# Subconsultas en SQL

Contenido

[Subconsultas en SQL 1](#_Toc377570669)

[Subconsultas 1](#_Toc377570670)

[Subconsultas como expresión 2](#_Toc377570671)

[Subconsultas con in 8](#_Toc377570672)

[Subconsultas any - some - all 16](#_Toc377570673)

[Subconsultas correlacionadas 28](#_Toc377570674)

[Exists y No Exists 35](#_Toc377570675)

[Subconsulta simil autocombinación 43](#_Toc377570676)

[Subconsulta en lugar de una tabla 50](#_Toc377570677)

[Subconsulta (update - delete) 59](#_Toc377570678)

[Subconsulta (insert) 65](#_Toc377570679)

[Crear tabla a partir de otra (select - into) 71](#_Toc377570680)

[Resumen Subconsultas 82](#_Toc377570681)

## 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 y estas últimas son más eficientes.

Las subconsultas se DEBEN incluir entre paréntesis.

Puede haber subconsultas dentro de subconsultas, se admiten hasta 32 niveles de anidación.

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:

* Las que retornan un solo valor escalar que se utiliza con un operador de comparación o en lugar de una expresión.
* Las que retornan una lista de valores, se combinan con "in", o los operadores "any", "some" y "all".
* Las 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.
* No se pueden emplear subconsultas que recuperen campos de tipos text o image.
* 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".
* No pueden emplearse las cláusulas "compute" y "compute by".
* "order by" puede emplearse solamente si se especifica "top" también.
* Una vista creada con una subconsulta no puede actualizarse.
* 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).

## 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.

Se pueden emplear en "select", "insert", "update" y "delete".

Para actualizar un registro empleando subconsulta la sintaxis básica es la siguiente:

update TABLA set CAMPO=NUEVOVALOR

where CAMPO= (SUBCONSULTA);

Para eliminar registros empleando subconsulta empleamos la siguiente sintaxis básica:

delete from TABLA

where CAMPO=(SUBCONSULTA);

Recuerde que 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" o "in").

No olvide que las subconsultas luego de un operador de comparación (que no es seguido por "any" o "all") no pueden incluir cláusulas "group by".

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla, si existe y la creamos:

if object\_id('libros') is not null

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30),

editorial varchar(20),

precio decimal(5,2)

);

Ingresamos los siguientes registros:

insert into libros values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',20.00);

insert into libros values('Alicia en el pais de las maravillas','Lewis Carroll','Plaza',35.00);

insert into libros values('Aprenda PHP','Mario Molina','Siglo XXI',40.00);

insert into libros values('El aleph','Borges','Emece',10.00);

insert into libros values('Ilusiones','Richard Bach','Planeta',15.00);

insert into libros values('Java en 10 minutos','Mario Molina','Siglo XXI',50.00);

insert into libros values('Martin Fierro','Jose Hernandez','Planeta',20.00);

insert into libros values('Martin Fierro','Jose Hernandez','Emece',30.00);

insert into libros values('Uno','Richard Bach','Planeta',10.00);

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

select titulo,precio,

precio-(select max(precio) from libros) as diferencia

from libros

where titulo='Uno';

Mostramos el título y precio del libro más costoso:

select titulo,autor, precio

from libros

where precio=

(select max(precio) from libros);

Actualizamos el precio del libro con máximo valor:

update libros set precio=45

where precio=

(select max(precio) from libros);

Eliminamos los libros con precio menor:

delete from libros

where precio=

(select min(precio) from libros);

**Código:**

if object\_id('libros') is not null

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30),

editorial varchar(20),

precio decimal(5,2)

);

insert into libros values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',20.00);

insert into libros values('Alicia en el pais de las maravillas','Lewis Carroll','Plaza',35.00);

insert into libros values('Aprenda PHP','Mario Molina','Siglo XXI',40.00);

insert into libros values('El aleph','Borges','Emece',10.00);

insert into libros values('Ilusiones','Richard Bach','Planeta',15.00);

insert into libros values('Java en 10 minutos','Mario Molina','Siglo XXI',50.00);

insert into libros values('Martin Fierro','Jose Hernandez','Planeta',20.00);

insert into libros values('Martin Fierro','Jose Hernandez','Emece',30.00);

insert into libros values('Uno','Richard Bach','Planeta',10.00);

select titulo,precio,

precio-(select max(precio) from libros) as diferencia

from libros

where titulo='Uno';

select titulo,autor, precio

from libros

where precio=

(select max(precio) from libros);

update libros set precio=45

where precio=

(select max(precio) from libros);

delete from libros

where precio=

(select min(precio) from libros);

**Problema:**

Un profesor almacena el documento, nombre y la nota final de cada alumno de su clase en una tabla llamada "alumnos".

1- Elimine la tabla, si existe:

if object\_id('alumnos') is not null

drop table alumnos;

2- Créela con los campos necesarios. Agregue una restricción "primary key" para el campo "documento" y una "check" para validar que el campo "nota" se encuentre entre los valores 0 y 10:

create table alumnos(

documento char(8),

nombre varchar(30),

nota decimal(4,2),

primary key(documento),

constraint CK\_alumnos\_nota\_valores check (nota>=0 and nota <=10),

);

3-Ingrese algunos registros:

insert into alumnos values('30111111','Ana Algarbe',5.1);

insert into alumnos values('30222222','Bernardo Bustamante',3.2);

insert into alumnos values('30333333','Carolina Conte',4.5);

insert into alumnos values('30444444','Diana Dominguez',9.7);

insert into alumnos values('30555555','Fabian Fuentes',8.5);

insert into alumnos values('30666666','Gaston Gonzalez',9.70);

4- Obtenga todos los datos de los alumnos con la nota más alta, empleando subconsulta.

2 registros.

5- realice la misma consulta anterior pero intente que la consulta interna retorne, además del máximo valor de precio, el título:

Mensaje de error, porque la lista de selección de una subconsulta que va luego de un operador de comparación puede incluir sólo un campo o expresión (excepto si se emplea "exists" o "in").

6- Muestre los alumnos que tienen una nota menor al promedio, su nota, y la diferencia con el promedio.

3 registros.

7- Cambie la nota del alumno que tiene la menor nota por 4.

1 registro modificado.

8- Elimine los alumnos cuya nota es menor al promedio.

3 registros eliminados.

**Solución:**

if object\_id('alumnos') is not null

drop table alumnos;

create table alumnos(

documento char(8),

nombre varchar(30),

nota decimal(4,2),

primary key(documento),

constraint CK\_alumnos\_nota\_valores check (nota>=0 and nota <=10),

);

insert into alumnos values('30111111','Ana Algarbe',5.1);

insert into alumnos values('30222222','Bernardo Bustamante',3.2);

insert into alumnos values('30333333','Carolina Conte',4.5);

insert into alumnos values('30444444','Diana Dominguez',9.7);

insert into alumnos values('30555555','Fabian Fuentes',8.5);

insert into alumnos values('30666666','Gaston Gonzalez',9.70);

select alumnos.\*

from alumnos

where nota=

(select max(nota) from alumnos);

select titulo,autor, precio

from libros

where precio=

(select titulo, max(precio) from libros);

select alumnos.\*,

(select avg(nota) from alumnos)-nota as diferencia

from alumnos

where nota<

(select avg(nota) from alumnos);

update alumnos set nota=4

where nota=

(select min(nota) from alumnos);

delete from alumnos

where nota<

(select avg(nota) from alumnos);

## 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 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 ha 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');

**Ejercicio:**

Trabajamos con las tablas "libros" y "editoriales" de una librería.

Eliminamos las tablas si existen y las creamos:

if object\_id('libros') is not null

drop table libros;

if object\_id('editoriales') is not null

drop table editoriales;

create table editoriales(

codigo tinyint identity,

nombre varchar(30),

primary key (codigo)

);

create table libros (

codigo int identity,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint,

primary key(codigo),

constraint FK\_libros\_editorial

foreign key (codigoeditorial)

references editoriales(codigo)

on update cascade,

);

Ingresamos algunos registros:

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Paidos');

insert into editoriales values('Siglo XXI');

insert into libros values('Uno','Richard Bach',1);

insert into libros values('Ilusiones','Richard Bach',1);

insert into libros values('Aprenda PHP','Mario Molina',4);

insert into libros values('El aleph','Borges',2);

insert into libros values('Puente al infinito','Richard Bach',2);

Queremos conocer el 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

from editoriales as e

join libros

on codigoeditorial=e.codigo

where autor='Richard Bach';

También podemos 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');

**Código:**

if object\_id('libros') is not null

drop table libros;

if object\_id('editoriales') is not null

drop table editoriales;

create table editoriales(

codigo tinyint identity,

nombre varchar(30),

primary key (codigo)

);

create table libros (

codigo int identity,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint,

primary key(codigo),

constraint FK\_libros\_editorial

foreign key (codigoeditorial)

references editoriales(codigo)

on update cascade,

);

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Paidos');

insert into editoriales values('Siglo XXI');

insert into libros values('Uno','Richard Bach',1);

insert into libros values('Ilusiones','Richard Bach',1);

insert into libros values('Aprenda PHP','Mario Molina',4);

insert into libros values('El aleph','Borges',2);

insert into libros values('Puente al infinito','Richard Bach',2);

select nombre

from editoriales

where codigo in

(select codigoeditorial

from libros

where autor='Richard Bach');

select codigoeditorial

from libros

where autor='Richard Bach';

select distinct nombre

from editoriales as e

join libros

on codigoeditorial=e.codigo

where autor='Richard Bach';

select nombre

from editoriales

where codigo not in

(select codigoeditorial

from libros

where autor='Richard Bach');

**Problema:**

Una empresa tiene registrados sus clientes en una tabla llamada "clientes", también tiene una tabla "ciudades" donde registra los nombres de las ciudades.

