[1.基础查询](#9848-1597288935694)

[1.1 查询表中的单个字段](#1381-1597289017178)

[1.2 查询表中的多个字段](#5373-1597289280093)

[1.3 查询表中的所有字段](#7785-1597289317787)

[1.4 查询常量值](#4171-1597290434224)

[1.5 查询表达式](#2970-1597291006305)

[1.6 查询函数](#9370-1597291027006)

[1.7 起别名](#7060-1597293099398)

[1.8 去重](#1913-1597293336431)

[1.8 +号的作用](#6336-1597293433226)

[2. 条件查询](#8713-1597296788084)

[2.1 按条件表达式筛选](#8350-1597297294703)

[2.2 按逻辑表达式筛选](#4672-1597297431976)

[2.3 模糊查询](#6620-1597297989754)

[2.3.1 like](#1218-1597298195585)

[2.3.2 between and](#8937-1597298568822)

[2.3.3 in关键字](#3086-1597298699630)

[2.3.4 is null 关键字](#4551-1597299187780)

[2.4 安全等于 <=>](#7523-1597299344374)

[3.排序查询](#8450-1597299533967)

[4.常见函数](#5200-1597462785440)

[4.1.单行函数](#8780-1597463120391)

[4.1.1.字符函数](#3764-1597463161115)

[4.1.2.数学函数](#9987-1597461054169)

[4.1.3 日期函数](#6440-1597472323743)

[4.1.4 其他函数](#4510-1597461054729)

[4.1.5 流程控制函数](#9259-1597473494429)

[4.2 分组函数](#5011-1597475438656)

[5.分组查询](#3179-1597477558748)

[6.连接查询](#6731-1597461054920)

[6.1 sql92标准](#5894-1597541085216)

[6.1.1 等值连接](#3465-1597541106560)

[6.1.2 非等值连接](#2776-1597461055327)

[6.1.3 自连接](#3583-1597461055690)

[6.2 sql99语法](#6257-1597544147304)

[6.2.1 内连接](#9666-1597551409459)

[6.2.2外连接](#2092-1597560909476)

[6.2.3 子查询](#2025-1597562955226)

[8.分页查询](#9699-1597581674559)

1.DQL data query language

2.DML data manipulation language

3.DDL data define language

4.TCL transaction control language

5.DCL data control language

**1.基础查询**

语法:

select 查询列表 from 表名；

类似于:System.out,printIn(打印东西)；

特点：

1.查询列表可以是：表中的字段、常量值、表达式、函数

2.查询的结果是一个虚拟的表格

**1.1 查询表中的单个字段**

select last\_name from employees;

**1.2 查询表中的多个字段**

select last\_name,salary,email from employees;

**1.3 查询表中的所有字段**

select \* from employees;

**1.4 查询常量值**

select 100;

select 'john';

**1.5 查询表达式**

select 100%98;

**1.6 查询函数**

select version();

**1.7 起别名**

select 100%98 as 别名

select last\_name as 姓, first\_name as 名 from employees;

select last\_name 姓, first\_name 名 from employees;

用处：1.便于理解；2.如果要查询的字段有重名的情况，使用别名可以区分开来。

eg.查询salary,显示结果为out put

select salary as out put from employees;#out被识别为关键字，出现错误

select salary as "out put" from employees;

**1.8 去重**

eg.查询员工表中涉及到的所有部分编号

select department\_id from employees;//会有重复信息

select distinct department\_id from employees;

**1.8 +号的作用**

只能作为运算符使用。

如果有字符型，尝试转换成数值型，再进行运算（“1”->1）;如果转换失败，则转换成0，再进行运行。

只要其中一方为null，结果肯定为null

eg.查询员工名和员工的姓，连接成一个字段，并显示为 姓名

select

concat(last\_name,first\_name) as 姓名

from

employees;

eg.查询出表employees的全部列，各个列之间用逗号连接，列头显示成out\_put

select

concat('first\_name', ',' , 'last\_name', ',' ,'job\_id', ',' ,IFNULL(commission\_pct,0)) as

out\_put

from

employees;

**2. 条件查询**

语法：

select

查询列表

from

表名

where

筛选条件;

分类：

1.按条件表达式筛选

条件运算符 > < = != <> >= <=

2.按逻辑表达式筛选

逻辑运算符 && || !

