Oracle 内置 SQL 函数－分类整理大全

F.1 字符函数——返回字符值

这些函数全都接收的是字符族类型的参数(CHR 除外)并且返回字符值.除了特别说明的之外,这些函数大部分返回 VARCHAR2 类型的数值.字符函数的返回类型所受的限制和基本数据库类型所受的限制是相同的,比如: VARCHAR2 数值被限制为 2000 字符(ORACLE 8 中为 4000 字符),而 CHAR 数值被限制为 255 字符(在 ORACLE8 中是 2000).当在过程性语句中使用时,它们可以被赋值给 VARCHAR2 或者 CHAR 类型的 PL/SQL 变量.

- **CHR**

  语法: chr(x) **给出整数，返回对应的字符。**

  功能：返回在数据库字符集中与 X 拥有等价数值的字符。CHR 和 ASCII 是一对反函数。经过 CHR 转换后的字符再经过 ASCII 转换又得到了原来的字符。

  使用位置：过程性语句和 SQL 语句。

- **CONCAT**

  语法： CONCAT（string1,string2）**连接字符串**

  功能：返回 string1，并且在后面连接 string2。

  使用位置：过程性语句和 SQL 语句。

- **INITCAP**

  语法：INITCAP（string）

  功能：返回字符串的**每个单词的第一个字母大写而单词中的其他字母小写**的 string。单词是用.空格或给字母数字字符进行分隔。不是字母的字符不变动。

  使用位置：过程性语句和 SQL 语句。

- **LTRIM**

  语法：LTRIM（string1,string2）

  功能：返回删除从左边算起出现在 string2 中的字符的 string1。String2 被缺省设置为单个的空格。数据库将扫描 string1，从最左边开始。当遇到不在 string2 中的第一个字符，结果就被返回了。LTRIM 的行为方式与 RTRIM 很相似。

  使用位置：过程性语句和 SQL 语句。

- **NLS_INITCAP**

  语法：NLS_INITCAP（string[,nlsparams]）

  功能:返回字符串每个单词第一个字母大写而单词中的其他字母小写的 string，nlsparams 指定了不同于该会话缺省值的不同排序序列。如果不指定参数，则功能和 INITCAP 相同。Nlsparams 可以使用的形式是：

  ‘NLS_SORT=sort’

  这里 **sort** 制订了一个语言排序序列。

  使用位置：过程性语句和 SQL 语句。

- **NLS_LOWER**

语法：NLS_LOWER（string[,nlsparams]）

功能：返回字符串中的所有字母都是小写形式的 string。不是字母的字符不变。Nlsparams 参数的形式与用途和 **NLS_INITCAP** 中的 **nlsparams** 参数是相同的。如果 **nlsparams** 没有被包含，那么 **NLS_LOWER** 所作的处理和 **LOWER** 相同。

使用位置：过程性语句和 SQL 语句。

- **NLS_UPPER**

语法：nls_upper（string[,nlsparams]）

功能：返回字符串中的所有字母都是大写的形式的 string。不是字母的字符不变。nlsparams 参数的形式与用途和 **NLS_INITCAP** 中的相同。如果没有设定参数，则 NLS_UPPER 功能和 UPPER 相同。

使用位置：过程性语句和 SQL 语句。

- **REPLACE**

语法：REPLACE（string，search_str[,replace_str]）

功能：把 string 中的所有的子字符串 search_str 用可选的 replace_str 替换，如果没有指定 replace_str，所有的 string 中的子字符串 search_str 都将被删除。**REPLACE** 是 **TRANSLATE** 所提供的功能的一个子集。

使用位置：过程性语句和 SQL 语句。

- **RPAD/LPAD**

语法：RPAD/LPAD（string1,x[,string2]） 粘贴字符

功能：返回在 X 字符长度的位置上插入一个 string2 中的字符的 string1。如果 string2 的长度要比 X 字符少，就按照需要进行复制。如果 string2 多于 X 字符，则仅 string1 前面的 X 各字符被使用。如果没有指定 string2，那么使用空格进行填充。X 是使用显示长度可以比字符串的实际长度要长。RPAD 的行为方式与 LPAD 很相似，除了它是在右边而不是在左边进行填充。

使用位置：过程性语句和 SQL 语句。

- **RTRIM/LTRIM**

语法： RTRIM/LTRIM（string1,[,string2]）删除字符两边出现的空格

Trim(String):减掉前面和后面的字符。

功能： 返回删除从右边算起出现在 string1 中出现的字符 string2. string2 被缺省设置为单个的空格.数据库将扫描 string1,从右边开始.当遇到不在 string2 中的第一个字符,结果就被返回了 **RTRIM** 的行为方式与 **LTRIM** 很相似.

使用位置：过程性语句和 SQL 语句。

- **SOUNDEX**

语法： SOUNDEX（string）

功能： 返回一个与给定的字符串读音相同的字符串

使用位置：过程性语句和 SQL 语句。

- **SUBSTR**

语法： SUBSTR（string,a[,b]）取字符串，从 a 开始，取 b 个， 下标 a 从 1 开始

功能: 返回从字母为值 a 开始 b 个字符长的 string 的一个子字符串.如果 a 是 0,那么它就被认为从第一个字符开始.如果是正数,返回字符是从左边向右边进行计算的.如果 b 是负数,那么返回的字符是从 string 的末尾开始从右向左进行计算的.如果 b 不存在,那么它将缺省的设置为整个字符串.如果 b 小于 1,那么将返回 NULL.如果 a 或 b 使用了浮点数,那么该数值将在处理进行以前首先被却为一个整数.

使用位置: 过程性语句和 SQL 语句。

- TRANSLATE

语法: TRANSLATE(string,from_str,to_str)

功能: 返回将所出现的 from_str 中的每个字符替换为 to_str 中的相应字符以后的 string. TRANSLATE 是 REPLACE 所提供的功能的一个超集.如果 from_str 比 to_str 长,那么在 from_str 中而不在 to_str 中而外的字符将从 string 中被删除,因为它们没有相应的替换字符. to_str 不能为空.Oracle 把空字符串认为是 NULL,并且如果 TRANSLATE 中的任何参数为 NULL,那么结果也是 NULL.

使用位置: 过程性语句和 SQL 语句。

- **UPPER**

语法: UPPER（string）

功能: 返回大写的 string.不是字母的字符不变.如果 string 是 CHAR 数据类型的,那么结果也是 CHAR 类型的.如果 string 是 VARCHAR2 类型的,那么结果也是 VARCHAR2 类型的.

使用位置: 过程性语句和 SQL 语句。

F.2 字符函数——返回数字

这些函数接受字符参数回数字结果.参数可以是 CHAR 或者是 VARCHAR2 类型的.尽管实际下许多结果都是整数值,但是返回结果都是简单的 NUMBER 类型的,没有定义任何的精度或刻度范围.

- **ASCII**

语法: ASCII（string）

功能: 数据库字符集返回 string 的第一个字节的十进制表示.请注意该函数仍然称作为 ASCII. 尽管许多字符集不是 7 位 ASCII.CHR 和 ASCII 是互为相反的函数.CHR 得到给定字符编码的响应字符. ASCII 得到给定字符的字符编码.

返回与指定的字符对应的十进制数。

使用位置: 过程性语句和 SQL 语句。

- **INSTR**

语法: INSTR（string1, string2[a,b]）

功能: 得到在 **string1** 中包含 string2 的位置. string1 时从左边开始检查的,开始的位置为 a,如果 a 是一个负数,那么 string1 是从右边开始进行扫描的.第 b 次出现的位置将被返回. a 和 b 都缺省设置为 1,这将会返回在 string1 中第一次出现 string2 的位置.如果 string2 在 a 和 b 的规定下没有找到,那么返回 0.位置的计算是相对于 string1 的开始位置的,不管 a 和 b 的取值是多少.

