TEMA 9 PARTE 2.
LENGUAJE DE CONSULTA DE DATOS (DQL)

9.4. OPERADORES SQL

Símbolos y palabras reservadas empleadas para especificar una acción a ser realizada en una o más expresiones llamadas operandos o argumentos.

9.4.1. Tipos de operadores:

- Unarios: `<operador><operando>`
- Binarios: `<operando><operador><operando>`
- Operadores aritméticos: `+,-,*,/,%`
- Operadores de concatenación:
  - `||` (SQL estándar, Oracle, DB2)
  - `+` (SQL Server)
  - `CONCAT` DB2

9.4.2. Ejemplos aritmética con fechas en Oracle

La unidad de medida para realizar aritmética con fechas en Oracle es de un día:

```
select sysdate, sysdate+10 as resultado
from dual;
```

```
SYSDATE             RESULTADO
=================== ===================
2010-11-22 23:16:25 2010-12-02 23:16:25
```

Agregar 2 horas a la fecha actual:

```
select sysdate, sysdate + (2/24) as resultado
from dual;
```

```
SYSDATE             RESULTADO
=================== ===================
2010-11-22 23:17:42 2010-11-23 01:17:42
```

9.4.3. Operadores lógicos:

Empleados para evaluar un conjunto de condiciones, el resultado es un valor booleano.

Operador	Descripción
all	Evalúa a true si todo el conjunto de comparaciones evalúa a true
and	Evalúa a true si ambas expresiones son verdaderas
any	Evalúa a true si alguna de las comparaciones evalúa a true (similar a SOME)
some	Evalúa a true si alguna(s) de las comparaciones evalúa a true. (similar a ANY)
between	Evalúa a true si el operando está dentro de un rango de valores.
exists	Evalúa a true si un subquery regresa algún valor (contiene al menos un registro)
in	Evalúa a true si el operando es igual a algún valor de una lista de valores o expresiones.
like	Evalúa a true si el operando hace match con algún patrón
not	Negación de un operador booleano
or	Evalúa a true si alguna expresión booleana es verdadera.

Ejemplos:

- Seleccionar los **puestos** que tengan sueldo superior a 8000, 10000 y 12000

```
select *
from puesto
where sueldo_tabulador > all(8000,10000,12000);
```

- Seleccionar todos los productos donde su tipo sea B, D, ó A

```
select *
from producto
where tipo in ('a','b','d');
```

- Seleccionar a todos los empleados que hayan nacido en 1980

```
select *
from empleado
where fecha_nacimiento
between to_date('01-01-1980','dd-mm-yyyy')
and to_date('31-12-1980','dd-mm-yyyy');
```

9.4.4. Like

Se emplea cuando la condición de la cláusula `where` es parcialmente conocida. Emplea los llamados "wildcards" para especificar la parte de la condición desconocida:

Wildcard	Descripción
%	Representa cualquier cadena de 0 o más caracteres
_ (guión bajo)	Representa cualquier carácter

Ejemplos:

```
select  * from cliente where  nombre like 'WILL%'
select  * from cliente where  nombre like '%ON'
select  * from cliente where  nombre like 'WILE%_EAL%'
select  * from cliente where  nombre like '_ _ _E% _ES%'
```

9.5. FUNCIONES SQL

Clasificación:

- Funciones numéricas (matemáticas)
- Funciones de cadenas
- Funciones de fechas y horas
- Funciones de agregación
- Funciones de conversión de tipos de datos.
- Funciones de sistema
- Funciones definidas por el usuario

9.5.1. Funciones de agregación.

Funciones que realizan cálculos o estadísticas sobre algún conjunto de datos. El SQL estándar define una gran variedad de estas funciones. Las más comunes se describen a continuación:

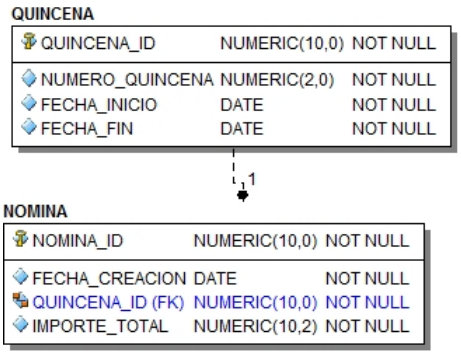
Función	Descripción
count	Cuenta el número de registros
min	Obtiene el valor menor de un atributo
max	Obtiene el valor mayor de un atributo
sum	Realiza la suma aritmética de un conjunto de valores
avg	Calcula el promedio de un conjunto de valores.

