第 5 章　处理字符型数据

Oracle 中的数据类型大概有 23 种，可以简单地分为数字型（number）、字符型（character）、日期型（date）和其他类型 4 种。在 SQL 中，经常会使用到针对数据类型的内置函数。本章着重讲述 Oracle 中的字符型数据及其相关的函数，主要从字符型定义和字符型函数两个方面进行讲解。

5.1　字符型简介

字符类型，即字母、数字、标点符号和空格的混合形式。字符型是用来存放字符和字符串的，在 Oracle 中的字符型有 varchar2(n)、nvarchar2(n)、char(n)、nchar2(n)、long 等类型。

5.1.1　构建固定长度的字符串类型

char(n)指定变量或列的数据类型为固定长度的字符串。其中，n 代表字符串的长度。如果设置了长度小于 char(n)列的值，则 Oracle 会自动用空格填充。当然，Oracle 不允许实际字符串的长度大于 n。

数据库中的列指定为 char(n)类型时，n 的最大值不能大于 2000。否则，Oracle 将抛出错误，如示例 5-1 所示。

【示例 5-1】当 char(n)作为列的数据类型时，测试其最大长度：

```
SQL> create table test_字符(name  char(2001));

SQL 错误: ORA-00910: 指定的长度对于数据类型而言过长
```

create table test_字符(name char(2001));用于创建表 test_字符，并将列 name 的长度指定为 char(2001)；“SQL 错误: ORA-00910: 指定的长度对于数据类型而言过长”表明对于 char 类型来说，2001 已经超过了其最大长度。

尝试利用 2000 作为 char 类型的长度：

```
SQL> create table test_char(f_char char(2000));

create table 成功。
```

可见，将 char 类型的长度指定为 2000 时，Oracle 可以成功创建数据表。这印证了 char 类型在用于声明列时，其最大长度为 2000。

5.1.2　构建可变长度的字符串类型

Oracle 中提供了 varchar(n)的数据类型。varchar(n)用于存储可变长度的字符串。该类型是 Oracle 迎合工业标准中的 varchar 而制定的。当实际字符串的长度不足时，不会使用空格进行填充。同样，实际字符串的长度也不允许超出 n。

【示例 5-2】当作为列的数据类型出现时，varchar 的最大长度不能大于 4000，如下所示；

```
SQL> alter table test_字符  add  class  varchar(4001)

SQL 错误：ORA-00910：指定的长度对于数据类型而言过长

SQL> alter table test_字符  add  class  varchar(4000)

alter table test_字符 成功。
```

alter table test_字符 add class varchar(4001)用于为表 test_字符添加类型为 varchar(4001)的列；“SQL 错误: ORA-00910: 指定的长度对于数据类型而言过长”表明，指定长度已经大于数据类型 varchar(n)的最大长度；alter table test_字符 add class varchar(4000)表明，将长度修改为 4000 之后，可以成功添加列 class。

5.1.3　构建可变长度的字符串新类型

与 varchar(n)类型相同，varchar2(n)同样是可变长度的字符串类型。Oracle 在工业标准之外，自定义了该数据类型。在今后的 Oracle 版本中也许会有变化。所以提醒用户，尽量使用 varchar2(n)，而非 varchar(n)。因为使用 varchar2(n)可以获得 Oracle 向后兼容性的保证。

【示例 5-3】当作为列的数据类型出现时，varchar2 的长度同样不能大于 4000，如下所示：

```
SQL> alter table test_字符  add  DEPT  varchar2(4001);

SQL 错误：ORA-00910：指定的长度对于数据类型而言过长

SQL> alter table test_字符  add  DEPT  varchar2(4000);

 alter table test_字符 成功
```

由此可以看出，varchar2 与 varchar 的最大长度均为 4000。

5.2　对比三种字符串数据类型

char 数据类型是一种有固定长度和最大长度的字符串类型，varchar2 数据类型是一种可变长度的、有最大长度的字母数字型数据。对于一般用途的数据表来说，最常用的字符串类型为 varchar2(n)。本节将着重分析 char(n)与 varchar2(n)的区别。