使用位置: 过程性语句和 SQL 语句。

- **INSTRB**

语法: **INSTRB（string1, string2[a,[b]]）**

功能: 和 INSTR 相同,只是操作的对参数字符使用的位置的是字节.

使用位置: 过程性语句和 SQL 语句。

- **LENGTH**

语法: **LENGTH（string）**

功能: 返回字符串的长度.CHAR 数值是填充空格类型的,如果 string 由数据类型 CHAR,它的结尾的空格都被计算到字符串长度中间.如果 string 是 NULL,返回结果是 NULL,而不是 0.

使用位置: 过程性语句和 SQL 语句。

- **LENGTHB**

语法: **LENGTHB（string）**

功能: 返回以字节为单位的 string 的长度.对于单字节字符集 LENGTHB 和 LENGTH 是一样的.

使用位置: 过程性语句和 SQL 语句。

- **NLSSORT**

语法: NLSSORT（string[,nlsparams]）

功能: 得到用于排序 string 的字符串字节.所有的数值都被转换为字节字符串,这样在不同数据库之间就保持了一致性. Nlsparams 的作用和 NLS_INITCAP 中的相同.如果忽略参数,会话使用缺省排序.

使用位置: 过程性语句和 SQL 语句。

F.3 数字函数

函数接受 NUMBER 类型的参数并返回 NUMBER 类型的数值.超越函数和三角函数的返回值精确到 36 位.ACOS、ASIN、ATAN、ATAN2 的结果精确到 36 位.

- **ABS**

语法: ABS(x)

功能: **得到 x 的绝对值**.

使用位置: 过程性语言和 SQL 语句。

- **ACOS**

语法: ACOS(x)

功能: **返回 x 的反余弦值**. x 应该从 0 到 1 之间的数,结果在 0 到 pi 之间,以弧度为单位.

使用位置: 过程性语言和 SQL 语句。

- **ASIN**

语法: ASIN(x)

功能: **计算 x 的反正弦值**. X 的范围应该是－1 到 1 之间,返回的结果在－pi/2 到 pi/2 之间,以弧度为单位.

使用位置: 过程性语言和 SQL 语句。

- **ATAN**

语法: ATAN(x)
功能: **计算 x 的反正切值**.返回值在－pi/2 到 pi/2 之间,单位是弧度.
使用位置: 过程性语言和 SQL 语句。

- **ATAN2**

语法: ATAN2(x,y)
功能: 计算 x 和 y 的反正切值.结果在负的 pi/2 到正的 pi/2 之间,单位是弧度.
使用位置: 过程性语言和 SQL 语句。

- **CEIL**

语法: CEIL(x)
功能: 计算大于或等于 x 的最小整数值.
使用位置: 过程性语言和 SQL 语句。

- **COS**

语法: COS(x)
功能: 返回 x 的余弦值. X 的单位是弧度.
使用位置: 过程性语言和 SQL 语句。

- **COSH**

语法: COSH(x)
功能: 计算 x 的双曲余弦值.

- **EXP**

语法: EXP(x)
功能: 计算 e 的 x 次幂. e 为自然对数,约等于 2.71828.
使用位置: 过程性语言和 SQL 语句。

- **FLOOR**

语法: FLOOR(x)
功能: 返回小于等于 x 的最大整数值.
使用位置: 过程性语言和 SQL 语句。

- **LN**

语法: LN(x)
功能: 返回 x 的自然对数. x 必须是正数,并且大于 0
使用位置: 过程性语言和 SQL 语句。

- **LOG**

语法: LOG(x)

功能: 计算以 x 为底的 y 的对数.底必须大于 0 而且不等于 1, y 为任意正数.
使用位置: 过程性语言和 SQL 语句。

- **MOD**

语法: MOD(x,y)
功能: 返回 x 除以 y 的余数.如果 y 是 0,则返回 x
使用位置: 过程性语言和 SQL 语句。

- **POWER**

语法: POWER(x,y)
功能: 计算 x 的 y 次幂.
使用位置: 过程性语言和 SQL 语句。

- **ROUND**

语法: ROUND(x[,y])
功能: 计算保留到小数点右边 y 位的 x 值. y 缺省设置为 0,这会将 x 保留为最接近的整数.
如果 y 小于 0,保留到小数点左边相应的位. Y 必须是整数.
使用位置: 过程性语言和 SQL 语句。

- **SIGN**

语法: SIGN(x)
功能: 获得 x 的符号位标志.如果 x<0 返回－1.如果 x=0 返回 0.如果 x>0 返回 1.

```
select sign(300) from dual;
select sign(0) from dual;
select sign(-300) from dual;
```

使用位置: 过程性语言和 SQL 语句。

- **SIN**

语法:SIN(x)
功能:计算 x 的正弦值. X 是一个以弧度表示的角度.
使用位置: 过程性语言和 SQL 语句。

- **SINH**

语法:SINH(x)
功能:返回 x 的双曲正弦值.
使用位置: 过程性语言和 SQL 语句。

- **SQRT**

语法: SQRT(x)
功能: 返回 x 的平方根. x 必须是正数.
使用位置: 过程性语言和 SQL 语句。

- **TAN**

语法: TAN(x)

功能: 计算 x 的正切值, x 是一个以弧度位单位的角度.
使用位置: 过程性语言和 SQL 语句。

TANH
语法: TANH(x)
功能: 计算 x 的双曲正切值.
使用位置: 过程性语言和 SQL 语句。

- **TRUNC**

语法: TRUNC(x[,y])
功能: 计算截尾到 y 位小数的 x 值. y 缺省为 0,结果变为一个整数值.如果 y 是一个负数,那么就截尾到小数点左边对应的位上.
使用位置: 过程性语言和 SQL 语句。

F.4 日期函数

日期函数接受 DATE 类型的参数.除了 MONTHS_BETWEEN 函数返回的是 NUMBER 类型的结果,所有其他的日期函数返回的都是 DATE 类型的数值.

- **ADD_MONTHS**

语法: ADD_MONTHS(d,x)
功能: 返回日期 d 加上 x 个月后的月份。x 可以是任意整数。如果结果日期中的月份所包含的天数比 d 日期中的“日”分量要少。(即相加后的结果日期中的日分量信息已经超过该月的最后一天，例如，8 月 31 日加上一个月之后得到 9 月 31 日，而 9 月只能有 30 天）返回结果月份的最后一天。
使用位置: 过程性语言和 SQL 语句。

- **LAST_DAY**

语法:LAST_DAY(d)
功能:计算包含日期的 d 的月份最后一天的日期.这个函数可以用来计算当月中剩余天数.
使用位置: 过程性语言和 SQL 语句。

- **MONTHS_BETWEEN**

语法: MONTHS_BETWEEN(date 1,date2)
功能: 计算 date 1 和 date2 之间月数.如果 date 1,date2 这两个日期中日分量信息是相同的,或者这两个日期都分别是所在月的最后一天,那么返回的结果是一个整数,否则包括一个小数,小数为富余天数除以 31.
使用位置: 过程性语言和 SQL 语句。

- **NEW_TIME**

语法: NEW_TIME(d,zone1,zone2)
功能: 计算当时区 zone1 中的日期和时间是 s 时候,返回时区 zone2 中的日期和时间. zone1 和 zone2 是字符串.
使用位置: 过程性语言和 SQL 语句。

- **NEXT_DAY**

语法: NEXT_DAY(d,string)

功能: 计算在日期 d 后满足由 string 给出的条件的第一天. String 使用位置;当前会话的语言指定了一周中的某一天.返回值的时间分量与 d 的时间分量是相同的. String 的内容可以忽略大小写.

