HIVE:first day

课前说明

今日任务

1. hive的简介
2. hive的特点
3. hive的架构
4. hive与Hadoop的关系
5. hive的安装

今日目标

1. 了解部分

- hive的简介
- hive的特点
- hive与hadoop的关系
- hive的架构
- 数据仓库概念

2. 掌握部分

- hive的安装
- 配置hive的远程模式

正课内容

第一节：hive的简介

1. hive出现的原因

FaceBook网站每天产生海量的数据。为了对这些数据进行管理，并且因为机器学习的需求，产生了hive这门技术，并继续发展成为一个成功的Apache项目。

2. 什么是hive

hive是一个构建在Hadoop上的数据仓库框架（工具），可以将结构化的数据映射成一张数据库表，并可以使用类sql的方式来对大数据集进行读，写以及管理（元数据），这套HIVE SQL 简称HQL。hive的执行引擎可以是MR、spark、tez。如果执行引擎是MR的话，hive会将sql翻译成MR进行数据的计算。 用户可以使用命令行工具和JDBC驱动程序来连接到hive

3. 为什么使用hive

- 直接使用MapReduce所面临的问题是：
  - 人员学习成本高
  - 项目周期要求太短
  - MapReduce实现复杂查询逻辑开发难度大
- 使用hive的优势：
  - 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（适合java语言不好的，sql熟练的人），减少开发人员的学习成本。
  - 功能扩展很方便
  - 适合进行离线分析处理

4. hive的优缺点

1. hive的优点

- 学习成本低:

  提供了类SQL查询语言HQL

- 可扩展

  为超大数据集设计了计算/扩展能力（MR作为计算引擎，HDFS作为存储系统），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。

- 提供了统一的元数据管理

2. hive的缺点

- hive的HQL表达能力有限
- 迭代式算法无法表达，比如pagerank

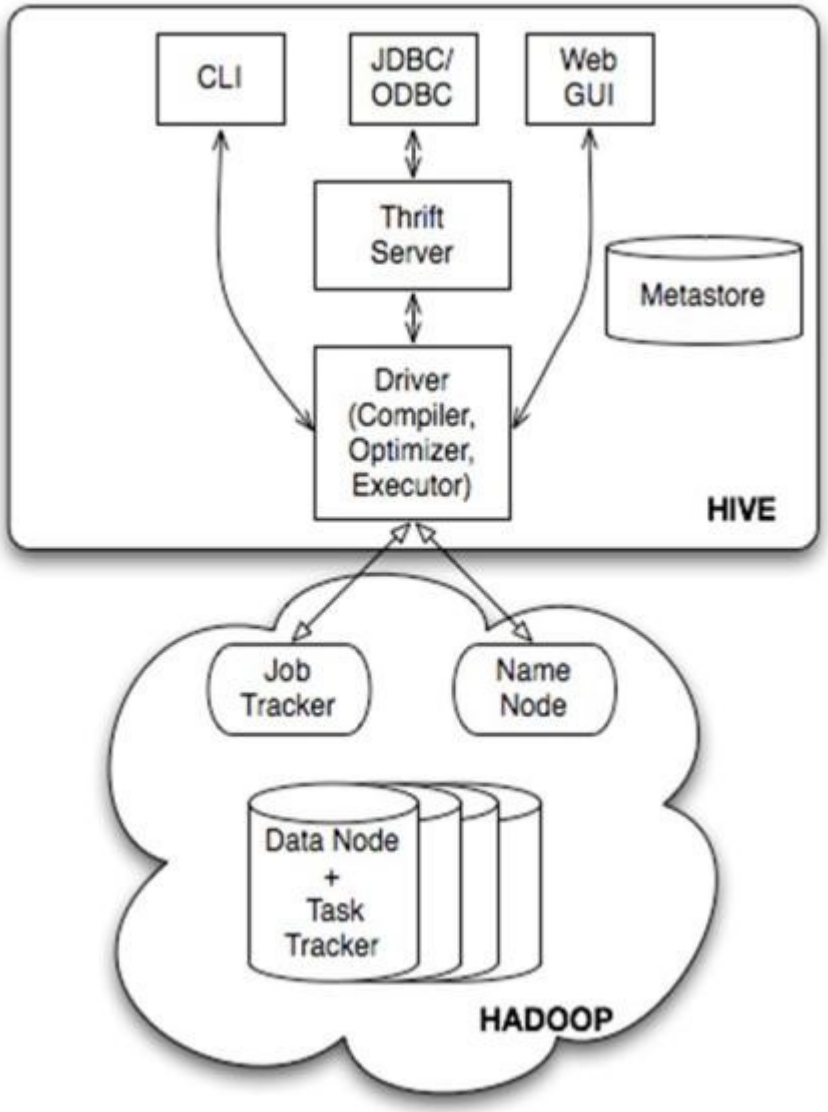
```
    - 数据挖掘方面，比如kmeans
```

  - hive的效率比较低
    - hive自动生成的mapreduce作业，通常情况下不够智能化

```
- hive调优比较困难，粒度较粗
```

第二节：hive的架构

1. hive的架构简介



从上图可以看出，Hive的体系结构分为以下几部分:

1. 用户连接工具

主要有三个：CLI，Client 和 WUI。其中最常用的是CLI，Cli启动的时候，会同时启动一个Hive副本。Client是Hive的客户端，用户连接至Hive Server。在启动 Client模式的时候，需要指出Hive Server所在节点，并且在该节点启动Hive Server。 WUI是通过浏览器访问Hive。

