Oracle to_char函数的使用方法

Oracle to_char函数的功能是将数值型或者日期型转化为字符型，下面就为您详细介绍Oracle to_char函数的使用，希望对您能有所帮助。

Postgres 格式化函数提供一套有效的工具用于把各种数据类型（日期/时间，int，float，numeric）转换成格式化的字符串以及反过来从格式化的字符串转换成原始的数据类型。

注意：所有格式化函数的第二个参数是用于转换的模板。

表 5-7. 格式化函数

函数	返回	描述	例子
to_char(timestamp, text)	text	把 timestamp 转换成 string	**to_char**(timestamp 'now','HH12:MI:SS')
to_char(int, text)	text	把 int4/int8 转换成 string	**to_char**(125, '999')
to_char(float, text)	text	把 float4/float8 转换成 string	**to_char**(125.8, '999D9')
to_char(numeric, text)	text	把 numeric 转换成 string	**to_char**(numeric '-125.8', '999D99S')
to_date(text, text)	date	把 string 转换成 date	to_date('05 Dec 2000', 'DD Mon YYYY')
to_timestamp(text, text)	date	把 string 转换成 timestamp	to_timestamp('05 Dec 2000', 'DD Mon YYYY')
to_number(text, text)	numeric	把 string 转换成 numeric	to_number('12,454.8-', '99G999D9S')

表 5-8. 用于 date/time 转换的模板

模板	描述
HH	一天的小时数 (01-12)
HH12	一天的小时数 (01-12)
HH24	一天的小时数 (00-23)
MI	分钟 (00-59)
SS	秒 (00-59)
SSSS	午夜后的秒 (0-86399)
AM or A.M. or PM or P.M.	正午标识（大写）

am or a.m. or pm or p.m.	正午标识（小写）
Y,YYY	带逗号的年（4 和更多位）
YYYY	年（4和更多位）
YYY	年的后三位
YY	年的后两位
Y	年的最后一位
BC or B.C. or AD or A.D.	年标识（大写）
bc or b.c. or ad or a.d.	年标识（小写）
MONTH	全长大写月份名（9字符）
Month	全长混合大小写月份名（9字符）
month	全长小写月份名（9字符）
MON	大写缩写月份名（3字符）
Mon	缩写混合大小写月份名（3字符）
mon	小写缩写月份名（3字符）
MM	月份 (01-12)
DAY	全长大写日期名（9字符）
Day	全长混合大小写日期名（9字符）
day	全长小写日期名（9字符）
DY	缩写大写日期名（3字符）
Dy	缩写混合大小写日期名（3字符）
dy	缩写小写日期名（3字符）
DDD	一年里的日子(001-366)
DD	一个月里的日子(01-31)
D	一周里的日子(1-7；SUN=1)
W	一个月里的周数

WW	一年里的周数
CC	世纪（2 位）
J	Julian 日期（自公元前4712年1月1日来的日期）
Q	季度
RM	罗马数字的月份（I-XII；I=JAN） - 大写
rm	罗马数字的月份（I-XII；I=JAN） - 小写

所有模板都都允许使用前缀和后缀修改器。模板里总是允许使用修改器。前缀 'FX' 只是一个全局修改器。

表 5-9. 用于日期/时间模板 to_char() 的后缀

后缀	描述	例子
FM	填充模式前缀	FMMonth
TH	大写顺序数后缀	DDTH
th	小写顺序数后缀	DDTH
FX	固定模式全局选项（见下面）	FX Month DD Day
SP	拼写模式（还未实现）	DDSP

用法须知：

- 如果没有使用 FX 选项，to_timestamp 和 to_date 忽略空白。FX 必须做为模板里的第一个条目声明。
- 反斜杠（"\"）必须用做双反斜杠（"\\"），例如 '\\HH\\MI\\SS'。
- 双引号（'""）之间的字串被忽略并且不被分析。如果你想向输出写双引号，你必须在双引号前面放置一个双反斜杠（'\\'），例如 '\\"YYYY Month\\"'。
- to_char **支持不带前导双引号（'""）的文本，但是在双引号之间的任何字串会被迅速处理并且还保证不会被当作模板关键字解释（例如：**'"Hello Year: "YYYY'**）。**

表 5-10. 用于 to_char(numeric**) 的模板**

模板	描述
9	带有指定位数的值

9	带有指定位数的值
0	前导零的值
.（句点）	**小数**点
,（逗号）	分组（千）分隔符
PR	尖括号内负值
S	带负号的负值（使用本地化）
L	货币符号（使用本地化）
D	**小数**点（使用本地化）
G	分组分隔符（使用本地化）
MI	在指明的位置的负号（如果数字 < 0）
PL	在指明的位置的正号（如果数字 > 0）
SG	在指明的位置的正/负号
RN	罗马数字（输入在 1 和 3999 之间）
TH or th	转换成序数
V	移动 n 位（**小数**）（参阅注解）
EEEE	科学记数。现在不支持。

