

# { SebastiánCatalán; }

{ Blog dedicado a la investigación y desarrollo en distintos lenguajes de programación. Aportando información útil a estudiantes del área Informática; }

viernes, 27 de marzo de 2015

## Uso de funciones de una fila (PL/SQL)

### Funciones de Caracteres

Funciones de conversión del texto a mayúsculas y minúsculas:

- LOWER: Convierte el texto a minúsculas
- UPPER: Convierte el texto a mayúsculas
- INITCAP: Coloca la primera letra de cada palabra en mayúscula

Funciones de reemplazo o manipulación de caracteres:

- CONCAT: concatena el valor del primer carácter con el valor del segundo carácter. Equivalente al operador de concatenación ||.
- SUBSTR: obtiene los n siguientes caracteres de la columna o texto a partir de la posición m. Si no se indica n, se recuperan los caracteres desde la posición m hasta el final.
- LENGTH: obtiene el número de caracteres o largo de la expresión.
- RTRIM-LTRIM: Elimina los espacios en blanco a la derecha (RTRIM) ó a la izquierda (LTRIM) de la columna o expresión.
- REPLACE: busca el texto especificado en la columna o expresión y lo cambia por el texto indicado como reemplazo
- TRIM: Elimina los espacios en blanco a la izquierda y a la derecha del texto o columna.
- LPAD-RPAD: rellena el texto a la izquierda (LPAD) ó a la derecha (RPAD) n posiciones con el carácter indicado.
- INSTR: obtiene la posición en la se encuentra el texto buscado en la columna o expresión. Opcionalmente se puede indicar la posición inicial (m) desde donde se desea comenzar la búsqueda y la ocurrencia o número de posición (n) del texto buscado. Por defecto ambas posiciones es 1.
- TRIM('caracter' FROM columna|expresión): Elimina el carácter especificado de la derecha e izquierda de la columna o expresión

```
1 -- Ejemplo
2 SELECT last_name, UPPER(last_name), job_id, LOWER(job_id), INITCAP(job_id)
3 FROM employees;
```

En la sentencia del ejemplo, la función UPPER muestra el apellido del empleado en mayúscula, la función LOWER muestra la identificación del trabajo en minúscula y la función INITCAP muestra en mayúscula las primeras letras de la identificación del trabajo.

```
1 -- Ejemplo
2 SELECT last_name apellido, CONCAT('Su salario es ',salary),
3        SUBSTR(last_name,2,3), LENGTH(last_name), INSTR(last_name, 'a')
4 FROM employees;
```

En el ejemplo, la función CONCAT muestra el texto Su salario es unido al valor del salario, SUBSRT a partir de la segunda letra del apellido muestra tres caracteres del apellido, LENGTH muestra el largo del apellido e INSTR muestra la posición en la se encuentra la primera letra a en el apellido primera letra a en el apellido primera letra a en el apellido de cada empleado.

```
1 -- Ejemplo
2 SELECT last_name "Apellido", INSTR(last_name, 'e',1,2) "Resultado INSTR",
3        SUBSTR(last_name,-2,2) "Resultado SUBSTR",
4        TRIM('B' FROM last_name) "Resultado 1er. TRIM",
5        salary, TRIM(1 FROM salary) "Resultado 2do. TRIM"
6 FROM employees
7 WHERE salary between 9500 AND 10000
8 ORDER BY last_name;
```

En el ejemplo, la función INSTR muestra la posición donde se encuentra la segunda letra "e" en el apellido (la búsqueda comienza desde la posición 1), la función SUBSTR muestra desde la penúltima posición del apellido dos caracteres, la primera función TRIM muestra el apellido del empleado eliminando desde la izquierda y la derecha la letra B y la segunda función TRIM muestra el salario sin el número 1 en la derecha y la izquierda. La información se visualiza para los empleados cuyo salario esté entre los 9500 y los 10000 y ordenada en forma ascendente por apellido.

```
1 -- Ejemplo
2 SELECT last_name apellido, REPLACE(last_name,'A','Hola'),
3        salary salario, LPAD(salary,10,'*'), RPAD(salary,10,'*')
4 FROM employees
5 ORDER BY last_name;
```

En la sentencia, la función INSTR muestra la posición donde se encuentra la segunda letra a en el apellido (la búsqueda comienza desde la posición 1) y la función SUBSTR muestra desde la penúltima posición del apellido dos caracteres.

