# SQL Primera parte

## Objetivos y alcances del tutorial de Microsoft Sql Server

El curso brinda un concepto teórico corto, luego un problema resuelto que hay que ejecutar, modificar y jugar con el mismo. Por último, y lo más importante, una serie de ejercicios propuestos que nos permitirá saber si podemos aplicar el concepto.

Usaremos Microsoft SQLServer Express y Microsoft SQL Server Management Studio.

## Previo:

Crear la Base de datos:

create database PruebaSQL

Para asegurarnos que las operaciones sobre tablas se realicen en esa Base de datos tendríamos que indicar siempre la base de datos en la que vamos a trabajar añadiendo en las sentencias:

Use NombreBaseDatos;

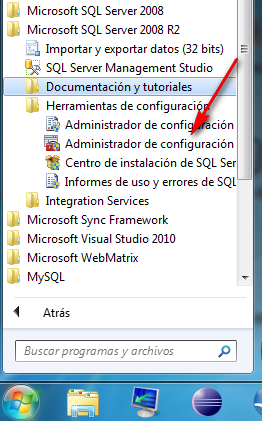
Para asegurarnos que un objeto está en una base de datos sería mediante:

NombreBaseDatos.dbo.NombreTabla

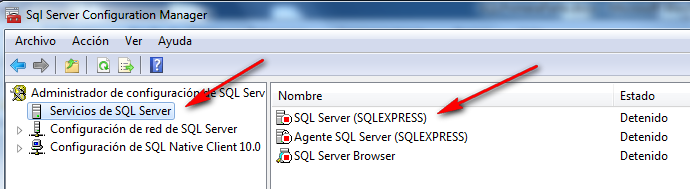
Donde dbo indica el propietario de la Base de Datos. Si no coinciden tendríamos nombres de tablas diferentes.

Para ejecutar los procedimientos junto a otras sentencias SQL hay que poner delante el comando exec

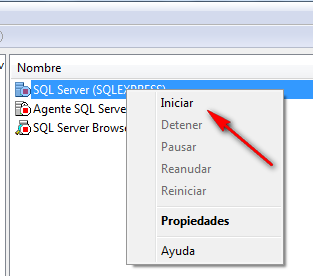
Para activar el Servicio que permite acceder a la base de datos hay que hacerlo mediante el administrador de configuración de SQL Server:



Este nos mostrará los Servicios del SQL Server, de los que tendremos que activar el que indica SQL Server y entre paréntesis el nombre de la instancia a poner en marcha, normalmente SQLEXPRESS:

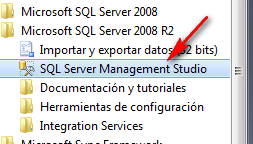


Botón de la derecha sobre el servicio y clic sobre Iniciar:

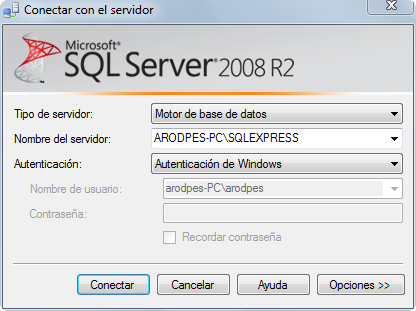


Debe quedar con un triángulo verde: 

Para trabajar con las consultas usaremos el Management Studio del SQL Server:

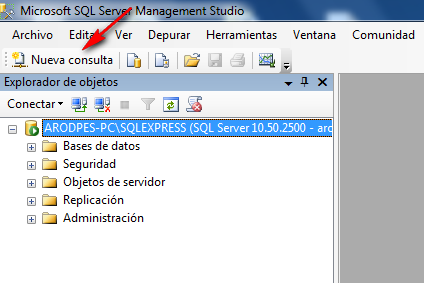


Al iniciar el programa nos solicitará las credenciales de acceso:

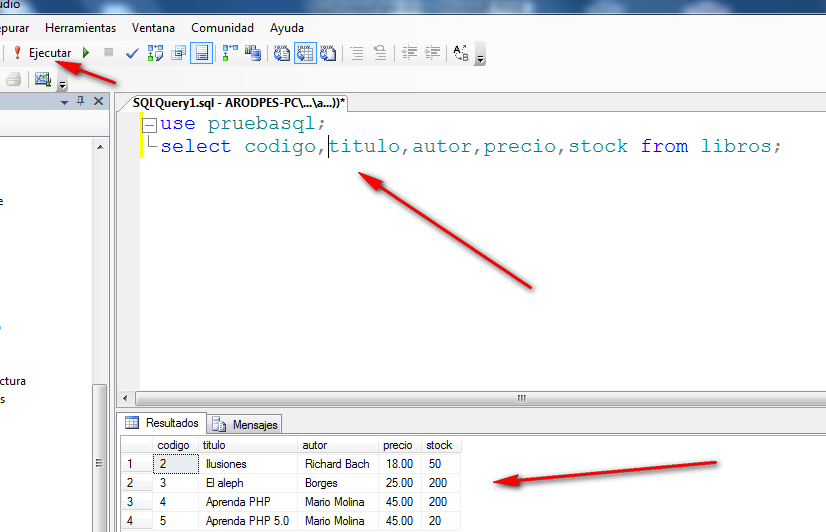


Normalmente estarán correctas, indicando en nombre del equipo, una barra y el nombre de la instancia, después autenticación Windows, que permitirá hacer todas las operaciones posibles si somos Administradores del equipo. Esto se podrá configurar, pero se verá más adelante.