5.2.1　varchar2(n) 和 char(n)

varchar2(n)为可变字符串类型，而 char(n)为固定字符串类型。这二者的区别在于是否使用空格来补齐不足的部分。

【示例 5-4】在表 test_char 中，列 f_char 和 f_varchar2 的长度分别为 2000 和 4000。先向其中插入新的记录，并为两列赋予相同的值。

```
SQL> insert into test_字符(name, DEPT)
    values('000', '000');

1 行 已插入
```

再为列 name、DEPT 都赋予了值“000”，其长度为 3。通过数据库查询工具查询时，用户无法察觉到二者的区别。

```
SQL> select * from   test_字符;

NAME        CLASS               DEPT
---------- -----  ------------ ------------------------
00                              000
```

可以通过 length()函数来查看此时两列的实际长度。

```
SQL>  select    length(name),length(dept)  from   test_字符;

LENGTH(NAME)        LENGTH(DEPT)
------------------ --------------------------
2000                3

1 rows selected
```

可见，此时 name 为固定字符串类型，因此长度为 2000；而 dept 为可变字符型，长度为 3。这代表 name 已经占用了 2000 字节的空间。而 dept 则占用了 3 个字节的空间。

5.2.2　总结 varchar2(n) 和 char(n)的不同

通过示例 5-4 可以看出，char(n)类型的列通常占用较大的存储空间；而 varchar2(n)类型的列占用的空间较小。所以，varchar2(n)类型是我们在进行数据库设计时的一般选择。但这并不意味着 char(n)类型应该被摒弃。相反，char(n)在效率方面要高于 varchar2(n)。这是因为可变长度的字符串类型在实际数据长度发生改变时，总需要不断调整存储空间。尤其是频繁修改数据，而数据长度也不断改变的情况下，这种效率的损耗尤其明显。

大多数的应用程序，并不将数据库端的效率作为首要考虑的需求。而更倾向于较小的空间代价，因此大多使用 varchar2(n)来定义列。而 char(n)则是典型的“以空间换时间”，读者可以在实际开发中酌情选择。

5.2.3　构建变量中的字符串类型

Character 数据类型用来存储字母数字型数据。当在 Oracle 中定义一个字符数据时，通

常需要指定字段的长度，它是该字段的最大长度。常用的三种字符串类型——char(n)、varchar(n)、varchar2(n)，都可用于声明变量。但是，利用三者声明时，最大长度均为 32767。

【示例 5-5】测试 char(n)、varchar(n)及 varchar2(n)用于变量声明时的最大长度。

```
SQL> declare   str   char(32768);
    begin
      null;
    end;

 ORA-06550: 第 1 行, 第 19 列:
PLS-00215: 字符串长度限制在范围 (1...32767)
```

declare str char(32768)用于声明名为 str 的变量，长度为 32768，字符串长度限制在范围 (1...32767)，该错误表示字符串类型的长度必须介于 1～32767 之间；同样，我们还可以证明 varchar(n)和 varchar2(n)中的 n 最大值均为 3277。

另外，需要注意的是，char(n)并不适用于声明变量。

```
SQL> declare str   char(32767);
    begin
      str:= '000';
      dbms_output.put_line(length(str));
    end;

32767

anonymous block completed
```

虽然我们为变量 str 赋值为“000”，但是经过测试，其长度仍然为 32767。这显然会造成内存空间的浪费。

5.3　常见字符串操作

Oracle 提供了丰富的字符串函数来处理字符型数据。字符串函数也称为字符函数，可以分为两类：返回值为字符的函数和返回值为数字的函数。本节将着重介绍这些函数。

5.3.1　向左补全字符串

字符串函数中有些函数是用来进行剪切和粘贴的。lpad 和 rpad 可以在串的左边或右边连接空格或其他字符。lpad 和 rpad 是十分相似的函数。lpad()函数用于向左补全字符串，该函数主要用于字符串的格式化。格式化的方式为：将字符串格式化为指定长度，如有不足部分，则在字符串的左端填充特定字符。其调用方式如下所示：

```
lpad(string, padded_length, [pad_string])
```

