Contenido

[Trabajar con varias tablas 1](#_Toc377570730)

[Combinación interna (inner join) 1](#_Toc377570731)

[Combinación externa izquierda (left join) 8](#_Toc377570732)

[Combinación externa derecha (right join) 13](#_Toc377570733)

[Combinación externa completa (full join) 17](#_Toc377570734)

[Combinaciones cruzadas (cross join) 20](#_Toc377570735)

[Autocombinación 23](#_Toc377570736)

[Combinaciones y funciones de agrupamiento 27](#_Toc377570737)

[Combinación de más de dos tablas 30](#_Toc377570738)

[Combinaciones con update y delete 36](#_Toc377570739)

[Clave foránea 38](#_Toc377570740)

[Restricciones (foreign key) 38](#_Toc377570741)

[Restricciones foreign key en la misma tabla 43](#_Toc377570742)

[Restricciones foreign key (acciones) 46](#_Toc377570743)

[Restricciones foreign key deshabilitar y eliminar (with check - nocheck) 51](#_Toc377570744)

[Restricciones foreign key (información) 61](#_Toc377570745)

[Restricciones al crear la tabla 61](#_Toc377570746)

[Unión 68](#_Toc377570747)

# Trabajar con varias tablas

Hasta el momento hemos trabajado con una sola tabla, pero generalmente, se trabaja con más de una.

Para evitar la repetición de datos y ocupar menos espacio, se separa la información en varias tablas. Cada tabla almacena parte de la información que necesitamos registrar.

Por ejemplo, los datos de nuestra tabla "libros" podrían separarse en 2 tablas, una llamada "libros" y otra "editoriales" que guardará la información de las editoriales.

En nuestra tabla "libros" haremos referencia a la editorial colocando un código que la identifique.

Veamos:

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(30) not null default 'Desconocido',

codigoeditorial tinyint not null,

precio decimal(5,2),

primary key (codigo)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20) not null,

primary key(codigo)

);

De esta manera, evitamos almacenar tantas veces los nombres de las editoriales en la tabla "libros" y guardamos el nombre en la tabla "editoriales"; para indicar la editorial de cada libro agregamos un campo que hace referencia al código de la editorial en la tabla "libros" y en "editoriales".

Al recuperar los datos de los libros con la siguiente instrucción:

Select codigo,titulo,autor,codigoeditorial,precio from libros;

Vemos que en el campo "editorial" aparece el código, pero no sabemos el nombre de la editorial.

Para obtener los datos de cada libro, incluyendo el nombre de la editorial, necesitamos consultar ambas tablas, traer información de las dos.

Cuando obtenemos información de más de una tabla decimos que hacemos un "join" (combinación).

Veamos un ejemplo:

select codigo,titulo,autor,codigoeditorial,precio,codigo,nombre

from libros

join editoriales

on libros.codigoeditorial=editoriales.codigo;

Resumiendo: si distribuimos la información en varias tablas evitamos la redundancia de datos y ocupamos menos espacio físico en el disco. Un join es una operación que relaciona dos o más tablas para obtener un resultado que incluya datos (campos y registros) de ambas; las tablas participantes se combinan según los campos comunes a ambas tablas.

Hay tres tipos de combinaciones. En los siguientes capítulos explicamos cada una de ellas.

# Combinación interna (inner join)

Combinación interna (inner join)

Un join es una operación que relaciona dos o más tablas para obtener un resultado que incluya datos (campos y registros) de ambas; las tablas participantes se combinan según los campos comunes a ambas tablas.

Hay tres tipos de combinaciones:

combinaciones internas (inner join o join),

combinaciones externas y

combinaciones cruzadas.

También es posible emplear varias combinaciones en una consulta "select", incluso puede combinarse una tabla consigo misma.

La combinación interna emplea "join", que es la forma abreviada de "inner join". Se emplea para obtener información de dos tablas y combinar dicha información en una salida.

La sintaxis básica es la siguiente:

select CAMPOS

from TABLA1

join TABLA2

on CONDICIONdeCOMBINACION;

Ejemplo:

select codigo,titulo,autor,codigoeditorial,precio,codigo,nombre

from libros

join editoriales

on codigoeditorial=editoriales.codigo;

Analicemos la consulta anterior.

- especificamos los campos que aparecerán en el resultado en la lista de selección;

- indicamos el nombre de la tabla luego del "from" ("libros");

- combinamos esa tabla con "join" y el nombre de la otra tabla ("editoriales"); se especifica qué tablas se van a combinar y cómo;

- cuando se combina información de varias tablas, es necesario especificar qué registro de una tabla se combinará con qué registro de la otra tabla, con "on". Se debe especificar la condición para enlazarlas, es decir, el campo por el cual se combinarán, que tienen en común.

"on" hace coincidir registros de ambas tablas basándose en el valor de tal campo, en el ejemplo, el campo "codigoeditorial" de "libros" y el campo "codigo" de "editoriales" son los que enlazarán ambas tablas. Se emplean campos comunes, que deben tener tipos de datos iguales o similares.

La condicion de combinación, es decir, el o los campos por los que se van a combinar (parte "on"), se especifica según las claves primarias y externas.

Note que en la consulta, al nombrar el campo usamos el nombre de la tabla también. Cuando las tablas referenciadas tienen campos con igual nombre, esto es necesario para evitar confusiones y ambigüedades al momento de referenciar un campo. En el ejemplo, si no especificamos "editoriales.codigo" y solamente tipeamos "codigo", SQL Server no sabrá si nos referimos al campo "codigo" de "libros" o de "editoriales" y mostrará un mensaje de error indicando que "codigo" es ambiguo.

Entonces, si las tablas que combinamos tienen nombres de campos iguales, DEBE especificarse a qué tabla pertenece anteponiendo el nombre de la tabla al nombre del campo, separado por un punto (.).

Si una de las tablas tiene clave primaria compuesta, al combinarla con la otra, en la cláusula "on" se debe hacer referencia a la clave completa, es decir, la condición referenciará a todos los campos clave que identifican al registro.

Se puede incluir en la consulta join la cláusula "where" para restringir los registros que retorna el resultado; también "order by", "distinct", etc..

Se emplea este tipo de combinación para encontrar registros de la primera tabla que se correspondan con los registros de la otra, es decir, que cumplan la condición del "on". Si un valor de la primera tabla no se encuentra en la segunda tabla, el registro no aparece.

Para simplificar la sentencia podemos usar un alias para cada tabla:

select l.codigo,titulo,autor,nombre

from libros as l

join editoriales as e

on l.codigoeditorial=e.codigo;

En algunos casos (como en este ejemplo) el uso de alias es para fines de simplificación y hace más legible la consulta si es larga y compleja, pero en algunas consultas es absolutamente necesario.

**Ejercicio:**

Una librería almacena la información de sus libros para la venta en dos tablas, "libros" y "editoriales".

Eliminamos ambas tablas, si existen:

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

drop table libros;

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

drop table editoriales;

Creamos las tablas:

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30) default 'Desconocido',

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

Ingresamos algunos registros en ambas tablas:

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

insert into libros values('El aleph','Borges',2,20);

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',3,50);

insert into libros values('Java en 10 minutos',default,3,45);

Recuperamos los datos de libros:

select codigo,titulo,autor,codigoeditorial,precio

from libros;

Vemos que en el campo "editorial" aparece el código, pero no sabemos el nombre de la editorial.

Realizamos un join para obtener datos de ambas tablas (titulo, autor y nombre de la editorial):

select titulo, autor, nombre

from libros

join editoriales

on codigoeditorial=editoriales.codigo;

Mostramos el código del libro, título, autor, nombre de la editorial y el precio realizando un join y empleando alias:

select l.codigo,titulo,autor,nombre,precio

from libros as l

join editoriales as e

on codigoeditorial=e.codigo;

Note que al listar el campo "codigo" especificamos a qué tabla pertenece; si no lo hacemos SQL Server no sabrá si nos referimos al de la tabla "libros" o "editoriales". Los demás campos no tienen referencia a la tabla porque tienen nombres que no se repiten.

Realizamos la misma consulta anterior agregando un "where" para obtener solamente los libros de la editorial "Siglo XXI":

select l.codigo,titulo,autor,nombre,precio

from libros as l

join editoriales as e

on codigoeditorial=e.codigo

where e.nombre='Siglo XXI';

Obtenemos título, autor y nombre de la editorial, esta vez ordenados por título:

select titulo,autor,nombre

from libros as l

join editoriales as e

on codigoeditorial=e.codigo

order by titulo;

**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),

autor varchar(30) default 'Desconocido',

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

insert into libros values('El aleph','Borges',2,20);

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',3,50);

insert into libros values('Java en 10 minutos',default,3,45);

select codigo,titulo,autor,codigoeditorial,precio

from libros;

select titulo, autor, nombre

from libros

join editoriales

on codigoeditorial=editoriales.codigo;

select l.codigo,titulo,autor,nombre,precio

from libros as l

join editoriales as e

on codigoeditorial=e.codigo;

select l.codigo,titulo,autor,nombre,precio

from libros as l

join editoriales as e

on codigoeditorial=e.codigo

where e.nombre='Siglo XXI';

select titulo,autor,nombre

from libros as l

join editoriales as e

on codigoeditorial=e.codigo

order by titulo;

**Problema1:**

Una librería almacena la información de sus libros para la venta en dos tablas, "libros" y "editoriales".

Eliminamos ambas tablas, si existen:

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

drop table libros;

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

drop table editoriales;

Creamos las tablas:

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30) default 'Desconocido',

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

Ingresamos algunos registros en ambas tablas:

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

insert into libros values('El aleph','Borges',2,20);

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',3,50);

insert into libros values('Java en 10 minutos',default,3,45);

1.- Recuperamos los datos de libros

Vemos que en el campo "editorial" aparece el código, pero no sabemos el nombre de la editorial.

2.- Realizamos un join para obtener datos de ambas tablas (titulo, autor y nombre de la editorial)

3.- Mostramos el código del libro, título, autor, nombre de la editorial y el precio realizando un join y empleando alias.

Note que al listar el campo "codigo" especificamos a qué tabla pertenece; si no lo hacemos SQL Server no sabrá si nos referimos al de la tabla "libros" o "editoriales". Los demás campos no tienen referencia a la tabla porque tienen nombres que no se repiten.

4.- Realizamos la misma consulta anterior agregando un "where" para obtener solamente los libros de la editorial "Siglo XXI".

5.- Obtenemos título, autor y nombre de la editorial, esta vez ordenados por título.

**Solución1:**

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),

autor varchar(30) default 'Desconocido',

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

insert into libros values('El aleph','Borges',2,20);

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',3,50);

insert into libros values('Java en 10 minutos',default,3,45);

select codigo,titulo,autor,codigoeditorial,precio

from libros;

select titulo, autor, nombre

from libros

join editoriales

on codigoeditorial=editoriales.codigo;

select l.codigo,titulo,autor,nombre,precio

from libros as l

join editoriales as e

on codigoeditorial=e.codigo;

select l.codigo,titulo,autor,nombre,precio

from libros as l

join editoriales as e

on codigoeditorial=e.codigo

where e.nombre='Siglo XXI';

select titulo,autor,nombre

from libros as l

join editoriales as e

on codigoeditorial=e.codigo

order by titulo;

**Problema2:**

Una empresa tiene registrados sus clientes en una tabla llamada "clientes", también tiene una tabla

"provincias" donde registra los nombres de las provincias.

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

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

drop table clientes;

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

drop table provincias;

2- Créelas con las siguientes estructuras:

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

codigoprovincia tinyint not null,

primary key(codigo)

);

create table provincias(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

3- Ingrese algunos registros para ambas tablas:

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

insert into provincias (nombre) values('Santa Fe');

insert into provincias (nombre) values('Corrientes');

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

insert into clientes values ('Perez Ana','San Martin 222','Cruz del Eje',1);

insert into clientes values ('Garcia Juan','Rivadavia 333','Villa Maria',1);

insert into clientes values ('Perez Luis','Sarmiento 444','Rosario',2);

insert into clientes values ('Pereyra Lucas','San Martin 555','Cruz del Eje',1);

insert into clientes values ('Gomez Ines','San Martin 666','Santa Fe',2);

insert into clientes values ('Torres Fabiola','Alem 777','Ibera',3);

4- Obtenga los datos de ambas tablas, usando alias.

5- Obtenga la misma información anterior pero ordenada por nombre de provincia.

6- Recupere los clientes de la provincia "Santa Fe" (2 registros devueltos)

**Solución2:**

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

drop table clientes;

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

drop table provincias;

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

codigoprovincia tinyint not null,

primary key(codigo)

);

create table provincias(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

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

insert into provincias (nombre) values('Santa Fe');

insert into provincias (nombre) values('Corrientes');

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

insert into clientes values ('Perez Ana','San Martin 222','Cruz del Eje',1);

insert into clientes values ('Garcia Juan','Rivadavia 333','Villa Maria',1);

insert into clientes values ('Perez Luis','Sarmiento 444','Rosario',2);

insert into clientes values ('Pereyra Lucas','San Martin 555','Cruz del Eje',1);

insert into clientes values ('Gomez Ines','San Martin 666','Santa Fe',2);

insert into clientes values ('Torres Fabiola','Alem 777','Ibera',3);

select c.nombre,domicilio,ciudad,p.nombre

from clientes as c

join provincias as p

on c.codigoprovincia=p.codigo;

select c.nombre,domicilio,ciudad,p.nombre

from clientes as c

join provincias as p

on c.codigoprovincia=p.codigo

order by p.nombre;

select c.nombre,domicilio,ciudad

from clientes as c

join provincias as p

on c.codigoprovincia=p.codigo

where p.nombre='Santa Fe';

# Combinación externa izquierda (left join)

Vimos que una combinación interna (join) encuentra registros de la primera tabla que se correspondan con los registros de la segunda, es decir, que cumplan la condición del "on" y si un valor de la primera tabla no se encuentra en la segunda tabla, el registro no aparece.

Si queremos saber qué registros de una tabla NO encuentran correspondencia en la otra, es decir, no existe valor coincidente en la segunda, necesitamos otro tipo de combinación, "outer join" (combinación externa).

Las combinaciones externas combinan registros de dos tablas que cumplen la condición, más los registros de la segunda tabla que no la cumplen; es decir, muestran todos los registros de las tablas relacionadas, aún cuando no haya valores coincidentes entre ellas.

Este tipo de combinación se emplea cuando se necesita una lista completa de los datos de una de las tablas y la información que cumple con la condición. Las combinaciones externas se realizan solamente entre 2 tablas.

Hay tres tipos de combinaciones externas: "left outer join", "right outer join" y "full outer join"; se pueden abreviar con "left join", "right join" y "full join" respectivamente.

Vamos a estudiar las primeras.

Se emplea una combinación externa izquierda para mostrar todos los registros de la tabla de la izquierda. Si no encuentra coincidencia con la tabla de la derecha, el registro muestra los campos de la segunda tabla seteados a "null".

En el siguiente ejemplo solicitamos el título y nombre de la editorial de los libros:

select titulo,nombre

from editoriales as e

left join libros as l

on codigoeditorial = e.codigo;

El resultado mostrará el título y nombre de la editorial; las editoriales de las cuales no hay libros, es decir, cuyo código de editorial no está presente en "libros" aparece en el resultado, pero con el valor "null" en el campo "titulo".

Es importante la posición en que se colocan las tablas en un "left join", la tabla de la izquierda es la que se usa para localizar registros en la tabla de la derecha.

Entonces, un "left join" se usa para hacer coincidir registros en una tabla (izquierda) con otra tabla (derecha); si un valor de la tabla de la izquierda no encuentra coincidencia en la tabla de la derecha, se genera una fila extra (una por cada valor no encontrado) con todos los campos correspondientes a la tabla derecha seteados a "null". La sintaxis básica es la siguiente:

select CAMPOS

from TABLAIZQUIERDA

left join TABLADERECHA

on CONDICION;

En el siguiente ejemplo solicitamos el título y el nombre la editorial, la sentencia es similar a la anterior, la diferencia está en el orden de las tablas:

select titulo,nombre

from libros as l

left join editoriales as e

on codigoeditorial = e.codigo;

El resultado mostrará el título del libro y el nombre de la editorial; los títulos cuyo código de editorial no está presente en "editoriales" aparecen en el resultado, pero con el valor "null" en el campo "nombre".

