CONSULTAS SQL: SENTENCIA SELECT

Bases de Datos

18 DE FEBRERO DE 2025

ERIK AMO TOQUERO

Contenido

Sentencia de Consultas:

```
SELECT [ALL|DISTINCT] [COLUMNA 1, COLUMNA2, EXPRESION1 [AS NOMBRE],
EXPRESION2…]
FROM [NOMBRETABLA1, NOMBRETABLA2…]
[WHERE CONDICION1 OR|AND CONDICION2…]
[ORDER BY EXPRESION|COLUMNA [DESC|ASC]];
```

- FROM especifica las tablas de las que se sacarán los datos
- WHERE pondrá una serie de condiciones a cumplir en la consulta
- ORDER BY se encarga de ordenar en función de las expresiones elegidas (por defecto en descendiente]
- ALL recupera todas las filas (viene por defecto)
- DISTINCT recupera filas evitando repeticiones de estas. Es decir, si tenemos 2 filas iguales solo nos sacará 1
- AS permitirá poner un nombre a la tabla. Estará en mayúsculas salvo si va entrecomillado (“nombre”)

Operadores

- Aritméticos: ○ Suma + ○ Resta – ○ Multiplicación * ○ División /
- Lógicos y comparativos ○ Igual a = ○ Mayor que > ○ Mayor o igual > ○ Menor que < ○ Menor o igual <= ○ Distinto de <> || !=
  - ○ AND –Devuelve true si las dos condiciones son verdaderas ○ OR –Devuelve true si una de ellas como mínimo es verdadera ○ NOT –Devuelve false si la condición es verdadera

Operador LIKE:

Utiliza caracteres especiales para realizar búsquedas y comparaciones en cadenas:

- ‘%’ representa una cadena con cualquier tipo de carácter.
- ‘-‘ representa un solo carácter cualquiera

NULL y NOT NULL

Comprueba si la columna en esa posición tiene valor (NOT NULL), o no (NULL)

(IS NULL | IS NOT NULL)

IN Y BETWEEN

Para rangos de valores se puede utilizer los operadores IN y BETWEEN

- IN: Por medio de una lista de valores (o una subconsulta) se comprueba si el valor está dentro de la lista
- BETWEEN: Comprueba si el valor se encuentra entre dos valores (BETWEEN VALOR1 AND VALOR2)

SUBCONSULTAS:

Se pueden usar consultas dentro de consultas para obtener unos datos de forma muy precisa, como por ejemplo el Nombre de la gente que tiene la misma nota que Alvaro:

```
SELECT NOMBRE
FROM ALUMNOS
WHERE NOTA = (SELECT NOTA
    FROM ALUMNOS
    WHERE NOMBRE LIKE 'Alvaro');
```

Condiciones en subconsultas:

- Comparación en subconsultas (=>, <=, <, >, !=, =)
- Test de pertenencia (IN)
- Test de existencia (EXISTS) –Devuelve verdadero si la subconsulta concatenada no está vacía
- Test de comparación cuantificada (ANY, ALL)

ALL, SOME y ANY

ALL establece una condición sobre todos los valores. Se tiene que utilizar en comparaciones como si queremos algo que sea mayor a todo lo que hay en una subconsulta WHERE valor>ALL (SELECT…)

SOME y ANY se usan para ver si la condición cumple alguno de los valores que devuelven, esto rompe lo que se llama el "Igual Restricto". Si no se usa el ANY o SOME con un =, en el momento en el que haya más de un resultado de la subconsulta, el código se romperá.

Subconsultas Sincronizadas:

Pongamos el caso del EXISTS: Queremos sacar los empleados que pertenezcan al departamento Marketing. La siguiente forma saca una lista de empleados que cumplan con la condición de que exista un departamento que tenga el mismo código de dpt que ellos y su nombre sea 'Marketing'

```
SELECT *  FROM EMPLEADOS e
WHERE EXISTS ( SELECT * FROM DEPARTAMENTOS d
                WHERE d.NOMDPT = 'Marketing'
                and e.COD_DPT = d.COD_DPT);
```

Lo que está haciendo es una subconsulta sincronizada, está cogiendo datos de la consulta principal y los está usando en la subconsulta.

Consultas de varias tablas

Funciona igual que las consultas normales, pero se añaden los datos de una más. Es muy recomendable etiquetar cada tabla para poder usar más fácilmente sus columnas.