and or not

3.模糊查询

like

between and

in

is null

**2.1 按条件表达式筛选**

eg. 查询工资>12000的员工信息

SELECT

\*

FROM

employees

WHERE

salary>12000;

eg. 查询部门编号不等于90号的员工名和部门编号

SELECT

last\_name,

department\_id

FROM

employees

WHERE

department\_id!=90;//不标准，最好是department\_id<>90;

**2.2 按逻辑表达式筛选**

eg.查询工资z在10000到20000之间的员工名，工资和奖金

SELECT

last\_name,

salary,

commission\_pct

FROM

employees

WHERE

salary>=10000 AND salary<=20000;

eg.查询部门编号不是在90到110之间，或者工资高于15000的员工信息

SELECT

\*

FROM

employees

WHERE

department\_id<90 OR department\_id>110 OR salary>15000;

**2.3 模糊查询**

**2.3.1 like**

特点：

- 一般和通配符搭配使用

% 任意多个字符，包含0个字符

\_ 任意单个字符

eg. 查询员工名共包含字符a的员工信息

SELECT

\*

FROM

employees

WHERE

last\_name LIKE '%a%';//默认大小写a是一回事

eg.查询员工名中第三个字符为e，第五个字符为a的员工名和工资

SELECT

last\_name

FROM

employees

WHERE

last\_name LIKE '\_\_\_e\_a%';

eg. 查询员工名中第二个字符为下划线的员工名

SELECT

last\_name

FROM

employees

WHERE

last\_name LIKE '\_\\_%';

SELECT

last\_name

FROM

employees

WHERE

last\_name LIKE '\_$\_%' ESCAPE '$';

**2.3.2 between and**

注意：

- 使用 between and 可以提高语句的简洁度

- 包含临界值

- 两个临界值不要调换顺序

eg. 查询员工编号在100到120之间的员工信息

SELECT

\*

FROM

employees

WHERE

employee\_id BETWEEN 100 AND 120;

**2.3.3 in关键字**

含义：判断某字符的值是否属于in列表中的某一项

特点：

- 使用in提高语句简洁度

- in列表的值必须是同样的类型，或者可以兼容的类型，即可以隐式转换

eg. 查询员工的公众编号是 IT\_PROG、AD\_VP、AD\_PRES中的一个的员工名和工种编号

SELECT

last\_name,

job\_id

FROM

employees

WHERE

job\_id IN ('IT\_PROG','AD\_VP','AD\_PRES');

**2.3.4 is null 关键字**

= 或 <> 不能用于判断null值

is null 或 is not null 可以判断null值

eg. 查询没有奖金的员工名和奖金率

SELECT

last\_name,

commission\_pct

FROM

employees

WHERE

commission\_pct IS NULL;

**2.4 安全等于 <=>**

特点：可读性差

eg. 查询没有奖金的员工名和奖金率

SELECT

last\_name,

commission\_pct

FROM

employees

WHERE

commission\_pct <=> NULL;

eg. 查询工资为12000的员工信息

SELECT

\*

FROM

employees

WHERE

salary <=> 12000;

Tips：

IS NULL：仅仅可以判断NULL值，可读性较高，建议使用

<=>: 既可以判断NULL，又可以判断普通的数值， 可读性较低

**3.排序查询**

语法：

SELECT 查询列表

FROM 表

(WHERE 筛选条件）

ORDER BY 排序列表

特点：

1.ASC 代表升序，DESC代表降序，不写默认升序

2.ORDER BY 子句中可以支持单个字段、多个字段、表达式、函数、别名

3.ORDER BY 子句一般是放在查询语句的最后面，LIMIT子句除外

eg.查询员工信息，要求工资从高到低排序

SELECT \* FROM employees ORDER BY salary DESC;(降序排序)

SELECT \* FROM employees ORDER BY salary ASC;(升序排序)

SELECT \* FROM employees ORDER BY salary ;(默认为升序排序)

eg.查询部门编号>=90的员工信息，要求按入职时间排序

SELECT

\*

FROM

employees

WHERE

department\_id >= 90

ORDER BY

hiredate ASC;

eg.按年薪的高低显示员工信息和年薪【按别名排序】

SELECT

\*,salary\*12\*(1+IFNULL(commission\_pct,0)) 年薪

FROM

employees

ORDER BY

salary\*12\*(1+IFNULL(commission\_pct,0)) DESC;

eg.按姓名的长度显示员工的姓名和工资【按函数排序】

SELECT

LENGTH(last\_name) 字节长度，last\_name,salary

FROM

employees

ORDER BY

LENGTH(last\_name) DESC;

eg.查询员工信息，要求先按工资排序，再按员工编号排序【按多个字段排序】

SELECT

\*

FROM

employees

ORDER BY

salary ASC，employee\_id DESC;

