ORACLE® ACADEMY

Database Programming with SQL

4-1
Case and Character Manipulation

Objectives

This lesson covers the following objectives:

- Select and apply single-row functions that perform case conversion and/or character manipulation
- Select and apply character case-manipulation functions LOWER, UPPER, and INITCAP in a SQL query
- Select and apply character-manipulation functions CONCAT, SUBSTR, LENGTH, INSTR, LPAD, RPAD, TRIM, and REPLACE in a SQL query
- Write flexible queries using substitution variables

Purpose

- Being able to change the way in which data is presented is important when dealing with data from a database.
- Most of the time in SQL, we need to change the way that data appears depending on the requirements of the task we are trying to accomplish.
- In this lesson, you will learn several ways in which to transform data to fit a particular situation.

DUAL Table

- The DUAL table has one row called "X" and one column called "DUMMY."

DUMMY
X

- The DUAL table is used to create SELECT statements and execute functions not directly related to a specific database table.
- Queries using the DUAL table return one row as a result. DUAL can be useful to do calculations and also to evaluate expressions that are not derived from a table.

DUAL Table

- DUAL will be used to learn many of the single-row functions.
- In this example the DUAL table is used to execute a SELECT statement that contains a calculation.
- As you can see the SELECT statement returns a value that does not exist in the DUAL table.
- The value returned is a result of the calculation executed.

```
SELECT (319/29) + 12
FROM DUAL;
```

(319/29)+12
23

Single-Row Character Functions

- Single-row Character Functions are divided into two categories:
  - Functions that convert the case of character strings
  - Functions that can join, extract, show, find, pad, and trim character strings
- Single-row functions can be used in the SELECT, WHERE, and ORDER BY clauses.

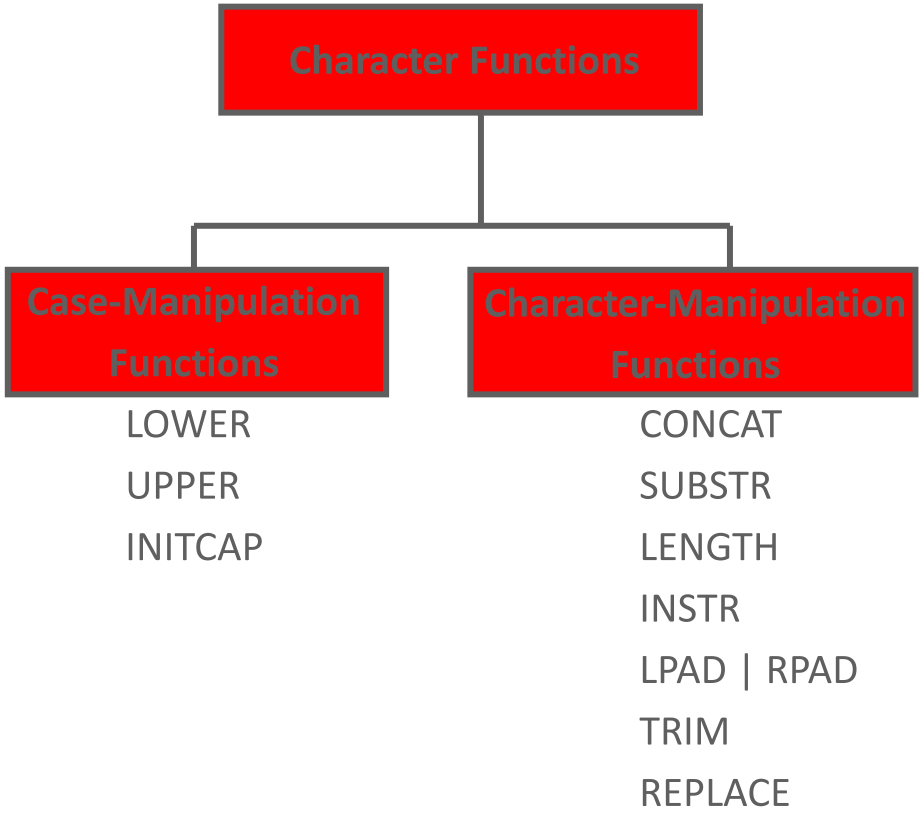
Single-Row Character Functions

- Case-manipulation functions are important because you may not always know in which case (upper, lower, or mixed) the data is stored in the database.
- Case manipulation allows you to temporarily convert the database data to a case of your choosing.
- Mismatches between database case storage and query case requests are avoided.

