项目六 Hive数据查询



《Hive数据仓库应用教程》

任务 6-1 Hive基本查询

6.1.1 任务描述

在当前的任务中，我们将深入探讨Hive，这是一个基于Hadoop的数据仓库工具。我们将详细介绍Hive的架构、基本数据类型及其如何融入整个大数据生态系统。通过本次任务，学生们将更好地理解Hive的功能、优点以及它在大规模数据处理和分析中的角色。







语法树

- 从哪里查询取决于FROM关键字后面的table_reference。可以是普通物理表、视图、join结果或子查询结果。
- 表名和列名不区分大小写。

```
[WITH CommonTableExpression (, CommonTableExpression)*]
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[ORDER BY col_list]
[
    CLUSTER BY col_list
    | [DISTRIBUTE BY col_list] [SORT BY col_list]
]
[LIMIT [offset,] rows];
```

案例： 美国Covid-19新冠数据之select查询

- 准备一下select语法测试环境，在附件资料中有一份数据文件《us-covid19-counties.dat》，里面记录了2021-01-28美国各个县累计新冠确诊病例数和累计死亡病例数。

数据环境准备(1)

```
--step1:创建数据库和普通表t_usa_covid19
hive-bee
create database unit6db;
show databases;
use unit6db;
show tables;

drop table if exists t_usa_covid19;
CREATE TABLE t_usa_covid19(
count_date string,
county string,
state string,
fips int,
cases int,
deaths int)

row format delimited
fields terminated by ',';
show tables;


--将源数据load加载到t_usa_covid19表对应的路径下
load data local inpath '/home/hadoop/us-covid19-counties.dat'
into table t_usa_covid19;
SELECT * FROM t_usa_covid19 LIMIT 10;
```

数据环境准备(2)

```
--step2:创建一张分区表 基于count_date日期,state州进行分区
CREATE TABLE if not exists t_usa_covid19_p(
county string,
fips int,
cases int,
deaths int)

partitioned by(count_date string, state string)
row format delimited fields terminated by ',';
```

```
0: jdbc:hive2://localhost:10000> show tables;
+-----------------+
|     tab_name    |
+-----------------+
| t_usa_covid19   |
| t_usa_covid19_p |
+-----------------+
```

数据环境准备(3)

```
--step3:使用动态分区插入将数据导入t_usa_covid19_p中
set hive.exec.dynamic.partition.mode = nonstrict;

insert into table t_usa_covid19_p
partition (count_date,state)
select county,fips,cases,deaths,count_date,state
from t_usa_covid19;  --执行所需时间(107.849秒)

select * from t_usa_covid19_p limit 10;
```

t_usa_covid19_p.county	t_usa_covid19_p.fips	t_usa_covid19_p.cases	t_usa_covid19_p.deaths	t_usa_covid19_p.count_date	t_usa_covid19_p.state
Autauga	1001	5554	69	2021-01-28	Alabama
Baldwin	1003	17779	225	2021-01-28	Alabama

基础语法

```
[WITH CommonTableExpression (, CommonTableExpression)*]
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[ORDER BY col_list]
[
    CLUSTER BY col_list
    | [DISTRIBUTE BY col_list] [SORT BY col_list]
]
[LIMIT [offset,] rows];
```

6.1.3 Hive SQL select查询基础语法语法

select_expr

- Select_expr 表示检索查询返回的列，必须至少有一个select_expr

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
```

```
--1、select_expr（选择表达式）
--查询所有字段或者指定字段
select * from t_usa_covid19_p;
select county, cases, deaths from t_usa_covid19_p;

--查询匹配正则表达式的所有字段
SET hive.support.quoted.identifiers = none; --带反引号的名称被解释为正则表达式
select `^c.*` from t_usa_covid19_p;

--查询当前数据库
select current_database(); --省去from关键字

--查询使用函数
select count(county) from t_usa_covid19_p;
```

ALL、DISTINCT

- 用于指定查询返回结果中重复的如何处理
  - 如果没有给出这些选项，则默认值为ALL(返回所有匹配的行)
  - DISTINCT指定从结果集中删除重复的行。

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
```

```
--2. ALL 和 DISTINCT
--返回所有匹配的行（包含重复值）
select state
from t_usa_covid19_p;

--相当于使用ALL关键字（默认行为）
select all state
from t_usa_covid19_p;

--返回所有匹配的行，去除重复的结果
select distinct state
from t_usa_covid19_p;      --执行所需时间(66.265秒)

--多个字段使用DISTINCT - 整体组合去重
select distinct county, state from t_usa_covid19_p;
--执行所需时间(48.53秒)
```