Un "left join" puede tener clausula "where" que restringa el resultado de la consulta considerando solamente los registros que encuentran coincidencia en la tabla de la derecha, es decir, cuyo valor de código está presente en "libros":

select titulo,nombre

from editoriales as e

left join libros as l

on e.codigo=codigoeditorial

where codigoeditorial is not null;

También podemos mostrar las editoriales que NO están presentes en "libros", es decir, que NO encuentran coincidencia en la tabla de la derecha:

select titulo,nombre

from editoriales as e

left join libros as l

on e.codigo=codigoeditorial

where codigoeditorial is null;

**Ejercicio:**

Una librería almacena la información de sus libros para la venta en dos tablas, "libros" y "editoriales".

Eliminamos ambas 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 libros(

codigo int identity,

titulo varchar(40),

autor varchar(30) default 'Desconocido',

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

Ingresamos algunos registros en ambas tablas:

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

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

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',2,50);

insert into libros values('Java en 10 minutos',default,4,45);

Realizamos una combinación izquierda para obtener los datos de los libros, incluyendo el nombre de la editorial:

select titulo,nombre

from editoriales as e

left join libros as l

on codigoeditorial = e.codigo;

Las editoriales de las cuales no hay libros, es decir, cuyo código de editorial no está presente en "libros" aparece en el resultado, pero con el valor "null" en el campo "titulo".

Realizamos la misma consulta anterior pero cambiamos el orden de las tablas:

select titulo,nombre

from libros as l

left join editoriales as e

on codigoeditorial = e.codigo;

El resultado mostrará el título del libro y el nombre de la editorial; los títulos cuyo código de editorial no está presente en "editoriales" aparecen en el resultado, pero con el valor "null" en el campo "nombre".

Restringimos el resultado de una consulta considerando solamente los registros que encuentran coincidencia en la tabla de la derecha, es decir, cuyo valor de código está presente en "libros":

select titulo,nombre

from editoriales as e

left join libros as l

on e.codigo=codigoeditorial

where codigoeditorial is not null;

Mostramos las editoriales que NO están presentes en "libros", es decir, que NO encuentran coincidencia en la tabla de la derecha:

select titulo,nombre

from editoriales as e

left join libros as l

on e.codigo=codigoeditorial

where codigoeditorial is null;

**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),

autor varchar(30) default 'Desconocido',

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

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

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',2,50);

insert into libros values('Java en 10 minutos',default,4,45);

select titulo,nombre

from editoriales as e

left join libros as l

on codigoeditorial = e.codigo;

select titulo,nombre

from libros as l

left join editoriales as e

on codigoeditorial = e.codigo;

select titulo,nombre

from editoriales as e

left join libros as l

on e.codigo=codigoeditorial

where codigoeditorial is not null;

select titulo,nombre

from editoriales as e

left join libros as l

on e.codigo=codigoeditorial

where codigoeditorial is null;

**Problema:**

Una empresa tiene registrados sus clientes en una tabla llamada "clientes", también tiene una tabla

"provincias" donde registra los nombres de las provincias.

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

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

drop table clientes;

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

drop table provincias;

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

codigoprovincia tinyint not null,

primary key(codigo)

);

create table provincias(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

2- Ingrese algunos registros para ambas tablas:

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

insert into provincias (nombre) values('Santa Fe');

insert into provincias (nombre) values('Corrientes');

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

insert into clientes values ('Perez Ana','San Martin 222','Cruz del Eje',1);

insert into clientes values ('Garcia Juan','Rivadavia 333','Villa Maria',1);

insert into clientes values ('Perez Luis','Sarmiento 444','Rosario',2);

insert into clientes values ('Gomez Ines','San Martin 666','Santa Fe',2);

insert into clientes values ('Torres Fabiola','Alem 777','La Plata',4);

insert into clientes values ('Garcia Luis','Sucre 475','Santa Rosa',5);

3- Muestre todos los datos de los clientes, incluido el nombre de la provincia:

4- Realice la misma consulta anterior pero alterando el orden de las tablas:

5- Muestre solamente los clientes de las provincias que existen en "provincias" (5 registros):

6- Muestre todos los clientes cuyo código de provincia NO existe en "provincias" ordenados por nombre del cliente (2 registros):

7- Obtenga todos los datos de los clientes de "Cordoba" (3 registros):

**Solución:**

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

drop table clientes;

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

drop table provincias;

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

codigoprovincia tinyint not null,

primary key(codigo)

);

create table provincias(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

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

insert into provincias (nombre) values('Santa Fe');

insert into provincias (nombre) values('Corrientes');

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

insert into clientes values ('Perez Ana','San Martin 222','Cruz del Eje',1);

insert into clientes values ('Garcia Juan','Rivadavia 333','Villa Maria',1);

insert into clientes values ('Perez Luis','Sarmiento 444','Rosario',2);

insert into clientes values ('Gomez Ines','San Martin 666','Santa Fe',2);

insert into clientes values ('Torres Fabiola','Alem 777','La Plata',4);

insert into clientes values ('Garcia Luis','Sucre 475','Santa Rosa',5);

select c.nombre,domicilio,ciudad, p.nombre

from clientes as c

left join provincias as p

on codigoprovincia = p.codigo;

select c.nombre,domicilio,ciudad, p.nombre

from provincias as p

left join clientes as c

on codigoprovincia = p.codigo;

select c.nombre,domicilio,ciudad, p.nombre

from clientes as c

left join provincias as p

on codigoprovincia = p.codigo

where p.codigo is not null;

select c.nombre,domicilio,ciudad, p.nombre

from clientes as c

left join provincias as p

on codigoprovincia = p.codigo

where p.codigo is null

order by c.nombre;

select c.nombre,domicilio,ciudad, p.nombre

from clientes as c

left join provincias as p

on codigoprovincia = p.codigo

where p.nombre='Cordoba';

# Combinación externa derecha (right join)

Vimos que una combinación externa izquierda (left join) encuentra registros de la tabla izquierda que se correspondan con los registros de la tabla derecha y si un valor de la tabla izquierda no se encuentra en la tabla derecha, el registro muestra los campos correspondientes a la tabla de la derecha seteados a "null".

Una combinación externa derecha ("right outer join" o "right join") opera del mismo modo sólo que la tabla derecha es la que localiza los registros en la tabla izquierda.

En el siguiente ejemplo solicitamos el título y nombre de la editorial de los libros empleando un "right join":

select titulo,nombre

from libros as l

right join editoriales as e

on codigoeditorial = e.codigo;

El resultado mostrará el título y nombre de la editorial; las editoriales de las cuales no hay libros, es decir, cuyo código de editorial no está presente en "libros" aparece en el resultado, pero con el valor "null" en el campo "titulo".

Es FUNDAMENTAL tener en cuenta la posición en que se colocan las tablas en los "outer join". En un "left join" la primera tabla (izquierda) es la que busca coincidencias en la segunda tabla (derecha); en el "right join" la segunda tabla (derecha) es la que busca coincidencias en la primera tabla (izquierda).

En la siguiente consulta empleamos un "left join" para conseguir el mismo resultado que el "right join" anterior":

select titulo,nombre

from editoriales as e

left join libros as l

on codigoeditorial = e.codigo;

Note que la tabla que busca coincidencias ("editoriales") está en primer lugar porque es un "left join"; en el "right join" precedente, estaba en segundo lugar.

Un "right join" hace coincidir registros en una tabla (derecha) con otra tabla (izquierda); si un valor de la tabla de la derecha no encuentra coincidencia en la tabla izquierda, se genera una fila extra (una por cada valor no encontrado) con todos los campos correspondientes a la tabla izquierda seteados a "null". La sintaxis básica es la siguiente:

select CAMPOS

from TABLAIZQUIERDA

right join TABLADERECHA

on CONDICION;

Un "right join" también puede tener cláusula "where" que restringa el resultado de la consulta considerando solamente los registros que encuentran coincidencia en la tabla izquierda:

select titulo,nombre

from libros as l

right join editoriales as e

on e.codigo=codigoeditorial

where codigoeditorial is not null;

Mostramos las editoriales que NO están presentes en "libros", es decir, que NO encuentran coincidencia en la tabla de la derecha empleando un "right join":

select titulo,nombre

from libros as l

rightjoin editoriales as e

on e.codigo=codigoeditorial

where codigoeditorial is null;

**Ejercicio:**

Una librería almacena la información de sus libros para la venta en dos tablas, "libros" y "editoriales".

Eliminamos ambas 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 libros(

codigo int identity,

titulo varchar(40),

autor varchar(30) default 'Desconocido',

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

Ingresamos algunos registros en ambas tablas:

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

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

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',2,50);

insert into libros values('Java en 10 minutos',default,4,45);

Solicitamos el título y nombre de la editorial de los libros empleando un "right join":

select titulo,nombre

from libros as l

right join editoriales as e

on codigoeditorial = e.codigo;

Las editoriales de las cuales no hay libros, es decir, cuyo código de editorial no está presente en "libros" aparece en el resultado, pero con el valor "null" en el campo "titulo".

Realizamos la misma consulta anterior agregando un "where" que restringa el resultado considerando solamente los registros que encuentran coincidencia en la tabla izquierda:

select titulo,nombre

from libros as l

right join editoriales as e

on e.codigo=codigoeditorial

where codigoeditorial is not null;

Mostramos las editoriales que NO están presentes en "libros" (que NO encuentran coincidencia en "editoriales"):

select titulo,nombre

from libros as l

right join editoriales as e

on e.codigo=codigoeditorial

where codigoeditorial is null;

**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),

autor varchar(30) default 'Desconocido',

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

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

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',2,50);

insert into libros values('Java en 10 minutos',default,4,45);

select titulo,nombre

from libros as l

right join editoriales as e

on codigoeditorial = e.codigo;

select titulo,nombre

from libros as l

right join editoriales as e

on e.codigo=codigoeditorial

where codigoeditorial is not null;

select titulo,nombre

from libros as l

right join editoriales as e

on e.codigo=codigoeditorial

where codigoeditorial is null;

**Problema:**

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

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

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

drop table clientes;

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

drop table provincias;

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

codigoprovincia tinyint not null,

primary key(codigo)

);

create table provincias(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

2- Ingrese algunos registros para ambas tablas:

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

insert into provincias (nombre) values('Santa Fe');

insert into provincias (nombre) values('Corrientes');

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

insert into clientes values ('Perez Ana','San Martin 222','Cruz del Eje',1);

insert into clientes values ('Garcia Juan','Rivadavia 333','Villa Maria',1);

insert into clientes values ('Perez Luis','Sarmiento 444','Rosario',2);

insert into clientes values ('Gomez Ines','San Martin 666','Santa Fe',2);

insert into clientes values ('Torres Fabiola','Alem 777','La Plata',4);

insert into clientes values ('Garcia Luis','Sucre 475','Santa Rosa',5);

3- Muestre todos los datos de los clientes, incluido el nombre de la provincia empleando un "right

join".

4- Obtenga la misma salida que la consulta anterior pero empleando un "left join".

5- Empleando un "right join", muestre solamente los clientes de las provincias que existen en

"provincias" (5 registros)

6- Muestre todos los clientes cuyo código de provincia NO existe en "provincias" ordenados por

ciudad (2 registros)

**Solución:**

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

drop table clientes;

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

drop table provincias;

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

codigoprovincia tinyint not null,

primary key(codigo)

);

create table provincias(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

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

insert into provincias (nombre) values('Santa Fe');

insert into provincias (nombre) values('Corrientes');

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

insert into clientes values ('Perez Ana','San Martin 222','Cruz del Eje',1);

insert into clientes values ('Garcia Juan','Rivadavia 333','Villa Maria',1);

insert into clientes values ('Perez Luis','Sarmiento 444','Rosario',2);

insert into clientes values ('Gomez Ines','San Martin 666','Santa Fe',2);

insert into clientes values ('Torres Fabiola','Alem 777','La Plata',4);

insert into clientes values ('Garcia Luis','Sucre 475','Santa Rosa',5);

select c.nombre,domicilio,ciudad, p.nombre

from provincias as p

right join clientes as c

on codigoprovincia = p.codigo;

select c.nombre,domicilio,ciudad, p.nombre

from clientes as c

left join provincias as p

on codigoprovincia = p.codigo;

select c.nombre,domicilio,ciudad, p.nombre

from provincias as p

right join clientes as c

on codigoprovincia = p.codigo

where p.codigo is not null;

select c.nombre,domicilio,ciudad, p.nombre

from provincias as p

right join clientes as c

on codigoprovincia = p.codigo

where p.codigo is null

order by ciudad;

# Combinación externa completa (full join)

Vimos que un "left join" encuentra registros de la tabla izquierda que se correspondan con los registros de la tabla derecha y si un valor de la tabla izquierda no se encuentra en la tabla derecha, el registro muestra los campos correspondientes a la tabla de la derecha con valor "null". Aprendimos también que un "right join" opera del mismo modo sólo que la tabla derecha es la que localiza los registros en la tabla izquierda.

Una combinación externa completa ("full outer join" o "full join") retorna todos los registros de ambas tablas. Si un registro de una tabla izquierda no encuentra coincidencia en la tabla derecha, las columnas correspondientes a campos de la tabla derecha aparecen con valor "null", y si la tabla de la derecha no encuentra correspondencia en la tabla izquierda, los campos de esta última aparecen conteniendo "null".

Veamos un ejemplo:

select titulo,nombre

from editoriales as e

full join libros as l

on codigoeditorial = e.codigo;

La salida del "full join" precedente muestra todos los registros de ambas tablas, incluyendo los libros cuyo código de editorial no existe en la tabla "editoriales" y las editoriales de las cuales no hay correspondencia en "libros".

**Ejercicio:**

Una librería almacena la información de sus libros para la venta en dos tablas, "libros" y "editoriales".

Eliminamos ambas 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 libros(

codigo int identity,

titulo varchar(40),

autor varchar(30) default 'Desconocido',

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

Ingresamos algunos registros en ambas tablas:

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

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

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',2,50);

insert into libros values('Java en 10 minutos',default,4,45);

Realizamos una combinación externa completa para obtener todos los registros de ambas tablas, incluyendo los libros cuyo código de editorial no existe en la tabla "editoriales" y las editoriales de las cuales no hay correspondencia en "libros":

select titulo,nombre

from editoriales as e

full join libros as l

on codigoeditorial = e.codigo;

**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),

autor varchar(30) default 'Desconocido',

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

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

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',2,50);

insert into libros values('Java en 10 minutos',default,4,45);

select titulo,nombre

from editoriales as e

full join libros as l

on codigoeditorial = e.codigo;

**Problema:**

Un club dicta clases de distintos deportes. Almacena la información en una tabla llamada "deportes" en la cual incluye el nombre del deporte y el nombre del profesor y en otra tabla llamada

"inscritos" que incluye el documento del socio que se inscribe, el deporte y si la matricula está

paga o no.

1- Elimine las tablas si existen y cree las tablas:

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

drop table deportes;

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

drop table inscritos;

create table deportes(

codigo tinyint identity,

nombre varchar(30),

profesor varchar(30),

primary key (codigo)

);

create table inscritos(

documento char(8),

codigodeporte tinyint not null,

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

);

2- Ingrese algunos registros para ambas tablas:

insert into deportes values('tenis','Marcelo Roca');

insert into deportes values('natacion','Marta Torres');

insert into deportes values('basquet','Luis Garcia');

insert into deportes values('futbol','Marcelo Roca');

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

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

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

insert into inscritos values('22222222',2,'s');

insert into inscritos values('23333333',2,'s');

insert into inscritos values('22222222',4,'n');

insert into inscritos values('22222222',5,'n');

3- Muestre toda la información de la tabla "inscritos", y consulte la tabla "deportes" para obtener el nombre de cada deporte (6 registros)

4- Empleando un "left join" con "deportes" obtenga todos los datos de los inscritos (7 registros)

5- Obtenga la misma salida anterior empleando un "rigth join".

