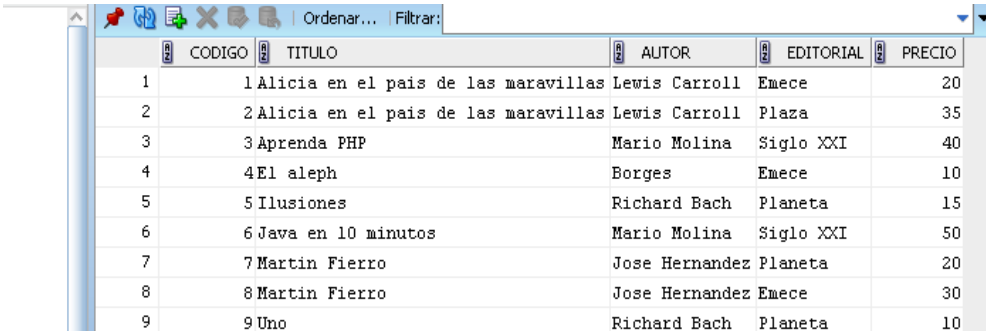
Ejemplos con Subconsultas

SUBCONSULTAS COMO EXPRESIÓN

Trabajamos con la tabla "libros1" de una librería. La tabla tiene los siguientes registros:



	CODIGO	TITULO	AUTOR	EDITORIAL	PRECIO
1	1	Alicia en el pais de las maravillas	Lewis Carroll	Emece	20
2	2	Alicia en el pais de las maravillas	Lewis Carroll	Plaza	35
3	3	Aprenda PHP	Mario Molina	Siglo XXI	40
4	4	El aleph	Borges	Emece	10
5	5	Ilusiones	Richard Bach	Planeta	15
6	6	Java en 10 minutos	Mario Molina	Siglo XXI	50
7	7	Martin Fierro	Jose Hernandez	Planeta	20
8	8	Martin Fierro	Jose Hernandez	Emece	30
9	9	Uno	Richard Bach	Planeta	10

Obtenemos el título, precio de un libro específico (Uno) y la diferencia entre su precio y el máximo valor:

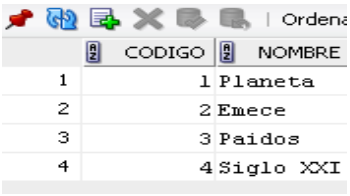
```
select titulo,precio,
 precio-(select max(precio) from libros1) as diferencia
 from libros1
 where titulo='Uno';
```

Mostramos el título y precio del libro más costoso:

```
select titulo,autor, precio
 from libros1
 where precio=
  (select max(precio) from libros1);
```

SUBCONSULTAS CON IN

Trabajamos con las tablas "libros2" y "editoriales2" de una librería. Estas tablas tienen los siguientes registros:



	CODIGO	NOMBRE
1	1	Planeta
2	2	Emece
3	3	Paidos
4	4	Siglo XXI

	CODIGO	TITULO	AUTOR	CODIGOEDITORIAL	PRECIO
1	100	Uno	Richard Bach	1	15
2	101	Ilusiones	Richard Bach	1	18
3	102	Aprenda PHP	Mario Molina	4	20
4	103	El aleph	Borges	2	40
5	104	Puente al infinito	Richard Bach	2	10
6	105	Antología	Borges	1	20
7	106	Cervantes y el quijote	Borges	3	25

Queremos conocer el nombre de las editoriales que han publicado libros del autor "Richard Bach":

```
select nombre
 from editoriales2
 where codigo in
  (select codigoeditorial
    from libros2
    where autor='Richard Bach');
```

Probamos la subconsulta separada de la consulta exterior para verificar que devuelve una lista de valores de un solo campo:

```
select codigoeditorial
 from libros2
 where autor='Richard Bach';
```

Podemos reemplazar por un "join" la primera consulta:

```
select distinct nombre
 from editoriales2 e
 inner join libros2
 on codigoeditorial=e.codigo
 where autor='Richard Bach';
```

También podemos buscar las editoriales que no han publicado libros de "Richard Bach":

```
select nombre
 from editoriales2
 where codigo not in
  (select codigoeditorial
    from libros2
    where autor='Richard Bach');
```

Subconsultas any, some y all

Trabajamos con las tablas "libros2" y "editoriales2" de una librería.