使用位置: 过程性语言和 SQL 语句。

- **ROUND**

语法: ROUND(d[,format])

功能: 将日期d按照由format指定的格式进行处理.如果没有给format则使用缺省设置`DD`.

使用位置: 过程性语言和 SQL 语句。

- **SYSDATE**

语法: SYSDATE

功能: 取得当前的日期和时间,类型是 DATE.它没有参数.但在分布式 SQL 语句中使用时,SYSDATE 返回本地数据库的日期和时间.

使用位置: 过程性语言和 SQL 语句。

- **TRUNC**

语法: TRUNC(d,format)

功能: 计算截尾到由 format 指定单位的日期 d.可以使用位置:格式和效果.缺省参数同 ROUNG.

使用位置: 过程性语言和 SQL 语句。

F.5 转 换 函 数

转换函数用于在PL/SQL数据类型之间进行转换.PL/SQL尽可能地自动进行转换,也就是采用隐含方式转换.隐含转换无法转换格式信息,并且有些类型的数据之间不支持隐含转换,所以对这些可以采用显示转换.使用显示转换也是一种好的编程习惯

.

- **CHARTOROWID**

语法: CHARTOROWID(string)

功能: 把包含外部格式的 ROWID 的 CHAR 或 VARCHAR2 数值转换为内部的二进制格式.参数 string 必须是包含外部格式的 ROWID 的 18 字符的字符串.oracle7 和 oracle8 中的外部格式是不同的.CHARTOROWID 是 ROWIDTOCHAR 的反函数.

使用位置: 过程性语言和 SQL 语句。

- **CONVERT**

语法: CONVERT(string,dest_set[,source_set])

功能: 将字符串 string 从 source_set 所表示的字符集转换为由 dest_set 所表示的字符集.如果 source_set 没有被指定,它缺省的被设置为数据库的字符集.

使用位置: 过程性语言和 SQL 语句。

- **HEXTORAW**

语法: **HEXTORAW(string)**

功能: 将由 string 表示的二进制数值转换为一个 RAW 数值. String 应该包含一个十六进制的数值. String 中的每两个字符表示了结果 RAW 中的一个字节..HEXTORAW 和 RAWTOHEX 为相反的两个函数.

使用位置: 过程性语言和 SQL 语句。

- **RAWTOHEX**

语法: **RAWTOHEX(rawvalue)**

功能: 将 RAW 类数值 rawvalue 转换为一个相应的十六进制表示的字符串. rawvalue 中的每个字节都被转换为一个双字节的字符串. RAWTOHEX 和 HEXTORAW 是两个相反的函数.

使用位置: 过程性语言和 SQL 语句。

- **ROWIDTOCHAR**

语法: ROWIDTOCHAR(rowid)

功能: 将 ROWID 类型的数值 rowid 转换为其外部的 18 字符的字符串表示,在 oracle7 和 oracle8 之间有些不一样的地方. ROWIDTOCHAR 和 CHARTOROWID 是两个相反的函数.

使用位置: 过程性语言和 SQL 语句。

- **TO_CHAR(dates)**

语法: TO_CHAR(d [,format[,nlsparams]])

功能: 将日期 d 转换为一个 VARCHAR2 类型的字符串.如果指定了 format,那么就使用位置:它控制结果的方式.格式字符串是由格式元素构成的.第一个元素返回日期数值一个部份,例如日子.如果没有给定 format,使用的就是该会话的缺省日期格式.如果指定了 nlsparams,它就控制着返回字符串的月份和日分量信息所使用的语言. nlsparams 的格式是:

"NLS_DATE_LANGUAGE"

使用位置: 过程性语言和 SQL 语句。

- **TO_CHAR(labels)**

语法: **TO_CHAR(labels[,**format**])**

功能: 将 MISLABEL 的 LABEL 转换为一个 VARCHAR2 类型的变量.

to_char(round(cb.prem/10000,2),'**fm999999990.00**'))

round(char,i):小数点后第 i 位开始四舍五入.

fm：除空格。

9999999.0099：允许小数点左边最大正数为 7 位，小数点右边最少 2 位，最多 4 位，且在第 5 位进行四舍五入

使用位置: 在 trusted 数据库的过程性语句和 SQL 语句。

- **TO_CHAR(numbers)**

语法: TO_CHAR(num[,format[,nlsparams]])

功能: 将 NUMBER 类型的参数 num 转换为一个 VARCHAR2 类型的变量.如果指定了 format,那么它会控制这个转换处理.表 5-5 列除了可以使用的数字格式.如果没有指定 format,它会控制这个转换过程.下面列出了可以使用的数字格式.如果没有指定 format,那么结果字符串将包含和 num 中有效位的个数相同的字符. nlsparams 用来指定小数点和千分位分隔符和货币符号.可以使用的格式:

`NLS_NUMERIC_CHARS=”dg”NLS_CURRENCY=”string”

d 和 g 分别表示列小数点和千分位分隔符. String 表示了货币的符号.例如,在美国小数点分隔符通常是一个句点(.),分组分隔符通常是一个逗号(,),而千分位符号通常是一个 $.

使用位置: 过程性语言和 SQL 语句。

- **TO_DATE**

语法: TO_DATE(String[,format[,nlsparams]])

功能: 把 CHAR 或者 VARCHAR2 类型的 String 转换为一个 DATE 类型的变量. format 是一个日期格式字符串.当不指定 format 的时候,使用该会话的缺省日期格式.

使用位置: 过程性语言和 SQL 语句。

- **TO_LABEL**

语法: TO_LABEL(String[,format])

功能: 将 String 转换为一个 MLSLABEL 类型的变量. String 可以是 VARCHAR2 或者 CHAR 类型的参数.如果指定了 format,那么它就会被用在转换中.如果没有指定 format,那么使用缺省的转换格式.

使用位置: 过程性语言和 SQL 语句。

- **TO_MULTI_BYTE**

语法: TO_MULTI_BYTE(String)

功能: 计算所有单字节字符都替位换位等价的多字节字符的 String.该函数只有当数据库字符集同时包含多字节和单字节的字符的时候有效.否则, String 不会进行任何处理. TO_MULTI_BYTE 和 TO_SINGLE_BYTE 是相反的两个函数.

使用位置: 过程性语言和 SQL 语句。

- **TO_NUMBER**

语法: TO_NUMBER(String[,format[,nlsparams]])

功能: 将 CHAR 或者 VARCHAR2 类型的 String 转换为一个 NUMBER 类型的数值.如果指定了 format,那么 String 应该遵循相应的数字格式. Nlsparams 的行为方式和 TO_CHAR 中的完全相同.TO_NUMBER 和 TO_CHAR 是两个相反的函数.

```
select to_number(300,'999.99') from dual;
```

使用位置: 过程性语言和 SQL 语句。

- **TO_SINGLE_BYTE**

语法: TO_SINGLE_BYTE(String)

功能: 计算 String 中所有多字节字符都替换为等价的单字节字符.该函数只有当数据库字符集同时包含多字节和单字节的字符的时候有效.否则, String 不会进行任何处理.

TO_MULTI_BYTE 和 TO_SINGLE_BYTE 是相反的两个函数.

使用位置: 过程性语言和 SQL 语句。

F.6 分 组 函 数

分组函数返回基于多个行的单一结果,这和单行函数正好形成对比,后者是对单行返回一个结果.这些函数仅仅对于查询的选择列表和 GROUP BY 子句有效.