6- Muestre los deportes para los cuales no hay inscritos, empleando un "left join" (1 registro)

7- Muestre los documentos de los inscritos a deportes que no existen en la tabla "deportes" (1 registro)

8- Emplee un "full join" para obtener todos los datos de ambas tablas, incluyendo las inscripciones

a deportes inexistentes en "deportes" y los deportes que no tienen inscritos (8 registros)

**Solución:**

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

drop table deportes;

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

drop table inscritos;

create table deportes(

codigo tinyint identity,

nombre varchar(30),

profesor varchar(30),

primary key (codigo)

);

create table inscritos(

documento char(8),

codigodeporte tinyint not null,

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

);

insert into deportes values('tenis','Marcelo Roca');

insert into deportes values('natacion','Marta Torres');

insert into deportes values('basquet','Luis Garcia');

insert into deportes values('futbol','Marcelo Roca');

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

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

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

insert into inscritos values('22222222',2,'s');

insert into inscritos values('23333333',2,'s');

insert into inscritos values('22222222',4,'n');

insert into inscritos values('22222222',5,'n');

select documento,d.nombre,matricula

from inscritos as i

join deportes as d

on codigodeporte=codigo;

select documento,d.nombre,matricula

from inscritos as i

left join deportes as d

on codigodeporte=codigo;

select documento,d.nombre,matricula

from deportes as d

right join inscritos as i

on codigodeporte=codigo;

select nombre

from deportes as d

left join inscritos as i

on codigodeporte=codigo

where codigodeporte is null;

select documento

from inscritos as i

left join deportes as d

on codigodeporte=codigo

where codigo is null;

select documento,nombre,profesor,matricula

from inscritos as i

full join deportes as d

on codigodeporte=codigo;

# Combinaciones cruzadas (cross join)

Vimos que hay tres tipos de combinaciones: 1) combinaciones internas (join), 2) combinaciones externas (left, right y full join) y 3) combinaciones cruzadas.

Las combinaciones cruzadas (cross join) muestran todas las combinaciones de todos los registros de las tablas combinadas. Para este tipo de join no se incluye una condición de enlace. Se genera el producto cartesiano en el que el número de filas del resultado es igual al número de registros de la primera tabla multiplicado por el número de registros de la segunda tabla, es decir, si hay 5 registros en una tabla y 6 en la otra, retorna 30 filas.

La sintaxis básica es ésta:

select CAMPOS

from TABLA1

cross join TABLA2;

Veamos un ejemplo. Un pequeño restaurante almacena los nombres y precios de sus comidas en una tabla llamada "comidas" y en una tabla denominada "postres" los mismos datos de sus postres.

Si necesitamos conocer todas las combinaciones posibles para un menú, cada comida con cada postre, empleamos un "cross join":

select c.nombre as 'plato principal', p.nombre as 'postre'

from comidas as c

cross join postres as p;

La salida muestra cada plato combinado con cada uno de los postres.

Como cualquier tipo de "join", puede emplearse una cláusula "where" que condicione la salida.

**Ejercicio:**

Un pequeño restaurante tiene almacenados los nombres y precios de sus comidas en una tabla llamada "comidas" y en una tabla denominada "postres" los mismos datos de sus postres.

Eliminamos las tablas, si existen:

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

drop table comidas;

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

drop table postres;

Creamos las tablas:

create table comidas(

codigo tinyint identity,

nombre varchar(30),

precio decimal(4,2)

);

create table postres(

codigo tinyint identity,

nombre varchar(30),

precio decimal(4,2)

);

Ingresamos algunos registros:

insert into comidas values('ravioles',5);

insert into comidas values('tallarines',4);

insert into comidas values('milanesa',7);

insert into comidas values('cuarto de pollo',6);

insert into postres values('flan',2.5);

insert into postres values('porcion torta',3.5);

El restaurante quiere combinar los registros de ambas tablas para mostrar los distintos menúes que ofrece. Lo hacemos usando un "cross join":

select c.nombre as 'plato principal',

p.nombre as 'postre',

c.precio+p.precio as 'total'

from comidas as c

cross join postres as p;

La salida muestra cada plato combinado con cada uno de los postres, agregamos una columna que calcula el precio total de cada menú. Se obtienen 8 registros.

**Código:**

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

drop table comidas;

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

drop table postres;

create table comidas(

codigo tinyint identity,

nombre varchar(30),

precio decimal(4,2)

);

create table postres(

codigo tinyint identity,

nombre varchar(30),

precio decimal(4,2)

);

insert into comidas values('ravioles',5);

insert into comidas values('tallarines',4);

insert into comidas values('milanesa',7);

insert into comidas values('cuarto de pollo',6);

insert into postres values('flan',2.5);

insert into postres values('porcion torta',3.5);

select c.nombre as 'plato principal',

p.nombre as 'postre',

c.precio+p.precio as 'total'

from comidas as c

cross join postres as p;

**Problema:**

Una agencia matrimonial almacena la información de sus clientes de sexo femenino en una tabla llamada "mujeres" y en otra la de sus clientes de sexo masculino llamada "varones".

1- Elimine las tablas si existen y créelas:

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

drop table mujeres;

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

drop table varones;

create table mujeres(

nombre varchar(30),

domicilio varchar(30),

edad int

);

create table varones(

nombre varchar(30),

domicilio varchar(30),

edad int

);

2- Ingrese los siguientes registros:

insert into mujeres values('Maria Lopez','Colon 123',45);

insert into mujeres values('Liliana Garcia','Sucre 456',35);

insert into mujeres values('Susana Lopez','Avellaneda 98',41);

insert into varones values('Juan Torres','Sarmiento 755',44);

insert into varones values('Marcelo Oliva','San Martin 874',56);

insert into varones values('Federico Pereyra','Colon 234',38);

insert into varones values('Juan Garcia','Peru 333',50);

3- La agencia necesita la combinación de todas las personas de sexo femenino con las de sexo masculino. Use un "cross join" (12 registros)

4- Realice la misma combinación pero considerando solamente las personas mayores de 40 años (6 registros)

5- Forme las parejas pero teniendo en cuenta que no tengan una diferencia superior a 10 años (8 registros)

**Solución:**

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

drop table mujeres;

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

drop table varones;

create table mujeres(

nombre varchar(30),

domicilio varchar(30),

edad int

);

create table varones(

nombre varchar(30),

domicilio varchar(30),

edad int

);

insert into mujeres values('Maria Lopez','Colon 123',45);

insert into mujeres values('Liliana Garcia','Sucre 456',35);

insert into mujeres values('Susana Lopez','Avellaneda 98',41);

insert into varones values('Juan Torres','Sarmiento 755',44);

insert into varones values('Marcelo Oliva','San Martin 874',56);

insert into varones values('Federico Pereyra','Colon 234',38);

insert into varones values('Juan Garcia','Peru 333',50);

select m.nombre,m.edad,v.nombre,v.edad

from mujeres as m

cross join varones as v;

select m.nombre,m.edad,v.nombre,v.edad

from mujeres as m

cross join varones as v

where m.edad>40 and

v.edad>40;

select m.nombre,m.edad,v.nombre,v.edad

from mujeres as m

cross join varones as v

where m.edad-v.edad between -10 and 10;

# Autocombinación

Dijimos que es posible combinar una tabla consigo misma.

Un pequeño restaurante tiene almacenadas sus comidas en una tabla llamada "comidas" que consta de los siguientes campos:

- nombre varchar(20),

- precio decimal (4,2) y

- tipoplato char(6)-- que indica con 'plato' si es un plato principal y 'postre' si es postre.

Podemos obtener la combinación de platos empleando un "cross join" con una sola tabla:

select c1.nombre as 'plato principal',

c2.nombre as postre,

c1.precio+c2.precio as total

from comidas as c1

cross join comidas as c2;

En la consulta anterior aparecen filas duplicadas, para evitarlo debemos emplear un "where":

select c1.nombre as 'plato principal',

c2.nombre as postre,

c1.precio+c2.precio as total

from comidas as c1

cross join comidas as c2

where c1.tipoplato='plato' and

c2.tipoplato='postre';

En la consulta anterior se empleó un "where" que especifica que se combine "plato" con "postre".

En una autocombinación se combina una tabla con una copia de si misma. Para ello debemos utilizar 2 alias para la tabla. Para evitar que aparezcan filas duplicadas, debemos emplear un "where".

También se puede realizar una autocombinación con "join":

select c1.nombre as 'plato principal',

c2.nombre as postre,

c1.precio+c2.precio as total

from comidas as c1

join comidas as c2

on c1.codigo<>c2.codigo

where c1.tipoplato='plato' and

c2.tipoplato='postre';

Para que no aparezcan filas duplicadas se agrega un "where".

**Ejercicio:**

Un pequeño restaurante tiene almacenados los nombres, precios y tipoplato de sus comidas en una tabla llamada "comidas".

Eliminamos la tabla, si existe:

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

drop table comidas;

Creamos la tabla:

create table comidas(

codigo int identity,

nombre varchar(30),

precio decimal(4,2),

tipoplato char(6),-- 'plato'=plato principal', 'postre'=postre

primary key(codigo)

);

Ingresamos algunos registros:

insert into comidas values('ravioles',5,'plato');

insert into comidas values('tallarines',4,'plato');

insert into comidas values('milanesa',7,'plato');

insert into comidas values('cuarto de pollo',6,'plato');

insert into comidas values('flan',2.5,'postre');

insert into comidas values('porcion torta',3.5,'postre');

Realizamos un "cross join":

select c1.nombre as 'plato principal',

c2.nombre as postre,

c1.precio+c2.precio as total

from comidas as c1

cross join comidas as c2;

Note que aparecen filas duplicadas, por ejemplo, "ravioles" se combina con "ravioles" y la combinación "ravioles- flan" se repite como "flan- ravioles". Debemos especificar que combine el tipoplato "plato" con "postre":

select c1.nombre as 'plato principal',

c2.nombre as postre,

c1.precio+c2.precio as total

from comidas as c1

cross join comidas as c2

where c1.tipoplato='plato' and

c2.tipoplato='postre';

La salida muestra cada plato combinado con cada postre, y una columna extra que calcula el total del menú.

También se puede realizar una autocombinación con "join":

select c1.nombre as 'plato principal',

c2.nombre as postre,

c1.precio+c2.precio as total

from comidas as c1

join comidas as c2

on c1.codigo<>c2.codigo

where c1.tipoplato='plato' and

c2.tipoplato='postre';

Para que no aparezcan filas duplicadas se agrega un "where".

**Código:**

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

drop table comidas;

create table comidas(

codigo int identity,

nombre varchar(30),

precio decimal(4,2),

tipoplato char(6),-- 'plato'=plato principal', 'postre'=postre

primary key(codigo)

);

insert into comidas values('ravioles',5,'plato');

insert into comidas values('tallarines',4,'plato');

insert into comidas values('milanesa',7,'plato');

insert into comidas values('cuarto de pollo',6,'plato');

insert into comidas values('flan',2.5,'postre');

insert into comidas values('porcion torta',3.5,'postre');

select c1.nombre as 'plato principal',

c2.nombre as postre,

c1.precio+c2.precio as total

from comidas as c1

cross join comidas as c2;

select c1.nombre as 'plato principal',

c2.nombre as postre,

c1.precio+c2.precio as total

from comidas as c1

cross join comidas as c2

where c1.tipoplato='plato' and

c2.tipoplato='postre';

select c1.nombre as 'plato principal',

c2.nombre as postre,

c1.precio+c2.precio as total

from comidas as c1

join comidas as c2

on c1.codigo<>c2.codigo

where c1.tipoplato='plato' and

c2.tipoplato='postre';

**Problema:**

Una agencia matrimonial almacena la información de sus clientes en una tabla llamada "clientes".

1- Elimine la tabla si existe y créela:

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

drop table clientes;

create table clientes(

nombre varchar(30),

sexo char(1),--'f'=femenino, 'm'=masculino

edad int,

domicilio varchar(30)

);

2- Ingrese los siguientes registros:

insert into clientes values('Maria Lopez','f',45,'Colon 123');

insert into clientes values('Liliana Garcia','f',35,'Sucre 456');

insert into clientes values('Susana Lopez','f',41,'Avellaneda 98');

insert into clientes values('Juan Torres','m',44,'Sarmiento 755');

insert into clientes values('Marcelo Oliva','m',56,'San Martin 874');

insert into clientes values('Federico Pereyra','m',38,'Colon 234');

insert into clientes values('Juan Garcia','m',50,'Peru 333');

3- La agencia necesita la combinación de todas las personas de sexo femenino con las de sexo masculino. Use un "cross join" (12 registros)

4- Obtenga la misma salida enterior pero realizando un "join".

5- Realice la misma autocombinación que el punto 3 pero agregue la condición que las parejas no tengan una diferencia superior a 5 años (5 registros)

**Solución:**

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

drop table clientes;

create table clientes(

nombre varchar(30),

sexo char(1),--'f'=femenino, 'm'=masculino

edad int,

domicilio varchar(30)

);

insert into clientes values('Maria Lopez','f',45,'Colon 123');

insert into clientes values('Liliana Garcia','f',35,'Sucre 456');

insert into clientes values('Susana Lopez','f',41,'Avellaneda 98');

insert into clientes values('Juan Torres','m',44,'Sarmiento 755');

insert into clientes values('Marcelo Oliva','m',56,'San Martin 874');

insert into clientes values('Federico Pereyra','m',38,'Colon 234');

insert into clientes values('Juan Garcia','m',50,'Peru 333');

select cm.nombre,cm.edad,cv.nombre,cv.edad

from clientes as cm

cross join clientes cv

where cm.sexo='f' and cv.sexo='m';

select cm.nombre,cm.edad,cv.nombre,cv.edad

from clientes as cm

join clientes cv

on cm.nombre<>cv.nombre

where cm.sexo='f' and cv.sexo='m';

select cm.nombre,cm.edad,cv.nombre,cv.edad

from clientes as cm

cross join clientes cv

where cm.sexo='f' and cv.sexo='m' and

cm.edad-cv.edad between -5 and 5;

# Combinaciones y funciones de agrupamiento

Podemos usar "group by" y las funciones de agrupamiento con combinaciones de tablas.

Para ver la cantidad de libros de cada editorial consultando la tabla "libros" y "editoriales", tipeamos:

select nombre as editorial,

count(\*) as cantidad

from editoriales as e

join libros as l

on codigoeditorial=e.codigo

group by e.nombre;

Note que las editoriales que no tienen libros no aparecen en la salida porque empleamos un "join".

Empleamos otra función de agrupamiento con "left join". Para conocer el mayor precio de los libros de cada editorial usamos la función "max()", hacemos un "left join" y agrupamos por nombre de la editorial:

select nombre as editorial,

max(precio) as 'mayor precio'

from editoriales as e

left join libros as l

on codigoeditorial=e.codigo

group by nombre;

En la sentencia anterior, mostrará, para la editorial de la cual no haya libros, el valor "null" en la columna calculada.

**Ejercicio:**

Una librería almacena la información de sus libros para la venta en dos tablas, "libros" y "editoriales".

Eliminamos ambas 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 libros(

codigo int identity,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

Ingresamos algunos registros en ambas tablas:

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

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

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',3,50);

insert into libros values('Uno','Richard Bach',3,15);

insert into libros values('Java en 10 minutos',default,4,45);

Contamos la cantidad de libros de cada editorial consultando ambas tablas:

select nombre as editorial,

count(\*) as cantidad

from editoriales as e

join libros as l

on codigoeditorial=e.codigo

group by e.nombre;

Buscamos el libro más costoso de cada editorial con un "left join":

select nombre as editorial,

max(precio) as 'mayor precio'

from editoriales as e

left join libros as l

on codigoeditorial=e.codigo

group by nombre;

**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),

autor varchar(30),

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

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

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',3,50);

insert into libros values('Uno','Richard Bach',3,15);

insert into libros values('Java en 10 minutos',default,4,45);

select nombre as editorial,

count(\*) as cantidad

from editoriales as e

join libros as l

on codigoeditorial=e.codigo

group by e.nombre;

select nombre as editorial,

max(precio) as 'mayor precio'

from editoriales as e

left join libros as l

on codigoeditorial=e.codigo

group by nombre;

**Problema:**

Un comercio que tiene un stand en una feria registra en una tabla llamada "visitantes" algunos datos

de las personas que visitan o compran en su stand para luego enviarle publicidad de sus productos y

en otra tabla llamada "ciudades" los nombres de las ciudades.

