ABS(x)
【功能】返回x的绝对值
【参数】x，数字型表达式
【返回】数字

【示例】

```
 select abs(100),abs(-100) from dual;
```

sign(x)
【功能】返回x的正负值
【参数】x，数字型表达式
【返回】数字，若为正值返回1，负值返回-1，0返回0

【示例】

```
 select sign(100),sign(-100),sign(0) from dual;
```

ceil(x)
【功能】返回大于等于x的最小整数值
【参数】x，数字型表达式
【返回】数字

【示例】

```
  select ceil(3.1),ceil(2.8+1.3),ceil(0) from dual;
```

返回4，5，0

floor(x)
【功能】返回小于等于x的最大整数值
【参数】x，数字型表达式
【返回】数字

【示例】

```
  select floor(3.1),floor(2.8+1.3),floor(0) from dual;
```

返回4，5，0

power(x,y)
【功能】返回x的y次幂
【参数】x，y 数字型表达式
【返回】数字

【示例】
```
  select power(2.5,2),power(1.5,0),power(20,-1) from dual;
```
返回：6.25,1,0.05

【相近】exp(y)
返回e的y次幂。（e为数学常量）

【关系】z=power(x,y),则y=1/log(z,x)　（条件z,x>0）

exp(y)
【功能】返回e的y次幂（e为数学常量）
【参数】y，数字型表达式
【返回】数字

【示例】
```
  select exp(3),exp(0),exp(-3) from dual;
```
返回：20.0855369,1 ,0.049787068

【相近】power(x,y)
返回e的y次幂。

【相反】ln(y)
返回e为底的自然对数。

log(x,y)
【功能】返回以x为底的y的对数
【参数】x，y,数字型表达式,
【条件】x,y都必须大于0
【返回】数字

【示例】
```
  select power(4,2),log(16,2),1/log(16,4) from dual;
```
返回：16,0.25,2

```
  select power(6.5,3),log(274.625,3),1/log(power(6.5,3),6.5) from
dual;
```
返回： 274.625 ,   0.195642521   ,   3

【相近】ln(y)
返回e为底的y的对数。（e为数学常量）

【关系】z=power(x,y),则y=1/log(z,x)   (条件z,x>0)

ln(y)
【功能】返回以e为底的y的对数（e为数学常量）
【参数】y，数字型表达式 (条件y>0)
【返回】数字

【示例】
```
  select exp(3),exp(-3),ln(20.0855369),ln(0.049787068) from dual;
```
返回：20.0855369 , 0.049787068 , 3 , -3

【相近】log(x,y)
返回以x为底的y的对数

【相反】exp(y)
返回e的y次幂

mod(x,y)
【功能】返回x除以y的余数
【参数】x,y，数字型表达式
【返回】数字

【示例】

```
  select mod(23,8),mod(24,8) from dual;
```

返回：7,0

round(x[,y])
【功能】返回四舍五入后的值
【参数】x,y，数字型表达式,如果y不为整数则截取y整数部分，如果y>0则四舍五入为y位小数，如果y小于0则四舍五入到小数点向左第y位。
【返回】数字

【示例】

```
  select round(5555.6666,2.1),round(5555.6666,-2.6),round(5555.6666)
from dual;
返回:    5555.67     ,    5600    ,    5556
```

【相近】trunc(x[,y])
返回截取后的值，用法同round(x[,y]),只是不四舍五入

trunc(x[,y])
【功能】返回x按精度y截取后的值
【参数】x,y，数字型表达式,如果y不为整数则截取y整数部分，如果y>0则截取到y位小数，如果y小于0则截取到小数点向左第y位，小数前其它数据用0表示。
【返回】数字

【示例】

```
  select
trunc(5555.66666,2.1),trunc(5555.66666,-2.6),trunc(5555.033333)
from dual;
```

返回：5555.66　　5500　　5555

【相近】round(x[,y])
返回截取后的值，用法同trunc(x[,y]),只是要做四舍五入

sqrt(x)
【功能】返回x的平方根
【参数】x数字型表达式
【返回】数字

【示例】

```
  select sqrt(64),sqrt(10) from dual;
```

返回：8 , 3.16227766

SIN(x)
【功能】返回一个数字的正弦值
【示例】select sin(1.57079) from dual;
返回： 1

SIGH(x)
【功能】返回双曲正弦的值
【示例】select sin(20),sinh(20) from dual;
返回：0.91294525, 242582598

COS(x)
【功能】返回一个给定数字的余弦
【示例】select cos(-3.1415927) from dual;
返回： -1

COSH(x)
【功能】返回一个数字反余弦值
【示例】select cosh(20) from dual;
返回：242582598

TAN
【功能返回数字的正切值
【示例】select tan(20),tan(10) from dual;
返回：2.2371609 ,0.64836083

TANH
【功能返回数字n的双曲正切值
【示例】select tanh(20),tan(20) from dual;
返回：1 ,2.2371609

ASIN(x)
【功能】给出反正弦的值
【示例】select asin(0.5) from dual;
返回：0.52359878

ACOS(x)
【功能】给出反余弦的值
【示例】select acos(-1) from dual;
返回：3.1415927

ATAN(x)
【功能】返回一个数字的反正切值
【示例】 select atan(1) from dual;
返回：0.78539816

ASCII(x1)
【功能】：返回字符表达式最左端字符的ASCII 码值。
【参数】：x1，字符表达式
【返回】：数值型
【示例】

```
SQL> select ascii('A') A,ascii('a') a,ascii(' ') space,ascii('示')
hz from dual;

A          A          SPACE         hz
---------- ---------- ---------- ----------
65         97         32         51902
```

【说明】在ASCII（）函数中，纯数字的字符串可不用‘’括起来，但含其它字符的字符串必须用‘’括起来使用，否则会出错。
　　如果最左端是汉字，只取汉字最左半边字符的ASCII 码

【互反函数】：chr()

CHR(n1)
【功能】：将ASCII 码转换为字符。
【参数】：n1,为0 ~ 255, 整数
【返回】：字符型
【示例】

```
SQL> select chr(54740) zhao,chr(65) chr65 from dual;

ZH C
-- -
赵 A
```

【互反函数】：ASCII

CONCAT(c1,c2)
【功能】连接两个字符串
【参数】c1，c2 字符型表达式

【返回】字符型

同：c1||c2

【示例】

```
select concat('010-','88888888')||'转23' 高乾竞电话 from dual;
```

```
高乾竞电话
----------------
010-88888888转23
```

INITCAP(c1)
【功能】返回字符串并将字符串的第一个字母变为大写，其它字母小写;
【参数】c1字符型表达式

【返回】字符型

【示例】

```
 SQL> select initcap('smith abc aBC') upp from dual;

UPP
-----
Smith Abc Abc
```

LOWER(c1)
【功能】：将字符串全部转为小写
【参数】：c1，字符表达式
【返回】：字符型
【示例】

```
SQL> select lower('AaBbCcDd')AaBbCcDd from dual;

AABBCCDD
--------
aabbccdd
```

【同类】UPPER()将字符串全部转为大写。

UPPER(c1)
【功能】将字符串全部转为大写
【参数】c1，字符表达式
【返回】字符型
【示例】

```
SQL> select upper('AaBbCcDd') upper from dual;

UPPER
--------
AABBCCDD
```

【同类】LOWER()将字符串全部转为小写

NLS_INITCAP(x[,y])
【功能】返回字符串并将字符串的第一个字母变为大写，其它字母小写;
【参数】x字符型表达式
【参数】Nls_param可选,
查询数据级的NLS设置:select * from nls_database_parameters;

例如:
指定排序的方式(nls_sort=) 。
nls_sort=SCHINESE_RADICAL_M（部首、笔画）
nls_sort=SCHINESE_STROKE_M（笔画、部首SCHINESE_PINYIN_M（拼音））

【返回】字符型

【示例】

```
 select nls_initcap('ab cde') "test",
nls_initcap('a c b d e','nls_sort= SCHINESE_PINYIN_M') "test1" from
dual;
```

返回：Ab Cde, A C B D E

```
 select nls_initcap('ab cde') "test",
nls_initcap('a c b d e','NLS_LANGUAGE=AMERICAN') "test1" from dual;
```

NLS_LOWER(x[,y])
【功能】返回字符串并将字符串的变为小写;
【参数】x字符型表达式
【参数】Nls_param可选，指定排序的方式(nls_sort=) 。
SCHINESE_RADICAL_M（部首、笔画）
SCHINESE_STROKE_M（笔画、部首SCHINESE_PINYIN_M（拼音））

【返回】字符型

【示例】
```
 select nls_LOWER('ab cde') "test",nls_LOWER('a c b d e','nls_sort=
SCHINESE_PINYIN_M') "test1" from dual;
```
返回：ab cde,a c b d e

NLS_UPPER(x[,y])
【功能】返回字符串并将字符串的转换为大写;
【参数】x字符型表达式
【参数】Nls_param可选，指定排序的方式(nls_sort=) 。
SCHINESE_RADICAL_M（部首、笔画）
SCHINESE_STROKE_M（笔画、部首SCHINESE_PINYIN_M（拼音））

【返回】字符型

【示例】
```
 select NLS_UPPER('ab cde') "test",NLS_UPPER('a c b d e','nls_sort=
SCHINESE_PINYIN_M') "test1" from dual;
```
返回：AB CDE,A C B D E

INSTR(C1,C2[,I[,J]])
【功能】在一个字符串中搜索指定的字符,返回发现指定的字符的位置;
【说明】多字节符(汉字、全角符等)，按1个字符计算
【参数】
C1　　被搜索的字符串
C2　　希望搜索的字符串
I　　搜索的开始位置,默认为1
J　　第J次出现的位置,默认为1
【返回】数值

【示例】select instr('oracle traning','ra',1,2) instring from dual;
返回：9

【示例】select instr('重庆某软件公司','某',1,1),instrb('重庆某软件公司','某',1,1) instring from dual;
返回：3,5

INSTRB(C1,C2[,I[,J]])
【功能】在一个字符串中搜索指定的字符,返回发现指定的字符的位置;
【说明】多字节符(汉字、全角符等)，按2个字符计算
【参数】
C1　　被搜索的字符串
C2　　希望搜索的字符串
I　　搜索的开始位置,默认为1
J　　第J次出现的位置,默认为1
【返回】数值

【示例】select instr('重庆某软件公司','某',1,1),instrb('重庆某软件公司','某',1,1) instring from dual;
返回：3,5

LENGTH(c1)
【功能】返回字符串的长度;
【说明】多字节符(汉字、全角符等)，按1个字符计算
【参数】C1 字符串
【返回】数值型
【示例】

```
SQL> select length('高乾竞'),length('北京市海锭区'),length('北京
TO_CHAR') from dual;

length('高乾竞')    length('北京市海锭区')     length('北京TO_CHAR')
----------------- ----------------            ----------------------
------
             3              6                                    9
```

LENGTH(c1)
【功能】返回字符串的长度;
【说明】多字节符(汉字、全角符等)，按2个字符计算
【参数】C1 字符串
【返回】数值型
【示例】

```
SQL> select length('高乾竞'),lengthB('高乾竞') from dual;

length('高乾竞')    lengthB('高乾竞')
----------------- ----------------
              3              6
```

LENGTHC(c1).LENGTH2(c1).LENGTH4(c1)
【功能】返回字符串的长度;
【说明】多字节符(汉字、全角符等)，按1个字符计算
【参数】C1 字符串
【返回】数值型
【示例】

```
SQL> select length('高乾竞'),length('北京市海锭区'),length('北京
TO_CHAR') from dual;
```

Oracle中的字符函数中，有一类函数是求字符长度的函数，length、lengthB、lengthC、length2、length4几个函数中比较常用的是length、lengthB。

他们的含义分别是:
Length函数返回字符的个数，使用定义是给定的字符集来计算字符的个数
LENGTHB给出该字符串的byte
LENGTHC使用纯Unicode
LENGTH2使用UCS2
LENGTH4使用UCS4

下面使一些例子:

```
Select length('你好') from dual;  2

Select lengthB('你好'),lengthC('你好'),length2('你好'), length4('你好')
from dual;
```