这些函数大都可以接受对参数的修饰符.可以使用位置:的修饰符有 DISTINCT 和 ALL.如果使用位置:了 DISTINCT 修饰符,那么在处理中仅仅会考虑由查询返回的不同的取值.ALL 修饰符会使得该函数考虑由查询返回的所有数值.如果没有指定任何修饰符,那么缺省使用位置:的是 ALL 修饰符.

- **AVG**

语法: AVG([DISTINCT| ALL]col)

功能: 返回一列数据的平均值.

使用位置: 查询列表和 GROUP BY 子句.

- **COUNT**

语法: **COUNT(*| [DISTINCT| ALL] col)**

功能: 得到查询中行的数目.如果使用了*获得行的总数.如果在参数中传递的是选择列表,那么计算的是非空数值.

- 获得由 label 界定的最大下界.函数仅用于 trusted oracle.**GLB**

语法: **GLB ([DISTINCT| ALL]label)**

功能: 获得由 label 界定的最大下界.函数仅用于 trusted oracle.

使用位置:trusted 数据库的选择列表和 GROUP BY 子句.

- **LUB**

语法: **LUB ([DISTINCT| ALL]label)**

功能: 获得由 label 界定的最小上界.用于 trusted oracle.数据库.

使用位置: trusted 数据库的选择列表和 GROUP BY 子句.
过程性语言和 SQL 语句。

- **MAX**

语法: **MAX([DISTINCT| ALL]col)**

功能: 获得选择列表项目的最大值.

使用位置: 仅用于查询选择和 GROUP BY 子句.

- **MIN**

语法: MIN([DISTINCT| ALL]col)
功能: 获得选择列表的最小值.
使用位置: 仅用于查询选择和 GROUP BY 子句.

- **STDDEV**

语法: STDDEV([DISTINCT| ALL]col)
功能: 获得选择列表的标准差.标准差为方差的平方根.
使用位置: 仅用于查询选择和 GROUP BY 子句.

- **SUM**

语法:SUM**([DISTINCT| ALL]col)**
功能:返回选择的数值和总和
使用位置: 仅用于查询选择和 GROUP BY 子句.

- **VARIANCE**

语法: VARIANCE([DISTINCT| ALL]col)
功能:返回选择列表项目的统计方差.
使用位置: 仅用于查询选择和 GROUP BY 子句.

F.7 其 他 函 数

- **BFILENAME**

语法: BFILENAME(directory,file_name)
功能: 获得操作系统中与物理文件 file_name 相关的 BFILE 位置指示符. directory 必须是数据字典中的 DIRECTORY 类型的对象.
使用位置: 过程性语言和 SQL 语句。

- **DECODE**

语法:

```
DECODE(base_expr,comparel,valuel,
       Compare2,value2,
       …
       default)
```

功能: 把 base_expr 与后面的每个 compare (n) 进行比较,如果匹配返回相应的 value (n) . 如果没有发生匹配,则返回 default
使用位置: 过程性语言和 SQL 语句。

- **DUMP**

语法:DUMP(expr[,number_format[,start_position][,length]])

功能:获得有关 expr 的内部表示信息的 VARCHAR2 类型的数值. number_format 指定了按照下面返回数值的基数(base):

number_format	结果
8	八进制表示
10	十进制表示
16	十六进制表示
17	单字符

默认的值是十进制.

如果指定了 start_position 和 length,那么返回从 start_position 开始的长为 length 的字节. 缺省返回全部.

数据类型按照下面规定的内部数据类型的编码作为一个数字进行返回.

代码	数据类型
1	VARCHAR2
2	NUMBER
8	LONG
12	DATE
23	RAW
69	ROWID
96	CHAR
106	MLSLABEL

使用位置: SQL 语句.

- **EMPTY_CLOB/EMPTY_BLOB**

语法: EMPTY_CLOB
EMPTY_BLOB

功能: 获得一个空的 LOB 提示符 (locator) .EMOTY_CLOB 返回一个字符指示符,而 EMPTY_BLOB 返回一个二进制指示符.

使用位置: 过程性语言**和 SQL 语句.**

- **GREATEST**

语法: GREATEST(expr1[,expr2]…)

功能: 计算参数中最大的表达式.所有表达式的比较类型以 expr1 为准.

使用位置: 过程性语言和 SQL 语句.

- **GREATEST_LB**

语法: GREATEST_LB(label1[,label2]…)

功能: 返回标签(label)列表中最大的下界.每个标签必须拥有数据类型 MLSLABEL、RAWMLSLABEL 或者是一个表因字符串文字.函数只能用于 truested oracle 库.

使用位置: 过程性语言和 SQL 语句.

- **LEAST**

语法: LEAST(expr1[,:expr2]…)
功能: 获得参数中最小的表达式.
使用位置: 过程性语言和 SQL 语句.

- **LEAST_UB**

语法: LEAST_UB(label1[,label2]…)
功能: 与 GREATEST_UB 函数相似,本函数返回标签列表的最小上界.
使用位置: 过程性语言和 SQL 语句.

- **NVL**

语法: NVL (expr1, expr2)
功能: 如果 expr1 是 NULL,那么返回 expr2,否则返回 expr1.
如果 expr1 不是字符串,那么返回值的数据类型和 expr1 是相同的,否则,返回值的数据类型是 VARCHAR2.此函数对于检查并确定查询的活动集不包含 NULL 值十分有用.
通过查询获得某个字段的合计值，如果这个值为 null 将给出一个预设的默认值。

```
任何一个值与null进行运算结果都是null；3+null为空；
--判断一个字段是否为空：
select 1 from dual where null is null;--返回为1；
--Null=null返回为空。
select nvl(case when 1=0 then 1 end,0) from dual;
select nvl(decode(2,1,'函件',2,'包裹'),0) from dual;
```

使用位置: 过程性语言和 SQL 语句.

- **UID**

语法:
功能: 获得当前数据库用的惟一标识,标识是一个整数.
使用位置: 过程性语言和 SQL 语句.

- **USER**

语法:
功能: 取得当前 oracle 用户的名字,返回的结果是一个 VARCHAR2 型字符串.
使用位置: 过程性语言和 SQL 语句.

- **USERENV**

语法: USERENV(option)
功能: 根据参数 option,取得一个有关当前会话信息的 VARCHAR2 数值. Option 可以是: ENTRYID, SESSIONID, TERMINAL, ISDBA, LABLE, LANGUAGE, CLIENT_INFO, LANG, VSIZE。

SYS_CONTEXT

这两个函数用来记录连接的 session 信息，经常用于触发器中，记录客户端的连接信息（比如 IP）。

```
--查询当前会话标示符
select userenv('sessionid') from v$session
--查询Oracle Server端的字符集 ，dual是一个虚拟表，用来构成select的语法规则
```

```
select userenv('LANGUAGE') from dual;

--USERENV是一个遗留功能,这些功能是保留向后兼容性。甲骨文建议您使用SYS_CONTEXT函数使用内建的USERENV命名空间的当前功能。
--SYS_CONTEXT返回指明命名空间的参数所对应的值，这个函数可以在SQL中和PL/SQL中使用。这个函数不像USERENV一样区分大小写，该函数对于命名空间和参数名是不区分大小写的。
select SYS_CONTEXT('USERENV','language') terminal from dual
SELECT SYS_CONTEXT ('USERENV', 'sessionid')  FROM DUAL;
--可以看到和上面使用 USERENV 的时候获得的信息是一样的
```

使用位置: 过程性语言和 SQL 语句.