### Funciones de Números

- ROUND: redondea la columna, expresión o valor a n posiciones decimales. Si no se especifica n o su valor es cero, el valor se redondea al valor entero. Si n es negativo, los

números a la izquierda del punto decimal se redondean a decenas, centenas etc.

- TRUNC: Trunca la columna, expresión o valor a n posiciones decimales. Si no se especifica n decimales el valor es 0, por lo tanto se trunca el valor sólo en su parte entera. Por defecto en cero. Si n es negativo trunca hacia la izquierda del punto decimal (coloca cero).
- MOD: Devuelve el resto que resulta de dividir m por n.

```
1 -- Ejemplo
2 SELECT ROUND(1234.5678,2), ROUND(1234.5678), ROUND(1235.5678,-1),
3       TRUNC(1234.5678,2), TRUNC(1234.5678), TRUNC(1234.5678,-2)
4 FROM dual;
```

En el ejemplo, las funciones ROUND y TRUNC se utilizan para redondear y truncar el valor 1234.5678 de diferentes formas (recordar que en oracle el punto corresponde a decimales). La primera función ROUND redondea el valor en dos decimales, la segunda función ROUND redondea al valor entero (sin decimales, esto es similar a ROUND(1234.5678,0), la tercera función ROUND redondeado el valor en su parte entera a la décima más cercana a 35, la primera función TRUNC muestra el valor truncado a dos decimales, la segunda función TRUNC muestra el valor truncado solo en su valor entero (esto es similar a TRUNC(1234.5678,0) y la última función TRUNC muestra el valor truncado en su parte entera y reemplazada los dos últimos números por ceros).

```
1 -- Ejemplo
2 SELECT last_name, salary, MOD(salary,5000)
3 FROM employees
4 WHERE job_id = 'ST_MAN';
```

En la sentencia del ejemplo, se muestra el apellido, salario y el resto de la división del salario por 5000 de los empleados que poseen el trabajo ST\_MAN.

## Funciones de Fecha

La Base de Datos Oracle almacena las fechas en un formato numérico interno: siglo, año, mes, día, horas, minutos y segundos.

El formato por defecto de visualización de las fechas es :DD-MON-RR, donde DD corresponde al día, MON al mes en 3 letras y RR al año en 2 dígitos según el siglo.

Cuando se desea generar una condición por una fecha en particular, ésta debe ir entre comillas simples.

- SYSDATE: retorna la fecha y hora actual de la Base de Datos
- MONTHS\_BETWEEN: obtiene la diferencia en meses entre las dos fechas. El resultado puede ser positivo o negativo. Si fecha1 es posterior a fecha2, el resultado es positivo, si fecha1 es anterior a fecha2, el resultado es negativo. La parte no entera del resultado representa una porción de la mes.
- ADD\_MONTHS: añade a la fecha el número de meses indicado por n. El valor de n debe ser un número entero y puede ser negativo.
- NEXT\_DAY: retorna la fecha del día de la semana del argumento busca y que es posterior a la fecha entregada. El día puede ser el nombre del día (inglés o español según como esté configurada la Base de Datos) ó el número del día de la semana 1=Lunes, 2=Martes etc.
- LAST\_DAY: obtiene el último día del mes de la fecha especificada.
- ROUND: redondea la fecha al formato indicado. El formato puede ser:
  - YEAR: redondea la fecha al año.
  - MONTH: redondea la fecha al mes.
- TRUNC: trunca la fecha al formato indicado. El formato es el mismo al usado por ROUND.

```
1 -- Ejemplo
2 SELECT MONTHS_BETWEEN('01/ENE/2014','01/OCT/2013') "MONTHS_BETWEEN",
3        ADD_MONTHS('20/ENE/2014',6) "ADD_MONTHS",
4        NEXT_DAY('11/MAR/2014','DOMINGO') "NEXT_DAY",
5        LAST_DAY('01/FEB/2014') "LAST_DAY"
6 FROM dual;
```

