

MySQL

08_ Subconsultas



SQL_ Subconsultas

Contenido

1. _Subconsultas	4
2. _Subconsultas como expresión	5
3. _Subconsultas IN	8
4. _Subconsulta ANY SOME ALL	11
5. _Subconsultas Correlacionadas	15
6. _Subconsultas EXISTS-NO EXISTS	18
7. _Subconsultas Autocombinación	21
8. _Subconsultas UPDATE DELETE	24
9. _Subconsultas INSERT	27
10. _Subconsultas Tabla derivada	30

SQL_ Subconsultas

Introducción

En MySQL, una subconsulta es una consulta dentro de otra consulta. Es decir, una subconsulta es una consulta que se utiliza dentro de otra consulta para proporcionar datos adicionales que se utilizan en la consulta principal.

Las subconsultas en MySQL se pueden utilizar en diferentes situaciones, como para filtrar los resultados de una consulta, comparar los valores de una columna con los valores de otra tabla o realizar operaciones de agregación en subconjuntos de datos.

Las subconsultas se pueden utilizar en diferentes cláusulas de una consulta, como la cláusula WHERE, la cláusula FROM o la cláusula HAVING. Por ejemplo, se puede utilizar una subconsulta en la cláusula WHERE para filtrar los resultados de una consulta principal basada en los resultados de otra consulta.

Las subconsultas también se pueden utilizar en combinación con otras operaciones SQL, como UNION, JOIN o EXISTS. Por ejemplo, se puede utilizar una subconsulta en una cláusula EXISTS para verificar si existe un registro en una tabla antes de realizar una acción en otra tabla.

Las subconsultas pueden ser anidadas, lo que significa que una subconsulta puede contener otra subconsulta. Sin embargo, se debe tener cuidado al utilizar subconsultas anidadas, ya que pueden aumentar la complejidad y el tiempo de ejecución de una consulta.

En resumen, una subconsulta en MySQL es una consulta dentro de otra consulta que se utiliza para proporcionar datos adicionales que se utilizan en la consulta principal. Las subconsultas se pueden utilizar en diferentes situaciones y se pueden utilizar en diferentes cláusulas de una consulta y en combinación con otras operaciones SQL.



SQL_ Subconsultas

1. Subconsultas

Una subconsulta (subquery) es una sentencia "select" anidada en otra sentencia "select", "insert", "update" o "delete" (o en otra subconsulta).

Las subconsultas se emplean cuando una consulta es muy compleja, entonces se la divide en varios pasos lógicos y se obtiene el resultado con una única instrucción y cuando la consulta depende de los resultados de otra consulta.

Generalmente, una subconsulta se puede reemplazar por combinaciones.

Las subconsultas se DEBEN incluir entre paréntesis.

Puede haber subconsultas dentro de subconsultas.

Se pueden emplear subconsultas:

- en lugar de una expresión, siempre que devuelvan un solo valor o una lista de valores.
- que retornen un conjunto de registros de varios campos en lugar de una tabla o para obtener el mismo resultado que una combinación (join).

Hay tres tipos básicos de subconsultas:

1. las que retornan un solo valor escalar que se utiliza con un operador de comparación o en lugar de una expresión.
2. las que retornan una lista de valores, se combinan con "in", o los operadores "any", "some" y "all".
3. los que testean la existencia con "exists".

Reglas a tener en cuenta al emplear subconsultas:

- la lista de selección de una subconsulta que va luego de un operador de comparación puede incluir sólo una expresión o campo (excepto si se emplea "exists" y "in").
- si el "where" de la consulta exterior incluye un campo, este debe ser compatible con el campo en la lista de selección de la subconsulta.
- las subconsultas luego de un operador de comparación (que no es seguido por "any" o "all") no pueden incluir cláusulas "group by" ni "having".
- "distinct" no puede usarse con subconsultas que incluyan "group by".
- una subconsulta puede estar anidada dentro del "where" o "having" de una consulta externa o dentro de otra subconsulta.
- si una tabla se nombra solamente en un subconsulta y no en la consulta externa, los campos no serán incluidos en la salida (en la lista de selección de la consulta externa).

SQL_ Subconsultas

2. Subconsultas como expresión

Una subconsulta puede reemplazar una expresión. Dicha subconsulta debe devolver un valor escalar (o una lista de valores de un campo).

Las subconsultas que retornan un solo valor escalar se utiliza con un operador de comparación o en lugar de una expresión:

```
select CAMPOS
from TABLA
where CAMPO OPERADOR (SUBCONSULTA);

select CAMPO OPERADOR (SUBCONSULTA)
from TABLA;
```

Si queremos saber el precio de un determinado libro y la diferencia con el precio del libro más costoso, anteriormente debíamos averiguar en una consulta el precio del libro más costoso y luego, en otra consulta, calcular la diferencia con el valor del libro que solicitamos. Podemos conseguirlo en una sola sentencia combinando dos consultas:

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

En el ejemplo anterior se muestra el título, el precio de un libro y la diferencia entre el precio del libro y el máximo valor de precio.

Queremos saber el título, autor y precio del libro más costoso:

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

Note que el campo del "where" de la consulta exterior es compatible con el valor retornado por la expresión de la subconsulta.

SQL_ Subconsultas

Servidor de MySQL instalado en forma local.

Ingrese al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL para probar subconsultas como expresión:

```
drop table if exists libros;

create table libros(
  codigo int auto_increment,
  titulo varchar(40),
  autor varchar(30),
  editorial varchar(20),
  precio decimal(5,2),
  primary key(codigo)
);

insert into libros(titulo,autor,editorial,precio)
values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',20.00);
insert into libros(titulo,autor,editorial,precio)
values('Alicia en el pais de las maravillas','Lewis Carroll','Plaza',35.00);
insert into libros(titulo,autor,editorial,precio)
values('Aprenda PHP','Mario Molina','Siglo XXI',40.00);
insert into libros(titulo,autor,editorial,precio)
values('El aleph','Borges','Emece',10.00);
insert into libros(titulo,autor,editorial,precio)
values('Ilusiones','Richard Bach','Planeta',15.00);
insert into libros(titulo,autor,editorial,precio)
values('Java en 10 minutos','Mario Molina','Siglo XXI',50.00);
insert into libros(titulo,autor,editorial,precio)
values('Martin Fierro','Jose Hernandez','Planeta',20.00);
insert into libros(titulo,autor,editorial,precio)
values('Martin Fierro','Jose Hernandez','Emece',30.00);
insert into libros(titulo,autor,editorial,precio)
values('Uno','Richard Bach','Planeta',10.00);

-- Obtenemos el título, precio todos los libros y la diferencia entre
-- su precio y el máximo valor:
select titulo,precio,
((select max(precio) from libros) - precio) as diferencia
from libros;

-- Obtenemos el título, precio de un libro específico y la diferencia entre
-- su precio y el máximo valor:
select titulo,precio,
((select max(precio) from libros) - precio) as diferencia
from libros
where titulo='Uno';
```