1- Elimine las tablas "clientes" y "ciudades", si existen:

if (object\_id('ciudades')) is not null

drop table ciudades;

if (object\_id('clientes')) is not null

drop table clientes;

2- Cree la tabla "clientes" (codigo, nombre, domicilio, ciudad, codigociudad) y "ciudades" (codigo, nombre). Agregue una restricción "primary key" para el campo "codigo" de ambas tablas y una "foreing key" para validar que el campo "codigociudad" exista en "ciudades" con actualización en cascada:

create table ciudades(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

codigociudad tinyint not null,

primary key(codigo),

constraint FK\_clientes\_ciudad

foreign key (codigociudad)

references ciudades(codigo)

on update cascade,

);

3- Ingrese algunos registros para ambas tablas:

insert into ciudades (nombre) values('Cordoba');

insert into ciudades (nombre) values('Cruz del Eje');

insert into ciudades (nombre) values('Carlos Paz');

insert into ciudades (nombre) values('La Falda');

insert into ciudades (nombre) values('Villa Maria');

insert into clientes values ('Lopez Marcos','Colon 111',1);

insert into clientes values ('Lopez Hector','San Martin 222',1);

insert into clientes values ('Perez Ana','San Martin 333',2);

insert into clientes values ('Garcia Juan','Rivadavia 444',3);

insert into clientes values ('Perez Luis','Sarmiento 555',3);

insert into clientes values ('Gomez Ines','San Martin 666',4);

insert into clientes values ('Torres Fabiola','Alem 777',5);

insert into clientes values ('Garcia Luis','Sucre 888',5);

4- Necesitamos conocer los nombres de las ciudades de aquellos clientes cuyo domicilio es en calle "San Martin", empleando subconsulta.

3 registros.

5- Obtenga la misma salida anterior pero empleando join.

6- Obtenga los nombres de las ciudades de los clientes cuyo apellido no comienza con una letra específica, empleando subconsulta.

2 registros.

7- Pruebe la subconsulta del punto 6 separada de la consulta exterior para verificar que retorna una lista de valores de un solo campo.

**Solución:**

if (object\_id('ciudades')) is not null

drop table ciudades;

if (object\_id('clientes')) is not null

drop table clientes;

create table ciudades(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

codigociudad tinyint not null,

primary key(codigo),

constraint FK\_clientes\_ciudad

foreign key (codigociudad)

references ciudades(codigo)

on update cascade,

);

insert into ciudades (nombre) values('Cordoba');

insert into ciudades (nombre) values('Cruz del Eje');

insert into ciudades (nombre) values('Carlos Paz');

insert into ciudades (nombre) values('La Falda');

insert into ciudades (nombre) values('Villa Maria');

insert into clientes values ('Lopez Marcos','Colon 111',1);

insert into clientes values ('Lopez Hector','San Martin 222',1);

insert into clientes values ('Perez Ana','San Martin 333',2);

insert into clientes values ('Garcia Juan','Rivadavia 444',3);

insert into clientes values ('Perez Luis','Sarmiento 555',3);

insert into clientes values ('Gomez Ines','San Martin 666',4);

insert into clientes values ('Torres Fabiola','Alem 777',5);

insert into clientes values ('Garcia Luis','Sucre 888',5);

select nombre

from ciudades

where codigo in

(select codigociudad

from clientes

where domicilio like 'San Martin %');

select distinct ci.nombre

from ciudades as ci

join clientes as cl

on codigociudad=ci.codigo

where domicilio like 'San Martin%';

select nombre

from ciudades

where codigo not in

(select codigociudad

from clientes

where nombre like 'G%');

select codigociudad

from clientes

where nombre like 'G%';

## 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 TODAS las editoriales que publicaron libros de "Borges" coinciden con TODAS las editoriales que publicaron libros de "Richard Bach":

select titulo

from libros

where autor='Borges' and

codigoeditorial = all

(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, 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".

**Ejercicio:**

Trabajamos con las tablas "libros" y "editoriales" de una librería.

Eliminamos las tablas si existen y las creamos:

if object\_id('libros') is not null

drop table libros;

if object\_id('editoriales') is not null

drop table editoriales;

create table editoriales(

codigo tinyint identity,

nombre varchar(30),

primary key (codigo)

);

create table libros (

codigo int identity,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint,

precio decimal(5,2),

primary key(codigo),

constraint FK\_libros\_editorial

foreign key (codigoeditorial)

references editoriales(codigo)

on update cascade,

);

Ingresamos algunos registros:

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Paidos');

insert into editoriales values('Siglo XXI');

insert into libros values('Uno','Richard Bach',1,15);

insert into libros values('Ilusiones','Richard Bach',4,18);

insert into libros values('Puente al infinito','Richard Bach',2,20);

insert into libros values('Aprenda PHP','Mario Molina',4,40);

insert into libros values('El aleph','Borges',2,10);

insert into libros values('Antología','Borges',1,20);

insert into libros values('Cervantes y el quijote','Borges',3,25);

Mostramos los títulos de los libros de "Borges" de editoriales que han publicado también 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%');

Realizamos la misma consulta pero empleando "all" en lugar de "any":

select titulo

from libros

where autor like '%Borges%' and

codigoeditorial = all

(select e.codigo

from editoriales as e

join libros as l

on codigoeditorial=e.codigo

where l.autor like '%Bach%');

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

select titulo,precio

from libros

where autor like '%Borges%' and

precio > any

(select precio

from libros

where autor like '%Bach%');

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

select titulo,precio

from libros

where autor like '%Borges%' and

precio > all

(select precio

from libros

where autor like '%Bach%');

Empleamos la misma subconsulta para eliminación:

delete from libros

where autor like '%Borges%' and

precio > all

(select precio

from libros

where autor like '%Bach%');

**Código:**

if object\_id('libros') is not null

drop table libros;

if object\_id('editoriales') is not null

drop table editoriales;

create table editoriales(

codigo tinyint identity,

nombre varchar(30),

primary key (codigo)

);

create table libros (

codigo int identity,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint,

precio decimal(5,2),

primary key(codigo),

constraint FK\_libros\_editorial

foreign key (codigoeditorial)

references editoriales(codigo)

on update cascade,

);

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Paidos');

insert into editoriales values('Siglo XXI');

insert into libros values('Uno','Richard Bach',1,15);

insert into libros values('Ilusiones','Richard Bach',4,18);

insert into libros values('Puente al infinito','Richard Bach',2,20);

insert into libros values('Aprenda PHP','Mario Molina',4,40);

insert into libros values('El aleph','Borges',2,10);

insert into libros values('Antología','Borges',1,20);

insert into libros values('Cervantes y el quijote','Borges',3,25);

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%');

select titulo

from libros

where autor like '%Borges%' and

codigoeditorial = all

(select e.codigo

from editoriales as e

join libros as l

on codigoeditorial=e.codigo

where l.autor like '%Bach%');

select titulo,precio

from libros

where autor like '%Borges%' and

precio > any

(select precio

from libros

where autor like '%Bach%');

select titulo,precio

from libros

where autor like '%Borges%' and

precio > all

(select precio

from libros

where autor like '%Bach%');

delete from libros

where autor like '%Borges%' and

precio > all

(select precio

from libros

where autor like '%Bach%');

**Problema:**

Un club dicta 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.

1- Elimine las tablas si existen:

if object\_id('inscritos') is not null

drop table inscritos;

if object\_id('socios') is not null

drop table socios;

2- Cree las tablas:

create table socios(

numero int identity,

documento char(8),

nombre varchar(30),

domicilio varchar(30),

primary key (numero)

);

create table inscritos (

numerosocio int not null,

deporte varchar(20) not null,

cuotas tinyint

constraint CK\_inscritos\_cuotas

check (cuotas>=0 and cuotas<=10)

constraint DF\_inscritos\_cuotas default 0,

primary key(numerosocio,deporte),

constraint FK\_inscritos\_socio

foreign key (numerosocio)

references socios(numero)

on update cascade

on delete cascade,

);

3- Ingrese algunos registros:

insert into socios values('23333333','Alberto Paredes','Colon 111');

insert into socios values('24444444','Carlos Conte','Sarmiento 755');

insert into socios values('25555555','Fabian Fuentes','Caseros 987');

insert into socios values('26666666','Hector Lopez','Sucre 344');

insert into inscritos values(1,'tenis',1);

insert into inscritos values(1,'basquet',2);

insert into inscritos values(1,'natacion',1);

insert into inscritos values(2,'tenis',9);

insert into inscritos values(2,'natacion',1);

insert into inscritos values(2,'basquet',default);

insert into inscritos values(2,'futbol',2);

insert into inscritos values(3,'tenis',8);

insert into inscritos values(3,'basquet',9);

insert into inscritos values(3,'natacion',0);

insert into inscritos values(4,'basquet',10);

4- Muestre el número de socio, el nombre del socio y el deporte en que está inscrito con un join de ambas tablas.

5- Muestre los socios que se serán compañeros en tenis y también en natación (empleando subconsulta)

3 filas devueltas.

6- vea si el socio 1 se ha inscrito en algún deporte en el cual se haya inscrito el socio 2.