Mostramos los títulos de los libros de "Borges" de editoriales que han publicado también libros de "Richard Bach":

```
select titulo
 from libros2
 where autor like '%Borges%' and
 codigoeditorial = any
  (select e.codigo
   from editoriales2 e
   inner join libros2 l
   on codigoeditorial=e.codigo
   where l.autor like '%Bach%');
```

Realizamos la misma consulta pero empleando "all" en lugar de "any". Lo que queremos saber es si TODAS las editoriales que publicaron libros de 'Borges' coinciden con TODAS las editoriales que publicaron libros de 'Bach'

```
select titulo
 from libros2
 where autor like '%Borges%' and
 codigoeditorial = all
  (select e.codigo
   from editoriales2 e
   join libros2 l
   on codigoeditorial=e.codigo
   where l.autor like '%Bach%');
```

La consulta interna (subconsulta) devuelve una lista de valores de un solo campo (puedes ejecutar la subconsulta como una consulta para probarla), luego, la consulta externa compara cada valor de "codigoeditorial" con cada valor de la lista, si TODOS coinciden, devuelve los títulos. En este caso no coincide con todos, por lo que no hay resultados.

Mostramos los títulos y precios de los libros de "Borges" cuyo precio supera a ALGUN precio de los libros de "Richard Bach":

```
select titulo,precio
 from libros2
 where autor like '%Borges%' and
 precio > any
  (select precio
   from libros2
   where autor like '%Bach%');
```

Veamos la diferencia si empleamos "all" en lugar de "any":

```
select titulo,precio
 from libros2
 where autor like '%Borges%' and
 precio > all
  (select precio
   from libros2
   where autor like '%Bach%');
```

El precio de cada libro de "Borges" es comparado con cada valor de la lista de valores devuelta por la subconsulta; si cumple la condición, es decir, si es mayor a TODOS los precios de "Richard Bach" (o al mayor), se lista.

Emplear "= any" es lo mismo que emplear "in".

Emplear "<> all" es lo mismo que emplear "not in".

SUBCONSULTAS CORRELACIONADAS

Un almacén almacena la información de sus ventas en una tabla llamada "facturas" en la cual guarda el número de factura, la fecha y el nombre del cliente y una tabla denominada "detalles" en la cual se almacenan los distintos items correspondientes a cada factura: numero de factura, numero de item, el nombre del artículo, el precio (unitario) y la cantidad.

Detalles

Ordenar... Filtrar:

	NUMEROFACTURA	NUMEROITEM	ARTICULO	PRECIO	CANTIDAD
1	1200	1	lapiz	1	100
2	1200	2	goma	0,5	150
3	1201	1	regla	1,5	80
4	1201	2	goma	0,5	200
5	1201	3	cuaderno	4	90
6	1202	1	lapiz	1	200
7	1202	2	escuadra	2	100
8	1300	1	lapiz	1	300

Facturas

Ordenar... Filtrar:

	NUMERO	FECHA	CLIENTE
1	1200	15/01/07	Juan Lopez
2	1201	15/01/07	Luis Torres
3	1202	15/01/07	Ana Garcia
4	1300	20/01/07	Juan Lopez

Se necesita una lista de todas las facturas que incluya el número, la fecha, el cliente, la cantidad de artículos comprados y el total:

```
select f.*,
  (select count(d.numeroitem)
    from Detalles d
    where f.numero=d.numerofactura) as cantidad,
  (select sum(d.precio*cantidad)
    from Detalles d
    where f.numero=d.numerofactura) as total
from facturas f;
```

El segundo "select" retorna una lista de valores de una sola columna con la cantidad de items por factura (el número de factura lo toma del "select" exterior); el tercer "select" retorna una lista de valores de una sola columna con el total por factura (el número de factura lo toma del "select" exterior); el primer "select" (externo) devuelve todos los datos de cada factura.