- thriftserver：第三方服务
- 元数据

  Hive将元数据存储在数据库中，如mysql、derby。Hive中的元数据库名、表名、字段名、字段类型、分区、分桶、创建时间、创建人等）。
- 解释器：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。
- hadoop

Hive的数据存储在HDFS中，大部分的查询、计算由MapReduce完成（包含*的查询，比如select * from tbl不会生成MapRedcue任务）。

2. hive和hadoop的关系

- 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。

3. hive与数据库的区别

1. Hive采用了SQL的查询语言HQL，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。
2. 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。
3. Hive 不适合用于联机(online) 事务处理，也不提供实时查询功能。它最适合应用在基于大量不可变数据的批处理作业。Hive 的特点是可 伸缩（在Hadoop 的集群上动态的添加设备），可扩展、容错、输入格式的松散耦合。Hive 的入口是DRIVER ，执行的 SQL 语句首先提交到 DRIVER 驱动，然后调用 COMPILER 解释驱动，最终解释成 MapReduce 任务执行，最后将结果返回。
4. MapReduce 开发人员可以把自己写的 Mapper 和 Reducer 作为插件支持 Hive 做更复杂的数据分析。 它与关系型数据库的 SQL 略有不同，但支持了绝大多数的语句（如 DDL、DML）以及常见的聚合函数、连接查询、条件查询等操作。
5. Hive和数据库的比较如下表：

比较项	SQL	HiveQL
ANSI SQL	支持	不完全支持
更新	UPDATE\INSERT\DELETE	insert OVERWRITE\INTO TABLE
事务	支持	不支持
模式	写模式	读模式
数据保存	块设备、本地文件系统	HDFS
延时	低	高
多表插入	不支持	支持
子查询	完全支持	只能用在From子句中
视图	Updatable	Read-only
可扩展性	低	高
数据规模	小	大
....	.....	.....

第三节：hive的安装部署

准备环境：jdk-1.8 hadoop-2.7.6 hive-1.2.1 mysql-5.7.21

1. 内嵌模式

1. 简介：
   使用hive自带数据库derby来进行存储元数据，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。
2. 安装步骤：
3. 将hive安装包上传到/opt/software/,进行解压

```
$ tar -zxvf apache-hive-1.2.1-bin.tar.gz -C /opt/apps/
```

2. 创建一个软连接

```
$ ln -s apache-hive-1.2.1-bin/ hive
```

3. 配置环境变量

```
$ vi ~/.bash_profile
  #hive environment
HIVE_HOME=/opt/apps/hive
PATH=$HIVE_HOME/bin:$PATH
 export HIVE_HOME PATH
```

4. 使.bash_profile生效

```
]$ source ~/.bash_profile
```

```
注意：在使用hive命令的位置会生成一个derby文件和元数据的目录。

    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

    /opt/apps/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. 本地模式

1. 说明 使用mysql替换derby进行元数据的存储。 hive的相关进程都是在同一台机器上，即本地模式。mysql因为是独立的进程，所有mysql可以在同一机器，也可以在其他机器上
2. 搭建步骤：
3. 环境搭建在内嵌模式的基础上
4. 还要要有mysql环境：【参考linux第三天的笔记】

注意：

- 先查询是否有mysql相关包，如果有先卸载
- 可能会依赖于perl,libaio,numactl

  yum -y install perl
  yum -y install libaio
  yum -y install numactl
- 安装成功后，要先启动服务 sudo service mysqld start
- 查看密码：cat /var/log/mysqld.log
- 登陆：mysql -u root -p
- 修改密码：12345
- 创建数据库hive

  create database hive character set latin1

5. 将mysql的驱动包mysql-connecter-5.1.7.jar，放入hive的lib中

6. conf目录下、配置hive-env.sh

```
$ cp hive-env.sh.template hive-env.sh

$ vi hive-env.sh

    HADOOP_HOME=/opt/apps/hadoop
    JAVA_HOME=/opt/apps/jdk
```

5. conf目录下，配置hive-site.xml

```
$ cp hive-default.xml.template hive-site.xml

$ vi hie-site.xml，找到相应属性修改值。或者删除所有默认配置，添加下面的属性

    <!--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>123456</value>
    <description>password to use against metastore database</description>
    </property>
```

6. 使用hive命令进行连接即可，完美收官。

```
]$ hive

前提：集群和mysql都要先启动
```

3. 远程模式

1. 说明

将hive作为一个server进程开启,客户端可以在任何机器上，只要连接到这个server，就可以进行操作。客户端可以不需要密码。

2. 服务端的配置

```
<property>
      <name>hive.metastore.warehouse.dir</name>
      <value>/user/hive/warehouse</value>
      <description>location of default database for the warehouse</description>
    </property>
    <property>
      <name>javax.jdo.option.ConnectionURL</name>
      <value>jdbc:mysql://slave3:3306/hive?
createDatabaseIfNotExist=true&amp;characterEncoding=UTF-8</value>
    </property>
    <property>
      <name>javax.jdo.option.ConnectionDriverName</name>
      <value>com.mysql.jdbc.Driver</value>
    </property>
    <property>
      <name>javax.jdo.option.ConnectionUserName</name>
      <value>root</value>
    </property>
    <property>
      <name>javax.jdo.option.ConnectionPassword</name>
      <value>123456</value>
    </property>
```

客户端安装与配置：

1. 方式1：可以单独上传，解压，配置环境配置
2. 方法2：scp已经存在的节点上的hive环境和环境变量
3. 配置第三方服务

```
    <property>
          <name>hive.metastore.uris</name>
          <value>thrift://slave3:9083</value>
        </property>
```

启动服务端的服务