3 filas.

7- Obtenga el mismo resultado anterior pero empleando join.

8- Muestre los deportes en los cuales el socio 2 pagó más cuotas que ALGUN deporte en los que se inscribió el socio 1.

2 registros.

9- Muestre los deportes en los cuales el socio 2 pagó más cuotas que TODOS los deportes en que se inscribió el socio 1.

1 registro.

10- Cuando un socio no ha pagado la matrícula de alguno de los deportes en que se ha inscrito, se le borra de la inscripción de todos los deportes. Elimine todos los socios que no pagaron ninguna cuota en algún deporte.

7 registros.

**Solución:**

if object\_id('inscritos') is not null

drop table inscritos;

if object\_id('socios') is not null

drop table socios;

create table socios(

numero int identity,

documento char(8),

nombre varchar(30),

domicilio varchar(30),

primary key (numero)

);

create table inscritos (

numerosocio int not null,

deporte varchar(20) not null,

cuotas tinyint

constraint CK\_inscritos\_cuotas

check (cuotas>=0 and cuotas<=10)

constraint DF\_inscritos\_cuotas default 0,

primary key(numerosocio,deporte),

constraint FK\_inscritos\_socio

foreign key (numerosocio)

references socios(numero)

on update cascade

on delete cascade,

);

insert into socios values('23333333','Alberto Paredes','Colon 111');

insert into socios values('24444444','Carlos Conte','Sarmiento 755');

insert into socios values('25555555','Fabian Fuentes','Caseros 987');

insert into socios values('26666666','Hector Lopez','Sucre 344');

insert into inscritos values(1,'tenis',1);

insert into inscritos values(1,'basquet',2);

insert into inscritos values(1,'natacion',1);

insert into inscritos values(2,'tenis',9);

insert into inscritos values(2,'natacion',1);

insert into inscritos values(2,'basquet',default);

insert into inscritos values(2,'futbol',2);

insert into inscritos values(3,'tenis',8);

insert into inscritos values(3,'basquet',9);

insert into inscritos values(3,'natacion',0);

insert into inscritos values(4,'basquet',10);

select numero,nombre,deporte

from socios as s

join inscritos as i

on numerosocio=numero;

select nombre

from socios

join inscritos as i

on numero=numerosocio

where deporte='natacion' and

numero= any

(select numerosocio

from inscritos as i

where deporte='tenis');

select deporte

from inscritos as i

where numerosocio=1 and

deporte= any

(select deporte

from inscritos as i

where numerosocio=2);

select i1.deporte

from inscritos as i1

join inscritos as i2

on i1.deporte=i2.deporte

where i1.numerosocio=1 and

i2.numerosocio=2;

select deporte

from inscritos as i

where numerosocio=2 and

cuotas>any

(select cuotas

from inscritos

where numerosocio=1);

select deporte

from inscritos as i

where numerosocio=2 and

cuotas>all

(select cuotas

from inscritos

where numerosocio=1);

delete from inscritos

where numerosocio=any

(select numerosocio

from inscritos

where cuotas=0);

## 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: 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.numero, f.fecha, f.cliente,

(select count(d.numeroitem)

from Detalles as d

where f.numero=d.numerofactura) as cantidad,

(select sum(d.preciounitario\*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 SQL Server repite la evaluación.

**Ejercicio:**

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.

Eliminamos las tablas si existen:

if object\_id('detalles') is not null

drop table detalles;

if object\_id('facturas') is not null

drop table facturas;

Las creamos con las siguientes estructuras:

create table facturas(

numero int not null,

fecha datetime,

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),

constraint FK\_detalles\_numerofactura

foreign key (numerofactura)

references facturas(numero)

on update cascade

on delete cascade,

);

Ingresamos algunos registros:

insert into facturas values(1200,'2007-01-15','Juan Lopez');

insert into facturas values(1201,'2007-01-15','Luis Torres');

insert into facturas values(1202,'2007-01-15','Ana Garcia');

insert into facturas values(1300,'2007-01-20','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);

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.numero, f.fecha, f.cliente,

(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;

**Código:**

if object\_id('detalles') is not null

drop table detalles;

if object\_id('facturas') is not null

drop table facturas;

create table facturas(

numero int not null,

fecha datetime,

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),

constraint FK\_detalles\_numerofactura

foreign key (numerofactura)

references facturas(numero)

on update cascade

on delete cascade,

);

insert into facturas values(1200,'2007-01-15','Juan Lopez');

insert into facturas values(1201,'2007-01-15','Luis Torres');

insert into facturas values(1202,'2007-01-15','Ana Garcia');

insert into facturas values(1300,'2007-01-20','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.numero, f.fecha, f.cliente,

(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;

**Problema:**

Un club dicta 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.

1- Elimine las tablas si existen:

if object\_id('inscritos') is not null

drop table inscritos;

if object\_id('socios') is not null

drop table socios;

2- Cree las tablas:

create table socios(

numero int identity,

documento char(8),

nombre varchar(30),

domicilio varchar(30),

primary key (numero)

);

create table inscritos (

numerosocio int not null,

deporte varchar(20) not null,

cuotas tinyint

constraint CK\_inscritos\_cuotas

check (cuotas>=0 and cuotas<=10)

constraint DF\_inscritos\_cuotas default 0,

primary key(numerosocio,deporte),

constraint FK\_inscritos\_socio

foreign key (numerosocio)

references socios(numero)

on update cascade

on delete cascade,

);

3- Ingrese algunos registros:

insert into socios values('23333333','Alberto Paredes','Colon 111');

insert into socios values('24444444','Carlos Conte','Sarmiento 755');

insert into socios values('25555555','Fabian Fuentes','Caseros 987');

insert into socios values('26666666','Hector Lopez','Sucre 344');

insert into inscritos values(1,'tenis',1);

insert into inscritos values(1,'basquet',2);

insert into inscritos values(1,'natacion',1);

insert into inscritos values(2,'tenis',9);

insert into inscritos values(2,'natacion',1);

insert into inscritos values(2,'basquet',default);

insert into inscritos values(2,'futbol',2);

insert into inscritos values(3,'tenis',8);

insert into inscritos values(3,'basquet',9);

insert into inscritos values(3,'natacion',0);

insert into inscritos values(4,'basquet',10);

4- Se necesita un listado de todos los socios que incluya nombre y domicilio, la cantidad de

deportes a los cuales se ha inscrito, empleando subconsulta.

4 registros.

5- Se necesita el nombre de todos los socios, el total de cuotas que debe pagar (10 por cada

deporte) y el total de cuotas pagas, empleando subconsulta.

4 registros.

6- Obtenga la misma salida anterior empleando join.

**Solución:**

if object\_id('inscritos') is not null

drop table inscritos;

if object\_id('socios') is not null

drop table socios;

create table socios(

numero int identity,

documento char(8),

nombre varchar(30),

domicilio varchar(30),

primary key (numero)

);

create table inscritos (

numerosocio int not null,

deporte varchar(20) not null,

cuotas tinyint

constraint CK\_inscritos\_cuotas

check (cuotas>=0 and cuotas<=10)

constraint DF\_inscritos\_cuotas default 0,

primary key(numerosocio,deporte),

constraint FK\_inscritos\_socio

foreign key (numerosocio)

references socios(numero)

on update cascade

on delete cascade,

);

insert into socios values('23333333','Alberto Paredes','Colon 111');

insert into socios values('24444444','Carlos Conte','Sarmiento 755');

insert into socios values('25555555','Fabian Fuentes','Caseros 987');

insert into socios values('26666666','Hector Lopez','Sucre 344');

insert into inscritos values(1,'tenis',1);

insert into inscritos values(1,'basquet',2);

insert into inscritos values(1,'natacion',1);

insert into inscritos values(2,'tenis',9);

insert into inscritos values(2,'natacion',1);

insert into inscritos values(2,'basquet',default);

insert into inscritos values(2,'futbol',2);

insert into inscritos values(3,'tenis',8);

insert into inscritos values(3,'basquet',9);

insert into inscritos values(3,'natacion',0);

insert into inscritos values(4,'basquet',10);

select nombre,domicilio,

(select count(\*)

from inscritos as i

where s.numero=i.numerosocio) as 'deportes'

from socios as s;

select nombre,

(select (count(\*)\*10)

from inscritos as i

where s.numero=i.numerosocio) as total,

(select sum(i.cuotas)

from inscritos as i

where s.numero=i.numerosocio) as pagas

from socios as s;

select nombre,

count(i.deporte)\*10 as total,

sum(i.cuotas) as pagas

from socios as s

join inscritos as i

on numero=numerosocio

group by nombre;

## 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", SQL Server analiza si hay datos que coinciden con la subconsulta, no se devuelve ningún registro, es como un test de existencia; SQL Server 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');

**Ejercicio:**

Un comercio que vende artículos de librería y papelería almacena la información de sus ventas en una tabla llamada "facturas" y otra "detalles".