LPAD(c1,n[,c2])
【功能】在字符串c1的左边用字符串c2填充，直到长度为n时为止
【参数】C1 字符串
n 追加后字符总长度
c2 追加字符串,默认为空格
【返回】字符型
【说明】如果c1长度大于n，则返回c1左边n个字符
如果如果c1长度小于n，c2和c1连接后大于n，则返回连接后的右边n个字符

【示例】

```
SQL> select lpad('gao',10,'*') from dual;

lpad('gao',10,'*')
------------------
*******gao
```

不够字符则用*来填满

【相似】RPAD()在列的右边粘贴字符
【相反】LTRIM() 删除左边出现的字符串

RPAD(c1,n[,c2])
【功能】在字符串c1的右边用字符串c2填充，直到长度为n时为止
【参数】C1 字符串
n 追加后字符总长度
c2 追加字符串,默认为空格
【返回】字符型
【说明】如果c1长度大于n，则返回c1左边n个字符
如果如果c1长度小于n，c1和c2连接后大于n，则返回连接后的左边n个字符
如果如果c1长度小于n，c1和c2连接后小于n，则返回c1与多个重复c2连接(总长度>=n)后的左边n个字符

【示例】

```
SQL> select rpad('gao',10,'*a') from dual;

rpad('gao',10,'*a')
------------------
gao*a*a*a*
```

【相似】LPAD()在列的左边粘贴字符
【相反】RTRIM() 删除右边出现的字符串

LTRIM(c1,[,c2])
【功能】删除左边出现的字符串
【参数】C1 字符串
c2 追加字符串,默认为空格
【返回】字符型

【示例】

```
SQL> select LTRIM('   gao qian jing',' ') text from dual;
或：select LTRIM('   gao qian jing') text from dual;

text
------------------
gao qian jing
```

【相似】RTRIM()删除右边出现的字符串
【相反】LPAD() 在列的左边粘贴字符

RTRIM(c1,[,c2])
【功能】删除右边出现的字符串
【参数】C1 字符串
c2 追加字符串,默认为空格
【返回】字符型

【示例】

```
SQL> select RTRIM('gao qian jingXXXX','X') text from dual;

text
------------------
gao qian jing
```

【相似】LTRIM()删除左边出现的字符串
【相反】RPAD() 在列的右边粘贴字符

REPLACE(c1,c2[,c3])
【功能】将字符表达式值中，部分相同字符串，替换成新的字符串
【参数】
c1　希望被替换的字符或变量
c2　被替换的字符串
c3　要替换的字符串，默认为空(即删除之意，不是空格)
【返回】字符型

【示例】

```
SQL> select replace('he love you','he','i') test from dual;

test
------------------------------
i love you
```

SOUNDEX(c1)
【功能】返回字符串参数的语音表示形式
【参数】c1,字符型
【返回】字符串

【说明】相对于比较一些读音相同，但是拼写不同的单词是非常有用的。

计算语音的算法：

1.保留字符串首字母，但删除a、e、h、i、o、w、y

2.将下表中的数字赋给相对应的字母

(1) 1：b、f、p、v

(2) 2：c、g、k、q、s、x、z

(3) 3：d、t

(4) 4：l

(5) 5：m、n

(6) 6：r

3. 如果字符串中存在拥有相同数字的2个以上（包含2个）的字母在一起（例如b和f），或者只有h或w，则删除其他的，只保留1个

4.只返回前4个字节，不够用0填充

示例：

soundex('two'),soundex('too'),soundex('to')，他们的结果都是T000

soundex('cap'),soundex('cup')，他们的结果都是C100

soundex('house'),soundex('horse')，他们的结果都分别是H200，H620

SUBSTR(c1,n1[,n2])
【功能】取子字符串
【说明】多字节符(汉字、全角符等)，按1个字符计算
【参数】在字符表达式c1里，从n1开始取n2个字符;若不指定n2,则从第y个字符直到结束的字串.
【返回】字符型

【示例】
```
SQL> select substr('13088888888',3,8) test from dual;

test
--------
08888888
```

SUBSTRB(c1,n1[,n2])
【功能】取子字符串
【说明】多字节符(汉字、全角符等)，按2个字符计算
【参数】在字符表达式c1里，从n1开始取n2个字符;若不指定n2,则从第y个字符直到结束的字串.
【返回】字符型,如果从多字符右边开始,则用空格表示。

【示例】
select substr('我手机13012345678',4,11),substrb('我手机13012345678',4,11),substrb('我手机13012345678',3,11) test from dual;
返回:13012345678, 机13012345,手机1301234

TRANSLATE(c1,c2,c3)
【功能】将字符表达式值中，指定字符替换为新字符
【说明】多字节符(汉字、全角符等)，按1个字符计算
【参数】
c1　希望被替换的字符或变量
c2　查询原始的字符集
c3　替换新的字符集，将c2对应顺序字符，替换为c3对应顺序字符
如果c3长度大于c2，则c3长出后面的字符无效
如果c3长度小于c2，则c2长出后面的字符均替换为空(删除)
如果c3长度为0，则返回空字符串。
如果c2里字符重复，按首次位置为替换依据

【返回】字符型

【示例】
```
select TRANSLATE('he love you','he','i'),
TRANSLATE('重庆的人','重庆的','上海男'),
TRANSLATE('重庆的人','重庆的重庆','北京男士们'),
TRANSLATE('重庆的人','重庆的重庆','1北京男士们'),
TRANSLATE('重庆的人','1重庆的重庆','北京男士们') from dual;
```
返回：i love you,上海男人,北京男人,1北京人,京男士人

TRIM(c1 from c2)
【功能】删除左边和右边出现的字符串
【参数】C2 删除前字符串
c1 删除字符串,默认为空格
【返回】字符型

【示例】
```
 select TRIM('X' from 'XXXgao qian jingXXXX'),TRIM('X' from
'XXXgaoXXjingXXXX') text from dual;
```
返回：gao qian jing　　gaoXXjing

【相似】LTRIM()删除左边出现的字符串　RTRIM()删除右边出现的字符串

sysdate
【功能】：返回当前日期。
【参数】：没有参数，没有括号
【返回】：日期

【示例】select sysdate hz from dual;
返回：2008-11-5

add_months(d1,n1)
【功能】：返回在日期d1基础上再加n1个月后新的日期。
【参数】：d1，日期型，n1数字型
【返回】：日期

【示例】select sysdate,add_months(sysdate,3) hz from dual;
返回：2008-11-5,2009-2-5

last_day(d1)
【功能】：返回日期d1所在月份最后一天的日期。
【参数】：d1，日期型
【返回】：日期

【示例】select sysdate,last_day(sysdate) hz from dual;
返回：2008-11-5，2008-11-30

months_between(d1,d2)
【功能】：返回日期d1到日期d2之间的月数。
【参数】：d1，d2 日期型

【返回】：数字
如果d1>d2，则返回正数
如果d1<d2，则返回负数

【示例】
```
select sysdate,
months_between(sysdate,to_date('2006-01-01','YYYY-MM-DD')),
months_between(sysdate,to_date('2016-01-01','YYYY-MM-DD')) from
dual;
```

返回：2008-11-5,34.16,-85.84

NEW_TIME(dt1,c1,c2)
【功能】：给出时间dt1在c1时区对应c2时区的日期和时间
【参数】：dt1, d2 日期型

【返回】：日期时间

【参数】：c1,c2对应的 时区及其简写
大西洋标准时间：AST或ADT
阿拉斯加_夏威夷时间：HST或HDT
英国夏令时：BST或BDT
美国山区时间：MST或MDT
美国中央时区：CST或CDT
新大陆标准时间：NST
美国东部时间：EST或EDT
太平洋标准时间：PST或PDT
格林威治标准时间：GMT
Yukou标准时间：YST或YDT

【示例】

```
 select to_char(sysdate,'yyyy.mm.dd hh24:mi:ss') bj_time,
to_char(new_time(sysdate,'PDT','GMT'),'yyyy.mm.dd hh24:mi:ss')
los_angles from dual;
```

返回：

```
BJ_TIME             LOS_ANGLES
------------------- -------------------
2008.11.05 20:11:58 2008.11.06 03:11:58
```

【示例】

```
 select sysdate bj_time,
new_time(sysdate,'PDT','GMT') los_angles from dual;
```

返回：

```
BJ_TIME             LOS_ANGLES
------------------- -------------------
2008-11-05 20:11:58 2008-11-06 03:11:58
```

round(d1[,c1])
【功能】：给出日期d1按期间(参数c1)四舍五入后的期间的第一天日期（与数值四舍五入意思相近）
【参数】：d1日期型,c1为字符型(参数)，c1默认为j（即最近0点日期)
【参数表】：c1对应的参数表：
最近0点日期：取消参数c1或j
最近的星期日：day或dy或d
最近月初日期：month或mon或mm或rm
最近季日期：q
最近年初日期：syear或year或yyyy或yyy或yy或y(多个y表示精度)
最近世纪初日期：cc或scc

【返回】：日期

【示例】

```
select sysdate 当时日期,
round(sysdate) 最近0点日期,
round(sysdate,'day') 最近星期日,
round(sysdate,'month') 最近月初,
round(sysdate,'q') 最近季初日期,
round(sysdate,'year') 最近年初日期 from dual;
```

trunc(d1[,c1])
【功能】：返回日期d1所在期间(参数c1)的第一天日期
【参数】：d1日期型,c1为字符型(参数)，c1默认为j（即当前日期）
【参数表】：c1对应的参数表：
最近0点日期：取消参数c1或j
最近的星期日：day或dy或d （每周顺序：日，一，二，三，四，五，六）
最近月初日期：month或mon或mm或rm
最近季日期：q
最近年初日期：syear或year或yyyy或yyy或yy或y(多个y表示精度)
最近世纪初日期：cc或scc

【返回】：日期

【示例】

```
select sysdate 当时日期,
trunc(sysdate) 今天日期,
trunc(sysdate,'day') 本周星期日,
trunc(sysdate,'month') 本月初,
trunc(sysdate,'q') 本季初日期,
trunc(sysdate,'year') 本年初日期 from dual;
```

next_day(d1[,c1])
【功能】：返回日期d1在下周，星期几(参数c1)的日期
【参数】：d1日期型,c1为字符型(参数)，c1默认为j（即当前日期)
【参数表】：c1对应:星期一，星期二，星期三……星期日
【返回】：日期

【示例】

```
select sysdate 当时日期,
next_day(sysdate,'星期一') 下周星期一,
next_day(sysdate,'星期二') 下周星期二,
next_day(sysdate,'星期三') 下周星期三,
next_day(sysdate,'星期四') 下周星期四,
next_day(sysdate,'星期五') 下周星期五,
next_day(sysdate,'星期六') 下周星期六,
next_day(sysdate,'星期日') 下周星期日 from dual;
```

extract(c1 from d1)
【功能】：日期/时间d1中，参数(c1)的值
【参数】：d1日期型(date)/日期时间型(timestamp),c1为字符型(参数)
【参数表】：c1对应的参数表详见示例

【返回】：字符

【示例】

```
select
extract(hour from timestamp '2001-2-16 2:38:40 ' ) 小时,
extract(minute from timestamp '2001-2-16 2:38:40 ' ) 分钟,
extract(second from timestamp '2001-2-16 2:38:40 ' ) 秒,
extract(DAY from timestamp '2001-2-16 2:38:40 ' ) 日,
extract(MONTH from timestamp '2001-2-16 2:38:40 ' ) 月,
extract(YEAR from timestamp '2001-2-16 2:38:40 ' ) 年
 from dual;

select extract (YEAR from date '2001-2-16' ) from dual;


select sysdate 当前日期,
extract(hour from timestamp timestamp sysdate) 小时,
extract(DAY from sysdate ) 日,
extract(MONTH from sysdate ) 月,
extract(YEAR from sysdate ) 年
 from dual;
```

localtimestamp
【功能】：返回会话中的日期和时间
【参数】：没有参数，没有括号
【返回】：日期

【示例】select localtimestamp from dual;
返回：14-11月-08 12.35.37.453000 上午

current_timestamp
【功能】：以timestamp with time zone数据类型返回当前会话时区中的当前日期
【参数】：没有参数，没有括号
【返回】：日期