Importante cruzar tablas por medio de una columna que tengan en común, si no, el resultado no será el esperado.

```
SELECT (a.COL1, a.COL2, b.COL1)
FROM TABLA1 a, TABLA2 b
WHERE (a.COL1 = b.COL1);
```

FUNCIONES:

- Aritméticas ○ ABS(n) devuelve el valor absoluto de n ○ CEIL(n) devuelve el valor entero superior o igual a n ○ FLOOR(n) devuelve el valor inferior o igual a n ○ MOD(m,n) devuelve el resto de la división m/n ○ NVL(valor, exp) si valor es nulo, lo transforma en exp
    - POWER(m, n) devuelve n elevado a la N
    - ROUND(n, [m]) redondea a las m decimales. Si es 0 o no se pone, será al propio número. Si es negativo, contará a las decenas, centenas…
    - SIGN(n) devuelve 1 si es positivo y -1 si es negativo ○ TRUNC(n, [m]) trunca n a las m decimales.
    - SQRT(n, m) hace la raíz $\sqrt[m]{n}$
- Grupo de valores ○ AVG(columna) calcula la media de todas las columnas de "n" ignorando nulos ○ COUNT(*|columna) calcula el número de veces que n no es nulo.
    - MAX(columna) calcula el valor máximo de la columna ○ MIN(columna) calcula el valor mínimo de la columna
    - SUM(columna) calcula la suma total de los valores de la columna
- Listas ○ GREATEST(valor1, valor2…) saca el mayor de la lista ○ LEAST(valor1, valor2…) saca el menor de la lista - Cadena de caracteres (devuelven valores de carácter):
    - CHR(n) Devuelve el carácter cuyo valor es igual a "n" ○ CONCAT(cad1, cad2) concatena cadenas, también sirve ||
    - LOWER(cad) minúsculas ○ UPPER(cad) mayúsculas
    - INITCAP(cad) la primera letra en mayúscula, el resto en minúsculas ○ LPAD(cad, n) añade caracteres a la izquierda hasta llegar a n caracteres ○ RPAD(cad, n) añade caracteres a la derecha hasta llegar a n caracteres ○ LTRIM(cad) Si la cadena tiene blancos a la izquierda, los quita ○ RTRIM(cad) Si la cadena tiene blancos a la derecha, los quita
    - REPLACE(cad, cadena_busqueda) sustituye caracteres de una cadena con 0 o más ○ SUBSTR (cad, m, [n]) borra una parte de la cadena, empezando a escribir desde la posición m y acabando en la n (si se pone la n)

    - TRANSLATE(cad1, cad2, cad3): Convierte caracteres de una cadena en otros de acuerdo a un plan de sustitución

- Cadena de caracteres (devuelven números) ○ ASCII(cad) devuelve la primera letra de la cadena en ASCII
    - INSTR(cad1, cad2 [, comienzo, [m]]), devuelve la posición donde coincide un patrón de la cadena 1 y la cadena 2
    - LENGTH(cad) devuelve la longitud de la cadena.
- Conversión ○ TO_CHAR(numero|fecha, 'formato'), convierte a varchar2 un numero o fecha según un formato
    - TO_DATE(cad, 'formato') convierte una cadena a tipo DATE ○ TO_NUMBER(cadena [, formato] transforma una cadena a número
- Otras ○ USER: Devuelve el nombre del usuario ○ UID: Devuelve el UID ○ SYSDATE: Devuelve la fecha actual según el sistema

Agrupamiento "GROUP BY"

```
SELECT …
FROM …
GROUP BY COL1, COL2…
--Sirve para agrupar los resultados en función de un valor. Es importante meter todos los valores que no se vayan a agrupar, es decir, que si vamos a usar una suma de valores y un código normal, hay que meter en el order el código
HAVING (CONDICION)
--Al igual que las consultas normales tienen el WHERE, el HAVING es un condicional de GROUP. Se utiliza como condición de las filas a las que le afecta el agrupamiento (es decir, lo que hay dentro de la función grupal)
```

Cruces de Tablas

Cuando se necesita hacer una consulta con dos tablas relacionadas, no sirve poner un from con las dos tablas y tirar, ya que se produce lo que se llama un **producto cartesiano** (saca los datos de cada tabla tantas veces como posibles ocurrencias tenga).
Se realiza lo que se llama **cruce de tablas**, que no es más que relacionarlas mediante una condición. Este cruce se puede hacer de dos formas

Cruce por Where

```
SELECT t.valor1, q.valor2
FROM TABLA1 t, TABLA2 q
WHERE (t.valor1 = t.valor2)
--Aquí lo que se está haciendo es igualar dos valores de dos tablas, por lo que muestra los elementos de las dos tablas que comparten ese valor en común.
```


Cruce por Join

El método Join se utiliza de la misma forma del cruce de tablas, pero tiene una matiz que no tiene el cruce de tablas: La **aparición de nulos.**

SELECT TABLA1.valor1, TABLA2.valor1

FROM TABLA1 [tipo] JOIN TABLA2 ON (condición)

[WHERE …]

Hay 4 joins distintos:

- **(Inner) Join:** Incluye los elementos que aparecen en la relación entre tabla1 y tabla2
- **Left Join:** Aparecen todas las filas de la tabla1, estén o no relacionadas con la tabla2. Los campos de la tabla1 que no estén relacionados con la tabla2 pondrá "Null" en los valores que vengan de la tabla2
- **Right Join:** Aparecen todas las filas de la tabla2, estén o no relacionadas con la tabla1. Los campos de la tabla 2 que no estén relacionados con la tabla 1 pondrán "Null" en los valores que vengan de la tabla 1.
- **Outer/Full Join:** Aparecen todas las filas de ambas tablas, estén o no relacionadas entre ellas. Las que están relacionadas aparecen juntas en la misma fila, mientras que las no relacionadas funcionarán como el Right y el Left por separado.