查询模块

Distinct关键字去重

Select \* from tableName;

在查询中过滤行：

赋值符号 :=

等于 =

Select \* from tableName where empoly\_id = 5;

不等于 != <>

Between …and… 在两个值之间（包含边界）这个和>= <=的效果是一样的，不包含边界的话用> < 去实现。

Select \* from tableName where empoly\_salary Between 4000 and 7000;

Select \* from tableName where empoly\_salary not Between 4000 and 7000; <4000 >7000

In(set) 等于值列表中的一个 这个和多个or的效果是一样的

Select \* from tableName where empoly\_id In(2,3,4,5);

Select \* from tableName where empoly\_id=2 or empoly\_id=3 or empoly\_id=4 or empoly\_id=5;

Like模糊查询

Select \* from tableName where empoly\_id like ‘吴%’;

Select \* from tableName where empoly\_id like ‘%a%’;

Select \* from tableName where last\_name like ‘%a%e%’ or last\_name like ‘%e%a%’;

员工中第二位是字符a的员工有哪些？\_表示第一位

Select \* from tableName where empoly\_id like ‘\_a%’;

员工中第三位是字符a的员工有哪些？\_\_表示占位，一个\_占一位

Select \* from tableName where empoly\_id like ‘\_\_a%’;

员工名字中含有\_的员工 通过escape转义进行处理

Select \* from tableName where empoly\_id like ‘\\_%’ escape ‘\’;

Is null 空值

Select \* from tableName where empoly\_salary Is null;

Select \* from tableName where empoly\_salary Is not null;

日期查询：查询1998-4-24来公司的员工有哪些？

注意：日期必须放到单引号中，且必须是指定的格式

Select \* from tableName where hire\_date =’24-4月-1998’

也可以用转化函数进行查询

Select \* from tableName where to\_char(hire\_date,’yyyy-MM-dd’) > ‘1998-04-24’;

Select \* from tableName where to\_char(hire\_date,’yyyy-MM-dd’) between ‘1998-05-01’ and ‘1998-09-01’;

在查询中进行排序：

Order by

Asc 升序（默认）

Dsc 降序

多层排序：

Select \* from tableName order by salary desc ,last\_name asc;

先按照第一层salary进行排序，当第一层有重复的数据时，按照第二层last\_name asc的规则进行排序；

单行函数

目标：

SQL中不同类型的函数

在select 语句中使用字符，数字，日期和转换函数

使用条件表达式

字符函数

字符函数：大小写控制函数，字符控制函数

大小写控制函数：

lower(‘ATGUIGU java’) 🡪 atguigu java

upper(‘ATGUIGU java’) 🡪 ATGUIGU JAVA

initcap(‘ATGUIGU java’) 🡪 Atguigu Java

CONCAT(‘hello’,’world’) 🡪helloworld 连接函数

SUBSTR(‘HelloWorld’,1,5) 🡪Hello 截取函数 sql中脚标是从1开始的，该例子中1表示起始位置，5表示输出的个数;

LENGTH(‘HelloWorld’) 🡪10 长度函数

INSTR(‘HelloWorld’,’W’) 🡪6 所在字符所在的位置

字符控制函数:

LPAD(salary,10,’\*’) 🡪\*\*\*\*\*24000 右对齐，输出的salary占用10位，不足的部分用\*填充；

RPAD(salary,10,’\*’) 🡪24000\*\*\*\*\* 左对齐，输出的salary占用10位，不足的部分用\*填充；

TRIM(‘H’ FROM ‘HelloWorld’) 🡪elloWorld 去除首尾的H

REPLACE(‘abcd’,’b’,’m’)🡪amcd 将所有的b替换成m

应用场景：

Select \* from employees where lower(last\_name)=’king’;

数字函数

ROUND:四舍五入

Round（45.926,2） -->45.93

TRUNC:截取

trunc（45.926,2） -->45.92

MOD:取余

Mod(1600,300) -->100

日期函数

函数SYSDATE 返回：日期和时间

日期的数学运算：

在日期上加上或者减去一个数字结果仍为日期。

两个日期相减返回日期之间相差的天数。【日期不允许做加法运算，无意义】。

可以用数字除以24来向日期中加上或者减去相应的天数。

MONTHS\_BETWEEN -->两个日期相差的月数

Select id ,months\_between(sysdate,hire\_date) from employees;

ADD\_MONTHS -->向指定日期中加上若干月数

Select add\_months(sysdate,2),add\_months(sysdate,-3),next\_day(sysdate,’星期日’) from employees;

NEXT\_DAY -->指定日期的下一个星期“对应的日期”

LAST\_DAY -->本月的最后一天

来公司的员工中，hire\_date是每月倒数第二天来公司的有哪些？

Select id from employees where hire\_date =last\_day(hire\_date) -1;