其中，第一个参数指定原始字符串；第二个参数指定格式化之后的字符串长度；第三个参数指定使用哪个字符来填充不足位数。

【示例 5-6】employeenew 表中员工工号原来是 3 位，现在希望被格式化为 5 位数字，

不足部分使用“0”进行填充。例如工号 1，应被格式化为“00001”，那么此时可以利用lpad()函数。

```
SQL>  select lpad(id, 5, '0')  empplyee_id from employeenew;

EMPPLYEE_ID
-----------
00100
00101
00102
00103
00104
00105
00106
00107
00108

00207
00208

109 rows selected
```

其中，lpad(id, 5, '0')用于格式化字符串 id 为长度为 5 的字符串，并利用“0”自左端起，将不足部分补全。

当然，如果原字符串长度本身已超过预期长度，也将从字符串左端进行截取。

```
SQL> select  lpad('12345', 4, '0')  empplyee_no from dual;

EMPPLYEE_NO
-----------
1234
```

lpad('12345', 4, '0')试图返回依据字符串“12345”格式化后的 4 位字符，并使用“0”自左端填充不足位数。但“12345”已经超出了预期长度 4，Oracle 将返回自左端截取的 4 位字符。

5.3.2　向右补全字符串

与 lpad()函数相似，rpad()函数完成字符串格式化为特定位数的操作，允许在列的右边填充一组字符。填充的字符可以是任意字符，包括空格、句号、逗号、字母或数字，甚至是感叹号。只是该函数自右端补全不足位数。

【示例 5-7】利用 rpad()函数实现右端补全字符串。

```
SQL> select rpad(id, 5, '0')  empplyee_id from employeenew;

EMPPLYEE_ID
-----------
10000
10100
10200
10300
10400
10500
10600
10700
```

```
10800

20700
20800

109 rows selected
```

rpad(id, 5, '0')用于右端补全字符串 id，预期长度为 5，并使用“0”来补全不足位数。

需要注意的是，当原始字符串长度大于预期长度时，rplad()函数同样是自左端截取字符串，如下所示：

```
SQL> select rpad('12345', 4, '*') empplyee_no from dual;

EMPPLYEE_NO
-----------
1234
```

rpad('12345', 4, '*')中，原字符串为“12345”，预期长度为 4，格式化结果“1234”是自原始字符串左端截取 4 位所得到的字符串。

5.3.3　字符串转化为小写形式

lower()函数用于返回字符串的小写形式。该函数仅有一个参数，即原始字符串。

【示例 5-8】在数据库中，当查询条件可以忽略大小写形式时，我们通常利用 lower()函数进行字符串大小写形式的统一。

```
SQL> select  country_name, lower(country_name) from  countries


COUNTRY_NAME             LOWER(COUNTRY_NAME)
------------------------ -------------------------
Argentina                argentina
Australia                australia
Belgium                  belgium
Brazil                   brazil
Canada                   canada
Switzerland              switzerland
China                    china
Germany                  germany
Denmark                  denmark

Zambia                   zambia
Zimbabwe                 zimbabwe

25 rows selected
```

利用 lower(country_name)将列 country_name 转换为小写形式，以便在比较时可以忽略大小写形式。

5.3.4　字符串转化为大写形式

与 lower()函数相反，upper()函数用于返回字符串的大写形式。该函数仅有一个参数，即原始字符串。

【示例 5-9】我们同样可以用 upper()函数来改写示例 5-8 的查询语句。

```
SQL> select country_name, upper(country_name) from  countries;

COUNTRY_NAME            UPPER(COUNTRY_NAME)
----------------------- ----------------------------
Argentina               ARGENTINA
Australia               AUSTRALIA
Belgium                 BELGIUM
Brazil                  BRAZIL
Canada                  CANADA
Switzerland             SWITZERLAND
China                   CHINA
Germany                 GERMANY
Denmark                 DENMARK

Zambia                  ZAMBIA
Zimbabwe                ZIMBABWE

25 rows selected
```