Case Manipulation Functions

- Case-manipulation functions are used to convert data from the state it is stored in a table to lower, upper or mixed case.
- These conversions can be used to format the output and can also be used to search for specific strings.

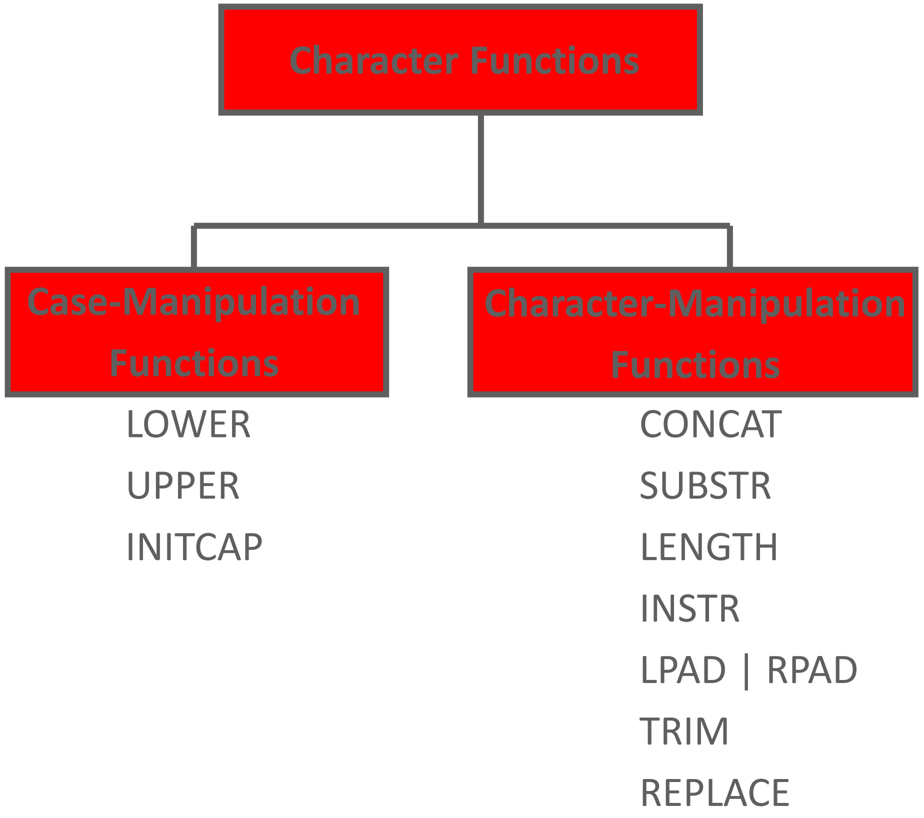
CHARACTER FUNCTIONS



Case Manipulation Functions

- Case-manipulation functions can be used in most parts of a SQL statement.

CHARACTER FUNCTIONS



Case Manipulation Functions

- Case-manipulation functions are often helpful when you are searching for data and you do not know whether the data you are looking for is in upper or lower case.
- From the point of view of the database, 'V' and 'v' are NOT the same character and, as such, you need to search using the correct case.
- LOWER(column | expression) converts alpha characters to lower-case.

```
SELECT last_name
FROM employees
WHERE LOWER(last_name) = 'abel';
```

Case Manipulation Functions

- UPPER(column | expression) converts alpha characters to upper-case.

```
SELECT last_name
FROM employees
WHERE UPPER(last_name) = 'ABEL';
```

- INITCAP(column | expression) converts alpha character values to uppercase for the first letter of each word.

```
SELECT last_name
FROM employees
WHERE INITCAP(last_name) = 'Abel';
```

Character Manipulation Functions

- Character-manipulation functions are used to extract, change, format, or alter in some way a character string.
- One or more characters or words are passed into the function and the function will then perform its functionality on the input character strings and return the changed, extracted, counted, or altered value.