A este tipo de subconsulta se la denomina consulta correlacionada. La consulta interna se evalúa tantas veces como registros tiene la consulta externa, se realiza la subconsulta para cada registro de la consulta externa. El campo de la tabla dentro de la subconsulta (f.numero) se compara con el campo de la tabla externa.

En este caso, específicamente, la consulta externa pasa un valor de "numero" a la consulta interna. La consulta interna toma ese valor y determina si existe en "detalles", si existe, la consulta interna devuelve la suma. El proceso se repite para el registro de la consulta externa, la consulta externa pasa otro "numero" a la consulta interna y SQL repite la evaluación.

EXISTS Y NOT EXISTS

Los operadores "exists" y "not exists" se emplean para determinar si hay o no datos en una lista de valores.

Estos operadores pueden emplearse con subconsultas correlacionadas para restringir el resultado de una consulta exterior a los registros que cumplen la subconsulta (consulta interior). Estos operadores devuelven "true" (si las subconsultas devuelven registros) o "false" (si las subconsultas no devuelven registros).

Cuando se coloca en una subconsulta el operador "exists", SQL analiza si hay datos que coinciden con la subconsulta, no se devuelve ningún registro, es como un test de existencia; SQL termina la recuperación de registros cuando por lo menos un registro cumple la condición "where" de la subconsulta.

La sintaxis básica es la siguiente:

```
... where exists (SUBCONSULTA);
```

En este ejemplo se usa una subconsulta correlacionada con un operador "exists" en la cláusula "where" para devolver una lista de clientes que compraron el artículo "lapiz":

```
select cliente,numero
 from facturas f
 where exists
  (select *from Detalles d
    where f.numero=d.numerofactura
    and d.articulo='lapiz');
```

Puede obtener el mismo resultado empleando una combinación:

```
select cliente,numero
  from facturas f inner join Detalles d
  on f.numero=d.numerofactura
  and d.articulo='lapiz';
```

Podemos buscar los clientes que no han adquirido el artículo "lapiz" empleando "not exists":

```
 select cliente,numero
  from facturas f
  where not exists
   (select *from Detalles d
     where f.numero=d.numerofactura
     and d.articulo='lapiz');
```

Veamos **otro ejemplo**

Un club imparte clases de distintos deportes a sus socios. El club tiene una tabla llamada "inscritos" en la cual almacena el número de "socio", el código del deporte en el cual se inscribe y la cantidad de cuotas pagas (desde 0 hasta 10 que es el total por todo el año), y una tabla denominada "socios" en la que guarda los datos personales de cada socio.

Creemos una subconsulta con el operador "exists" para devolver la lista de socios que se inscribieron en el deporte 'natacion'

```
select nombre
  from socios s
  where exists
   (select * from inscritos i
     where s.numero=i.numerosocio
     and i.deporte='natacion');
```

Busca los socios que NO se han inscrito en el deporte 'natacion' empleando "not exists".

```
select nombre
  from socios s
  where not exists
   (select *from inscritos i
     where s.numero=i.numerosocio
     and i.deporte='natacion');
```

Muestra todos los datos de los socios que han pagado todas las cuotas.

```
select s.*
  from socios s
  where exists
   (select *from inscritos i
     where s.numero=i.numerosocio
     and i.cuotas=10);
```

Obtener el mismo resultado de la consulta anterior pero esta vez empleando una combinación.

```
select s.* from socios s
  inner join inscritos i
  on s.numero=i.numerosocio
  where i.cuotas=10;
```

Ejemplos de la página Web http://www.oracleya.com.ar/ (que ya no existe)