WHERE

- WHERE后面是·一个布尔表达式， 用于查询过滤
- 在WHERE表达式中，可以使用Hive支持的任何函数和运算符，但聚合函数除外。那么为社么不能在where子句中使用聚合函数呢？因为聚合函数要使用它的提前结果集已经确定。而where自居还处于“确定”结果集的过程中，因而不能使用聚合函数。
- 从Hive0.13开始，WHERE字句支持某些类型的子查询。

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
```

6.1.3 Hive SQL select查询基础语法语法

WHERE

```
--3、WHERE 子句
select * from t_usa_covid19_p where 1 > 2; -- 1 > 2 返回false
select * from t_usa_covid19_p where 1 = 1; -- 1 = 1 返回true

--where条件中使用函数：找出州名字母长度超过10位的有哪些
SELECT * from t_usa_covid19_p where length(state) >10 ;

--where子句支持子查询
SELECT *
FROM A
WHERE A.a IN (SELECT foo FROM B);

--注意：where条件中不能使用聚合函数
--报错 SemanticException: Not yet supported place for UDAF 'count'
--聚合函数要使用它的前提是结果集已经确定。
--而where子句还处于"确定"结果集的过程中，因而不能使用聚合函数。
select state,count(deaths)
from t_usa_covid19_p where count(deaths) >100 group by state;
HAVING count(deaths)>100;
```

分区查询、分区裁剪

- 针对Hive分区表，在查询时可以指定分区查询，减少全表扫描，也叫做分区裁剪。
- 所谓分区裁剪指：对分区表进行查询时，会检查WHERE子句或JOIN中的ON子句中是否存在对分区字段的过滤，如果存在，则仅访问查询符合条件的分区，即裁剪掉没必要访问的分区。

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
```

```
--4. 分区查询、分区裁剪
--找出来自加州California，累计死亡人数大于1000的县 state字段 就是分区字段
--进行分区裁剪 避免全表扫描
select * from t_usa_covid19_p
where state = "California" and deaths > 1000;

--多分区裁剪(cái jiǎn)
select * from t_usa_covid19_p
where count_date = "2021-01-28"
and state = "California" and deaths > 1000;
```

GROUP BY概念

- GROUP BY语句用于结合聚合函数，根据一个或多个列对结果集进行分组。
- 注意：出现在GROUP BY中select_expr的字段：

✓ 要么是GROUP BY分组的字段；要么是被聚合函数应用的字段。

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
```



GROUP BY语法限制

- 出现在GROUP BY中select_expr的字段：要么是GROUP BY分组的字段；要么是被聚合函数应用的字段。
- 原因：避免出现一个字段多个值的歧义。

1. 分组字段出现在select_expr中一定没有歧义，因为就是基于该字段分组的，同一组中其值必然相同；
2. 被聚合函数应用的字段也没有歧义，因为聚合函数的本质是多进一出，最终返回一个结果。



基于category进行分组，相同颜色的分在同一组中。在select_expr中，如果出现category字段则没有问题，因为同一组中category值一样，但是返回day就有问题了，day的结果不一样。

GROUP BY使用

```
-- 5、GROUP BY
-- 根据state州进行分组
-- SemanticException: Expression not in GROUP BY key 'deaths'
-- deaths不是分组字段 报错
-- state是分组字段 可以直接出现在select expr中

select state, deaths
from t_usa_covid19_p where count_date = "2021-01-28" group by state;

-- 被聚合函数应用(执行需要时间：117.954秒)
select state, count(deaths)
from t_usa_covid19_p where count_date = "2021-01-28" group by state;
```

6.1.3 Hive SQL select查询基础语法语法

HAVING

- 在SQL中增加HAVING子句的原因是，WHERE关键字无法与聚合函数一起使用。
- HAVING子句可以让我们筛选分组后的各组数据，并且可以在HAVING中使用聚合函数，因为此时WHERE、GROUP BY已经执行结束，结果集已经确定。

