

大数据数据库系统

6.5 HQL的基本使用

6.5 HQL的基本使用

◆ 主要内容

1、基本查询

- 全表查询与特定列查询 (select...from)
- 列别名
- 算术运算
- limit语句
- where语句
- 比较运算 (between/in/is NULL)、逻辑运算
- 聚合函数
- LIKE

6.5 HQL的基本使用

◆ 主要内容

2、分组查询

- Group By 语句
- Having语句

3、连接 (Join on语句)

内连接、左连接、右连接、全连接、多表连接

4、排序

基本查询

◆ 本节课用到的两张表:

```
hive (emp_test)> select * from emp;
OK
7369 SMITH      CLERK    7902    1980-12-17    800.0    NULL    20
7499 ALLEN      SALESMAN    7698    1981-2-20    1600.0    300.0    30
7521 WARD      SALESMAN    7698    1981-2-22    1250.0    500.0    30
7566 JONES     MANAGER    7839    1981-4-2     2975.0    NULL    20
7654 MARTIN    SALESMAN    7698    1981-9-28    1250.0    1400.0    30
7698 BLAKE     MANAGER    7839    1981-5-1     2850.0    NULL    30
7782 CLARK     MANAGER    7839    1981-6-9     2450.0    NULL    10
7788 SCOTT     ANALYST    7566    1987-4-19    3000.0    NULL    20
7839 KING     PRESIDENT    NULL    1981-11-17    5000.0    NULL    10
7844 TURNER    SALESMAN    7698    1981-9-8     1500.0    0.0     30
7876 ADAMS     CLERK      7788    1987-5-23    1100.0    NULL    20
7900 JAMES     CLERK      7698    1981-12-3    950.0     NULL    30
7902 FORD      ANALYST    7566    1981-12-3    3000.0    NULL    20
7934 MILLER    CLERK      7782    1982-1-23    1300.0    NULL    10
Time taken: 0.048 seconds, Fetched: 14 row(s)
```

```
hive (emp_test)> select * from dept;
OK
10      ACCOUNTING      NEW YORK
20      RESEARCH           DALLAS
30      SALES             CHICAGO
40      OPERATIONS        BOSTON
Time taken: 0.034 seconds, Fetched: 4 row(s)
hive (emp_test)>
```

基本查询

◆ 全表查询与特定列查询 (select...from)

◆ 全表查询: select * from emp;

```
hive (emp_test)> select * from emp;
OK
emp.empno emp.ename emp.job emp.mgr emp.hiredate emp.sal emp.comm emp.deptno
7369 SMITH CLERK 7902 1980-12-17 800.0 NULL 20
7499 ALLEN SALESMAN 7698 1981-2-20 1600.0 300.0 30
7521 WARD SALESMAN 7698 1981-2-22 1250.0 500.0 30
7566 JONES MANAGER 7839 1981-4-2 2975.0 NULL 20
7654 MARTIN SALESMAN 7698 1981-9-28 1250.0 1400.0 30
7698 BLAKE MANAGER 7839 1981-5-1 2850.0 NULL 30
7782 CLARK MANAGER 7839 1981-6-9 2450.0 NULL 10
7788 SCOTT ANALYST 7566 1987-4-19 3000.0 NULL 20
7839 KING PRESIDENT NULL 1981-11-17 5000.0 NULL 10
7844 TURNER SALESMAN 7698 1981-9-8 1500.0 0.0 30
7876 ADAMS CLERK 7788 1987-5-23 1100.0 NULL 20
7900 JAMES CLERK 7698 1981-12-3 950.0 NULL 30
7902 FORD ANALYST 7566 1981-12-3 3000.0 NULL 20
7934 MILLER CLERK 7782 1982-1-23 1300.0 NULL 10
Time taken: 0.039 seconds, Fetched: 14 row(s)
```

◆ 特定列查询: select empno, ename, sal, deptno from emp;

```
hive (emp_test)> select empno, ename, sal, deptno from emp;
empno ename sal deptno
7369 SMITH 800.0 20
7499 ALLEN 1600.0 30
7521 WARD 1250.0 30
7566 JONES 2975.0 20
7654 MARTIN 1250.0 30
7698 BLAKE 2850.0 30
7782 CLARK 2450.0 10
7788 SCOTT 3000.0 20
7839 KING 5000.0 10
7844 TURNER 1500.0 30
7876 ADAMS 1100.0 20
7900 JAMES 950.0 30
7902 FORD 3000.0 20
7934 MILLER 1300.0 10
Time taken: 15.951 seconds, Fetched: 14 row(s)
```