利用 upper(country_name)将列 country_name 转换为大写形式，以便在比较时可以忽略大小写形式。

5.3.5　单词首字符大写

initcap()函数用于将单词转换为首字符大写、其他字符小写的形式。

【示例 5-10】利用 initcap()函数格式化单词。

```
SQL> select first_name, initcap(first_name) from  employeenew;

FIRST_NAME                INITCAP(FIRST_NAME)
--------------------    --------------------
adam                      Adam
alana                     Alana
alberto                   Alberto
alexander                 Alexander
alexander                 Alexande
alexis                    Alexis
allan                     Allan
alyssa                    Alyssa
amit                      Amit
anthony                   Anthony
britney                   Britney
Bruce                     Bruce
Charles                   Charles

winston                   Winston
109 rows selected
```

分析查询结果可知，无论我们使用全小写形式的 adam，还是首字母大写形式的 Bruce，经过 initcap()函数转换之后，都将返回首字符大写、其余字符小写的形式。

利用 initcap()函数转换的是以单词为单位，而非整个字符串。例如，某个字符串含有多个单词，那么每个单词都将执行相同的操作。

```
SQL> select initcap('welcome to beijing') new_string from dual;
```

```
NEW_STRING
------------------
Welcome To Beijing

1 rows selected
```

initcap('welcome to beijing')中的各个单词——welcome、to、beijing 都将执行 initcap()函数的格式化操作。这里需要注意的是，只要是非单词字符都将作为单词的分隔符。例如，利用“-“进行分隔的字符串将被视作多个单词。

```
SQL> select initcap('welcome-to-beijing') new_string from dual;

NEW_STRING
------------------
Welcome-To-Beijing

1 rows selected
```

5.3.6　获取字符串长度

length()函数可用于返回字符串长度，如果字符串为 null，那么将返回 null 值。

【示例 5-11】利用 length()函数返回字符串长度，查询 job_history 表中 job_id 的长度。

```
SQL> select job_id,length(job_id)  长度  from job_history

JOB_ID           长度
-------------- ----------------
IT_PROG          7
AC_ACCOUNT       10
AC_MGR           6
MK_REP           6
ST_CLERK         8
ST_CLERK         8
AD_ASST          7
SA_REP           6
SA_MAN           6
AC_ACCOUNT       10

10 rows selected
```

对于 length()函数来说，无论是单字节字符还是双字节字符，都将被视作一个字符进行计算。

```
SQL> select length('数据库') len from dual;

LEN
----------------------
3

1 rows selected
```

分析查询结果可知，对于 length()函数来说，“数据库”中的 3 个双字节字符，均被计算为 1。

length()函数的参数不仅可以为字符串，还可以为其他数据类型，如数值型。

```
SQL> select length(1363.187) len from dual;

 LEN
----------
  8
1 rows selected
```

length(1363.187)用于返回数字 1363.187 的长度。Oracle 首先将其转换为字符串“1363.187”，然后获得字符串的长度 8。

对于字符型函数来说，如果传入的参数是一个空值，那么函数的返回值往往也是空值。对于 Oracle 函数来说，这是一个非常重要的特点，因此要格外注意。

【示例 5-12】查看参数为空值时，函数的返回值。

```
SQL> select length('') from dual;

LENGTH('')
----------
```

length('')用于获取空字符串的长度。按照常理推断，返回结果应该为 0。但是 Oracle 将空字符串一律视为 null。将 null 作为参数的函数，其返回值为 null。

5.3.7　截取字符串

使用 substr()函数可以对提供的参数进行截取。该函数的一般调用形式如下所示：

```
substr(string, start_index, length)
```

其中，第一个参数 string 指定原始字符串；第二个参数 start_index 指定开始截取的位置；第三个参数指定截取的长度。需要注意的是，Oracle 中字符串第一个字符的位置为 1。这与很多编程语言，如 Java 中自 0 开始的习惯不同。