```sql
-- 6、HAVING
-- 统计死亡病例数大于10000的州
-- where语句中不能使用聚合函数 语法报错
select state, sum(deaths)
from t_usa_covid19_p
where count_date = "2021-01-28" and sum(deaths) > 10000
group by state;

--先where分组前过滤（此处是分区裁剪），
--再进行group by分组，分组后每个分组结果集确定再使用having过滤
--(执行需要时间：53.652秒)
select state, sum(deaths)
from t_usa_covid19_p
where count_date = "2021-01-28"
group by state
having sum(deaths) > 10000;

-- 这样写更好 即在group by的时候聚合函数已经作用得出结果
-- having直接引用结果过滤 不需要再单独计算一次了
--(执行需要时间：57.286秒)
select state, sum(deaths) as cnts
from t_usa_covid19_p
where count_date = "2021-01-28"
group by state
having cnts > 10000;
```

HAVING与WHERE区别

- having是在分组后对数据进行过滤
- where是在分组前对数据进行过滤
- having后面可以使用聚合函数
- where后面不可以使用聚合函数

LIMIT

- LIMIT用于限制SELECT语句返回的行数。
- LIMIT接受一个或两个数字参数，这两个参数都必须是非负整数常量。
- 第一个参数指定要返回的第一行的偏移量（从Hive 2.0.0开始），第二个参数指定要返回的最大行数。当给出单个参数时，它代表最大行数，并且偏移量默认为0。

6.1.3 Hive SQL select查询基础语法语法

LIMIT

```
-- 7、limit
-- 没有限制返回2021.1.28加州的所有记录
select * from t_usa_covid19_p
where count_date = "2021-01-28"
and state = "California";

-- 返回结果集的前5条
select * from t_usa_covid19_p
where count_date = "2021-01-28"
and state = "California"
limit 5;

-- 返回结果集从第1行开始 共3行
select * from t_usa_covid19_p
where count_date = "2021-01-28"
and state = "California"
limit 2,3; -- 注意 第一个参数偏移量是从0开始的
```

执行顺序

- 在查询过程中执行顺序：from > where > group (含聚合) > having > order > select;
  1. 聚合语句（sum, min, max, avg, count）要比having子句优先执行
  2. where子句在查询过程中执行优先级别优先于聚合语句（sum, min, max, avg, count）
- 结合下面SQL感受：

```
select state, sum(deaths) as cnts
from t_usa_covid19_p
where count_date = "2021-01-28"
group by state
having cnts > 10000;
```

6.1.4 Hive SQL-DQL-Select语法树

高阶语法

```
[WITH CommonTableExpression (, CommonTableExpression)*]
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
  FROM table_reference
  [WHERE where_condition]
  [GROUP BY col_list]
  [ORDER BY col_list]
  [CLUSTER BY col_list
    | [DISTRIBUTE BY col_list] [SORT BY col_list]
  ]
[LIMIT [offset,] rows];
```

6.1.5 Hive SQL select查询高阶语法

ORDER BY