9.5.2. La función sum

- Realiza la suma aritmética de los valores de la columna especificada como parámetro.

Ejemplo:

Calcular el importe total de las nóminas generadas en el mes de enero del 2010




```
select sum(importe_total)
from nomina n,quincena q
where n.quincena_id=q.quincena_id
and fecha_inicio >= to_date('01/01/2010','dd/mm/yyyy')
and fecha_fin <= to_date('31/01/2010','dd/mm/yyyy');

SUM(IMPORTE_TOTAL)
==================
10563371.77
```

9.5.3. La función count.

Realiza un conteo de los registros de una tabla.

Ejemplo:

Seleccionar el número de empleados existentes en la empresa

```
select count(*)
from empleado;
```

- Observar que la función `count` recibe un parámetro. Este parámetro corresponde con el nombre de una columna o el valor '*'
- La diferencia de emplear * o el nombre de una columna es la siguiente:
  - '*' realiza el conteo de registros sin considerar los valores de las columnas.
  - Si se especifica el nombre de una columna, la función `count` realiza un conteo de todos los registros descartando aquellos que tengan un valor nulo para la columna especificada.

Ejemplos:

Suponer la siguiente tabla de datos:

EMPLEADO

Emp_id	nombre	Email
1	Juan	
2	Jorge	Jorge @mail.com
3	Mario	
4	Eva	

- Generar sentencias que muestren el número total de registros.

```
select count(*) from empleado;
select count(emp_id) from empleado;
select count(nombre) from empleado;
```

- Observar que las 3 instrucciones anteriores regresan en mismo resultado: 4 registros. Lo anterior e debe a que las 4 columnas están definidas como `not null`.
- Para estos casos en donde no se requiere distinguir entre valores nulos, lo correcto es usar `count(*)`

Ejemplo:

Mostrar el número de empleados que cuentan con correo electrónico.

```
select count(email) from empleado;
```

En este caso, la función `count` regresará el valor 1 ya que excluye a todos los registros que no tengan valor para el campo email.

9.5.4. La función max.

- Obtiene el valor mayor de una columna. Si existen varios registros con el máximo valor, la función regresa un solo valor.

Ejemplos:

Funciones empleadas como condiciones:

Seleccionar los datos del puesto mejor pagado:

```
select *
from puesto
where sueldo_tabulador =
(select max(sueldo_tabulador)
 from puesto);
```

Seleccionar al empleado más joven:

```
select *
from empleado
where fecha_nacimiento =
(select max(fecha_nacimiento)
 from empleado);
```

9.5.5. Agrupación de registros:

9.5.5.1. Group By

Se utiliza normalmente cuando en la lista de columnas en la instrucción `select` existen funciones de agregación, se emplean para obtener información como la siguiente:

- Determinar el precio mínimo para cada tipo de producto
- Generar el precio promedio de los productos de un determinado código.

Como su nombre lo indica, `GROUP BY` se aplica para realizar cálculos sobre grupos de datos. Estos cálculos los realiza una función de agregación.

Sintaxis:

```
select <column-list>
from <table-list>
[where <condition-list>]
[group by <column-list>]
[having <column-list>]
[ order by <column-list> [asc|desc] ]
```

Reglas importantes para el uso de `group by`

- Se emplea en conjunto con funciones de agregación.
- La lista de columnas de la cláusula `group by` corresponden a todas las columnas que no usan una función de agregación.

Considerar los siguientes datos del inventario de una tienda de libros:

LIBRO_ID	NOMBRE	CLASIFICACION	STATUS	PRECIO
1	FABULAS DIVERSAS	A	COMPRADO	89.45
2	PRINCIPITO	A	COMPRADO	45.98
3	CRONICAS PARA INFANTES	A	VENDIDO	189.45
4	FISICA I	B	COMPRADO	289.45
5	PRINCIPIOS DE QUIMICA	B	DEFECTUOSO	345.98
6	BIOLOGIA I	B	COMPRADO	442.45
7	EL ARTE DE COCINAR	C	AGOTADO	201.45
8	DECORACION DE INTERIORES	C	DEFECTUOSO	145.98
9	CURSO DE COSTURA	A	AGOTADO	133.45

Ejemplo:

Mostrar el precio promedio para cada clasificación de libros existente en el inventario.