- **VSIZE**

语法: VSIZE(value)
功能: 获得 value 的内部表示的字节数.如果 value 是 NULL,结果是 NULL.
使用位置: 过程性语言和 SQL 语句.

wm_concat:连接字符串，可以用于将多行数据，转换为一行。

```
select wm_concat('xieyang'||'('||'英文名'||')') as nick_name from dual;

select concat('xieyang','_xieyang') from dual;
```

shopping:

u_id	goods	num
1	苹果	2
2	梨子	5
1	西瓜	4
3	葡萄	1
3	香蕉	1
1	橘子	3

```
select u_id, wmsys.wm_concat(goods || '(' || num || 斤)' ) goods_sum  from shopping group by u_id 。
```

u_id	goods_sum
1	苹果(2 斤),西瓜(4 斤),橘子(3 斤)
2	梨子(5 斤)
3	葡萄(1 斤),香蕉(1 斤)

备注:

oracle 中的 round 函数（四舍五入）（sql 也可以用）
oracle 中的 trunc 返回的是当天日期的[00:00:00]
oracle 中的 floor,floor(n)取小于等于数值 n 的最大整数（sql 也可以用）
oracle 中的 length 函数 sqlserver 中是 len
oracle 中的 datalength 函数 sqlserver 中是 datalength
oracle 中的 lob_to_char 函数 改为 convert(char,opertime,120)
oracle 中的 to_char 函数 改为 convert(char,opertime,120)
oracle 中的 substr 函数 改为 substring

```
lower(str):将字符串转化为小写;
upper(str):将字符串转化为大写;
initcap(str):将字符串首字母转化为大写;
select lower('AaBb') as aaa, upper('AaBb') as AAA, initcap('Aa
Bb') as Aaa from dual;
```

```
length(str):返回字符串的长度
lengthb(str):返回字符串的长度(按照字节)
select  length('Mart') name, length('烟台') addr, lengthb('烟台
') addrb from dual;
```

```
lpad rpad
lpad(char1,n,char2):
在字符串 char1 的左端填充字符串 char2，直至字符串总长度为 n,
char2 的默认值为空格，如果 char1 长度大于 n，则该函数返回 char1 左端的前 n
个字符。
如果输入参数值存在 NULL，则返回 NULL
select lpad(rpad('gao', 10, '$'), 17, '#') from dual; --######
#gao$$$$$$$
select lpad(rpad('gao', 10, '$'), 15, '#') from dual; --#####g
ao$$$$$$$
select lpad(rpad('gao', 10, '$'), 10, '#') from dual; --gao$$$
$$$$
select lpad(rpad('gao', 10, '$'), 8, '#') from dual; --gao$$$$
$
```

```
select lpad(rpad('gao', 5, '$'), 17, '#') from dual; --#######
#####gao$$
select lpad(rpad('gao', 2, '$'), 17, '#') from dual; --#######
########ga


-- trim ltrim rtrim
select rtrim('  abce  '), ltrim('  abce  '), trim('  abce  ')
from dual;
select trim('s' from 'string'), trim('g' from 'string'), trim(
'r' from 'string') from dual; --tring
select trim (both from ' DWEYE ') "TRIM e.g." from dual;
select trim (trailing from ' DWEYE ') "TRIM e.g." from dual; -
-  DWEYE
select trim (leading from ' DWEYE ') "TRIM e.g." from dual; --
DWEYE
select trim ('x' from 'xxxxDWEYExxxx') "TRIM e.g." from dual;
--DWEYE
select trim (both 'x' from 'xxxxDWEYExxxx') "TRIM e.g." from d
ual; --DWEYE
select trim (trailing 'x' from 'xxxxDWEYExxxx') "TRIM e.g." fr
om dual; --xxxxDWEYE
select trim (leading 'x' from 'xxxxDWEYExxxx') "TRIM e.g." fro
m dual; --DWEYExxxx
-- 'both' 参数表示同时去除字符串前后所指定的内容(默认情况下删除空格)
-- 'trailing' 参数表示去除字符串后所指定的内容(默认情况下删除空格)
-- 'leading' 参数表示去除字符串前所指定的内容(默认情况下删除空格)
-- 注意:
trim 删除指定字符的功能中，参数只允许包含一个字符，不支持多字符。可用
ltrim rtrim 来实现相应的功能
select ltrim(rtrim('   Mart Li   ', ' '), ' ') from dual; --Ma
rt Li
select ltrim(rtrim('xxxMart Lizzz', 'z'), 'x') from dual; --Ma
rt Li
```

```
select rtrim('xyxxDWEYExyyx','xy') "e.g." from dual; --xyxxDWEYE
select ltrim('xyxxDWEYExyyx','xy') "e.g." from dual; --DWEYExyyx
-- 使用RTRIM和LTRIM函数时的注意事项：
-- “xy”不表示整个“xy”字符串进行匹配，而是发现任意的字符“x”或字符“y”均做删除操作


-- substr(src,start,length) 截取字符串，下标从1开始
select substr('1234567890', 3, 5) aaa from dual; --34567
select substr('1234567890', 0, length('1234567890')-1) from dual; --123456789
select substr('1234567890', 1, length('1234567890')-1) from dual; --123456789


-- replace('String','oldstr','newstr') 字符串替换
String    用于指定字符串
oldstr    用于指定要被替换的子串
newstr    用于指定替换后的子串
如果 oldstr 为 NULL，则返回原有字符串，如果 newstr 为 NULL，则会去掉指定子串。


-- count(column_name)  统计一个表中 column_name 列的数量
select count(column_name) from authors;
-- 如果相同的 column_name 出现了不止一次，则会被计算多次。
可以通过关键字 distinct 得到不同 column_name 的数目：
select count(distinct column_name) from authors
-- 如果相同的column出现了不止一次，它将只被计算一次。
count(column_name),如果某一行的 column_name 是null，那么这一行将不计算，所以count(column_name)是计算所有值不为NULL的数据。
即count(column_name) 就只会统计非空行.
```

count(*) 和 count('x'),count('y'),count(1),count(2) 这 3 个都代表所有行,因为每行数据都不为 null

补充，在表的主键不同情况下，count(*)和 count(数字)的执行效率是不同的:

count(字段)时，如果字段为主键，则效率最高，count(1)次之，count(*)最慢;

如果字段不为主键，则 count(1)最快，count(*)次之，count(字段)最慢;

```
-- abs(n)
```

用于返回数字 n 的绝对值，如果输入为 NULL，则返回值也是 NULL

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

```
-- ceil(n)
```

返回大于等于数字 n 的最小整数，若输入 NULL，则返回为 NULL

```
select ceil(3.1415927) from dual; --4
```

```
-- floor(n)
```

返回小于等于数字 n 的最大整数，若输入 NULL，则返回 NULL

```
select floor(2345.67) from dual; --2345
```

```
-- sign(n) 用于检测数字的正负
```

若 n<0; 则返回-1;

若 n=0; 则返回 0;

若 n>0; 则返回 1;

若输入值为 NULL，则返回 NULL

```
select sign(123), sign(-100), sign(0) from dual; -- 1 -1 0
```

```
-- mod(n1,n2)
```

返回 n1 除以 n2 的余数

```
select mod(10, 3), mod(3, 3), mod(2, 3) from dual; --1 0 2
```

```
-- roundn(n[,m]) truncn(n[,m]) cut
```

round(n[,m]): 返回 n 的四舍五入值,其中 n 可为任意数字，m 必须为整数。

若省略 m 或者 m=0,则四舍五入到整数位;

若 m<0, 则四舍五入到小数点前的第 m 位;

若 m>0, 则四舍五入到小数点后的第 m 位;

若输入 NULL, 则输出 NULL.

trunc(n[,m]): 该函数用于截取数字,其中 n 可以是任意数字, m 必须是整数。

若 m 省略，则会将数字 n 的小数部分截去;

若 m>0, 则将数字 n 截取至小数点后的第 m 位;