SQL_ Subconsultas

-- Mostramos el título y precio del libro más costoso:

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

Genera una salida similar a esta:

The screenshot shows a SQL query editor window titled 'Query 1'. The query is as follows:

```
28 insert into libros(titulo,autor,editorial,precio)
29 values('Uno','Richard Bach','Planeta',10.00);
30
31 -- Obtenemos el título, precio de un libro específico y la diferencia entre
32 -- su precio y el máximo valor:
33 select titulo,precio,
34 precio-(select max(precio) from libros) as diferencia
35 from libros
36 where titulo='Uno';
37
38 -- Mostramos el título y precio del libro más costoso:
39 select titulo,autor, precio
40 from libros
41 where precio=
42 (select max(precio) from libros);
43
```

Below the query editor, the 'Result Grid' is displayed, showing the results of the query. The grid has three columns: 'titulo', 'autor', and 'precio'. The first row shows the result for the book 'Java en 10 minutos' by Mario Molina, with a price of 50.00.

titulo	autor	precio
Java en 10 minutos	Mario Molina	50.00

SQL_ Subconsultas

3. Subconsultas con in

Vimos que una subconsulta puede reemplazar una expresión. Dicha subconsulta debe devolver un valor escalar o una lista de valores de un campo; las subconsultas que retornan una lista de valores reemplazan a una expresión en una cláusula "where" que contiene la palabra clave "in".

El resultado de una subconsulta con "in" (o "not in") es una lista. Luego que la subconsulta retorna el resultados, la consulta exterior los usa.

La sintaxis básica es la siguiente:

```
...where EXPRESION in (SUBCONSULTA);
```

Este ejemplo muestra los nombres de las editoriales que han publicado libros de un determinado autor:

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

La subconsulta (consulta interna) retorna una lista de valores de un solo campo (codigo) que la consulta exterior luego emplea al recuperar los datos.

Podemos reemplazar por un "join" la consulta anterior:

```
select distinct nombre
from editoriales as e
join libros
on codigoeditorial=e.codigo
where autor='Richard Bach';
```

Una combinación (join) siempre puede ser expresada como una subconsulta; pero una subconsulta no siempre puede reemplazarse por una combinación que retorne el mismo resultado. Si es posible, es aconsejable emplear combinaciones en lugar de subconsultas, son más eficientes.

Se recomienda probar las subconsultas antes de incluirlas en una consulta exterior, así puede verificar que retorna lo necesario, porque a veces resulta difícil verlo en consultas anidadas.

También podemos buscar valores No coincidentes con una lista de valores que retorna una subconsulta; por ejemplo, las editoriales que no han publicado libros de un autor específico:

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


SQL_ Subconsultas

Servidor de MySQL instalado en forma local.

Ingrese al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists editoriales;
drop table if exists libros;

create table editoriales(
  codigo int auto_increment,
  nombre varchar(30),
  primary key (codigo)
);

create table libros (
  codigo int auto_increment,
  titulo varchar(40),
  autor varchar(30),
  codigoeditorial smallint,
  primary key(codigo)
);

insert into editoriales(nombre) values('Planeta');
insert into editoriales(nombre) values('Emece');
insert into editoriales(nombre) values('Paidos');
insert into editoriales(nombre) values('Siglo XXI');

insert into libros(titulo,autor,codigoeditorial) values('Uno','Richard Bach',1);
insert into libros(titulo,autor,codigoeditorial) values('Ilusiones','Richard Bach',1);
insert into libros(titulo,autor,codigoeditorial) values('Aprenda PHP','Mario Molina',4);
insert into libros(titulo,autor,codigoeditorial) values('El aleph','Borges',2);
insert into libros(titulo,autor,codigoeditorial) values('Puente al infinito','Richard Bach',2);

-- nombre de las editoriales que han publicado libros del autor "Richard Bach":
select nombre
from editoriales
where codigo in
(select codigoeditorial
 from libros
 where autor='Richard Bach');

-- probamos la subconsulta separada de la consulta exterior para verificar que retorna
-- una lista de valores de un solo campo:
select codigoeditorial
from libros
where autor='Richard Bach';

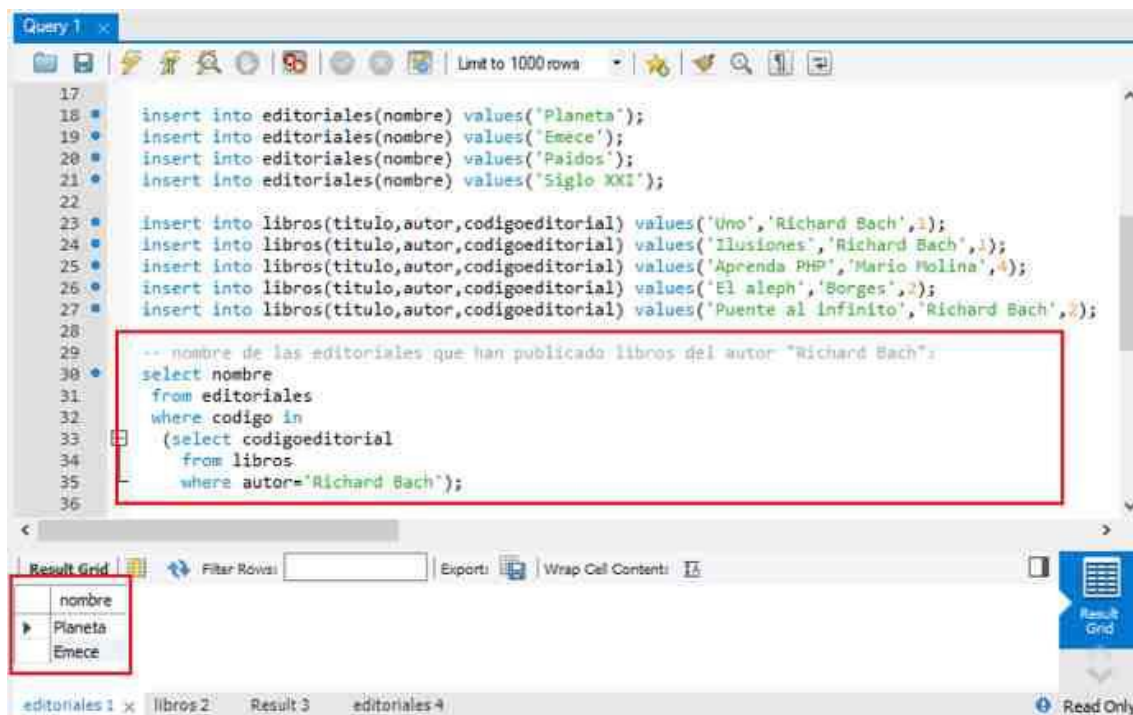
-- podemos reemplazar por un "join" la primera consulta:
select distinct nombre
```

SQL_ Subconsultas

