Teoría de prácticas de Bases de Datos. Curso 2016/2017

Manuel Jesús Corbacho Sánchez

17 de julio de 2017

1. Manipulación de datos

El formato de consultas corresponde al SGBD SQL+ y pueden variar según el SGBD usado.

1.1. Formato de consultas.

```
SELECT [distinct ]columna1 "nombre1", columna2 "nombre2", ..., columnaX "nombreX"
FROM tabla1 variable1,tabla2 variable2, ..., tablaX variableX
[WHERE condiciones]
[GROUP BY nombresdecolumnas]
[HAVING condicionesdeselecciondegrupos]
[ORDER BY nombredecolumna1 [asc|desc],nombredecolumna2 [asc|desc] ];
```

1.2. SELECT

Es el equivalente en SQL a la proyección(π) del álgebra relacional.
La orden *SELECT* se utiliza para realizar consultas a una o multiples tablas o vistas.
La clausula from indica las tablas o vistas a las que pertenecen las tuplas poyectadas.
SELECT y *FROM* aparecerán en todas las consultas de SQL.
Para eliminar las tuplas repetidas en el select usaremos la clausula *DISTINCT*.

1.3. WHERE

Es el equivalente en SQL a la selección(σ) del álgebra relacional.
En la proyección nos permite seleccionar un conjunto de tuplas en lugar de toda la tabla.
Estas tuplas cumplen las condiciones especificadas en el *WHERE*.
Las condiciones toman valor *verdadero* o *falso* y se pueden combinar con *NOT*, *OR* y *AND*.

1.3.1. Operadores de comparación

- Comparaciones con un valor: $\{=, !=, <, >, <=, >=\}$
- Comparaciones con un patrón: $\{LIKE\}$
- Comprobar pertenencia a intervalo: $\{BETWEEN\}$
- Comprobar pertenencia a un conjunto de valores: $\{IN\}$
- Comparación *TODOS* o *ALGÚN*: $\{ALL, ANY\}$
- Comprobar indeterminación: $\{NULL\}$
- Comprobar existencia: $\{EXISTS\}$

Las columnas del **WHERE** deben pertenecer a las tablas de la clausula **FROM**.
Los operadores pueden invertir su valor con anteponiendo *NOT*.
IMPORTANTE: aunque *LIKE* puede funcionar como = es una mala práctica usarlo para ello, no confundir cadenas de carácteres con patrones, aunque un patrón es del tipo varchar2, así que se escribe entre comillas simple, un patrón usa %(equivale a cualquier cadena) o _ (equivale a 1 caracter) .

1.3.2. ANY y ALL

Los operadores *ANY* y *ALL* son equivalentes a "algún" y "para todos" respectivamente, permite comparar un valor con una lista, además se dan las siguientes equivalencias:

1. $IN \Leftrightarrow = ANY(...)$
2. $NOT\ IN \Leftrightarrow != ALL(...)$

1.4. ORDER BY

La clausula tiene la siguiente forma:

```
ORDER BY {columna| entero| alias} [ASC|DESC]
```

Por defecto el orden es *ASC*.

Las columnas de ordenación deben pertenecer a las tablas especificadas en el *FROM* o bien aparecer en el *SELECT*, si se usa *DISTINCT*, las columnas deben obligatoriamente aparecer en el *SELECT*.

2. Expresiones y Funciones

2.1. GROUP BY

Esta clausula nos sirve para agrupar las tuplas que contienen una o varias columnas iguales.

Las columnas que no aparezcan en el *GROUP BY* no pueden aparecer en el *SELECT*, salvo como argumentos de una función de grupo.

2.2. HAVING

La clausula *HAVING* equivale a la clausula *WHERE* pero aplicada a grupos. Permite filtrar los grupos generados por la clausula *GROUP BY*.

Las tuplas devueltas por los grupos que no cumplan las condiciones del *HAVING* no apareceran en el resultado del *SELECT*.

2.3. Funciones de grupo

- **COUNT(*|distinct|all expr)**: devuelve el nº de tuplas, si se especifica expr solo cuenta aquellas no nulas(se deberá usar nvl para aquellas tuplas en las que expr sea nula si se quieren añadir).
- **AVG([distinct] n)**: devuelve la media de la columna seleccionada
- **MAX([distinct] n)**: devuelve el valor máximo de n.
- **MIN([distinct] n)**: devuelve el valor mínimo de n.
- **SUM([distinct] n)**: devuele el sumatorio de n.