若 m<0, 则将数字 n 截取至小数点前的第 m 位.

```
select round(55.6), round(-55.4), trunc(55.6), trunc(-55.4) fr
om dual; --56 -55 55 -55
```

round(d[,fmt]):返回日期时间的四舍五入结果。

参数 d 用于指定日期时间值，参数 fmt 用于指定四舍五入的方式。

如果设置 fmt 为 YEAR，则 7 月 1 日为分界线;

如果设置 fmt 为 MONTH，则 16 日为分界线;

如果设置 fmt 为 DAY，则中午 12：00 时为分界线。

trunc(d[,fmt]): 用于截断日期时间数据。fmt 用于指定截断日期时间数据的方法。

如果设置 fmt 为 YEAR，则结果为本年度的 1 月 1 日;

如果设置 fmt 为 MONTH，则结果为本月 1 日。

```
select round(sysdate,'YEAR') from dual; --2012-1-1 (today is 2
011-07-01)
select round(sysdate,'YEAR')-1 from dual; --2011-12-31(today i
s 2011-07-01)
select trunc(sysdate,'YEAR') from dual; --2011-1-1(today is 20
11-07-27)
select trunc(124.1666, -2) trunc1, trunc(124.16666, -1) trunc2
 from dual; --100 120
select trunc(124.1666, 0) trunc1, trunc(124.16666, 2) trunc2 f
rom dual; --124 124.16
```

```
-- add_months(d,n)
```

返回特定日期时间之后或之前的几个月所对应的日期时间

d:给定的日期时间;

n:可以是任意整数; 当 n<0 时,返回 d 之前 n 个月对应的日期时间;当 n>0 时,返回 d 之后 n 个月对应的日期时间.

```
select to_char(add_months(to_date('199912','yyyymm'),2),'yyyym
m') from dual;
select to_char(add_months(to_date('199912','yyyymm'),-2),'yyyy
mm') from dual;
select to_char(add_months(to_date('2010-12', 'yyyy-mm'), 2), '
yyyy-mm')
  from dual; --2011-02
select to_char(add_months(to_date('2010-12', 'yyyy-mm'), -2),
'yyyy-mm')
  from dual; --2010-10
```

```
-- last_day(d)
```

返回特定日期所在月份的最后一天，参数 d 表示给定日期。

```
select to_char(sysdate, 'yyyy-mm-dd'), to_char((sysdate) + 1,
'yyyy-mm-dd') from dual; --2011-07-01 2011-07-02
select to_char(last_day(sysdate), 'yyyy-mm-dd') from dual; --2
011-07-31
```

```
-- months_between(d1,d2) 返回日期d1 和d2 之间相差的月数;
```

如果 d1 小于 d2,则返回负数;

如果 d1 和 d2 的天数相同或都是月底,则返回整数;

否则 Oracle 以每月 31 天为准，计算结果的小数部分。

```
select months_between('19-12 月-2010', '19-3 月
-2010') mon_between from dual; --9
select months_between(to_date('2010-03-19', 'yyyy-mm-dd'), to_
date('2010-12-19', 'yyyy-mm-dd')) mon_betw from dual; -- -9
select months_between(to_date('2005.05.20', 'yyyy.mm.dd'), to_
date('2010.05.20', 'yyyy.mm.dd')) mon_betw from dual; -- -60
```

```
-- new_time(date,zone1,zone2)
select to_char(sysdate, 'yyyy.mm.dd hh24:mi:ss') bj_time,
```

```
       to_char(new_time(sysdate, 'PDT', 'GMT'), 'yyyy.mm.dd hh
24:mi:ss') los_angles
  from dual; --2011.07.01 14:45:31 2011.07.01 21:45:31


-- next_day(date,char) 返回特定日期之后的第一个工作日所对应的日期。
  date 表示给定日期时间;
  char 用于指定工作日,工作日必须与日期语言匹配。
select next_day('18-5月-2010', '星期五
') next_day from dual; --2010-5-21
select next_day(to_date('2011-07-01', 'yyyy-mm-dd'), '星期一
') next_day from dual; --2011-7-4
select next_day(to_date('2011-07-01', 'yyyy-mm-dd'), '星期二
') next_day from dual; --2011-7-5


-- sysdate current_date getdate()
sysdate: Oracle 数据库 获得系统当前时间的函数
current_date: Oracle 数据库 获得当前会话时区所对应的日期时间的函数
getdate():SQLServer 数据库 获得当前的日期和时间的函数
select getdate() from dual; --NOV 30 1997 3:29AM
select to_char(sysdate, 'yyyy-mm-dd day') from dual; --2011-07
-01 星期五


-- extract() 从给定的日期时间值中获取所需要的特定数据
select extract(year from sysdate) from dual; --2011


-- greatest least
greatest:返回一组表达式中的最大值,即比较字符的编码大小.
least:返回一组表达式中的最小值.
select greatest('AA','AB','AC') gtc from dual; --AC
select greatest('啊','安','天') gtc from dual; --天
select least('啊','安','天') ltc from dual; --啊
```

```
-- avg(distinct|all) 函数返回数值列的平均值
all:表示对所有的值求平均值;
distinct:只对不同的值求平均值
NULL 值不包括在计算中。
select avg(distinct t.data_length) from user_tab_cols t; --323
.2325
select avg(all t.data_length) from user_tab_cols t; --48.39710


-- max min
函数 MAX（）返回一个数值型字段的所有值中的最大值。
函数 MIN（）返回一个数值型字段的所有值中的最小值。
如果字段是空的，MAX() MIN()返回空值
select max(all t.data_length) from user_tab_cols t; --4000
select min(all t.data_length) from user_tab_cols t; --0


-- decode(condition, case1, value1, [case2, value2, ...,] defa
ult_value)
-- 条件判断函数
select decode('red','red','红灯','green','绿灯','黄灯
') color from dual; --红灯
select decode('green','red','红灯','green','绿灯','黄灯
') color from dual; --绿灯
select decode('yellow','red','红灯','green','绿灯','黄灯
') color from dual; --黄灯


-- NVL(expr1, expr2)
若 expr1 为 NULL，返回 expr2;
若 expr1 不为 NULL，返回 expr1;
注意两者的类型要一致
常用于 存储过程中 条件的判断
```

```
select nvl('abc','def') from dual; -- abc
select nvl(null,'Replace Null Column') ideacol from dual; --Replace Null Column
```

```
-- NVL2 (expr1, expr2, expr3)
```

若 expr1 不为 NULL，返回 expr2;

若 expr1 为 NULL，返回 expr3;

expr2 和 expr3 类型不同的话，expr3 会转换为 expr2 的类型

```
-- NULLIF (expr1, expr2)
```

相等返回 NULL，不等返回 expr1

```
-- sum(colname)  计算字段值的和
select sum(null) from dual; --(空)
select sum(t.num_rows) from user_tables t; --598441
```

函数 SUM（）的返回值代表字段中所有值的和。字段的数据类型也许是 MONEY 型，但你也可以对其它数值型字段使用函数 SUM（）。

```
-- to_char(n[,fmt]) 数字转换成字符
```

其实:to_number,to_char,to_date 等转换函数都可以在很多数据类型之间进行转换,

to_lob 一般只能将 long、long raw 转换为 clob、blob、nclob 类型

```
-- like 模糊查询
```

用到 like 关键字和通配符，有时候还得用到转义字符:

其中通配符有两个："_"代表一个字符，"%"代表零个或多个字符;

如果一个字符串中本身含有 "%"或"_",那么我们就要用转义字符, ' \ ' 就是一个转义字符，我们还可以自己定义一个转移字符:

```
select ename from emp where ename like '%$%%' escape '$';
```

其中 escape 将指明$符号为转移字符,默认情况下 oracle 把 ' \ ' 作为转移字符.