```
1. 方法1: 直接调用hiveserver2。会进入监听状态不退出。
2. 方法2: hive --service hiveserver2 &    进入后台启动
```

远程连接:

```
beeline 是hive的一个轻量级的连接客户端

1. 方式1:
    1. beeline  回车
    2. !connect jdbc:hive2://slave3:10000 回车
    3. 输入用户名  回车
    4. 输入密码       回车
2. 方法2(直连):
      1. beeline -u jdbc:hive2://slave3:10000 -n root

            解析: hive2, 是hive的协议
                    10000, 是hiveserver2的端口号
```

HIVE:first day

课前说明

今日任务

1. 分区表的相关内容
2. 分区表的案例演示
3. 分区表的相关操作
4. 分区表的分类及其创建
5. 分桶的概念
6. 分桶的创建和加载数据
7. 分桶的查询操作

今日目标

1. 了解部分
    - 为什么要分区
    - 如何分区
    - 分区的意义
    - 分区表的分类

- 分桶的概述
- 分区和分桶的相关参数

2. **掌握**部分
   - 如何创建分区
   - 如何创建动态分区
   - 如何创建混合分区
   - 如何创建分桶表和加载数据
   - 分桶的查询操作

正课内容

第一节：分区表的相关内容

1. 为什么分区

   Hive的Select查询时，一般会扫描整个表内容。随着系统运行的时间越来越长，表的数据量越来越大，而hive查询做全表扫描，会消耗很多时间，降低效率。而有时候，我们需求的数据只需要扫描表中的一部分数据即可。这样，hive在建表时引入了partition概念。即在建表时，将整个表存储在不同的子目录中，每一个子目录对应一个分区。在查询时，我们就可以指定分区查询，避免了hive做全表扫描，从而提高查询效率。

2. 如何分区

   根据业务需求而定，不过通常以年、月、日、小时、地区等进行分区。

3. 分区的技术

   partitioned by (colName colType [comment '...'],...)

   - hive的分区名的字段名不区分大小写，值区分大小写，不支持中文
   - hive的分区字段是一个伪字段，但是可以用来进行操作
   - 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。
   - 分区是以字段的形式在表结构中存在，通过describe table命令可以查看到字段存在，但是该字段不存放实际的数据内容，仅仅是分区的表示。

4. 分区的意义

   让用户在做数据统计的时候缩小数据扫描的范围，在进行select操作时可以指定要统计哪个分区

5. 分区的本质

   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

第二节：分区案例

1. 一级分区

1. 建表语句

   create table if not exists part1(id int, name string, age int) partitioned by (dt string) row format delimited fields terminated by '\t' lines terminated by '\n';

2. 加载数据

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

2. 二级分区

1. 建表语句

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

2. 加载数据 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);

3. 三级分区

1. 建表语句

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

2. 加载数据

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

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

4. 测试是否区分大小写

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

2. 加载数据

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

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

5. 查看分区:

1. 查看

   show partitions part4;

6. 修改分区:

1. 修改分区 （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/mydb1.db/part1/dt=2018-03-21';    --错误使用
alter table part5 partition(year='2018',month='03',day='21')
set location
'hdfs://master:8020/user/hive/warehouse/qf1903.db/part5/year=2018/month=03/day=AA/'
;

注意：修改后，原有的分区子目录存储，数据也存在，但是分区内的文件不再使用。
```

7. 增加分区并设置数据

1. 新增分区（空）

   alter table part5 add partition(dt='2018-03-27'); alter table part5 add partition(dt='2018-03-20') partition(dt='2018-03-17');

2. 新增分区 (带数据)

   alter table part5 add partition(dt='2018-03-27') location '/user/hive/warehouse/mydb1.db/part1/dt=2018-03-20';

3. 新增多分区

   alter table part5 add partition(dt='2018-03-26') location '/user/hive/warehouse/mydb1.db/part1/dt=2018-03-20' partition(dt='2018-03-24') location '/user/hive/warehouse/qf1704.db/part1/dt=2018-03-21';

8. 删除分区

1. 删除单个分区

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

2. 删除多个分区

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

```
注意：删除分区实质上就是删除目录及其里面的内容
```

第三节：分区类型详解

1. 分区的种类

1. 静态分区：直接加载数据文件到指定的分区，即静态分区表。
2. 动态分区：数据未知，根据分区的值来确定需要创建的分区(分区目录不是指定的，而是根据数据的值自动分配的)
3. 混合分区：静态和动态都有。

2. 分区属性设置

```
    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
严格模式下会阻止以下三种查询：
- 对分区表查询，where条件中过滤字段不是分区字段
- 笛卡尔积的join不使用on条件或者where条件
- 对order by 查询不带limit语句
```

3. 创建动态分区的案例

1. 创建动态分区表 create table dy_part1(sid int, name string, gender string, age int, academy string) partitioned by (dt string) row format delimited fields terminated by ',' ;
2. 动态分区加载数据
    1. 错误方式：

       load data local inpath '/hivedata/user.txt' into table dy_part1 partition(dt);#####错误方式
    2. 正确方式：要从别的表中加载数据
        1. 先创建临时表：

           create table temp_part1(sid int, name string, gender string, age int, academy string, dt string) row format delimited fields terminated by ',' ;

           ```
           创建临时表时，必须要有动态分区表中的分区字段。
           ```
        2. 导入数据到临时表：

           load data local inpath './data/students2.txt' into table temp_part1;
        3. 动态加载到表

           insert into dy_part1 partition(dt) select sid,name,gender,age,academy,dt from temp_part1;
           ###2.5.14 混合分区示例
3. 创建一个分区表：

   create table dy_part2(id int, name string) partitioned by (year string,month string,day string) row format delimited fields terminated by ',' ;
2. 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 ',' ;