基本查询

◆ 几个注意点：

- (1) SQL 语言大小写不敏感
- (2) SQL 可以写在一行或者多行
- (3) 关键字不能被缩写也不能分行
- (4) 各子句一般要分行写
- (5) 使用缩进提高语句的可读性

基本查询

◆ 指定列别名

查询名称和部门

```
select ename AS name, deptno dn from emp;
```

- 1) 重命名一个列
- 2) 便于计算
- 3) 紧跟列名，也可以在列名和别名之间加入关键字 ‘AS’

基本查询

◆ 算术运算符

select后的字段可以使用算术运算符

例：查询出所有员工的薪水后加 1 显示。

```
select sal +1 from emp;
```

运算符	描述
A+B	A 和 B 相加
A-B	A 减去 B
A*B	A 和 B 相乘
A/B	A 除以 B
A%B	A 对 B 取余
A&B	A 和 B 按位取与
A B	A 和 B 按位取或
A^B	A 和 B 按位取异或
~A	A 按位取反

基本查询

◆ Where、limit、distinct语句

1、查询部门编号是30的员工

```
select * from emp where deptno='30';
```

2、Limit用于限制返回的行数，例：查看前3条记录

```
select * from emp limit 3;
```

```
hive (emp_test)> select * from emp limit 3;
OK
emp.empno emp.ename emp.job emp.mgr emp.hiredate emp.sal emp.comm emp.deptno
7369 SMITH CLERK 7902 1980-12-17 800.0 NULL 20
7499 ALLEN SALESMAN 7698 1981-2-20 1600.0 300.0 30
7521 WARD SALESMAN 7698 1981-2-22 1250.0 500.0 30
Time taken: 0.131 seconds, Fetched: 3 row(s)
hive (emp_test)>
```

3、去重语句distinct，例：查询当前有哪些部门

```
select distinct deptno from emp;
```

```
OK
deptno
10
20
30
Time taken: 24.175 seconds, Fetched: 3 row(s)
hive (emp_test)> █
```

◆ 比较运算符（Between/In/ Is Null等）

下表中描述了比较运算符，这些操作符可以用于WHERE、JOIN...ON 和 HAVING 语句中

操作符	支持的数据类型	描述
A=B	基本数据类型	如果 A 等于 B 则返回 TRUE，反之返回 FALSE
A<B	基本数据类型	A 或者 B 为 NULL，则返回 NULL；如果 A 小于 B，则返回 TRUE，反之返回 FALSE
A<=B	基本数据类型	A 或者 B 为 NULL，则返回 NULL；如果 A 小于等于 B，则返回 TRUE，反之返回 FALSE
A>B	基本数据类型	A 或者 B 为 NULL，则返回 NULL；如果 A 大于 B，则返回 TRUE，反之返回 FALSE
A>=B	基本数据类型	A 或者 B 为 NULL，则返回 NULL；如果 A 大于等于 B，则返回 TRUE，反之返回 FALSE
A [NOT] BETWEEN B AND C	基本数据类型	如果 A, B 或者 C 任一为 NULL，则结果为 NULL。如果 A 的值大于等于 B 而且小于或等于 C，则结果为 TRUE，反之为 FALSE。如果使用 NOT 关键字则可达到相反的效果。

基本查询

- ◆ 下表中描述了谓词操作符，这些操作符同样可以用于WHERE JOIN...ON 和 HAVING 语句中

A IS NULL	所有数据类型	如果 A 等于 NULL，则返回 TRUE，反之返回 FALSE
A IS NOT NULL	所有数据类型	如果 A 不等于 NULL，则返回 TRUE，反之返回 FALSE
IN(数值 1, 数值 2)	所有数据类型	使用 IN 运算显示列表中的值
A [NOT] LIKE B	STRING 类型	B 是一个 SQL 下的简单正则表达式，也叫通配符模式，如果 A 与其匹配的话，则返回 TRUE；反之返回 FALSE。B 的表达式说明如下：‘x%’表示 A 必须以字母‘x’开头，‘%x’表示 A 必须以字母‘x’结尾，而‘%x%’表示 A 包含有字母‘x’，可以位于开头，结尾或者字符串中间。如果使用 NOT 关键字则可达到相反的效果。