**4.常见函数**

概念：类似于java的方法，将一组逻辑语句封装在方法体中，对外暴露方法名

好处：1.隐藏了实现细节；2.提高了代表的重用性；

调用：SELECT 函数名(实参列表) (from 表)

特点：1.叫什么(函数名)；2.干什么(函数功能)

分类：

1.单行函数；

功能：做处理使用

如 concat、length、ifnull等

又分为 字符函数、数学函数、日期函数、其他函数、流程控制函数

2.分组函数

功能：做统计使用、又称为统计函数、聚合函数、组函数

**4.1.单行函数**

**4.1.1.字符函数**

1.length

select length(‘john’); #4

select length('张三丰hahaha'); #15 uft8 一个中文字占3个字节，一个英文字占1个字节

2.concat

select concat(first\_name,'\_',last\_name) from employees;

3.upper lower

eg. 姓变大写，名变小写，然后拼接

select concat(upper(last\_name),lower(first\_name)) 姓名 from employees;

4.substr、substring

注意：索引从1开始

select substr('李莫愁爱上了陆展元',7) out\_put; #李莫愁爱上了

select substr('李莫愁爱上了陆展元',1,3) out\_put; #李莫愁

eg.姓名中首字符大写，其他字符小写然后用\_拼接，显示出来

select concat(upper(substr(last\_name,1,1)),'\_',lower(substr(last\_name,2))) out\_put from employees;

5.instr 返回子串第一次出现的索引，如果找不到返回0

select instr('杨不悔爱上了殷六侠','殷六侠') as out\_put; #7

6.trim 去掉前后的空格

select length(trim(' 张翠山 ')) as out\_put; #9

select length(trim('a' from 'aaaaaa张aa翠山aaaaa')) as out\_put;#张aa翠山 11

7.lpad 用指定的字符实现左填充指定长度

select lpad('殷素素',10,'\*') as out\_put;#\*\*\*\*\*\*\*殷素素

8.rpad 用指定的字符实现右填充指定长度

select rpad('殷素素',10,'\*') as out\_put;#殷素素\*\*\*\*\*\*\*

9.replace 替换

select replace('张无忌爱上了周芷若','周芷若','赵敏') as out\_put;

**4.1.2.数学函数**

1.round四舍五入

select round(1.65);#2

select round(-1.65);#-2

select round(-1.45);#-1

select round(1.567,2);#1.57

2.ceil 向上取整

select ceil(1.002);#2

select ceil(1.00);#1

3.floor 向下取整

select ceil(1.002);#1

select ceil(0.99);#0

select ceil(9.99);#9

select ceil(-9.99);#-10

4.truncate 截断

select truncate(1.65,1);#1.6

5.mod 取余 取模 mod(a,b) a - (a/b\*b)

select mod(10,3); #1

select 10%3;#1

select mod(-10,-3);#-1 被除数为正，结果为正，被除数为负，结果为负

select mod(-10,3);#-1

select mod(10,-3);#1

select mod(10,3);#1

**4.1.3 日期函数**

1.now 返回当前系统日期+时间

select now();

2.curdate 返回当前系统日期，不包含时间

select curdate();

3.curtime 返回当前系统时间，不包含日期

select curtime();

4.可以获取指定的部分、年、月、日、小时、分钟、秒

select year(now()) 年;

select year(now('1998-1-1')) 年;

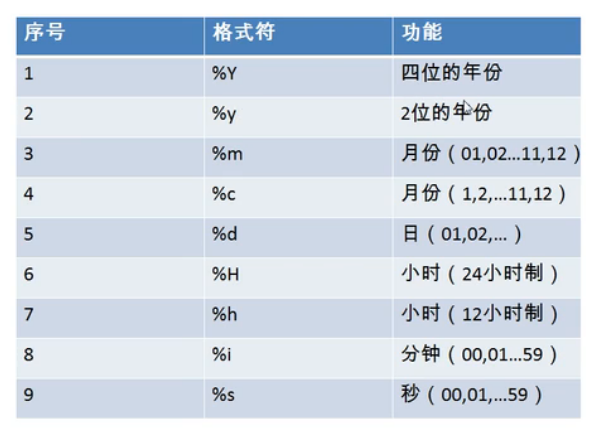
select year(hiredate) 年 from employees;

select month(now()) 月; #8

select monthname(now()) 月; #August

5.str\_to\_date 将日期格式的字符转换成指定格式的日期

select str\_to\_date(1998-3-2','%Y-%c-%d') as out\_put; #1990-03-02



eg.查询入职日期为1992-4-3的员工信息

select \* from employees where hiredate = str\_to\_date('1992-4-3','%c-%d %Y');