1- Elimine las tablas si existen:

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

drop table visitantes;

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

drop table ciudades;

2- Cree las tablas:

create table visitantes(

nombre varchar(30),

edad tinyint,

sexo char(1) default 'f',

domicilio varchar(30),

codigociudad tinyint not null,

mail varchar(30),

montocompra decimal (6,2)

);

create table ciudades(

codigo tinyint identity,

nombre varchar(20)

);

3- Ingrese algunos registros:

insert into ciudades values('Cordoba');

insert into ciudades values('Carlos Paz');

insert into ciudades values('La Falda');

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

insert into visitantes values

('Susana Molina', 35,'f','Colon 123', 1, null,59.80);

insert into visitantes values

('Marcos Torres', 29,'m','Sucre 56', 1, 'marcostorres@hotmail.com',150.50);

insert into visitantes values

('Mariana Juarez', 45,'f','San Martin 111',2,null,23.90);

insert into visitantes values

('Fabian Perez',36,'m','Avellaneda 213',3,'fabianperez@xaxamail.com',0);

insert into visitantes values

('Alejandra Garcia',28,'f',null,2,null,280.50);

insert into visitantes values

('Gaston Perez',29,'m',null,5,'gastonperez1@gmail.com',95.40);

insert into visitantes values

('Mariana Juarez',33,'f',null,2,null,90);

4- Cuente la cantidad de visitas por ciudad mostrando el nombre de la ciudad (3 filas)

5- Muestre el promedio de gastos de las visitas agrupados por ciudad y sexo (4 filas)

6- Muestre la cantidad de visitantes con mail, agrupados por ciudad (3 filas)

7- Obtenga el monto de compra más alto de cada ciudad (3 filas)

**Solución:**

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

drop table visitantes;

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

drop table ciudades;

create table visitantes(

nombre varchar(30),

edad tinyint,

sexo char(1) default 'f',

domicilio varchar(30),

codigociudad tinyint not null,

mail varchar(30),

montocompra decimal (6,2)

);

create table ciudades(

codigo tinyint identity,

nombre varchar(20)

);

insert into ciudades values('Cordoba');

insert into ciudades values('Carlos Paz');

insert into ciudades values('La Falda');

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

insert into visitantes values

('Susana Molina', 35,'f','Colon 123', 1, null,59.80);

insert into visitantes values

('Marcos Torres', 29,'m','Sucre 56', 1, 'marcostorres@hotmail.com',150.50);

insert into visitantes values

('Mariana Juarez', 45,'f','San Martin 111',2,null,23.90);

insert into visitantes values

('Fabian Perez',36,'m','Avellaneda 213',3,'fabianperez@xaxamail.com',0);

insert into visitantes values

('Alejandra Garcia',28,'f',null,2,null,280.50);

insert into visitantes values

('Gaston Perez',29,'m',null,5,'gastonperez1@gmail.com',95.40);

insert into visitantes values

('Mariana Juarez',33,'f',null,2,null,90);

select c.nombre,

count(\*) as cantidad

from ciudades as c

join visitantes as v

on codigociudad=c.codigo

group by c.nombre;

select c.nombre,sexo,

avg(montocompra) as 'promedio de compra'

from ciudades as c

join visitantes as v

on codigociudad=c.codigo

group by c.nombre,sexo;

select c.nombre,

count(mail) as 'tienen mail'

from ciudades as c

join visitantes as v

on codigociudad=c.codigo

group by c.nombre;

select c.nombre,

max(montocompra)

from visitantes as v

join ciudades as c

on codigociudad=c.codigo

group by c.nombre;

# Combinación de más de dos tablas

Podemos hacer un "join" con más de dos tablas.

Cada join combina 2 tablas. Se pueden emplear varios join para enlazar varias tablas. Cada resultado de un join es una tabla que puede combinarse con otro join.

La librería almacena los datos de sus libros en tres tablas: libros, editoriales y autores.

En la tabla "libros" un campo "codigoautor" hace referencia al autor y un campo "codigoeditorial" referencia la editorial.

Para recuperar todos los datos de los libros empleamos la siguiente consulta:

select titulo,a.nombre,e.nombre

from autores as a

join libros as l

on codigoautor=a.codigo

join editoriales as e on codigoeditorial=e.codigo;

Analicemos la consulta anterior. Indicamos el nombre de la tabla luego del "from" ("autores"), combinamos esa tabla con la tabla "libros" especificando con "on" el campo por el cual se combinarán; luego debemos hacer coincidir los valores para el enlace con la tabla "editoriales" enlazándolas por los campos correspondientes. Utilizamos alias para una sentencia más sencilla y comprensible.

Note que especificamos a qué tabla pertenecen los campos cuyo nombre se repiten en las tablas, esto es necesario para evitar confusiones y ambiguedades al momento de referenciar un campo.

Note que no aparecen los libros cuyo código de autor no se encuentra en "autores" y cuya editorial no existe en "editoriales", esto es porque realizamos una combinación interna.

Podemos combinar varios tipos de join en una misma sentencia:

select titulo,a.nombre,e.nombre

from autores as a

right join libros as l

on codigoautor=a.codigo

left join editoriales as e on codigoeditorial=e.codigo;

En la consulta anterior solicitamos el título, autor y editorial de todos los libros que encuentren o no coincidencia con "autores" ("right join") y a ese resultado lo combinamos con "editoriales", encuentren o no coincidencia.

Es posible realizar varias combinaciones para obtener información de varias tablas. Las tablas deben tener claves externas relacionadas con las tablas a combinar.

En consultas en las cuales empleamos varios "join" es importante tener en cuenta el orden de las tablas y los tipos de "join"; recuerde que la tabla resultado del primer join es la que se combina con el segundo join, no la segunda tabla nombrada. En el ejemplo anterior, el "left join" no se realiza entre las tablas "libros" y "editoriales" sino entre el resultado del "right join" y la tabla "editoriales".

**Ejercicio:**

Una librería almacena la información de sus libros para la venta en tres tablas, "libros", "autores" y "editoriales".

Eliminamos las tablas, si existen y las creamos:

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

drop table libros;

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

drop table autores;

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

drop table editoriales;

create table libros(

codigo int identity,

titulo varchar(40),

codigoautor int not null,

codigoeditorial tinyint not null,

precio decimal(5,2),

primary key(codigo)

);

create table autores(

codigo int identity,

nombre varchar(20),

primary key (codigo)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

Ingresamos algunos registros:

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

insert into editoriales values('Plaza');

insert into autores values ('Richard Bach');

insert into autores values ('Borges');

insert into autores values ('Jose Hernandez');

insert into autores values ('Mario Molina');

insert into autores values ('Paenza');

insert into libros values('El aleph',2,2,20);

insert into libros values('Martin Fierro',3,1,30);

insert into libros values('Aprenda PHP',4,3,50);

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

insert into libros values('Java en 10 minutos',0,3,45);

insert into libros values('Matematica estas ahi',0,0,15);

insert into libros values('Java de la A a la Z',4,0,50);

Recuperamos todos los datos de los libros consultando las tres tablas:

select titulo,a.nombre,e.nombre,precio

from autores as a

join libros as l

on codigoautor=a.codigo

join editoriales as e

on codigoeditorial=e.codigo;

Los libros cuyo código de autor no se encuentra en "autores" (caso de "Java en 10 minutos" y "Matematica estas ahi") y cuya editorial no existe en "editoriales" (caso de "Matematica estas ahi" y "Java de la A a la Z"), no aparecen porque realizamos una combinación interna.

Podemos combinar varios tipos de join en una misma sentencia:

select titulo,a.nombre,e.nombre,precio

from autores as a

right join libros as l

on codigoautor=a.codigo

left join editoriales as e

on codigoeditorial=e.codigo;

En la consulta anterior solicitamos el título, autor y editorial de todos los libros que encuentren o no coincidencia con "autores" ("right join") y a ese resultado lo combinamos con "editoriales", encuentren o no coincidencia.

**Código:**

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

drop table libros;

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

drop table autores;

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

drop table editoriales;

create table libros(

codigo int identity,

titulo varchar(40),

codigoautor int not null,

codigoeditorial tinyint not null,

precio decimal(5,2),

primary key(codigo)

);

create table autores(

codigo int identity,

nombre varchar(20),

primary key (codigo)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

insert into editoriales values('Plaza');

insert into autores values ('Richard Bach');

insert into autores values ('Borges');

insert into autores values ('Jose Hernandez');

insert into autores values ('Mario Molina');

insert into autores values ('Paenza');

insert into libros values('El aleph',2,2,20);

insert into libros values('Martin Fierro',3,1,30);

insert into libros values('Aprenda PHP',4,3,50);

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

insert into libros values('Java en 10 minutos',0,3,45);

insert into libros values('Matematica estas ahi',0,0,15);

insert into libros values('Java de la A a la Z',4,0,50);

select titulo,a.nombre,e.nombre,precio

from autores as a

join libros as l

on codigoautor=a.codigo

join editoriales as e

on codigoeditorial=e.codigo;

select titulo,a.nombre,e.nombre,precio

from autores as a

right join libros as l

on codigoautor=a.codigo

left join editoriales as e

on codigoeditorial=e.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('socios') is not null

drop table socios;

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

drop table deportes;

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

drop table inscritos;

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,

anio char(4),

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

primary key(documento,codigodeporte,anio)

);

3- Ingrese algunos registros en "socios":

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');

4- Ingrese algunos registros en "deportes":

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');

5- Inscriba a varios socios en el mismo deporte en el mismo año:

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

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

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

6- Inscriba a un mismo socio en el mismo deporte en distintos años:

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

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

7- Inscriba a un mismo socio en distintos deportes el mismo año:

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

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

8- Ingrese una inscripción con un código de deporte inexistente y un documento de socio que no exista en "socios":

insert into inscritos values ('26666666',0,'2006','s');

9- Muestre el nombre del socio, el nombre del deporte en que se inscribió y el año empleando diferentes tipos de join.

10- Muestre todos los datos de las inscripciones (excepto los códigos) incluyendo aquellas inscripciones cuyo código de deporte no existe en "deportes" y cuyo documento de socio no se encuentra en "socios".

11- Muestre todas las inscripciones del socio con documento "22222222".

**Solución:**

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

drop table socios;

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

drop table deportes;

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

drop table inscritos;

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,

anio char(4),

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

primary key(documento,codigodeporte,anio)

);

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');

insert into inscritos values ('26666666',0,'2006','s');

select s.nombre,d.nombre,anio

from deportes as d

right join inscritos as i

on codigodeporte=d.codigo

left join socios as s

on i.documento=s.documento;

select s.nombre,d.nombre,anio,matricula

from deportes as d

full join inscritos as i

on codigodeporte=d.codigo

full join socios as s

on s.documento=i.documento;

select s.nombre,d.nombre,anio,matricula

from deportes as d

join inscritos as i

on codigodeporte=d.codigo

join socios as s

on s.documento=i.documento

where s.documento='22222222';

# Combinaciones con update y delete

Las combinaciones no sólo se utilizan con la sentencia "select", también podemos emplearlas con "update" y "delete".

Podemos emplear "update" o "delete" con "join" para actualizar o eliminar registros de una tabla consultando otras tablas.

En el siguiente ejemplo aumentamos en un 10% los precios de los libros de cierta editorial, necesitamos un "join" para localizar los registros de la editorial "Planeta" en la tabla "libros":

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

from libros

join editoriales as e

on codigoeditorial=e.codigo

where nombre='Planeta';

Eliminamos todos los libros de editorial "Emece":

delete libros

from libros

join editoriales

on codigoeditorial = editoriales.codigo

where editoriales.nombre='Emece';

**Ejercicio:**

Una librería almacena la información de sus libros para la venta en dos tablas, "libros" y "editoriales".

Eliminamos ambas tablas, si existen:

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

drop table libros;

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

drop table editoriales;

Creamos las tablas:

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30) default 'Desconocido',

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

Ingresamos algunos registros en ambas tablas:

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

insert into libros values('El aleph','Borges',2,20);

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',3,50);

insert into libros values('Java en 10 minutos',default,3,45);

Aumentamos en un 10% los precios de los libros de editorial "Planeta":

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

from libros

join editoriales as e

on codigoeditorial=e.codigo

where nombre='Planeta';

Veamos el resultado:

select titulo,autor,e.nombre,precio

from libros as l

join editoriales as e

on codigoeditorial=e.codigo;

Eliminamos todos los libros de editorial "Emece":

delete libros

from libros

join editoriales

on codigoeditorial = editoriales.codigo

where editoriales.nombre='Emece';

Veamos si se eliminaron:

select titulo,autor,e.nombre,precio

from libros as l

join editoriales as e

on codigoeditorial=e.codigo;

**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),

autor varchar(30) default 'Desconocido',

codigoeditorial tinyint not null,

precio decimal(5,2)

);

create table editoriales(

codigo tinyint identity,

nombre varchar(20),

primary key (codigo)

);

insert into editoriales values('Planeta');

insert into editoriales values('Emece');

insert into editoriales values('Siglo XXI');

insert into libros values('El aleph','Borges',2,20);

insert into libros values('Martin Fierro','Jose Hernandez',1,30);

insert into libros values('Aprenda PHP','Mario Molina',3,50);

insert into libros values('Java en 10 minutos',default,3,45);

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

from libros

join editoriales as e

on codigoeditorial=e.codigo

where nombre='Planeta';

select titulo,autor,e.nombre,precio

from libros as l

join editoriales as e

on codigoeditorial=e.codigo;

delete libros

from libros

join editoriales

on codigoeditorial = editoriales.codigo

where editoriales.nombre='Emece';

select titulo,autor,e.nombre,precio

from libros as l

join editoriales as e

on codigoeditorial=e.codigo;

# Clave foránea

UN CAMPO QUE NO ES CLAVE PRIMARIA EN UNA TABLA Y SIRVE PARA ENLAZAR SUS VALORES CON OTRA TABLA EN LA CUAL ES CLAVE PRIMARIA SE DENOMINA CLAVE FORÁNEA, EXTERNA O AJENA.

En el ejemplo de la librería en que utilizamos las tablas "libros" y "editoriales" con estos campos:

libros: codigo (clave primaria), titulo, autor, codigoeditorial, precio y

editoriales: codigo (clave primaria), nombre.

El campo "codigoeditorial" de "libros" es una clave foránea, se emplea para enlazar la tabla "libros" con "editoriales" y es clave primaria en "editoriales" con el nombre "codigo".

Las claves foráneas y las claves primarias deben ser del mismo tipo para poder enlazarse. Si modificamos una, debemos modificar la otra para que los valores se correspondan.

Cuando alteramos una tabla, debemos tener cuidado con las claves foráneas. Si modificamos el tipo, longitud o atributos de una clave foránea, ésta puede quedar inhabilitada para hacer los enlaces.

Entonces, una clave foránea es un campo (o varios) empleados para enlazar datos de 2 tablas, para establecer un "join" con otra tabla en la cual es clave primaria.

# Restricciones (foreign key)

Hemos visto que una de las alternativas que SQL Server ofrece para asegurar la integridad de datos es el uso de restricciones (constraints). Aprendimos que las restricciones se establecen en tablas y campos asegurando que los datos sean válidos y que las relaciones entre las tablas se mantengan; vimos que existen distintos tipos de restricciones:

1) de los campos: default y check

2) de la tabla: primary key y unique.

3) referencial: foreign key, la analizaremos ahora.

Con la restricción "foreign key" se define un campo (o varios) cuyos valores coinciden con la clave primaria de la misma tabla o de otra, es decir, se define una referencia a un campo con una restricción "primary key" o "unique" de la misma tabla o de otra.

LA INTEGRIDAD REFERENCIAL ASEGURA QUE SE MANTENGAN LAS REFERENCIAS ENTRE LAS CLAVES PRIMARIAS Y LAS EXTERNAS. POR EJEMPLO, CONTROLA QUE SI SE AGREGA UN CÓDIGO DE EDITORIAL EN LA TABLA "LIBROS", TAL CÓDIGO EXISTA EN LA TABLA "EDITORIALES".

TAMBIÉN CONTROLA QUE NO PUEDA ELIMINARSE UN REGISTRO DE UNA TABLA NI MODIFICAR LA CLAVE PRIMARIA SI UNA CLAVE EXTERNA HACE REFERENCIA AL REGISTRO. POR EJEMPLO, QUE NO SE PUEDA ELIMINAR O MODIFICAR UN CÓDIGO DE "EDITORIALES" SI EXISTEN LIBROS CON DICHO CÓDIGO.

La siguiente es la sintaxis parcial general para agregar una restricción "foreign key":

alter table NOMBRETABLA1

add constraint NOMBRERESTRICCION

foreign key (CAMPOCLAVEFORANEA)

references NOMBRETABLA2 (CAMPOCLAVEPRIMARIA);

Analicémosla:

- NOMBRETABLA1 referencia el nombre de la tabla a la cual le aplicamos la restricción,

- NOMBRERESTRICCION es el nombre que le damos a la misma,

- luego de "foreign key", entre paréntesis se coloca el campo de la tabla a la que le aplicamos la restricción que será establecida como clave foránea,

- luego de "references" indicamos el nombre de la tabla referenciada y el campo que es clave primaria en la misma, a la cual hace referencia la clave foránea. La tabla referenciada debe tener definida una restricción "primary key" o "unique"; si no la tiene, aparece un mensaje de error.