ROUND -->日期四舍五入

可以按照月份是否过半，小时是否过半进行运算

TRUNC -->日期截断

转换函数

数据类型转换：隐式和显式

隐式数据类型转换：

Oracle自动完成下列转换

|  |  |
| --- | --- |
| 源数据类型 | 目标数据类型 |
| VARCHAR2 or CHAR | NUMBER |
| VARCHAR2 or CHAR | DATE |
| NUMBER | VARCHAR2 |
| DATE | VARCHAR2 |

在数据库sql中字符串可以自动转化为number类型的，进行加减运算。

显式数据类型转换：

Date-->char

Select \* from table where to\_char(hire\_time,’yyyy-MM-dd’)=’1990-09-12’;

Select to\_char(sysdate,’yyyy”年”mm”月”dd”日” hh:mi:ss’) from dual;

char-->Date

Select \* from table where to\_date(’1990-09-12’ , ’yyyy-mm-dd’)=hire\_time;

NUMBER-->char

to\_char(number,’format\_model’);

下面是在TO\_CHAR函数中经常使用的几种格式：

|  |  |
| --- | --- |
| 9 | 数字 |
| 0 | 零 |
| $ | 美元符 |
| L | 本地货币符号 |
| . | 小数点 |
| ， | 千位符 |

Select to\_char(1234567.89,’999,999,999.99’) from dual;

CHAR-->NUMBER

Select to\_number(‘￥1,234,567.89’ , ’L999,999,999.99’) from dual;

通用函数：

这些函数适用于任何数据类型，同时也适用于空值；

**NVL(expr1,expr2)：**将空值转换成一个已知的值 ，可以使用的数据类型有日期、字符、数字。

当expr1不为空的时候NVL(expr1,expr2)的值为expr1本身，为空的时候值为expr2

例子：

1.求公司员工的年薪

Select id,last\_name,salary\*12(1+nvl(commission\_pct,0)) “别名” from table ;

1. 输出last\_name,department\_id,当department\_id为null时，显示‘没有部门’；

Select last\_name,nvl(to\_char(department\_id,’9999999’),’没有部门’) from table;

**NVL2(expr1,expr2,expr3) :**当expr1不为空的时候NVL2(expr1,expr2,expr3)的值为expr2，为空的时候值为expr3;

例子：

略

NULLIF(expr1,expr2)：相等返回null，不相等返回expr1;

COALESCE(expr1,expr2,.....expr(n-1),exprn);

CASE的使用

查询部门号为10,20,30的员工信息，若部门号为10，则打印其工资的1.1倍，20部门，则打印其工资的1.2倍，30号部门，则打印其工资的1.3倍

Select department\_id，last\_name ，case department\_id

when 10 then salary\*1.1

when 20 then salary\*1.2

else 30 then salary\*1.3 end “别名new salary”

from

table

where department\_id in (10,20,30);

DECODE函数

Select department\_id，last\_name ，decode(department\_id,10,salary\*1.1,

20,salary\*1.2

salary) new\_sal

from

table

where department\_id in (10,20,30);

练习

打印出“2009年10月14日 9:25:40”格式的当前系统日期和时间

Select to\_char(sysdate,’yyyy”年”mm”月”dd”日” hh:mi:ss’) from dual;

查询员工工号，姓名，工资，以及工资提高百分之20后的结果

Select id\_num,name,salary,salary\*1.2 new\_salary from table;

将员工的姓名按首字母排序，并写出姓名的长度

select last\_name,length(last\_name) from table order by last\_name asc;

查询各员工的姓名，并显示出各员工在公司工作的月份数（worked month）;

Select name ,round(months\_between(sysdate,hire\_date),1) work\_months from table;

查询员工的姓名，以及在公司工作的月份数，并按照月份降序排序；

Select name ,round(months\_between(sysdate,hire\_date),1) work\_months from table order by hire\_date desc;

多表

笛卡尔集的错误

Select employee\_id,department\_id,department\_name from employees,department

错误原因：少了连接条件

等值连接

正确的操作：

Select e.employee\_id,e.department\_id,d.department\_name from employees e,department d where e.department\_id=d.department\_id;

employee\_id在连表中的一张表中，e.可以省略；department\_name在连表中的一张表中，d.可以省略；department\_id两张表中都有，需要指定其中的一张表，e.或者d.都行；

Select employee\_id,e.department\_id,department\_name from employees e,department d where e.department\_id=d.department\_id;

三表：

Select e.employee\_id,e.department\_id,d.department\_name l.city from employees e,department d ,localtion l where e.department\_id=d.department\_id and d.location\_id = l.location\_id;

总结：

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；

非等值连接

Select employee\_id,last\_name,salary\_level from employees e,job\_grades j

where e.salary between j.lowest\_sal and j.highest\_sal;

外连