En el primer ejemplo, la función MONTHS\_BETWEEN muestra cuántos meses existen entre el 01 de enero del 2014 y el 01 de octubre del 2013 (dependiendo de las fechas comparadas, la función puede retornar un valor entero o con decimales), la función, la función ADD\_MONTHS muestra la fecha que corresponde al sumar 6 meses a la fecha 20 de enero del 2014, la función NEXT\_DAY muestra la fecha que corresponde al día domingo posterior al 11 de marzo del 2014 y la función LAST\_DAY muestra el último día de febrero del 2014.

```
1 -- Ejemplo
2 SELECT ROUND(SYSDATE,'MONTH'), ROUND(SYSDATE,'YEAR'),
3        TRUNC(SYSDATE,'MONTH'), TRUNC(SYSDATE,'YEAR')
4 FROM dual;
```

En el segundo ejemplo, al utilizar las funciones ROUND y TRUNC con fechas el resultado estará basado en redondear las fechas al mes o al año y mostrar el primer día del mes o del año. La primera función redondea la fecha al mes de octubre del 2014 y como el día es 05 el resultado es 01/10/2014, la segunda función redondea la fecha al año 2014 y como el mes es 10 el resultado es 01/01/2015, la tercera función muestra el primer día del mes de octubre y la última función muestra el primer día del año 2014.

Las fechas en la Base de Datos se almacenan como números, por lo tanto se pueden realizar cálculos usando operadores aritméticos:

Fecha + Número: retorna una fecha. Suma un número de días a la fecha.

Fecha - Número: retorna una fecha. Resta un número de días desde la fecha.

Fecha - Fecha: retorna el número de días. Resta una fecha desde otra

```
1 -- Ejemplo
2 SELECT last_name, hire_date, ROUND((SYSDATE - hire_date) / 7 ) "SEMANAS CONTRATADO",
3        hire_date - 2 "FECHA CONTRATO MENOS 2 DIAS"
4 FROM employees
5 WHERE department_id = 90;
```

En el ejemplo, la sentencia muestra el apellido del empleado, la fecha de contrato, el número de semanas (entre la fecha actual y la fecha) que lleva contratado y la fecha resultante al restar la fecha de contrato menos 2 días de cada empleado del departamento 90.

## Conversión de Tipos de Datos

En algunos casos, el servidor Oracle usa tipos de datos distintos a los que se requieren. Cuando esto sucede, se debe convertir al tipo de datos que se requiere.

La conversión de un tipo de dato a otro puede ser efectuada en forma implícita o automática por el servidor Oracle o en forma explícita por el usuario usando funciones de conversión.

```

1 -- Ejemplo
2   SELECT employee_id, hire_date
3     FROM employees
4    WHERE hire_date > '05/03/2008'
5      ORDER BY hire_date;

```

En el ejemplo, en la expresión hire\_date > '05/03/2008' el string es convertido implícitamente a fecha al momento de comparar.

Oracle proporciona tres funciones para convertir en forma explícita un tipo de dato en otro:

- TO\_CHAR: Obtiene un texto a partir de un número o fecha. Opcionalmente se puede dar un formato específico de conversión.
- TO\_NUMBER: Convierte textos en números, indicándole, si se desea, el formato de salida.
- TO\_DATE: Convierte textos en fechas, indicándole, si se desea, el formato de salida.

```

1 -- Ejemplo
2   TO_CHAR(fecha, ' formato_conversión')

```

Se puede usar la función TO\_CHAR para convertir la fecha desde su formato por defecto a un formato especificado por el usuario.

Los principales elementos de formatos de fechas válidos son:

- YYYY: año en formato de 4 dígitos.
- YEAR: año en palabras.
- MM: mes en formato de 2 dígitos.
- MONTH: nombre completo del mes.
- MON: las tres primeras letras del mes.
- DY: día de la semana abreviado en tres letras.
- DAY: día completo de la semana en palabras.
- DD: día del mes en formato de dos dígitos.

Los principales elementos de horas válidos para los formatos de fechas son:

- AM o PM: indicador de meridiano.
- HH: hora del día
- HH12: hora del día 1 a 12
- HH24: hora del día de 0 a 23
- MI: minutos (0-59).
- SS: segundos (0-59).
- / . , : separadores que se ven reflejados en el formato final de la fecha.
- "caracteres\_a\_visualizar": cadena de caracteres a visualizar en el formato final de la fecha.