Para agregar una restricción "foreign key" al campo "codigoeditorial" de "libros", tipeamos:

alter table libros

add constraint FK\_libros\_codigoeditorial

foreign key (codigoeditorial)

references editoriales(codigo);

En el ejemplo implementamos una restricción "foreign key" para asegurarnos que el código de la editorial de la de la tabla "libros" ("codigoeditorial") esté asociada con un código válido en la tabla "editoriales" ("codigo").

Cuando agregamos cualquier restricción a una tabla que contiene información, SQL Server controla los datos existentes para confirmar que cumplen con la restricción, si no los cumple, la restricción no se aplica y aparece un mensaje de error. Por ejemplo, si intentamos agregar una restricción "foreign key" a la tabla "libros" y existe un libro con un valor de código para editorial que no existe en la tabla "editoriales", la restricción no se agrega.

Actúa en inserciones. Si intentamos ingresar un registro (un libro) con un valor de clave foránea (codigoeditorial) que no existe en la tabla referenciada (editoriales), SQL server muestra un mensaje de error. Si al ingresar un registro (un libro), no colocamos el valor para el campo clave foránea (codigoeditorial), almacenará "null", porque esta restricción permite valores nulos (a menos que se haya especificado lo contrario al definir el campo).

Actúa en eliminaciones y actualizaciones. Si intentamos eliminar un registro o modificar un valor de clave primaria de una tabla si una clave foránea hace referencia a dicho registro, SQL Server no lo permite (excepto si se permite la acción en cascada, tema que veremos posteriormente). Por ejemplo, si intentamos eliminar una editorial a la que se hace referencia en "libros", aparece un mensaje de error.

Esta restricción (a diferencia de "primary key" y "unique") no crea índice automaticamente.

La cantidad y tipo de datos de los campos especificados luego de "foreign key" DEBEN coincidir con la cantidad y tipo de datos de los campos de la cláusula "references".

Esta restricción se puede definir dentro de la misma tabla (lo veremos más adelante) o entre distintas tablas.

Una tabla puede tener varias restricciones "foreign key".

No se puede eliminar una tabla referenciada en una restricción "foreign key", aparece un mensaje de error.

Una restriccion "foreign key" no puede modificarse, debe eliminarse y volverse a crear.

Para ver información acerca de esta restricción podemos ejecutar el procedimiento almacenado "sp\_helpconstraint" junto al nombre de la tabla. Nos muestra el tipo, nombre, la opción para eliminaciones y actualizaciones, el estado (temas que veremos más adelante), el nombre del campo y la tabla y campo que referencia.

También informa si la tabla es referenciada por una clave foránea.

**Ejercicio:**

Una librería almacena la información de sus libros para la venta en dos tablas, "libros" y "editoriales".

Eliminamos ambas tablas, si existen:

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

drop table libros;

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

drop table editoriales;

Creamos las tablas:

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint

);

create table editoriales(

codigo tinyint,

nombre varchar(20),

primary key (codigo)

);

Ingresamos algunos registros en ambas tablas:

insert into editoriales values(1,'Emece');

insert into editoriales values(2,'Planeta');

insert into editoriales values(3,'Siglo XXI');

insert into libros values('El aleph','Borges',1);

insert into libros values('Martin Fierro','Jose Hernandez',2);

insert into libros values('Aprenda PHP','Mario Molina',2);

Agregamos una restricción "foreign key" a la tabla "libros":

alter table libros

add constraint FK\_libros\_codigoeditorial

foreign key (codigoeditorial)

references editoriales(codigo);

Ingresamos un libro sin especificar valores:

insert into libros default values;

Note que en "codigoeditorial" almacenó "null", porque esta restricción permite valores nulos (a menos que se haya especificado lo contrario al definir el campo).

Veamos las restricciones de "libros":

sp\_helpconstraint libros;

Aparece la restricción "foreign key".

Veamos las restricciones de "editoriales":

sp\_helpconstraint editoriales;

Aparece la restricción "primary key" y nos informa que la tabla es rerenciada por una "foreign key" de la tabla "libros" llamada "FK\_libros\_codigoeditorial".

**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),

autor varchar(30),

codigoeditorial tinyint

);

create table editoriales(

codigo tinyint,

nombre varchar(20),

primary key (codigo)

);

insert into editoriales values(1,'Emece');

insert into editoriales values(2,'Planeta');

insert into editoriales values(3,'Siglo XXI');

insert into libros values('El aleph','Borges',1);

insert into libros values('Martin Fierro','Jose Hernandez',2);

insert into libros values('Aprenda PHP','Mario Molina',2);

alter table libros

add constraint FK\_libros\_codigoeditorial

foreign key (codigoeditorial)

references editoriales(codigo);

insert into libros default values;

sp\_helpconstraint libros;

sp\_helpconstraint editoriales;

**Problema:**

Una empresa tiene registrados sus clientes en una tabla llamada "clientes", también tiene una tabla

"provincias" donde registra los nombres de las provincias.

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

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

drop table clientes;

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

drop table provincias;

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

codigoprovincia tinyint

);

create table provincias(

codigo tinyint not null,

nombre varchar(20)

);

En este ejemplo, el campo "codigoprovincia" de "clientes" es una clave foránea, se emplea para enlazar la tabla "clientes" con "provincias".

2- Intente agregar una restricción "foreign key" a la tabla "clientes" que haga referencia al campo

"codigo" de "provincias", no se puede porque "provincias" no tiene restricción "primary key" ni "unique".

3- Establezca una restricción "primary key" al campo "codigo" de "provincias".

4- Ingrese algunos registros para ambas tablas:

insert into provincias values(1,'Cordoba');

insert into provincias values(2,'Santa Fe');

insert into provincias values(3,'Misiones');

insert into provincias values(4,'Rio Negro');

insert into clientes values('Perez Juan','San Martin 123','Carlos Paz',1);

insert into clientes values('Moreno Marcos','Colon 234','Rosario',2);

insert into clientes values('Acosta Ana','Avellaneda 333','Posadas',3);

insert into clientes values('Luisa Lopez','Juarez 555','La Plata',6);

5- Intente agregar la restricción "foreign key" del punto 2 a la tabla "clientes". No se puede porque hay un registro en "clientes" cuyo valor de "codigoprovincia" no existe en "provincias".

6- Elimine el registro de "clientes" que no cumple con la restricción y establezca la restricción nuevamente.

7- Intente agregar un cliente con un código de provincia inexistente en "provincias".

No se puede.

8- Intente eliminar el registro con código 3, de "provincias".

No se puede porque hay registros en "clientes" al cual hace referencia.

9- Elimine el registro con código "4" de "provincias".

Se permite porque en "clientes" ningún registro hace referencia a él.

10- Intente modificar el registro con código 1, de "provincias".

No se puede porque hay registros en "clientes" al cual hace referencia.

11- Vea las restricciones de "clientes". Aparece la restricción "foreign key".

12- Vea las restricciones de "provincias". Aparece la restricción "primary key" y nos informa que la tabla es rerenciada por una "foreign key" de la tabla "clientes" llamada "FK\_clientes\_codigoprovincia".

**Solución:**

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

drop table clientes;

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

drop table provincias;

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

codigoprovincia tinyint

);

create table provincias(

codigo tinyint not null,

nombre varchar(20)

);

alter table clientes

add constraint FK\_clientes\_codigoprovincia

foreign key (codigoprovincia)

references provincias(codigo);

alter table provincias

add constraint PK\_provincias\_codigo

primary key (codigo);

insert into provincias values(1,'Cordoba');

insert into provincias values(2,'Santa Fe');

insert into provincias values(3,'Misiones');

insert into provincias values(4,'Rio Negro');

insert into clientes values('Perez Juan','San Martin 123','Carlos Paz',1);

insert into clientes values('Moreno Marcos','Colon 234','Rosario',2);

insert into clientes values('Acosta Ana','Avellaneda 333','Posadas',3);

insert into clientes values('Luisa Lopez','Juarez 555','La Plata',6);

alter table clientes

add constraint FK\_clientes\_codigoprovincia

foreign key (codigoprovincia)

references provincias(codigo);

delete from clientes where codigoprovincia=6;

alter table clientes

add constraint FK\_clientes\_codigoprovincia

foreign key (codigoprovincia)

references provincias(codigo);

insert into clientes values('Garcia Marcos','Colon 877','Lules',9);

delete from provincias where codigo=3;

delete from provincias where codigo=4;

update provincias set codigo=7 where codigo=1;

sp\_helpconstraint clientes;

sp\_helpconstraint provincias;

# Restricciones foreign key en la misma tabla

La restricción "foreign key", que define una referencia a un campo con una restricción "primary key" o "unique" se puede definir entre distintas tablas (como hemos aprendido) o dentro de la misma tabla.

Veamos un ejemplo en el cual definimos esta restricción dentro de la misma tabla.

Una mutua almacena los datos de sus afiliados en una tabla llamada "afiliados". Algunos afiliados inscriben a sus familiares. La tabla contiene un campo que hace referencia al afiliado que lo incorporó a la mutua, del cual dependen.

La estructura de la tabla es la siguiente:

create table afiliados(

numero int identity not null,

documento char(8) not null,

nombre varchar(30),

afiliadotitular int,

primary key (documento),

unique (numero)

);

En caso que un afiliado no haya sido incorporado a la mutua por otro afiliado, el campo "afiliadotitular" almacenará "null".

Establecemos una restricción "foreign key" para asegurarnos que el número de afiliado que se ingrese en el campo "afiliadotitular" exista en la tabla "afiliados":

alter table afiliados

add constraint FK\_afiliados\_afiliadotitular

foreign key (afiliadotitular)

references afiliados (numero);

La sintaxis es la misma, excepto que la tabla se autoreferencia.

Luego de aplicar esta restricción, cada vez que se ingrese un valor en el campo "afiliadotitular", SQL Server controlará que dicho número exista en la tabla, si no existe, mostrará un mensaje de error.

Si intentamos eliminar un afiliado que es titular de otros afiliados, no se podrá hacer, a menos que se haya especificado la acción en cascada (próximo tema).

**Ejercicio:**

Una mutua almacena los datos de sus afiliados en una tabla llamada "afiliados". Algunos afiliados inscriben a sus familiares. La tabla contiene un campo que hace referencia al afiliado que lo incorporó a la mutua, del cual dependen.

Eliminamos la tabla "afiliados" si existe y la creamos:

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

drop table afiliados;

create table afiliados(

numero int identity not null,

documento char(8) not null,

nombre varchar(30),

afiliadotitular int,

primary key (documento),

unique (numero)

);

En caso que un afiliado no haya sido incorporado a la mutua por otro afiliado, el campo "afiliadotitular" almacenará "null".

Establecemos una restricción "foreign key" para asegurarnos que el número de afiliado que se ingrese en el campo "afiliadotitular" exista en la tabla "afiliados":

alter table afiliados

add constraint FK\_afiliados\_afiliadotitular

foreign key (afiliadotitular)

references afiliados (numero);

Ingresamos algunos registros:

insert into afiliados values('22222222','Perez Juan',null);

insert into afiliados values('23333333','Garcia Maria',null);

insert into afiliados values('24444444','Lopez Susana',null);

insert into afiliados values('30000000','Perez Marcela',1);

insert into afiliados values('31111111','Morales Luis',1);

insert into afiliados values('32222222','Garcia Maria',2);

Podemos eliminar un afiliado, siempre que no haya otro afiliado que haga referencia a él en "afiliadotitular", es decir, si el "numero" del afiliado está presente en algún registro en el campo "afiliadotitular":

delete from afiliados where numero=5;

Veamos la información referente a "afiliados":

sp\_helpconstraint afiliados;

**Código:**

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

drop table afiliados;

create table afiliados(

numero int identity not null,

documento char(8) not null,

nombre varchar(30),

afiliadotitular int,

primary key (documento),

unique (numero)

);

alter table afiliados

add constraint FK\_afiliados\_afiliadotitular

foreign key (afiliadotitular)

references afiliados (numero);

insert into afiliados values('22222222','Perez Juan',null);

insert into afiliados values('23333333','Garcia Maria',null);

insert into afiliados values('24444444','Lopez Susana',null);

insert into afiliados values('30000000','Perez Marcela',1);

insert into afiliados values('31111111','Morales Luis',1);

insert into afiliados values('32222222','Garcia Maria',2);

delete from afiliados where numero=5;

sp\_helpconstraint afiliados;

**Problema:**

Una empresa registra los datos de sus clientes en una tabla llamada "clientes". Dicha tabla contiene un campo que hace referencia al cliente que lo recomendó denominado "referenciadopor". Si un cliente no ha sido referenciado por ningún otro cliente, tal campo almacena "null".

1- Elimine la tabla si existe y créela:

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

drop table clientes;

create table clientes(

codigo int not null,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

referenciadopor int,

primary key(codigo)

);

2- Ingresamos algunos registros:

insert into clientes values (50,'Juan Perez','Sucre 123','Cordoba',null);

insert into clientes values(90,'Marta Juarez','Colon 345','Carlos Paz',null);

insert into clientes values(110,'Fabian Torres','San Martin 987','Cordoba',50);

insert into clientes values(125,'Susana Garcia','Colon 122','Carlos Paz',90);

insert into clientes values(140,'Ana Herrero','Colon 890','Carlos Paz',9);

3- Intente agregar una restricción "foreign key" para evitar que en el campo "referenciadopor" se ingrese un valor de código de cliente que no exista.

No se permite porque existe un registro que no cumple con la restricción que se intenta establecer.

4- Cambie el valor inválido de "referenciadopor" del registro que viola la restricción por uno válido.

5- Agregue la restricción "foreign key" que intentó agregar en el punto 3.

6- Vea la información referente a las restricciones de la tabla "clientes".

7- Intente agregar un registro que infrinja la restricción.

No lo permite.

8- Intente modificar el código de un cliente que está referenciado en "referenciadopor".

No se puede.

9- Intente eliminar un cliente que sea referenciado por otro en "referenciadopor".

No se puede.

10- Cambie el valor de código de un cliente que no referenció a nadie.

11- Elimine un cliente que no haya referenciado a otros.

**Solución:**

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

drop table clientes;

create table clientes(

codigo int not null,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

referenciadopor int,

primary key(codigo)

);

insert into clientes values (50,'Juan Perez','Sucre 123','Cordoba',null);

insert into clientes values(90,'Marta Juarez','Colon 345','Carlos Paz',null);

insert into clientes values(110,'Fabian Torres','San Martin 987','Cordoba',50);

insert into clientes values(125,'Susana Garcia','Colon 122','Carlos Paz',90);

insert into clientes values(140,'Ana Herrero','Colon 890','Carlos Paz',9);

alter table clientes

add constraint FK\_clientes\_referenciadopor

foreign key (referenciadopor)

references clientes (codigo);

update clientes set referenciadopor=90 where referenciadopor=9;

alter table clientes

add constraint FK\_clientes\_referenciadopor

foreign key (referenciadopor)

references clientes (codigo);

sp\_helpconstraint clientes;

insert into clientes values(150,'Karina Gomez','Caseros 444','Cruz del Eje',8);

update clientes set codigo=180 where codigo=90;

delete from clientes where nombre='Marta Juarez';

update clientes set codigo=180 where codigo=125;

delete from clientes where codigo=110;

# Restricciones foreign key (acciones)

Continuamos con la restricción "foreign key".

SI INTENTAMOS ELIMINAR UN REGISTRO DE LA TABLA REFERENCIADA POR UNA RESTRICCIÓN "FOREIGN KEY" CUYO VALOR DE CLAVE PRIMARIA EXISTE REFERENCIADA EN LA TABLA QUE TIENE DICHA RESTRICCIÓN, LA ACCIÓN NO SE EJECUTA Y APARECE UN MENSAJE DE ERROR. Esto sucede porque, por defecto, para eliminaciones, la opción de la restricción "foreign key" es "no action". Lo mismo sucede si intentamos actualizar un valor de clave primaria de una tabla referenciada por una "foreign key" existente en la tabla principal.

La restricción "foreign key" tiene las cláusulas "on delete" y "on update" que son opcionales.

Estas cláusulas especifican cómo debe actuar SQL Server frente a eliminaciones y modificaciones de las tablas referenciadas en la restricción.

Las opciones para estas cláusulas son las siguientes:

- "no action": indica que si intentamos eliminar o actualizar un valor de la clave primaria de la tabla referenciada (TABLA2) que tengan referencia en la tabla principal (TABLA1), se genere un error y la acción no se realice; es la opción predeterminada.

