Single-Row SQL Functions

Speaker: Kevin Throckmorton

Introduction

Most the time, retrieving a column or columns as is from a table is sufficient. There are times however, when you need to enhance the display of a column, apply a date calculation, or handle Null values in arithmetic calculations. This is where the single-row functions come in. We will discuss some of the more commonly used ones, but a complete list can be found in the Oracle SQL Reference Manual.

Refer to your textbook for excellent examples of the following functions and their use in SQL.

Case Conversion Functions

To change the case of a column's value, Oracle uses the LOWER, UPPER and INITCAP functions. These functions can be used in the SELECT clause to affect the display, or used in the WHERE clause to affect the comparison. LOWER changes the value to lowercase, UPPER changes the value to uppercase, and INITCAP changes the first letter of each word to uppercase, and the remaining letters to lowercase. Words are determined by at least one space between each one.

Character Conversion Functions

Oracle has several functions that allow character columns to be manipulated to achieve a desired effect in the display results. Individual positions of a column can be extracted or searched and replaced, a column's length can be determined, columns can be padded or trimmed, and columns can be strung together (concatenated).

The SUBSTR (substring) function returns a portion of a column, based on the starting position and number of characters to extract. A good example would be extracting the area code from a phone number. A positive number in the starting position causes the function to work left to right from the starting position, whereas a negative number means to start the extraction that many positions from the end of the string. SUBSTR can be used on the SELECT or WHERE clause.

The INSTR (in string) function is used when you need to know the location of a specific character(s) in a string. This might be used to determine where the first space in a string resides. This function returns a number that represents the starting position of the value being searched for. If the search string is not found, it returns a value of zero. By default the INSTR function starts searching at the beginning of the string, but you can specify another starting position. You can also control which occurrence of the search string you want it to search for. This function is often used in tandem with the SUBSTR function by nesting one inside the other.

The LENGTH function returns the number of characters stored in a column. This can be useful to know for planning purposes when the data will be retrieved and displayed in another application or printed out. LENGTH works best on a VARCHAR (variable character) column, but on a CHAR column it returns the column's physical size, not necessarily the number of characters stored in that column.

LPAD and RPAD functions allow a string to be padded on the left or right with a specific character, or blank. You can control how much padding takes place by providing the number of characters the field should be after padding. The classic example is a check, where the amount payable is padded on the left with asterisks to avoid tampering.

The opposite functions to LPAD and RPAD are LTRIM and RTRIM, which remove characters from the left or right side of a string. The amount of data to be removed is determined by the length of the string to be removed. The string being removed must be found in its entirety for any trimming to take place. Maybe the string "Mr." or "Mrs." needs removed from the first name field. LTRIM could be used.

The REPLACE and TRANSLATE functions are similar in that they allow a search and replace task to be performed. REPLACE can search for and replace a string, but only one string in a single command. TRANSLATE can only search for and replace single characters, but may work with more than one string in a single command. If we wanted to display "Iowa" in place of "IA" for state, we might invoke the REPLACE function.

Recall from Learning Unit 2 that the concatenation operator (||) combined data from two or more columns or string literals together into a single display column. You can also use the CONCAT function, but it can use only two strings, so the concatenation operator is preferred due to its flexibility.

Number Functions

Numeric data retrieved from a column, or calculated from an expression, often needs formatted to satisfy a company's standard for displaying numeric data. Rounding using the ROUND function, or truncating using the TRUNC function, is common. Rounding and truncating can be done on the right side of the decimal point (fractional) by using a positive number or on the left side of the decimal point (whole number) by using a negative number.

Oracle also supports several numeric functions that allow a numeric value to be manipulated, such as getting the remainder of a division operation (MODulus), the absolute value of a number (ABS), or raising a number to a power (POWER).

Date Functions

Most databases have date fields, such as an employee's birth date or hire date, the date of a sale, or the date an inventory item was created. Oracle has several Date functions that allow special formatting of a date field in the display, or performing calculations on dates, such as finding the difference between two dates. Even though date fields get displayed with dashes by default, they are stored internally in a numeric format called Julian. This allows for flexibility in using them in calculations. Dates can also be rounded to the next higher month (16th or greater) or year (July 16th or greater) by using the ROUND function. Some of the more common date functions are MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, TO_DATE, and CURRENT_DATE or SYSDATE.

The MONTHS_BETWEEN function returns the number of months between two dates, and returns a floating point value that represents whole and partial month. If needed, this function can be nested inside a ROUND or TRUNC function to give only whole months.

The ADD_MONTHS function allows a whole number of months to be added to a date, and returns a new date with the months added to it. Oracle performs the needed adjustments, based on its internal calendar, for correct date processing.

Sometimes it's helpful to find, for example, the next date of a given day of the week. The NEXT_DAY function accomplishes this by returning the date of whatever day you give it. A similar function is LAST_DAY, which returns the date of the last day of the month.

The TO_DATE function allows a date in any format to be converted into Oracle's default format. This can useful in the WHERE clause to allow for flexibility in date comparisons. This function requires arguments that format the date in the same manner as the date it's comparing to.

To complete our Date discussion, you need to be familiar with the CURRENT_DATE and SYSDATE functions, which return the current date and time. The difference is that CURRENT_DATE uses the user's local computer, whereas SYSDATE uses the server the database is running on. These could be in different time zones, so be aware when determining which function should be used. These functions are used frequently with the other date functions when doing date calculations. For instance, to determine the number of months between the order date and right now, we might code:
SELECT MONTHS_BETWEEN(SYSDATE, OrderDate) "Days Elapsed" FROM Orders;
In this case, OrderDate is subtracted from the current date on the database server and returned as the number of months.

Other Useful Functions

Some other functions you may find helpful are the Regular Expressions functions, which expand the pattern matching capabilities of the LIKE operator. You may want to reference the SQL manual for a complete list of functions and operators.

The NVL and NVL2 functions allow for null values in columns that typically have values, but sometimes don't. An example would be an order that hasn't shipped yet, so the ship date is null, or calculating a discount from a base amount, but the discount isn't required so it might be null. These functions allow a "substitute" value to be used in the event there is a null. NVL2 enhances NVL in that one substitute value is to be used if a column is null, and another substitute value is to be used if it's not null.

The function to TO_CHAR allows numeric data to be converted to string data, which can then have formatting applied for display purposes. This is useful for dates and decimal defined fields. There are a variety of format characters that can be used with both the TO_DATE and TO_CHAR functions. Again, refer to the SQL Manual for a complete list.

The DECODE function and CASE expression are similar in that they both essentially achieve a logical IF/THEN/ELSE or CASE scenario, but there are some differences. The DECODE function allows a value to be searched for against a list, and if found, returns a corresponding value. If not found, it returns a default value if one is coded, or returns a null if no default is used. It operates on an equality basis only. The CASE expression not only can test for equality, but can use the comparison operators for flexibility. Which one you choose to use will depend on the requirements of your SQL statement.

The TO_NUMBER function will convert a string value to its numeric equivalent, if possible. This is handy in order to compare values numerically, or use the converted values in calculations.

Let's end with a brief discussion about an Oracle system table called DUAL. It is useful when learning and testing new functions before applying them against tables with actual data, or if your SELECT statement displays columns that are not derived from any data tables. An example might be to find three months from the current day: SELECT ADD_MONTHS(CURRENT_DATE, 3) FROM DUAL;