-- rownum rowid

Oracle 中有两个伪列叫做：rownum 和 rowid。

rowid 是每条记录的唯一标示，而 rownum 则是每条记录在表中从第一条记录到最后的一个排序标号;

可以通过 rownum 选出某个表中前几条记录入：

select * from emp where rownum < 5 则可以选出前五行记录

但是不能用 rownum 选出后几条记录来,如：

select * from emp where rownum > 10 后十条记录：这是错误的方法。

凡是大于或等于的都选不出来。

rownum 本身不显示在表中，它是隐藏着的，只有通过 select 语句显示的在 select 列表中注明 rownum 字段才能被选出来展示，

如子查询选出的一张新表，它同时也具有了一个伪列

--扩展: Oracle 分页的实现：

```
select ename, sal
  from (select ename, sal, rownum
      from (select ename, sal from emp order by sal desc ))
 where rownum >= 6
   and rownum <=10;
```

最后总结一句话就是 rownum 是 oracle 加在你最后取出来的结果集上的一个伪字段，它记录着每一行的行号，

而这个行号只能和小于、小于等于使用,不能和等于、大于或大于等于一起使用,

要强行使用的话只能用子查询解决。

-- REGEXP_LIKE 正则表达式

/*

ORACLE 中的支持正则表达式的函数主要有下面四个：

1，REGEXP_LIKE ：与 LIKE 的功能相似

2，REGEXP_INSTR ：与 INSTR 的功能相似

3，REGEXP_SUBSTR ：与 SUBSTR 的功能相似

4，REGEXP_REPLACE ：与 REPLACE 的功能相似

它们在用法上与 Oracle SQL 函数 LIKE、INSTR、SUBSTR 和 REPLACE 用法相同，但是它们使用 POSIX 正则表达式代替了老的百分号（%）和通配符（_）字符。

POSIX 正则表达式由标准的元字符（metacharacters）所构成：

'^' 匹配输入字符串的开始位置，在方括号表达式中使用，此时它表示不接受该字符集合。
'$' 匹配输入字符串的结尾位置。如果设置了 RegExp 对象的 Multiline 属性，则 $ 也匹配 '\n' 或 '\r'。
'.' 匹配除换行符之外的任何单字符。
'?' 匹配前面的子表达式零次或一次。
'+' 匹配前面的子表达式一次或多次。
'*' 匹配前面的子表达式零次或多次。
'|' 指明两项之间的一个选择。例子'^([a-z]+|[0-9]+)$'表示所有小写字母或数字组合成的字符串。
'( )' 标记一个子表达式的开始和结束位置。
'[]' 标记一个中括号表达式。
'{m,n}' 一个精确地出现次数范围,m=<出现次数<=n,'{m}'表示出现 m 次,'{m,}'表示至少出现 m 次。
\num 匹配 num，其中 num 是一个正整数。对所获取的匹配的引用。
字符簇:
[[:alpha:]] 任何字母。
[[:digit:]] 任何数字。
[[:alnum:]] 任何字母和数字。
[[:space:]] 任何白字符。
[[:upper:]] 任何大写字母。
[[:lower:]] 任何小写字母。
[[:punct:]] 任何标点符号。
[[:xdigit:]] 任何 16 进制的数字，相当于[0-9a-fA-F]。
各种操作符的运算优先级
\转义符
(), (?:), (?=), [] 圆括号和方括号
*, +, ?, {n}, {n,}, {n,m} 限定符
^, $, anymetacharacter 位置和顺序
|
*/
--创建表

```
create table fzq (
  id varchar(4),
  value varchar(10)
```

```
);
--数据插入
insert into fzq values ('1','1234560');
insert into fzq values ('2','1234560');
insert into fzq values ('3','1b3b560');
insert into fzq values ('4','abc');
insert into fzq values ('5','abcde');
insert into fzq values ('6','ADREasx');
insert into fzq values ('7','123 45');
insert into fzq values ('8','adc de');
insert into fzq values ('9','adc,.de');
insert into fzq values ('10','1B');
insert into fzq values ('10','abcbvbnb');
insert into fzq values ('11','11114560');
insert into fzq values ('11','11124560');
--regexp_like
--查询value 中以1 开头60 结束的记录并且长度是7 位
select * from fzq where value like '1____60';
select * from fzq where regexp_like(value,'1....60');
--查询value 中以1 开头60 结束的记录并且长度是7 位并且全部是数字的记录。
--使用like 就不是很好实现了。
select * from fzq where regexp_like(value,'1[0-9]{4}60');
--  也可以这样实现，使用字符集。
select * from fzq where regexp_like(value,'1[[:digit:]]{4}60')
;
--  查询value 中不是纯数字的记录
select * from fzq where not regexp_like(value,'^[[:digit:]]+$'
);
--  查询value 中不包含任何数字的记录。
select * from fzq where regexp_like(value,'^[^[:digit:]]+$');
--查询以12 或者1b 开头的记录.不区分大小写。
select * from fzq where regexp_like(value,'^1[2b]','i');
--查询以12 或者1b 开头的记录.区分大小写。
select * from fzq where regexp_like(value,'^1[2B]');
--  查询数据中包含空白的记录。
```

```
select * from fzq where regexp_like(value,'[[:space:]]');
--查询所有包含小写字母或者数字的记录。
select * from fzq where regexp_like(value,'^([a-z]+|[0-9]+)$')
;
--查询任何包含标点符号的记录。
select * from fzq where regexp_like(value,'[[:punct:]]');


-- uid 返回标识当前用户的唯一整数
select username,user_id from dba_users where user_id=uid;


-- user 返回当前用户的名字
select user from  dual;


sql%rowcount;
--统计上一条SQL 语句影响的行数


-- datepart() datename()
```

detepart() 用来抽取日期的特定部分

detepart() 有两个参数，第一个指定要抽取日期的哪一部分；第二个变量是实际的数据。

日期的各部分及其简写如下：

日期部分	简写	值
year	yy	1753--9999
quarter	qq	1--4
month	mm	1--12
day of year	dy	1--366
day	dd	1--31
week	wk	1--53
weekday	dw	1--7(Sunday--Saturday)
hour	hh	0--23
minute	mi	0--59

```
second              ss                  0--59
milisecond          ms                  0--999
```

函数 DATENAME()和 DATEPART()接收同样的参数;其返回值是一个字符串(数字对应的字符串);

```
--选择某个周的时间段
Select case when to_char(sysdate, 'D') = 1 then to_char(sysdat
e - to_char(sysdate, 'D') - 5, 'yyyy-mm-dd')
            else to_char(sysdate - to_char(sysdate, 'D') + 2,
'yyyy-mm-dd') end week_begin
  from dual ;
Select case when to_char(sysdate, 'D') = 1 then to_char(sysdat
e - to_char(sysdate, 'D') + 2, 'yyyy-mm-dd')
            else to_char(sysdate - to_char(sysdate, 'D') + 9,
'yyyy-mm-dd') end week_begin
  from dual ;
Select case when to_char(sysdate, 'D') = 1 then to_char(sysdat
e - to_char(sysdate, 'D') - 12, 'yyyy-mm-dd')
            else to_char(sysdate - to_char(sysdate, 'D') - 5,
'yyyy-mm-dd') end week_begin
  from dual ;
-------------------------------------------------------------
--
Select to_char(sysdate, 'D'),
     case when to_char(sysdate, 'D') = 1 then to_char(sysdate
- to_char(sysdate, 'D') + 1, 'yyyy-mm-dd')
            else to_char(sysdate - to_char(sysdate, 'D') + 8,
'yyyy-mm-dd') end week_end
  from dual ;
Select to_char(sysdate, 'D'),
     case when to_char(sysdate, 'D') = 1 then to_char(sysdate
- to_char(sysdate, 'D') + 8, 'yyyy-mm-dd')
            else to_char(sysdate - to_char(sysdate, 'D') + 15,
 'yyyy-mm-dd') end week_end
```

```
  from dual ;
Select case when to_char(sysdate, 'D') = 1 then to_char(sysdat
e - to_char(sysdate, 'D') - 6, 'yyyy-mm-dd')
            else to_char(sysdate - to_char(sysdate, 'D') + 1,
'yyyy-mm-dd') end week_end
  from dual;
```

round(no):四舍五入,精确到个位: `select round(123.45) from dual; --123`

round(no,n):四舍五入,若 n>0,精确到小数点右边第 n 位;若 n<0,则精确到小数点左边第(n+1)位;

```
select round(123.456,2) from dual; --123.46
select round(123.456,-2) from dual; --100
```

instr(C1,C2[,I[,J]]):

instrb(C1,C2[,I[,J]]):