4、查询员工编号大于7500: select * from emp where empno>7500;

```
Total MapReduce CPU Time Spent: 1 seconds 230 msec
OK
emp.empno emp.ename emp.job emp.mgr emp.hiredate emp.sal emp.comm emp.deptno
7521 WARD SALESMAN 7698 1981-2-22 1250.0 500.0 30
7566 JONES MANAGER 7839 1981-4-2 2975.0 NULL 20
7654 MARTIN SALESMAN 7698 1981-9-28 1250.0 1400.0 30
7698 BLAKE MANAGER 7839 1981-5-1 2850.0 NULL 30
7782 CLARK MANAGER 7839 1981-6-9 2450.0 NULL 10
7788 SCOTT ANALYST 7566 1987-4-19 3000.0 NULL 20
7839 KING PRESIDENT NULL 1981-11-17 5000.0 NULL 10
7844 TURNER SALESMAN 7698 1981-9-8 1500.0 0.0 30
7876 ADAMS CLERK 7788 1987-5-23 1100.0 NULL 20
7900 JAMES CLERK 7698 1981-12-3 950.0 NULL 30
7902 FORD ANALYST 7566 1981-12-3 3000.0 NULL 20
7934 MILLER CLERK 7782 1982-1-23 1300.0 NULL 10
Time taken: 15.673 seconds, Fetched: 12 row(s)
hive (emp_test)>
```

5、查询薪资在2000到3000之间: select * from emp where sal between 2000 and 3000;

```
Job 0: Map: 1 Cumulative CPU: 1.28 sec HDFS Read: 893 HDFS Write: 231 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 280 msec
OK
emp.empno emp.ename emp.job emp.mgr emp.hiredate emp.sal emp.comm emp.deptno
7566 JONES MANAGER 7839 1981-4-2 2975.0 NULL 20
7698 BLAKE MANAGER 7839 1981-5-1 2850.0 NULL 30
7782 CLARK MANAGER 7839 1981-6-9 2450.0 NULL 10
7788 SCOTT ANALYST 7566 1987-4-19 3000.0 NULL 20
7902 FORD ANALYST 7566 1981-12-3 3000.0 NULL 20
Time taken: 15.788 seconds, Fetched: 5 row(s)
```

6、查询奖金不为空的员工: select * from emp where comm is not null;

```
emp.empno emp.ename emp.job emp.mgr emp.hiredate emp.sal emp.comm emp.deptno
7499 ALLEN SALESMAN 7698 1981-2-20 1600.0 300.0 30
7521 WARD SALESMAN 7698 1981-2-22 1250.0 500.0 30
7654 MARTIN SALESMAN 7698 1981-9-28 1250.0 1400.0 30
7844 TURNER SALESMAN 7698 1981-9-8 1500.0 0.0 30
Time taken: 15.772 seconds, Fetched: 4 row(s)
hive (emp_test)>
```

7、查询工资是 1500 或 5000 的员工信息: select * from emp where sal IN (1500, 5000);

```
hive (default)> select * from emp where sal IN (1500, 5000);
OK
emp.empno emp.ename emp.job emp.mgr emp.hiredate emp.sal emp.comm emp.deptno
7839 KING PRESIDENT NULL 1981-11-17 5000.0 NULL 10
7844 TURNER SALESMAN 7698 1981-9-8 1500.0 0.0 30
Time taken: 0.127 seconds, Fetched: 2 row(s)
```

基本查询

◆ 逻辑运算符 (And/Or/Not)

操作符	含义
AND	逻辑并
OR	逻辑或
NOT	逻辑否

1、查询薪水大于 1000，部门是 30

```
hive (default)> select * from emp where sal>1000 and deptno=30;
```

```
hive (default)> select * from emp where sal>1000 and deptno=30;
```

OK

emp.empno	emp.ename	emp.job	emp.mgr	emp.hiredate	emp.sal	emp.comm	e
7499	ALLEN	SALESMAN	7698	1981-2-20	1600.0	300.0	30
7521	WARD	SALESMAN	7698	1981-2-22	1250.0	500.0	30
7654	MARTIN	SALESMAN	7698	1981-9-28	1250.0	1400.0	30
7698	BLAKE	MANAGER 7839	1981-5-1	2850.0	NULL	30	
7844	TURNER	SALESMAN	7698	1981-9-8	1500.0	0.0	30

Time taken: 0.128 seconds, Fetched: 5 row(s)

```
hive (default)>
```

2、查询薪水大于 1000，或者部门是 30

```
hive (default)> select * from emp where sal>1000 or deptno=30;
```

```
hive (default)> select * from emp where sal>1000 or deptno=30;
```

OK

emp.empno	emp.ename	emp.job	emp.mgr	emp.hiredate	emp.sal	emp.comm	e
7499	ALLEN	SALESMAN	7698	1981-2-20	1600.0	300.0	30
7521	WARD	SALESMAN	7698	1981-2-22	1250.0	500.0	30
7566	JONES	MANAGER 7839	1981-4-2	2975.0	NULL	20	
7654	MARTIN	SALESMAN	7698	1981-9-28	1250.0	1400.0	30
7698	BLAKE	MANAGER 7839	1981-5-1	2850.0	NULL	30	
7782	CLARK	MANAGER 7839	1981-6-9	2450.0	NULL	10	
7788	SCOTT	ANALYST 7566	1987-4-19	3000.0	NULL	20	
7839	KING	PRESIDENT	NULL	1981-11-17	5000.0	NULL	10
7844	TURNER	SALESMAN	7698	1981-9-8	1500.0	0.0	30
7876	ADAMS	CLERK 7788	1987-5-23	1100.0	NULL	20	
7900	JAMES	CLERK 7698	1981-12-3	950.0	NULL	30	
7902	FORD	ANALYST 7566	1981-12-3	3000.0	NULL	20	
7934	MILLER	CLERK 7782	1982-1-23	1300.0	NULL	10	