6.date\_format 将日期转换成字符

select date\_format('2018/6/6','%Y年%m月%d日‘) 2018年06月06日

eg.查询有奖金的员工名和入职日期(\*\*月/\*\*日 \*\*年)

select last\_name,date\_format(hiredate,'%m月/%d日 %y年') 入职日期 from employees where commission\_pct is not null;

**4.1.4 其他函数**

select version();#查看版本

select database();#查看当前数据库

select user();#查看当前用户

**4.1.5 流程控制函数**

1.if函数：if else的效果

select if(10>5,'大','小'); #大

select last\_name,commission\_pct,if(commision\_pct is null,'没奖金，哈哈','有奖金，嘻嘻') 备注 from employees;

2.case函数

(1) switch case 的效果

语法：

case 要判断的字段或表达式

when 常量1 then 要显示的值或语句

when 常量2 then 要显示的值2或语句2

···

else 要显示的值n或语句n

end

eg.查询员工的工资，要求部门号为30，显示的工资为1.1倍；部门号为40，显示的工资为1.2倍；部门号为50，显示的工资为1.3倍；其他部分，显示的工资为原工资。

select salary 原始工资,department\_id,

case department\_id

when 30 then salary\*1.1

when 40 then salary\*1.2

when 50 then salary\*1.3

else salary

end as 新工资

from employees;

(2)类似于多重if

语法

case

when 条件1 then 要显示的值或语句

when 条件2 then 要显示的值2或语句2

···

else 要显示的值n或语句n

end

eg.查询员工的工资的情况，如果工资>20000,显示A级别；如果工资>15000,显示B级别；如果工作>10000,显示C级别；否则，显示D级别。

select salary,

case

when salary>20000 then ’A'

when salary>15000 then 'B'

when salary>10000 the 'C'

else 'D'

end as 工资级别

from employees;

**4.2 分组函数**

sum 求和 avg 平均值 max 最大值 min 最小值 count 计算个数

1.简单使用

select sum(salary) from employees;

select round(avg(salary),2) from employees;

select min(salary),max(salary) from employees;

2.参数支持哪些类型

sum,avg字符型，但没有意义

select max(last\_name),min(last\_name) from employees;#支持，也有意义

select max(hiredate),min(hiredate) from employees;#支持，也有意义

count 都支持，也有意义，会忽略null值

3.是否忽略null值

会忽略null值:sum、avg、max、min、count

select avg(commisssion\_pct),sum(commisssion\_pct)/35,sum(commisssion\_pct)/127 from employees;

4.和distinct搭配(去重)

select sum(distinct salary) from employees;

5.count函数介绍

select count(\*) from employees; #统计行数，效率一般更高

select count(1) from employees; #统计行数 写1 写2都行 写啥都一样，除了null

6.和分组函数一同查询的字段有限制，要求是group by 之后的字段

select avg(salary),employee\_id from employees;#没有任何意义

**5.分组查询**

语法：

select 分组函数、列（要求出现在group by的后面）

from 表

【where 筛选条件】

group by 分组的列表

【order by 子句】

注意：查询列表必须特殊，要求是分组函数和group by后出现的字段

特点：

1.分组查询中的筛选条件分为

数据源 位置

分组前筛选 原始表 where

分组后筛选 分组后的结果集 having

tips:分组函数做条件肯定是在having语句之中；能用分组前筛选的，优先考虑使用分组前筛选

2.group by子句支持单个字段分组、多个字段分组（没有顺序要求）、表达式分组、函数分组

3.也可以添加排序（排序放在整个分组查询的最后）

1.简单分组查询

eg.查询每个部门的平均工资

select avg(salary) from employees;

eg.查询每个工种的最高工资

select max(salary),job\_id

from employees

group by job\_id;

eg.查询每个位置上的部门个数

select

count(\*),location\_id

from

departments

group by

location\_id;