   ```
   数据如下：
   1,廉德枫,2019,06,25
   2,刘浩（小）,2019,06,25
   3,王鑫,2019,06,25
   4,司翔,2019,06,26
   ```
3. 导入数据到分区表
    - 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;

4. 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修改元数据和hdfs数据内容。
5. Hive分区和Mysql分区的区别 mysql分区字段用的是表内字段；而hive分区字段采用表外字段。

第四节：分桶的概念

1. 分桶的概述

1. 为什么要分桶
   - 数据分区可能导致有些分区数据过多，有些分区数据极少。分桶是将数据集分解为若干部分(数据文件)的另一种技术。
   - 分区和分桶其实都是对数据更细粒度的管理。当单个分区或者表中的数据越来越大，分区不能细粒度的划分数据时，我们就采用分桶技术将数据更细粒度的划分和管理
   - [CLUSTERED BY (col_name, col_name, ...)
2. 分桶的原理
   - 与MapReduce中的HashPartitioner的原理一模一样
     - MapReduce：使用key的hash值对reduce的数量进行取模(取余)
     - hive：使用分桶字段的hash值对分桶的数量进行取模(取余)。针对某一列进行分桶存储。每一条记录都是通过分桶字段的值的hash对分桶个数取余，然后确定放入哪个桶。
3. 分桶的意义
   1. 为了保存分桶查询的分桶结构（数据已经按照分桶字段进行了hash散列）
   2. 分桶表适合进行数据抽样

      抽样更高效。处理大数据时，如果能在数据集的一部分上运行查询进行测试会带来很多方便
   3. join操作时可以提高MR的查询效率

      连接查询两个在相同列上划分了桶的表，可以在map端进行高效的连接操作。 比如jion操作。对于两个表都有一个相同的列，如果对两个表都进行桶操作，那么hive底层会对相同列值的桶进行join操作。效率很高

2. 分桶的操作

1. 创建分桶表和加载数据
2. 错误的方式：

- 建表语句： 语句正确

create table student(id int, name string, sex string, age int, academy string) clustered by (sno) into 4 buckets #即指定了分桶字段也指定了排序字段 row format delimited fields terminated by ',' ;

- 加载数据：方式错误，load实际上也是copy，没有分桶效果。

load data local inpath './data/students.txt' into table student;

2. 正确的方式：

    1. 建表语句：语句正确

```
        create table student(
        sno int,
        name string,
        sex string,
        age int,
        academy string
        )
        clustered by (sno) sorted by (age desc) into 4 buckets #分桶字段和排序字段可以不一致
        row format delimited
        fields terminated by ','
        ;
1. 加载数据：分两步
  - 第一步：先创建临时表

     create table temp_student(
        sno int,
        name string,
        sex string,
        age int,
        academy string
        )
        clustered by (sno) sorted by (age desc) into 4 buckets
        row format delimited
        fields terminated by ','
        ;
```

```
       load data local inpath './data/students.txt' into table temp_student;

  - 从临时表中查询并导入数据

       insert into|overwirte table student
       select * from temp_student
       distribute by(sno)
       sort by (age desc)
       ;
```

3. 小贴士：

- 需要设置reduce数量和分桶数量相同：

set mapreduce.job.reduces=4;

- 如果数据量比较大，我们可以使用MR的本地模式：

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

- 强行分桶设置： （常规配置）

set hive.enforce.bucketing=true; 默认是false

测试 insert overwrite table student select * from temp_student distribute by(sno) sort by (sage desc) ;

- 强行排序：（常规配置）

set hive.enforce.sorting=true; 测试： insert overwrite table student select * from temp_student distribute by(sno) sort by (sage desc) ;

2. 分桶的查询
    1. 语法：

       语法:tablesample(bucket x out of y on sno) x:代表从第几桶开始查询，x不能大于y y:代表查询的总桶数,y可以是总桶数的因子或者倍数

    2. 查询全部

       select * from student; select * from student tablesample(bucket 1 out of 1);

    3. 指定桶查询

    查询第一桶 select * from student tablesample(bucket 1 out of 4 on sno); 查询第一桶和第三桶 select * from student tablesample(bucket 1 out of 2 on sno); 查询第二桶和第四桶的数据 select * from student tablesample(bucket 2 out of 2 on sno); 查询对8取余的第一桶的数据： select * from student tablesample(bucket 1 out of 8 on sno);

    3. 其他查询

       查询三行数据 select * from student limit 3; select * from student tablesample(3 rows); 查询百分比的数据 select * from student tablesample(13 percent); 查询固定大小的数据 select * from student tablesample(68b); 单位（K,KB,MB,GB...） 随机抽三行数据 select * from student order by rand() limit 3;

3. 小总结:

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)
;
```

3. 注意事项

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

HIVE:third day

课前提示

今日任务

1. 查询语句基本语法
2. Join的语法和特点
3. Join案例
4. Hive专有Join的特点
5. Where语句特点
6. Group by 语句特点
7. Having语句特点
8. limit语句特点

今日目标

1. 了解部分
   - 基本语法
   - 各种子句的相关特点
2. **掌握**部分
   - 每个子句的使用

正课内容

第一节：查询语句的基本语法

1. 子句与关键字

```
select ..
from ..
join on ..
where ..
group by ..
having ..
order by ..
sort by ..
limit ..
union | union all ...
```

2. 执行顺序