Time taken: 0.129 seconds, Fetched: 13 row(s)

3、查询除了 10 部门和 20 部门以外的员工信息

```
hive (default)> select * from emp where deptno not IN(10, 20);
```

```
hive (default)> select * from emp where comm is not null;
```

OK

emp.empno	emp.ename	emp.job	emp.mgr	emp.hiredate	emp.sal	emp.comm	e
7499	ALLEN	SALESMAN	7698	1981-2-20	1600.0	300.0	30
7521	WARD	SALESMAN	7698	1981-2-22	1250.0	500.0	30
7654	MARTIN	SALESMAN	7698	1981-9-28	1250.0	1400.0	30
7844	TURNER	SALESMAN	7698	1981-9-8	1500.0	0.0	30

Time taken: 0.105 seconds, Fetched: 4 row(s)

基本查询

◆ 常用聚合函数

select后的字段可以使用一些常用函数进行计算

1) 求总行数 (count) , 求指定列的非NULL行数count(column)

```
hive (default)> select count(*) cnt from emp;
```

2) 求工资的最大值 (max)

```
hive (default)> select max(sal) max_sal from emp;
```

3) 求工资的最小值 (min)

```
hive (default)> select min(sal) min_sal from emp;
```

4) 求工资的总和 (sum)

```
hive (default)> select sum(sal) sum_sal from emp;
```

5) 求工资的平均值 (avg)

```
hive (default)> select avg(sal) avg_sal from emp;
```


基本查询

◆ count(1)、count(列名)与 count(*)

- count(1) and count(*): 都是求表的总行数count(*)包括了所有的列，相当于求记录总行数，在统计结果的时候，不会忽略NULL
- count(列名) 会统计该列字段记录个数，会忽略字段为null 的情况，即不统计字段为null 的记录

1、查看emp里有多少条数据：

```
select count(*) cnt from emp;
```

```
Total MapReduce CPU Time Spent: 1 seconds 990 msec
OK
cnt
14
Time taken: 23.685 seconds, Fetched: 1 row(s)
```

2、求emp里最高工资：

```
select max(sal) max_sal from emp;
```

```
Total MapReduce CPU Time Spent: 2 seconds 20 msec
OK
max_sal
5000.0
Time taken: 23.791 seconds, Fetched: 1 row(s)
hive (emp_test)>
```

3、求emp里工资总和与平均工资：

```
select sum(sal), avg(sal) from emp;
```

```
Total MapReduce CPU Time Spent: 4 seconds 710 msec
OK
_c0      _c1
29025.0  2073.214285714286
Time taken: 23.677 seconds, Fetched: 1 row(s)
hive (default)>
```

基本查询

◆ LIKE语句

使用 LIKE 运算选择类似的值

✓ 选择条件可以包含字符或数字:

- % 代表零个或多个字符(任意个字符)。
- _ 代表一个字符。

1、查找名字以 A 开头的员工信息

hive (default)> select * from emp where ename LIKE 'A%';

```
hive (default)> select * from emp where ename LIKE 'A%';
OK
emp.empno      emp.ename      emp.job emp.mgr emp.hiredate      emp.sal emp.comm e
emp.deptno
7499      ALLEN      SALESMAN      7698      1981-2-20      1600.0 300.0 30
7876      ADAMS      CLERK      7788      1987-5-23      1100.0 NULL 20
Time taken: 0.132 seconds, Fetched: 2 row(s)
```

2、查找名字中第二个字母为 A 的员工信息

hive (default)> select * from emp where ename LIKE '_A%';

```
hive (default)> select * from emp where ename LIKE '_A%';
OK
emp.empno      emp.ename      emp.job emp.mgr emp.hiredate      emp.sal emp.comm e
emp.deptno
7521      WARD      SALESMAN      7698      1981-2-22      1250.0 500.0 30
7654      MARTIN      SALESMAN      7698      1981-9-28      1250.0 1400.0 30
7900      JAMES      CLERK      7698      1981-12-3      950.0 NULL 30
Time taken: 0.109 seconds, Fetched: 3 row(s)
```