2.添加筛选条件

eg.查询邮箱中包含a字符的每个部门的平均工资

select avg(salary),department\_id

from employees

where email like '%a%'

group by department\_id

eg.查询有奖金的每个部门的每个领导手下员工的最高工资

select

max(salary),manager\_id,

from

employees

where commission\_pct is not null

group by manager\_id;

3.添加复杂筛选条件

eg.查询哪个部门的员工个数>2

tips:1.查询每个部门的员工个数；2.根据1的结果进行筛选，查询哪个部门的员工个数>2

select

count(\*),department\_id

from

employees

group by

department\_id

having

count(\*)>2;

eg.查询每个工种有奖金的员工的最高工资>12000的工种编号和最高工资

select

job\_id,max(salary)

from

employees

where

commission\_pct is not null

group by

job\_id

having

max(salary)>12000;

eg.查询领导编号>102的每个领导手下的最低工资>5000的领导编号是哪个，以及其最低工资

select

min(salary),manager\_id

from

employees

where

manager\_id>192

group by

manager\_id

having

min(salary)>5000

4.按表达式或函数分组

eg.按员工姓名的长度分组，查询每一组的员工个数，筛选员工个数>5的有哪些

select

count(\*),length(last\_name)

from

employees

group by

length(last\_name)

having

count(\*)>5;

5.按多个字段分组

eg.查询每个部门每个工种的员工的平均工资

select

avg(salary),department\_id,job\_id

from

employees

group by

job\_id,department\_id;

6.添加排序

eg.查询每个部门每个工种的员工的平均工资，并且按平均工资的高低显示

select

avg(salary),department\_id,job\_id

from

employees

where department\_id is not null

group by

job\_id,department\_id

order by

avg(salary) desc;

**6.连接查询**

按年代分类：

sql92标准：仅仅支持内连接

sql99标准：【推荐】支持内连接+外连接+交叉连接

按功能分类：

内连接：等值连接、非等值连接、自连接

外连接：左外连接、右外连接、全外连接

交叉连接：

**6.1 sql92标准**

**6.1.1 等值连接**

(1)多表等值连接的结果为多表的交集部分

(2)n表连接，至少需要n-1个连接条件

(3)多表的顺序没有要求

(4)一般需要为表起别名

(5)可以搭配前面介绍的所有子句使用，比如排序、分组、筛选

eg.查询女神名和对应的男神名

select name,boyname

from bos,beauty

where beauty.boyfriend\_id=boys.id;

eg.查询员工名和对应的部门名

select last\_name,department\_name

from employees,departments

where employees.department\_id=departments.department\_id;

6.1.2 为表起别名：1.提高语句的简洁度；2.区分多个重名的字段；3.如果为表起了别名，则查询的字段不能使用原来的表名去限定

eg.查询员工名、工种号、工种名

select last\_name,e.job\_id,title

from employees as e,jobs

where e.job\_id=jobs.job\_id

3.表名顺序可以替换

4.可以加筛选

eg.查询有奖金的员工名、部门名

select last\_name,department\_name

from employees e,departments d

where e.department\_id=d.department\_id

and e.commission\_pct is not null

eg.查询城市名中第二个字符为o的部门名和城市名

select department\_name,city

from departments d,location l

where d.location\_id = l.location\_id

and city like '\_o%';

5.可以加分组

eg.查询每个城市的部门个数

select count(\*) 个数,city

from departments d,locations l

where d.location\_id=l.location\_id

group by city;

eg.查询有奖金的每个部门的部门名和部门的领导编号和该部门的最低工资

select department\_name,manager\_id,min(salary)

from departments d,employees e

where d.department\_id=e.department\_id

and commission\_pct is not null

group by department\_name,d.manager\_id

6.可以加排序

eg.查询每个工种的工种名和员工个数，并且按员工个数降序

select job\_title,count(\*)

from employees e,jobs j

where e.job\_id=j.job\_id

group by job\_title

order by count(\*) desc;

7.可以实现三表连接

eg.查询员工名、部门名、所在的城市

select last\_name,department\_name,city

from employees e,departments d,location l

where e.department\_id=d.department\_id

and d.location\_id=l.location\_id;

**6.1.2 非等值连接**

eg.查询员工的工资和工资级别

select salary,grade\_level

from employees e,job\_grades g

where salary between g.lowest\_sal and g.highest\_sal;