- "cascade": indica que si eliminamos o actualizamos un valor de la clave primaria en la tabla referenciada (TABLA2), los registros coincidentes en la tabla principal (TABLA1), también se eliminen o modifiquen; es decir, si eliminamos o modificamos un valor de campo definido con una restricción "primary key" o "unique", dicho cambio se extiende al valor de clave externa de la otra tabla (integridad referencial en cascada).

La sintaxis completa para agregar esta restricción a una tabla es la siguiente:

alter table TABLA1

add constraint NOMBRERESTRICCION

foreign key (CAMPOCLAVEFORANEA)

references TABLA2(CAMPOCLAVEPRIMARIA)

on delete OPCION

on update OPCION;

Sintetizando, si al agregar una restricción foreign key:

- no se especifica acción para eliminaciones (o se especifica "no\_action"), y se intenta eliminar un registro de la tabla referenciada (editoriales) cuyo valor de clave primaria (codigo) existe en la tabla principal (libros), la acción no se realiza.

- se especifica "cascade" para eliminaciones ("on delete cascade") y elimina un registro de la tabla referenciada (editoriales) cuyo valor de clave primaria (codigo) existe en la tabla principal(libros), la eliminación de la tabla referenciada (editoriales) se realiza y se eliminan de la tabla principal (libros) todos los registros cuyo valor coincide con el registro eliminado de la tabla referenciada (editoriales).

- no se especifica acción para actualizaciones (o se especifica "no\_action"), y se intenta modificar un valor de clave primaria (codigo) de la tabla referenciada (editoriales) que existe en el campo clave foránea (codigoeditorial) de la tabla principal (libros), la acción no se realiza.

- se especifica "cascade" para actualizaciones ("on update cascade") y se modifica un valor de clave primaria (codigo) de la tabla referenciada (editoriales) que existe en la tabla principal (libros), SQL Server actualiza el registro de la tabla referenciada (editoriales) y todos los registros coincidentes en la tabla principal (libros).

Veamos un ejemplo. Definimos una restricción "foreign key" a la tabla "libros" estableciendo el campo "codigoeditorial" como clave foránea que referencia al campo "codigo" de la tabla "editoriales". La tabla "editoriales" tiene como clave primaria el campo "codigo". Especificamos la acción en cascada para las actualizaciones y eliminaciones:

alter table libros

add constraint FK\_libros\_codigoeditorial

foreign key (codigoeditorial)

references editoriales(codigo)

on update cascade

on delete cascade;

Si luego de establecer la restricción anterior, eliminamos una editorial de "editoriales" de las cuales hay libros, se elimina dicha editorial y todos los libros de tal editorial. Y si modificamos el valor de código de una editorial de "editoriales", se modifica en "editoriales" y todos los valores iguales de "codigoeditorial" de libros también se modifican.

**Ejercicio:**

Una librería almacena la información de sus libros para la venta en dos tablas, "libros" y "editoriales".

Eliminamos ambas tablas, si existen:

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

drop table libros;

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

drop table editoriales;

Creamos las tablas:

create table libros(

codigo int not null,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint,

primary key (codigo)

);

create table editoriales(

codigo tinyint not null,

nombre varchar(20),

primary key (codigo)

);

Ingresamos algunos registros en ambas tablas:

insert into editoriales values(1,'Emece');

insert into editoriales values(2,'Planeta');

insert into editoriales values(3,'Siglo XXI');

insert into libros values(1,'El aleph','Borges',1);

insert into libros values(2,'Martin Fierro','Jose Hernandez',2);

insert into libros values(3,'Aprenda PHP','Mario Molina',2);

Establecemos una restricción "foreign key" para evitar que se ingrese en "libros" un código de editorial inexistente en "editoriales" con la opción "on cascade" para actualizaciones y eliminaciones:

alter table libros

add constraint FK\_libros\_codigoeditorial

foreign key (codigoeditorial)

references editoriales(codigo)

on update cascade

on delete cascade;

Si actualizamos un valor de código de "editoriales", la modificación se extiende a todos los registros de la tabla "libros" que hacen referencia a ella en "codigoeditorial":

update editoriales set codigo=10 where codigo=1;

Veamos si la actualización se extendió a "libros":

select titulo, autor, e.codigo,nombre

from libros as l

join editoriales as e

on codigoeditorial=e.codigo;

Si eliminamos una editorial, se borra tal editorial de "editoriales" y todos los registros de "libros" de dicha editorial:

delete from editoriales where codigo=2;

Veamos si la eliminación se extendió a "libros":

select codigo,titulo,autor,codigoeditorial from libros;

**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 not null,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint,

primary key (codigo)

);

create table editoriales(

codigo tinyint not null,

nombre varchar(20),

primary key (codigo)

);

insert into editoriales values(1,'Emece');

insert into editoriales values(2,'Planeta');

insert into editoriales values(3,'Siglo XXI');

insert into libros values(1,'El aleph','Borges',1);

insert into libros values(2,'Martin Fierro','Jose Hernandez',2);

insert into libros values(3,'Aprenda PHP','Mario Molina',2);

alter table libros

add constraint FK\_libros\_codigoeditorial

foreign key (codigoeditorial)

references editoriales(codigo)

on update cascade

on delete cascade;

update editoriales set codigo=10 where codigo=1;

select titulo, autor, e.codigo,nombre

from libros as l

join editoriales as e

on codigoeditorial=e.codigo;

delete from editoriales where codigo=2;

select codigo,titulo,autor,codigoeditorial from libros;

**Problema:**

Una empresa tiene registrados sus clientes en una tabla llamada "clientes", también tiene una tabla

"provincias" donde registra los nombres de las provincias.

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

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

drop table clientes;

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

drop table provincias;

2- Créelas con las siguientes estructuras:

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

codigoprovincia tinyint,

primary key(codigo)

);

create table provincias(

codigo tinyint,

nombre varchar(20),

primary key (codigo)

);

3- Ingrese algunos registros para ambas tablas:

insert into provincias values(1,'Cordoba');

insert into provincias values(2,'Santa Fe');

insert into provincias values(3,'Misiones');

insert into provincias values(4,'Rio Negro');

insert into clientes values('Perez Juan','San Martin 123','Carlos Paz',1);

insert into clientes values('Moreno Marcos','Colon 234','Rosario',2);

insert into clientes values('Acosta Ana','Avellaneda 333','Posadas',3);

4- Establezca una restricción "foreign key" especificando la acción "en cascade" para actualizaciones y "no\_action" para eliminaciones.

5- Intente eliminar el registro con código 3, de "provincias".

No se puede porque hay registros en "clientes" al cual hace referencia y la opción para eliminaciones se estableció como "no action".

6- Modifique el registro con código 3, de "provincias".

7- Verifique que el cambio se realizó en cascada, es decir, que se modificó en la tabla "provincias" y en "clientes":

select codigo,nombre from provincias;

select codigo, nombre, domicilio, ciudad, codigoprovincia

from clientes;

8- Intente modificar la restricción "foreign key" para que permita eliminación en cascada.

Mensaje de error, no se pueden modificar las restricciones.

9- Intente eliminar la tabla "provincias".

No se puede eliminar porque una restricción "foreign key" hace referencia a ella.

**Solución:**

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

drop table clientes;

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

drop table provincias;

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

codigoprovincia tinyint,

primary key(codigo)

);

create table provincias(

codigo tinyint,

nombre varchar(20),

primary key (codigo)

);

insert into provincias values(1,'Cordoba');

insert into provincias values(2,'Santa Fe');

insert into provincias values(3,'Misiones');

insert into provincias values(4,'Rio Negro');

insert into clientes values('Perez Juan','San Martin 123','Carlos Paz',1);

insert into clientes values('Moreno Marcos','Colon 234','Rosario',2);

insert into clientes values('Acosta Ana','Avellaneda 333','Posadas',3);

alter table clientes

add constraint FK\_clientes\_codigoprovincia

foreign key (codigoprovincia)

references provincias(codigo)

on update cascade

on delete no action;

delete from provincias where codigo=3;

update provincias set codigo=9 where codigo=3;

select codigo,nombre from provincias;

select codigo, nombre, domicilio, ciudad, codigoprovincia

from clientes;

alter table clientes

add constraint FK\_clientes\_codigoprovincia

foreign key (codigoprovincia)

references provincias(codigo)

on update cascade,

on delete cascade;

drop table provincias;

# Restricciones foreign key deshabilitar y eliminar (with check - nocheck)

Sabemos que si agregamos una restricción a una tabla que contiene datos, SQL Server los controla para asegurarse que cumplen con la restricción; es posible deshabilitar esta comprobación.

Podemos hacerlo al momento de agregar la restricción a una tabla con datos, incluyendo la opción "with nocheck" en la instrucción "alter table"; si se emplea esta opción, los datos no van a cumplir la restricción.

Se pueden deshabilitar las restricciones "check" y "foreign key", a las demás se las debe eliminar.

La sintaxis básica al agregar la restriccción "foreign key" es la siguiente:

alter table NOMBRETABLA1

with OPCIONDECHEQUEO

add constraint NOMBRECONSTRAINT

foreign key (CAMPOCLAVEFORANEA)

references NOMBRETABLA2 (CAMPOCLAVEPRIMARIA)

on update OPCION

on delete OPCION;

La opción "with OPCIONDECHEQUEO" especifica si se controlan los datos existentes o no con "check" y "nocheck" respectivamente. Por defecto, si no se especifica, la opción es "check".

En el siguiente ejemplo agregamos una restricción "foreign key" que controla que todos los códigos de editorial tengan un código válido, es decir, dicho código exista en "editoriales". La restricción no se aplica en los datos existentes pero si en los siguientes ingresos, modificaciones y actualizaciones:

alter table libros

with nocheck

add constraint FK\_libros\_codigoeditorial

foreing key (codigoeditorial)

references editoriales(codigo);

La comprobación de restricciones se puede deshabilitar para modificar, eliminar o agregar datos a una tabla sin comprobar la restricción. La sintaxis general es:

alter table NOMBRETABLA

OPCIONDECHEQUEO constraint NOMBRERESTRICCION;

En el siguiente ejemplo deshabilitamos la restricción creada anteriormente:

alter table libros

nocheck constraint FK\_libros\_codigoeditorial;

Para habilitar una restricción deshabilitada se ejecuta la misma instrucción pero con la cláusula "check" o "check all":

alter table libros

check constraint FK\_libros\_codigoeditorial;

Si se emplea "check constraint all" no se coloca nombre de restricciones, habilita todas las restricciones que tiene la tabla nombrada ("check" y "foreign key").

Para saber si una restricción está habilitada o no, podemos ejecutar el procedimiento almacenado "sp\_helpconstraint" y entenderemos lo que informa la columna "status\_enabled".

Entonces, las cláusulas "check" y "nocheck" permiten habilitar o deshabilitar restricciones "foreign key" (y "check"). Pueden emplearse para evitar la comprobación de datos existentes al crear la restricción o para deshabilitar la comprobación de datos al ingresar, actualizar y eliminar algún registro que infrinja la restricción.

Podemos eliminar una restricción "foreign key" con "alter table". La sintaxis básica es la misma que para cualquier otra restricción:

alter table TABLA

drop constraint NOMBRERESTRICCION;

Eliminamos la restricción de "libros":

alter table libros

drop constraint FK\_libros\_codigoeditorial;

No se puede eliminar una tabla si una restricción "foreign key" hace referencia a ella.

Cuando eliminamos una tabla que tiene una restricción "foreign key", la restricción también se elimina.

Problema:

Una librería almacena la información de sus libros para la venta en dos tablas, "libros" y "editoriales".

Eliminamos ambas tablas, si existen:

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

drop table libros;

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

drop table editoriales;

Creamos las tablas:

create table libros(

codigo int not null,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint,

primary key (codigo)

);

create table editoriales(

codigo tinyint not null,

nombre varchar(20),

primary key (codigo)

);

Ingresamos algunos registros:

insert into editoriales values(1,'Planeta');

insert into editoriales values(2,'Emece');

insert into editoriales values(3,'Paidos');

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

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

insert into libros values(3,'Aprenda PHP','Mario Molina',5);

Agregamos una restricción "foreign key" a la tabla "libros" para evitar que se ingresen códigos de editoriales inexistentes en "editoriales". Incluimos la opción "with nocheck" para evitar la comprobación de la restricción en los datos existentes (note que hay un libro que tiene un código de editorial inválido):

alter table libros

with nocheck

add constraint FK\_libros\_codigoeditorial

foreign key (codigoeditorial)

references editoriales(codigo);

La deshabilitación de la comprobación de la restricción no afecta a los siguientes ingresos, modificaciones y actualizaciones. Para poder ingresar, modificar o eliminar datos a una tabla sin que SQL Server compruebe la restricción debemos deshabilitarla:

alter table libros

nocheck constraint FK\_libros\_codigoeditorial;

Veamos si la restricción está habilitada o no:

sp\_helpconstraint libros;

En la columna "status\_enabled" de la restricción "foreign key" aparece "Disabled".

Veamos las restricciones de "editoriales":

sp\_helpconstraint editoriales;

Aparece la restricción "primary key" y nos informa que hay una restricción "foreign key" que hace referencia a ella de la tabla "libros" (aunque esté deshabilitada).

Ahora podemos ingresar un registro en "libros" con código inválido:

insert into libros values(4,'Ilusiones','Richard Bach',6);

También podemos modificar:

update editoriales set codigo=8 where codigo=1;

También realizar eliminaciones:

delete from editoriales where codigo=2;

Habilitamos la restricción:

alter table libros

check constraint FK\_libros\_codigoeditorial;

Veamos si la restricción está habilitada o no:

sp\_helpconstraint libros;

En la columna "status\_enabled" aparece "Enabled".

Eliminamos la restricción:

alter table libros

drop constraint FK\_libros\_codigoeditorial;

Ejecutamos el procedimiento almacenado sp\_helpconstraint para ver si la restricción se eliminó:

sp\_helpconstraint libros;

No aparece la restricción "foreign key".

Vemos las restricciones de "editoriales":

sp\_helpconstraint editoriales;

No aparece la restricción "foreign key" que hace referencia a esta tabla.

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 not null,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint,

primary key (codigo)

);

create table editoriales(

codigo tinyint not null,

nombre varchar(20),

primary key (codigo)

);

insert into editoriales values(1,'Planeta');

insert into editoriales values(2,'Emece');

insert into editoriales values(3,'Paidos');

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

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

insert into libros values(3,'Aprenda PHP','Mario Molina',5);

alter table libros

with nocheck

add constraint FK\_libros\_codigoeditorial

foreign key (codigoeditorial)

references editoriales(codigo);

alter table libros

nocheck constraint FK\_libros\_codigoeditorial;

sp\_helpconstraint libros;

sp\_helpconstraint editoriales;

insert into libros values(4,'Ilusiones','Richard Bach',6);

update editoriales set codigo=8 where codigo=1;

delete from editoriales where codigo=2;

alter table libros

check constraint FK\_libros\_codigoeditorial;

sp\_helpconstraint libros;

alter table libros

drop constraint FK\_libros\_codigoeditorial;

sp\_helpconstraint libros;

sp\_helpconstraint editoriales;

Problema:

Una librería almacena la información de sus libros para la venta en dos tablas, "libros" y "editoriales".

Eliminamos ambas tablas, si existen:

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

drop table libros;

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

drop table editoriales;

Creamos las tablas:

create table libros(

codigo int not null,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint,

primary key (codigo)

);

create table editoriales(

codigo tinyint not null,

nombre varchar(20),

primary key (codigo)

);

Ingresamos algunos registros:

insert into editoriales values(1,'Planeta');

insert into editoriales values(2,'Emece');

insert into editoriales values(3,'Paidos');

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

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

insert into libros values(3,'Aprenda PHP','Mario Molina',5);

Agregamos una restricción "foreign key" a la tabla "libros" para evitar que se ingresen códigos de editoriales inexistentes en "editoriales". Incluimos la opción "with nocheck" para evitar la comprobación de la restricción en los datos existentes (note que hay un libro que tiene un código de editorial inválido):

alter table libros

with nocheck

add constraint FK\_libros\_codigoeditorial

foreign key (codigoeditorial)

references editoriales(codigo);

La deshabilitación de la comprobación de la restricción no afecta a los siguientes ingresos, modificaciones y actualizaciones. Para poder ingresar, modificar o eliminar datos a una tabla sin que SQL Server compruebe la restricción debemos deshabilitarla:

alter table libros

nocheck constraint FK\_libros\_codigoeditorial;

Veamos si la restricción está habilitada o no:

sp\_helpconstraint libros;

En la columna "status\_enabled" de la restricción "foreign key" aparece "Disabled".