分组查询

◆ Group By, having 语句

- GROUP BY 语句通常会和聚合函数一起使用，按照一个或者多个列队结果进行分组，对每个组执行聚合操作。
- having 只用于 group by 分组统计语句

1、求每个部门的平均工资

```
select deptno,avg(sal) from emp group by deptno;
```

```
Job 0: Map: 1 Reduce: 1 Cumulative CPU: 1.99 sec HDFS Read: 893 HDFS Write: 54 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 990 msec
OK
deptno    _c1
10        2916.6666666666665
20        2175.0
30        1566.6666666666667
Time taken: 23.576 seconds, Fetched: 3 row(s)
```

2、求部门平均工资大于2000的部门以及平均工资

```
select deptno,avg(sal) avg from emp group by deptno having avg >2000;
```

```
Total MapReduce CPU Time Spent: 2 seconds 440 msec
OK
deptno    avg
10        2916.6666666666665
20        2175.0
Time taken: 24.169 seconds, Fetched: 2 row(s)
```

连接（Join on语句）

◆ 两张表，均有deptno作为外键

```
create table emp(  
  empno int,  
  ename string,  
  job string,  
  mgr int,  
  hiredate string,  
  sal double,  
  comm double,  
  deptno int  
)  
row format delimited fields terminated by '\t';  
load data local inpath '/opt/datas/emp.txt' into  
  table emp;
```

```
create table dept(  
  deptno int,  
  dname string,  
  loc string  
)  
row format delimited fields terminated by  
  '\t';  
load data local inpath '/opt/datas/dept.txt'  
  overwrite into table dept;
```

连接（Join on语句）

◆ Join语句

等值join（inner join）可以将两张表都有的字段进行连接

```
select a.empno,a.ename,a.sal,b.deptno,b.dname from emp a inner join  
dept b on a.deptno=b.deptno;
```

a.empno	a.ename	a.sal	b.deptno	b.dname
7369	SMITH	800.0	20	RESEARCH
7499	ALLEN	1600.0	30	SALES
7521	WARD	1250.0	30	SALES
7566	JONES	2975.0	20	RESEARCH
7654	MARTIN	1250.0	30	SALES
7698	BLAKE	2850.0	30	SALES
7782	CLARK	2450.0	10	ACCOUNTING
7788	SCOTT	3000.0	20	RESEARCH
7839	KING	5000.0	10	ACCOUNTING
7844	TURNER	1500.0	30	SALES
7876	ADAMS	1100.0	20	RESEARCH
7900	JAMES	950.0	30	SALES
7902	FORD	3000.0	20	RESEARCH
7934	MILLER	1300.0	10	ACCOUNTING

Time taken: 20.602 seconds, Fetched: 14 row(s)
hive (emp_test)>

给表定义别名，
增加可读性

◆ Join语句

left join: 左表的字段为基准

- select a.empno,a.ename,a.sal,b.deptno,b.dname from emp a left join dept b on a.deptno=b.deptno;

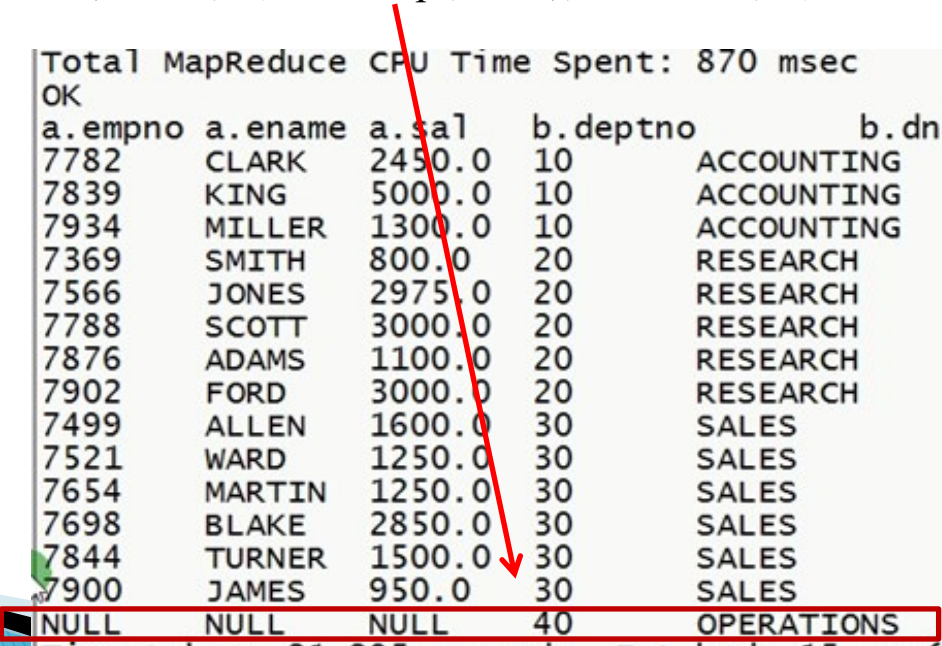
right join: 以右表的字段为基准

- select a.empno,a.ename,a.sal,b.deptno,b.dname from emp a right join dept b on a.deptno=b.deptno;
- Dept表中有40这个部门而emp表没有人在40这个部门里