```
[ORDER BY col_list]
[CLUSTER BY col_list
  | [DISTRIBUTE BY col_list] [SORT BY col_list]
]
```

- Hive SQL中的ORDER BY语法类似于标准SQL语言中的ORDER BY语法，会对输出的结果进行全局排序。
- 因此当底层使用MapReduce引擎执行的时候，只会有一个reducetask执行。如果输出的行数太大，会导致需要很长的时间才能完成全局排序。
- 默认排序为升序（ASC），也可以指定为DESC降序。
- 在Hive 2.1.0和更高版本中，支持在ORDER BY子句中为每个列指定null类型结果排序顺序。
- ASC顺序的默认空排序顺序为NULLS FIRST，而DESC顺序的默认空排序顺序为NULLS LAST。

6.1.5 Hive SQL select查询高阶语法

ORDER BY

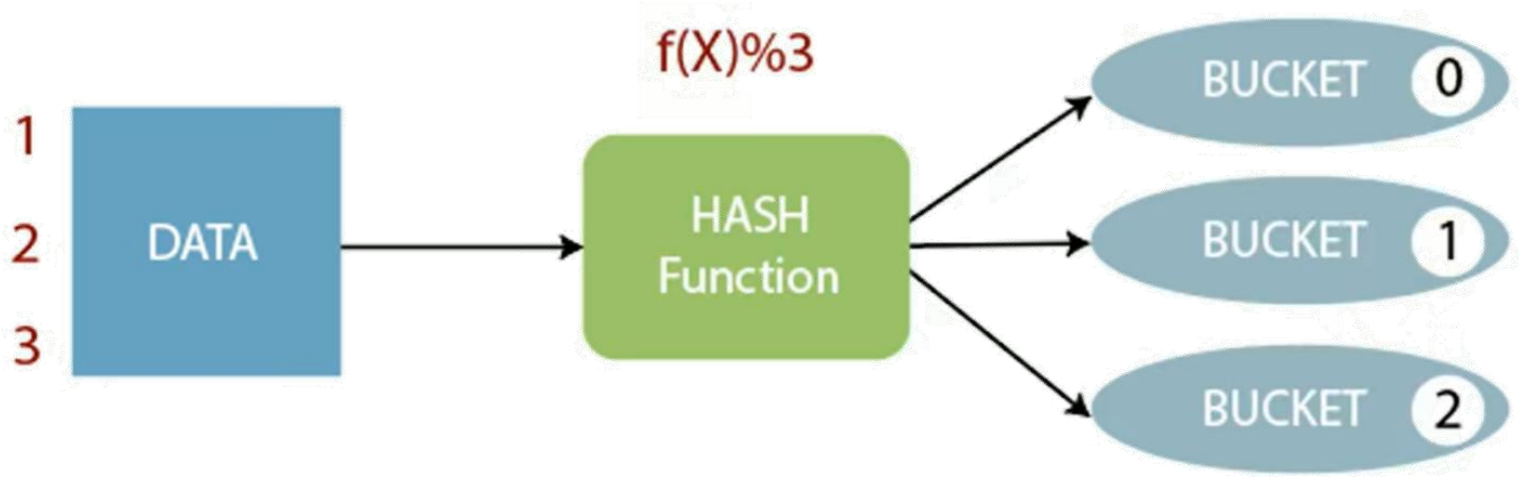
```
--- 1、order by
-- 根据字段进行排序
select * from t_usa_covid19_p
where count_date = "2021-01-28"
and state = "California"
order by deaths; -- 默认asc, nulls first 也可以手动指定nulls last
--(执行需要时间：35.475秒)


select * from t_usa_covid19_p
where count_date = "2021-01-28"
and state = "California"
order by deaths desc; -- 指定desc null last
--(执行需要时间：29.855秒)

-- 强烈建议将LIMIT与ORDER BY一起使用。避免数据集行数过大
-- 当hive.mapred.mode设置为strict严格模式时，使用不带LIMIT的ORDER BY时会引发异常。
select * from t_usa_covid19_p
where count_date = "2021-01-28"
and state = "California"
order by deaths desc
limit 3;
--(执行需要时间：28.344秒)
```

6.1.5 Hive SQL select查询高阶语法

CLUSTER BY

```
[ORDER BY col_list]
[CLUSTER BY col_list
  | [DISTRIBUTE BY col_list] [SORT BY col_list]
]
```

- 根据指定字段将数据分组，每组内再根据该字段正序排序（只能正序）。
- 概况起来就是：根据同一个字段，分且排序。
- 分组规则hash散列（分桶表规则一样）： Hash_Func(col_name) % reducetask个数
- 分为几组取决于reducetask的个数



6.1.5 Hive SQL select查询高阶语法

CLUSTER BY

```
-- 默认情况下 创建的表就是内部表
create table student(
num int,
name string,
sex string,
age int,
dept string)
row format delimited
fields terminated by ',';

cp ~/student.dat /usr/datas/.

LOAD DATA LOCAL inpath '/usr/datas/student.dat'
OVERWRITE INTO TABLE student;

SELECT * FROM student;
-- 不指定reduce task个数
-- 日志显示：Number of reduce tasks not specified. Estimated from input data size: 1
select * from student cluster by num;  --(执行需要时间：27.383秒)

-- 手动设置reduce task个数
set mapreduce.job.reduces = 2;
select * from student cluster by num; --(执行需要时间：39.043秒)
```

6.1.5 Hive SQL select查询高阶语法

CLUSTER BY

```
-- 默认情况下 创建的表就是内部表
create table student(
num int,
name string,
sex string,
age int,
dept string)
row format delimited
fields terminated by ',';

cp ~/student.dat /usr/datas/.