Eliminamos las tablas si existen:

if object\_id('detalles') is not null

drop table detalles;

if object\_id('facturas') is not null

drop table facturas;

Las creamos con las siguientes estructuras:

create table facturas(

numero int not null,

fecha datetime,

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),

constraint FK\_detalles\_numerofactura

foreign key (numerofactura)

references facturas(numero)

on update cascade

on delete cascade

);

Ingresamos algunos registros:

insert into facturas values(1200,'2007-01-15','Juan Lopez');

insert into facturas values(1201,'2007-01-15','Luis Torres');

insert into facturas values(1202,'2007-01-15','Ana Garcia');

insert into facturas values(1300,'2007-01-20','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);

Empleamos una subconsulta correlacionada con un operador "exists" en la cláusula "where" para devolver la 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');

Buscamos los clientes que NO han comprado el artículo "lapiz":

select cliente,numero

from facturas as f

where not exists

(select \*from detalles as d

where f.numero=d.numerofactura

and d.articulo='lapiz');

**Código:**

if object\_id('detalles') is not null

drop table detalles;

if object\_id('facturas') is not null

drop table facturas;

create table facturas(

numero int not null,

fecha datetime,

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),

constraint FK\_detalles\_numerofactura

foreign key (numerofactura)

references facturas(numero)

on update cascade

on delete cascade

);

insert into facturas values(1200,'2007-01-15','Juan Lopez');

insert into facturas values(1201,'2007-01-15','Luis Torres');

insert into facturas values(1202,'2007-01-15','Ana Garcia');

insert into facturas values(1300,'2007-01-20','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 cliente,numero

from facturas as f

where exists

(select \*from detalles as d

where f.numero=d.numerofactura

and d.articulo='lapiz');

select cliente,numero

from facturas as f

where not exists

(select \*from detalles as d

where f.numero=d.numerofactura

and d.articulo='lapiz');

**Problema:**

Un club dicta 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.

1- Elimine las tablas si existen:

if object\_id('inscritos') is not null

drop table inscritos;

if object\_id('socios') is not null

drop table socios;

2- Cree las tablas:

create table socios(

numero int identity,

documento char(8),

nombre varchar(30),

domicilio varchar(30),

primary key (numero)

);

create table inscritos (

numerosocio int not null,

deporte varchar(20) not null,

cuotas tinyint

constraint CK\_inscritos\_cuotas

check (cuotas>=0 and cuotas<=10)

constraint DF\_inscritos\_cuotas default 0,

primary key(numerosocio,deporte),

constraint FK\_inscritos\_socio

foreign key (numerosocio)

references socios(numero)

on update cascade

on delete cascade,

);

3- Ingrese algunos registros:

insert into socios values('23333333','Alberto Paredes','Colon 111');

insert into socios values('24444444','Carlos Conte','Sarmiento 755');

insert into socios values('25555555','Fabian Fuentes','Caseros 987');

insert into socios values('26666666','Hector Lopez','Sucre 344');

insert into inscritos values(1,'tenis',1);

insert into inscritos values(1,'basquet',2);

insert into inscritos values(1,'natacion',1);

insert into inscritos values(2,'tenis',9);

insert into inscritos values(2,'natacion',1);

insert into inscritos values(2,'basquet',default);

insert into inscritos values(2,'futbol',2);

insert into inscritos values(3,'tenis',8);

insert into inscritos values(3,'basquet',9);

insert into inscritos values(3,'natacion',0);

insert into inscritos values(4,'basquet',10);

4- Emplee una subconsulta con el operador "exists" para devolver la lista de socios que se

inscribieron en un determinado deporte.

3 registros.

5- Busque los socios que NO se han inscrito en un deporte determinado empleando "not exists".

1 registro.

6- Muestre todos los datos de los socios que han pagado todas las cuotas.

1 registro.

**Solución:**

if object\_id('inscritos') is not null

drop table inscritos;

if object\_id('socios') is not null

drop table socios;

create table socios(

numero int identity,

documento char(8),

nombre varchar(30),

domicilio varchar(30),

primary key (numero)

);

create table inscritos (

numerosocio int not null,

deporte varchar(20) not null,

cuotas tinyint

constraint CK\_inscritos\_cuotas

check (cuotas>=0 and cuotas<=10)

constraint DF\_inscritos\_cuotas default 0,

primary key(numerosocio,deporte),

constraint FK\_inscritos\_socio

foreign key (numerosocio)

references socios(numero)

on update cascade

on delete cascade,

);

insert into socios values('23333333','Alberto Paredes','Colon 111');

insert into socios values('24444444','Carlos Conte','Sarmiento 755');

insert into socios values('25555555','Fabian Fuentes','Caseros 987');

insert into socios values('26666666','Hector Lopez','Sucre 344');

insert into inscritos values(1,'tenis',1);

insert into inscritos values(1,'basquet',2);

insert into inscritos values(1,'natacion',1);

insert into inscritos values(2,'tenis',9);

insert into inscritos values(2,'natacion',1);

insert into inscritos values(2,'basquet',default);

insert into inscritos values(2,'futbol',2);

insert into inscritos values(3,'tenis',8);

insert into inscritos values(3,'basquet',9);

insert into inscritos values(3,'natacion',0);

insert into inscritos values(4,'basquet',10);

select nombre

from socios as s

where exists

(select numerosocio,deporte,cuotas from inscritos as i

where s.numero=i.numerosocio

and i.deporte='natacion');

select nombre

from socios as s

where not exists

(select numerosocio,deporte,cuotas from inscritos as i

where s.numero=i.numerosocio

and i.deporte='natacion');

select s.numero, s.documento, s.nombre, s.domicilio

from socios as s

where exists

(select numerosocio,deporte,cuotas from inscritos as i

where s.numero=i.numerosocio

and i.cuotas=10);

## Subconsulta simil autocombinación

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');

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.

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla, si existe y la creamos:

if object\_id('libros') is not null

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30),

editorial varchar(20),

precio decimal(5,2)

);

Ingresamos los siguientes registros:

insert into libros values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',20.00);

insert into libros values('Alicia en el pais de las maravillas','Lewis Carroll','Plaza',35.00);

insert into libros values('Aprenda PHP','Mario Molina','Siglo XXI',40.00);

insert into libros values('El aleph','Borges','Emece',10.00);

insert into libros values('Ilusiones','Richard Bach','Planeta',15.00);

insert into libros values('Java en 10 minutos','Mario Molina','Siglo XXI',50.00);

insert into libros values('Martin Fierro','Jose Hernandez','Planeta',20.00);

insert into libros values('Martin Fierro','Jose Hernandez','Emece',30.00);

insert into libros 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

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);

**Código:**

if object\_id('libros') is not null

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30),

editorial varchar(20),

precio decimal(5,2)

);

insert into libros values('Alicia en el pais de las maravillas','Lewis Carroll','Emece',20.00);

insert into libros values('Alicia en el pais de las maravillas','Lewis Carroll','Plaza',35.00);

insert into libros values('Aprenda PHP','Mario Molina','Siglo XXI',40.00);

insert into libros values('El aleph','Borges','Emece',10.00);

insert into libros values('Ilusiones','Richard Bach','Planeta',15.00);

insert into libros values('Java en 10 minutos','Mario Molina','Siglo XXI',50.00);

insert into libros values('Martin Fierro','Jose Hernandez','Planeta',20.00);

insert into libros values('Martin Fierro','Jose Hernandez','Emece',30.00);

insert into libros values('Uno','Richard Bach','Planeta',10.00);

select distinct l1.titulo

from libros as l1

where l1.titulo in

(select l2.titulo

from libros as l2

where l1.editorial <> l2.editorial);

select distinct l1.titulo

from libros as l1

join libros as l2

on l1.titulo=l2.titulo

where l1.editorial<>l2.editorial;

select titulo

from libros

where titulo<>'El aleph' and

precio =

(select precio

from libros

where titulo='El aleph');

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;

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);

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);

**Problema:**

Un club dicta clases de distintos deportes a sus socios. El club tiene una tabla llamada "deportes" en la cual almacena el nombre del deporte, el nombre del profesor que lo dicta, el día de la semana que se dicta y el costo de la cuota mensual.

1- Elimine la tabla si existe:

if object\_id('deportes') is not null

drop table deportes;

2- Cree la tabla:

create table deportes(

nombre varchar(15),

profesor varchar(30),

dia varchar(10),

cuota decimal(5,2),

);

3- Ingrese algunos registros. Incluya profesores que dicten más de un curso:

insert into deportes values('tenis','Ana Lopez','lunes',20);

insert into deportes values('natacion','Ana Lopez','martes',15);

insert into deportes values('futbol','Carlos Fuentes','miercoles',10);

insert into deportes values('basquet','Gaston Garcia','jueves',15);

insert into deportes values('padle','Juan Huerta','lunes',15);

insert into deportes values('handball','Juan Huerta','martes',10);

4- Muestre los nombres de los profesores que dictan más de un deporte empleando subconsulta.

5- Obtenga el mismo resultado empleando join.

6- Buscamos todos los deportes que se dictan el mismo día que un determinado deporte (natacion) empleando subconsulta.

7- Obtenga la misma salida empleando "join".

**Solución:**

if object\_id('deportes') is not null

drop table deportes;

create table deportes(

nombre varchar(15),

profesor varchar(30),

dia varchar(10),

cuota decimal(5,2),

);

insert into deportes values('tenis','Ana Lopez','lunes',20);

insert into deportes values('natacion','Ana Lopez','martes',15);

insert into deportes values('futbol','Carlos Fuentes','miercoles',10);

insert into deportes values('basquet','Gaston Garcia','jueves',15);

insert into deportes values('padle','Juan Huerta','lunes',15);

insert into deportes values('handball','Juan Huerta','martes',10);

select distinct d1.profesor

from deportes as d1

where d1.profesor in

(select d2.profesor

from deportes as d2

where d1.nombre <> d2.nombre);

select distinct d1.profesor

from deportes as d1

join deportes as d2

on d1.profesor=d2.profesor

where d1.nombre<>d2.nombre;

select nombre

from deportes

where nombre<>'natacion' and

dia =

(select dia

from deportes

where nombre='natacion');

select d1.nombre

from deportes as d1

join deportes as d2

on d1.dia=d2.dia

where d2.nombre='natacion' and

d1.nombre<>d2.nombre;

## Subconsulta en lugar de una tabla

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 subsonsulta.