```
- FROM \<left_table>
- ON \<join_condition>
- \<join_type> JOIN \<right_table>
- WHERE \<where_condition>
- GROUP BY \<group_by_list>
- HAVING \<having_condition>
- SELECT
- DISTINCT \<select_list>
- ORDER BY <order_by_condition>
- LIMIT <limit_number>
```

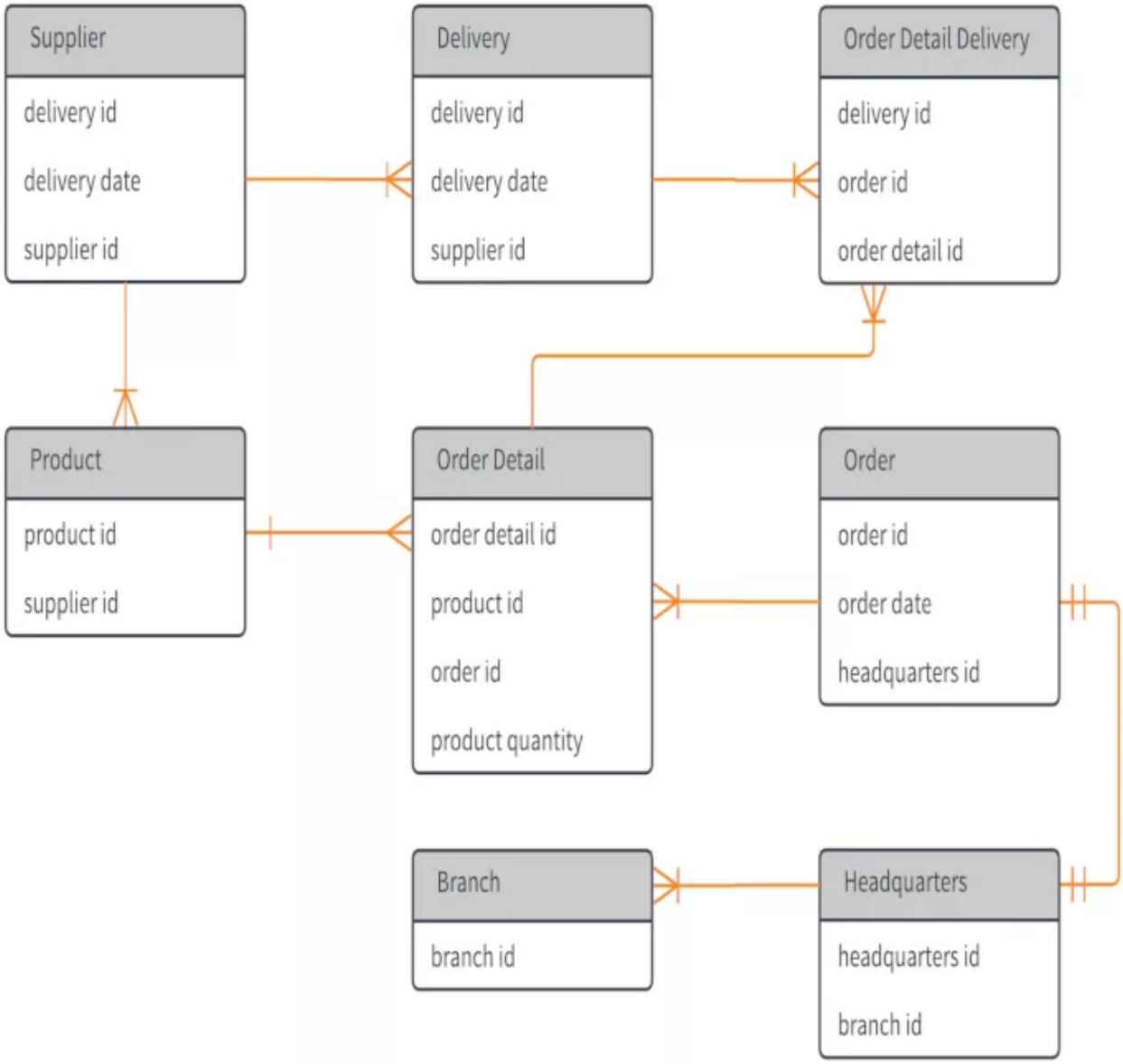
3. 查询原则

1. 尽量不使用子查询、尽量不使用in 或者not in
2. 尽量避免join连接查询，但是通常避免不了
3. 查询永远是小表驱动大表（小表放在前面）

   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。



第二节：语法特点