【示例】select current_timestamp from dual;
返回：14-11月-08 12.37.34.609000 上午 +08:00

current_date
【功能】：返回当前会话时区中的当前日期
【参数】：没有参数，没有括号
【返回】：日期

【示例】select current_date from dual;
返回：2008-11-14

dbtimezone
【功能】：返回时区
【参数】：没有参数，没有括号
【返回】：字符型

【示例】select dbtimezone from dual;

SESSIONTIMEZONE
【功能】：返回会话时区
【参数】：没有参数，没有括号
【返回】：字符型

【示例】select dbtimezone,SESSIONTIMEZONE from dual;
返回:+00:00   +08:00

INTERVAL c1 set1
【功能】：变动日期时间数值
【参数】：c1为数字字符串或日期时间字符串，set1为日期参数
【参数表】：set1具体参照示例

【返回】：日期时间格式的数值,前面多个+号
以天或天更小单位时可用数值表达式借用，如1表示1天，1/24表示1小时，1/24/60表示1分钟

【示例】

```
select
trunc(sysdate)+(interval '1' second), --加1秒(1/24/60/60)
trunc(sysdate)+(interval '1' minute), --加1分钟(1/24/60)
trunc(sysdate)+(interval '1' hour), --加1小时(1/24)
trunc(sysdate)+(INTERVAL '1' DAY),  --加1天(1)
trunc(sysdate)+(INTERVAL '1' MONTH), --加1月
trunc(sysdate)+(INTERVAL '1' YEAR), --加1年
trunc(sysdate)+(interval '01:02:03' hour to second), --加指定小时到秒
trunc(sysdate)+(interval '01:02' minute to second), --加指定分钟到秒
trunc(sysdate)+(interval '01:02' hour to minute), --加指定小时到分钟
trunc(sysdate)+(interval '2 01:02' day to minute) --加指定天数到分钟
from dual;
```

chartorowid(c1) 。。
【功能】转换varchar2类型为rowid值
【参数】c1,字符串，长度为18的字符串，字符串必须符合rowid格式
【返回】返回rowid值
【示例】

```
SELECT chartorowid('AAAADeAABAAAAZSAAA') FROM DUAL;
```

【说明】
在Oracle中，每一条记录都有一个rowid，rowid在整个数据库中是唯一的，rowid确定了每条记录是在Oracle中的哪一个数据文件、块、行上。
在重复的记录中，可能所有列的内容都相同，但rowid不会相同.

ROWIDTOCHAR(rowid) 。。
【功能】转换rowid值为varchar2类型
【参数】rowid,固定参数
【返回】返回长度为18的字符串

【示例】
```
SELECT ROWIDTOCHAR(rowid) FROM DUAL;
```

【说明】
在Oracle中，每一条记录都有一个rowid，rowid在整个数据库中是唯一的，rowid确定了每条记录是在Oracle中的哪一个数据文件、块、行上。
在重复的记录中，可能所有列的内容都相同，但rowid不会相同.

CONVERT(c1,set1,set2)
【功能】将源字符串c1 从一个语言字符集set2转换到另一个目的set1字符集
【参数】c1,字符串，set1,set2为字符型参数

【返回】字符串

【示例】

```
select convert('strutz','we8hp','f7dec') "conversion" from dual;

conver
------
strutz


select convert(name,'us7ascii','zhs16cgb231280') "conversion" from
dual;
```

HEXTORAW(c1)

【功能】将一个十六进制构成的字符串转换为二进制

【参数】c1,十六进制的字符串

【返回】字符串

【示例】

```
select HEXTORAW('A123')  from dual;
```

RAWTOHEX(c1)

【功能】将一个二进制构成的字符串转换为十六进制

【参数】c1,二进制的字符串

【返回】字符串

【示例】

```
select RAWTOHEX('A123')  from dual;
```

TO_CHAR(x[[,c2],C3])

【功能】将日期或数据转换为char数据类型

【参数】

x是一个date或number数据类型。

c2为格式参数

c3为NLS设置参数

如果x为日期nlsparm=NLS_DATE_LANGUAGE 控制返回的月份和日份所使用的语言。

如果x为数字nlsparm=NLS_NUMERIC_CHARACTERS 用来指定小数位和千分位的分隔符，以及货币符号。

NLS_NUMERIC_CHARACTERS ="dg", NLS_CURRENCY="string"

【返回】varchar2字符型

【说明1】x为数据型时

c1格式表参考:

序号	格式	简例	说明
1	,(逗号)	'9999,999'	逗号,一般以千分位出现,作为分组符号使用.如果需要您也可以当作是十分位,百分位出现,可以出现N次,视乎数字的大小而定. 变态的例子是 to_char(1234,'9,9,9,9'). 注意事项:只能出现在整数部分.
2	.(点号)	'99.99'	点号,不要念为"句号",句号是个圆圈,点好只能出现在小数点对应的地方.只能出现一次. to_char(1234.34,'9,9,9,9.99') 注意事项:只能出现在一个地方,就是原来数据小数点位置
3	$(美元符号)	'$999.99'	美元.其实你可以放在任意地方(在10G下) to_char(1234.34,'9,9,9,9.$99') 注意事项:只能出现一次.
4	0(零)	'0999.99'	零.在对应位置返回对应的字符,如果没有则以'0'填充. to_char(0.34,'9,9,9,0.$99')='$0.34';to_char(1234,'9999.00')='1234.00'; 注意事项:这是一个强制的符号,对应位没有,则以'o'填充,这是9很大不同地方
5	9	'999.99'	9.在小数位,则表示转换为对应字符,如果没有则以0表示;在整数位,没有对应则不填充字符. to_char(123,'999.99')=123.00; TO_CHAR(123,'99999.9')=123.0; 注意事项:对于0和9而言,如果格式的位数不如数字的位数多,会返回'#'. 譬如to_char(12345,'9999')='#####'
6	B(空格符)	'B999'	没有其它特别作用,在整数部分最前面加一个空格,可以出现在任意位置. 'S'||TO_CHAR(1234,'99B99')='S 1234'; 注意事项:只能出现在整数部位.
			在特定的位置返回一个ISO货币符号(就是NLS_ISO_CURRENCY参数所代表的值)

7	C(国际货币符号)	'C9999'	TO_CHAR(1233,'C9999')='CNY1234' ,这是新的国际标准RMB,关于这个可查询"国际货币符号" 注意事项:只能出现在整数部位第一位. 可以通过alter session set NLS_ISO_CURRENCY='JAPAN';来修改当前会话的设置.
8	D(ISO小数位符号)	'999D99'	这是"点号"的国际版本(ISO),作用等同于点号,也是只能出现一次.所不同的是,数据库会根据NLS_NUMERIC_CHARACTER的参数值来设置内容.默认的这个值是点号. 注意事项:没有特别需要一般不要用这个格式符号.也不要轻易修改参数值. 也可用alter sesssion set 来修改. alter session set nls_numeric_characters='!,'; to_char(1234.34,'9999d99')=1234!34
9	EEEE(科学计算符)	9.9EEEE	科学计算符号 TO_CHAR(2008032001,'9.9EEEE')='2.01E+09',由于是科学计算方法,所以小数位前面加一个9或者0即可,多个是没有意义的.
10	G(分组符号)	999G999	是逗号(,)的的ISO标准,作为分组符号使用,可以放在多个地方使用. TO_CHAR(123456,'999G9G99')=123,4,56 注意事项:同第八项 -D, 此外如果要转换出小数点,则要和D配合使用,不能和点号配合.
11	L(本地货币符号)	'L999'	是C的本地版本.可以放在整个格式的最前面和最后面. TO_CHAR(123456,'999G9G99D00L')=123,4,56.00￥ 注意事项:同第七项 C
12	MI(负号)	'9999MI'	如果是负数,在尾部加上负号(-),如果是正数,则尾巴加上空格 to_char(1234,'9999mi')||'S'||TO_CHAR(-5678,'9999MI') =1234 S5678- 注意事项:只能放在格式尾巴
13	PR(符号)	9999PR	是表达负数的另外一种方式.如果是正数,则头部加上空格;如果是负数,则用小简括号<>把数字包起来. TO_CHAR(-1234.89,'9G999D00PR')=<1,234.89> 注意事项:同12

14	RN(rn)	RN(rn)	把整数(1-3999)转换为罗马字符.RN表示转为大写,rn表示小写的.
			declare
			i int;
			begin
			for i in 1..20 loop
			dbms_output.put_line(to_char(i,'RN'));
			end loop;
			end;
			注意事项:只能自己使用,不能和其它符号组合使用.

15	S	'9999S'	是12,13的综合改进版本.为整数加一个正号+,为负数加一个符号-.S在前则加在前,在后则在后. TO_CHAR(-1234,'S9999')=-1234;TO_CHAR(1234,'S9999')=+1234
16	TM	TM9/TMe	使用这个参数等于没有用参数to_char(number)一样,应为'tm9'是默认的格式参数. to_char(1234,'tme')=1234 注意事项:格式要么是TM9,要么是TME. 当数字长度超过64位时候,TM9的输出等同于TME的输出.
17	U	U999	双币符号,例如欧元.作用同11的L TO_CHAR(999,'U999')=￥999 注意事项: 通过NLS_DUAL_CURRENCY 控制
			这是个比较古怪, 又不是很常使用的符号。它的作用在于做一个计算。

18	V	999V9	例如TO_CHAR(N,'999V9'),以p表示V的位置,则该表达式＝to_char(N×(10的P-1次方)).但是9个数又必须保证大于等于乘积之后表示的位数. TO_CHAR(5,'9V')=5*1=5; TO_CHAR(5,'9V9')=5*10=50 TO_CHAR(5,'9V99')=500 TO_CHAR(50,'9V99')='######' 9的个数不够 注意事项：格式中不能和小数表达写在一起，但是可以混合货币等。
19	X	xxxx	转换为16进制。 TO_CHAR(100,'XX')= 64 注意事项：数值必须是大于等于0的整数。前面只能和0或者FM组合使用.
20			通过以上的例子，我们了解了各种数字的格式。可以说格式太多样，难于记在脑子，最好是作为一个参考存在着. 归类： **数值类：** 0,9, **分组类：** (.),(,),D,G ,其中点好和逗号因为表示不明显，所以用小括号凸显。 **货币类：** $,C,L,U **计算转换类**：EEEE,RN,V,X **正负符号**：MI,PR,S **其它类**：B **正统类**：TM

【示例】

to_char(1210.73,'9999.9') 返回 '1210.7'

to_char(1210.73,'9,999.99') 返回 '1,210.73'

to_char(1210.73,'$9,999.00') 返回 '$1,210.73'

to_char(21,'000099') 返回 '000021'

to_char(852,'xxxx') 返回' 354'

【说明2】x为日期型,c2可用参数

序号	格式	简例	说明
1	- / , . ; : "text"	略	时间分隔符号，除了标准的几个，还允许用文字作为分割符号。 例如 to_char(sysdate,'YYYY"年"mm"月"dd"日"')=2008年04月24日
2	AD A.D.		即拉丁文Anno Domini的简写，表示公元.会根据nls的不同转换为公元或者ad等 无特殊注意事项
3	AM A.M.		上午的简写 ,同pm, p.m. （下午）, 中文环境输出为上午（如果是上午）
4	BC B.C.		虽然标准的写法是B.c. (c小写） 或者BC,好在Oracle不讲究这个。表示公元前
5	CC		返回世纪，以阿拉伯数字表示 如果年的后两位介于01-99那么，返回前两位+1，否则返回前两