Los sufijos que se pueden utilizar en los elementos de fechas y horas para modificar la forma de visualización:

- SP: muestra el número en palabras. Ej: para el 04 muestra la palabra cuarto.
- SPTH o THSP: muestra en palabras el número ordinal. Ej: para 04 muestra la palabra cuarto.

```

1 -- Ejemplo
2   SELECT last_name "Apellido", TO_CHAR(hire_date, 'dd/mm/yyyy') "Formato Fecha 1",
3          TO_CHAR(hire_date,'DD Month YYYY') "Formato Fecha 2",
4          TO_CHAR(SYSDATE,'dd "de" MONTH "del" yyyy hh24:mi:ss') "Fecha-Hora del Sistema"
5   FROM employees;

```

En el ejemplo, el primer formato de fecha de contrato se muestra en número (año en 4 dígitos) y separado por /. El segundo formato muestra el día y año de contrato en número y el mes en palabras. El tercer formato muestra la fecha y hora del sistema incorporando las palabras de y del (la sentencia fue ejecutada el 09 de Enero del 2014 a las 16:04).

Cuando se desean trabajar los valores numéricos como una cadena de caracteres se deben convertir esos números a un tipo de dato carácter utilizando la función TO\_CHAR. Esta función convierte un dato de tipo NUMBER a un dato de tipo

Otros elementos de formatos de números válidos:

- G: devuelve el separador de grupo en la posición especificada. Ejemplo: 9G999 Resultado: 1.234
- D: devuelve el carácter decimal en la posición especificada (defecto es punto). Ejemplo: 9999D99 Resultado: 1234,00
- V: multiplica por 10 n veces (n = número de 9s después V). Ejemplo: 99999V9999 Resultado: 12340000

```

1 -- Ejemplo
2   SELECT salary, TO_CHAR(salary,'$99,999.00') Formato1,
3          TO_CHAR(salary,'$099,999.00') Formato2,
4          TO_CHAR(salary,'$99G999000') Formato3,
5          TO_CHAR(salary,'$99999V000') Formato4
6   FROM employees
7   WHERE last_name = 'Ernst';

```

En la sentencia del ejemplo, se muestra el salario del empleado Ernest en 4 formatos diferentes. En el formato1 se muestra separado por miles y decimales (en una BD Oracle la coma es el separador de miles y punto de decimales). En el formato2 se antepondrán ceros hasta completar 4 caracteres antes del separador de miles, en el formato3 se mostrará un punto como separador de miles y una coma para los decimales y en formato4 dado que la cantidad de ceros después de V son tres, se muestra el valor del salario multiplicado por 1000.

## Funciones Anidadas

- Las funciones que operan sobre un fila se pueden anidar sin límites.
- Se evalúan desde el nivel más interno hasta el nivel más externo.

```

1 -- Ejemplo
2 SELECT last_name, department_id, UPPER(CONCAT(SUBSTR(last_name,1, 8), '_chile'))
3 FROM employees
4 WHERE department_id between 10 AND 40;

```

En el ejemplo, la función SUBSTR retorna los primeros ocho caracteres del apellido del empleado. La función CONCAT concatena el resultado de la función SUBSTR con el string '\_chile'. Finalmente la función UPPER convierte en mayúscula el resultado entregado por la función CONCAT.

## Funciones Generales

Estas funciones trabajan con cualquier tipo de datos y se relacionan con el uso de valores nulos en la lista de expresiones:

- NVL: Si el valor de expr1 es NULO , devuelve el valor de expr2. Si no es NULO retorna el valor de expr1.
- NVL2: Devuelve el valor de expr2 si el valor de expr1 no es NULO. Si expr1 es NULO devuelve el valor de expr3.
- NULLIF: devuelve NULO si expr1 y expr2 son iguales. Si no lo son devuelve el valor de expr1.
- COALESCE: retorna el valor de expr1 si no es NULO. Si es NULO devuelve el valor de expr2 si no es NULO. Si los valores de expr1 y expr2 son NULOS devuelve el valor de expr3 sino es NULO y así sucesivamente.

```

1 -- Uso de la función NVL
2 SELECT last_name, salary, NVL(commission_pct,0) "PORCENTAJE COMISION",
3       (salary*12) + (salary*12*NVL(commission_pct,0)) "SALARIO ANUAL",
4       NVL(TO_CHAR(manager_id, 'No posee Jefe') JEFE
5 FROM employees;