LOAD DATA LOCAL inpath '/usr/datas/student.dat'
OVERWRITE INTO TABLE student;

SELECT * FROM student;
-- 不指定reduce task个数
-- 日志显示：Number of reduce tasks not specified. Estimated from input data size: 1
select * from student cluster by num;  --(执行需要时间：27.383秒)

-- 手动设置reduce task个数
set mapreduce.job.reduces = 2;
select * from student cluster by num; --(执行需要时间：39.043秒)
```

6.1.5 Hive SQL select查询高阶语法

CLUSTER BY

```
0: jdbc:hive2://localhost:10000> select * from student cluster by num;
WARN  : Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions
g Hive 1.X releases.
+--------------+----------------+--------------+--------------+---------------+
| student.num  | student.name   | student.sex  | student.age  | student.dept  |
+--------------+----------------+--------------+--------------+---------------+
| 95002        | 刘晨             | 女            | 19           | IS            |
| 95004        | 张立             | 男            | 19           | IS            |
| 95006        | 孙庆             | 男            | 23           | CS            |
| 95008        | 李娜             | 女            | 18           | CS            |
| 95010        | 孔小涛            | 男            | 19           | CS            |
| 95012        | 孙花             | 女            | 20           | CS            |
| 95014        | 王小丽            | 女            | 19           | CS            |
| 95016        | 钱国             | 男            | 21           | MA            |
| 95018        | 王一             | 女            | 19           | IS            |
| 95020        | 赵钱             | 男            | 21           | IS            |
| 95022        | 郑明             | 男            | 20           | MA            |
```

6.1.5 Hive SQL select查询高阶语法

```
[ORDER BY col_list]
[CLUSTER BY col_list
  | [DISTRIBUTE BY col_list] [SORT BY col_list]
]
```

CLUSTER BY局限性

- 需求：根据sex性别分为两个部分，每个分组内再根据age年龄的倒序排序。
  - CLUSTER BY无法单独完成，因为分和排序的字段只能是同一个；
  - ORDER BY更不能在这里使用，因为是全局排序，只有一个输出，无法满足分的需求。

```
select * from student;
```

22 rows

	num	name	sex	age	dept
1	95001	李勇	男	20	CS
2	95002	刘晨	女	19	IS
3	95003	王敏	女	22	MA
4	95004	张立	男	19	IS
5	95005	刘刚	男	18	MA
6	95006	孙庆	男	23	CS
7	95007	易思玲	女	19	MA
8	95008	李娜	女	18	CS
9	95009	梦圆圆	女	18	MA
10	95010	孔小涛	男	19	CS
11	95011	包小柏	男	18	MA
12	95012	孙花	女	20	CS
13	95013	冯伟	男	21	CS
14	95014	王小丽	女	19	CS
15	95015	王君	男	18	MA

```
select * from student cluster by sex order by age desc;
select * from student cluster by sex sort by age desc;
```

```
adoop.hive.ql.parse.SemanticException:1:45 Cannot have both CLUSTER BY and SORT BY clauses. Error encountered near token 'age'
```

6.1.5 Hive SQL select查询高阶语法

```
[ORDER BY col_list]
[CLUSTER BY col_list
  | [DISTRIBUTE BY col_list] [SORT BY col_list]
]
```

DISTRIBUTED BY + SORT BY

- DISTRIBUTED BY +SORT BY就相当于把CLUSTER BY的功能一分为二：
  - DISTRIBUTED BY负责根据指定字段分组
  - SORT BY负责分组内排序规则
- 分组和排序的字段可以不同