2.4. Función NVL

Esta función sirve para asignar un valor a aquellas columnas de una tupla cuyo valor sea nulo.
Ejemplo de uso:

```
select nvl(clt_pais,'e') from clientes;
```

En este ejemplo aquellas columnas donde clt_pais sea nulo, se asignará el valor 'e'.

2.5. Funciones numéricas

- **abs(n)**: valor absoluto de n.
- **cos(n)**: coseno de n.
- **sin(n)**: seno de n.
- **tan(n)**: tangente de n.
- **sqrt(n)**: raiz cuadrada de n.

- **mod(m,n)**: m %n.
- **floor(n)**: parte entera por exceso.
- **ceil(n)**: parte entera por defecto.
- **round(n)**: redondea n.
- **trunc(m[,n])**: trunca m a n decimales(0 por defecto).
- **power(m,n)**: m^n.

3. Otros operadores

3.1. Producto Cartesiano($\times$)

Se realiza automaticamente al escribir varias tablas en el *FROM*, ejemplo:

```
SELECT * FROM ventas,articulos;
```

Nos devuelve el producto cartesiano de ventas y articulos.

3.2. Producto Theta($\bowtie_\theta$)

Si se añade la clausula *WHERE* al producto cartesiano, esta actua sobre el producto cartesiano de ventas y articulos, dando como resultado un producto theta. Ejemplo:

```
SELECT * FROM ventas,articulos WHERE art_num=3;
```

3.3. Producto Natural($\bowtie$)

En SQL el comando es *NATURAL JOIN*.
Ejemplos:

```
SELECT * FROM articulo NATURAL JOIN proveedores;
```

En realidad daría error, al no llamarse igual las columnas art_prv(de la tabla Articulos) y prv_num(de la tabla Proveedores).
Sería equivalente a la siguiente consulta si las columnas comparadas se llamaran igual:

```
SELECT * FROM articulo,proveedores WHERE art_prv = prv_num;
```

3.4. Operaciones de Conjuntos

- Unión: se utiliza el operador *UNION*
- Intersección: se utiliza el operador *INTERSECT*
- Resta: se utiliza el operador *MINUS*

3.5. Autouniones

Se utiliza para comparar una tabla consigo mismo, cuando una columna se va a comparar consigo misma, para ello usaremos alias en el from.
Por ejemplo, para saber los clientes que viven en los mismos paises que los clientes cuyo nombre empieza por m, usaríamos la siguiente consulta:

```
1 select t1.clt_num numero, t1.clt_nom nombre, t1.clt_apell
2 from clientes t1, clientes t2
3 where t1.clt_pais = t2.clt_pais and t2.clt_nom like 'm%'
4 and t2.clt_num > 10;
```

3.6. Unión Externa

En inglés *outer-join*, en *SQL+* se escribe (+). La unión externa se utiliza para resolver consultas del tipo:
Se desea obtener una lista de filas en función de un criterio que es función de la unión con contra tabla. si la condición no se cumple, normalmente la tupla no aparece, si queremos que estas tuplas aparezcan, debemos usar una unión externa.

Ejemplo de uso:

```
1 select art_num numero, art_nom nombre,
2 to_date(vnt_fch,'yymmdd') fecha
3 from articulos,ventas
4 where art_num = vnt_art(+) and art_num > 3;

    NUMERO NOMBRE               FECHA
---------- -------------------- --------
         4 lampara              06/01/91
         4 lampara              09/01/91
         5 lampara
         6 lampara              11/01/91
         7 lampara
         8 pesacartas 1-500
         9 pesacartas 1-1000    11/01/91
        10 boligrafo            06/01/91
        10 boligrafo            11/01/91
        11 boligrafo            06/01/91
        12 boligrafo lujo       10/01/91
        13 boligrafo lujo       10/01/91
        13 boligrafo lujo       22/02/92
        14 boligrafo lujo       06/01/91
        15 boligrafo lujo       09/01/91
        15 boligrafo lujo       09/05/01
```