```
from editoriales as e
join libros
on codigoeditorial=e.codigo
where autor='Richard Bach';
```

```
-- buscar las editoriales que no han publicado libros de "Richard Bach":
select nombre
from editoriales
where codigo not in
(select codigoeditorial
from libros
where autor='Richard Bach');
```

Genera una salida similar a esta:



The screenshot shows a SQL query editor with a query window titled "Query 1". The query is as follows:

```
17
18 insert into editoriales(nombre) values('Planeta');
19 insert into editoriales(nombre) values('Emece');
20 insert into editoriales(nombre) values('Paidós');
21 insert into editoriales(nombre) values('Siglo XXI');
22
23 insert into libros(titulo,autor,codigoeditorial) values('Uno','Richard Bach',1);
24 insert into libros(titulo,autor,codigoeditorial) values('Ilusiones','Richard Bach',1);
25 insert into libros(titulo,autor,codigoeditorial) values('Apeenda PHP','Mario Molina',4);
26 insert into libros(titulo,autor,codigoeditorial) values('El aleph','Borges',2);
27 insert into libros(titulo,autor,codigoeditorial) values('Puente al infinito','Richard Bach',2);
28
29 -- nombre de las editoriales que han publicado libros del autor "Richard Bach":
30 select nombre
31 from editoriales
32 where codigo in
33 (select codigoeditorial
34 from libros
35 where autor="Richard Bach");
36
```

The query is highlighted with a red box. Below the query, the result grid shows the names of the editoriales: Planeta and Emece.

nombre
Planeta
Emece

SQL_ Subconsultas

4. Subconsultas any - some - all

"any" y "some" son sinónimos. Chequean si alguna fila de la lista resultado de una subconsulta se encuentra el valor especificado en la condición.

Compara un valor escalar con los valores de un campo y devuelven "true" si la comparación con cada valor de la lista de la subconsulta es verdadera, sino "false".

El tipo de datos que se comparan deben ser compatibles.

La sintaxis básica es:

```
...VALORESCALAR OPERADORDECOMPARACION  
ANY (SUBCONSULTA);
```

Queremos saber los títulos de los libros de "Borges" que pertenecen a editoriales que han publicado también libros de "Richard Bach", es decir, si los libros de "Borges" coinciden con ALGUNA de las editoriales que publicó libros de "Richard Bach":

```
select titulo  
from libros  
where autor='Borges' and  
codigoeditorial = any  
(select e.codigo  
from editoriales as e  
join libros as l  
on codigoeditorial=e.codigo  
where l.autor='Richard Bach');
```

La consulta interna (subconsulta) retorna una lista de valores de un solo campo (puede ejecutar la subconsulta como una consulta para probarla), luego, la consulta externa compara cada valor de "codigoeditorial" con cada valor de la lista devolviendo los títulos de "Borges" que coinciden.

"all" también compara un valor escalar con una serie de valores. Chequea si TODOS los valores de la lista de la consulta externa se encuentran en la lista de valores devuelta por la consulta interna. Sintaxis:

```
VALORESCALAR OPERADORDECOMPARACION all (SUBCONSULTA);
```

Queremos saber si los libros de brorges que tiene un precio superiores a TODOS los libros de libros de "Richard Bach":

```
select *  
from libros  
where autor like '%Borges%' and  
precio > all  
(select precio  
from libros as l  
left join editoriales as e  
on codigoeditorial=e.codigo  
where l.autor like '%Bach%');
```

SQL_ Subconsultas

La consulta interna (subconsulta) retorna una lista de valores de un solo campo (puede 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.

Veamos otro ejemplo con un operador de comparación diferente:

Queremos saber si ALGUN precio de los libros de "Borges" es mayor a ALGUN precio de los libros de "Richard Bach":

```
select titulo,precio
from libros
where autor='Borges' and
precio > any
(select precio
from libros
where autor='Bach');
```

El precio de cada libro de "Borges" es comparado con cada valor de la lista de valores retornada por la subconsulta; si ALGUNO cumple la condición, es decir, es mayor a ALGUN precio de "Richard Bach", se lista.

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

```
select titulo,precio
from libros
where autor='borges' and
precio > all
(select precio
from libros
where autor='bach');
```

El precio de cada libro de "Borges" es comparado con cada valor de la lista de valores retornada 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".

Recuerde que solamente las subconsultas luego de un operador de comparación al cual es seguido por "any" o "all" pueden incluir cláusulas "group by".

SQL_ Subconsultas

Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists editoriales;
drop table if exists libros;

create table editoriales(
  codigo int auto_increment,
  nombre varchar(30),
  primary key (codigo)
);

create table libros (
  codigo int auto_increment,
  titulo varchar(40),
  autor varchar(30),
  codigoeditorial smallint,
  precio decimal(5,2),
  primary key(codigo)
);

insert into editoriales(nombre) values('Planeta');
insert into editoriales(nombre) values('Emece');
insert into editoriales(nombre) values('Paidos');
insert into editoriales(nombre) values('Siglo XXI');

insert into libros(titulo,autor,codigoeditorial,precio) values('Uno','Richard Bach',1,15);
insert into libros(titulo,autor,codigoeditorial,precio) values('Ilusiones','Richard Bach',4,18);
insert into libros(titulo,autor,codigoeditorial,precio) values('Puente al infinito','Richard Bach',2,20);
insert into libros(titulo,autor,codigoeditorial,precio) values('Aprenda PHP','Mario Molina',4,40);
insert into libros(titulo,autor,codigoeditorial,precio) values('El aleph','Borges',2,10);
insert into libros(titulo,autor,codigoeditorial,precio) values('Antología','Borges',1,20);
insert into libros(titulo,autor,codigoeditorial,precio) values('Cervantes y el quijote','Borges',3,25);

-- Queremos saber los títulos de los libros de "Borges" que pertenecen a editoriales que han
publicado también libros de "Richard Bach", es decir, si los libros de "Borges" coinciden con
ALGUNA de las editoriales que publicó libros de "Richard Bach"

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

SQL_ Subconsultas

-- Queremos visualizar los libros de "Borges" que tiene un precio superiores a alguno de los libros de libros de "Richard Bach"

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

-- Queremos visualizar los libros de "Borges" que tiene un precio superiores a TODOS los libros de libros de "Richard Bach"

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

SQL_ Subconsultas

5. Subconsultas correlacionadas

Una subconsulta correlacionada es una subconsulta que contiene una referencia a una tabla que también aparece en la consulta exterior. Por ejemplo:

```
SELECT * FROM t1 WHERE column1 = ANY
(SELECT column1 FROM t2 WHERE t2.column2 = t1.column2);
```

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: el nombre del artículo, el precio (unitario) y la cantidad. 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 as d
 where f.numero=d.numerofactura) as cantidad,
(select sum(d.precio*cantidad)
 from Detalles as d
 where f.numero=d.numerofactura) as total
from facturas as 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 MySQL repite la evaluación.

SQL_ Subconsultas

Servidor de MySQL instalado en forma local.

Ingrese al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists facturas;
drop table if exists detalles;

create table facturas(
  numero int not null,
  fecha date,
  cliente varchar(30),
  primary key(numero)
);

create table detalles(
  numerofactura int not null,
  numeroitem int not null,
  articulo varchar(30),
  precio decimal(5,2),
  cantidad int,
  primary key(numerofactura,numeroitem)
);

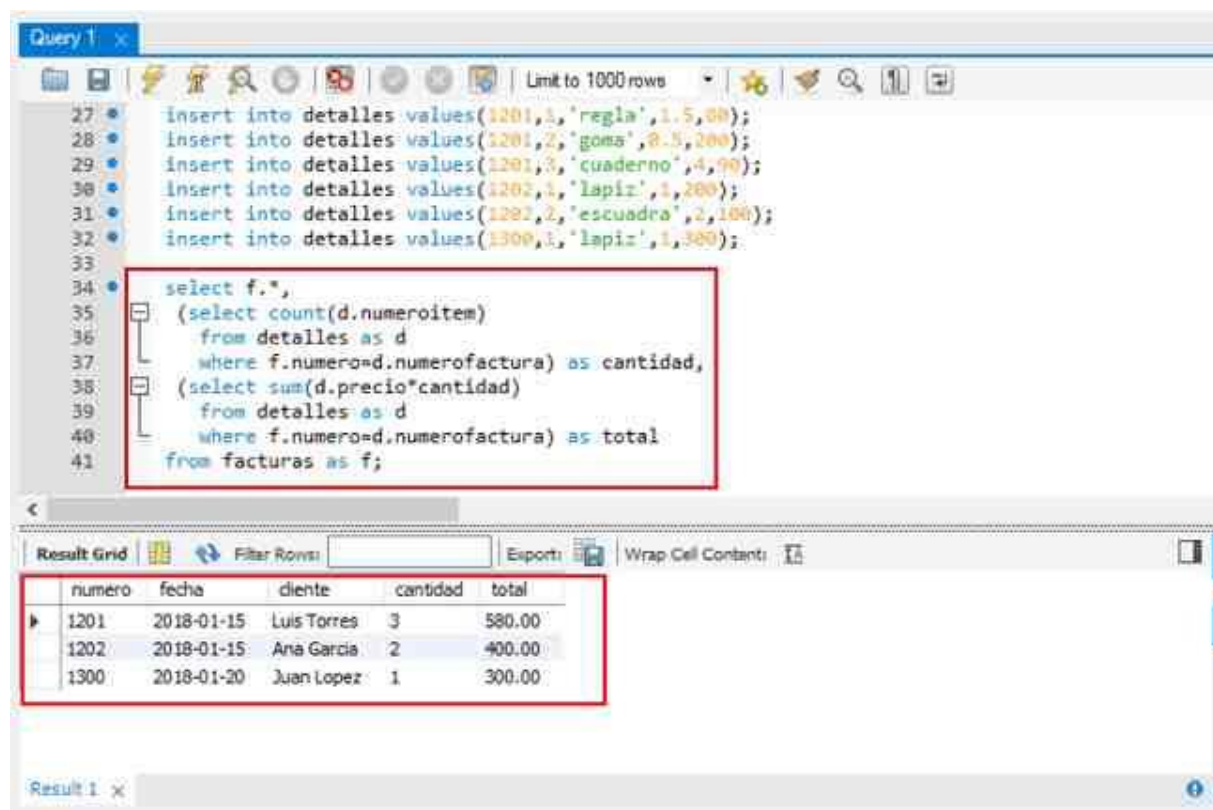
insert into facturas values(1200,'2018-01-15','Juan Lopez');
insert into facturas values(1201,'2018-01-15','Luis Torres');
insert into facturas values(1202,'2018-01-15','Ana Garcia');
insert into facturas values(1300,'2018-01-20','Juan Lopez');

insert into detalles values(1200,1,'lapiz',1,100);
insert into detalles values(1200,2,'goma',0.5,150);
insert into detalles values(1201,1,'regla',1.5,80);
insert into detalles values(1201,2,'goma',0.5,200);
insert into detalles values(1201,3,'cuaderno',4,90);
insert into detalles values(1202,1,'lapiz',1,200);
insert into detalles values(1202,2,'escuadra',2,100);
insert into detalles values(1300,1,'lapiz',1,300);

select f.*,
  (select count(d.numeroitem)
   from detalles as d
   where f.numero=d.numerofactura) as 'nº Lineas Factura',
  (select sum(d.precio*cantidad)
   from detalles as d
   where f.numero=d.numerofactura) as total
from facturas as f;
```


SQL_ Subconsultas

Genera una salida similar a esta:



The screenshot shows a SQL query editor with a toolbar at the top. The query text is as follows:

```
27 • insert into detalles values(1201,1,'regla',1.5,60);
28 • insert into detalles values(1201,2,'goma',0.5,200);
29 • insert into detalles values(1201,3,'cuaderno',4,90);
30 • insert into detalles values(1202,1,'lapis',1,200);
31 • insert into detalles values(1202,2,'escuadra',2,180);
32 • insert into detalles values(1300,1,'lapis',1,300);
33
34 • select f.*,
35   (select count(d.numeroitem)
36    from detalles as d
37    where f.numero=d.numerofactura) as cantidad,
38   (select sum(d.precio*cantidad)
39    from detalles as d
40    where f.numero=d.numerofactura) as total
41   from facturas as f;
```

Below the query editor is the 'Result Grid' section, which displays the following data:

numero	fecha	cliente	cantidad	total
1201	2018-01-15	Luis Torres	3	580.00
1202	2018-01-15	Ana Garcia	2	400.00
1300	2018-01-20	Juan Lopez	1	300.00

SQL_ Subconsultas

6. Subconsultas (Exists y No 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 retornan "true" (si las subconsultas retornan registros) o "false" (si las subconsultas no retornan registros).

Cuando se coloca en una subconsulta el operador "exists", MySQL analiza si hay datos que coinciden con la subconsulta, no se devuelve ningún registro, es como un test de existencia; MySQL 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 as f
where exists
(select * from detalles as d
 where f.numero=d.numerofactura
 and d.articulo='lapiz');
```

Puede obtener el mismo resultado empleando una combinación.

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

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

SQL_ Subconsultas

Servidor de MySQL instalado en forma local.

Ingrese al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists facturas;
drop table if exists detalles;

create table facturas(
  numero int not null,
  fecha date,
  cliente varchar(30),
  primary key(numero)
);

create table detalles(
  numerofactura int not null,
  numeroitem int not null,
  articulo varchar(30),
  precio decimal(5,2),
  cantidad int,
  primary key(numerofactura,numeroitem)
);

insert into facturas values(1200,'2018-01-15','Juan Lopez');
insert into facturas values(1201,'2018-01-15','Luis Torres');
insert into facturas values(1202,'2018-01-15','Ana Garcia');
insert into facturas values(1300,'2018-01-20','Juan Lopez');

insert into detalles values(1200,1,'lapiz',1,100);
insert into detalles values(1200,2,'goma',0.5,150);
insert into detalles values(1201,1,'regla',1.5,80);
insert into detalles values(1201,2,'goma',0.5,200);
insert into detalles values(1201,3,'cuaderno',4,90);
insert into detalles values(1202,1,'lapiz',1,200);
insert into detalles values(1202,2,'escuadra',2,100);
insert into detalles values(1300,1,'lapiz',1,300);

-- Visualizar Cliente y Factura de aquellos que hayan comprado un "lapiz"

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

-- Visualizar Cliente y Factura de aquellos que NO hayan comprado un "lapiz"
```

SQL_ Subconsultas

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

Genera una salida similar a esta:

Query 1

```
33
34
35 select cliente,numero
36 from facturas as f
37 where exists
38 (select * from detalles as d
39  where f.numero=d.numerofactura
40  and d.articulo='lapiz');
41
42 select cliente,numero
43 from facturas as f
44 where not exists
45 (select * from detalles as d
46  where f.numero=d.numerofactura
47  and d.articulo='lapiz');
```

Result Grid

cliente	numero
Juan Lopez	1200
Ana Garcia	1202
Juan Lopez	1300
...	...

facturas 1 x facturas 2

SQL_ Subconsultas

7. Subconsulta autocombinación

La autocombinación se utiliza para unir una tabla consigo misma, comparando valores de dos columnas con el mismo tipo de datos. La sintaxis es la siguiente:

```
SELECT
alias1.columna, alias2.columna, ...
FROM
tabla1 as alias1, tabla2 as alias2
WHERE
alias1.columna = alias2.columna
AND
otras condiciones
```

Algunas sentencias en las cuales la consulta interna y la externa emplean la misma tabla pueden reemplazarse por una autocombinación.

Por ejemplo, queremos una lista de los libros que han sido publicados por distintas editoriales.

```
select distinct l1.titulo
from libros as l1
where l1.titulo in
(select l2.titulo
 from libros as l2
 where l1.editorial <> l2.editorial);
```

En el ejemplo anterior empleamos una subconsulta correlacionada y las consultas interna y externa emplean la misma tabla. La subconsulta devuelve una lista de valores por ello se emplea "in" y sustituye una expresión en una cláusula "where".

Con el siguiente "join" se obtiene el mismo resultado:

```
select distinct l1.titulo
from libros as l1
join libros as l2
on l1.titulo=l1.titulo and
l1.autor=l2.autor
where l1.editorial<>l2.editorial;
```

Otro ejemplo: Buscamos todos los libros que tienen el mismo precio que "El aleph" empleando subconsulta:

```
select titulo
from libros
where titulo<>'El aleph' and
precio =
(select precio
 from libros
 where titulo='El aleph');
```

SQL_ Subconsultas

La subconsulta retorna un solo valor.

Buscamos los libros cuyo precio supere el precio promedio de los libros por editorial:

```
select l1.titulo,l1.editorial,l1.precio
from libros as l1
where l1.precio >
(select avg(l2.precio)
from libros as l2
where l1.editorial= l2.editorial);
```

Por cada valor de l1, se evalúa la subconsulta, si el precio es mayor que el promedio.

Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists libros;

create table libros(
codigo int auto_increment,
titulo varchar(40),
autor varchar(30),
editorial varchar(20),
precio decimal(5,2),
primary key(codigo)
);

insert into libros(titulo,autor,editorial,precio)
values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',20.00);
insert into libros(titulo,autor,editorial,precio)
values('Alicia en el pais de las maravillas','Lewis Carroll','Plaza',35.00);
insert into libros(titulo,autor,editorial,precio)
values('Aprenda PHP','Mario Molina','Siglo XXI',40.00);
insert into libros(titulo,autor,editorial,precio)
values('El aleph','Borges','Emece',10.00);
insert into libros(titulo,autor,editorial,precio)
values('Ilusiones','Richard Bach','Planeta',15.00);
insert into libros(titulo,autor,editorial,precio)
values('Java en 10 minutos','Mario Molina','Siglo XXI',50.00);
insert into libros(titulo,autor,editorial,precio)
values('Martin Fierro','Jose Hernandez','Planeta',20.00);
insert into libros(titulo,autor,editorial,precio)
values('Martin Fierro','Jose Hernandez','Emece',30.00);
insert into libros(titulo,autor,editorial,precio)
values('Uno','Richard Bach','Planeta',10.00);