Observar que, en este caso, se empleará la función `avg` para calcular el promedio. Dicho promedio se debe calcular para cada "**grupo**" de libros que pertenecen a una misma clasificación (A, B o C), por lo tanto, se debe emplear `group by` empleando el campo `clasificacion` como criterio de agrupamiento. Por cada valor diferente de dicho campo se genera un grupo nuevo, y por cada grupo, se aplica la función `AVG`:

Clasificacion	group by
A	Grupo 1, se aplica AVG a los precios de estos 4 registros
A	
A	
A	
B	Grupo 2, se aplica AVG a los precios de estos 3 registros
B	
B	
C	Grupo 3 Se aplica AVG a los precios de estos 2 registros
C	

```
select clasificacion, avg(precio)
from inventario_libro
group by clasificacion
```

```
CLASIFICACION AVG(PRECIO)
============= ======================================
A             114.5825
B             359.29333333333333333333333333333333333
C             173.715
```

Ejemplo:

Generar una sentencia SQL que muestre el número de libros `COMPRADOS`, `VENDIDOS`, `AGOTADOS`, etc., (todos los estatus existentes).

En este caso el criterio de agrupación es el `status`. A cada grupo se le aplica la función `count`.

Status	Group by
AGOTADO	Grupo 1
AGOTADO	
COMPRADO	Grupo 2
COMPRADO	
COMPRADO	
COMPRADO	
DEFECTUOSO	Grupo 3
DEFECTUOSO	
VENDIDO	Grupo 4

```
select  status,count(*)
from inventario_libro
group by status;

STATUS    COUNT(*)
========== ========
VENDIDO    1
DEFECTUOSO 2
AGOTADO    2
COMPRADO   4
```

Ejemplo :

Generar una sentencia SQL que muestre el número de libros existentes con base a su clasificación y a su estatus.
Lo que nos dice este ejemplo, es que ahora se requiere hacer un conteo formando grupos empleando los campos `clasificacion` y `status`. A cada grupo se le aplica la función `count`.

Clasificación	Status	Group by
A	AGOTADO	Grupo 1
A	COMPRADO	Grupo 2
A	COMPRADO	
A	VENDIDO	Grupo 3
B	COMPRADO	Grupo 4
B	COMPRADO	
B	DEFECTUOSO	Grupo 5
C	AGOTADO	Grupo 6
C	DEFECTUOSO	Grupo 7

```
select clasificacion, status, count(*)
from inventario_libro
group by clasificacion,status;
```

```
CLASIFICACION STATUS    COUNT(*)
============ ========== ========
C            AGOTADO    1
C            DEFECTUOSO 1
B            DEFECTUOSO 1
B            COMPRADO   2
A            AGOTADO    1
A            VENDIDO    1
A            COMPRADO   2
```

Ejemplo:

Para cada clasificación y status determinar el precio máximo que un libro ha tenido.
Nuevamente, en este caso, se requiere formar grupos por `clasificacion` Y `status`, a cada grupo se le aplicará la función `max`, pero al campo `precio`:

Clasificacion	Status	Group by
A	AGOTADO	Grupo 1
A	COMPRADO	Grupo 2
A	COMPRADO	
A	VENDIDO	Grupo 3
B	COMPRADO	Grupo 4
B	COMPRADO	
B	DEFECTUOSO	Grupo 5
C	AGOTADO	Grupo 6
C	DEFECTUOSO	Grupo 7

```
select clasificacion,status,max(precio)
from inventario_libro
group by clasificacion,status;
```

```
CLASIFICACION STATUS     MAX(PRECIO)
============ ========== ===========
B            COMPRADO   442.45
C            DEFECTUOSO 145.98
C            AGOTADO    201.45
A            COMPRADO   89.45
A            VENDIDO    189.45
B            DEFECTUOSO 345.98
A            AGOTADO    133.45
```

- Comprobar el resultado, por ejemplo, para el grupo 2, los precios son:

89.45
45.98

  Por lo tanto, el valor máximo es 89.45

- Observar en el SQL, para comprobar que la sentencia esté bien formada, la cláusula `group by` debe contener TODA la lista de campos que aparece después de la cláusula `select`, a los que no se le aplica la función de agregación, en este caso `clasificacion` y `status`. Si no se cumple esta regla, el manejador generará un error.