Total MapReduce CPU Time Spent: 870 msec
OK

a.empno	a.ename	a.sal	b.deptno	b.dname
7782	CLARK	2450.0	10	ACCOUNTING
7839	KING	5000.0	10	ACCOUNTING
7934	MILLER	1300.0	10	ACCOUNTING
7369	SMITH	800.0	20	RESEARCH
7566	JONES	2975.0	20	RESEARCH
7788	SCOTT	3000.0	20	RESEARCH
7876	ADAMS	1100.0	20	RESEARCH
7902	FORD	3000.0	20	RESEARCH
7499	ALLEN	1600.0	30	SALES
7521	WARD	1250.0	30	SALES
7654	MARTIN	1250.0	30	SALES
7698	BLAKE	2850.0	30	SALES
7844	TURNER	1500.0	30	SALES
7900	JAMES	950.0	30	SALES
NULL	NULL	NULL	40	OPERATIONS

Time taken: 21.295 seconds, Fetched: 15 row(s)
hive (emp_test)>



连接 (Join on语句)

```
hive (emp_test)> select * from emp;
OK
7369 SMITH CLERK 7902 1980-12-17 800.0 NULL 20
7499 ALLEN SALESMAN 7698 1981-2-20 1600.0 300.0 30
7521 WARD SALESMAN 7698 1981-2-22 1250.0 500.0 30
7566 JONES MANAGER 7839 1981-4-2 2975.0 NULL 20
7654 MARTIN SALESMAN 7698 1981-9-28 1250.0 1400.0 30
7698 BLAKE MANAGER 7839 1981-5-1 2850.0 NULL 30
7782 CLARK MANAGER 7839 1981-6-9 2450.0 NULL 10
7788 SCOTT ANALYST 7566 1987-4-19 3000.0 NULL 20
7839 KING PRESIDENT NULL 1981-11-17 5000.0 NULL 10
7844 TURNER SALESMAN 7698 1981-9-8 1500.0 0.0 30
7876 ADAMS CLERK 7788 1987-5-23 1100.0 NULL 20
7900 JAMES CLERK 7698 1981-12-3 950.0 NULL 30
7902 FORD ANALYST 7566 1981-12-3 3000.0 NULL 20
7934 MILLER CLERK 7782 1982-1-23 1300.0 NULL 10
Time taken: 0.048 seconds, Fetched: 14 row(s)
```

```
hive (emp_test)> select * from dept;
OK
10 ACCOUNTING NEW YORK
20 RESEARCH DALLAS
30 SALES CHICAGO
40 OPERATIONS BOSTON
Time taken: 0.034 seconds, Fetched: 4 row(s)
hive (emp_test)>
```

◆ Join语句

full join: 以两张表中所有的字段为基准

```
select a.empno,a.ename,a.sal,b.deptno,b.dname from emp a full join dept b on  
a.deptno=b.deptno;
```

a.empno	a.ename	a.sal	b.deptno	b.dname
7934	MILLER	1300.0	10	ACCOUNTING
7839	KING	5000.0	10	ACCOUNTING
7782	CLARK	2450.0	10	ACCOUNTING
7876	ADAMS	1100.0	20	RESEARCH
7788	SCOTT	3000.0	20	RESEARCH
7369	SMITH	800.0	20	RESEARCH
7566	JONES	2975.0	20	RESEARCH
7902	FORD	3000.0	20	RESEARCH
7844	TURNER	1500.0	30	SALES
7499	ALLEN	1600.0	30	SALES
7698	BLAKE	2850.0	30	SALES
7654	MARTIN	1250.0	30	SALES
7521	WARD	1250.0	30	SALES
7900	JAMES	950.0	30	SALES
NULL	NULL	NULL	40	OPERATIONS

