**sql笔记整理总结---江瀚**

**DQL操作**

DQL数据查询语言

数据库执行DQL语句不会对数据进行改变，而是让数据库发送结果集给客户端。

查询返回的结果集是一张虚拟表。

查询关键字：SELECT

语法： SELECT 列名 FROM表名

【WHERE --> GROUP BY -->HAVING--> ORDER BY】

语法：

SELECT         selection\_list /\*要查询的列名称\*/

FROM            table\_list /\*要查询的表名称\*/

WHERE          condition /\*行条件\*/

GROUP BY     grouping\_columns /\*对结果分组\*/

HAVING          condition /\*分组后的行条件\*/

ORDER BY     sorting\_columns /\*对结果分组\*/

LIMIT             offset\_start, row\_count /\*结果限定\*/

**1　基础查询**

查询所有列

**SELECT \* FROM stu;**

查询指定列

**SELECT sid, sname, age FROM  stu;**

**2　条件查询**

2.1　条件查询介绍

条件查询就是在查询时给出WHERE子句，在WHERE子句中可以使用如下运算符及关键字：

=、!=、<>、<、<=、>、>=；

BETWEEN…AND；

IN(set)；

IS NULL； IS NOT NULL

AND；

OR；

NOT；

and   多个值查询

2.2　查询性别为女，并且年龄50的记录

**SELECT \* FROM stu**

**WHERE gender='female' AND ge<50;**

**OR**

2.3　查询学号为S\_1001，或者姓名为liSi的记录

**SELECT \* FROM stu**

**WHERE sid ='S\_1001' OR sname='liSi';**

**IN**

2.4　查询学号为S\_1001，S\_1002，S\_1003的记录

**SELECT \* FROM stu**

**WHERE sid IN ('S\_1001','S\_1002','S\_1003');**

**NOT IN**

2.5　查询学号不是S\_1001，S\_1002，S\_1003的记录

**SELECT \* FROM tab\_student**

**WHERE s\_number NOT IN ('S\_1001','S\_1002','S\_1003');**

**IS NULL**

2.6　查询年龄为null的记录

**SELECT \* FROM stu**

**WHERE age IS NULL;**

**BETWEEN   AND或者and**

2.7　查询年龄在20到40之间的学生记录

**SELECT \***

**FROM stu**

**WHERE age>=20 AND age<=40;**

或者

**SELECT \***

**FROM stu**

**WHERE age BETWEEN 20 AND 40;**

**不等于的写法：!=；<>；not...=...**

2.8　查询性别非男的学生记录

**SELECT \***

**FROM stu**

**WHERE gender!='male';**

或者

**SELECT \***

**FROM stu**

**WHERE gender<>'male';**

或者

**SELECT \***

**FROM stu**

**WHERE NOT gender='male';**

**IS NOT NUL**

2.9　查询姓名不为null的学生记录

**SELECT \***

**FROM stu**

**WHERE sname IS NOT NULL;**

或者

**SELECT \***

**FROM stu**

**WHERE NOT sname IS NULL;**

**2　模糊查询**

当想查询姓名中包含a字母的学生时就需要使用模糊查询了。模糊查询需要使用关键字LIKE。

通配符:

\_ 任意一个字符

 例如：'张\_'

%：任意0~n个字符

 例如：   %张%'

3.1　查询姓名由5个字母构成的学生记录

**SELECT \***

**FROM stu**

**WHERE sname LIKE '\_ \_ \_ \_ \_';**

模糊查询必须使用LIKE关键字。其中 “\_”匹配任意一个字母，5个“\_”表示5个任意字母。

3.2　查询姓名由5个字母构成，并且第5个字母为“i”的学生记录

**SELECT \***

**FROM stu**

**WHERE sname LIKE '\_ \_ \_ \_i';**

3.3　查询姓名以“z”开头的学生记录

**SELECT \***

**FROM stu**

**WHERE sname LIKE 'z%';**

其中“%”匹配0~n个任何字母。

3.4　查询姓名中第2个字母为“i”的学生记录

**SELECT \***

**FROM stu**

**WHERE sname LIKE '\_i%';**

3.5　查询姓名中包含“a”字母的学生记录

**SELECT \***

**FROM stu**

**WHERE sname LIKE '%a%';**

**4　字段控制查询**

4.1　去除重复记录

去除重复记录（两行或两行以上记录中系列的上的数据都相同），例如emp表中sal字段就存在相同的记录。当只查询emp表的sal字段时，那么会出现重复记录，那么想去除重复记录，需要使用DISTINCT：

DISTINCT：

**SELECT DISTINCT 列名 FROM 表名**

**SELECT DISTINCT sal FROM emp;**

4.2　查看雇员的月薪与佣金之和

　　因为sal和comm两列的类型都是数值类型，所以可以做加运算。如果sal或comm中有一个字段不是数值类型，那么会出错。

**SELECT \*,sal+comm FROM emp;**

**IFNULL：**

把NULL转换成数值0

comm列有很多记录的值为NULL，因为任何东西与NULL相加结果还是NULL，所以结算结果可能会出现NULL。下面使用了把NULL转换成数值0的函数IFNULL：