El requisito anterior se debe a que las funciones de agregación se aplican sobre un conjunto de datos y es necesario indicar el en SQL, la forma en la que se deberán armar dichos grupos, empleando para ello la cláusula `group by`. En el ejemplo anterior, `max(precio)` se deberá a aplicar a los grupos de datos que se formen al emplear las columnas `clasificacion` y `status`.

9.5.5.2. Having

`having` es similar a `where`, con la siguiente distinción:

- `where` opera sobre columnas y expresiones de los registros de una tabla
- `having` opera sobre el resultado que se produce al emplear `group by`.

Ejemplo:

Generar una sentencia SQL que muestre el número de libros comprados existentes clasificados con base a su status.

En este caso, la función `count` se aplicará a los grupos formados por los valores de los campos `clasificacion` y `status`. Del resultado obtenido solo nos interesan los registros con estatus COMPRADO:

```
select clasificacion, status, count(*)
from inventario_libro
group by clasificacion,status
having status='COMPRADO';

CLASIFICACION ESTATUS  COUNT(*)
============= ======== ========
B             COMPRADO 2
A             COMPRADO 2
```

Ejemplo:

Generar una sentencia SQL que muestre el número de libros existentes que han sido vendidos o que están agotados, y que el número de existencias sea mayor a 1.
En este caso, solo nos interesan los libros comprados y que la cantidad (el resultado de la función `count`) sea mayor a 1.

```
select clasificacion, estatus, count(*) as existencias
from inventario_libro
where estatus='comprado'
or estatus='agotado'
group by clasificacion, estatus having count(*)>1;

CLASIFICACION ESTATUS  EXISTENCIAS
============= ======== ===========
B             COMPRADO 2
A             COMPRADO 2
```

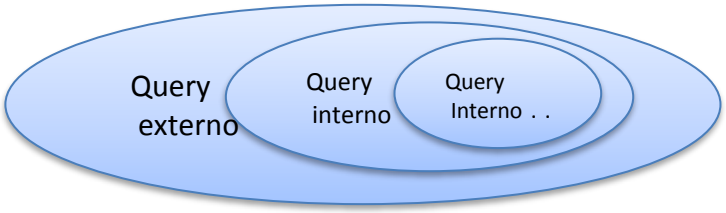
Otra solución:

```
select clasificacion, estatus, count(*) as existencias
from inventario_libro
group by clasificacion, estatus
having count(*)>1
and (estatus='comprado' or estatus='agotado');
```

9.6. SUBCONSULTAS.

Las subconsultas llamados también consultas anidadass, son sentencias SQL que se encuentran inmersas o incluidas dentro de otra sentencia. Típicamente las subconsultas se emplean cuando se tiene una condición pero no se conoce su valor para ser incluida por ejemplo de forma directa en la cláusula `where`.

Es posible tener varios niveles de anidamiento.



Ejemplo:

"Obtener los datos del profesor con el mayor número de cursos impartidos".

Para poder determinar los datos del profesor, primero necesitamos conocer al profesor que ha impartido el mayor número de cursos. Esta condición no la conocemos de forma directa, por lo que primero se deberá realizar una consulta que calcule el número de cursos impartidos por profesor y seleccione al mayor. Después en otra consulta se obtendrán los datos del profesor que cumple con la condición. En SQL es posible lanzar las 2 consultas en una sola sentencia empleando *subconsultas*.

9.6.1. Características de una subconsulta:

- Es una sentencia `select` delimitada por paréntesis (<subconsulta>).
- A la primera consulta se le llama consulta exterior.
- A la consulta que se encuentra dentro de otra se le llama consulta interior o subconsulta.
- La consulta interior se ejecuta primero.
- La salida de la primera consulta (interna) es empleada como entrada de la consulta exterior.