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

select f.numero, f.fecha, f.codigocliente,

(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.

**Ejercicio:**

Un comercio que vende artículos de librería y papelería almacena la información de sus ventas en una tabla llamada "facturas", otra "detalles" y otra "clientes".

Eliminamos las tablas si existen:

if object\_id('detalles') is not null

drop table detalles;

if object\_id('facturas') is not null

drop table facturas;

if object\_id('clientes') is not null

drop table clientes;

Las creamos con las siguientes estructuras:

create table clientes(

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

primary key(codigo)

);

create table facturas(

numero int not null,

fecha datetime,

codigocliente int not null,

primary key(numero),

constraint FK\_facturas\_cliente

foreign key (codigocliente)

references clientes(codigo)

on update cascade

);

create table detalles(

numerofactura int not null,

numeroitem int not null,

articulo varchar(30),

precio decimal(5,2),

cantidad int,

primary key(numerofactura,numeroitem),

constraint FK\_detalles\_numerofactura

foreign key (numerofactura)

references facturas(numero)

on update cascade

on delete cascade,

);

Ingresamos algunos registros:

insert into clientes values('Juan Lopez','Colon 123');

insert into clientes values('Luis Torres','Sucre 987');

insert into clientes 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,'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);

Vamos a realizar un "select" para recuperar el número de factura, el código de cliente, la fecha y la suma total de todas las facturas:

select f.numero, f.fecha, f.codigocliente

(select sum(d.precio\*cantidad)

from detalles as d

where f.numero=d.numerofactura) as total

from facturas as f;

Esta consulta contiene una subconsulta correlacionada.

Ahora utilizaremos el resultado de la consulta anterior como una tabla derivada que emplearemos en lugar de una tabla para realizar un "join" y recuperar el número de factura, el nombre del cliente y el monto total por factura:

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;

Note que para referenciar la tabla derivada debimos colocar un alias a la consulta.

**Código:**

if object\_id('detalles') is not null

drop table detalles;

if object\_id('facturas') is not null

drop table facturas;

if object\_id('clientes') is not null

drop table clientes;

create table clientes(

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

primary key(codigo)

);

create table facturas(

numero int not null,

fecha datetime,

codigocliente int not null,

primary key(numero),

constraint FK\_facturas\_cliente

foreign key (codigocliente)

references clientes(codigo)

on update cascade

);

create table detalles(

numerofactura int not null,

numeroitem int not null,

articulo varchar(30),

precio decimal(5,2),

cantidad int,

primary key(numerofactura,numeroitem),

constraint FK\_detalles\_numerofactura

foreign key (numerofactura)

references facturas(numero)

on update cascade

on delete cascade,

);

insert into clientes values('Juan Lopez','Colon 123');

insert into clientes values('Luis Torres','Sucre 987');

insert into clientes 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,'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.numero, f.fecha, f.codigocliente

(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;

**Problema:**

Un club dicta clases de distintos deportes. En una tabla llamada "socios" guarda los datos de los socios, en una tabla llamada "deportes" la información referente a los diferentes deportes que se dictan y en una tabla denominada "inscritos", las inscripciones de los socios a los distintos deportes.

Un socio puede inscribirse en varios deportes el mismo año. Un socio no puede inscribirse en el mismo deporte el mismo año. Distintos socios se inscriben en un mismo deporte en el mismo año.

1- Elimine las tablas si existen:

if object\_id('inscritos') is not null

drop table inscritos;

if object\_id('socios') is not null

drop table socios;

if object\_id('deportes') is not null

drop table deportes;

2- Cree las tablas con las siguientes estructuras:

create table socios(

documento char(8) not null,

nombre varchar(30),

domicilio varchar(30),

primary key(documento)

);

create table deportes(

codigo tinyint identity,

nombre varchar(20),

profesor varchar(15),

primary key(codigo)

);

create table inscritos(

documento char(8) not null,

codigodeporte tinyint not null,

año char(4),

matricula char(1),--'s'=paga, 'n'=impaga

primary key(documento,codigodeporte,año),

constraint FK\_inscritos\_socio

foreign key (documento)

references socios(documento)

on update cascade

on delete cascade

);

3- Ingrese algunos registros en las 3 tablas:

insert into socios values('22222222','Ana Acosta','Avellaneda 111');

insert into socios values('23333333','Betina Bustos','Bulnes 222');

insert into socios values('24444444','Carlos Castro','Caseros 333');

insert into socios values('25555555','Daniel Duarte','Dinamarca 44');

insert into deportes values('basquet','Juan Juarez');

insert into deportes values('futbol','Pedro Perez');

insert into deportes values('natacion','Marina Morales');

insert into deportes values('tenis','Marina Morales');

insert into inscritos values ('22222222',3,'2006','s');

insert into inscritos values ('23333333',3,'2006','s');

insert into inscritos values ('24444444',3,'2006','n');

insert into inscritos values ('22222222',3,'2005','s');

insert into inscritos values ('22222222',3,'2007','n');

insert into inscritos values ('24444444',1,'2006','s');

insert into inscritos values ('24444444',2,'2006','s');

4- Realice una consulta en la cual muestre todos los datos de las inscripciones, incluyendo el nombre del deporte y del profesor.

Esta consulta es un join.

5- Utilice el resultado de la consulta anterior como una tabla derivada para emplear en lugar de una tabla para realizar un "join" y recuperar el nombre del socio, el deporte en el cual está inscrito, el año, el nombre del profesor y la matrícula.

**Solución:**

if object\_id('inscritos') is not null

drop table inscritos;

if object\_id('socios') is not null

drop table socios;

if object\_id('deportes') is not null

drop table deportes;

create table socios(

documento char(8) not null,

nombre varchar(30),

domicilio varchar(30),

primary key(documento)

);

create table deportes(

codigo tinyint identity,

nombre varchar(20),

profesor varchar(15),

primary key(codigo)

);

create table inscritos(

documento char(8) not null,

codigodeporte tinyint not null,

año char(4),

matricula char(1),--'s'=paga, 'n'=impaga

primary key(documento,codigodeporte,año),

constraint FK\_inscritos\_socio

foreign key (documento)

references socios(documento)

on update cascade

on delete cascade

);

insert into socios values('22222222','Ana Acosta','Avellaneda 111');

insert into socios values('23333333','Betina Bustos','Bulnes 222');

insert into socios values('24444444','Carlos Castro','Caseros 333');

insert into socios values('25555555','Daniel Duarte','Dinamarca 44');

insert into deportes values('basquet','Juan Juarez');

insert into deportes values('futbol','Pedro Perez');

insert into deportes values('natacion','Marina Morales');

insert into deportes values('tenis','Marina Morales');

insert into inscritos values ('22222222',3,'2006','s');

insert into inscritos values ('23333333',3,'2006','s');

insert into inscritos values ('24444444',3,'2006','n');

insert into inscritos values ('22222222',3,'2005','s');

insert into inscritos values ('22222222',3,'2007','n');

insert into inscritos values ('24444444',1,'2006','s');

insert into inscritos values ('24444444',2,'2006','s');

select i.documento,i.codigodeporte,d.nombre as deporte, año, matricula, d.profesor

from deportes as d

join inscritos as i

on d.codigo=i.codigodeporte;

select s.nombre,td.deporte,td.profesor,td.año,td.matricula

from socios as s

join (select i.documento,i.codigodeporte,d.nombre as deporte, año, matricula, d.profesor

from deportes as d

join inscritos as i

on d.codigo=i.codigodeporte) as td

on td.documento=s.documento;

## Subconsulta (update - delete)

Dijimos que podemos emplear subconsultas en sentencias "insert", "update", "delete", además de "select".

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 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 in (SUBCONSULTA);

Eliminamos todos los libros de las editoriales que tienen publicados libros de "Juan Perez":

delete from libros

where codigoeditorial in

(select e.codigo

from editoriales as e

join libros

on codigoeditorial=e.codigo

where autor='Juan Perez');

La subconsulta es una combinación que retorna una lista de valores que la consulta externa emplea al seleccionar los registros para la eliminación.

**Ejercicio:**

Trabajamos con las tablas "libros" y "editoriales" de una librería.