Time taken: 30.972 seconds, Fetched: 15 row(s)
hive (emp_test)>

连接（Join on语句）

◆ Join语句

多表连接，连接多张表，但连接 n 个表，至少需要 $n-1$ 个连接条件。

例如：连接三个表，至少需要两个连接条件。

连接（Join on语句）

◆ Join语句

多表连接查询：查询员工姓名（来自emp表）、部门名称（来自dept表）、以及部门所在城市名称（来自location表）

```
hive (default)>SELECT e.ename, d.dname, l.loc_name
```

```
FROM emp e
```

```
JOIN dept d
```

```
ON d.deptno = e.deptno
```

```
JOIN location l
```

```
ON d.loc = l.loc;
```

```
Total MapReduce CPU Time Spent: 2 seconds 910 msec
OK
e.ename d.dname l.loc_name
SMITH RESEARCH London
ALLEN SALES Tokyo
WARD SALES Tokyo
JONES RESEARCH London
MARTIN SALES Tokyo
BLAKE SALES Tokyo
CLARK ACCOUNTING Beijing
SCOTT RESEARCH London
KING ACCOUNTING Beijing
TURNER SALES Tokyo
ADAMS RESEARCH London
JAMES SALES Tokyo
FORD RESEARCH London
MILLER ACCOUNTING Beijing
Time taken: 29.688 seconds, Fetched: 14 row(s)
hive (default)>
```


连接（Join on语句）

◆ 对多表连接的说明

```
hive (default)>SELECT e.ename, d.dname, l.loc_name
```

```
FROM emp e
```

```
JOIN dept d
```

```
ON d.deptno = e.deptno
```

```
JOIN location l
```

```
ON d.loc = l.loc_name;
```

- 大多数情况下，Hive 会对每对 JOIN 连接对象启动一个 MapReduce 任务，即首先启动一个 MapReduce job 对表 e 和表 d 进行连接操作，然后会再启动一个 MapReduce job 将第一个 MapReduce job 的输出和表 l 进行连接操作。
- 为什么不是表 d 和表 l 先进行连接操作呢？这是因为 Hive 总是按照从左到右的顺序执行的

排序

◆ 全局排序order by

只使用一个reducer，对某一列进行全局排序

- ASC (ascend) : 升序 (默认)
- DESC (descend) : 降序

注意：应放在 SELECT 语句的结尾，效率较低，海量数据不能只靠1个reduce排序

例：select empno,ename,deptno,sal from emp order by sal desc;

Job 0: Map: 1 Reduce: 1 Cumulative CPU: 2.02 sec HDFS Read: 893 HDFS Write: 292 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 20 msec

OK

empno	ename	deptno	sal
7839	KING	10	5000.0
7902	FORD	20	3000.0
7788	SCOTT	20	3000.0
7566	JONES	20	2975.0
7698	BLAKE	30	2850.0
7782	CLARK	10	2450.0
7499	ALLEN	30	1600.0
7844	TURNER	30	1500.0
7934	MILLER	10	1300.0
7654	MARTIN	30	1250.0
7521	WARD	30	1250.0
7876	ADAMS	20	1100.0
7900	JAMES	30	950.0
7369	SMITH	20	800.0



排序

◆ 内部排序Sort By

每个 Reducer 内部排序 (Sort By)

- 对于大规模的数据集 order by 的效率非常低。
- 在很多情况下，并不需要全局排序，此时可以使用 sort by
- Sort by 为每个 reducer 产生一个排序文件，每个 reducer内部进行排序，如果只有一个reducer，等同于order by；大于1个reducer的话对全局结果集来说不是排序

排序

◆ 首先，需要设置reducer的数目（以下设置2个）

```
set mapreduce.job.reduces=2
```

set mapreduce.job.reduces查看目前启动多少个reduce

◆ 执行语句：

```
insert overwrite local directory '/opt/datas/sort' select  
empno,ename,deptno,sal from emp sort by sal desc;
```

由于我们设置了两个reduce，因此，如果使用insert overwrite local directory 指令导出数据的话，就会有两个数据文件，并且这两个数据文件内的数据是排序的。如果只有一个reducer，等同于order by；

1、执行后在‘/opt/datas/sort/’目录下可以看到两个数据文件

```
[hpsk@bigdata-training01 datas]$ cd sort/
[hpsk@bigdata-training01 sort]$ ll
total 8
-rw-r--r-- 1 hpsk hpsk 187 May 24 13:05 000000_0
-rw-r--r-- 1 hpsk hpsk 105 May 24 13:05 000001_0
[hpsk@bigdata-training01 sort]$
```

2、分别使用more命令查看这两个数据文件

```
[hpsk@bigdata-training01 sort]$ more 000000_0
7902 FORD 20 3000.0
7788 SCOTT 20 3000.0
7566 JONES 20 2975.0
7844 TURNER 30 1500.0
7521 WARD 30 1250.0
7654 MARTIN 30 1250.0
7876 ADAMS 20 1100.0
7900 JAMES 30 950.0
7369 SMITH 20 800.0
[hpsk@bigdata-training01 sort]$
```

```
[hpsk@bigdata-training01 sort]$ more 000001_0
7839 KING 10 5000.0
7698 BLAKE 30 2850.0
7782 CLARK 10 2450.0
7499 ALLEN 30 1600.0
7934 MILLER 10 1300.0
[hpsk@bigdata-training01 sort]$
```