**SELECT \*,sal+IFNULL(comm,0) FROM emp;**

4.3　给列名添加别名

在上面查询中出现列名为sal+IFNULL(comm,0)，这很不美观，现在我们给这一列给出一个别名，为total：

起别名如果要加空格，用“”把别名引起来

**SELECT \*, sal+IFNULL(comm,0) AS total FROM emp;**

给列起别名时，是可以省略AS关键字的：

**SELECT \*,sal+IFNULL(comm,0)  total FROM emp;**

**5　排序  order by 列名 asc(默认) desc**

**ORDER BY 列名 ASC;  升序**

5.1　查询所有学生记录，按年龄升序排序

**SELECT \***

**FROM stu**

**ORDER BY sage ASC;**

或者

**SELECT \***

**FROM stu**

**ORDER BY sage（默认升序）;**

**ORDER BY 列名 DESC;  降序**

5.2　查询所有学生记录，按年龄降序排序

**SELECT \***

**FROM stu**

**ORDER BY age DESC;**

如果相同，再用编号排序

5.3　查询所有雇员，按月薪降序排序，如果月薪相同时，按编号升序排序

**SELECT \* FROM emp**

**ORDER BY sal DESC,empno ASC;**

**6　聚合函数  sum avg max min count**

聚合函数是用来做纵向运算的函数：

COUNT()：统计指定列不为NULL的记录行数；

MAX()：计算指定列的最大值，如果指定列是字符串类型，那么使用字符串排序运算；

MIN()：计算指定列的最小值，如果指定列是字符串类型，那么使用字符串排序运算；

SUM()：计算指定列的数值和，如果指定列类型不是数值类型，那么计算结果为0；

AVG()：计算指定列的平均值，如果指定列类型不是数值类型，那么计算结果为0；

6.1　COUNT

当需要纵向统计时可以使用COUNT()。

查询emp表中记录数，也就是表的行数：

**SELECT COUNT(\*) AS cnt FROM emp;**

查询emp表中有佣金的人数：

**SELECT COUNT(comm) cnt FROM emp;**

注意，因为count()函数中给出的是comm列，那么只统计comm列非NULL的行数。

查询emp表中月薪大于2500的人数：

**SELECT COUNT(\*) FROM emp**

**WHERE sal > 2500;**

统计月薪与佣金之和大于2500元的人数：

**SELECT COUNT(\*) AS cnt FROM emp WHERE sal+IFNULL(comm,0) > 2500;**

查询有佣金的人数，有领导的人数：

**SELECT COUNT(comm), COUNT(mgr) FROM emp;**

6.2　SUM和AVG

当需要纵向求和时使用sum()函数。

**SUM**

查询所有雇员月薪和：

**SELECT SUM(sal) FROM emp;**

**SUM两列**

查询所有雇员月薪和，以及所有雇员佣金和：

**SELECT SUM(sal), SUM(comm) FROM emp;**

查询所有雇员月薪+佣金和：

**SELECT SUM(sal+IFNULL(comm,0)) FROM emp;**

**AVG**

统计所有员工平均工资：

**SELECT AVG(sal) FROM emp;**

6.3　MAX和MIN

查询最高工资和最低工资：

**SELECT MAX(sal), MIN(sal) FROM emp;**

**7　分组查询**

 注：凡和聚合函数同时出现的列名，一定要写在group by 之后

例：查询每个部门的部门编号和每个部门的工资和：

**SELECT deptno, SUM(sal)**

**FROM emp**

**GROUP BY deptno;**

例：查询每个部门的部门编号以及每个部门的人数：

**SELECT deptno,COUNT(\*)**

**FROM emp**

**GROUP BY deptno;**

例：查询每个部门的部门编号以及每个部门工资大于1500的人数：

**SELECT deptno,COUNT(\*)**

**FROM emp**

**WHERE sal>1500**

**GROUP BY deptno;**

7.2　HAVING子句

类似where，筛选关键字

查询工资总和大于9000的部门编号以及工资和：

**SELECT deptno, SUM(sal)**

**FROM emp**

**GROUP BY deptno**

**HAVING SUM(sal) > 9000;**

注：having与where的区别:

1.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

2.having后面可以使用聚合函数(统计函数)

where后面不可以使用聚合函数。

WHERE是对分组前记录的条件，如果某行记录没有满足WHERE子句的条件，那么这行记录不会参加分组；而HAVING是对分组后数据的约束。

**8　LIMIT  分页**

LIMIT用来限定查询结果的起始行，以及总行数。

8.1　查询5行记录，起始行从0开始

**SELECT \* FROM emp LIMIT 0, 5;**

注意：起始行从0开始，即第一行开始！

8.2　查询10行记录，起始行从3开始

**SELECT \* FROM emp LIMIT 3, 10;**

8.3　分页查询

如果一页记录为10条，希望查看第3页记录应该怎么查呢？

第一页记录起始行为0，一共查询10行；

第二页记录起始行为10，一共查询10行；

第三页记录起始行为20，一共查询10行；

8.3　查询代码的书写顺序和执行顺序

       查询语句书写顺序：select – from- where- group by- having- order by-limit

       查询语句执行顺序：from - where -group by - having - select - order by-limit