	SCC		位
6	D		一周之中的某天，返回的是序号1-7
7	DAY		一周之中的某天，不过返回的是星期几而已，这和语言设置有关系，在中国环境 NLS_DATE_LANGUAGE=SIMPLIFIED CHINESE，用星期一到星期天表示
8	DD		月份中的某天(1-31)
9	DDD		年份中的某天（1-366）
10	DL	'DL'	返回长的日期格式。受到NLS_TERRITORY,NLS_LANGUAGE参数控制。例 2008年4月28日 星期一 限制：除了DL，其它什么的都不能设置。
11	DS		返回短的日期格式。受到NLS_TERRITORY,NLS_LANGUAGE参数控制。 例如 2008-04-28 限制：除了DL，其它什么的都不能设置。
12	DY		日期的简称，就是星期几（当然这指的是中国环境下）
13	E		纪元简称，但是只适合以下集中日历：日本皇室，中华民国,太过佛历
14	EE		纪元全程，适合情况同E
15	FF [1..9]		就是毫秒,如果不更上数字就是用默认的精度。 只能用于timestamp类型的。
16	FM		值得注意的一个函数：不返回任何内容。 有点不明白oracle为什么设置这个东西.
17	FX		同上
18	HH		表示小时，为12小时制，同hh12(1-12)
19	HH12		表示小时，为12小时制(1-12)
20	HH24		表示小时，为24小时制（0-23）
21	IW		ISO标准的星期序号(1-52,或者1-53）
22	IYYY IYY IY I		IYY,IY,I, ISO年(4位）的4,3，2，1位数字(倒数） to_char(to_date(21120401,'yyyymmdd'),'iyyy,iyy,iy,i')=2112, 112,12,2
23	J		儒略日(多用于天文的一种日历）,从公元前4712年一月一日算起，得出的结果是个整数，算法大体为 （公元日期+4712）*儒略日历年平均天数
24	MI		秒（0-59）
25	MM		2位月（1-12）
26	MON		月的简称，和国家有关系NLS_DATE_LANGUAGE，例如04在中文环境下用4月表示.
27	MONTH		月的名称，国家有关系NLS_DATE_LANGUAGE,目前在中文下04表示为4月。
28	PM P.M.		同am,a.m.表示下午
29	Q		季度（1-4）
30	RM		用罗马数字表示的月份，I ,II ,III ,IV ,V ,VI ,VII ,VIII,IX ,X ,XI ,XII
			有点四舍五入表示年的意思，具体的用法有那么一点点复杂。以s表示输入的年份最后两位，c表示当前的年份最后两位,其输出结果(新的年份前两位）可以用函数r=f(s,c)来表示,s2,c2分

31	RR		别表示s,c的前两位。 1）s=[0,49],c=[0,49],则r=c2 2） s=[0,49],c=[50,99],则 r=c2+1 3） s=[50,99],c=[0,49],则r=c2-1 4） s=[50,99],c=[50,99],则 r=c2 简而言之就是靠近当前年份原则，如果和当前年份同区域那么就一样，如果比当前区域大，那么就是当作是当前世纪前一世纪，否则就是下一个世纪。 举例来说，以to_date为例子 SQL> select to_date('89-01-01','rr-mm-dd') ,to_date('12-01-01','rr-mm-dd') FROM DUAL; TO_DATE('89-01-01','RR-MM-DD') TO_DATE('12-01-01','RR-MM-DD') ---------- ---------- 1989-01-01 2012-01-01 我想oracle会搞这个东东出来，估计有两个考虑一个是为了方便，一个是为了对付百年或者千年问题。
32	RRRR		如果输入参数只有两位，则同rr，否则就同yyyy作用.
33	SS		秒（0-59），一分钟内
34	SSSSS		一天从午夜开始的累积秒数.(0-86399)
35	TS		返回短日期格式内容，包括时分秒等,只能和dl,ds组合使用，格式是: dl ts或者dl ts ,中间以空格间隔开。 TO_CHAR(SYSDATE,'TS')=下午 4:50:04 表现形式受NLS_TERRITORY 和NLS_LANGUAGE影响。
36	TZD		夏令时制信息，时区简写加上夏令时信息，必须和格式tzr设置的时区对应。 包括下面三个TZ开头的，都是和时区相关，并不是直接用在to_char
37	TZH		时区中的小时，例如hh:mi:ss.fftzh:tzm'
38	TZM		时区中的分钟.
39	TZR		时区中的区域信息，必须是数据库支持的时区，例如US/Pacific
40	WW		和iw类似，也是表示星期的序号，从年的第一天算起到年的最后一个第七天。二者取值基本相同。（1-53），例如2008-01-01 到2008-01-07 算1，2008-01-09~2008-01-13 算2
41	W		一个月中的星期序号，其算法同ww,不过是局限在一月之内而已，和iso的不同。
42	X		代表本地根符号，没有特别用处，只能和timestamp类型一起使用.
43	Y,YYY		四位年，用都好分隔 例如2,008
44	YEAR SYEAR		发音表达的年，例如 2008=two thousand eight S前缀表示公元前BC
45	YYYY SYYYY		四位年，S前缀表示公元前BC
46	YYY		一次表示后面3，2，1位的年，例如2008 可以分别取值为

	YY Y		008, 08, 8
	总结		从以上看，主要就是表示时间几个部分的格式：世纪、年，月，日，时，分,秒，毫秒，以及其它一些混合格式。每个时间部分都可以有多种的表达方式，通过这样归类就比较容易记忆。 很多格式可以组合使用，这样最终可以形成足够丰富的表达其形势； 其次很多格式和nls是密切相关的；最后某些输出（返回）和格式大小写是有关系的，这在中文环境下体现不出来（目前来没有看到），但是english环境下就名下，以to_char(sysdate,'day')为例子，如果是西文环境是返回sun(假设sysdate位于周末），如果to_char(sysdate,'DAY')则返回SUN

【示例】

to_char(sysdate,'d') 每周第几天
to_char(sysdate,'dd') 每月第几天
to_char(sysdate,'ddd') 每年第几天
to_char(sysdate,'ww') 每年第几周
to_char(sysdate,'mm') 每年第几月
to_char(sysdate,'q') 每年第几季
to_char(sysdate,'yyyy') 年

SQL> select to_char(sysdate,' PM yyyy-mm-dd hh24:mi:sssss AD year mon day ddd iw') FROM DUAL;
TO_CHAR(SYSDATE,'PMYYYY-MM-DDH

上午 2008-03-27 09:58:35917 公元 two thousand eight 3月 星期四 087 13
SQL> SELECT TO_CHAR(SYSTIMESTAMP,'HH24:MI:SS.FF5') FROM DUAL;
TO_CHAR(SYSTIMESTAMP,'HH24:MI:

10:02:28.90000
SQL>SELECT TO_CHAR(SYSDATE,'DS DL') FROM DUAL
TO_CHAR(SYSDATE,'DSDL')

2008-03-27 2008年3月27日 星期四

【示例】带C3示例

```
select to_char(to_date('2002-08-26','yyyy-mm-dd'),'day','NLS_DATE_LANGUAGE = American')
from dual;
```

返回：monday

TO_DATE(X[,c2[,c3]])
【功能】将字符串X转化为日期型
【参数】c2,c3,字符型，参照to_char()
【返回】字符串

如果x格式为日期型(date)格式时，则相同表达：date x
如果x格式为日期时间型(timestamp)格式时，则相同表达：timestamp x

【相反】 to_char(date[,c2[,c3]])

【示例】

```
select to_date('199912','yyyymm'),
to_date('2000.05.20','yyyy.mm.dd'),
(date '2008-12-31') XXdate,
to_date('2008-12-31 12:31:30','yyyy-mm-dd hh24:mi:ss'),
(timestamp '2008-12-31 12:31:30') XXtimestamp
from dual;
```

TO_NUMBER(X[[,c2],c3])
【功能】将字符串X转化为数字型
【参数】c2,c3,字符型，参照to_char()
【返回】数字串

【相反】 to_char(date[[,c2],c3])

【示例】
```
select TO_NUMBER('199912'),TO_NUMBER('450.05') from dual;
```

转换为16进制。
```
TO_CHAR(100,'XX')= 64
```

TO_MULTI_BYTE(c1)
【功能】将字符串中的半角转化为全角
【参数】c1,字符型
【返回】字符串

【示例】

```
SQL> select to_multi_byte('高A') text from dual;

test
--
高Ａ
```

to_single_byte(c1)
【功能】将字符串中的全角转化为半角
【参数】c1,字符型
【返回】字符串

【示例】

```
SQL> select to_multi_byte('高Ａ') text from dual;

test
----
高A
```

nls_charset_id(c1)
【功能】返回字符集名称参应id值
【参数】c1,字符型
【返回】数值型

```
sql> select nls_charset_id('zhs16gbk') from dual;

nls_charset_id('zhs16gbk')
--------------------------
     852
```

nls_charset_name(n1)
【功能】返回字符集名称参应id值
【参数】n1,数值型
【返回】字符型

```
sql> select nls_charset_name(852) from dual;

nls_char
--------
zhs16gbk
```

AVG([distinct|all]x)
【功能】统计数据表选中行x列的平均值。

【参数】all表示对所有的值求平均值,distinct只对不同的值求平均值，默认为all
如果有参数distinct或all，需有空格与x(列)隔开。

【参数】x，只能为数值型字段

【返回】数字值

【示例】
环境：

```
create table table3(xm varchar(8),sal number(7,2));
insert into table3 values('gao',1111.11);
insert into table3 values('gao',1111.11);
insert into table3 values('zhu',5555.55);
commit;
```

执行统计：

```
select avg(distinct sal),avg(all sal),avg(sal) from table3;
```

结果： 3333.33 2592.59 2592.59

SUM([distinct|all]x)
【功能】统计数据表选中行x列的合计值。

【参数】all表示对所有的值求合计值,distinct只对不同的值求合计值，默认为all
如果有参数distinct或all，需有空格与x(列)隔开。

【参数】x，只能为数值型字段

【返回】数字值

【示例】
环境：

```
create table table3(xm varchar(8),sal number(7,2));
insert into table3 values('gao',1111.11);
insert into table3 values('gao',1111.11);
insert into table3 values('zhu',5555.55);
commit;
```

执行统计：

```
select SUM(distinct sal),SUM(all sal),SUM(sal) from table3;
```

结果： 6666.66   7777.77   7777.77

STDDEV([distinct|all]x)
【功能】统计数据表选中行x列的标准误差。

【参数】all表示对所有的值求标准误差,distinct只对不同的值求标准误差，默认为all
如果有参数distinct或all，需有空格与x(列)隔开。

【参数】x，只能为数值型字段

【返回】数字值

【示例】
环境：

```
create table table3(xm varchar(8),sal number(7,2));
insert into table3 values('gao',1111.11);
insert into table3 values('gao',1111.11);
insert into table3 values('zhu',5555.55);
commit;
```

执行统计：

```
select STDDEV(distinct sal),STDDEV(all sal),STDDEV(sal) from table3;
```

结果： 3142.69366257674 2565.99863039714 2565.99863039714

VARIANCE([distinct|all]x)
【功能】统计数据表选中行x列的方差。

【参数】all表示对所有的值求方差,distinct只对不同的值求方差，默认为all
如果有参数distinct或all，需有空格与x(列)隔开。

【参数】x，只能为数值型字段

【返回】数字值

【示例】
环境：

```
create table table3(xm varchar(8),sal number(7,2));
insert into table3 values('gao',1111.11);
insert into table3 values('gao',1111.11);
insert into table3 values('zhu',5555.55);
commit;
```

执行统计：

```
select VARIANCE(distinct sal),VARIANCE(all sal),VARIANCE(sal) from
table3;
```

结果： 9876523.4568 6584348.9712 6584348.9712

count(*|[distinct|all]x)
【功能】统计数据表选中行x列的合计值。

【参数】
*表示对满足条件的所有行统计，不管其是否重复或有空值(NULL)

all表示对所有的值统计,默认为all
distinct只对不同的值统计,
如果有参数distinct或all，需有空格与x(列)隔开，均忽略空值(NULL)。

【参数】x，可为数字、字符、日期型及其它类型的字段

【返回】数字值

count(*)=sum(1)

【示例】
环境：
```
create table table3(xm varchar(8),sal number(7,2));
insert into table3 values('gao',1111.11);
insert into table3 values('gao',1111.11);
insert into table3 values('zhu',5555.55);
insert into table3 values('',1111.11);
insert into table3 values('zhu',0);
commit;
```

执行统计：
```
select count(*),count(xm),count(all xm),count(distinct
sal),count(all sal),count(sal),sum(1) from table3;
```
结果： 5 4 4 3 5 5 5