【示例 5-13】利用 substr()函数截取字符串。

```
SQL> select product_name,substr( product_name, 1, 7) sub_string from
product_information;

SUB_STRING
----------
PRODUCT_NAME                    SUB_STRING
--------------------------- ------------------------------
LCD Monitor 11/PM               LCD Mon
LCD Monitor 9/PM                LCD Mon
Monitor 17/HR                   Monitor
Monitor 17/HR/F                 Monitor
Monitor 17/SD                   Monitor
Monitor 19/SD                   Monitor
Monitor 19/SD/M                 Monitor
Monitor 21/D                    Monitor
Monitor 21/HR                   Monitor
Monitor 21/HR/M                 Monitor
Monitor 21/SD                   Monito
Monitor Hinge - HD              Monitor
Monitor Hinge - STD             Monitor
Plasma Monitor 10/LE/VGA        Plasma
```

```
Plastic Stock - O                  Plastic
Plastic Stock - W/HD               Plastic

288 rows selected
```

substr(product_name, 1,7)用于将 product_name 字段的第 1 位开始，截取长度为 7 的子字符串。

当没有指定第三个参数，即没有指定截取长度时，substr()函数将返回自 start_index 开始，直至整个字符串末尾的子字符串。

```
SQL> select substr('123456789', 2) sub_string from dual;

PRODUCT_NAME                       SUB_STRING
------------------------------ --------------
LCD Monitor 11/PM                  nitor 11/PM
LCD Monitor 9/PM                   nitor 9/PM
Monitor 17/HR                      r 17/HR
Monitor 17/HR/F                    r 17/HR/F
Monitor 17/SD                      r 17/SD
Monitor 19/SD                      r 19/SD
Monitor 19/SD/M                    r 19/SD/M

Plastic Stock - O                  c Stock - O
Plastic Stock - W/HD               c Stock - W/HD

288 rows selected
```

substr(product_name, 7)用于截取字符串中，自第 7 位开始至字符串末尾的所有字符。

5.3.8　查询子字符串

利用 instr()函数可以在一个字符串中检索另一个给定字符所出现的位置，也可以作为判定原来字符串是否包含指定字符串的标准，返回结果是一个整数。如果子字符串未出现在父字符串中，该函数将返回 0。其调用形式如下所示：

```
instr(string, sub_string[, start_index] [, times])
```

其中，string 为父字符串；sub_string 为子字符串；start_index 为可选参数，指定进行搜寻的起始位置；times 为可选参数，表示第几次获得子字符串。

【示例 5-14】利用 instr()函数获得子字符串的出现位置。

```
SQL> select product_name,instr( product_name, 'Mon') sub_string from
product_information;

PRODUCT_NAME                       SUB_STRING
------------------------------ -------------------------
LCD Monitor 11/PM                  5
LCD Monitor 9/PM                   5
Monitor 17/HR                      1
Monitor 17/HR/F                    1
Monitor 17/SD                      1
Monitor 19/SD                      1
Monitor 19/SD/M                    1
Monitor 21/D                       1
Monitor 21/HR                      1
```

```
Monitor 21/HR/M                1
Monitor 21/SD                  1
Monitor Hinge - HD             1
Monitor Hinge - STD            1
Plasma Monitor 10/LE/VGA       8
Plasma Monitor 10/TFT/XGA      8
Plasma Monitor 10/XGA          8

Plastic Stock - O              0
Plastic Stock - W/HD           0

288 rows selected
```

instr(product_name, 'Mon')用于获取子字符串“Mon”在字段 product_name 中的位置。利用第三个参数可以指定从父字符串中的哪个位置开始进行搜寻。

```
SQL> select product_name,instr( product_name, 'Mon',3) sub_string from
product_information

PRODUCT_NAME               SUB_STRING
-------------------------- --------------
LCD Monitor 11/PM          5
LCD Monitor 9/PM           5
Monitor 17/HR              0
Monitor 17/HR/F            0
Monitor 17/SD              0
Monitor 19/SD              0
Monitor 19/SD/M            0
Monitor 21/D               0
Monitor 21/HR              0
Monitor 21/HR/M            0
Monitor 21/SD              0
Monitor Hinge - HD         0
Monitor Hinge - STD        0
Plasma Monitor 10/LE/VGA   8
Plasma Monitor 10/TFT/XGA  8
Plasma Monitor 10/XGA      8

Plastic Stock - O          0
Plastic Stock - W/HD       0

288 rows selected
```