**6.1.3 自连接**

eg.查询员工名和上级的名称

select employees\_id,last\_name,employee\_id,last\_name

from employees e,employees m

where e.manager\_id=m.employee\_id;

**6.2 sql99语法**

语法：

select 查询列表

from 表1 别名 【连接类型】

join 表2 别名

on 连接条件

【where 筛选条件】

【group by 分组】

【order by 排序列表】

分组

内连接：inner

外连接

左外：left【outer】

右外：right【outer】

全外：full【outer】

交叉连接：cross

**6.2.1 内连接**

语法：

select 查询列表

from 表1，别名

inner join 表2，别名

on 连接条件

特点：

(1)添加排序、分组、筛选

(2)inner可以省略

(3)筛选条件放在where后面，连接条件放在on后面，提高分离性，便于阅读

(4)inner join连接和sql92语法中的等值连接效果是一样的，都是查询多表的交集

(5)

(6)

(7)

1.等值连接

eg.查询员工名、部门名

select

last\_name,department\_name

from

employee e

inner join

departments d

on

e.department\_id=d.department\_id;

eg.查询名字中包含e的员工名和工种名（添加筛选）

select

last\_name,job\_title

from

employee e

inner join

jobs j

on

e.job\_id=j.job\_id

where

e.last\_name like "%e%";

eg.查询部门个数>3的城市名和部门个数（添加分组+筛选）

select city,count(\*) 部门个数

from departments d

inner join locations l

on d.location\_id=l.location\_id

group by city

having count(\*)>3;

eg.查询哪个部门的部门员工个数>3的部门名和员工个数，并按个数排序

select

count(\*),department\_name

from

employees e

inner join

departments d

on

e.department\_id=d.department\_id

group by

department\_name

having

count(\*)>3

order by

count(\*);

eg.查询员工名，部门名，工种名，并按部门名降序

select

last\_name,department\_name,job\_title

from

employees e

inner join

departments d

on

e.department\_id = d.department\_id

inner join

jobs j

on

e.job\_id = j.job\_id

order by

department\_name desc;

2.非等值连接

eg.查询员工的工资级别

select salary,grade\_level

from employees e

join job\_grades g

on e.salary between g.lowest\_sal and g.highest\_sal

eg.查询工资级别的个数>20的个数，并且按工资级别降序

select count(\*),grade\_level

from employees e

join job\_grades g

on e.salary between g.lowest\_sal and g.highest\_sal

group by grade\_level

having count(\*)>20

order by grade\_level desc;

3.自连接

eg.查询员工的名字、上级的名字

select e.last\_name,m.last\_name

from employees e

join employees m

on e.manager\_id = m.employee\_id;

eg.查询姓名中包含字符K的员工的名字、上级的名字

select e.last\_name,m.last\_name

from employees e

join employees m

on e.manager\_id = m.employee\_id;

where e.last\_name like '%k%';

**6.2.2 外连接**

应用场景：用于查询一个表中有，另一个表没有的记录

特点：

1.外连接的查询结果为主表中的所有记录

如果从表中有和它匹配的，则显示匹配的值

如果从表中没有和它匹配的，则显示null

外连接查询结果=内连接结果+主表中有而从表没有的记录

2.左外连接，left join 左边的是主表

右外连接，right join 右边的是主表

3.左外和右外交换两个表的顺序，可以实现同样的效果

4.全外连接=内连接的结果+表1中有但表2没有的+表2中有单表1没有的

eg.查询男朋友不在男神表的女神名

#右外连接

select b.name,bo.\*

from beauty b

left outer join boys bo

on b.boyfriend\_id = bo.id

where bo.id is null;

#左外连接

select b.name,bo.\*

from boys bo

right outer join beauty b

on b.boyfriend\_id = bo.id

where bo.id is null;

eg.查询哪个部门没有员工

#左外连接

select d.\*,e.employee\_id

from departments d

left outer join employees e

on d.department\_id = e.department\_id

where e.employee\_id is null

#右外连接

select d.\*,e.employee\_id

from employees e

right outer join departments d

on d.department\_id = e.department\_id

where e.employee\_id is null;