-- Obtenemos la lista de los libros que han sido publicados
-- por distintas editoriales empleando una consulta correlacionada:
select distinct l1.titulo
```

SQL_ Subconsultas

```
from libros as l1
where l1.titulo in
(select l2.titulo
 from libros as l2
 where l1.editorial <> l2.editorial);

-- El siguiente "join" retorna el mismo resultado:
select distinct l1.titulo
from libros as l1
join libros as l2
on l1.titulo=l2.titulo
where l1.editorial<>l2.editorial;

-- Buscamos todos los libros que tienen el mismo precio que "El aleph" empleando subconsulta:
select titulo
from libros
where titulo<>'El aleph' and
precio =
(select precio
 from libros
 where titulo='El aleph');

-- Obtenemos la misma salida empleando "join":
select l1.titulo
from libros as l1
join libros as l2
on l1.precio=l2.precio
where l2.titulo='El aleph' and
l1.titulo<>l2.titulo;

-- Buscamos los libros cuyo precio supera el precio promedio de los libros por editorial:
select l1.titulo,l1.editorial,l1.precio
from libros as l1
where l1.precio >
(select avg(l2.precio)
 from libros as l2
 where l1.editorial= l2.editorial);

-- Obtenemos la misma salida pero empleando un "join" con "having":
select l1.editorial,l1.titulo,l1.precio
from libros as l1
join libros as l2
on l1.editorial=l2.editorial
group by l1.editorial, l1.titulo, l1.precio
having l1.precio > avg(l2.precio);
```

SQL_ Subconsultas

8. Subconsulta Update -Delete

Es posible que en algunas ocasiones necesitemos hacer una actualización a una tabla en donde la clave a utilizar en el where nos sea desconocida, y que sólo podamos obtenerla partiendo de una segunda clave en una segunda tabla.

La sintaxis básica para realizar actualizaciones con subconsulta es la siguiente:

```
update TABLA set CAMPO=NUEVOVALOR
where CAMPO = (SUBCONSULTA);
```

Actualizamos el precio de todos los libros de la editorial "Emece":

```
update libros set precio=precio+(precio*0.1)
where codigoeditorial=
(select codigo
 from editoriales
 where nombre='Emece');
```

La subconsulta retorna un único valor. También podemos hacerlo con un join.

La sintaxis básica para realizar eliminaciones con subconsulta es la siguiente:

```
delete from TABLA
where CAMPO = (SUBCONSULTA);
```

Eliminamos todos los libros de la editorial "Planeta":

```
delete from libros
where codigoeditorial =
(select e.codigo
 from editoriales as e
 where nombre='Planeta');
```


SQL_ Subconsultas

Servidor de MySQL instalado en forma local.

Ingresemos al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists editoriales;
drop table if exists libros;

create table editoriales(
  codigo int auto_increment,
  nombre varchar(30),
  primary key (codigo)
);

create table libros (
  codigo int auto_increment,
  titulo varchar(40),
  autor varchar(30),
  codigoeditorial smallint,
  precio decimal(5,2),
  primary key(codigo)
);

insert into editoriales(nombre) values('Planeta');
insert into editoriales(nombre) values('Emece');
insert into editoriales(nombre) values('Paidos');
insert into editoriales(nombre) values('Siglo XXI');

insert into libros(titulo,autor,codigoeditorial,precio)
  values('Uno','Richard Bach',1,15);
insert into libros(titulo,autor,codigoeditorial,precio)
  values('Ilusiones','Richard Bach',2,20);
insert into libros(titulo,autor,codigoeditorial,precio)
  values('El aleph','Borges',3,10);
insert into libros(titulo,autor,codigoeditorial,precio)
  values('Aprenda PHP','Mario Molina',4,40);
insert into libros(titulo,autor,codigoeditorial,precio)
  values('Poemas','Juan Perez',1,20);
insert into libros(titulo,autor,codigoeditorial,precio)
  values('Cuentos','Juan Perez',3,25);
insert into libros(titulo,autor,codigoeditorial,precio)
  values('Java en 10 minutos','Marcelo Perez',2,30);

select titulo,autor,nombre,precio from libros as l
  inner join editoriales as e on e.codigo=l.codigoeditorial;

SET SQL_SAFE_UPDATES = 0;
update libros set precio=precio+(precio*0.1)
  where codigoeditorial=
```

SQL_ Subconsultas

```
(select codigo
from editoriales
where nombre='Emece');
```

