit较多时建议开启）
hive.limit.optimize.fetch.max=50000
```

4、本地模式：

```
hive.exec.mode.local.auto=false （建议打开）
hive.exec.mode.local.auto.inputbytes.max=134217728
hive.exec.mode.local.auto.input.files.max=4
```

5、并行执行：

```
hive.exec.parallel=false   （建议开启）
hive.exec.parallel.thread.number=8
```

6、严格模式：

```
hive.mapred.mode=nonstrict

Cartesian Product.
No partition being picked up for a query.
Orderby without limit.
Comparing bigints and strings.
Comparing bigints and doubles.

select
e.*
from dept d
join emp e
;

select
```

千锋好程序大数据学院

```
e.*
from emp e
order by e.empno desc
;

create table text9 (
uid bigint,
uid1 double
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
;

LOAD DATA LOCAL INPATH '/home/hivedata/text8' INTO TABLE text9;

select

- from text9 t
where t.uid = t.uid1
;
```

7、mapper和reducer的个数：
不是mapper和redcuer个数越多越好，也不是越少越好。

将小文件合并处理(将输入类设置为：CombineTextInputFormat)
通过配置将小文件合并：

```
mapred.max.split.size=256000000
mapred.min.split.size.per.node=1
mapred.min.split.size.per.rack=1
hive.input.format=org.apache.hadoop.hive.ql.io.CombineHiveInputFormat
```

手动设置：

```
set mapred.map.tasks=2;
```

reducer的个数(自动决定和手动设置)：

```
mapred.reduce.tasks=-1
hive.exec.reducers.max=1009
```

8、配置jvm重用：

```
mapreduce.job.jvm.numtasks=1   ###

mapred.job.reuse.jvm.num.tasks=1
```

9、数据倾斜：
数据倾斜：由于key分布不均匀造成的数据向一个方向偏离的现象。
本身数据就倾斜
join语句容易造成
count(distinct col) 很容易造成倾斜
group by 也可能会造成

找到造成数据倾斜的key，然后再通过hql语句避免。

```
hive.map.aggr=true
hive.groupby.skewindata=false  (建议开启)
hive.optimize.skewjoin=false
  Whether to enable skew join optimization.
  The algorithm is as follows: At runtime, detect the keys with a large skew.
Instead of
  processing those keys, store them temporarily in an HDFS directory. In a
follow-up map-reduce
```

千锋好程序大数据学院

```
  job, process those skewed keys. The same key need not be skewed for all the
tables, and so,
  the follow-up map-reduce job (for the skewed keys) would be much faster, since
it would be a
  map-join.
```

10、索引是一种hive的优化：

11、分区本身就是hive的一种优化：

12、job的数量：
一般是一个查询产生一个job，然后通常情况一个job、可以是一个子查询、一个join、一个group by 、一个limit等一些操作。

2.5.55、Hive和Mysql的比较

mysql用自己的存储存储引擎，hive使用的hdfs来存储。
mysql使用自己的执行引擎，而hive使用的是mapreduce来执行。
mysql使用环境环境几乎没有限制，hive是基于hadoop的。
mysql的低延迟，hive是高延迟。
mysql的handle的数据量较小，而hive的能handle数据量较大。
mysql的可扩展性较低，而hive的扩展性较高。
mysql的数据存储格式要求严格，而hive对数据格式不做严格要求。
mysql可以允许局部数据插入、更新、删除等，而hive不支持局部数据的操作。

千锋好程序大数据学院