Las subconsultas pueden regresar los siguientes datos:

- Un valor simple (1 registro, 1 columna)
- Una lista de valores (N registros, 1 columna)
- Una tabla virtual (N registros, N columnas)
- `null` o vacío (sin resultados).

Los subqueries puede aparecer en distintas partes de la sentencia `select`:

Lugares donde puede aparecer una subconsulta.	Ejemplos

En la cláusula `select`	`select atributo1,`
	`  atributo2,(subconsulta) as atributo3`
	`from ..`
	`where ..`

En la cláusula `from` con sintaxis anterior.	`select *` `from tabla1 t1, tabla 2 t2, (subconsulta) t3`

En la cláusula `join`, sintaxis estándar	`select *`
	`from tabla1 t1`
	`join (subconsulta) t2`
	`on ..`
	`where ..`

En la cláusula `where`	`select *`
	`from tabla1 t1`
	`where atributo1 = (subconsulta)`
	`and ..`
	
	`select *`
	`from tabla1 t1`
	`where (atributo1,atributo2) = (subconsulta)`
	`and ..`
	
	`select *`
	`from tabla1 t1`
	`where atributo1 in (subconsulta)`
	`and ..`
	
	`select *`
	`from tabla1 t1`
	`where exists (subconsulta)`
	`and ..`

En la cláusula having	`select c1, count(*)`
	`from tabla 1 t1`
	`where ..`
	`group by c1`
	`having count(*) = (subconsulta)`

9.6.2. Subconsultas en la cláusula select:

Generalmente después de la sentencia `select`, se especifica la lista de columnas a desplegar. En lugar de escribir el nombre de la columna, se puede emplear una subconsulta que calcule el valor de dicha columna. Es decir, se emplea una subconsulta para obtener el valor de una columna que no se puede obtener de forma directa.

Se requiere que la subconsulta regrese un único valor, el cual se ***repetirá*** para cada registro obtenido por la consulta exterior.

Ejemplo:

Seleccionar el nombre, precio, y precio promedio de todos los libros de clasificación A.

Observar que en este caso, la última columna corresponde al valor promedio de un campo, por lo que se puede escribir una subconsulta que calcule su valor (independiente a la consulta exterior).

```
select nombre, precio, clasificacion, (
   select avg(precio)
   from inventario_libro
   where clasificacion='A'
) as promedio
from  inventario_libro
where clasificacion ='A';
```

```
NOMBRE                PRECIO CLASIFICACION PROMEDIO
===================== ====== ============= ========
FABULAS DIVERSAS      89.45  A             114.5825
PRINCIPITO            45.98  A             114.5825
CRONICAS PARA INFANTES 189.45 A             114.5825
CURSO DE COSTURA      133.45 A             114.5825
```

Ejemplo :

Mostrar los datos de los libros clasificación B, así como el precio máximo para ese tipo de libro.

```
select nombre,clasificacion,precio, (
   select max(precio)
   from inventario_libro
   where clasificacion='B'
) as maxprecio
from inventario_libro
where clasificacion='B';
```

```
NOMBRE                CLASIFICACION PRECIO MAXPRECIO
===================== ============= ====== =========
FISICA I              B             289.45 442.45
PRINCIPIOS DE QUIMICA B             345.98 442.45
BIOLOGIA I            B             442.45 442.45
```

9.6.3. Subconsultas en la cláusula from

La cláusula `from` indica la lista de tablas a partir de la cual se obtendrán los resultados. Esta lista puede ser reemplazada por una subconsulta.

- Normalmente, a las tablas que aparecen en la cláusula `from` se les asocia un alias. De forma similar, a una subconsulta se le puede asociar un alias, y las columnas que incluya pueden ser referenciadas por la consulta exterior empleando el alias de la subconsulta.
- En este caso la subconsulta regresa una tabla virtual, que justamente sustituye a la lista de tablas de donde se obtendrá la información.
- Este tipo de subconsultas normalmente se emplean cuando tenemos una consulta complicada y se decide fraccionarla: una primera consulta que seleccione parte del resultado (consulta interior), y la otra (consulta exterior) realiza una selección sobre el resultado de la consulta interior.