instr(product_name, 'Mon')表示从字段 product_name 的第 3 个字符开始搜索子字符串“Mon”。显然，原来返回结果为 1 的现在返回结果是 0。

5.3.9　删除左侧空格

ltrim、rtrim 类似修剪机，它们从字符串的左边或者右边删除不需要的字符。ltrim()函数用于删除字符串的左侧空格。

【示例 5-15】利用 ltrim()函数删除字符串的左侧空格。

```
SQL> select first_name,ltrim(first_name) new_str from employeenew;

FIRST_NAME             NEW_STR
---------------------- ----------------------
  Bruce                Bruce
```

```
 Charles               Charles
 Curtis                Curtis
 Clara                 Clara
 britney               britney
 winston               winston
C  hristoph er         C  hristoph er

109 rows selected
```

ltrim(first_name)用于删除字符串左侧的所有空格。

5.3.10　删除右侧空格

rtrim()函数用于删除字符串的右侧空格。

【示例 5-16】利用 rtrim()函数删除字符串右侧空格。

```
SQL> select length(first_name), length(ltrim(first_name))from employeenew;

LENGTH(FIRST_NAME)       LENGTH(LTRIM(FIRST_NAME))
---------------------- -------------------------
9                         7
11                        9
10                        8
8                         7
9                         8
9                         8
14                        14

109 rows selected
```

ltrim(first_name)用于删除字符串的右侧空格。我们在此处通过求得字符串的长度来判定，查询结果，虽然原字符串左、右两侧均含有空格，但是 rtrim()函数仅删除了右侧空格。

5.3.11　删除两侧空格

trim()函数可以返回字符串两侧的空格，它实际是 ltrim()函数与 rtrim()函数的综合操作结果。

【示例 5-17】利用 trim()函数删除字符串两侧空格。

```
SQL> select length(first_name),length(trim(first_name))from
employeenew;

LENGTH(FIRST_NAME)      LENGTH(TRIM(FIRST_NAME))
---------------------- ------------------------
9                         5
11                        7
10                        6
8                         5
9                         7
9                         7
14                        14

4                         4
7                         7
```

```
109 rows selected
```

trim(first_name)用于删除字符串左、右两侧的空格。

5.3.12　连接字符串

concat()函数可以将两个字符串进行连接，它的效果和操作符“||”返回的效果是相同的。

【示例 5-18】利用 concat()函数连接字符串。

```
SQL> select first_name, last_name,concat(first_name, last_name)  name from
employeenew l;

FIRST_NAME          LAST_NAME               NAME
------------------- ----------------------- -------------
Bruce               Ernst                   Bruce  Ernst
Charles             Johnson                 Charles  Johnson
Curtis              Davies                  Curtis  Davies
Clara               Vishney                 Clara  Vishney
britney             Everett                 britney  Everett
winston             Taylor                  winston  Taylor
Christoph er        Olsen                   Christoph  erOlsen
Daniel              Faviet                  DanielFaviet

amit                Banda                   amit  Banda
anthony             Cabrio                  anthony  Cabrio
                    Abbey                   Abbey

109 rows selected
```

concat(first_name, last_name)将 first_name 和 last_name 两个字段进行连接。

需要注意的是，concat()函数只有两个参数，如果需要对多个字符串进行连接，则需要多个 concat()函数嵌套，如下所示：

```
SQL> select concat(concat('Hello ', 'Hello'), ' World') new_str from dual;

NEW_STR
-----------------
Hello Hello World
```

concat(concat('Hello ', 'Hello'), ' World')中，首先利用 concat('Hello ', 'Hello')将字符串“Hello ”与“Hello”连接，然后再与“ World”连接。