Veamos las restricciones de "editoriales":

sp\_helpconstraint editoriales;

Aparece la restricción "primary key" y nos informa que hay una restricción "foreign key" que hace referencia a ella de la tabla "libros" (aunque esté deshabilitada).

Ahora podemos ingresar un registro en "libros" con código inválido:

insert into libros values(4,'Ilusiones','Richard Bach',6);

También podemos modificar:

update editoriales set codigo=8 where codigo=1;

También realizar eliminaciones:

delete from editoriales where codigo=2;

Habilitamos la restricción:

alter table libros

check constraint FK\_libros\_codigoeditorial;

Veamos si la restricción está habilitada o no:

sp\_helpconstraint libros;

En la columna "status\_enabled" aparece "Enabled".

Eliminamos la restricción:

alter table libros

drop constraint FK\_libros\_codigoeditorial;

Ejecutamos el procedimiento almacenado sp\_helpconstraint para ver si la restricción se eliminó:

sp\_helpconstraint libros;

No aparece la restricción "foreign key".

Vemos las restricciones de "editoriales":

sp\_helpconstraint editoriales;

No aparece la restricción "foreign key" que hace referencia a esta tabla.

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 not null,

titulo varchar(40),

autor varchar(30),

codigoeditorial tinyint,

primary key (codigo)

);

create table editoriales(

codigo tinyint not null,

nombre varchar(20),

primary key (codigo)

);

insert into editoriales values(1,'Planeta');

insert into editoriales values(2,'Emece');

insert into editoriales values(3,'Paidos');

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

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

insert into libros values(3,'Aprenda PHP','Mario Molina',5);

alter table libros

with nocheck

add constraint FK\_libros\_codigoeditorial

foreign key (codigoeditorial)

references editoriales(codigo);

alter table libros

nocheck constraint FK\_libros\_codigoeditorial;

sp\_helpconstraint libros;

sp\_helpconstraint editoriales;

insert into libros values(4,'Ilusiones','Richard Bach',6);

update editoriales set codigo=8 where codigo=1;

delete from editoriales where codigo=2;

alter table libros

check constraint FK\_libros\_codigoeditorial;

sp\_helpconstraint libros;

alter table libros

drop constraint FK\_libros\_codigoeditorial;

sp\_helpconstraint libros;

sp\_helpconstraint editoriales;

problema:

Una empresa tiene registrados sus clientes en una tabla llamada "clientes", también tiene una tabla

"provincias" donde registra los nombres de las provincias.

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

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

drop table clientes;

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

drop table provincias;

2- Créelas con las siguientes estructuras:

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

codigoprovincia tinyint,

primary key(codigo)

);

create table provincias(

codigo tinyint,

nombre varchar(20),

primary key (codigo)

);

3- Ingrese algunos registros para ambas tablas:

insert into provincias values(1,'Cordoba');

insert into provincias values(2,'Santa Fe');

insert into provincias values(3,'Misiones');

insert into provincias values(4,'Rio Negro');

insert into clientes values('Perez Juan','San Martin 123','Carlos Paz',1);

insert into clientes values('Moreno Marcos','Colon 234','Rosario',2);

insert into clientes values('Garcia Juan','Sucre 345','Cordoba',1);

insert into clientes values('Lopez Susana','Caseros 998','Posadas',3);

insert into clientes values('Marcelo Moreno','Peru 876','Viedma',4);

insert into clientes values('Lopez Sergio','Avellaneda 333','La Plata',5);

4- Intente agregar una restricción "foreign key" para que los códigos de provincia de "clientes"

existan en "provincias" con acción en cascada para actualizaciones y eliminaciones, sin especificar

la opción de comprobación de datos:

alter table clientes

add constraint FK\_clientes\_codigoprovincia

foreign key (codigoprovincia)

references provincias(codigo)

on update cascade

on delete cascade;

No se puede porque al no especificar opción para la comprobación de datos, por defecto es "check" y

hay un registro que no cumple con la restricción.

5- Agregue la restricción anterior pero deshabilitando la comprobación de datos existentes:

alter table clientes

with nocheck

add constraint FK\_clientes\_codigoprovincia

foreign key (codigoprovincia)

references provincias(codigo)

on update cascade

on delete cascade;

6- Vea las restricciones de "clientes":

sp\_helpconstraint clientes;

Aparece la restricción "primary key" y "foreign key", las columnas "delete\_action" y "update\_action"

contienen "cascade" y la columna "status\_enabled" contiene "Enabled".

7- Vea las restricciones de "provincias":

sp\_helpconstraint provincias;

Aparece la restricción "primary key" y la referencia a esta tabla de la restricción "foreign key" de

la tabla "clientes.

8- Deshabilite la restricción "foreign key" de "clientes":

alter table clientes

nocheck constraint FK\_clientes\_codigoprovincia;

9- Vea las restricciones de "clientes":

sp\_helpconstraint clientes;

la restricción "foreign key" aparece inhabilitada.

10- Vea las restricciones de "provincias":

sp\_helpconstraint provincias;

informa que la restricción "foreign key" de "clientes" hace referencia a ella, aún cuando está

deshabilitada.

11- Agregue un registro que no cumpla la restricción "foreign key":

insert into clientes values('Garcia Omar','San Martin 100','La Pampa',6);

Se permite porque la restricción está deshabilitada.

12- Elimine una provincia de las cuales haya clientes:

delete from provincias where codigo=2;

13- Corrobore que el registro se eliminó de "provincias" pero no se extendió a "clientes":

select \*from clientes;

select \*from provincias;

14- Modifique un código de provincia de la cual haya clientes:

update provincias set codigo=9 where codigo=3;

15- Verifique que el cambio se realizó en "provincias" pero no se extendió a "clientes":

select \*from clientes;

select \*from provincias;

16- Intente eliminar la tabla "provincias":

drop table provincias;

No se puede porque la restricción "FK\_clientes\_codigoprovincia" la referencia, aunque esté deshabilitada.

17- Habilite la restricción "foreign key":

alter table clientes

check constraint FK\_clientes\_codigoprovincia;

18- Intente agregar un cliente con código de provincia inexistente en "provincias":

insert into clientes values('Hector Ludueña','Paso 123','La Plata',8);

No se puede.

19- Modifique un código de provincia al cual se haga referencia en "clientes":

update provincias set codigo=20 where codigo=4;

Actualización en cascada.

20- Vea que se modificaron en ambas tablas:

select \*from clientes;

select \*from provincias;

21- Elimine una provincia de la cual haya referencia en "clientes":

delete from provincias where codigo=1;

Acción en cascada.

22- Vea que los registros de ambas tablas se eliminaron:

select \*from clientes;

select \*from provincias;

23- Elimine la restriccion "foreign key":

alter table clientes

drop constraint FK\_clientes\_codigoprovincia;

24- Vea las restriciones de la tabla "provincias":

sp\_helpconstraint provincias;

Solamente aparece la restricción "primary key", ya no hay una "foreign key" que la referencie.

25- Elimine la tabla "provincias":

drop table provincias;

Puede eliminarse porque no hay restricción "foreign key" que la referencie.

Solución:

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

drop table clientes;

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

drop table provincias;

create table clientes (

codigo int identity,

nombre varchar(30),

domicilio varchar(30),

ciudad varchar(20),

codigoprovincia tinyint,

primary key(codigo)

);

create table provincias(

codigo tinyint,

nombre varchar(20),

primary key (codigo)

);

insert into provincias values(1,'Cordoba');

insert into provincias values(2,'Santa Fe');

insert into provincias values(3,'Misiones');

insert into provincias values(4,'Rio Negro');

insert into clientes values('Perez Juan','San Martin 123','Carlos Paz',1);

insert into clientes values('Moreno Marcos','Colon 234','Rosario',2);

insert into clientes values('Garcia Juan','Sucre 345','Cordoba',1);

insert into clientes values('Lopez Susana','Caseros 998','Posadas',3);

insert into clientes values('Marcelo Moreno','Peru 876','Viedma',4);

insert into clientes values('Lopez Sergio','Avellaneda 333','La Plata',5);

alter table clientes

add constraint FK\_clientes\_codigoprovincia

foreign key (codigoprovincia)

references provincias(codigo)

on update cascade

on delete cascade;

alter table clientes

with nocheck

add constraint FK\_clientes\_codigoprovincia

foreign key (codigoprovincia)

references provincias(codigo)

on update cascade

on delete cascade;

sp\_helpconstraint clientes;

sp\_helpconstraint provincias;

alter table clientes

nocheck constraint FK\_clientes\_codigoprovincia;

sp\_helpconstraint clientes;

sp\_helpconstraint provincias;

insert into clientes values('Garcia Omar','San Martin 100','La Pampa',6);

delete from provincias where codigo=2;

select \*from clientes;

select \*from provincias;

update provincias set codigo=9 where codigo=3;

select \*from clientes;

select \*from provincias;

drop table provincias;

alter table clientes

check constraint FK\_clientes\_codigoprovincia;

insert into clientes values('Hector Ludueña','Paso 123','La Plata',8);

update provincias set codigo=20 where codigo=4;

select \*from clientes;

select \*from provincias;

delete from provincias where codigo=1;

select \*from clientes;

select \*from provincias;

alter table clientes

drop constraint FK\_clientes\_codigoprovincia;

sp\_helpconstraint provincias;

drop table provincias;

# Restricciones foreign key (información)

Restricciones foreign key (información)

El procedimiento almacenado "sp\_helpconstraint" devuelve las siguientes columnas:

- constraint\_type: tipo de restricción. Si es una restricción de campo (default o check) indica sobre qué campo fue establecida. Si es de tabla (primary key o unique) indica el tipo de índice creado. Si es una "foreign key" lo indica.

- constraint\_name: nombre de la restricción.

- delete\_action: solamente es aplicable para restricciones de tipo "foreign key". Indica si la acción de eliminación actúa, no actúa o es en cascada. Indica "n/a" en cualquier restricción para la que no se aplique; "No Action" si no actúa y "Cascade" si es en cascada.

- update\_action: sólo es aplicable para restricciones de tipo "foreign key". Indica si la acción de actualización es: No Action, Cascade, or n/a. Indica "n/a" en cualquier restricción para la que no se aplique.

- status\_enabled: solamente es aplicable para restricciones de tipo "check" y "foreign key". Indica si está habilitada (Enabled) o no (Disabled). Indica "n/a" en cualquier restricción para la que no se aplique.

- status\_for\_replication: solamente es aplicable para restricciones de tipo "check" y "foreign key". Indica "n/a" en cualquier restricción para la que no se aplique.

- constraint\_keys: Si es una restricción "default" muestra la condición de chequeo; si es una restricción "default", el valor por defecto; si es una "primary key", "unique" o "foreign key" muestra el/ los campos a los que se aplicaron la restricción. En caso de valores predeterminados y reglas, el texto que lo define.

# Restricciones al crear la tabla

Restricciones al crear la tabla

Hasta el momento hemos agregado restricciones a tablas existentes con "alter table" (manera aconsejada), también pueden establecerse al momento de crear una tabla (en la instrucción "create table").

Podemos aplicar restricciones a nivel de campo (restricción de campo) o a nivel de tabla (restricción de tabla).

En el siguiente ejemplo creamos la tabla "libros" con varias restricciones:

create table libros(

codigo int identity,

titulo varchar(40),

codigoautor int not null,

codigoeditorial tinyint not null,

precio decimal(5,2)

constraint DF\_precio default (0),

constraint PK\_libros\_codigo

primary key clustered (codigo),

constraint UQ\_libros\_tituloautor

unique (titulo,codigoautor),

constraint FK\_libros\_editorial

foreign key (codigoeditorial)

references editoriales(codigo)

on update cascade,

constraint FK\_libros\_autores

foreign key (codigoautor)

references autores(codigo)

on update cascade,

constraint CK\_precio\_positivo check (precio>=0)

);

En el ejemplo anterior creamos:

- una restricción "default" para el campo "precio" (restricción a nivel de campo);

- una restricción "primary key" con índice agrupado para el campo "codigo" (a nivel de tabla);

- una restricción "unique" con índice no agrupado (por defecto) para los campos "titulo" y "codigoautor" (a nivel de tabla);

- una restricción "foreign key" para establecer el campo "codigoeditorial" como clave externa que haga referencia al campo "codigo" de "editoriales y permita actualizaciones en cascada y no eliminaciones (por defecto "no action");

- una restricción "foreign key" para establecer el campo "codigoautor" como clave externa que haga referencia al campo "codigo" de "autores" y permita actualizaciones en cascada y no eliminaciones;

- una restricción "check" para el campo "precio" que no admita valores negativos;

Si definimos una restricción "foreign key" al crear una tabla, la tabla referenciada debe existir.

Problema:

Trabajamos con las tablas "libros", "autores" y "editoriales" de una librería:

Eliminamos las tablas si existen:

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

drop table libros;

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

drop table editoriales;

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

drop table autores;

Creamos la tabla "editoriales" con una restricción "primary key":

create table editoriales(

codigo tinyint not null,

nombre varchar(30),

constraint PK\_editoriales primary key (codigo)

);

Creamos la tabla "autores" con una restricción "primary key", una "unique" y una "check":

create table autores(

codigo int not null

constraint CK\_autores\_codigo check (codigo>=0),

nombre varchar(30) not null,

constraint PK\_autores\_codigo

primary key (codigo),

constraint UQ\_autores\_nombre

unique (nombre),

);

Aplicamos varias restricciones cuando creamos la tabla "libros":

create table libros(

codigo int identity,

titulo varchar(40),

codigoautor int not null,

codigoeditorial tinyint not null,

precio decimal(5,2)

constraint DF\_libros\_precio default (0),

constraint PK\_libros\_codigo

primary key clustered (codigo),

constraint UQ\_libros\_tituloautor

unique (titulo,codigoautor),

constraint FK\_libros\_editorial

foreign key (codigoeditorial)

references editoriales(codigo)

on update cascade,

constraint FK\_libros\_autores

foreign key (codigoautor)

references autores(codigo)

on update cascade,

constraint CK\_libros\_precio\_positivo check (precio>=0)

);

Veamos las restricciones de "editoriales":

sp\_helpconstraint editoriales;

Aparece la restricción "primary key" para el campo "codigo" y la restricción "foreign key" de "libros" "FK\_libros\_editorial" que referencia esta tabla.

Veamos las restricciones de "autores":

sp\_helpconstraint autores;

Aparecen 4 restricciones: una restricción "check" para el campo "codigo", una restricción "primary key" para el campo "codigo", una restricción "unique" para el campo "nombre" y la restricción "foreign key" de "libros" "FK\_libros\_autores" que referencia esta tabla.

Veamos las restricciones de "libros":

sp\_helpconstraint libros;

Aparecen 6 restricciones: una restricción "check" sobre el campo "precio", una "default" sobre el campo "precio", una restricción "foreign key" que establece el campo "codigoeditorial" como clave externa que hace referencia al campo "codigo" de "editoriales" y permite actualizaciones en cascada y no eliminaciones, una restricción "foreign key" que establece el campo "codigoautor" como clave externa que hace referencia al campo "codigo" de "autores" y permite actualizaciones en cascada y no eliminaciones, una restricción "primary key" con índice agrupado para el campo "codigo" y una restricción "unique" con índice no agrupado para los campos "titulo" y "codigoautor".

Recuerde que si definimos una restricción "foreign key" al crear una tabla, la tabla referenciada debe existir, por ello creamos las tablas "editoriales" y "autores" antes que "libros".

También debemos ingresar registros en las tablas "autores" y "editoriales" antes que en "libros", a menos que deshabilitemos la restricción "foreign key".