```
--案例：把学生表数据根据性别分为两个部分，每个分组内根据年龄的倒序排序。
select *from student distribute by sex sort by age desc;
```

student.sno	student.sname	student.sex	student.sage	student.sdept
95006	孙庆	男	23	CS
95016	钱国	男	21	MA
95020	赵钱	男	21	IS
95013	冯伟	男	21	CS
95001	李勇	男	20	CS
95022	郑明	男	20	MA
95004	张立	男	19	IS
95010	孔小涛	男	19	CS
95011	包小柏	男	18	MA
95015	王君	男	18	MA
95005	刘刚	男	18	MA
95021	周二	男	17	MA
95003	王敏	女	22	MA
95012	孙花	女	20	CS
95014	王小丽	女	19	CS
95019	邢小丽	女	19	IS
95018	王一	女	19	IS
95007	易思玲	女	19	MA
95002	刘晨	女	19	IS
95008	李娜	女	18	CS
95009	梦圆圆	女	18	MA
95017	王风娟	女	18	IS

22 rows selected (21.822 seconds)

6.1.5 Hive SQL select查询高阶语法

```
[ORDER BY col_list]
[CLUSTER BY col_list
  | [DISTRIBUTE BY col_list] [SORT BY col_list]
]
```

DISTRIBUTED BY + SORT BY

- 如果DISTRIBUTED BY +SORT BY的字段一样：

  则： CLUSTER BY= DISTRIBUTED BY +SORT BY

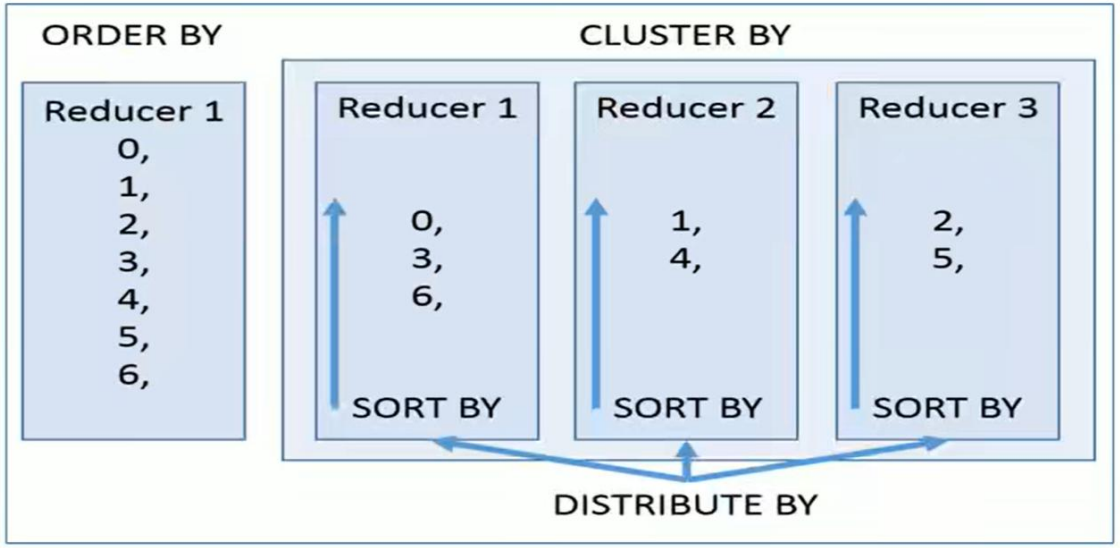
```
--DISTRIBUTED BY + SORT BY
--案例：把学生表数据根据性别分为两个部分，每个分组内根据年龄的倒序排序。
select * from student distribute by sex sort by age desc;
--(执行需要时间：39.828秒)

-- 下面两个语句执行结果一样
select * from student distribute by num sort by num limit 5;
--(执行需要时间：109.02秒)