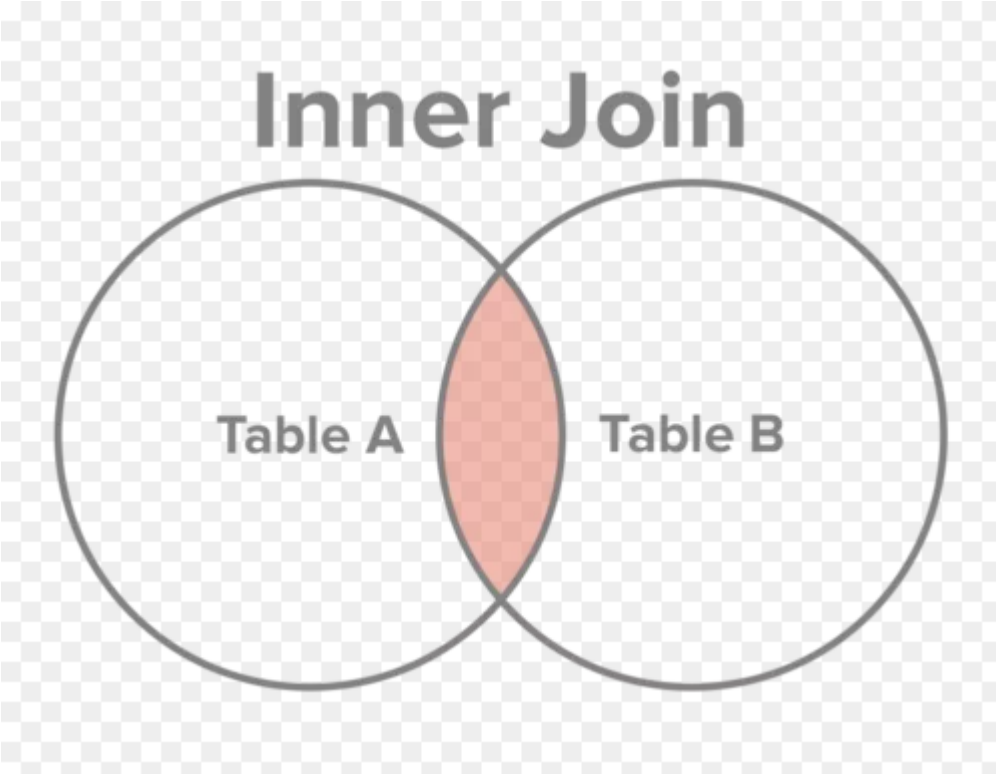
1. join的语法与特点

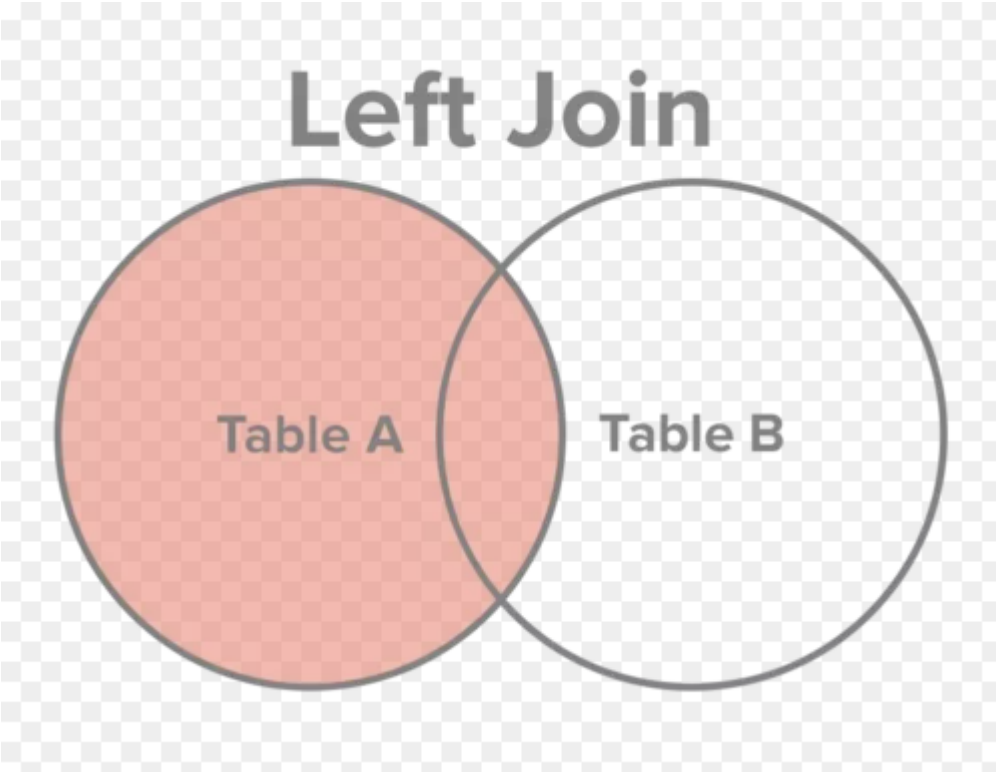
在关系型数据库里面，每个实体有自己的一张表（table），所有属性都是这张表的字段（field），表与表之间根据关联字段"连接"（join）在一起。所以，表的连接是关系型数据库的核心问题。

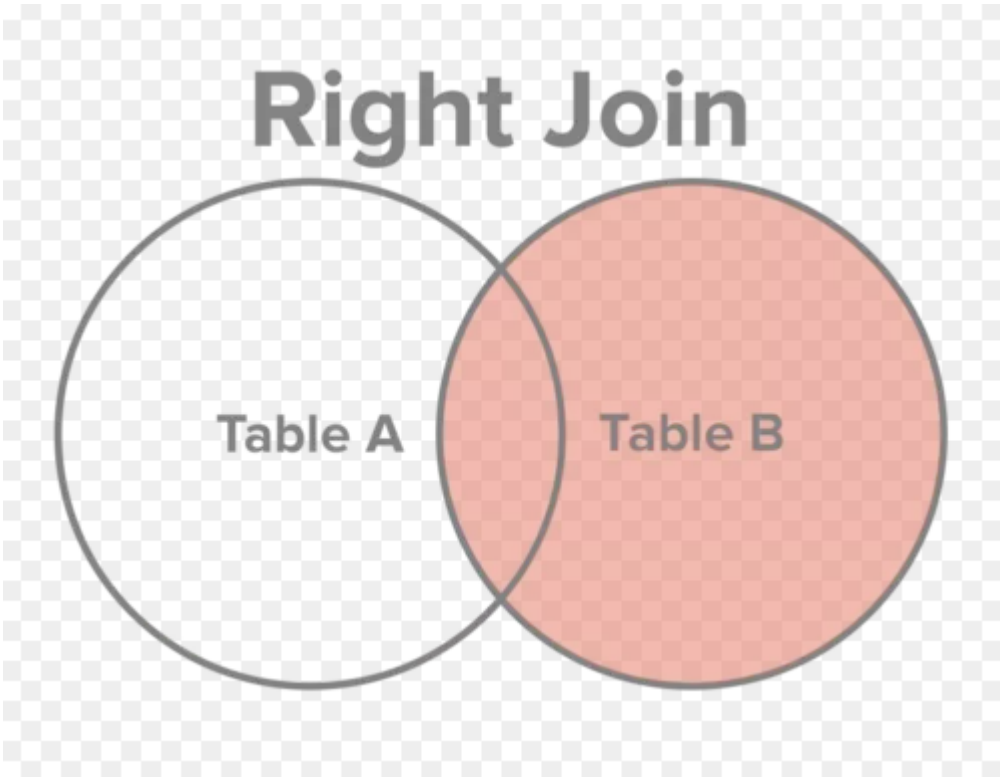
表的连接操作分为两大类：内连接和外连接，而外连接有细分为三种类型。见下图

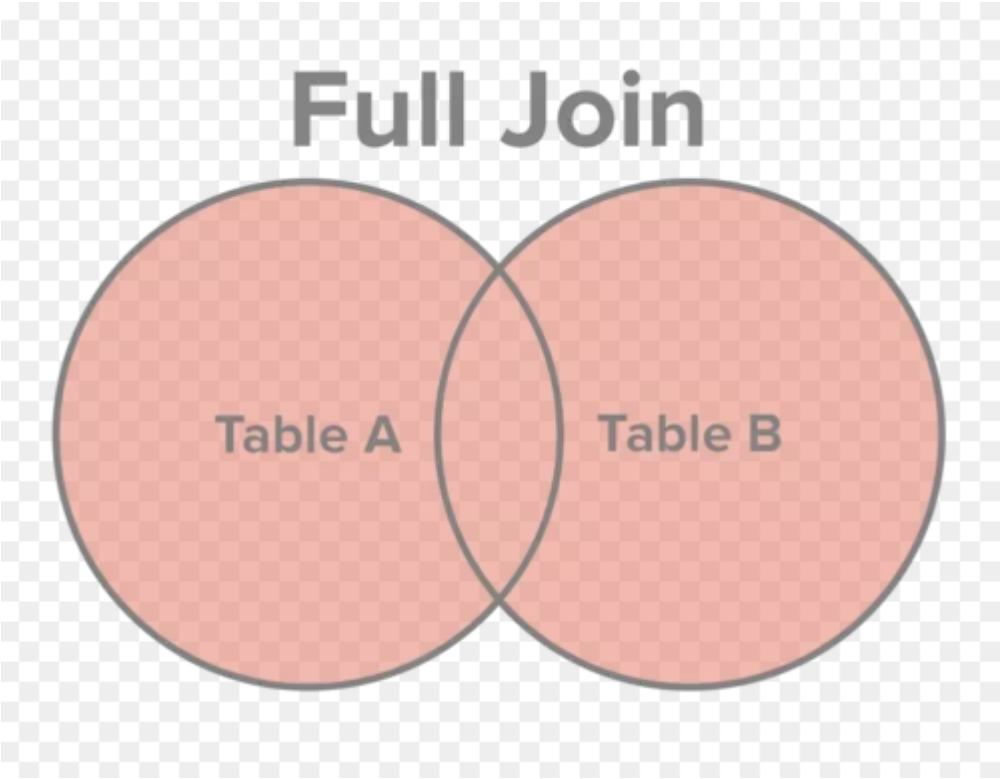
- 内连接: [inner] join
- 外连接 (outer join)：（引出一个驱动表的概念：驱动表里的数据全部显示)
  - 左外连接:left [outer] join, 左表是驱动表
  - 右外连接:right [outer] join, 右表是驱动表