用法须知：

- 使用 'SG'，'PL' 或 'MI' 的带符号字并不附着在数字上面；例如，**to_char**(-12, 'S9999') 生成 ’ -12’，而 **to_char**(-12, 'MI9999') 生成 ’- 12’。**Oracle** 里的实现不允许在 9 前面使用 MI，而是要求 9 在 MI前面。
- PL，SG，和 TH 是 Postgres 扩展。
- 9 表明一个与在 9 字串里面的一样的数字位数。如果没有可用的数字，那么使用一个空白（空格）。
- TH 不转换小于零的值，也不转换**小数**。TH 是一个 Postgres 扩展。
- V 方便地把输入值乘以 10^n，这里 n 是跟在 V 后面的数字。to_char 不支持把 V 与一个**小数**点绑在一起使用（例如. "99.9V99" 是不允许的）。

表 5-11. to_char **例子**

输入	输出

to_char(now(),'Day, HH12:MI:SS')	'Tuesday , 05:39:18'
to_char(now(),'FMDay, HH12:MI:SS')	'Tuesday, 05:39:18'
to_char(-0.1,'99.99')	' -.10'
to_char(-0.1,'FM9.99')	'-.1'
to_char(0.1,'0.9')	' 0.1'
to_char(12,'9990999.9')	' 0012.0'
to_char(12,'FM9990999.9')	'0012'
to_char(485,'999')	' 485'
to_char(-485,'999')	'-485'
to_char(485,'9 9 9')	' 4 8 5'
to_char(1485,'9,999')	' 1,485'
to_char(1485,'9G999')	' 1 485'
to_char(148.5,'999.999')	' 148.500'
to_char(148.5,'999D999')	' 148,500'
to_char(3148.5,'9G999D999')	' 3 148,500'
to_char(-485,'999S')	'485-'
to_char(-485,'999MI')	'485-'
to_char(485,'999MI')	'485'
to_char(485,'PL999')	'+485'
to_char(485,'SG999')	'+485'
to_char(-485,'SG999')	'-485'
to_char(-485,'9SG99')	'4-85'
to_char(-485,'999PR')	'<485>'
to_char(485,'L999')	'DM 485

to_char(485,'RN')	’ CDLXXXV’
to_char(485,'FMRN')	’CDLXXXV’
to_char(5.2,'FMRN')	V
to_char(482,'999th')	’ 482nd’
to_char(485, '"Good number:"999')	’Good number: 485’
to_char(485.8,'"Pre-decimal:"999" Post-decimal:" .999')	’Pre-decimal: 485 Post-decimal: .800’
to_char(12,'99V999')	’ 12000’
to_char(12.4,'99V999')	’ 12400’
to_char(12.45, '99V9')	’ 125’

Oracle to_char函数最简单的应用：

```
/*1.0123--->'1.0123'*/
Select TO_CHAR(1.0123) FROM DUAL
/*123--->'123'*/
Select TO_CHAR(123) FROM DUAL
```

接下来再看看下面：

```
/*0.123 ---> '.123' */
SELEC TO_CHAR(0.123) FROM DUAL
```

上面的结果 '.123' 在大多数情况下都不是我们想要的结果，我们想要的应该是 '0.123'。

我们来看一下to_char函数的具体用法：

TO_CHAR (n [, fmt [, 'nlsparam']])

Oracle to_char函数将NUMBER类型的n按数值格式fmt转换成VARCHAR2类型的值。'nlsparams'指定由数值格式的元素返回的字符,包括:

. 小数点字符

. 组分隔符

. 本地钱币符号

. 国际钱币符号

变元的形式为：

'NLS_NUMERIC_CHARACTERS="dg" NLS_CURRENCY="tcxt" NLS_ISO_CURRENCY=territory'

其中d为小数点字符,g为组分隔符。
例 :TO_CHAR (17145,'L099G999','NLS_NUMERIC_CHARACTERS="."
NLS_CURRENCY="NUD"')=NUD017,145
通过上面的了解，再查看fmt的一些格式，我们可以用以下表达式得到'0.123'的值：

```
/*0.123 ---> ' 0.123' */
Select TO_CHAR(0.123,'0.999') FROM DUAL
/*100.12 ---> '######' */
Select TO_CHAR(100.12,'0.999') FROM DUAL
/*1.12 ---> ' 1.120' */
Select TO_CHAR(1.12,'0.999') FROM DUAL
```

' 0.123'是出来了，可是前面又多了一个空格。
对于 100.12 的值却是######，以及'1.12'的值变成了 '1.120'。
我们重新确定一个新的需求：
1、去空格
2、小数点最多4位，最少保留2位。
1--->'1.00' ; 1.1--->'1.00' ; 1.12-->'1.12' ; 1.1234--->'1.1234' ;
1.12345--->'1.1235'
最终实现如下：

```
/*
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
*/
Select TO_CHAR(123.0233,'FM9999999.0099') FROM DUALv
```