```

En la sentencia del ejemplo, la primera y segunda función NVL retornan un cero cuando el porcentaje de comisión del empleado sea nulo. La última función NVL retorna el string No posee Jefe cuando la identificación del jefe sea nula. debido a que se retorna un string el valor de la columna a validar se debe convertir a un string usando la función TO\_CHAR(manager\_id).

```

1 -- Uso de la función NVL2
2 SELECT employee_id, salary, commission_pct,
3       NVL2(commission_pct,'SALARIO+COMISION','SOLO SALARIO')
4       AS "SALARIO MENSUAL CORRESPONDE A"
5 FROM employees
6 WHERE employee_id IN(100,101,114,147,148,149);

```

En el ejemplo, la función NVL2 mostrará el string SALARIO+COMISION si el porcentaje de comisión no es nulo. Si el porcentaje de comisión es nulo mostrará el string SOLO SALARIO para los empleados con identificación 100, 101, 114, 147,148 o 149.

```

1 -- Uso de la función NULLIF
2 SELECT first_name, LENGTH(first_name) "Largo Nombre", last_name,
3       LENGTH(last_name) "Largo Apellido",
4       NULLIF(LENGTH(first_name),LENGTH(last_name)) "Resultado Función NULLIF"
5 FROM employees
6 WHERE employee_id IN(100,104,106,110);

```

En la sentencia del ejemplo, las funciones LENGTH retornan el total de caracteres del nombre y apellido del empleado respectivamente. Si el total de caracteres del nombre y apellido del empleado son iguales la función NULLIF mostrará NULO y si son diferentes mostrará el total de caracteres del nombre. La información se muestra para los empleados con identificación 100, 104, 106 o 110.

```

1 -- Uso de la función COALESCE
2 SELECT last_name, commission_pct, manager_id,
3       COALESCE(commission_pct,manager_id,9999) "Resultado Función COALESCE"
4 FROM employees
5 WHERE department_id IN(10,20,90);

```

En el ejemplo, la función COALESCE mostrará el valor del porcentaje de comisión si no es nulo, de lo contrario mostrará el valor de la identificación del jefe si no es nulo y si ambos valores son nulos mostrará 9999 para los empleados que trabajan en el departamento 10, 20 o 90.

## Expresiones Condicionales

Existen dos métodos que se pueden utilizar para implementar el procesamiento condicional de lógica IF-THEN-ELSE en una sentencia SQL.

```

1 -- Uso de la función CASE
2 SELECT employee_id, job_id, department_id, salary,
3       CASE job_id WHEN 'PR_REP' THEN 1.15*salary
4             WHEN 'MK_MAN' THEN 1.20*salary
5             ELSE salary END "Salario Incrementado"
6 FROM employees
7 WHERE department_id IN(70,20,110);

```

En el ejemplo, en la expresión CASE de la sentencia, el valor de la columna job\_id es la condición de búsqueda. Si el trabajo del empleado es PR\_REP el salario se mostrará incrementado en 15%, si es MK\_MAN el salario se mostrará incrementado en 20% y para el resto de los trabajos no se incrementa el salario mostrándose sólo el salario actual

```

1 -- Uso de la función DECODE
2 SELECT employee_id, job_id, department_id, salary,
3       DECODE(job_id, 'PR_REP', 1.15*salary,
4               'MK_MAN', 1.20*salary,
5               salary) "Salario Incrementado"
6 FROM employees
7 WHERE department_id IN(70, 20, 110);

```

En el ejemplo, el valor de la columna job\_id es la condición de búsqueda para la función DECODE. Si el trabajo del empleado es PR\_REP el salario se mostrará incrementado en 15%, si es MK\_MAN el salario se mostrará incrementado en 20% y para el resto de los trabajos no se incrementa el salario mostrándose sólo el salario actual. La información de muestra para los empleados que trabajan en el departamento 20, 70 o 90.