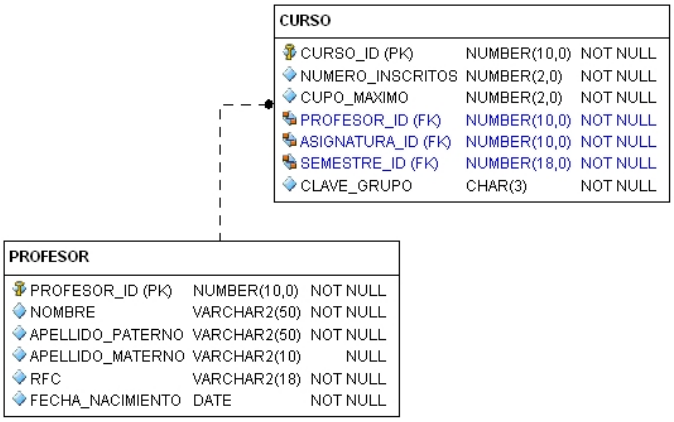
Ejemplo:

Suponer 2 tablas, `curso` y `profesor` del modelo de control escolar visto anteriormente. Se requiere mostrar los datos (todos) de los profesores que imparten 3 o más cursos:

Si observamos la tabla `CURSO`, es posible determinar mediante una primera consulta, los identificadores de todos los profesores que imparten más de 3 cursos haciendo un conteo de los cursos agrupados por profesor.

```
select profesor_id, count(*) as cuenta
from curso
c group by profesor_id;
```

```
PROFESOR_ID CUENTA
=========== ======
22          1
11          2
13          3
14          3
20          3
21          1
17          1
23          2
18          2
10          3
12          4
15          2
16          2
19          3
```



Del resultado anterior, podemos observar que los registros resaltados, corresponden a profesores que aparecen en más de 3 veces en la tabla `curso`. Esto lo podemos obtener con la siguiente consulta.

```
select profesor_id, count(*) as cuenta
from curso c
group by profesor_id
```

De este primer resultado (tabla virtual al que llamaremos **q1**) solo nos restaría excluir los que son menores a 3, y para los restantes, obtener los datos de los profesores empleando una consulta externa, por lo que esta primera consulta actuará como una subconsulta dentro de la cláusula `from`.

La consulta final quedaría así:

```
select p.*
from profesor p,(
   select profesor_id, count(*) as cuenta
   from curso c
   group by profesor_id ) q1
where  q1.profesor_id=p.profesor_id
and q1.cuenta >= 3
```

Observar que estamos excluyendo a los profesores con menos de 3 cursos, empleando la columna "cuenta" del subquery : `q1.cuenta >=3`, es decir, tratamos al resultado de la subconsulta como si fuera cualquier otra tabla identificada por el alias `q1`. Algunos autores a estas subconsultas se les nombra como "inline views".

Finalmente, debemos hacer un join entre la tabla `profesor`, con la tabla virtual obtenida por el subquery, para determinar los datos de los profesores deseados: **`q1`**`.profesor_id=p.profesor_id`

```
PROFESOR_ID NOMBRE    APELLIDO_PATERNO APELLIDO_MATERNO RFC            FECHA_NACIMIENTO
=========== ========= ================ ================ ============= ===================
13          MARGARITA LUJAN            HURTADO          LUHM631024RY3 1963-10-24 00:00:00
14          HUGO      FLORES           LINARES          FOLH510410IH1 1951-04-10 00:00:00
20          ELSA      PEDROZA          SOLANO           PELE670210HI9 1967-02-10 00:00:00
10          OMAR      KRAUCE           LOPEZ            OAKL701010PE0 1970-10-10 00:00:00
12          JULIAN    VALDEZ           SANCHEZ          VASJ5711082A4 1957-11-08 00:00:00
19          JAVIER    BARRERA          MUÑOZ            BAMJ720629FP4 1972-06-29 00:00:00
```

Otra solución es emplear la cláusula HAVING, para que desde la subconsulta se excluyan los registros con menos de 3 cursos. En la consulta exterior se elimina la condición `q1.cuenta >=3`

```
select p.*
from profesor p,(
   select profesor_id, count(*) as cuenta
   from curso c group by profesor_id
   having count(*) >=3 ) q1
where q1.profesor_id=p.profesor_id;
```

Oracle soporta un número ilimitado de subconsultas en la clausula `FROM`.