Eliminamos las tablas si existen y las creamos:

if object\_id('libros') is not null

drop table libros;

if object\_id('editoriales') is not null

drop table editoriales;

create table editoriales(

codigo tinyint identity,

nombre varchar(30),

primary key (codigo)

);

create table libros (

codigo int identity,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint,

precio decimal(5,2),

primary key(codigo)

);

Ingresamos algunos registros:

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Paidos');

insert into editoriales values('Siglo XXI');

insert into libros values('Uno','Richard Bach',1,15);

insert into libros values('Ilusiones','Richard Bach',2,20);

insert into libros values('El aleph','Borges',3,10);

insert into libros values('Aprenda PHP','Mario Molina',4,40);

insert into libros values('Poemas','Juan Perez',1,20);

insert into libros values('Cuentos','Juan Perez',3,25);

insert into libros values('Java en 10 minutos','Marcelo Perez',2,30);

Actualizamos el precio de todos los libros de editorial "Emece" incrementándolos en un 10%:

update libros set precio=precio+(precio\*0.1)

where codigoeditorial=

(select codigo

from editoriales

where nombre='Emece');

Eliminamos todos los libros de las editoriales que tiene publicados libros de "Juan Perez":

delete from libros

where codigoeditorial in

(select e.codigo

from editoriales as e

join libros

on codigoeditorial=e.codigo

where autor='Juan Perez');

**Código:**

if object\_id('libros') is not null

drop table libros;

if object\_id('editoriales') is not null

drop table editoriales;

create table editoriales(

codigo tinyint identity,

nombre varchar(30),

primary key (codigo)

);

create table libros (

codigo int identity,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint,

precio decimal(5,2),

primary key(codigo)

);

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Paidos');

insert into editoriales values('Siglo XXI');

insert into libros values('Uno','Richard Bach',1,15);

insert into libros values('Ilusiones','Richard Bach',2,20);

insert into libros values('El aleph','Borges',3,10);

insert into libros values('Aprenda PHP','Mario Molina',4,40);

insert into libros values('Poemas','Juan Perez',1,20);

insert into libros values('Cuentos','Juan Perez',3,25);

insert into libros values('Java en 10 minutos','Marcelo Perez',2,30);

update libros set precio=precio+(precio\*0.1)

where codigoeditorial=

(select codigo

from editoriales

where nombre='Emece');

delete from libros

where codigoeditorial in

(select e.codigo

from editoriales as e

join libros

on codigoeditorial=e.codigo

where autor='Juan Perez');

**Problema:**

Un club dicta 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 si la matricula está o no paga, y una tabla denominada "socios" en la que guarda los datos personales de cada socio.

1- Elimine las tablas si existen:

if object\_id('inscritos') is not null

drop table inscritos;

if object\_id('socios') is not null

drop table socios;

2- Cree las tablas:

create table socios(

numero int identity,

documento char(8),

nombre varchar(30),

domicilio varchar(30),

primary key (numero)

);

create table inscritos (

numerosocio int not null,

deporte varchar(20) not null,

matricula char(1),-- 'n' o 's'

primary key(numerosocio,deporte),

constraint FK\_inscritos\_socio

foreign key (numerosocio)

references socios(numero)

);

3- Ingrese algunos registros:

insert into socios values('23333333','Alberto Paredes','Colon 111');

insert into socios values('24444444','Carlos Conte','Sarmiento 755');

insert into socios values('25555555','Fabian Fuentes','Caseros 987');

insert into socios values('26666666','Hector Lopez','Sucre 344');

insert into inscritos values(1,'tenis','s');

insert into inscritos values(1,'basquet','s');

insert into inscritos values(1,'natacion','s');

insert into inscritos values(2,'tenis','s');

insert into inscritos values(2,'natacion','s');

insert into inscritos values(2,'basquet','n');

insert into inscritos values(2,'futbol','n');

insert into inscritos values(3,'tenis','s');

insert into inscritos values(3,'basquet','s');

insert into inscritos values(3,'natacion','n');

insert into inscritos values(4,'basquet','n');

4- Actualizamos la cuota ('s') de todas las inscripciones de un socio determinado (por documento) empleando subconsulta.

5- Elimine todas las inscripciones de los socios que deben alguna matrícula.

5 registros eliminados.

**Solución:**

if object\_id('inscritos') is not null

drop table inscritos;

if object\_id('socios') is not null

drop table socios;

create table socios(

numero int identity,

documento char(8),

nombre varchar(30),

domicilio varchar(30),

primary key (numero)

);

create table inscritos (

numerosocio int not null,

deporte varchar(20) not null,

matricula char(1),-- 'n' o 's'

primary key(numerosocio,deporte),

constraint FK\_inscritos\_socio

foreign key (numerosocio)

references socios(numero)

);

insert into socios values('23333333','Alberto Paredes','Colon 111');

insert into socios values('24444444','Carlos Conte','Sarmiento 755');

insert into socios values('25555555','Fabian Fuentes','Caseros 987');

insert into socios values('26666666','Hector Lopez','Sucre 344');

insert into inscritos values(1,'tenis','s');

insert into inscritos values(1,'basquet','s');

insert into inscritos values(1,'natacion','s');

insert into inscritos values(2,'tenis','s');

insert into inscritos values(2,'natacion','s');

insert into inscritos values(2,'basquet','n');

insert into inscritos values(2,'futbol','n');

insert into inscritos values(3,'tenis','s');

insert into inscritos values(3,'basquet','s');

insert into inscritos values(3,'natacion','n');

insert into inscritos values(4,'basquet','n');

update inscritos set matricula='s'

where numerosocio=

(select numero

from socios

where documento='25555555');

delete from inscritos

where numerosocio in

(select numero

from socios as s

join inscritos

on numerosocio=numero

where matricula='n');

## Subconsulta (insert)

Aprendimos 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 where nota>=5;

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 anteponemos "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".

**Ejercicio:**

Un profesor almacena las notas de sus alumnos en una tabla llamada "alumnos" (documento, nombre, nota). Tiene otra tabla llamada "aprobados" (documento,nota) en la que guarda los alumnos que han aprobado el ciclo.

Eliminamos las tablas si existen:

if object\_id('alumnos') is not null

drop table alumnos;

if object\_id('aprobados') is not null

drop table aprobados;

Creamos las tablas:

create table alumnos(

documento char(8) not null,

nombre varchar(30),

nota decimal(4,2)

constraint CK\_alumnos\_nota\_valores check (nota>=0 and nota <=10),

primary key(documento),

);

create table aprobados(

documento char(8) not null,

nota decimal(4,2)

constraint CK\_aprobados\_nota\_valores check (nota>=0 and nota <=10),

primary key(documento),

);

Ingresamos registros en "alumnos":

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);

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

insert into aprobados

select documento,nota

from alumnos

where nota>=4;

Note que no se listan los campos en los cuales se cargan los datos porque tienen el mismo nombre que los de la tabla de la cual extraemos la información.

Veamos si los registros se han cargado:

select documento,nota from aprobados;

**Código:**

if object\_id('alumnos') is not null

drop table alumnos;

if object\_id('aprobados') is not null

drop table aprobados;

create table alumnos(

documento char(8) not null,

nombre varchar(30),

nota decimal(4,2)

constraint CK\_alumnos\_nota\_valores check (nota>=0 and nota <=10),

primary key(documento),

);

create table aprobados(

documento char(8) not null,

nota decimal(4,2)

constraint CK\_aprobados\_nota\_valores check (nota>=0 and nota <=10),

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 documento,nota from aprobados;

**Problema:**

Un comercio que vende artículos de librería y papelería almacena la información de sus ventas en una tabla llamada "facturas" y otra "clientes".

1- Elimine las tablas si existen:

if object\_id('facturas') is not null

drop table facturas;

if object\_id('clientes') is not null

drop table clientes;

2-Créelas:

create table clientes(

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

primary key(codigo)

);

create table facturas(

numero int not null,

fecha datetime,

codigocliente int not null,

total decimal(6,2),

primary key(numero),

constraint FK\_facturas\_cliente

foreign key (codigocliente)

references clientes(codigo)

on update cascade

);

3-Ingrese algunos registros:

insert into clientes values('Juan Lopez','Colon 123');

insert into clientes values('Luis Torres','Sucre 987');

insert into clientes values('Ana Garcia','Sarmiento 576');

insert into clientes values('Susana Molina','San Martin 555');

insert into facturas values(1200,'2007-01-15',1,300);

insert into facturas values(1201,'2007-01-15',2,550);

insert into facturas values(1202,'2007-01-15',3,150);

insert into facturas values(1300,'2007-01-20',1,350);

insert into facturas values(1310,'2007-01-22',3,100);

4- El comercio necesita una tabla llamada "clientespref" en la cual quiere almacenar el nombre y domicilio de aquellos clientes que han comprado hasta el momento más de 500 pesos en mercaderías.

Elimine la tabla si existe y créela con esos 2 campos:

if object\_id ('clientespref') is not null

drop table clientespref;

create table clientespref(

nombre varchar(30),

domicilio varchar(30)

);

5- Ingrese los registros en la tabla "clientespref" seleccionando registros de la tabla "clientes" y

"facturas".

6- Vea los registros de "clientespref":

**Solución:**

if object\_id('facturas') is not null

drop table facturas;

if object\_id('clientes') is not null

drop table clientes;

create table clientes(

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

primary key(codigo)

);

create table facturas(

numero int not null,

fecha datetime,

codigocliente int not null,

total decimal(6,2),

primary key(numero),

constraint FK\_facturas\_cliente

foreign key (codigocliente)

references clientes(codigo)

on update cascade

);

insert into clientes values('Juan Lopez','Colon 123');

insert into clientes values('Luis Torres','Sucre 987');

insert into clientes values('Ana Garcia','Sarmiento 576');

insert into clientes values('Susana Molina','San Martin 555');

insert into facturas values(1200,'2007-01-15',1,300);

insert into facturas values(1201,'2007-01-15',2,550);

insert into facturas values(1202,'2007-01-15',3,150);

insert into facturas values(1300,'2007-01-20',1,350);

insert into facturas values(1310,'2007-01-22',3,100);

if object\_id ('clientespref') is not null

drop table clientespref;

create table clientespref(

nombre varchar(30),

domicilio varchar(30)

);

insert into clientespref

select nombre,domicilio

from clientes

where codigo in

(select codigocliente

from clientes as c

join facturas as f

on codigocliente=codigo

group by codigocliente

having sum(total)>500);

select nombre,domicilio from clientespref;

## Crear tabla a partir de otra (select - into)

Podemos crear una tabla e insertar datos en ella en una sola sentencia consultando otra tabla (o varias) con esta sintaxis:

select CAMPOSNUEVATABLA

into NUEVATABLA

from TABLA

where CONDICION;

Es decir, se crea una nueva tabla y se inserta en ella el resultado de una consulta a otra tabla.