## Guía:

```

1 /* 1.- La Gerencia desea contar con información de los jefes a cargo de cada empleado. Para ello se requiere que construya una consulta que ?
2 muestre el nombre completo del empleado (nombre y apellido concatenados con un espacio en blanco) y la identificación de su jefe. Si el empleo
3 NO posee jefe se debe mostrar el mensaje NO POSEE JEFE. La información se debe mostrar según el ejemplo y ordenada en forma descendente por
4 identificación del jefe: */

```

```

5 | SELECT FIRST_NAME || ' ' || LAST_NAME empleado, NVL(TO_CHAR(MANAGER_ID), 'NO TIENE JEFE') jefe
6 | FROM EMPLOYEES
7 | ORDER BY MANAGER_ID DESC;
8 |
9 | /* 2.- Los empleados han planteado la necesidad de que se les aumente el valor de movilización mensual que se les debe pagar por ley. Por el
10 | la Gerencia ha aceptado la petición y ha definido que el valor de movilización será un porcentaje del salario mensual del empleado. Este
11 | porcentaje corresponderá por cada $1000 del salario de cada empleado es decir, si el salario del empleado es 8200 el porcentaje de aumento de
12 | movilización será de 8%, si el salario del empleado es de 15000 el porcentaje de aumento será de 15% etc. Como primera etapa se debe generar i
13 | informe que muestre la identificación del empleado, su salario actual y el porcentaje del salario que le corresponderá como movilización. Dar
14 | a cada columna de salida un alias según se muestra en el ejemplo: */
15 | SELECT EMPLOYEE_ID empleado, SALARY salario, TRUNC(SALARY/1000) "PORCENTAJE DE MOVILIZACION"
16 | FROM EMPLOYEES;
17 |
18 | /* 3.- La empresa ha decidido modificar las políticas de asignación de usuario y clave a acceso del personal a los diferentes sistemas
19 | informáticos que existen. Desde el próximo mes la política de asignación de nombres de usuarios y claves será:
20 | ? Nombre de Usuario: corresponderá a las tres primeras letras del nombre el empleado (la primera en mayúscula y las otras dos en minúsculas),
21 | seguido del largo de su nombre y de la identificación del trabajo que desempeña.
22 | ? Clave del Usuario: corresponderá al mes y año (en 4 dígitos) de contrato del empleado seguido de las dos últimas letras de su apellido en
23 | mayúsculas.
24 | Se requiere que Ud. construya una consulta que permita obtener el nombre del empleado, su apellido, identificación de su trabajo, nombre de
25 | usuario y clave. La información se requiere de acuerdo como se muestra en el ejemplo y ordenada por apellido en forma ascendente: */
26 | SELECT FIRST_NAME nombre, LAST_NAME apellido, JOB_ID trabajo,
27 |     INITCAP(SUBSTR(FIRST_NAME,1,3)) || LENGTH(FIRST_NAME) || JOB_ID nombre_usuario,
28 |     TO_CHAR(HIRE_DATE, 'MMYYYY') || UPPER(SUBSTR(LAST_NAME,-2)) "CLAVE USUARIO"
29 | FROM EMPLOYEES
30 | ORDER BY LAST_NAME;
31 |
32 | /* 4.- Se desea poder informatizar el Listado de Salarios de los empleados que hasta ahora se maneja en forma manual. El informe que Ud.
33 | desarrolle deberá mostrar la información ajustada para que visualmente se vea ordenada. Para ello, a través de una sentencia SQL implemente
34 | lo requerido según se muestra en el ejemplo de la derecha: */
35 | SELECT RPAD(LAST_NAME,20, ' ') || 'posee un salario de' || LPAD(SALARY,10, ' ') "LISTADO DE SALARIOS"
36 | FROM EMPLOYEES;
37 |
38 | /* 5.- Para poder gestionar información del personal que trabaja en la empresa, el departamento de recursos humanos de la empresa requiere
39 | contar con un informe que permita saber la fecha en que cada empleado se ha contratado. Para ello, se requiere saber el nombre, apellido y
40 | fecha de contrato (el día en palabras) de todos los empleados. La información se debe mostrar ordenada por fecha de contrato en forma
41 | ascendente y en el formato del ejemplo: */
42 | SELECT 'El empleado ' || FIRST_NAME || ' ' || LAST_NAME || ' ha sido contratado el ' ||
43 |     INITCAP(TO_CHAR(HIRE_DATE,'day')) || TO_CHAR(HIRE_DATE,'dd') || ' de ' ||
44 |     INITCAP(TO_CHAR(HIRE_DATE,'month')) || ' del ' || TO_CHAR(HIRE_DATE,'yyyy') contratos
45 | FROM EMPLOYEES
46 | ORDER BY HIRE_DATE;
47 |
48 | /* 6.- Se ha efectuado una encuesta entre los empleados de la empresa para saber cuál sería el salario que ellos consideran el ideal según
49 | el trabajo que efectúan. Con excepción del empleado que NO posee jefe, coincidieron que a ellos les gustaría ganar tres veces más de su
50 | salario actual. Con esta información, se le solicita a Ud. que genere un reporte que muestre el apellido, salario actual y el salario soñado