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;

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 子串的第 J 次出现的位置,默认为 1

一般用来判断某值是否存在于某值中

```
select instr('oracle traning', 'ra', 1, 2) instring from dual;
select 'a' a from dual where instr('football,basketball,pingpa
ng', 'football') >=1;
```

```
-- trunc(date[,accuracy])截取
/*
date:日期
accuracy:精确度
accuracy='yyyy' --> 精确度为年,月日时分秒的值舍去
accuracy='MM' --> 精确度为月,日时分秒的值舍去
```

```
accuracy='dd'(或者省略) --> 精确度为日,时分秒的值舍去
accuracy='hh' --> 精确度为时,分秒的值舍去
accuracy='mi' --> 精确度为分,秒的值舍去
accuracy='yyyy' --> 报错
*/
--trunc(date[,accuracy]) 再 加上(或者减去) 任意时间，能够标示任意的日期时间值,如下:
select to_char(trunc(sysdate),'yyyy-MM-dd hh24:mi:ss') from dual;  --2012-08-03 00:00:00
select to_char(trunc(sysdate) + 10/(24*60*60),'yyyy-mm-dd hh24:mi:ss') from dual; --2012-08-03 00:00:10
select to_char(trunc(sysdate,'yyyy'),'yyyy-MM-dd hh24:mi:ss') from dual;  --2012-01-01 00:00:00
select to_char(trunc(sysdate,'yyyy') + 4/(24*60*60), 'yyyy-mm-dd hh24:mi:ss') from dual; --2012-01-01 00:00:04
select to_char(trunc(sysdate,'MM'),'yyyy-MM-dd hh24:mi:ss') from dual;  --2012-08-01 00:00:00
select to_char(trunc(sysdate,'MM') + (1+9/24), 'yyyy-mm-dd hh24:mi:ss') from dual; --2012-08-02 09:00:00
select to_char(trunc(sysdate,'dd'),'yyyy-MM-dd hh24:mi:ss') from dual;  --2012-08-03 00:00:00
select to_char(trunc(sysdate) + 5/24, 'yyyy-mm-dd hh24:mi:ss') from dual; --2012-08-03 05:00:00
select to_char(trunc(sysdate,'hh'),'yyyy-MM-dd hh24:mi:ss') from dual;  --2012-08-03 14:00:00
select to_char(trunc(sysdate,'hh') + 16/(24*60),'yyyy-mm-dd hh24:mi:ss') from dual; --2012-08-03 14:16:00
select to_char(trunc(sysdate,'mi'),'yyyy-MM-dd hh24:mi:ss') from dual;  --2012-08-03 14:26:00


-- convert(data_type(length), data, style)
把日期转换为新数据类型的通用函数;
用不同的格式显示日期/时间数据
data_type(length) 规定目标数据类型(带有可选的长度)
```

data 需要转换的值

style 规定日期/时间的输出格式;

可使用的 style 值如下:

Style_ID	Style_格式
100 or 0	mon dd yyyy hh:miAM （或者 PM）
101	mm/dd/yy
102	yy.mm.dd
103	dd/mm/yy
104	dd.mm.yy
105	dd-mm-yy
106	dd mon yy
107	Mon dd, yy
108	hh:mm:ss
109 or 9	mon dd yyyy hh:mi:ss:mmmAM（或者 PM）
110	mm-dd-yy
111	yy/mm/dd
112	yymmdd
113 or 13	dd mon yyyy hh:mm:ss:mmm(24h)
114	hh:mi:ss:mmm(24h)
120 or 20	yyyy-mm-dd hh:mi:ss(24h)
121 or 21	yyyy-mm-dd hh:mi:ss.mmm(24h)
126	yyyy-mm-ddThh:mm:ss.mmm（没有空格）
130	dd mon yyyy hh:mi:ss:mmmAM
131	dd/mm/yy hh:mi:ss:mmmAM

```
select CONVERT(VARCHAR(19),GETDATE()) from dual; -- Dec 29 2008 11:45 PM
select CONVERT(VARCHAR(10),GETDATE(),110) from dual;  -- 12-29-2008
select CONVERT(VARCHAR(11),GETDATE(),106) from dual; -- 29 Dec 08
select CONVERT(VARCHAR(24),GETDATE(),113) from dual; -- 29 Dec 2008 16:25:46.635
```

```
/* Oracle dbms_random 包用法整理 */
dbms_random.value
--返回一个具有38位精度的随机数，范围0.0 <= dbms_random.value < 1.0
SQL>select dbms_random.value from dual;

dbms_random.value(min_value, max_value)
--返回[min_value, max_value)范围内的随机数(精度38位)
SQL>select dbms_random.value(0,100) from dual;

trunc(dbms_random.value(min_value, max_value))
--返回指定范围内的整数
SQL>select trunc(dbms_random.value(0,100)) from dual;

dbms_random.random
--返回一个从 -power(2,31) 到 power(2,31) 的随机整数
SQL>select dbms_random.random from dual;

--dbms_random.random 与 dbms_random.value 区别
dbms_random.value 返回 number 类型,范围是 (0, 1] 的随机数
dbms_random.random 返回 BINARY_INTEGER 类型,任意大小的随机数

随机排序:Order By dbms_random.value;
--此语句功能是实现结果集的随机排序(dbms_random.value 是结果集的一个隐藏列,
按照该列排序,就是随机排序).

dbms_random.normal
--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;
--且返回值中有68%是介于[-1,1]之间,95%介于[-2,2]之间,99%介于[-3,3]之间)
SQL>select dbms_random.normal from dual;

substr(cast(dbms_random.value as varchar2(38)), 3, length)
--返回长度为 length 的随机数字串(3的目的是截掉整数位和小数点)
SQL>select substr(cast(dbms_random.value as varchar2(38)), 3, 20)
 from dual;
```

```
dbms_random.string(opt, length)
--返回由opt组成,长度为length的随机字符串;
--此方法可用于产生随机验证码、随机密码等;
```

opt 可取值如下:

'u','U':大写字母

'l','L':小写字母

'a','A':大,小写字母混合

'x','X':数字,大写字母混合

'p','P':任意可打印(显示)字符

如果 type 不是以上字符，则产生一个长度为 Length,随机的字母序列.

```
SQL>select dbms_random.string('U',4) FROM DUAL; -- 'u'或'U' 返回大写字母
SQL>select dbms_random.string('l',3) from dual; -- 'l'或'L' 返回小写字母
SQL>select dbms_random.string('a',9) from dual; -- 'a'或'A' 大小写字母混合
SQL>select dbms_random.string('X',6) FROM DUAL; -- 'x'或'X' 大写字母和数字混合
SQL>select dbms_random.string('p',7) from dual; -- 'p'或'P' 任意可显示字符
```