Character Manipulation Functions

- CONCAT: Joins two values together.
- Takes 2 character string arguments, and joins the second string to the first. could also be written using the concatenation operator - 'Hello' || 'World'

Examples:	Result

`SELECT CONCAT('Hello', 'World')`
`FROM DUAL;`
	HelloWorld

`SELECT CONCAT(first_name, last_name)`
`FROM employees;`
	EllenAbel
	CurtisDavies
	…

Character Manipulation Functions

- SUBSTR: Extracts a string of a determined length.
- The arguments are (character String, starting position, length).
- The Length argument is optional, and if omitted, returns all characters to the end of the string.

Examples:	Result
`SELECT SUBSTR('HelloWorld',1,5)` `FROM DUAL;`	Hello
`SELECT SUBSTR('HelloWorld', 6)` `FROM DUAL;`	World
`SELECT SUBSTR(last_name,1,3)` `FROM employees;`	Abe Dav

Character Manipulation Functions

- LENGTH: Shows the length of a string as a number value.
- The function takes a character string as an argument, and returns the number of characters in that character string.

Examples:	Result

`SELECT LENGTH('HelloWorld')`
`FROM DUAL;`
	12

`SELECT LENGTH(last_name)`
`FROM employees;`
	4
	6
	…

Character Manipulation Functions

- INSTR: Finds the numeric position of the specified character(s).
- INSTR searches for the first occurrence of a substring within a character string and returns the position as a number.
- If the substring is not found, the number zero is returned.

Examples:	Result

`SELECT INSTR('HelloWorld', 'W')`
`FROM DUAL;`
	6

`SELECT last_name, INSTR(last_name, 'a')`
`FROM employees;`
	Abel 0
	Davies 2
	…

Character Manipulation Functions

- LPAD: Pads the left side of a character string, resulting in a right-justified value.
- LPAD requires 3 arguments: a character string, the total number of characters in the padded string, and the character to pad with.

Examples:	Result

`SELECT LPAD('HelloWorld',15, '-')`
`FROM DUAL;`
	-----HelloWorld

`SELECT LPAD(last_name, 10,'*')`
`FROM employees;`
	******Abel
	****Davies
	…

Character Manipulation Functions

- RPAD: Pads the right-hand side of a character string, resulting in a left-justified value.

Examples:	Result

`SELECT RPAD('HelloWorld',15, '-')`
`FROM DUAL;`
	HelloWorld-----

`SELECT RPAD(last_name, 10,'*')`
`FROM employees;`
	Abel******
	Davies****
	…

Character Manipulation Functions

- TRIM: Removes all specified characters from either the beginning, the end, or both beginning and end of a string.
- The syntax for the trim function is:

Examples:	Result
`SELECT TRIM(LEADING 'a' FROM 'abcba') FROM DUAL;`	bcba
`SELECT TRIM(TRAILING 'a' FROM 'abcba') FROM DUAL;`	abcb
`SELECT TRIM(BOTH 'a' FROM 'abcba') FROM DUAL;`	bcb

Character Manipulation Functions

- REPLACE: Replaces a sequence of characters in a string with another set of characters.
- The syntax for the REPLACE function is:

```
REPLACE (string1, string_to_replace, [replacement_string] )
```

  – string1 is the string that will have characters replaced in it
  – string_to_replace is the string that will be searched for and taken out of string1
  – [replacement_string] is the new string to be inserted in string1

Character Manipulation Functions

Examples:	Result

`SELECT REPLACE('JACK and JUE','J','BL')`
`FROM DUAL;`
	BLACK and BLUE

`SELECT REPLACE('JACK and JUE','J')`
`FROM DUAL;`
	ACK and UE

`SELECT REPLACE(last_name,'a','*')`
`FROM employees;`
	Abel
	D*vies
	De H**n

Using Column Aliases With Functions

- All functions operate on values that are in parentheses, and each function name denotes its purpose, which is helpful to remember when constructing a query.
- Often a column alias is used to name a function.
- When a column alias is used, the column alias appears in the output instead of the actual function syntax.

Using Column Aliases With Functions

- In the following examples, the alias "User Name" has replaced the function syntax in the first query.
- By default, the column name in a SELECT statement appears as the column heading.
- In the second query example, however, there is no column in the table for the results produced, so the query syntax is used instead.