MAX([distinct|all]x)
【功能】统计数据表选中行x列的最大值。

【参数】all表示对所有的值求最大值,distinct只对不同的值求最大值，默认为all
如果有参数distinct或all，需有空格与x(列)隔开。

【参数】x，可为数字、字符或日期型字段

【返回】对应x字段类型

【示例】
环境：

```
create table table3(xm varchar(8),sal number(7,2));
insert into table3 values('gao',1111.11);
insert into table3 values('gao',1111.11);
insert into table3 values('zhu',5555.55);
insert into table3 values('',1111.11);
insert into table3 values('zhu',0);
commit;
```

执行统计：

```
select MAX(distinct sal),MAX(xm) from table3;
```

结果：5555.55   zhu

MIN([distinct|all]x)
【功能】统计数据表选中行x列的最大值。

【参数】all表示对所有的值求最大值,distinct只对不同的值求最大值，默认为all
如果有参数distinct或all，需有空格与x(列)隔开。

【参数】x，可为数字、字符或日期型字段

【返回】对应x字段类型
注：字符型字段，将忽略空值(NULL)

【示例】
环境：

```
create table table3(xm varchar(8),sal number(7,2));
insert into table3 values('gao',1111.11);
insert into table3 values('gao',1111.11);
insert into table3 values('zhu',5555.55);
insert into table3 values('',1111.11);
insert into table3 values('zhu',0);
commit;
```

执行统计：

```
select MIN(distinct sal),MIN(xm),MIN(distinct xm),MIN(all xm) from
table3;
```

结果：0 gao gao gao

oracle分析函数--SQL*PLUS环境
一、总体介绍
12.1 分析函数如何工作
语法 FUNCTION_NAME(<参数>,…) OVER (<PARTITION BY 表达式,…> <ORDER BY 表达式 <ASC DESC> <NULLS FIRST NULLS LAST>> <WINDOWING子句>) PARTITION子句 ORDER BY子句 WINDOWING子句 缺省时相当于RANGE UNBOUNDED PRECEDING
1. 值域窗(RANGE WINDOW)
RANGE N PRECEDING 仅对数值或日期类型有效,选定窗为排序后当前行之前,某列(即排序列)值大于/小于(当前行该列值 –/+ N)的所有行,因此与ORDER BY子句有关系。
2. 行窗(ROW WINDOW)
ROWS N PRECEDING 选定窗为当前行及之前N行。
还可以加上BETWEEN AND 形式,例如RANGE BETWEEN m PRECEDING AND n FOLLOWING
函数 AVG(<distinct all> eXPr)
一组或选定窗中表达式的平均值 CORR(expr, expr) 即COVAR_POP(exp1,exp2) / (STDDEV_POP(expr1) * STDDEV_POP(expr2)),两个表达式的互相关,-1(反相关) ~ 1(正相关),0表示不相关
COUNT(<distinct> <*> <expr>) 计数
COVAR_POP(expr, expr) 总体协方差
COVAR_SAMP(expr, expr) 样本协方差
CUME_DIST 累积分布,即行在组中的相对位置,返回0 ~ 1
DENSE_RANK 行的相对排序(与ORDER BY搭配),相同的值具有一样的序数(NULL计为相同),并不留空序数
FIRST_VALUE 一个组的第一个值
LAG(expr, <offset>, <default>) 访问之前的行,OFFSET是缺省为1 的正数,表示相对行数,DEFAULT是当超出选定窗范围时的返回值(如第一行不存在之前行)
LAST_VALUE 一个组的最后一个值
LEAD(expr, <offset>, <default>) 访问之后的行,OFFSET是缺省为1 的正数,表示相对行数,DEFAULT是当超出选定窗范围时的返回值(如最后行不存在之前行)
MAX(expr) 最大值
MIN(expr) 最小值
NTILE(expr) 按表达式的值和行在组中的位置编号,如表达式为4,则组分4份,分别为1 ~ 4的值,而不能等分则多出的部分在值最小的那组
PERCENT_RANK 类似CUME_DIST,1/(行的序数 - 1)
RANK 相对序数,答应并列,并空出随后序号
RATIO_TO_REPORT(expr) 表达式值 / SUM(表达式值)
ROW_NUMBER 排序的组中行的偏移
STDDEV(expr) 标准差
STDDEV_POP(expr) 总体标准差
STDDEV_SAMP(expr) 样本标准差
SUM(expr) 合计
VAR_POP(expr) 总体方差
VAR_SAMP(expr) 样本方差
VARIANCE(expr) 方差
REGR_ xxxx(expr, expr) 线性回归函数
REGR_SLOPE：返回斜率，等于COVAR_POP(expr1, expr2) / VAR_POP(expr2)
REGR_INTERCEPT：返回回归线的y截距，等于

AVG(expr1) - REGR_SLOPE(expr1, expr2) * AVG(expr2)
REGR_COUNT：返回用于填充回归线的非空数字对的数目
REGR_R2：返回回归线的决定系数，计算式为：
If VAR_POP(expr2) = 0 then return NULL
If VAR_POP(expr1) = 0 and VAR_POP(expr2) != 0 then return 1
If VAR_POP(expr1) > 0 and VAR_POP(expr2 != 0 then
return POWER(CORR(expr1,expr),2)
REGR_AVGX：计算回归线的自变量(expr2)的平均值，去掉了空对(expr1, expr2)后，等于AVG(expr2)
REGR_AVGY：计算回归线的应变量(expr1)的平均值，去掉了空对(expr1, expr2)后，等于AVG(expr1)
REGR_SXX： 返回值等于REGR_COUNT(expr1, expr2) * VAR_POP(expr2)
REGR_SYY： 返回值等于REGR_COUNT(expr1, expr2) * VAR_POP(expr1)
REGR_SXY：返回值等于REGR_COUNT(expr1, expr2) * COVAR_POP(expr1, expr2)

首先：创建表及接入测试数据

```
create table students
(id number(15,0),
area varchar2(10),
stu_type varchar2(2),
score number(20,2));
insert into students values(1, '111', 'g', 80 );
insert into students values(1, '111', 'j', 80 );
insert into students values(1, '222', 'g', 89 );
insert into students values(1, '222', 'g', 68 );
insert into students values(2, '111', 'g', 80 );
insert into students values(2, '111', 'j', 70 );
insert into students values(2, '222', 'g', 60 );
insert into students values(2, '222', 'j', 65 );
insert into students values(3, '111', 'g', 75 );
insert into students values(3, '111', 'j', 58 );
insert into students values(3, '222', 'g', 58 );
insert into students values(3, '222', 'j', 90 );
insert into students values(4, '111', 'g', 89 );
insert into students values(4, '111', 'j', 90 );
insert into students values(4, '222', 'g', 90 );
insert into students values(4, '222', 'j', 89 );
commit;
```

二、具体应用：
1、分组求和：
1）GROUP BY子句

```
--A、GROUPING SETS

select id,area,stu_type,sum(score) score
from students
group by grouping sets((id,area,stu_type),(id,area),id)
order by id,area,stu_type;

/*--------理解grouping sets
select a, b, c, sum( d ) from t
group by grouping sets ( a, b, c )
```

等效于

```
select * from (
select a, null, null, sum( d ) from t group by a
union all
select null, b, null, sum( d ) from t group by b
union all
select null, null, c, sum( d ) from t group by c
)
*/

--B、ROLLUP

select id,area,stu_type,sum(score) score
from students
group by rollup(id,area,stu_type)
order by id,area,stu_type;

/*--------理解rollup
select a, b, c, sum( d )
from t
group by rollup(a, b, c);
```

等效于

```
select * from (
select a, b, c, sum( d ) from t group by a, b, c
union all
select a, b, null, sum( d ) from t group by a, b
union all
select a, null, null, sum( d ) from t group by a
union all
select null, null, null, sum( d ) from t
)
*/

--C、CUBE

select id,area,stu_type,sum(score) score
from students
group by cube(id,area,stu_type)
order by id,area,stu_type;

/*--------理解cube
select a, b, c, sum( d ) from t
group by cube( a, b, c)
```

等效于

```
select a, b, c, sum( d ) from t
group by grouping sets(
( a, b, c ),
( a, b ), ( a ), ( b, c ),
( b ), ( a, c ), ( c ),
```

```
() )
*/

--D、GROUPING
/*从上面的结果中我们很容易发现,每个统计数据所对应的行都会出现null,
如何来区分到底是根据那个字段做的汇总呢,grouping函数判断是否合计列!*/

select decode(grouping(id),1,'all id',id) id,
decode(grouping(area),1,'all area',to_char(area)) area,
decode(grouping(stu_type),1,'all_stu_type',stu_type) stu_type,
sum(score) score
from students
group by cube(id,area,stu_type)
order by id,area,stu_type;
```

二、OVER()函数的使用

1、统计名次——DENSE_RANK(),ROW_NUMBER()

1)允许并列名次、名次不间断，DENSE_RANK()，结果如122344456……

```
将score按ID分组排名：dense_rank() over(partition by id order by score
desc)
将score不分组排名：dense_rank() over(order by score desc)
select id,area,score,
dense_rank() over(partition by id order by score desc) 分组id排序,
dense_rank() over(order by score desc) 不分组排序
from students order by id,area;
```

2)不允许并列名次、相同值名次不重复，ROW_NUMBER()，结果如123456……

```
将score按ID分组排名：row_number() over(partition by id order by score
desc)
将score不分组排名：row_number() over(order by score desc)
select id,area,score,
row_number() over(partition by id order by score desc) 分组id排序,
row_number() over(order by score desc) 不分组排序
from students order by id,area;
```

3)允许并列名次、复制名次自动空缺，rank()，结果如12245558……

```
将score按ID分组排名：rank() over(partition by id order by score desc)
将score不分组排名：rank() over(order by score desc)
select id,area,score,
rank() over(partition by id order by score desc) 分组id排序,
rank() over(order by score desc) 不分组排序
from students order by id,area;
```

4)名次分析，cume_dist()——-最大排名/总个数

```
函数：cume_dist() over(order by id)
select id,area,score,
cume_dist() over(order by id) a, --按ID最大排名/总个数
cume_dist() over(partition by id order by score desc) b, --ID分组中,
scroe最大排名值/本组总个数
row_number() over (order by id) 记录号
from students order by id,area;
```

5)利用cume_dist()，允许并列名次、复制名次自动空缺，取并列后较大名次，结果如22355778……
将score按ID分组排名：cume_dist() over(partition by id order by score desc)*sum(1) over(partition by id)
将score不分组排名：cume_dist() over(order by score desc)*sum(1) over()

```
select id,area,score,
sum(1) over() as 总数,
sum(1) over(partition by id) as 分组个数,
(cume_dist() over(partition by id order by score desc))*(sum(1) over(partition by id)) 分组id排序,
(cume_dist() over(order by score desc))*(sum(1) over()) 不分组排序
from students order by id,area
```

2、分组统计－－sum(),max(),avg(),RATIO_TO_REPORT()

```
select id,area,
sum(1) over() as 总记录数,
sum(1) over(partition by id) as 分组记录数,
sum(score) over() as 总计 ,
sum(score) over(partition by id) as 分组求和,
sum(score) over(order by id) as  分组连续求和,
sum(score) over(partition by id,area) as 分组ID和area求和,
sum(score) over(partition by id order by area) as 分组ID并连续按area求和,
max(score) over() as 最大值,
max(score) over(partition by id) as 分组最大值,
max(score) over(order by id) as 分组连续最大值,
max(score) over(partition by id,area) as 分组ID和area求最大值,
max(score) over(partition by id order by area) as 分组ID并连续按area求最大值,
avg(score) over() as 所有平均,
avg(score) over(partition by id) as 分组平均,
avg(score) over(order by id) as 分组连续平均,
avg(score) over(partition by id,area) as 分组ID和area平均,
avg(score) over(partition by id order by area) as 分组ID并连续按area平均,
RATIO_TO_REPORT(score) over() as "占所有%",
RATIO_TO_REPORT(score) over(partition by id) as "占分组%",
score from students;
```

3、LAG(COL,n,default)、LEAD(OL,n,default) --取前后边N条数据
取前面记录的值：lag(score,n,x) over(order by id)
取后面记录的值：lead(score,n,x) over(order by id)
参数：n表示移动N条记录，X表示不存在时填充值，iD表示排序列

```
select id,lag(score,1,0) over(order by id) lg,score from students;
select id,lead(score,1,0) over(order by id) lg,score from students;
```

