

PRÁCTICA: 4. LENGUAJE DE CONSULTAS (SELECT) - 2**OBJETIVOS:**

Conocer cómo se realiza la autocomposición de tablas. Conocer otras partes de la sentencia SELECT: ORDER BY, GROUP BY y HAVING

MATERIAL:

ORACLE9 versión para WINDOWS XP

BIBLIOGRAFIA:

GUÍA DE SQL
JAMES R. GROFF, PAUL N. WEINBERG
MCGRAW-HILL, 1998

ORACLE 9I: GUIA DE APRENDIZAJE
ABBEY, MICHAEL Y COREY, MIKE Y ABRAMSON, IAN
MCGRAW-HILL / INTERAMERICANA DE ESPAÑA, S.A.

1. Autocomposición

Es una composición de una tabla consigo misma. Para poder realizarla, obligatoriamente hay que emplear alias dentro de la orden SELECT que eviten los problemas de ambigüedad:

Por ejemplo, si tenemos la tabla paciente:

```
Paciente(nro, nombre, direccion, medico)
```

```
Medico → paciente
```

```
CREATE TABLE paciente(  
Nro INTEGER PRIMARY KEY,  
Nombre VARCHAR2(40),  
Direccion VARCHAR2(40),  
Medico INTEGER REFERENCES paciente);
```

```
INSERT INTO paciente VALUES (4, 'Lucía', 'Alarcos', NULL);  
INSERT INTO paciente VALUES (1, 'Paco', 'Alarcos', 4);  
INSERT INTO paciente VALUES (2, 'Luis', 'Azucena', 4);  
INSERT INTO paciente VALUES (3, 'Maria', 'Postas', 4);
```

Si quisiéramos obtener el nombre de los médicos que viven en la misma dirección que sus pacientes, tendríamos que utilizar una autocomposición. Para ello, como ya hemos dicho, tendremos que utilizar alias para el select:

```
SELECT doctor.nombre  
FROM paciente doctor, paciente enfermo  
WHERE doctor.nro=enfermo.medico  
AND doctor.direccion=enfermo.direccion;
```

La salida sería:

```
NOMBRE  
-----  
Lucía
```

2. Cláusula ORDER BY

Permite ordenar el resultado de la consulta.

```
ORDER BY <nombre de columna> [ASC | DESC] [...]
```

DESC: los valores de la columna afectada estarán en orden descendente.

Por ejemplo, si queremos saber el nombre y las direcciones de los pacientes (que no son médicos), ordenados alfabéticamente por el nombre:

```
SELECT nombre, direccion
FROM paciente
WHERE medico IS NOT NULL
ORDER BY nombre;
```

La salida sería:

NOMBRE	DIRECCION

Luis	Azucena
Maria	Postas
Paco	Alarcos

3. Cláusula GROUP BY

Se utiliza cuando se quieren realizar consultas agrupadas por una columna. Así, se puede pensar en estos conjuntos de filas como grupos, utilizándose las siguientes funciones (que también pueden ser utilizadas sin GROUP BY):

avg(nombre_columna) → Valor medio de todos los valores de nombre_columna

count(*) → Número de filas de la tabla

max(nombre_columna) → Valor máximo almacenado en nombre_columna

min(nombre_columna) → Valor mínimo almacenado en nombre_columna

Por ejemplo, si tengo una tabla clientes, que representa a todos los comercios que pertenecen a una misma cadena de ámbito nacional con la estructura y el contenido siguientes:

Id	Nombre	cant_ventas	fecha	ciudad
1	Pepe	100	12/01/99	Toledo
2	Paco	200	13/08/98	Teruel
3	Lolo	400	1/6/78	Toledo
4	Lola	500	1/5/90	Toledo
5	Pepa	600	3/5/98	Teruel
6	Paca	200	2/7/97	Cuenca
7	Lali	100	6/5/99	Cuenca

Si quiero obtener la media de las ventas realizadas por la cadena puedo obtenerlo con un select sobre toda la tabla:

```
select avg(cant_ventas) from clientes;
```

Obteniendo como salida:

```
avg(cant_ventas)
```

```
300
```

pero si lo que quiero son las ventas totales agrupadas por ciudades, entonces se debe realizar un agrupamiento por la columna ciudad.

```
select avg(cant_ventas) from clientes group by ciudad;
```

Obtendré como salida:

```
avg(cant_ventas)
```

```
900
```

```
1000
```

```
300
```

Hay que tener en cuenta que hay que agrupar todas las columnas que no aparezcan mencionadas en la cláusula group_by.

Por ejemplo:

```
select nombre, ciudad, sum(cant_ventas) from clientes group by nombre;
```

daría error ya que ciudad no está incluido en el group by. Correctamente sería:

```
select nombre, ciudad, sum(cant_ventas) from clientes group by nombre, ciudad;
```

4. Cláusula HAVING

No se pueden usar funciones de agrupación en la cláusula WHERE de un SELECT. O sea, no se puede usar el WHERE para, de forma selectiva eliminar datos que no interesan del resultado de una consulta agrupada.

Por ejemplo, en la tabla

```
exámenes(id_asignatura, nro_estudiante, nota, fecha)
```

Si hacemos:

```
SELECT nro_estudiante, avg(nota)
FROM exámenes
WHERE avg (nota) >6
GROUP BY nro_estudiante;
```

Darían un error por usar una función de agrupamiento en el WHERE

La cláusula HAVING hace una función parecida a la del WHERE cuando se trabaja con este tipo de funciones. Así, para listar aquellos alumnos cuya media es mayor que 6 sería:

```
SELECT nro_estudiante, avg(nota)
FROM exámenes
GROUP BY nro_estudiante
HAVING avg (nota) > 6;
```

El campo referenciado en la cláusula HAVING no puede tener más de un valor por grupo. Esto significa que, en la práctica, HAVING sólo puede referenciar a funciones de agregación y columnas que se están usando en el GROUP by

EJERCICIOS

1. Crear una tabla EMPLEADO que contenga los siguientes campos:

empleado (cod, nombre, apellido, calle, jefe)
jefe → empleado

Rellenar la tabla.

Obtener el nombre y apellido de un empleado y su jefe siempre y cuando ambos tengan el mismo apellido.

Borrar la tabla empleado.

2. Obtener el número de profesores en activo de cada área según su categoría.

3. Obtener el número de profesores en activo de departamento y área según su categoría

4. Listar el código y nombre de cada departamento y el número de profesores numerarios en activo que tiene.

5. Obtener todos los posibles tríos de profesores del mismo área de conocimiento. Tener en cuenta que (a,b,c)=(c,b,a).