SQL | Character Functions with Examples

Character functions accept character inputs and can return either characters or number values as output. SQL provides a number of different character datatypes which includes – CHAR, VARCHAR, VARCHAR2, LONG, RAW, and LONG RAW. The various datatypes are categorized into three different datatypes :

1. **VARCHAR2** – A variable-length character datatype whose data is converted by the RDBMS.
2. **CHAR** – The fixed-length datatype.
3. **RAW** – A variable-length datatype whose data is not converted by the RDBMS, but left in “raw” form.

Note : When a character function returns a character value, that value is always of type VARCHAR2 (variable length), with the following two exceptions: UPPER and LOWER. These functions convert to upper and to lower case, respectively, and return the CHAR values (fixed length) if the strings they are called on to convert are **fixed-length** CHAR arguments.

Character Functions

SQL provides a rich set of character functions that allow you to get information about strings and modify the contents of those strings in multiple ways. Character functions are of the following two types:
1. Case-Manipulative Functions (LOWER, UPPER and INITCAP)
2. Character-Manipulative Functions (CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM and REPLACE)

Case-Manipulative Functions

1. **LOWER :** This function converts alpha character values to lowercase. LOWER will actually return a fixed-length string if the incoming string is fixed-length. LOWER will not change any characters in the string that are not letters, since case is irrelevant for numbers and special characters, such as the dollar sign ($) or modulus (%).
   Syntax:

   ```
   LOWER(SQL course)


   Input1: SELECT LOWER('GEEKSFORGEEKS') FROM DUAL;
   Output1: geeksforgeeks

   Input2: SELECT LOWER('DATABASE@456') FROM DUAL;
   Output2: database@456
   ```

2. **UPPER :** This function converts alpha character values to uppercase. Also UPPER function too, will actually return a fixed-length string if the incoming string is fixed-length. UPPER will not change any characters in the string that are not letters, since case is irrelevant for numbers and special characters, such as the dollar sign ($) or modulus (%).
   Syntax:

   ```
   UPPER(SQL course)


   Input1: SELECT UPPER('geeksforgeeks') FROM DUAL;
   Output1: GEEKSFORGEEKS

   Input2: SELECT UPPER('dbms$508%7') FROM DUAL;
   Output2: DBMS$508%7
   ```

3. **INITCAP :** This function converts alpha character values to uppercase for the first letter of each word and all others in lowercase. The words in the string is must be separated by either # or _ or space.
   Syntax:

   ```
   INITCAP(SQL course)


   Input1: SELECT INITCAP('geeksforgeeks is a computer science portal for geeks')
   FROM DUAL;
   Output1: Geeksforgeeks Is A Computer Science Portal For Geeks

   Input2: SELECT INITCAP('PRACTICE_CODING_FOR_EFFICIENCY') FROM DUAL;
   Output2: Practice_Coding_For_Efficiency
   ```

Character-Manipulative Functions

1. **CONCAT :** This function always appends (concatenates) string2 to the end of string1. If either of the string is NULL, CONCAT function returns the non-NULL argument. If both

strings are NULL, CONCAT returns NULL.
Syntax:

```
 CONCAT('String1', 'String2')


Input1: SELECT CONCAT('computer' ,'science') FROM DUAL;
Output1: computerscience

Input2: SELECT CONCAT( NULL ,'Android') FROM DUAL;
Output2: Android

Input3: SELECT CONCAT( NULL ,NULL ) FROM DUAL;
Output3: -
```

2. **LENGTH :** This function returns the length of the input string. If the input string is NULL, then LENGTH function returns NULL and not Zero. Also, if the input string contains extra spaces at the start, or in between or at the end of the string, then the LENGTH function includes the extra spaces too and returns the complete length of the string.
Syntax:

```
LENGTH(Column|Expression)


Input1: SELECT LENGTH('Learning Is Fun') FROM DUAL;
Output1: 15

Input2: SELECT LENGTH('   Write an Interview  Experience ') FROM DUAL;
Output2: 34

Input3: SELECT LENGTH('') FROM DUAL; or SELECT LENGTH( NULL ) FROM DUAL;
Output3: -
```