9.6.4. Subconsultas en la cláusula where

Se emplean normalmente para indicar valores de condiciones que no son conocidos y que requieren una consulta a la base de datos. El resultado se compara con algún operador =, >, <, etc., por lo que la consulta debe regresar un valor simple.
Es posible emplear subconsultas haciendo uso de operadores lógicos como `in, not in, exists, all`, etc. En estos casos, la subconsulta debe regresar un valor simple o una lista de valores (1 columna, N registros).

Ejemplo:
Mostrar el número de cursos que imparte el profesor con la mayor edad.

Como se puede observar, la condición es que el profesor sea el mayor de edad. De forma directa no sabemos quién es el profesor con la mayor edad. Por lo tanto, la consulta se puede dividir en 2 partes:

- Q1: Consulta que determina la fecha de nacimiento mínima asociada a un profesor.
- Q2: Obtener el número de cursos que imparte el profesor que haya nacido en la fecha obtenida por q1

```
select count(*)
from curso c,  profesor p
where c.profesor_id = p.profesor_id
and fecha_nacimiento = (
   select min(fecha_nacimiento)
   from profesor
);
```

Es importante que la subconsulta regrese un solo valor, ya que se está realizando una comparación con el campo `fecha_nacimiento` empleando el operador "=".

9.6.5. Subconsultas correlacionales.

Ocurre cuando la consulta interna depende del resultado de la consulta externa. Por esta característica, en una consulta correlacional por cada registro seleccionado en la consulta externa, se ejecuta la consulta interior.

Ejemplo:

Para los cursos impartidos en el semestre 2008-1 (semestre_id=1) determinar por asignatura el curso que ha tenido el mayor número de alumnos inscritos. Mostrar los datos del curso (id, grupo, id de asignatura, número de alumnos inscritos).

CURSO

CURSO_ID	NUMBER(10,0)	NOT NULL
CUPO_MAXIMO	NUMBER(2,0)	NOT NULL
NUM_INSCRITOS	NUMBER(2,0)	NOT NULL
PROFESOR_ID (FK)	NUMBER(10,0)	NOT NULL
ASIGNATURA_ID (FK)	NUMBER(10,0)	NOT NULL
SEMESTRE_ID (FK)	NUMBER(18,0)	NOT NULL
CLAVE_GRUPO	CHAR(3)	NOT NULL

- La consulta externa se encargará de recorrer la lista de cursos impartidos en el semestre 2008-1
- Por cada curso encontrado, se ejecutará la subconsulta para determinar el número máximo de alumnos inscrito por asignatura. Observar que a la subconsulta le pasamos el valor del campo `asignatura_id`:

```
select c.curso_id, c.clave_grupo,c.numero_inscritos, c.asignatura_id
from curso c
where c.semestre_id = 1
and c.numero_inscritos = (
    select max(c2.numero_inscritos)
    from curso c2
    where c2.asignatura_id=c.asignatura_id
);
```

El procesamiento de la consulta se describe a continuación:

Observar que en la subconsulta se hace una igualación: `c2.asignatura_id=c.asignatura_id`. Es decir, el campo "**`c.asignatura_id`**" es un valor generado por la consulta exterior. Para cada una de las asignaturas que encuentre la consulta exterior, se sustituye en la subconsulta y se determina el valor máximo de los alumnos inscritos. Por ejemplo, considerando un extracto de los datos de la tabla CURSO, El primer registro a procesar es el primer registro de la siguiente tabla:

NUMERO_INSCRITOS	CUPO_MAXIMO	PROFESOR_ID	CURSO_ID	SEMESTRE_ID	CLAVE_GRUPO	ASIGNATURA_ID
35	**50**	**10**	**1**	**1**	**1**	**1**
30	**50**	**11**	**2**	**1**	**2**	**1**
44	50	10	3	1	1	2
45	50	12	4	1	2	2
46	**50**	**13**	**5**	**1**	**3**	**2**
44	50	12	6	1	1	3
44	50	13	7	1	2	3
44	50	14	8	1	3	3
39	50	12	9	1	4	3
27	40	14	10	1	1	4