51 | de los empleados que desean ganar tres veces de su salario actual. La información se debe mostrar según se muestra en el ejemplo asignando
52 | además el formato indicado a los valores del salario actual y soñado: */
53 | SELECT 'El empleado ' || LAST_NAME || ' posee un salario de ' ||
54 |     TO_CHAR(SALARY, '$99,999.99') || ' pero su sueño es ganar ' ||
55 |     TO_CHAR(SALARY * 3, '$99,999.99') "SALARIO SOÑADO"
56 | FROM EMPLOYEES
57 | WHERE MANAGER_ID IS NOT NULL;
58 |
59 | /* 7.- Se desea aumentar en un 25,8% los salarios de los empleados que ganan menos de $5000. Para ello se requiere que Ud. genere un informe
60 | mostrando la identificación del empleado, identificación del departamento en el que trabaja, valor actual de su salario, el valor del reajuste
61 | (redondeado) y el salario reajustado en 25,8% (redondeado). Mostrar el informe en el formato que se muestra y la información ordenada en form
62 | ascendente por departamento y por cada departamento ordenado en forma descendente por el salario reajustado como se muestra en el ejemplo: */
63 | SELECT EMPLOYEE_ID empleado, DEPARTMENT_ID departamento, SALARY "SALARIO ACTUAL",
64 |     ROUND(SALARY * 0.258) reajuste, ROUND(SALARY + (SALARY * 0.258)) "SALARIO REAJUSTADO"
65 | FROM EMPLOYEES
66 | WHERE SALARY < 5000
67 | ORDER BY DEPARTMENT_ID, SALARY DESC;
68 |
69 | /* 8.- La Gerencia ha definido que a contar del próximo mes se pagará un bono de acuerdo al grado del trabajo que desempeña cada empleado y
70 | que se clasificará de la siguiente manera:
71 | TRABAJO GRADO
72 | AD_PRES A
73 | ST_MAN B
74 | IT_PROG C
75 | SA REP D
76 | ST_CLERK E
77 | Cuquier otro trabajo O
78 | Para ello, se requiere de un listado de los empleados con su nueva categorización según el trabajo que desempeña. La información que se
79 | solicita es nombre completo del empleado, trabajo que desempeña y grado que le corresponde por su trabajo. Se debe mostrar ordenada por
80 | apellido del empleado y en el formato que se muestra en el ejemplo: */
81 | SELECT FIRST_NAME || ' ' || LAST_NAME "NOMBRE EMPLEADO", JOB_ID "TRABAJO QUE DESEMPEÑA",
82 |     CASE JOB_ID WHEN 'AD_PRES' THEN 'A'
83 |         WHEN 'ST_MAN' THEN 'B'
84 |         WHEN 'IT_PROG' THEN 'C'
85 |         WHEN 'SA REP' THEN 'D'
86 |         WHEN 'ST CLERK' THEN 'E'
87 |     ELSE 'O' END "GRADO SEGUN SU TRABAJO"
88 | FROM EMPLOYEES
89 | ORDER BY LAST_NAME;
90 |
91 | /* 9.- La empresa desea que se automatice algunos de los procesos involucrados en el cálculo de las remuneraciones mensuales de los
92 | empleados. Para ello, Ud. deberá implementar tres rutinas de acuerdo a los requerimientos planteados:
93 | ? En un esfuerzo por mejorar las necesidades económicas de los empleados, se desea saber el costo que significaría efectuar un aumento en
94 | sus salarios. Para ello, se ha pensado que el porcentaje de reajuste corresponderá al primer dígito del salario actual de cada empleado es
95 | decir, si el empleado posee un salario de 24000 su aumento será de 2%, si su salario es de 7500 su aumento será de 7% etc. Se requiere de
96 | un reporte que muestre el nombre y apellido del empleado concatenado, salario actual y porcentaje de reajuste. Dar a cada columna de salida
97 | el alias correspondiente y formato a los valores del salario y reajuste como se muestra en el ejemplo: */
98 | SELECT FIRST_NAME || ' ' || LAST_NAME "NOMBRE EMPLEADO", TO_CHAR(SALARY, '$99,999') salario,
99 |     TO_CHAR((SALARY * SUBSTR(SALARY,1,1)/100), '$999') reajuste
100 | FROM EMPLOYEES;
101 |
102 | /* ? Implementar el cálculo de las comisiones y salario total de cada uno de los empleados considerando las siguientes especificaciones:
103 | ? Si el empleado posee porcentaje de comisión se debe mostrar, de lo contrario se debe mostrar el valor cero.
104 | ? El valor de la comisión corresponderá al valor del salario actual multiplicado por el porcentaje de comisión. Si el empleado no posee
105 | porcentaje de comisión se debe mostrar el valor cero.
106 | ? El valor salario total corresponderá a la suma del valor del salario actual más el valor de la comisión. Si el empleado no posee
107 | comisión el salario total será igual al valor del salario actual.
108 | En esta primera etapa, solo se requiere de un listado que muestre la identificación del empleado con el alias, el valor del salario actual,
109 | el porcentaje de comisión, el valor de la comisión calculada y el valor del salario total calculado. La información se debe mostrar en el
110 | formato que se muestra en el ejemplo: */