4、FIRST_VALUE()、LAST_VALUE()

取第起始1行值：first_value(score,n) over(order by id)
取第最后1行值：LAST_value(score,n) over(order by id)

```
select id,first_value(score) over(order by id) fv,score from
students;
select id,last_value(score) over(order by id) fv,score from
students;
```

sum(...) over ...
【功能】连续求和分析函数
【参数】具体参示例
【说明】Oracle分析函数

NC示例：
```
select bdcode,sum(1) over(order by bdcode) aa from bd_bdinfo
```

【示例】
1.原表信息： SQL> break on deptno skip 1 -- 为效果更明显，把不同部门的数据隔段显示。
```
SQL> select deptno,ename,sal
   2   from emp
   3   order by deptno;

DEPTNO ENAME          SAL
---------- ---------- ----------
       10 CLARK          2450
          KING          5000
          MILLER           1300

       20 SMITH          800
          ADAMS          1100
          FORD          3000
          SCOTT          3000
          JONES          2975

       30 ALLEN          1600
          BLAKE          2850
          MARTIN           1250
          JAMES          950
          TURNER           1500
          WARD          1250
```

2.先来一个简单的，注意over(...)条件的不同，
使用 sum(sal) over (order by ename)... 查询员工的薪水“连续”求和，
注意over (order by ename)如果没有order by 子句，求和就不是“连续”的，
放在一起，体会一下不同之处：

```
SQL> select deptno,ename,sal,
   2   sum(sal) over (order by ename) 连续求和,
   3   sum(sal) over () 总和,                -- 此处sum(sal) over ()等同于sum(sal)
   4   100*round(sal/sum(sal) over (),4) "份额(%)"
   5   from emp
   6   /

DEPTNO ENAME          SAL 连续求和    总和 份额(%)
---------- ---------- ---------- ---------- ---------- ----------
       20 ADAMS          1100    1100    29025    3.79
       30 ALLEN          1600    2700    29025    5.51
       30 BLAKE          2850    5550    29025    9.82
       10 CLARK          2450    8000    29025    8.44
```

```
        20 FORD           3000    11000    29025    10.34
        30 JAMES           950    11950    29025    3.27
        20 JONES          2975     14925    29025     10.25
        10 KING           5000    19925    29025    17.23
        30 MARTIN            1250    21175    29025    4.31
        10 MILLER            1300    22475    29025    4.48
        20 SCOTT           3000     25475    29025     10.34
        20 SMITH           800     26275    29025    2.76
        30 TURNER            1500    27775    29025    5.17
        30 WARD           1250    29025    29025    4.31
```

3.使用子分区查出各部门薪水连续的总和。注意按部门分区。注意over(...)条件的不同，
sum(sal) over (partition by deptno order by ename) 按部门“连续”求总和
sum(sal) over (partition by deptno) 按部门求总和
sum(sal) over (order by deptno, ename) 不按部门“连续”求总和
sum(sal) over () 不按部门，求所有员工总和，效果等同于sum(sal)。

```
SQL> select deptno,ename,sal,
   2   sum(sal) over (partition by deptno order by ename) 部门连续求和,--各部门的薪水"连续"求和
   3   sum(sal) over (partition by deptno) 部门总和,   -- 部门统计的总和，同一部门总和不变
   4   100*round(sal/sum(sal) over (partition by deptno),4) "部门份额(%)",
   5   sum(sal) over (order by deptno,ename) 连续求和, --所有部门的薪水"连续"求和
   6   sum(sal) over () 总和,   -- 此处sum(sal) over () 等同于sum(sal)，所有员工的薪水总和
   7   100*round(sal/sum(sal) over (),4) "总份额(%)"
   8   from emp
   9   /

DEPTNO ENAME SAL 部门连续求和 部门总和 部门份额(%) 连续求和 总和   总份额(%)
------ ------ ----- ------------ ---------- ----------- ---------- -
------ ----------
10 CLARK 2450       2450    8750       28    2450   29025    8.44
   KING 5000       7450    8750    57.14    7450   29025    17.23
   MILLER   1300       8750    8750    14.86    8750   29025    4.48

20 ADAMS 1100       1100    10875    10.11    9850   29025    3.79
   FORD 3000       4100    10875    27.59    12850   29025    10.34
   JONES 2975       7075    10875    27.36    15825   29025    10.25
   SCOTT 3000        10075    10875    27.59    18825   29025
10.34
   SMITH 800        10875    10875        7.36    19625   29025
2.76

30 ALLEN 1600       1600    9400    17.02    21225   29025    5.51
   BLAKE 2850       4450    9400    30.32    24075   29025    9.82
   JAMES 950       5400    9400    10.11    25025   29025    3.27
   MARTIN   1250       6650    9400        13.3    26275   29025
4.31
   TURNER   1500       8150    9400    15.96    27775   29025
```

```
5.17
   WARD 1250       9400    9400        13.3    29025   29025    4.31
```

4.来一个综合的例子，求和规则有按部门分区的，有不分区的例子

```
SQL> select deptno,ename,sal,sum(sal) over (partition by deptno
order by sal) dept_sum,
   2   sum(sal) over (order by deptno,sal) sum
   3   from emp;

DEPTNO ENAME          SAL DEPT_SUM        SUM
---------- ---------- ---------- ---------- ----------
       10 MILLER           1300    1300    1300
          CLARK          2450    3750    3750
          KING          5000    8750    8750

       20 SMITH           800        800    9550
          ADAMS          1100    1900    10650
          JONES          2975    4875    13625
          SCOTT          3000    10875    19625
          FORD          3000    10875    19625

       30 JAMES           950        950    20575
          WARD          1250    3450    23075
          MARTIN           1250    3450    23075
          TURNER           1500    4950    24575
          ALLEN          1600    6550    26175
          BLAKE          2850    9400    29025
```

5.来一个逆序的，即部门从大到小排列，部门里各员工的薪水从高到低排列，累计和的规则不变。

```
SQL> select deptno,ename,sal,
   2   sum(sal) over (partition by deptno order by deptno desc,sal
desc) dept_sum,
   3   sum(sal) over (order by deptno desc,sal desc) sum
   4   from emp;

DEPTNO ENAME          SAL DEPT_SUM        SUM
---------- ---------- ---------- ---------- ----------
       30 BLAKE          2850    2850    2850
          ALLEN          1600    4450    4450
          TURNER           1500    5950    5950
          WARD          1250    8450    8450
          MARTIN           1250    8450    8450
          JAMES          950    9400    9400

       20 SCOTT          3000    6000    15400
          FORD          3000    6000    15400
          JONES          2975    8975    18375
          ADAMS          1100    10075    19475
          SMITH          800    10875    20275

       10 KING          5000    5000    25275
          CLARK          2450    7450    27725
          MILLER           1300    8750    29025
```

6.体会：在"... from emp;"后面不要加order by 子句，使用的分析函数的(partition by deptno order by sal)
里已经有排序的语句了，如果再在句尾添加排序子句，一致倒罢了，不一致，结果就令人费劲了。如：

```
SQL> select deptno,ename,sal,sum(sal) over (partition by deptno
order by sal) dept_sum,
   2   sum(sal) over (order by deptno,sal) sum
   3   from emp
   4   order by deptno desc;

DEPTNO ENAME          SAL DEPT_SUM        SUM
---------- ---------- ---------- ---------- ----------
       30 JAMES           950        950    20575
          WARD           1250    3450    23075
          MARTIN            1250    3450    23075
          TURNER            1500    4950    24575
          ALLEN           1600    6550    26175
          BLAKE           2850    9400    29025

       20 SMITH           800        800    9550
          ADAMS           1100    1900    10650
          JONES           2975    4875    13625
          SCOTT           3000    10875    19625
          FORD           3000    10875    19625

       10 MILLER            1300    1300    1300
          CLARK           2450    3750    3750
          KING           5000    8750    8750
```

RANK()
dense_rank()
【语法】RANK ( ) OVER ([query_partition_clause] order_by_clause)
dense_RANK ( ) OVER ([query_partition_clause] order_by_clause)

【功能】聚合函数RANK 和 dense_rank 主要的功能是计算一组数值中的排序值。
【参数】dense_rank与rank()用法相当，
【区别】dence_rank在并列关系是，相关等级不会跳过。rank则跳过
rank()是跳跃排序，有两个第二名时接下来就是第四名（同样是在各个分组内）
dense_rank()l是连续排序，有两个第二名时仍然跟着第三名。
【说明】Oracle分析函数

【示例】
聚合函数RANK 和 dense_rank 主要的功能是计算一组数值中的排序值。

在9i版本之前，只有分析功能（analytic ），即从一个查询结果中计算每一行的排序值，是基于order_by_clause子句中的value_exprs指定字段的。

其语法为：

```
RANK ( ) OVER ( [query_partition_clause] order_by_clause )
```

在9i版本新增加了合计功能（aggregate），即对给定的参数值在设定的排序查询中计算出其排序值。这些参数必须是常数或常值表达式，且必须和ORDER BY子句中的字段个数、位置、类型完全一致。

其语法为：

```
RANK ( expr [, expr]... ) WITHIN GROUP
( ORDER BY
expr [ DESC | ASC ] [NULLS { FIRST | LAST }]
[, expr [ DESC | ASC ] [NULLS { FIRST | LAST }]]...
)
```

例子1：

有表Table内容如下

```
COL1  COL2
   1  1
   2  1
   3  2
   3  1
   4  1
   4  2
   5  2
```

```
   5  2
   6  2
```

分析功能：列出Col2分组后根据Col1排序,并生成数字列。比较实用于在成绩表中查出各科前几名的信息。

```
SELECT a.*,RANK() OVER(PARTITION BY col2 ORDER BY col1) "Rank"
FROM table a;
```

结果如下：

```
COL1  COL2  Rank
   1  1     1
   2  1     2
   3  1     3
   4  1     4
   3  2     1
   4  2     2
   5  2     3
   5  2     3
   6  2     5
```

例子2：

TABLE：A （科目，分数）

```
数学，80
语文，70
数学，90
数学，60
数学，100
语文，88
语文，65
语文，77
```

现在我想要的结果是：（即想要每门科目的前3名的分数）

```
 数学，100
数学，90
数学，80
语文，88
语文，77
语文，70
```

那么语句就这么写：

```
select * from (select rank() over(partition by 科目 order by 分数
desc) rk,a.* from a) t
```

where t.rk<=3;

例子3：

合计功能：计算出数值(4,1)在Orade By Col1,Col2排序下的排序值，也就是col1=4,col2=1在排序以后的位置

SELECT RANK(4,3) WITHIN GROUP (ORDER BY col1,col2) "Rank" FROM table;

结果如下：

Rank

4

dense_rank与rank()用法相当，但是有一个区别：dence_rank在并列关系是，相关等级不会跳过。rank则跳过

例如：表

A	B	C
a	liu	wang
a	jin	shu
a	cai	kai
b	yang	du
b	lin	ying
b	yao	cai
b	yang	99

例如：当rank时为：

select m.a,m.b,m.c,rank() over(partition by a order by b) liu from test3 m

A	B	C	LIU
a	cai	kai	1
a	jin	shu	2
a	liu	wang	3
b	lin	ying	1
b	yang	du	2
b	yang	99	2
b	yao	cai	4

而如果用dense_rank时为：

select m.a,m.b,m.c,dense_rank() over(partition by a order by b) liu from test3 m

A	B	C	LIU

a	cai	kai	1
a	jin	shu	2
a	liu	wang	3
b	lin	ying	1
b	yang	du	2
b	yang	99	2
b	yao	cai	3

ROW_NUMBER()
【语法】ROW_NUMBER() OVER (PARTITION BY COL1 ORDER BY COL2)
【功能】表示根据COL1分组，在分组内部根据 COL2排序，而这个值就表示每组内部排序后的顺序编号（组内连续的唯一的）
row_number() 返回的主要是“行”的信息，并没有排名
【参数】
【说明】Oracle分析函数

主要功能：用于取前几名，或者最后几名等

【示例】
表内容如下：

name	seqno	description
A	1	test
A	2	test
A	3	test
A	4	test
B	1	test
B	2	test
B	3	test
B	4	test
C	1	test
C	2	test
C	3	test
C	4	test