Using Column Aliases With Functions

```
SELECT LOWER(last_name)|| LOWER(SUBSTR(first_name,1,1))
  AS "User Name"
FROM employees;
```

User Name
abele
daviesc
de haanl

```
SELECT LOWER (last_name)||LOWER(SUBSTR(first_name,1,1))
FROM f_staffs;
```

LOWER(LAST_NAME)||LOWER(SUBSTR(FIRST_NAME,1,1))
abele
daviesc
de haanl

Substitution Variables

- Occasionally you may need to run the same query with many different values to get different result sets.
- Imagine for instance if you had to write a report of employees and their departments, but the query must only return data for one department at a time.
- Without the use of substitution variables, this request would mean you would have to repeatedly edit the same statement to change the WHERE-clause.

Substitution Variables

- Luckily for us, Oracle Application Express supports substitution variables.
- To use them, all you have to do is replace the hardcoded value in your statement with a :named_variable.
- Oracle Application Express will then ask you for a value when you execute your statement.

Substitution Variables

- If this was the original query:

```
SELECT first_name, last_name, salary, department_id
FROM employees
WHERE department_id= 10;
```

  – Then run it again with different values: 20, 30, 40... etc.

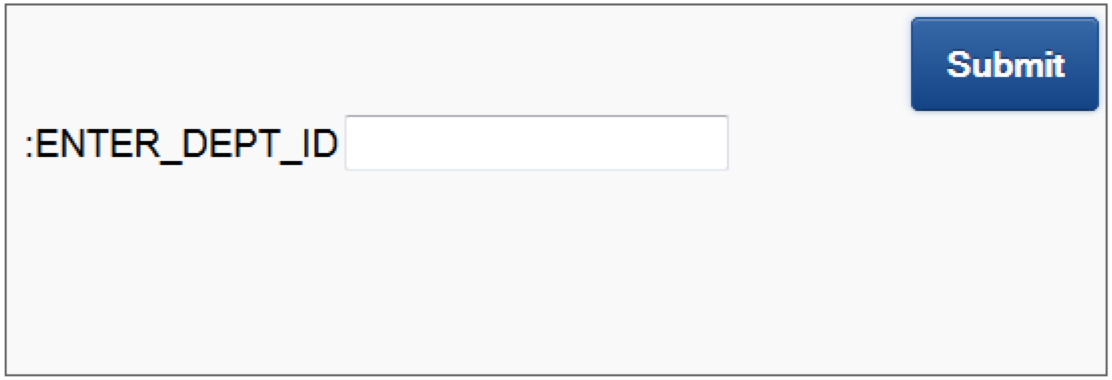
- It could be re-written as:

```
SELECT first_name, last_name, salary, department_id
FROM employees
WHERE department_id=:enter_dept_id;
```

- Note the use of : in front of enter_dept_id.
- It is the colon that is the magic bit and makes Oracle Application Express recognize the text that follows as a variable.

Substitution Variables

- When you click Run, a pop-up like the following is displayed by Oracle Application Express:



- NOTE: Pop-Up blockers must be disabled, otherwise APEX cannot ask for the variable value, as this is entered via a pop-up.

Substitution Variables

- Substitution variables are treated as character strings in Oracle Application Express, which means that when passing in character or date values, you do not need the single quotation marks that you would normally use to enclose the strings.
- So a WHERE-clause would look like this:

```
SELECT *
FROM    employees
WHERE last_name = :l_name;
```

Terminology

Key terms used in this lesson included:

- Character functions
- CONCAT
- DUAL
- Expression
- Format
- INITCAP
- Input
- INSTR

Terminology

Key terms used in this lesson included:

- LENGTH
- LOWER
- LPAD
- Output
- REPLACE
- RPAD
- Single-row functions
- SUBSTR

Terminology

Key terms used in this lesson included:

- TRIM
- UPPER
- Substitution variable

Summary

In this lesson, you should have learned how to:

- Select and apply single-row functions that perform case conversion and/or character manipulation
- Select and apply character case-manipulation functions LOWER, UPPER, and INITCAP in a SQL query
- Select and apply character-manipulation functions CONCAT, SUBSTR, LENGTH, INSTR, LPAD, RPAD, TRIM, and REPLACE in a SQL query
- Write flexible queries using substitution variables

ORACLE® ACADEMY