  - 全外连接:full [outer]join,hive支持，mysql不支持.两张表里的数据全部显示出来

注意： join连接只支持等值连接


Inner Join
Table A
Table B

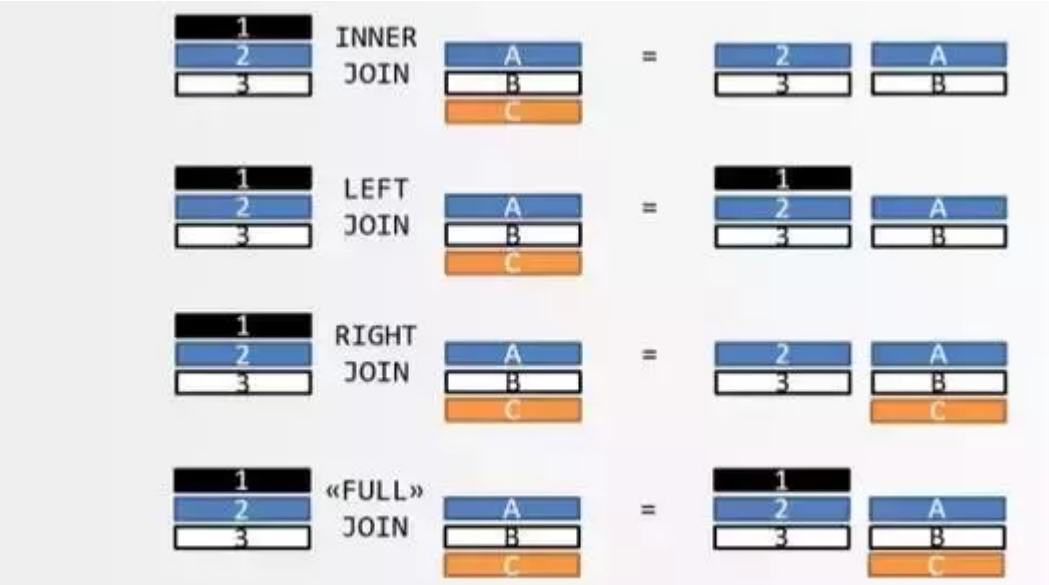

Left Join
Table A
Table B





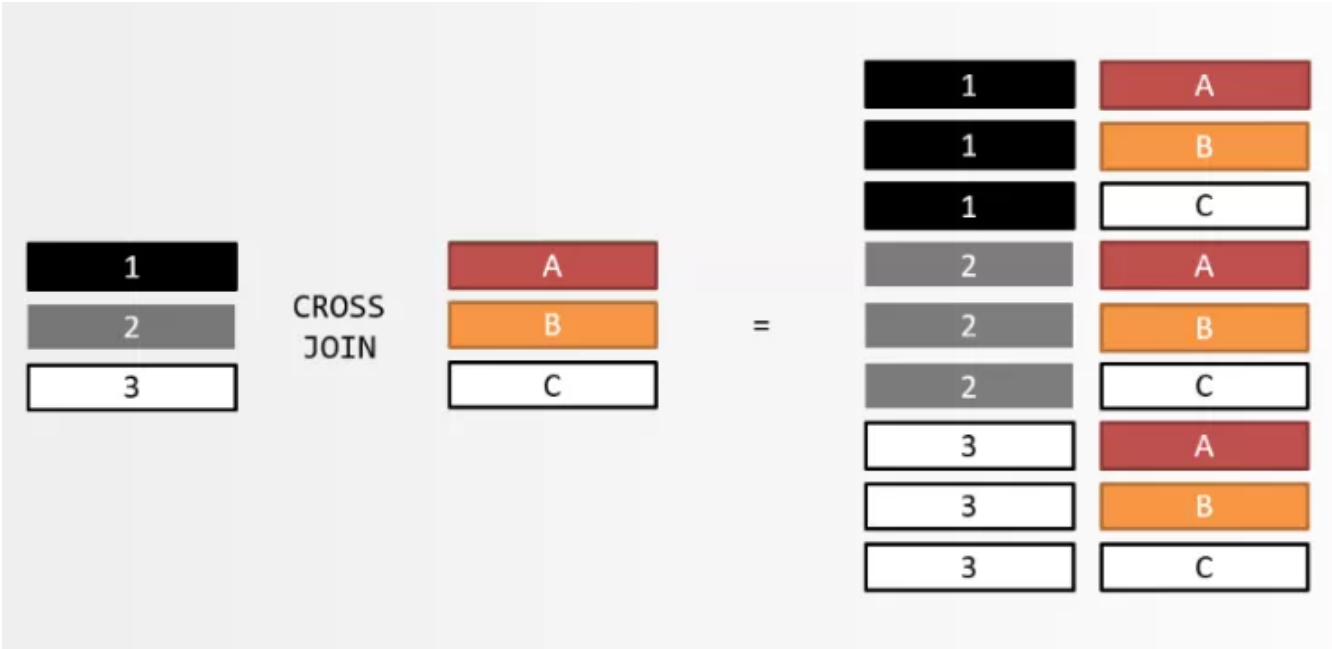
所谓"连接"，就是两张表根据关联字段，组合成一个数据集。问题是，两张表的关联字段的值往往是不一致的，如果关联字段不匹配，怎么处理？比如，表 A 包含张三和李四，表 B 包含李四和王五，匹配的只有李四这一条记录。

很容易看出，一共有四种处理方法。下图就是四种连接的图示。这张图比上面的维恩图更易懂，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。

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(笛卡尔积)的新表（见下图）。