5.3.13　翻译字符串

translate()函数的执行过程非常类似于翻译的过程。

【示例 5-19】利用 translate()函数翻译字符串。

```
SQL> select translate('+-*', '1+1363-30*', 'aAbBcCdDeEfRgGhHiIjJkKlLmMnN')
transfrom dual;

TRANS
```

```
-----
AdE

1 rows selected
```

translate('+-*', '1+1363-30*', 'aAbBcCdDeEfRgGhHiIjJkKlLmMnN')用于翻译第一个字符串“+-*”。翻译过程为：在第二个字符串“1+1363-30*”中寻找“+”，得到其位置为 2，然后在第三个字符串中获得位置为 2 的字符“A”；接着分别翻译“-”和“*”，在字符串“aAbBcCdDeEfRgGhHiIjJkKlLmMnN”中分别获得字符“d”和“E”。因此最终结果返回“AdE”。

transalate()函数的典型应用为获得字母与数字混杂的字符串中的纯字母或纯数字。

```
SQL> select translate('12ipfa430uioon7663', '#abcdefghijklmnopqistuvwxqz',
 ' ') num from dual;

NUM
---------
124307663
```

translate('12ipfa430uioon7663', '#abcdefghijklmnopqistuvwxqz', ' ')用于获得纯数字字符串。原字符串“12ipfa430uioon7663”既含有数字也含有字母，因此第二个参数中包含了 26 个英文字母和一个额外字符“#”；而第三个参数则仅含有一个空格，该空格与“#”对应，其余 26 个英文字母都将无法成功翻译，实际相当于删除了这些字符。

在上个示例中，我们利用了额外的“#”与空格符“ ”进行对应，而不能直接使用空字符串。

```
SQL> select translate('12ipfa430uioon7663', 'abcdefghijklmnopqistuvwxqz',
'') num from dual;
NUM
----
```

translate('12ipfa430uioon7663', 'abcdefghijklmnopqistuvwxqz', '')中的第三个参数为空字符串。因此，该函数的返回值为空。

5.4　本 章 小 结

如果信息为字符类型，则需要使用字符串函数对它进行修改。本章着重讲述了 Oracle 中常用的字符型及与之相关的内置函数。对于字符型函数，如果将 null 作为参数时，往往返回值为 null；即使字符型函数所处理的是一个变量，在内部处理时，都是针对参数的一个副本，而不会真正改变变量的值。

5.5　本 章 习 题

【题目 1】以首字母大写的方式显示数据表 employeenew 中所有员工的姓名，结果如下

所示：

```
FIRST_NAME            INITCAP(FIRST_NAME)
--------------------- --------------------
adam                  Adam
alana                 Alana
alberto               Alberto
alexander             Alexander
alexander             Alexande
alexis                Alexis
allan                 Allan
alyssa                Alyssa
amit                  Amit
anthony               Anthony
britney               Britney
Bruce                 Bruce
Charles               Charles

winston               Winston
109 rows selected
```

【题目 2】显示数据表 employeenew 中正好是 5 个字符的员工姓名，结果如下所示：

```
ID              FIRST_NAME
--------------- --------------------
203             Susan
214             Anker
209             Loney
222             aviva
186             Julia
188             Kelly
192             Sarah
195             Vance
196             Alana
197             Kevin

165             David
174             Ellen

38 rows selected
```

【题目 3】显示数据表 employeenew 中所有员工姓名的前 3 个字符，结果如下所示：

```
ID                    FIRST_NAME         SUB_STRING
----------------- -------------------- ----------
198                   Donald             Don
199                   Douglas            Dou
200                   Jennifer           Jen
201                   Michael            Mic
202                   Pat                Pat
203                   Susan              Sus
204                   Hermann            Her
205                   Shelley            She
206                   William            Wil
214                   Anker              Ank

184                   Nandita            Nan
185                   Alexis             Ale

119 rows selected
```