从结果可以看出这两个数据文件均按照降序排列

问：为什么按照这样划分呢？如何指定哪些数据划分给哪个reduce呢？

答：随机划分

distribute by

排序

◆ 分区排序 `distribute by`

`distribute by`: 对数据按照某个字段进行分区, 交给不同的 `reduce` 进行处理

注意:

- 1、 `distribute by` 的分区规则是根据分区字段的 `hash` 码与 `reducer` 的个数进行模除后, 余数相同的分到一个区。
- 2、 Hive 要求 `DISTRIBUTE BY` 语句要写在 `SORT BY` 语句之前。

例: 先按照员工号分区, 再按薪资降序

```
insert overwrite local directory '/opt/datas/distribute' select empno,ename,deptno,sal from emp  
distribute by empno sort by sal desc;
```

1、执行后在‘/opt/datas/distribute/’目录下可以看到两个数据文件

```
[hpsk@bigdata-training01 datas]$ cd distribute/  
[hpsk@bigdata-training01 distribute]$ ll  
total 8  
-rw-r--r-- 1 hpsk hpsk 211 May 24 13:09 000000_0  
-rw-r--r-- 1 hpsk hpsk 81 May 24 13:09 000001_0
```

2、分别使用more命令查看这两个数据文件

```
[hpsk@bigdata-training01 distribute]$ more 000000_0  
7788 SCOTT 20 3000.0  
7902 FORD 20 3000.0  
7566 JONES 20 2975.0  
7698 BLAKE 30 2850.0  
7782 CLARK 10 2450.0  
7844 TURNER 30 1500.0  
7934 MILLER 10 1300.0  
7654 MARTIN 30 1250.0  
7876 ADAMS 20 1100.0  
7900 JAMES 30 950.0  
  
[hpsk@bigdata-training01 distribute]$ more 000001_0  
7839 KING 10 5000.0  
7499 ALLEN 30 1600.0  
7521 WARD 30 1250.0  
7369 SMITH 20 800.0  
.. .. ..
```

从结果可以看出这两个数据文件均按照降序排列，但原先7521号员工到了000001_0中
Empno/2取余数分配reducer，正好奇偶划分

问：如果指定了多个reducer，如果还用order by语句排序，会是什么结果呢？

排序

问：如果指定了多个reduce，如果还用order by语句排序，会是什么结果呢？

输入代码：

```
set mapreduce.job.reduces=2;
```

```
insert overwrite local directory '/opt/datas/order' select empno,ename,deptno,sal from emp  
order by sal desc;
```

```
hive (emp_test)> insert overwrite local directory '/opt/datas/order' select empno,ename,deptno,sal fro  
m emp order by sal desc;  
Total jobs = 1  
Launching Job 1 out of 1  
Number of reduce tasks determined at compile time: 1  
In order to change the average load for a reducer (in bytes):
```

可以看出，即使设置多个reduce，也只启动一个reduce

总结

1、基本查询

- 全表查询与特定列查询 (select...from)
- 列别名 (as)
- 算术运算
- Where、limit、 distinct语句
- 比较运算 (between/in/is NULL 等)、逻辑运算 (and、or、 not)
- LIKE
- 常用聚合函数 (count、 max、 min、 sum、 avg)

2、分组

- Group By 语句
- Having语句

3、连接 (Join on语句)

内连接、左连接、有连接、全连接、多表连接

4、排序

order by 、sort by、 distribute by 、 定义reduce数量

总结

◆ SQL语句书写顺序和执行优先级

书写顺序：

select、form、where、group by、having、order by、limit

```
1 SELECT DISTINCT column, AGG_FUNC(column_or_expression), ...
2 FROM mytable
3     JOIN another_table
4         ON mytable.column = another_table.column
5 WHERE constraint_expression
6 GROUP BY column
7 HAVING constraint_expression
8 ORDER BY column ASC/DESC
9 LIMIT count OFFSET COUNT;
```

总结

◆ SQL语句书写顺序和执行优先级

SQL语句执行顺序

join、from、where、group by、having、select、order by、limit

```
1 SELECT DISTINCT column, AGG_FUNC(column_or_expression), ...  
2 FROM mytable  
3     JOIN another_table  
4     ON mytable.column = another_table.column  
5 WHERE constraint_expression  
6 GROUP BY column  
7 HAVING constraint_expression  
8 ORDER BY column ASC/DESC  
9 LIMIT count OFFSET COUNT;
```