Character Functions

Many implementations of SQL provide functions to manipulate characters and strings of characters.

CHR

**CHR** returns the character equivalent of the number it uses as an argument. The character it returns depends on the character set of the database. For this example the database is set to ASCII.

**INPUT:**

SQL> **SELECT CHR(65) FROM DUAL;**

**OUTPUT:**

A

CONCAT

It is similar to that of concatenate operator ( | | )

**INPUT:**

SQL> **SELECT CONCAT(‘KRISHNA’, ‘ KANTH’) FROM DUAL;**

INITCAP

**INITCAP** capitalizes the first letter of a word and makes all other characters lowercase.

**INPUT:**

SQL> **SELECT ENAME “BEFORE”, INITCAP (ENAME) “AFTER”FROM EMP;**

LOWER and UPPER

As you might expect, **LOWER** changes all the characters to lowercase; **UPPER** does just the changes all the characters to uppercase.

SQL>**SELECT ENAME,UPPER(ENAME) UPPER\_CASE,LOWER(ENAME) LOWER\_CASE FROM EMP;**

ENAME UPPER\_CASE LOWER\_CASE

---------- ---------- ----------

SMITH SMITH smith

ALLEN ALLEN allen

LPAD and RPAD

**LPAD** and **RPAD** take a minimum of two and a maximum of three arguments. The first argument is the character string to be operated on. The second is the number of characters to pad it with, and the optional third argument is the character to pad it with. The third argument defaults to a blank, or it can be a single character or a character string.

The following statement adds five pad characters, assuming that the field **LASTNAME** is defined as a 15-character field:

**INPUT:**

SQL> **SELECT LPAD(ENAME,15,’\*’) FROM EMP;**

**OUTPUT:**

LPAD(ENAME,15,'

---------------

\*\*\*\*\*\*\*\*\*\*SMITH

\*\*\*\*\*\*\*\*\*\*ALLEN

\*\*\*\*\*\*\*\*\*\*\*WARD

**15 locations allocated to display ename, out of that, name is occupying some space and in the remaining space to the left side of the name pads with \*.**

**INPUT**

SQL> **SELECT RPAD(5000,10,’\*’) FROM DUAL;**

**5000\*\*\*\*\*\***

LTRIM and RTRIM

LTRIM and RTRIM take at least one and at most two arguments. The first argument, like LPAD and RPAD, is a character string. The optional second element is either a character or character string or defaults to a blank. If you use a second argument that is not a blank, these trim functions will trim that character the same way they trim the blanks in the following examples.

**INPUT:**

SQL> **SELECT ENAME, RTRIM(ENAME,’R’) FROM EMP;**

**OUTPUT:**

ENAME RTRIM(ENAM

---------- ----------

SMITH SMITH

ALLEN ALLEN

WARD WARD

JONES JONES

ANALYSIS

Removes the rightmost character

REPLACE

**REPLACE** does just that. Of its three arguments, the first is the string to be searched. The second is the search key. The last is the optional replacement string. If the third argument is left out or **NULL**, each occurrence of the search key on the string to be searched is removed and is not replaced with anything.

**SYNTAX :**

**REPLACE(STRING,SEARCH\_STRING,REPLACE\_STRING)**

**INPUT:**

**SQL> SELECT REPLACE (‘RAMANA’,’MA’, VI’) FROM DUAL;**

**OUTPUT**

**RAVINA**

**INPUT**

**SQL> SELECT REPLACE(‘RAMANA’,’MA’) FROM DUAL;**

## ANALYSIS

**When the replace string is missing, search string removed from the given string**

## INPUT

**SQL> SELECT REPLACE (‘RAMANA’,’MA’, NULL) FROM DUAL;**

**OUTPUT**

**RANA**

TRANSLATE

The function **TRANSLATE** takes three arguments: the target string, the **FROM** string, and the **TO** string. Elements of the target string that occur in the **FROM** string are translated to the corresponding element in the **TO** string.

**INPUT:**

SQL> **SELECT TRANSLATE(‘RAMANA’,’MA’,’CD’) FROM DUAL;**

**OUTPUT:** RDCDND

Notice that the function is case sensitive.

**When search string matches, it replaces with corresponding replace string and if any one character is matching in the search string , it replaces with corresponding replace character.**

SUBSTR

This three-argument function enables you to take a piece out of a target string. The first argument is the target string. The second argument is the position of the first character to be output. The third argument is the number of characters to show.

**SYNTAX**

**SUBSTR(STRING,STARTING\_POSITION[,NO\_OF\_CHARACTERS])**

**INPUT:**

SQL> **SELECT SUBSTR(‘RAMANA’,1,3) FROM DUAL;**

**OUTPUT:** RAM

ANALYSIS

It takes first 3 characters from first character

**INPUT:**

SQL> **SELECT SUBSTR(‘RAMANA’,3,3) FROM DUAL;**

**OUTPUT:** MAN

ANALYSIS

It takes 3 characters from third position

**INPUT:**

SQL> **SELECT SUBSTR(‘RAMANA’,-2,2) FROM DUAL;**

**OUTPUT:** NA

ANALYSIS

You use a negative number as the second argument, the starting point is determined by counting backwards from the end.

**INPUT:**

SQL> **SELECT SUBSTR(‘RAMANA’,1,2) || SUBSTR(‘RAMANA’,-2,2) FROM DUAL;**

**OUTPUT:** RANA

ANALYSIS

First two characters and last two characters are combined together as a single string

**INPUT:**

SQL> **SELECT SUBSTR(‘RAMANA’,3) FROM DUAL;**

**OUTPUT:** MANA

ANALYSIS

When third argument is missing, it takes all the character from starting position

**INPUT:**

SQL> **SELECT \* FROM EMP WHERE SUBSTR(HIREDATE,4,3) = SUBSTR(SYSDATE,4,3);**

**OUTPUT:** RANA

ANALYSIS

Displays all the employees who joined in the current month

SYSDATE is a single row function, which gives the current date.

**INPUT:**

SQL> **SELECT SUBSTR (‘RAMANA’,1,2) || SUBSTR(‘RAMANA’,-2,2) FROM DUAL;**

**OUTPUT:** RANA

ANALYSIS

First two characters and Last two characters are combined together as a single string

INSTR

To find out where in a string a particular pattern occurs, use **INSTR**. Its first argument is the target string. The second argument is the pattern to match. The third and forth are numbers representing where to start looking and which match to report.

This example returns a number representing the first occurrence of **O** starting with the second

**SQL> SELECT INSTR(‘RAMANA’,’A’) FROM DUAL;**

**OUTPUT 2**

**ANALYSIS**

**Find the position of the first occurrence of letter A**

INPUT

SQL> SELECT INSTR(‘RAMANA’,’A’,1,2) FROM DUAL;

OUTPUT: 4

ANALYSIS

Find the position of the second occurrence of letter A from the beginning of the string.

Third argument represents from which position, Fourth argument represents, which occurrence.

SQL> SELECT INSTR (‘RAMANA’,’a’) FROM DUAL;

OUTPUT: 0

ANALYSIS

Function is case sensitive; it returns 0 (zero) when the given character is not found.

INPUT:

SQL> SELECT INSTR(‘RAMANA’,’A’,3,2) FROM DUAL;

OUTPUT: 6

ANALYSIS

Find the position of the second occurrence of letter A from 3rd position of the string