select * from student cluster by num limit 5;
--(执行需要时间：109.02秒)
```

```
+-------------+--------------+-------------+-------------+--------------+
| student.num | student.name | student.sex | student.age | student.dept |
+-------------+--------------+-------------+-------------+--------------+
| 95001       | 李勇         | 男          | 20          | CS           |
| 95002       | 刘晨         | 女          | 19          | IS           |
| 95003       | 王敏         | 女          | 22          | MA           |
| 95004       | 张立         | 男          | 19          | IS           |
| 95005       | 刘刚         | 男          | 18          | MA           |
+-------------+--------------+-------------+-------------+--------------+
5 rows selected (109.02 seconds)
0: jdbc:hive2://localhost:10000> select * from student cluster by num limit 5;
WARN  : Hive-on-MR is deprecated in Hive 2 and may not be available in the future v
er using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
+-------------+--------------+-------------+-------------+--------------+
| student.num | student.name | student.sex | student.age | student.dept |
+-------------+--------------+-------------+-------------+--------------+
| 95001       | 李勇         | 男          | 20          | CS           |
| 95002       | 刘晨         | 女          | 19          | IS           |
| 95003       | 王敏         | 女          | 22          | MA           |
| 95004       | 张立         | 男          | 19          | IS           |
| 95005       | 刘刚         | 男          | 18          | MA           |
+-------------+--------------+-------------+-------------+--------------+
```

CLUSTER、DISTRIBUTE、SORT、ORDER BY

1. orderby全局排序，因此只有一个reducer，结果输出在一个文件中，当输入规模大时，需要较长的计算时间。
2. distribute by根据指定字段将数据分组，算法是hash散列。sort by是在分组之后，每个组内局部排序。
3. cluster by既有分组，又有排序，但是两个字段只能是同一个字段。

如果distribute和sort的字段是同一个时，此时，cluster by = distribute by + sort by



Union联合查询

- UNION用于将来自于多个SELECT语句的结果合并为一个结果集。
  1. 使用DISTINCT关键字与只使用UNION默认值效果一样，都会删除重复行。1.2.0之前的Hive版本仅支持UNION ALL，在这种情况下不会消除重复的行。
  2. 使用ALL关键字，不会删除重复行，结果集包括所有SELECT语句的匹配行（包括重复行）。
  3. 每个select_statement返回的列的数量和名称必须相同。

```
select_statement
        UNION [ALL | DISTINCT]
select_statement
        UNION [ALL | DISTINCT]
select_statement ...;
```

From子句中子查询（Subqueries）

- 在Hive 0.12版本，仅在FROM子句中支持子查询。
- 必须要给子查询一个名称，因为FROM子句中的每个表都必须有一个名称。子查询返回结果中的列必须具有唯一的名称。子查询返回结果中的列在外部查询中可用，就像真实表的列一样。子查询也可以是带有UNION的查询表达式。
- Hive支持任意级别的子查询，也就是所谓的嵌套子查询。
- Hive 0.13.0和更高版本中的子查询名称之前可以包含可选关键字AS。

```
--from子句中子查询（Subqueries）
--子查询
SELECT num
FROM (
        select num,name from student_local
    ) tmp;

--包含UNION ALL的子查询的示例
SELECT t3.name
FROM (
        select num,name from student_local
        UNION distinct
        select num,name from student_hdfs
    ) t3;
```

where子句中子查询（Subqueries)

- 从Hive 0.13开始，WHERE子句支持下述类型的子查询：
  1. 不相关子查询：该子查询不引用父查询中的列，可以将查询结果视为IN和NOT IN语句的常量；
  2. 相关子查询：子查询引用父查询中的列；

```
-- where子句中子查询（Subqueries）
-- 不相关子查询，相当于IN、NOT IN，子查询只能选择一个列。
--  （1）执行子查询，其结果不被显示，而是传递给外部查询，作为外部查询的条件使用。
--  （2）执行外部查询，并显示整个结果。

SELECT *
FROM student_hdfs
WHERE student_hdfs.num IN (select num from student_local limit 2);
--(执行需要时间： 95.502秒)

-- 相关子查询，指 EXISTS和NOT EXISTS子查询
-- 子查询的WHERE子句中支持对父查询的引用

SELECT A
FROM T1
WHERE EXISTS (SELECT B FROM T2 WHERE T1.X = T2.Y);
```

6.1.6 Common Table Expressions(CTE)

CTE介绍

- 公用表表达式(CTE)是一个临时结果集：该结果集是从WITH子句中指定的简单查询派生而来的，紧接在SELECT或INSERT关键字之前。
- CTE仅在单个语句的执行范围内定义。
- CTE可以在 SELECT，INSERT，CREATE TABLE AS SELECT或CREATE VIEW AS SELECT语句中使用。

```
[WITH CommonTableExpression (, CommonTableExpression)*]
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
  FROM table_reference
  [WHERE where_condition]
  [GROUP BY col_list]
  [ORDER BY col_list]
  [CLUSTER BY col_list
    | [DISTRIBUTE BY col_list] [SORT BY col_list]
  ]
 [LIMIT [offset,] rows];
```