我想有一个sql语句，搜索的结果是

A	1	test
A	2	test
B	1	test
B	2	test
C	1	test
C	2	test

实现：

```
select name,seqno,description
from(select name,seqno,description,row_number() over (partition by
name order by seqno) id
from table_name) where id<=3;
```

lag()和lead()
【语法】
lag(EXPR,<OFFSET>,<DEFAULT>)
LEAD(EXPR,<OFFSET>,<DEFAULT>)
【功能】表示根据COL1分组，在分组内部根据 COL2排序，而这个值就表示每组内部排序后的顺序编号（组内连续的唯一的）
lead （） 下一个值 lag（） 上一个值

【参数】
EXPR是从其他行返回的表达式
OFFSET是缺省为1 的正数，表示相对行数。希望检索的当前行分区的偏移量
DEFAULT是在OFFSET表示的数目超出了分组的范围时返回的值。
【说明】Oracle分析函数

【示例】

```
-- Create table
create table LEAD_TABLE
(
 CASEID VARCHAR2(10),
 STEPID VARCHAR2(10),
 ACTIONDATE DATE
)
tablespace COLM_DATA
 pctfree 10
 initrans 1
 maxtrans 255
 storage
 (
 initial 64K
 minextents 1
 maxextents unlimited
 );

insert into LEAD_TABLE
values('Case1','Step1',to_date('20070101','yyyy-mm-dd'));
insert into LEAD_TABLE
values('Case1','Step2',to_date('20070102','yyyy-mm-dd'));
insert into LEAD_TABLE
values('Case1','Step3',to_date('20070103','yyyy-mm-dd'));
insert into LEAD_TABLE
values('Case1','Step4',to_date('20070104','yyyy-mm-dd'));
insert into LEAD_TABLE
values('Case1','Step5',to_date('20070105','yyyy-mm-dd'));
insert into LEAD_TABLE
values('Case1','Step4',to_date('20070106','yyyy-mm-dd'));
insert into LEAD_TABLE
values('Case1','Step6',to_date('20070101','yyyy-mm-dd'));
insert into LEAD_TABLE
values('Case1','Step1',to_date('20070201','yyyy-mm-dd'));
insert into LEAD_TABLE
values('Case2','Step2',to_date('20070202','yyyy-mm-dd'));
insert into LEAD_TABLE
values('Case2','Step3',to_date('20070203','yyyy-mm-dd'));
```

```
commit;
```

结果如下：

```
Case1 Step1 2007-1-1 Step2 2007-1-2
Case1 Step2 2007-1-2 Step3 2007-1-3 Step1 2007-1-1
Case1 Step3 2007-1-3 Step4 2007-1-4 Step2 2007-1-2
Case1 Step4 2007-1-4 Step5 2007-1-5 Step3 2007-1-3
Case1 Step5 2007-1-5 Step4 2007-1-6 Step4 2007-1-4
Case1 Step4 2007-1-6 Step6 2007-1-7 Step5 2007-1-5
Case1 Step6 2007-1-7 Step4 2007-1-6
Case2 Step1 2007-2-1 Step2 2007-2-2
Case2 Step2 2007-2-2 Step3 2007-2-3 Step1 2007-2-1
Case2 Step3 2007-2-3 Step2 2007-2-2
```

还可以进一步统计一下两者的相差天数

```
select caseid,stepid,actiondate,nextactiondate,nextactiondate-
actiondate datebetween from (
select caseid,stepid,actiondate,lead(stepid) over (partition by
caseid order by actiondate) nextstepid,
lead(actiondate) over (partition by caseid order by actiondate)
nextactiondate,
lag(stepid) over (partition by caseid order by actiondate)
prestepid,
lag(actiondate) over (partition by caseid order by actiondate)
preactiondate
from lead_table)
```

结果如下：

```
Case1 Step1 2007-1-1 2007-1-2 1
Case1 Step2 2007-1-2 2007-1-3 1
Case1 Step3 2007-1-3 2007-1-4 1
Case1 Step4 2007-1-4 2007-1-5 1
Case1 Step5 2007-1-5 2007-1-6 1
Case1 Step4 2007-1-6 2007-1-7 1
Case1 Step6 2007-1-7
Case2 Step1 2007-2-1 2007-2-2 1
Case2 Step2 2007-2-2 2007-2-3 1
Case2 Step3 2007-2-3
```

每一条记录都能连接到上/下一行的内容

lead () 下一个值 lag () 上一个值

```
select caseid,stepid,actiondate,lead(stepid) over (partition by
caseid order by actiondate) nextstepid,
lead(actiondate) over (partition by caseid order by actiondate)
nextactiondate,
lag(stepid) over (partition by caseid order by actiondate)
prestepid,
lag(actiondate) over (partition by caseid order by actiondate)
```

```
preactiondate
from lead_table
```

DUMP(w[,x[,y[,z]]])
【功能】返回数据类型、字节长度和在内部的存储位置.
【参数】
w为各种类型的字符串（如字符型、数值型、日期型……）
x为返回位置用什么方式表达，可为:8,10,16或17，分别表示：8/10/16进制和字符型，默认为10。
y和z决定了内部参数位置

【返回】类型 <[长度]>，符号/指数位 [数字1，数字2，数字3，.....，数字20]
如：Typ=2 Len=7: 60,89,67,45,23,11,102

SELECT DUMP('ABC',1016) FROM dual;
返回结果为：Typ=96 Len=3 CharacterSet=ZHS16GBK: 41,42,43

代码 数据类型
0 对应 VARCHAR2
1 对应 NUMBER
8 对应 LONG
12 对应 DATE
23 对应 RAW
24 对应 LONG RAW
69 对应 ROWID
96 对应 CHAR
106 对应 MSSLABEL

各位的含义如下：

1.类型：Number型,Type=2 (类型代码可以从Oracle的文档上查到)
2.长度:指存储的字节数
3.符号/指数位
在存储上,Oracle对正数和负数分别进行存储转换:
正数：加1存储(为了避免Null)
负数：被101减,如果总长度小于21个字节，最后加一个102(是为了排序的需要)

指数位换算：
正数：指数=符号/指数位 - 193 (最高位为1是代表正数)
负数：指数=62 - 第一字节

4.从<数字1>开始是有效的数据位

从<数字1>开始是最高有效位,所存储的数值计算方法为：

将下面计算的结果加起来：

每个<数字位>乘以100^(指数-N) (N是有效位数的顺序位，第一个有效位的N=0)

5、举例说明

```
SQL> select dump(123456.789) from dual;
返回：Typ=2 Len=6：195,13,35,57,79,91

<指数>：    195 - 193 = 2
<数字1>      13 - 1     = 12 *100^(2-0) 120000
<数字2>      35 - 1     = 34 *100^(2-1) 3400
<数字3>      57 - 1     = 56 *100^(2-2) 56
<数字4>      79 - 1     = 78 *100^(2-3) .78
<数字5>      91 - 1     = 90 *100^(2-4) .009
                              123456.789

SQL> select dump(-123456.789) from dual;
返回：Typ=2 Len=7：60,89,67,45,23,11,102
算法：
<指数>  62 - 60 = 2(最高位是0，代表为负数)
<数字1> 101 - 89 = 12 *100^(2-0) 120000
<数字2> 101 - 67 = 34 *100^(2-1) 3400
<数字3> 101 - 45 = 56 *100^(2-2) 56
<数字4> 101 - 23 = 78 *100^(2-3) .78
<数字5> 101 - 11 = 90 *100^(2-4) .009
                                  123456.789(-)
```

现在再考虑一下为什么在最后加102是为了排序的需要，-123456.789在数据库中实际存储为

60,89,67,45,23,11

而-123456.78901在数据库中实际存储为

60,89,67,45,23,11,91

可见，如果不在最后加上102，在排序时会出现-123456.789<-123456.78901的情况。

greatest(exp1,exp2,exp3,……,expn)
【功能】返回表达式列表中值最大的一个。如果表达式类型不同，会隐含转换为第一个表达式类型。
【参数】exp1……n，各类型表达式
【返回】exp1类型

【示例】

```
 SELECT greatest(10,32,'123','2006') FROM dual;

  SELECT greatest('kdnf','dfd','a','206') FROM dual;
```

least(exp1,exp2,exp3,……,expn)
【功能】返回表达式列表中值最小的一个。如果表达式类型不同，会隐含转换为第一个表达式类型。
【参数】exp1……n，各类型表达式
【返回】exp1类型

【示例】

```
  SELECT least(10,32,'123','2006') FROM dual;

SELECT least('kdnf','dfd','a','206') FROM dual;
```

【语法】NVL (expr1, expr2)
【功能】若expr1为NULL，返回expr2；expr1不为NULL，返回expr1。
注意两者的类型要一致

【语法】NVL2 (expr1, expr2, expr3)
【功能】expr1不为NULL，返回expr2；expr2为NULL，返回expr3。
expr2和expr3类型不同的话，expr3会转换为expr2的类型

user
【功能】返回当前会话对应的数据库用户名。
【参数】无

【返回】字符型

uid
【功能】返回当前会话所对应的用户id号。
【参数】无

【返回】字符型

userenv(parameter)
【功能】返回当前会话上下文属性。
【参数】Parameter是参数，可以用以下参数代替：
Isdba:若用户具有dba权限，则返回true,否则返回false.
Language:返回当前会话对应的语言、地区和字符集。
LANG:返回当前环境的语言的缩写
Terminal:返回当前会话所在终端的操作系统标识符。
Sessionid:返回正在使用的审计会话号.
Client_info:返回用户会话信息，若没有则返回null.
【返回】根据参数不同则类型不同

【示例】

```
Select
userenv('isdba'),userenv('Language'),userenv('Terminal'),userenv('Cl
ient_info') from dual;
```

decode(条件,值1,翻译值1,值2,翻译值2,...值n,翻译值n,缺省值)
【功能】根据条件返回相应值
【参数】c1, c2, ...,cn,字符型/数值型/日期型，必须类型相同或null
注：值1……n 不能为条件表达式,这种情况只能用case when then end解决

·含义解释：
decode(条件,值1,翻译值1,值2,翻译值2,...值n,翻译值n,缺省值)
该函数的含义如下：
IF 条件=值1 THEN
RETURN(翻译值1)
ELSIF 条件=值2 THEN
RETURN(翻译值2)
......
ELSIF 条件=值n THEN
RETURN(翻译值n)
ELSE
RETURN(缺省值)
END IF

或：
when case 条件=值1 THEN
RETURN(翻译值1)
ElseCase 条件=值2 THEN
RETURN(翻译值2)
......
ElseCase 条件=值n THEN
RETURN(翻译值n)
ELSE
RETURN(缺省值)
END

【示例】
·使用方法：
1、比较大小
select decode(sign(变量1-变量2),-1,变量1,变量2) from dual; --取较小值
sign()函数根据某个值是0、正数还是负数，分别返回0、1、-1
例如：
变量1=10，变量2=20
则sign(变量1-变量2)返回-1，decode解码结果为“变量1”，达到了取较小值的目的。

2、表、视图结构转化
现有一个商品销售表sale，表结构为：
month　　char(6)　　--月份
sell　　number(10,2)　　--月销售金额
现有数据为：
200001　1000

```
200002    1100
200003    1200
200004    1300
200005    1400
200006    1500
200007    1600
200101    1100
200202    1200
200301    1300
```

想要转化为以下结构的数据：

```
year      char(4)          --年份
month1    number(10,2)      --1月销售金额
month2    number(10,2)      --2月销售金额
month3    number(10,2)      --3月销售金额
month4    number(10,2)      --4月销售金额
month5    number(10,2)      --5月销售金额
month6    number(10,2)      --6月销售金额
month7    number(10,2)      --7月销售金额
month8    number(10,2)      --8月销售金额
month9    number(10,2)      --9月销售金额
month10    number(10,2)      --10月销售金额
month11    number(10,2)      --11月销售金额
month12    number(10,2)      --12月销售金额
```