```
准备数据
1,a
2,b
3,c
4,d
7,y
8,u

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

```
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 ','
;
```

2. Hive中join的专有特点

1. left semi join

   在hive中，有一种专有的join操作: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，主要用于解决hive中左表的数据是否存在的问题。

   - exists关键字：满足条件返回true,不满足条件返回false

     练习:查询有领导的员工信息 查询有下属的员工信息 查看所有员工的信息及其部门信息

   - left semi join的写法。

     小贴士： hive中不支持right semi join join\inner join\多表逗号分隔：内连接，都是相互连接成功才能返回结果 多表逗号分隔，使用where的内连接形式效率最低 join不加任何的on或者where条件称为笛卡尔积

2. 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。hive-1.2.1版本已经默认开启map-side join

   set hive.auto.convert.join=true 到底小表多大才会被转换为map-side join： set hive.mapjoin.smalltable.filesize=25000000 约23.8MB

3. where语句特点

```
where后不能使用聚合函数，可以使用子查询，也可以是普通函数。
```

4. 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

count的执行 执行效果上： count(*)包括了所有的列，相当于行数，在统计结果的时候不会忽略null值 count(1)包括了所有列，用1代表代码行，在统计结果的时候也不会忽略null值 count(列名)只包括列名那一列，在统计结果时，会忽略null值

执行效率上 列名为主键，count(列名)会比count(1)快 列名不为主键，count(1)会比count(列名)快 如果表中多个列并且美也逐渐，count（1）的效率高于count(*) 如果有主键count(主键)效率是最高的 如果表中只有一个字段count(*)效率最高

5. having子句特点

对分组以后的结果集进行过滤。

6. 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