Tenemos la tabla "libros" de una librería y queremos crear una tabla llamada "editoriales" que contenga los nombres de las editoriales.

La tabla "editoriales", que no existe, contendrá solamente un campo llamado "nombre". La tabla libros contiene varios registros.

Podemos crear la tabla "editoriales" con el campo "nombre" consultando la tabla "libros" y en el mismo momento insertar la información:

select distinct editorial as nombre

into editoriales

from libros;

La tabla "editoriales" se ha creado con el campo "nombre" seleccionado del campo "editorial" de "libros".

Los campos de la nueva tabla tienen el mismo nombre, tipo de dato y valores almacenados que los campos listados de la tabla consultada; si se quiere dar otro nombre a los campos de la nueva tabla se deben especificar alias.

Entonces, luego de la lista de selección de campos de la tabla a consultar, se coloca "into" seguido del nombre de la nueva tabla y se sigue con la consulta.

Podemos emplear "group by", funciones de agrupamiento y "order by" en las consultas. También podemos emplear "select... into" con combinaciones, para crear una tabla que contenga datos de 2 o más tablas.

Podemos crear estas tablas como temporales.

Tipos de tablas temporales

Las tablas temporales son de dos tipos en cuanto al alcance la tabla. Tenemos tablas temporales locales y tablas temporales globales.

* #locales: Las tablas temporales locales tienen una # como primer carácter en su nombre y sólo se pueden utilizar en la conexión en la que el usuario las crea. Cuando la conexión termina la tabla temporal desaparece.
* ##globales Las tablas temporales globales comienzan con ## y son visibles por cualquier usuario conectado al SQL Server. Y una cosa más, estás tablas desaparecen cuando ningún usuario está haciendo referencias a ellas, no cuado se desconecta el usuario que la creo.

Realmente hay un tipo más de tablas temporales. Si creamos una tabla dentro de la base de datos temp es una tabla real en cuanto a que podemos utilizarla como cualquier otra tabla en cualquier base de datos, y es temporal en cuanto a que desaparece en cuanto apagamos el servidor.

**Ejercicio:**

Tenemos la tabla "libros" de una librería y queremos crear una tabla llamada "editoriales" que contenga los nombres de las editoriales.

Eliminamos las tablas "libros" y "editoriales" si existen:

if object\_id('libros')is not null

drop table libros;

if object\_id('editoriales')is not null

drop table editoriales;

Creamos la tabla "libros":

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(30),

editorial varchar(20),

precio decimal(5,2),

primary key(codigo)

);

Ingresamos algunos registros;

insert into libros values('Uno','Richard Bach','Planeta',15);

insert into libros values('El aleph','Borges','Emece',25);

insert into libros values('Matematica estas ahi','Paenza','Nuevo siglo',18);

insert into libros values('Aprenda PHP','Mario Molina','Nuevo siglo',45);

insert into libros values('Ilusiones','Richard Bach','Planeta',14);

insert into libros values('Java en 10 minutos','Mario Molina','Nuevo siglo',50);

Creamos una tabla llamada "editoriales" que contenga los nombres de las editoriales:

select distinct editorial as nombre

into editoriales

from libros;

Veamos la nueva tabla:

select nombre from editoriales;

Necesitamos una nueva tabla llamada "librosporeditorial" que contenga la cantidad de libros de cada editorial. Primero eliminamos la tabla, si existe:

if object\_id('cantidadporeditorial') is not null

drop table cantidadporeditorial;

Creamos la nueva tabla:

select editorial as nombre,count(\*) as cantidad

into cantidadporeditorial

from libros

group by editorial;

Veamos los registros de la nueva tabla:

select nombre,cantidad from cantidadporeditorial;

La tabla "cantidadporeditorial" se ha creado con el campo llamado "nombre" seleccionado del campo "editorial" de "libros" y con el campo "cantidad" con el valor calculado con count(\*) de la tabla "libros".

Queremos una tabla llamada "ofertas4" que contenga los mismos campos que "libros" y guarde los 4 libros más económicos. Primero eliminamos, si existe, la tabla "ofertas4":

if object\_id('ofertas4') is not null

drop table ofertas4;

Creamos "ofertas4" e insertamos la consulta de "libros":

select top 4 codigo,titulo,autor,editorial,precio

into ofertas4

from libros

order by precio desc;

La consulta anterior retorna los 4 primeros libros de la tabla "libros", ordenados en forma ascendente por precio y los almacena en la nueva tabla ("ofertas4"). Note que no se listan los campos a extraer, se coloca un asterisco para indicar que se incluyen todos los campos.

Veamos los registros de la nueva tabla:

select codigo,titulo,autor,editorial,precio from ofertas4;

Agregamos una columna a la tabla "editoriales" que contiene la ciudad en la cual está la casa central de cada editorial:

alter table editoriales add ciudad varchar(30);

Actualizamos dicho campo:

update editoriales set ciudad='Cordoba' where nombre='Planeta';

update editoriales set ciudad='Cordoba' where nombre='Emece';

update editoriales set ciudad='Buenos Aires' where nombre='Nuevo siglo';

Queremos una nueva tabla llamada "librosdecordoba" que contenga los títulos y autores de los libros de editoriales de Cordoba. En primer lugar, la eliminamos, si existe:

if object\_id('librosdecordoba') is not null

drop table librosdecordoba;

Consultamos las 2 tablas y guardamos el resultado en la nueva tabla que estamos creando:

select titulo,autor

into librosdecordoba

from libros

join editoriales

on editorial=nombre

where ciudad='Cordoba';

Consultamos la nueva tabla:

select titulo,autor from librosdecordoba;

**Código:**

if object\_id('libros')is not null

drop table libros;

if object\_id('editoriales')is not null

drop table editoriales;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(30),

editorial varchar(20),

precio decimal(5,2),

primary key(codigo)

);

insert into libros values('Uno','Richard Bach','Planeta',15);

insert into libros values('El aleph','Borges','Emece',25);

insert into libros values('Matematica estas ahi','Paenza','Nuevo siglo',18);

insert into libros values('Aprenda PHP','Mario Molina','Nuevo siglo',45);

insert into libros values('Ilusiones','Richard Bach','Planeta',14);

insert into libros values('Java en 10 minutos','Mario Molina','Nuevo siglo',50);

select distinct editorial as nombre

into editoriales

from libros;

select nombrefrom editoriales;

if object\_id('cantidadporeditorial') is not null

drop table cantidadporeditorial;

select editorial as nombre,count(\*) as cantidad

into cantidadporeditorial

from libros

group by editorial;

select nombre,cantidad from cantidadporeditorial;

if object\_id('ofertas4') is not null

drop table ofertas4;

select top 4 codigo,titulo,autor,editorial,precio

into ofertas4

from libros

order by precio desc;

select codigo,titulo,autor,editorial,precio from ofertas4;

alter table editoriales add ciudad varchar(30);

update editoriales set ciudad='Cordoba' where nombre='Planeta';

update editoriales set ciudad='Cordoba' where nombre='Emece';

update editoriales set ciudad='Buenos Aires' where nombre='Nuevo siglo';

if object\_id('librosdecordoba') is not null

drop table librosdecordoba;

select titulo,autor

into librosdecordoba

from libros

join editoriales

on editorial=nombre

where ciudad='Cordoba';

select titulo,autor from librosdecordoba;

**Problema:**

Un supermercado almacena los datos de sus empleados en una tabla denominada "empleados" y en una tabla llamada "sucursales" los códigos y ciudades de las diferentes sucursales.

1- Elimine las tablas "empleados" y "sucursales" si existen:

if object\_id('empleados')is not null

drop table empleados;

if object\_id('sucursales')is not null

drop table sucursales;

2- Cree la tabla "sucursales":

create table sucursales(

codigo int identity,

ciudad varchar(30) not null,

primary key(codigo)

);

3- Cree la tabla "empleados":

create table empleados(

documento char(8) not null,

nombre varchar(30) not null,

domicilio varchar(30),

seccion varchar(20),

sueldo decimal(6,2),

codigosucursal int,

primary key(documento),

constraint FK\_empleados\_sucursal

foreign key (codigosucursal)

references sucursales(codigo)

on update cascade

);

4- Ingrese algunos registros para ambas tablas:

insert into sucursales values('Cordoba');

insert into sucursales values('Villa Maria');

insert into sucursales values('Carlos Paz');

insert into sucursales values('Cruz del Eje');

insert into empleados values('22222222','Ana Acosta','Avellaneda 111','Secretaria',500,1);

insert into empleados values('23333333','Carlos Caseros','Colon 222','Sistemas',800,1);

insert into empleados values('24444444','Diana Dominguez','Dinamarca 333','Secretaria',550,2);

insert into empleados values('25555555','Fabiola Fuentes','Francia 444','Sistemas',750,2);

insert into empleados values('26666666','Gabriela Gonzalez','Guemes 555','Secretaria',580,3);

insert into empleados values('27777777','Juan Juarez','Jujuy 777','Secretaria',500,4);

insert into empleados values('28888888','Luis Lopez','Lules 888','Sistemas',780,4);

insert into empleados values('29999999','Maria Morales','Marina 999','Contaduria',670,4);

5- Realice un join para mostrar todos los datos de "empleados" incluyendo la ciudad de la sucursal.