#全外连接

select b.\*,bo.\*

from beauty b

full outer join boys bo

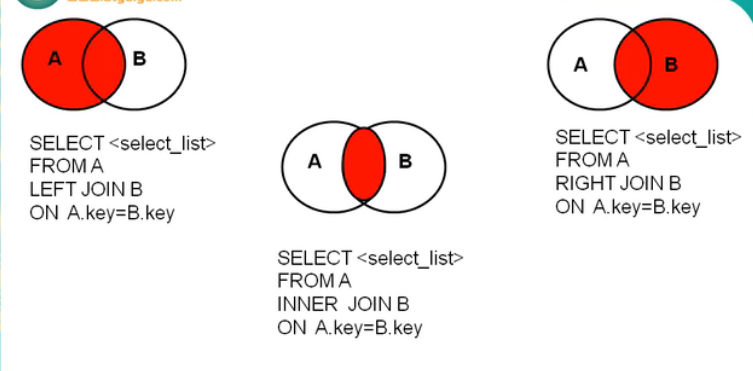
on b.boyfriend\_id = bo.id;

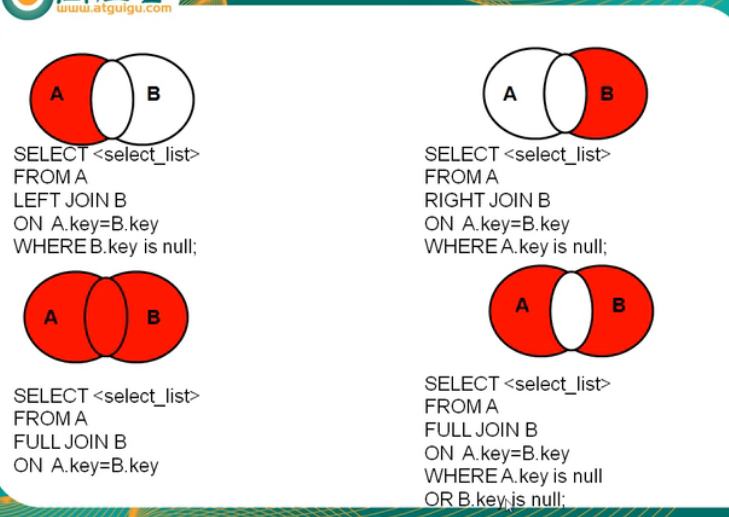
#交叉连接（笛卡尔乘积）

select b.\*,bo.\*

from beauty b

cross join boys bo;





**6.2.3 子查询**

含义：出现在其他语句中的select语句，称为子查询或内查询；内部嵌套其他select语句的查询，称为外查询或主查询

分类：

按子查询出现的位置：

select 后面：仅仅支持标量子查询

from 后面：支持表子查询

where或having后面：标量子查询（单行）、列子查询（多行）、行子查询

exists后面（相关子查询）

按结果集的行列数不同：

标量子查询（结果集只有一行一列）

列子查询（结果集只有一列多行）

行子查询（结果集有一行多列）

表子查询（结果集一般为多行多列）

一、where或having后面

1.标量子查询（单行子查询）

2.列子查询（多行子查询）

3.行子查询（多列多行）

特点：

1.子查询放在小括号内

2.子查询一般放在条件的右侧

3.标量子查询，一般搭配着单行操作符使用

> < >= <= = <>

列子查询，一般搭配着多行操作符使用

IN、ANY/SOME、ALL

1.标量子查询

eg.谁的工资比Abel高？

select \*

from employees

where salary > (

select salary

from employees

where last\_name = 'Abel'

);

eg.返回job\_id与141号员工相同，salary比143号员工多的员工 姓名，job\_id和工资

select

last\_name,job\_id,salary

from employees

where job\_id=(

select job\_id

from employees

where employee\_id = 141

) and salary>(

select salary

from employees

where employee\_id = 143

);

eg.返回公司工资最少的员工的last\_name,job\_id和salary

select

last\_name,job\_id,salary

from

employees

where

salary=(

select min(salary)

from employees

);

eg.查询最低工资大于50号部门最低工资的部门id和其最低工资

select min(salary),department\_id

from employees

group by department\_id

having min(salary)>(

select min(salary)

from employees

where department\_id = 50

)

#非法使用标量子查询

select min(salary),department\_id

from employees

group by department\_id

having min(salary)>(

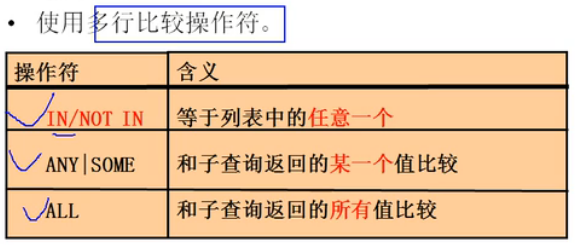
select salary #得到的就不是一个值了

from employees

where department\_id = 50

)