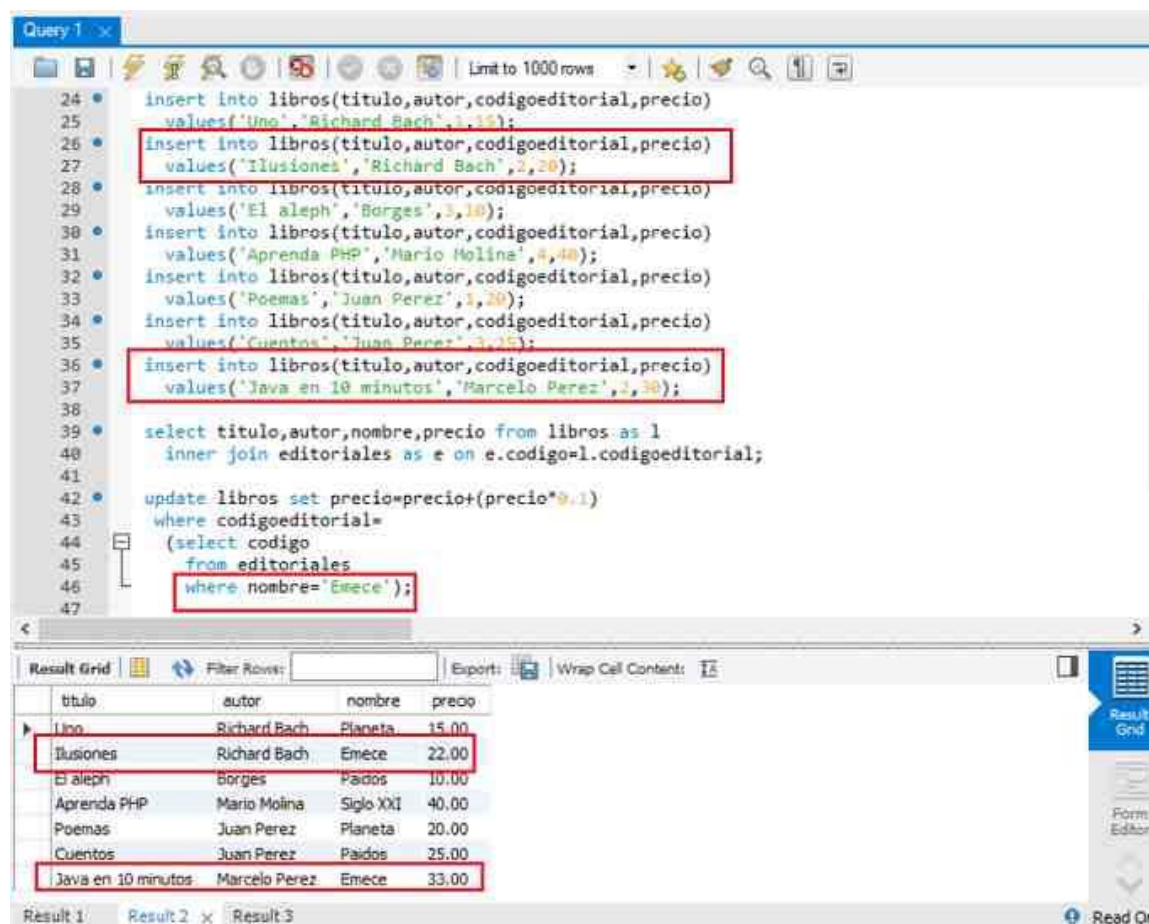
```
select titulo,autor,nombre,precio from libros as l
inner join editoriales as e on e.codigo=l.codigoeditorial;
```

```
SET SQL_SAFE_UPDATES = 0;
```

```
delete from libros
where codigoeditorial =
(select e.codigo
from editoriales as e
where nombre='Planeta');
```

```
select titulo,autor,nombre,precio from libros as l
inner join editoriales as e on e.codigo=l.codigoeditorial;
```

Genera una salida similar a esta:



The screenshot shows a SQL IDE interface. The top pane displays a SQL script with several queries. The bottom pane shows the 'Result Grid' with the output of the join query.

SQL Script:

```
24 insert into libros(titulo,autor,codigoeditorial,precio)
25 values('Uno','Richard Bach',1,15);
26 insert into libros(titulo,autor,codigoeditorial,precio)
27 values('Ilusiones','Richard Bach',2,20);
28 insert into libros(titulo,autor,codigoeditorial,precio)
29 values('El aleph','Borges',3,10);
30 insert into libros(titulo,autor,codigoeditorial,precio)
31 values('Aprenda PHP','Mario Molina',4,40);
32 insert into libros(titulo,autor,codigoeditorial,precio)
33 values('Poemas','Juan Perez',1,20);
34 insert into libros(titulo,autor,codigoeditorial,precio)
35 values('Cuentos','Juan Perez',3,25);
36 insert into libros(titulo,autor,codigoeditorial,precio)
37 values('Java en 10 minutos','Marcelo Perez',2,30);
38
39 select titulo,autor,nombre,precio from libros as l
40 inner join editoriales as e on e.codigo=l.codigoeditorial;
41
42 update libros set precio=precio+(precio*0.1)
43 where codigoeditorial=
44 (select codigo
45 from editoriales
46 where nombre='Emece');
```

Result Grid:

titulo	autor	nombre	precio
Uno	Richard Bach	Planeta	15.00
Ilusiones	Richard Bach	Emece	22.00
El aleph	Borges	Paidós	10.00
Aprenda PHP	Mario Molina	Siglo XXI	40.00
Poemas	Juan Perez	Planeta	20.00
Cuentos	Juan Perez	Paidós	25.00
Java en 10 minutos	Marcelo Perez	Emece	33.00

SQL_ Subconsultas

9. Subconsulta insert

Hemos visto que una subconsulta puede estar dentro de un "select", "update" y "delete"; también puede estar dentro de un "insert".

Podemos ingresar registros en una tabla empleando un "select".

La sintaxis básica es la siguiente:

```
insert into TABLAENQUESEINGRESA (CAMPOSTABLA1)
select (CAMPOSTABLACONSULTADA)
from TABLACONSULTADA;
```

Un profesor almacena las notas de sus alumnos en una tabla llamada "alumnos". Tiene otra tabla llamada "aprobados", con algunos campos iguales a la tabla "alumnos" pero en ella solamente almacenará los alumnos que han aprobado el ciclo.

Ingresamos registros en la tabla "aprobados" seleccionando registros de la tabla "alumnos":

```
insert into aprobados (documento,nota)
select (documento,nota)
from alumnos;
```

Entonces, se puede insertar registros en una tabla con la salida devuelta por una consulta a otra tabla; para ello escribimos la consulta y le antepone "insert into" junto al nombre de la tabla en la cual ingresaremos los registros y los campos que se cargarán (si se ingresan todos los campos no es necesario listarlos).

La cantidad de columnas devueltas en la consulta debe ser la misma que la cantidad de campos a cargar en el "insert".

Se pueden insertar valores en una tabla con el resultado de una consulta que incluya cualquier tipo de "join".

SQL_ Subconsultas

Servidor de MySQL instalado en forma local.

Ingrese al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists alumnos;
drop table if exists aprobados;

create table alumnos(
  documento char(8) not null,
  nombre varchar(30),
  nota decimal(4,2),
  primary key(documento)
);

create table aprobados(
  documento char(8) not null,
  nota decimal(4,2),
  primary key(documento)
);