```

```
111 | SELECT EMPLOYEE_ID "ID EMPLEADO", SALARY "SALARIO SIN COMISION",
112 |       NVL(COMMISSION_PCT,0) "PORC. COMISION",
113 |       NVL2(COMMISSION_PCT,SALARY*COMMISSION_PCT,0) "VALOR COMISION",
114 |       NVL2(COMMISSION_PCT,SALARY+(SALARY*COMMISSION_PCT),SALARY) "SALARIO TOTAL"
115 | FROM EMPLOYEES;
116 |
117 | /* ? La Gerencia ha decidido que a contar del mes Mayo a los empleados cuyo salario esté entre 1000 y 5000 se les aumentará el salario
118 | según los años que lleva trabajando en la empresa. Así por ejemplo, si el empleado lleva trabajando 9 años su salario se aumentará en un 9%
119 | si lleva trabajando 15 años su salario aumentará en un 15%, etc. Para ello, se requiere almacenar en la tabla COMISION la identificación de
120 | empleado, la fecha de contrato, salario actual, los años que lleva contratado en la empresa según el año actual y el valor del aumento
121 | (redondeado) para cada empleado. Dar a cada columna de la tabla el nombre correspondiente según se muestra en el ejemplo (la sentencia se
122 | ejecutó el año 2014): */
123 | SELECT EMPLOYEE_ID "ID EMPLEADO", HIRE_DATE "FECHA CONTRATO", SALARY salario,
124 |        ROUND(MONTHS_BETWEEN(SYSDATE,HIRE_DATE)/12) "AÑOS CONTRATADOS",
125 |        ROUND(MONTHS_BETWEEN(SYSDATE,HIRE_DATE)/12)*SALARY/100 "AUMENTO"
126 | FROM EMPLOYEES
127 | WHERE SALARY BETWEEN 1000 AND 5000;
```

Publicado por Sebastián Catalán A en 8:09



Etiquetas: PL/SQL

## 1 comentario:



Mauro Bravo 25 de mayo de 2016, 22:50

Me sacó algunas dudas por ahí. Gracias

[Responder](#)

Introduce tu comentario...

Comentar como: Mir2 (Google)

[Cerrar sesión](#)

[Publicar](#)

[Vista previa](#)

[Avisarme](#)

[Entrada más reciente](#)

[Página principal](#)

[Suscribirse a: Enviar comentarios \(Atom\)](#)

### Administrador



Sebastián Catalán A

[Ver todo mi perfil](#)

### Archivos del blog

- ▼ 2015 (13)
  - mayo (5)
  - abril (2)
  - ▼ marzo (6)
    - [Creando Vistas en la Base de Datos \(PL/SQL\)](#)
    - [Uso de Subconsultas para Resolver Consultas \(PL/S...\)](#)
    - [Mostrando Datos de Múltiples Tablas \(PL/SQL\)](#)
    - [Experiencia de Aprendizaje StringDecisionIteration...](#)
    - [Uso de Funciones de Grupo \(PL/SQL\)](#)
    - [Uso de funciones de una fila \(PL/SQL\)](#)