Código:

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

drop table libros;

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

drop table editoriales;

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

drop table autores;

create table editoriales(

codigo tinyint not null,

nombre varchar(30),

constraint PK\_editoriales primary key (codigo)

);

create table autores(

codigo int not null

constraint CK\_autores\_codigo check (codigo>=0),

nombre varchar(30) not null,

constraint PK\_autores\_codigo

primary key (codigo),

constraint UQ\_autores\_nombre

unique (nombre),

);

create table libros(

codigo int identity,

titulo varchar(40),

codigoautor int not null,

codigoeditorial tinyint not null,

precio decimal(5,2)

constraint DF\_libros\_precio default (0),

constraint PK\_libros\_codigo

primary key clustered (codigo),

constraint UQ\_libros\_tituloautor

unique (titulo,codigoautor),

constraint FK\_libros\_editorial

foreign key (codigoeditorial)

references editoriales(codigo)

on update cascade,

constraint FK\_libros\_autores

foreign key (codigoautor)

references autores(codigo)

on update cascade,

constraint CK\_libros\_precio\_positivo check (precio>=0)

);

sp\_helpconstraint editoriales;

sp\_helpconstraint autores;

sp\_helpconstraint libros;

problema:

Un club de barrio tiene en su sistema 4 tablas:

- "socios": en la cual almacena documento, número, nombre y domicilio de cada socio;

- "deportes": que guarda un código, nombre del deporte, día de la semana que se dicta y documento

del profesor instructor;

- "profesores": donde se guarda el documento, nombre y domicilio de los profesores e

- "inscritos": que almacena el número de socio, el código de deporte y si la matricula está paga o

no.

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('profesores') is not null

drop table profesores;

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

drop table deportes;

2- Considere que:

- un socio puede inscribirse en varios deportes, pero no dos veces en el mismo.

- un socio tiene un documento único y un número de socio único.

- el documento del socio debe contener 8 dígitos.

- un deporte debe tener asignado un profesor que exista en "profesores" o "null" si aún no tiene un

instructor definido.

- el campo "dia" de "deportes" puede ser: lunes, martes, miercoles, jueves, viernes o sabado.

- el campo "dia" de "deportes" por defecto debe almacenar 'sabado'.

- un profesor puede ser instructor de varios deportes o puede no dictar ningún deporte.

- un profesor no puede estar repetido en "profesores".

- el documento del profesor debe contener 8 dígitos.

- un inscripto debe ser socio, un socio puede no estar inscripto en ningún deporte.

- una inscripción debe tener un valor en socio existente en "socios" y un deporte que exista en

"deportes".

- el campo "matricula" de "inscritos" debe aceptar solamente los caracteres 's' o 'n'.

3- Cree las tablas con las restricciones necesarias:

create table profesores(

documento char(8) not null,

nombre varchar(30),

domicilio varchar(30),

constraint CK\_profesores\_documento\_patron check (documento like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),

constraint PK\_profesores\_documento

primary key (documento)

);

create table deportes(

codigo tinyint identity,

nombre varchar(20) not null,

dia varchar(30)

constraint DF\_deportes\_dia default('sabado'),

profesor char(8),--documento del profesor

constraint CK\_deportes\_dia\_lista check (dia in ('lunes','martes','miercoles','jueves','viernes','sabado')),

constraint PK\_deportes\_codigo

primary key (codigo)

);

create table socios(

numero int identity,

documento char(8),

nombre varchar(30),

domicilio varchar(30),

constraint CK\_documento\_patron check (documento like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),

constraint PK\_socios\_numero

primary key nonclustered(numero),

constraint UQ\_socios\_documento

unique clustered(documento)

);

create table inscritos(

numerosocio int not null,

codigodeporte tinyint,

matricula char(1),

constraint PK\_inscritos\_numerodeporte

primary key clustered (numerosocio,codigodeporte),

constraint FK\_inscritos\_deporte

foreign key (codigodeporte)

references deportes(codigo)

on update cascade,

constraint FK\_inscritos\_socios

foreign key (numerosocio)

references socios(numero)

on update cascade

on delete cascade,

constraint CK\_matricula\_valores check (matricula in ('s','n'))

);

4- Ingrese registros en "profesores":

insert into profesores values('21111111','Andres Acosta','Avellaneda 111');

insert into profesores values('22222222','Betina Bustos','Bulnes 222');

insert into profesores values('23333333','Carlos Caseros','Colon 333');

5- Ingrese registros en "deportes". Ingrese el mismo día para distintos deportes, un deporte sin día

confirmado, un deporte sin profesor definido:

insert into deportes values('basquet','lunes',null);

insert into deportes values('futbol','lunes','23333333');

insert into deportes values('natacion',null,'22222222');

insert into deportes values('padle',default,'23333333');

insert into deportes (nombre,dia) values('tenis','jueves');

6- Ingrese registros en "socios":

insert into socios values('30111111','Ana Acosta','America 111');

insert into socios values('30222222','Bernardo Bueno','Bolivia 222');

insert into socios values('30333333','Camila Conte','Caseros 333');

insert into socios values('30444444','Daniel Duarte','Dinamarca 444');

7- Ingrese registros en "inscritos". Inscriba a un socio en distintos deportes, inscriba varios

socios en el mismo deporte.

insert into inscritos values(1,3,'s');

insert into inscritos values(1,5,'s');

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

insert into inscritos values(4,1,'n');

insert into inscritos values(4,4,'s');

8- Realice un "join" (del tipo que sea necesario) para mostrar todos los datos del socio junto con

el nombre de los deportes en los cuales está inscripto, el día que tiene que asistir y el nombre del

profesor que lo instruirá.

5 registros.

9- Realice la misma consulta anterior pero incluya los socios que no están inscritos en ningún

deporte.

6 registros.

10- Muestre todos los datos de los profesores, incluido el deporte que dicta y el día, incluyendo

los profesores que no tienen asignado ningún deporte.

4 registros.

11- Muestre todos los deportes y la cantidad de inscritos, incluyendo aquellos deportes para los

cuales no hay inscritos.

5 registros.

12- Muestre las restricciones de "socios".

3 restricciones y 1 "foreign key" de "inscritos" que la referencia.

13- Muestre las restricciones de "deportes".

3 restricciones y 1 "foreign key" de "inscritos" que la referencia.

14- Muestre las restricciones de "profesores".

2 restricciones.

15- Muestre las restricciones de "inscritos".

4 restricciones.

Solución:

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

drop table inscritos;

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

drop table socios;

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

drop table profesores;

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

drop table deportes;

create table profesores(

documento char(8) not null,

nombre varchar(30),

domicilio varchar(30),

constraint CK\_profesores\_documento\_patron check (documento like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),

constraint PK\_profesores\_documento

primary key (documento)

);

create table deportes(

codigo tinyint identity,

nombre varchar(20) not null,

dia varchar(30)

constraint DF\_deportes\_dia default('sabado'),

profesor char(8),--documento del profesor

constraint CK\_deportes\_dia\_lista check (dia in ('lunes','martes','miercoles','jueves','viernes','sabado')),

constraint PK\_deportes\_codigo

primary key (codigo)

);

create table socios(

numero int identity,

documento char(8),

nombre varchar(30),

domicilio varchar(30),

constraint CK\_documento\_patron check (documento like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),

constraint PK\_socios\_numero

primary key nonclustered(numero),

constraint UQ\_socios\_documento

unique clustered(documento)

);

create table inscritos(

numerosocio int not null,

codigodeporte tinyint,

matricula char(1),

constraint PK\_inscritos\_numerodeporte

primary key clustered (numerosocio,codigodeporte),

constraint FK\_inscritos\_deporte

foreign key (codigodeporte)

references deportes(codigo)

on update cascade,

constraint FK\_inscritos\_socios

foreign key (numerosocio)

references socios(numero)

on update cascade

on delete cascade,

constraint CK\_matricula\_valores check (matricula in ('s','n'))

);

insert into profesores values('21111111','Andres Acosta','Avellaneda 111');

insert into profesores values('22222222','Betina Bustos','Bulnes 222');

insert into profesores values('23333333','Carlos Caseros','Colon 333');

insert into deportes values('basquet','lunes',null);

insert into deportes values('futbol','lunes','23333333');

insert into deportes values('natacion',null,'22222222');

insert into deportes values('padle',default,'23333333');

insert into deportes (nombre,dia) values('tenis','jueves');

insert into socios values('30111111','Ana Acosta','America 111');

insert into socios values('30222222','Bernardo Bueno','Bolivia 222');

insert into socios values('30333333','Camila Conte','Caseros 333');

insert into socios values('30444444','Daniel Duarte','Dinamarca 444');

insert into inscritos values(1,3,'s');

insert into inscritos values(1,5,'s');

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

insert into inscritos values(4,1,'n');

insert into inscritos values(4,4,'s');

select s.\*,d.nombre as deporte,d.dia,p.nombre as profesor

from socios as s

join inscritos as i

on numero=i.numerosocio

join deportes as d

on d.codigo=i.codigodeporte

left join profesores as p

on d.profesor=p.documento;

select s.\*,d.nombre as deporte,d.dia,p.nombre as profesor

from socios as s

full join inscritos as i

on numero=i.numerosocio

left join deportes as d

on d.codigo=i.codigodeporte

left join profesores as p

on d.profesor=p.documento;

select p.\*,d.nombre as deporte,d.dia

from profesores as p

left join deportes as d

on d.profesor=p.documento;

select d.nombre,count(i.codigodeporte) as cantidad

from deportes as d

left join inscritos as i

on d.codigo=i.codigodeporte

group by d.nombre;

sp\_helpconstraint socios;

sp\_helpconstraint deportes;

sp\_helpconstraint profesores;

sp\_helpconstraint inscritos;

# Unión

Unión

El operador "union" combina el resultado de dos o más instrucciones "select" en un único resultado.

Se usa cuando los datos que se quieren obtener pertenecen a distintas tablas y no se puede acceder a ellos con una sola consulta.

Es necesario que las tablas referenciadas tengan tipos de datos similares, la misma cantidad de campos y el mismo orden de campos en la lista de selección de cada consulta. No se incluyen las filas duplicadas en el resultado, a menos que coloque la opción "all".

Se deben especificar los nombres de los campos en la primera instrucción "select".

Puede emplear la cláusula "order by".

Puede dividir una consulta compleja en varias consultas "select" y luego emplear el operador "union" para combinarlas.

Una academia de enseñanza almacena los datos de los alumnos en una tabla llamada "alumnos" y los datos de los profesores en otra denominada "profesores".

La academia necesita el nombre y domicilio de profesores y alumnos para enviarles una tarjeta de invitación.

Para obtener los datos necesarios de ambas tablas en una sola consulta necesitamos realizar una unión:

select nombre, domicilio from alumnos

union

select nombre, domicilio from profesores;

El primer "select" devuelve el nombre y domicilio de todos los alumnos; el segundo, el nombre y domicilio de todos los profesores.

Los encabezados del resultado de una unión son los que se especifican en el primer "select".

Problema:

Una academia de enseñanza almacena los datos de los alumnos en una tabla llamada "alumnos" y los datos de los profesores en otra denominada "profesores".

Eliminamos las tablas si existen:

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

drop table alumnos;

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

drop table profesores;

Creamos las tablas:

create table profesores(

documento varchar(8) not null,

nombre varchar (30),

domicilio varchar(30),

primary key(documento)

);

create table alumnos(

documento varchar(8) not null,

nombre varchar (30),

domicilio varchar(30),

primary key(documento)

);

Ingresamos algunos registros:

insert into alumnos values('30000000','Juan Perez','Colon 123');

insert into alumnos values('30111111','Marta Morales','Caseros 222');

insert into alumnos values('30222222','Laura Torres','San Martin 987');

insert into alumnos values('30333333','Mariano Juarez','Avellaneda 34');

insert into alumnos values('23333333','Federico Lopez','Colon 987');

insert into profesores values('22222222','Susana Molina','Sucre 345');

insert into profesores values('23333333','Federico Lopez','Colon 987');

La academia necesita el nombre y domicilio de profesores y alumnos para enviarles una tarjeta de invitación.

Empleamos el operador "union" para obtener dicha información de ambas tablas:

select nombre, domicilio from alumnos

union

select nombre, domicilio from profesores;

Note que existe un profesor que también está presente en la tabla "alumnos"; dicho registro aparece una sola vez en el resultado de "union". Si queremos que las filas duplicadas aparezcan, debemos emplear "all":

select nombre, domicilio from alumnos

union all

select nombre, domicilio from profesores;

Ordenamos por domicilio:

select nombre, domicilio from alumnos

union

select nombre, domicilio from profesores

order by domicilio;

Podemos agregar una columna extra a la consulta con el encabezado "condicion" en la que aparezca el literal "profesor" o "alumno" según si la persona es uno u otro:

select nombre, domicilio, 'alumno' as condicion from alumnos

union

select nombre, domicilio,'profesor' from profesores

order by condicion;

Código:

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

drop table alumnos;

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

drop table profesores;

create table profesores(

documento varchar(8) not null,

nombre varchar (30),

domicilio varchar(30),

primary key(documento)

);

create table alumnos(

documento varchar(8) not null,

nombre varchar (30),

domicilio varchar(30),

primary key(documento)

);

insert into alumnos values('30000000','Juan Perez','Colon 123');

insert into alumnos values('30111111','Marta Morales','Caseros 222');

insert into alumnos values('30222222','Laura Torres','San Martin 987');

insert into alumnos values('30333333','Mariano Juarez','Avellaneda 34');

insert into alumnos values('23333333','Federico Lopez','Colon 987');

insert into profesores values('22222222','Susana Molina','Sucre 345');

insert into profesores values('23333333','Federico Lopez','Colon 987');

select nombre, domicilio from alumnos

union

select nombre, domicilio from profesores;

select nombre, domicilio from alumnos

union all

select nombre, domicilio from profesores;

select nombre, domicilio from alumnos

union

select nombre, domicilio from profesores

order by domicilio;

select nombre, domicilio, 'alumno' as condicion from alumnos

union

select nombre, domicilio,'profesor' from profesores

order by condicion;

problema:

Un supermercado almacena en una tabla denominada "proveedores" los datos de las compañías que le

proveen de mercaderías; en una tabla llamada "clientes", los datos de los comercios que le compran y

en otra tabla "empleados" los datos de los empleados.

1- Elimine las tablas si existen:

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

drop table clientes;

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

drop table proveedores;

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

drop table empleados;

2- Cree las tablas:

create table proveedores(

codigo int identity,

nombre varchar (30),

domicilio varchar(30),

primary key(codigo)

);

create table clientes(

codigo int identity,

nombre varchar (30),

domicilio varchar(30),

primary key(codigo)

);

create table empleados(

documento char(8) not null,

nombre varchar(20),

apellido varchar(20),

domicilio varchar(30),

primary key(documento)

);

3- Ingrese algunos registros:

insert into proveedores values('Bebida cola','Colon 123');

insert into proveedores values('Carnes Unica','Caseros 222');

insert into proveedores values('Lacteos Blanca','San Martin 987');

insert into clientes values('Supermercado Lopez','Avellaneda 34');

insert into clientes values('Almacen Anita','Colon 987');

insert into clientes values('Garcia Juan','Sucre 345');

insert into empleados values('23333333','Federico','Lopez','Colon 987');

insert into empleados values('28888888','Ana','Marquez','Sucre 333');

insert into empleados values('30111111','Luis','Perez','Caseros 956');

4- El supermercado quiere enviar una tarjeta de salutación a todos los proveedores, clientes y

empleados y necesita el nombre y domicilio de todos ellos. Emplee el operador "union" para obtener

dicha información de las tres tablas.

5- Agregue una columna con un literal para indicar si es un proveedor, un cliente o un empleado y

ordene por dicha columna.

Solución:

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

drop table clientes;

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

drop table proveedores;

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

drop table empleados;

create table proveedores(

codigo int identity,

nombre varchar (30),

domicilio varchar(30),

primary key(codigo)

);

create table clientes(

codigo int identity,

nombre varchar (30),

domicilio varchar(30),

primary key(codigo)

);

create table empleados(

documento char(8) not null,

nombre varchar(20),

apellido varchar(20),

domicilio varchar(30),

primary key(documento)

);

insert into proveedores values('Bebida cola','Colon 123');

insert into proveedores values('Carnes Unica','Caseros 222');

insert into proveedores values('Lacteos Blanca','San Martin 987');

insert into clientes values('Supermercado Lopez','Avellaneda 34');

insert into clientes values('Almacen Anita','Colon 987');

insert into clientes values('Garcia Juan','Sucre 345');

insert into empleados values('23333333','Federico','Lopez','Colon 987');

insert into empleados values('28888888','Ana','Marquez','Sucre 333');

insert into empleados values('30111111','Luis','Perez','Caseros 956');

select nombre, domicilio from proveedores

union

select nombre, domicilio from clientes

union

select (apellido+' '+nombre), domicilio from empleados;

select nombre, domicilio, 'proveedor' as categoria from proveedores

union

select nombre, domicilio, 'cliente' from clientes

union

select (apellido+' '+nombre), domicilio , 'empleado' from empleados

order by categoria;