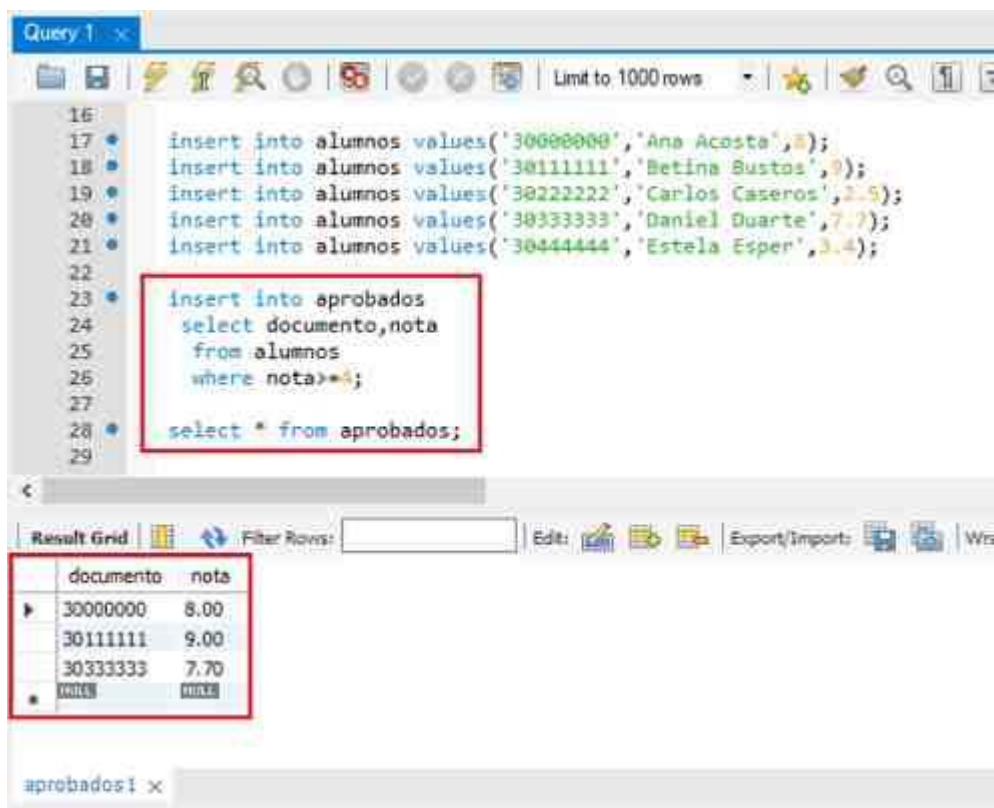
insert into alumnos values('30000000','Ana Acosta',8);
insert into alumnos values('30111111','Betina Bustos',9);
insert into alumnos values('30222222','Carlos Caseros',2.5);
insert into alumnos values('30333333','Daniel Duarte',7.7);
insert into alumnos values('30444444','Estela Esper',3.4);

insert into aprobados
select documento,nota
from alumnos
where nota>=4;

select * from aprobados;
```

SQL_ Subconsultas

Genera una salida similar a esta:



The screenshot shows a SQL query editor with a query named "Query 1". The query is as follows:

```
16  
17 insert into alumnos values('30000000','Ana Acosta',8);  
18 insert into alumnos values('30111111','Betina Bustos',9);  
19 insert into alumnos values('30222222','Carlos Caseros',2.5);  
20 insert into alumnos values('30333333','Daniel Duarte',7.7);  
21 insert into alumnos values('30444444','Estela Esper',3.4);  
22  
23 insert into aprobados  
24 select documento,nota  
25 from alumnos  
26 where nota>=4;  
27  
28 select * from aprobados;  
29
```

The result grid shows the data inserted into the 'aprobados' table:

documento	nota
30000000	8.00
30111111	9.00
30333333	7.70
30444444	3.40

SQL_ Subconsultas

10. Subconsulta tabla derivada

Se pueden emplear subconsultas que retornen un conjunto de registros de varios campos en lugar de una tabla.

Se la denomina tabla derivada y se coloca en la cláusula "from" para que la use un "select" externo.

La tabla derivada debe ir entre paréntesis y tener un alias para poder referenciarla. La sintaxis básica es la siguiente:

```
select ALIASdeTABLADERIVADA.CAMPO  
from (TABLADERIVADA) as ALIAS;
```

La tabla derivada es una subconsulta.

Podemos probar la consulta que retorna la tabla derivada y luego agregar el "select" externo:

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

La consulta anterior contiene una subconsulta correlacionada; retorna todos los datos de "facturas" y el monto total por factura de "detalles". Esta consulta retorna varios registros y varios campos y será la tabla derivada que emplearemos en la siguiente consulta:

```
select td.numero,c.nombre,td.total  
from clientes as c  
join (select f.*,  
(select sum(d.precio*cantidad)  
from Detalles as d  
where f.numero=d.numerofactura) as total  
from facturas as f) as td  
on td.codigocliente=c.codigo;
```

La consulta anterior retorna, de la tabla derivada (referenciada con "td") el número de factura y el monto total, y de la tabla "clientes", el nombre del cliente. Note que este "join" no emplea 2 tablas, sino una tabla propiamente dicha y una tabla derivada, que es en realidad una subconsulta.

SQL_ Subconsultas

Servidor de MySQL instalado en forma local.

Ingrese al programa "Workbench" y ejecutemos el siguiente bloque de instrucciones SQL:

```
drop table if exists clientes;
drop table if exists facturas;
drop table if exists detalles;

create table clientes(
  codigo int auto_increment,
  nombre varchar(30),
  domicilio varchar(30),
  primary key(codigo)
);

create table facturas(
  numero int not null,
  fecha date,
  codigocliente int not null,
  primary key(numero)
);

create table detalles(
  numerofactura int not null,
  numeroitem int not null,
  articulo varchar(30),
  precio decimal(5,2),
  cantidad int,
  primary key(numerofactura,numeroitem)
);

insert into clientes(nombre,domicilio) values('Juan Lopez','Colon 123');
insert into clientes(nombre,domicilio) values('Luis Torres','Sucre 987');
insert into clientes(nombre,domicilio) values('Ana Garcia','Sarmiento 576');

insert into facturas values(1200,'2007-01-15',1);
insert into facturas values(1201,'2007-01-15',2);
insert into facturas values(1202,'2007-01-15',3);
insert into facturas values(1300,'2007-01-20',1);

insert into detalles values(1200,1,'lapis',1,100);
insert into detalles values(1200,2,'goma',0.5,150);
insert into detalles values(1201,1,'regla',1.5,80);
insert into detalles values(1201,2,'goma',0.5,200);
insert into detalles values(1201,3,'cuaderno',4,90);
insert into detalles values(1202,1,'lapis',1,200);
insert into detalles values(1202,2,'escuadra',2,100);
insert into detalles values(1300,1,'lapis',1,300);
```

SQL_ Subconsultas

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

```
select td.numero,c.nombre,td.total  
from clientes as c  
join (select f.*,  
       (select sum(d.precio*cantidad)  
        from detalles as d  
        where f.numero=d.numerofactura) as total  
from facturas as f) as td  
on td.codigocliente=c.codigo;
```


SQL_ Subconsultas