3. **SUBSTR :** This function returns a portion of a string from a given start point to an end point. If a substring length is not given, then SUBSTR returns all the characters till the end of string (from the starting position specified).
Syntax:

```
SUBSTR('String',start-index,length_of_extracted_string)


Input1: SELECT SUBSTR('Database Management System', 9) FROM DUAL;
Output1: Management System

Input2: SELECT SUBSTR('Database Management System', 9, 7) FROM DUAL;
Output2: Manage
```

4. **INSTR :** This function returns numeric position of a character or a string in a given string. Optionally, you can provide a position *m* to start searching, and the occurrence *n* of string.

Also, if the starting position is not given, then it starts search from index 1, by default. If after searching in the string, no match is found then, INSTR function returns 0.

```
Syntax: INSTR(Column|Expression, 'String', [,m], [n])


Input: SELECT INSTR('Google apps are great applications','app',1,2) FROM DUAL;
Output: 23
```

5. **LPAD and RPAD :** These functions return the strings padded to the left or right (as per the use) ; hence the "L" in "LPAD" and the "R" in "RPAD" ; to a specified length, and with a specified pad string. If the pad string is not specified, then the given string is padded on the left or right (as per the use) with spaces.
   Syntax:

```
LPAD(Column|Expression, n, 'String')
Syntax: RPAD(Column|Expression, n, 'String')


LPAD Input1: SELECT LPAD('100',5,'*') FROM DUAL;
LPAD Output1: **100

LPAD Input2: SELECT LPAD('hello', 21, 'geek') FROM DUAL;
LPAD Output2: geekgeekgeekgeekhello

RPAD Input1: SELECT RPAD('5000',7,'*') FROM DUAL;
RPAD Output1: 5000***

RPAD Input1: SELECT RPAD('earn', 19, 'money') FROM DUAL;
RPAD Output1: earnmoneymoneymoney
```

6. **TRIM :** This function trims the string input from the start or end (or both). If no string or char is specified to be trimmed from the string and there exists some extra space at start or end of the string, then those extra spaces are trimmed off.

```
Syntax:

  TRIM(Leading|Trailing|Both, trim_character FROM trim_source)


  Input1: SELECT TRIM('G' FROM 'GEEKS') FROM DUAL;
  Output1: EEKS

  Input2: SELECT TRIM('        geeksforgeeks   ') FROM DUAL;
  Output2:geeksforgeeks
```

7. **REPLACE :** This function searches for a character string and, if found, replaces it with a given replacement string at all the occurrences of the string. REPLACE is useful for searching patterns of characters and then changing all instances of that pattern in a single

function call.
If a replacement string is not given, then REPLACE function removes all the occurrences of that character string in the input string. If neither a match string nor a replacement string is specified, then REPLACE returns NULL.
Syntax:

```
 REPLACE(Text, search_string, replacement_string)

Input1: SELECT REPLACE('DATA MANAGEMENT', 'DATA','DATABASE') FROM DUAL;
Output1: DATABASE MANAGEMENT

Input2: SELECT REPLACE('abcdeabcccabdddeeabcc', 'abc') FROM DUAL;
Output2: deccabdddeec
```

This article is contributed by **Anshika Goyal.** If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Last Updated : 21 Mar, 2018

20

Similar Reads

1. SQL | Functions (Aggregate and Scalar Functions)
2. SQL | Difference between functions and stored procedures in PL/SQL
3. Difference between Structured Query Language (SQL) and Transact-SQL (T-SQL)
4. Configure SQL Jobs in SQL Server using T-SQL
5. SQL | Date functions
6. SQL | NULL functions
7. SQL general functions | NVL, NVL2, DECODE, COALESCE, NULLIF, LNNVL and NANVL
8. SQL Server Mathematical functions (SQRT, PI, SQUARE, ROUND, CEILING & FLOOR)
9. SQL | Numeric Functions