6-Cree una tabla llamada "secciones" que contenga las secciones de la empresa (primero elimínela, si existe):

if object\_id('secciones') is not null

drop table secciones;

select distinct seccion as nombre

into secciones

from empleados;

7- Recupere la información de "secciones".

3 registros.

8- Se necesita una nueva tabla llamada "sueldosxseccion" que contenga la suma de los sueldos de los empleados por sección. Primero elimine la tabla, si existe:

if object\_id('sueldosxseccion') is not null

drop table sueldosxseccion;

9- Recupere los registros de la nueva tabla.

10- Se necesita una tabla llamada "maximossueldos" que contenga los mismos campos que "empleados" y guarde los 3 empleados con sueldos más altos. Primero eliminamos, si existe, la tabla "maximossueldos".

11- Vea los registros de la nueva tabla:

12- Se necesita una nueva tabla llamada "sucursalCordoba" que contenga los nombres y sección de los empleados de la ciudad de Córdoba. En primer lugar, eliminamos la tabla, si existe. Luego, consulte las tablas "empleados" y "sucursales" y guarde el resultado en la nueva tabla.

13- Consulte la nueva tabla.

**Solución:**

if object\_id('empleados')is not null

drop table empleados;

if object\_id('sucursales')is not null

drop table sucursales;

create table sucursales(

codigo int identity,

ciudad varchar(30) not null,

primary key(codigo)

);

create table empleados(

documento char(8) not null,

nombre varchar(30) not null,

domicilio varchar(30),

seccion varchar(20),

sueldo decimal(6,2),

codigosucursal int,

primary key(documento),

constraint FK\_empleados\_sucursal

foreign key (codigosucursal)

references sucursales(codigo)

on update cascade

);

insert into sucursales values('Cordoba');

insert into sucursales values('Villa Maria');

insert into sucursales values('Carlos Paz');

insert into sucursales values('Cruz del Eje');

insert into empleados values('22222222','Ana Acosta','Avellaneda 111','Secretaria',500,1);

insert into empleados values('23333333','Carlos Caseros','Colon 222','Sistemas',800,1);

insert into empleados values('24444444','Diana Dominguez','Dinamarca 333','Secretaria',550,2);

insert into empleados values('25555555','Fabiola Fuentes','Francia 444','Sistemas',750,2);

insert into empleados values('26666666','Gabriela Gonzalez','Guemes 555','Secretaria',580,3);

insert into empleados values('27777777','Juan Juarez','Jujuy 777','Secretaria',500,4);

insert into empleados values('28888888','Luis Lopez','Lules 888','Sistemas',780,4);

insert into empleados values('29999999','Maria Morales','Marina 999','Contaduria',670,4);

select documento,nombre,domicilio,seccion,sueldo,ciudad

from empleados

join sucursales on codigosucursal=codigo;

if object\_id('secciones') is not null

drop table secciones;

select distinct seccion as nombre

into secciones

from empleados;

select nombre from secciones;

if object\_id('sueldosxseccion') is not null

drop table sueldosxseccion;

select seccion, sum(sueldo) as total

into sueldosxseccion

from empleados

group by seccion;

select sección,total from sueldosxseccion;

if object\_id('maximossueldos') is not null

drop table maximossueldos;

select top 3 documento,nombre,domicilio,seccion,sueldo,ciudad

into maximossueldos

from empleados

order by sueldo;

select documento,nombre,domicilio,seccion,sueldo,ciudad from maximossueldos;

if object\_id('sucursalCordoba') is not null

drop table sucursalCordoba;

select nombre,ciudad

into sucursalCordoba

from empleados

join sucursales

on codigosucursal=codigo

where ciudad='Cordoba';

select nombre,ciudad from sucursalCordoba;

## Resumen Subconsultas

Una subconsulta es una instrucción

SELECT anidada dentro de una instrucción SELECT,

SELECT…INTO, INSERT…INTO, DELETE, o UPDATE o dentro de otra subconsulta.

Puede utilizar tres formas de sintaxis para crear una subconsulta:

comparación [ANY | ALL | SOME] (instrucción sql)

expresión [NOT] IN (instrucción sql)

[NOT] EXISTS (instrucción sql)

En donde:

* comparación: Es una expresión y un operador de comparación que compara la expresión con el resultado de la subconsulta.
* expresión: Es una expresión por la que se busca el conjunto resultante de la subconsulta.
* instrucción sql : Es una instrucción SELECT, que sigue el mismo formato y reglas que cualquier otra instrucción SELECT.
* Debe ir entre paréntesis.

Se puede utilizar una subconsulta en lugar de una expresión en la lista de campos de una instrucción SELECT o en una cláusula WHERE o HAVING. En una subconsulta, se utiliza una instrucción SELECT para proporcionar un conjunto de uno o más valores especificados para evaluar en la expresión de la cláusula WHERE o HAVING.

Se puede utilizar el predicado ANY o SOME, los cuales son sinónimos, para recuperar registros de la consulta principal, que satisfagan la comparación con cualquier otro registro recuperado en la subconsulta. El ejemplo siguiente devuelve todos los productos cuyo precio unitario es mayor que el de cualquier producto vendido con un descuento igual o mayor al 25 por ciento.:

SELECT NombreProducto, Precio\_Unidad

FROM Productos WHERE PrecioUnidad > ANY

(SELECT PrecioUnidad FROM DetallePedido WHERE Descuento >= 0 .25);

El predicado ALL se utiliza para recuperar únicamente aquellos registros de la consulta principal que satisfacen la comparación con todos los registros recuperados en la subconsulta. Si se cambia ANY por ALL en el ejemplo anterior, la consulta devolverá únicamente aquellos productos cuyo precio unitario sea mayor que el de todos los productos vendidos con un descuento igual o mayor al 25 por ciento. Esto es mucho más restrictivo.

El predicado IN se emplea para recuperar únicamente aquellos registros de la consulta principal para los que algunos registros de la subconsulta contienen un valor igual. El ejemplo siguiente devuelve todos los productos vendidos con un descuento igual o mayor al 25 por ciento:

SELECT NombreProducto, Precio\_Unidad FROM Productos WHERE IDProducto

IN

(SELECT IDProducto FROM DetallePedido WHERE Descuento >= 0.25);

Inversamente se puede utilizar NOT IN para recuperar únicamente aquellos registros de la consulta principal para los que no hay ningún registro de la subconsulta que contenga un valor igual. El predicado EXISTS (con la palabra reservada NOT opcional) se utiliza en comparaciones de verdad/falso para determinar si la subconsulta devuelve algún registro.

Se puede utilizar también alias del nombre de la tabla en una subconsulta para referirse a tablas listadas en la cláusula FROM fuera de la subconsulta. El ejemplo siguiente devuelve los nombres de los empleados cuyo salario es igual o mayor que el salario medio de todos los empleados con el mismo título. A la tabla Empleados se le ha dado el alias T1:

SELECT Apellido, Nombre, Titulo, Salario

FROM Empleados AS T1

WHERE Salario >= (SELECT Avg(Salario) FROM Empleados

WHERE T1.Titulo = Empleados.Titulo) ORDER BY Titulo;

En el ejemplo anterior, la palabra reservada AS es opcional.

SELECT Apellidos, Nombre, Cargo, Salario

FROM Empleados

WHERE Cargo LIKE "Agente Ven\*" AND Salario > ALL (SELECT Salario

FROM

Empleados WHERE (Cargo LIKE "\*Jefe\*") OR (Cargo LIKE "\*Director\*"));

Obtiene una lista con el nombre, cargo y salario de todos los agentes de ventas cuyo salario es mayor que el de todos los jefes y directores.

SELECT DISTINCTROW NombreProducto, Precio\_Unidad

FROM Productos

WHERE (Precio\_Unidad = (SELECT Precio\_Unidad FROM Productos WHERE

Nombre\_Producto = "Almíbar anisado");

Obtiene una lista con el nombre y el precio unitario de todos los productos con el mismo precio que el almíbar anisado.

SELECT DISTINCTROW Nombre\_Contacto,

Nombre\_Compañia, Cargo\_Contacto,

Telefono FROM Clientes WHERE (ID\_Cliente IN (SELECT DISTINCTROW

ID\_Cliente FROM Pedidos WHERE Fecha\_Pedido >= #04/1/93# <#07/1/93#);

Obtiene una lista de las compañías y los contactos de todos los clientes que han realizado un pedido en el segundo trimestre de 1993.

SELECT Nombre, Apellidos FROM Empleados

AS E WHERE EXISTS

(SELECT O.Id\_Producto FROM Pedidos AS O WHERE O.ID\_Empleado = E.ID\_Empleado);

Selecciona el nombre de todos los empleados que han reservado al menos un pedido.

SELECT DISTINCTROW Pedidos.Id\_Producto,

Pedidos.Cantidad,

(SELECT DISTINCTROW Productos.Nombre FROM Productos WHERE

Productos.Id\_Producto = Pedidos.Id\_Producto) AS ElProducto FROM

Pedidos WHERE Pedidos.Cantidad > 150 ORDER BY Pedidos.Id\_Producto;

Recupera el Código del Producto y la Cantidad pedida de la tabla pedidos, extrayendo el nombre del producto de la tabla de productos.