2.列子查询（多行子查询）



eg.返回location\_id是1400或1700的部门中所有员工姓名

select last\_name

from employees

where department\_id in(

select distinct department\_id

from departments

where location\_id in (1400,1700)

);

eg.返回其他部门中比job\_id为‘IT\_PROG’部门任一工资低的员工的员工号、姓名、job\_id以及salary

select

employee\_id,last\_name,job\_id,salary

from

employees

where

salary< any (

select distinct salary

from employees

where job\_id = 'IT\_PROG'

) and job\_id <>'IT\_PROG';

eg.返回其它部门中比job\_id为'IT\_PROG'部门所有工资都低的员工的员工号、姓名、job\_id以及salary

select

employee\_id,last\_name,job\_id,salary

from

employees

where

salary< all (

select distinct salary

from employees

where job\_id = 'IT\_PROG'

) and job\_id <>'IT\_PROG';

3.行子查询（结果集一行多列或多行多列）

eg.查询员工编号最小并且工资最高的员工信息

select \*

from employees

where employee\_id = (

select min(employee\_id)

from employees

) and salary = (

select max(salary)

from employees

);

select \*

from employees

where(employee\_id,salary)=(

select min(employee\_id),max(salary)

from employees

);

二、select后面

#只支持标量子查询

eg.查询每个部门的员工个数

select d.\*,(

select count(\*)

from employees e

where e.department\_id=d.department\_id

) 个数

from departments d;

eg.查询员工号=102的部门名

select (

select department\_name

from departments d

inner join employees e

on d.department\_id=e.department\_id

where e.employee\_id=102

) 部门名;

三、from后面的子查询

eg.查询每个部门的平均工资的工资等级

select ag\_dep.\*,g.grade\_level

from (

select avg(salary) ag,department\_id

from employees

group by department\_id

) ag\_dep

inner join job\_grades g

on ag\_dep.ag between lowest\_sal and highest\_sal;

四、exists后面（相关子查询）

/\*

语法：

exists（完整的查询语句）

结果：

1或0

\*/

select exists(select employee\_id from employees where salary=300000);

eg.查询有员工名的部门名

select department\_name

from departments

where exists(

select \*

from employees e

where d.department\_id = e.department\_id

);

select department\_name

from departments d

where d.department\_id in(

select department\_id

from employees

);

eg.查询没有女朋友的男神信息

select bo.\*

from boys bo

where bo.id not in (

select boyfriend\_id

from beauty

);

select bo.\*

from boys bo

where not exists(

select boyfriend\_id

from beauty b

where bo.id=b.boyfriend\_id

);

**7.分页查询**

应用场景：当要显示的数据、一页显示不全、需要分页提交sql请求

语法：

select 查询列表

from 表

【join type join 表2

on 连接条件

where 筛选条件

group by 分组字段

having 分组后的筛选

order by 排序的字段】

limit 【offset,】size;

#offset 要显示条目的起始索引（从0开始）

#size要显示的条目个数

特点：

1.limit语句放在查询语句的最后

2.公式：要显示的页数 page，每页的条目数 size

select 查询列表

from 表

limit (page-1)\*sizesize;

eg.查询前五条员工信息

select \* from employees limit 0,4;

select \* from employees limit 5;

eg.查询第11条-第25条

select \* from employees limit 10,15;

eg.有奖金的员工信息，并且工资较高的前10名显示出来

select \* from employees where commission\_pct is not null order by salary desc limit 10;

**8.联合查询**

union：将多条查询语句的结果合并成一个结果

应用场景：要查询的结果来自多个表，且多个表没有直接的连接关系，但查询的信息一致时

特点：两个表的列数必须一样；多条查询语句的每一列的类型和顺序最好一致；使用union时会自动去重，使用union all 会包含重复项

eg.查询部门编号>90或邮箱包含a的员工信息

select \* from employees where email like '%a%' or department\_id > 90;

select \* from employees where email like '%a%'

union

select \* from employees where department\_id > 90;

eg.查询中国用户中男性的信息>以及外国用户中男性的信息

select id,cname,csex from t\_ca where csex='男'

union

select t\_id,tName,tGender from t\_ua where tGender='male'