Por lo tanto, el valor de "**`c.asignatura_id`**" se sustituye por 1:

```
select max(c2.numero_inscritos)
from curso c2 where c2.asignatura_id=1
```

Al ejecutar la subconsulta nos da como resultado: **35**, como se puede observar en la tabla. Es decir, de los 2 primeros registros que tienen `asignatura_id=1`, se selecciona al mayor valor del campo `cupo_maximo`, que en este caso es 35.
Este valor obtenido por la subconsulta se sustituye en la consulta externa de la siguiente forma:

```
select c.curso_id, c.clave_grupo,c.numero_inscritos, c.asignatura_id
from curso c
where c.semestre_id = 1
and c.numero_inscritos = (35);
```

Se evalúa la consulta exterior, por lo que el primer registro de la consulta final, es el curso con id = **1**.

Continuando con el recorrido de la tabla, ahora se sustituye el valor de "**C.ASIGNATURA_ID**"con **2**.

```
select max(c2.numero_inscritos)
from curso c2
where c2.asignatura_id=2;
```

Se ejecuta el subquery , nos da como resultado **46** (como se puede observar en la tabla). Finalmente, se sustituye el valor en el query exterior:

```
select c.curso_id, c.clave_grupo,c.numero_inscritos, c.asignatura_id
from curso c
where c.semestre_id = 1
and c.numero_inscritos = (46);
```

Al evualuar la consulta exterior, el siguiente registro de la consulta final es el curso con id = **5**

Este procedimiento se aplica repetidamente hasta terminar con todos los registros de la tabla dando como resultado:

CURSO_ID	CLAVE_GRUPO	NUMERO_INSCRITOS	ASIGNATURA_ID
1	001	35	1
5	003	46	2
6	001	44	3
7	002	44	3
8	003	44	3
10	001	27	4
14	004	32	5
17	002	24	6
23	002	24	11
26	002	24	13
29	005	24	13
30	001	5	17
31	001	29	18
32	001	28	19

9.6.5.1. Solución sin emplear una subconsulta correlacional.

En este ejemplo tenemos varios conjuntos que determinar:

1. Todos los cursos impartidos en el semestre 2008-1

2. De estos cursos, necesitamos agruparlos por asignatura y determinar el mayor número de inscritos de cada grupo de registros empleando la siguiente consulta:

```
select  c.asignatura_id, max(c.numero_inscritos)
from curso c
where c.semestre_id=1
group by c.asignatura_id;
```

```
ASIGNATURA_ID MAX(C.NUMERO_INSCRITOS)
============= ======================
1             35
2             46
3             44
4             27
5             32
6             24
11            24
13            24
17            5
18            29
19            28
```

Posteriormente, agregaríamos otra consulta para que determine los datos de los cursos que cumplan con las 2 condiciones de los datos mostrados en la tabla anterior. Podemos emplear una subconsulta en la cláusula `from`:

```
select c.curso_id, c.clave_grupo,c.numero_inscritos, c.asignatura_id
from curso c,(
   select  asignatura_id, max(numero_inscritos) as max_inscritos
   from curso
   where semestre_id=1
   group by asignatura_id) q1
where c.asignatura_id=q1.asignatura_id
and c.numero_inscritos=q1.max_inscritos;
```

El resultado es el siguiente:

```
CURSO_ID CLAVE_GRUPO NUMERO_INSCRITOS ASIGNATURA_ID
======== =========== ================ =============
1        001         35               1
5        003         46               2
6        001         44               3
7        002         44               3
8        003         44               3
10       001         27               4
14       004         32               5
17       002         24               6
23       002         24               11
26       002         24               13
29       005         24               13
30       001         5                17
31       001         29               18
32       001         28               19
```

Observar, que en algunos casos la asignatura se repite debido a que existen 2 o más grupos con el número máximo de inscritos determinados por la subconsulta
En la práctica las subconsultas correlacionales se tratan de evitar ya que consumen más recursos que una subconsulta de otro tipo, sin embargo, no en todos los casos es posible.