结构转化的SQL语句为：

```
create or replace view
v_sale(year,month1,month2,month3,month4,month5,month6,
month7,month8,month9,month10,month11,month12)
as
select
substrb(month,1,4),
sum(decode(substrb(month,5,2),'01',sell,0)),
sum(decode(substrb(month,5,2),'02',sell,0)),
sum(decode(substrb(month,5,2),'03',sell,0)),
sum(decode(substrb(month,5,2),'04',sell,0)),
sum(decode(substrb(month,5,2),'05',sell,0)),
sum(decode(substrb(month,5,2),'06',sell,0)),
sum(decode(substrb(month,5,2),'07',sell,0)),
sum(decode(substrb(month,5,2),'08',sell,0)),
sum(decode(substrb(month,5,2),'09',sell,0)),
sum(decode(substrb(month,5,2),'10',sell,0)),
sum(decode(substrb(month,5,2),'11',sell,0)),
sum(decode(substrb(month,5,2),'12',sell,0))
from sale
group by substrb(month,1,4);
```

【语法】NULLIF (expr1, expr2)
【功能】expr1和expr2相等返回NULL，不相等返回expr1

COALESCE(c1, c2, ...,cn)
【功能】返回列表中第一个非空的表达式，如果所有表达式都为空值则返回1个空值
【参数】c1, c2, ...,cn,字符型/数值型/日期型，必须类型相同或null
【返回】同参数类型
【说明】从Oracle 9i版开始，COALESCE函数在很多情况下就成为替代CASE语句的一条捷径
【示例】

```
select COALESCE(null,3*5,44) hz from dual; 返回15
select COALESCE(0,3*5,44) hz from dual; 返回0
select COALESCE(null,'','AAA') hz from dual; 返回AAA
select COALESCE('','AAA') hz from dual; 返回AAA
```

rownum
【功能】返回当前行号
【参数】无

【返回】数值型

BFILENAME(dir,file)
【功能】函数返回一个空的BFILE位置值指示符，函数用于初始化BFILE变量或者是BFILE列。
【参数】dir是一个directory类型的对象，file为一文件名。

```
insert into lobdemo(key,bfile_col) values
(-1,biflename('utils','file1'));
```

VSIZE(X)
【功能】返回X的大小(字节)数
【参数】x

```
select vsize(user),user from dual;
返回：6 asdied


   select length('adfad合理') "bytesLengthIs" from dual --7

   select lengthb('adfad') "bytesLengthIs" from dual --5

   select lengthb('adfad合理') "bytesLengthIs" from dual --9

   select vsize('adfad合理') "bytesLengthIs" from dual --9

   select lengthc('adfad合理')"bytesLengthIs" from dual --7


   lengthb=vsize

   lengthc=length
```

```
case [<表达式>]
when <表达式条件值1> then <满足条件时返回值1>
[when <表达式条件值2> then <满足条件时返回值2>
……
[else  <不满足上述条件时返回值>]]
end
```

【功能】当：<表达式>=<表达式条件值1……n> 时，返回对应 <满足条件时返回值1……n>
当<表达式条件值1……n>不为条件表达式时，与函数decode()相同，
decode(<表达式>,<表达式条件值1>,<满足条件时返回值1>,<表达式条件值2>,<满足条件时返回值2> ……,<不满足上述条件时返回值>)

【参数】
<表达式> 默认为true (逻辑型)
<表达式条件值1……n> 类型要与<表达式>类型一致，
若<表达式>为字符型，则<表达式条件值1……n>也要为字符型

【注意点】
1、以CASE开头，以END结尾
2、分支中WHEN 后跟条件，THEN为显示结果
3、ELSE 为除此之外的默认情况，类似于高级语言程序中switch case的default，可以不加
4、END 后跟别名
5、只返回第一个符合条件的值,剩下的when部分将会被自动忽略，得注意条件先后顺序

【示例】
建立环境：

```
create table xqb
(xqn number(1,0));
insert into xqb xqn values(1);
insert into xqb xqn values(2);
insert into xqb xqn values(3);
insert into xqb xqn values(4);
insert into xqb xqn values(5);
insert into xqb xqn values(6);
insert into xqb xqn values(7);
commit;
```

查询结果：

```
SELECT xqn,
       CASE
          WHEN xqn = 1  THEN '星期一'
          WHEN xqn = 2  THEN '星期二'
          WHEN xqn = 3  THEN '星期三'
          else '星期三以后'
       END 星期
FROM xqb
```

另类写法

```
SELECT xqn,
```

```
       CASE xqn
          WHEN 1  THEN '星期一'
          WHEN 2  THEN '星期二'
          WHEN 3  THEN '星期三'
          else '星期三以后'
       END 星期
FROM xqb
```

decode正确表达：

```
SELECT xqn,
decode(xqn,1,'星期一',2,'星期二',3,'星期三','星期三以后') 星期
FROM xqb
```

decode错误表达：

```
SELECT xqn,
decode(TRUE,xqn=1,'星期一',xqn=2,'星期二',xqn=3,'星期三','星期三以后') 星
期
FROM xqb
```

组合条件表达：

```
SELECT xqn,
       CASE
          WHEN xqn <= 1  THEN '星期一'
          WHEN xqn <= 2  THEN '星期二'   --条件同：not(xqn<=1) and
xqn<=2
          WHEN xqn <= 3  THEN '星期三'   --条件同：not(xqn<=1 and
xqn<=2) and xqn<=3
          else '星期三以后'
       END 星期
FROM xqb
```

【语法】sys_guid()
【功能】生产32位的随机数，不过中间包括一些大写的英文字母。

【返回】长度为32位的字符串，包括0－9和大写A－F

【示例】

```
select sys_guid() from  dual
```

【语法】SYS_CONTEXT(c1,c2)
【功能】返回系统c1对应的c2的值。可以使用在SQL/PLSQL中，但不可以用在并行查询或者RAC环境中

【参数】
c1，'USERENV'
c2,参数表，详见示例

【返回】字符串

【示例】
```
select
SYS_CONTEXT('USERENV','TERMINAL') terminal,
SYS_CONTEXT('USERENV','LANGUAGE') language,
SYS_CONTEXT('USERENV','SESSIONID') sessionid,
SYS_CONTEXT('USERENV','INSTANCE') instance,
SYS_CONTEXT('USERENV','ENTRYID') entryid,
SYS_CONTEXT('USERENV','ISDBA') isdba,
SYS_CONTEXT('USERENV','NLS_TERRITORY') nls_territory,
SYS_CONTEXT('USERENV','NLS_CURRENCY') nls_currency,
SYS_CONTEXT('USERENV','NLS_CALENDAR') nls_calendar,
SYS_CONTEXT('USERENV','NLS_DATE_FORMAT') nls_date_format,
SYS_CONTEXT('USERENV','NLS_DATE_LANGUAGE') nls_date_language,
SYS_CONTEXT('USERENV','NLS_SORT') nls_sort,
SYS_CONTEXT('USERENV','CURRENT_USER') current_user,
SYS_CONTEXT('USERENV','CURRENT_USERID') current_userid,
SYS_CONTEXT('USERENV','SESSION_USER') session_user,
SYS_CONTEXT('USERENV','SESSION_USERID') session_userid,
SYS_CONTEXT('USERENV','PROXY_USER') proxy_user,
SYS_CONTEXT('USERENV','PROXY_USERID') proxy_userid,
SYS_CONTEXT('USERENV','DB_DOMAIN') db_domain,
SYS_CONTEXT('USERENV','DB_NAME') db_name,
SYS_CONTEXT('USERENV','HOST') host,
SYS_CONTEXT('USERENV','OS_USER') os_user,
SYS_CONTEXT('USERENV','EXTERNAL_NAME') external_name,
SYS_CONTEXT('USERENV','IP_ADDRESS') ip_address,
SYS_CONTEXT('USERENV','NETWORK_PROTOCOL') network_protocol,
SYS_CONTEXT('USERENV','BG_JOB_ID') bg_job_id,
SYS_CONTEXT('USERENV','FG_JOB_ID') fg_job_id,
SYS_CONTEXT('USERENV','AUTHENTICATION_TYPE') authentication_type,
SYS_CONTEXT('USERENV','AUTHENTICATION_DATA') authentication_data
from dual
```

Oracle dbms_random包的用法

from:http://space.myfarmer.cn/?action-viewthread-tid-17039

1.dbms_random.value方法

dbms_random是一个可以生成随机数值或者字符串的程序包。这个包有initialize()、seed()、terminate()、value()、normal()、random()、string()等几个函数，但value()是最常用的，value()的用法一般有两个种，第一
function value return number;
这种用法没有参数，会返回一个具有38位精度的数值，范围从0.0到1.0，但不包括1.0，如下示例：

```
SQL> set serverout on
SQL> begin
   2    for i in 1..10 loop
   3      dbms_output.put_line(round(dbms_random.value*100));
   4    end loop;
   5  end;
   6  /
46
19
45
37
33
57
61
20
82
8

PL/SQL 过程已成功完成。

SQL>
```

第二种value带有两个参数，第一个指下限，第二个指上限，将会生成下限到上限之间的数字，但不包含上限，“学无止境”兄说的就是第二种，如下：

```
SQL> begin
   2    for i in 1..10 loop
   3      dbms_output.put_line(trunc(dbms_random.value(1,101)));
   4    end loop;
   5  end;
   6  /
97
77
13
86
68
16
55
36
54
46

PL/SQL 过程已成功完成。
```

2. dbms_random.string 方法

某些用户管理程序可能需要为用户创建随机的密码。使用10G下的dbms_random.string可以实现这样的功能。

例如:

```
SQL> select dbms_random.string('P',8 ) from dual ;

DBMS_RANDOM.STRING('P',8)
----
3q<M"yf[
```

第一个参数的含义:

- 'u', 'U' - returning string in uppercase alpha characters
- 'l', 'L' - returning string in lowercase alpha characters
- 'a', 'A' - returning string in mixed case alpha characters
- 'x', 'X' - returning string in uppercase alpha-numeric characters
- 'p', 'P' - returning string in any printable characters. Otherwise the returning string is in uppercase alpha characters.

P 表示 printable，即字符串由任意可打印字符构成

而第二个参数表示返回的字符串长度。

3. dbms_random.random 方法

random返回的是BINARY_INTEGER类型值,产生一个任意大小的随机数

与dbms_random.value 的区别举例:

```
Order By dbms_random.value;
```

这条语句功能是实现记录的随机排序

另外:

dbms_random.value 和

dbms_random.random 两者之间有什么区别?

1。Order By dbms_random.value ，为结果集的每一行计算一个随机数，dbms_random.value 是结果集的一个列（虽然这个列并不在select list 中），然后根据该列排序，得到的顺序自然就是随机的啦。

2。看看desc信息便知道vlue和random这两个函数的区别了，value返回的是number类型，并且返回的值介于1和0之间，而random返回的是BINARY_INTEGER类型（以二进制形式存储的数字，据说运算的效率高于number但我没测试过，但取值范围肯定小于number，具体限制得查资料了）

如果你要实现随机排序，还是用value函数吧

4. dbms_random.normal方法

NORMAL函数返回服从正态分布的一组数。此正态分布标准偏差为1，期望值为0。这个函数返回的数值中有68%是介于-1与+1之间，95%介于-2与+2之间，99%介于-3与+3之间。

5. dbms_random.send方法

用于生成一个随机数种子,设置种子的目的是可以重复生成随机数，用于调试。否则每次不同，难以调度。

Oracle包utl_inaddr
作用:用于取得局域网或Internet环境中的主机名和IP地址.

1、utl_inaddr.get_host_address 环境中IP地址
如果查询失败，则提示系统错误
查询www.qq.com的IP地址
```
select UTL_INADDR.get_host_address('www.qq.com') from dual;
```
查询本机IP地址
```
select UTL_INADDR.get_host_address() from dual;
```
查询局域网内yuechu的IP地址
```
select UTL_INADDR.get_host_address('yuechu') from dual;
```

2、UTL_INADDR.get_host_name返回环境中主机名
返回本机主机名
```
select UTL_INADDR.get_host_name() from dual;
```
返回局域网内指定IP地址的主机名
```
select UTL_INADDR.get_host_name('192.168.0.156') from dual;
```
返回intrenet中指定IP地址的网址
```
select UTL_INADDR.get_host_name('219.153.50.84') from dual;
```