Una vez se abra el programa tendremos disponible múltiples opciones de configuración y acceso al SGBD. Para efectuar una consulta haremos clic en Nueva consulta:



Se abrirá una página en blanco donde podremos escribir las sentencias SQL:



Para ejecutarla haremos click en el botón Ejecutar, nos dará los resultados en las ventanas de debajo.

## Crear una tabla (create table - sp\_tables - sp\_columns - drop table)

Cuando se crea una tabla debemos indicar su nombre y definir al menos un campo con su tipo de dato. En esta tabla "usuarios" definimos 2 campos:

nombre: que contendrá una cadena de caracteres de 30 caracteres de longitud, que almacenará el nombre de usuario y

clave: otra cadena de caracteres de 10 de longitud, que guardará la clave de cada usuario.

Cada usuario ocupará un registro de esta tabla, con su respectivo nombre y clave.

Para nombres de tablas, se puede utilizar cualquier caracter permitido para nombres de directorios, el primero debe ser un caracter alfabético y es conveniente que no contenga espacios. La longitud máxima es de 128 caracteres.

Si intentamos crear una tabla con un nombre ya existente (existe otra tabla con ese nombre), mostrará un mensaje indicando que ya hay un objeto llamado 'usuarios' en la base de datos y la sentencia no se ejecutará.

El formato básico de creación de tablas será:

Create table NOMBRE\_DE\_LA\_TABLA

(

NOMBRE\_CAMPO\_1 TIPO\_CAMPO\_1 ,

NOMBRE\_CAMPO\_2 TIPO\_CAMPO\_2 ,

AÑADIREMOS TANTOS CAMPOS COMO NECESITEMOS

NOMBRE\_CAMPO\_N TIPO\_CAMPO\_N

);

La sentencia para crear la tabla será:

create table usuarios (

nombre varchar(30),

clave varchar(10)

);

Para ver la estructura de una tabla usamos el procedimiento almacenado "sp\_columns" junto al nombre de la tabla:

sp\_columns usuarios;

Aparece mucha información que no analizaremos en detalle, como el nombre de la tabla, su propietario, los campos, el tipo de dato de cada campo, su longitud, etc.:

...COLUMN\_NAME TYPE\_NAME LENGHT

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

nombre varchar 30

clave varchar 10

Para eliminar una tabla usamos "drop table" junto al nombre de la tabla a eliminar:

drop table usuarios;

Si intentamos eliminar una tabla que no existe, aparece un mensaje de error indicando tal situación y la sentencia no se ejecuta. Para evitar este mensaje podemos agregar a la instrucción lo siguiente:

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

drop table usuarios;

En la sentencia precedente especificamos que elimine la tabla "usuarios" si existe.

Hay que tener en cuenta que este NO es el procedimiento habitual, ya que las tablas usualmente no se borran nunca, ya que perderíamos la información que contienen. En este caso lo haremos para asegurarnos unos resultados en nuestras pruebas de aprendizaje.

**Ejercicio:**

Vamos a crear una tabla llamada "usuarios". En primer lugar vamos a eliminar la tabla "usuarios" averiguando si existe (a esto vamos a repetirlo siempre porque puede haber otro usuario que haya creado una tabla con el mismo nombre):

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

drop table usuarios;

Recordar que debemos finalizar cada comando con un punto y coma.

La tabla "usuarios" contendrá los siguientes campos:

- nombre: varchar de 30 caracteres de longitud,

- clave: varchar de 10 caracteres de longitud.

Ahora sí creamos la tabla:

create table usuarios (

nombre varchar(30),

clave varchar(10)

);

Aparece un mensaje indicando que el comando se completó exitosamente.

Veamos las tablas existentes:

sp\_tables @table\_owner='dbo';

Veamos la estructura de la tabla "usuarios":

sp\_columns usuarios;

aparece mucha información que no analizaremos en detalle, como el nombre de la tabla, su propietario, los campos y sus tipos de datos, su longitud, etc.:

...COLUMN\_NAME TYPE\_NAME LENGHT...

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

nombre varchar 30

clave varchar 10

Intentemos crear una tabla con el mismo nombre, mostrará un mensaje indicando que ya hay un objeto llamado 'usuarios' en la base de datos y la sentencia no se ejecutará:

create table usuarios (

nombre varchar(30),

clave varchar(10)

);

Eliminemos la tabla:

drop table usuarios;

Verifiquemos si se ha eliminado:

sp\_tables @table\_owner='dbo';

No debe aparecer la tabla "usuarios".

**Código:**

use PruebaSQL

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

drop table usuarios;

create table usuarios (

nombre varchar(30),

clave varchar(10)

);

sp\_tables @table\_owner='dbo';

sp\_columns usuarios;

drop table usuarios;

sp\_tables @table\_owner='dbo';

**Problema:**

Necesita almacenar los datos de sus amigos en una tabla. Los datos que guardará serán: apellido, nombre, domicilio y teléfono.

1- Elimine la tabla "agenda" si existe:

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

drop table agenda;

2- Intente crear una tabla llamada "/agenda":

create table /agenda(

apellido varchar(30),

nombre varchar(20),

domicilio varchar(30),

telefono varchar(11)

);

aparece un mensaje de error porque usamos un caracter inválido ("/") para el nombre.

3- Cree una tabla llamada "agenda", debe tener los siguientes campos: apellido, varchar(30); nombre, varchar(20); domicilio, varchar (30) y telefono, varchar(11):

create table agenda(

apellido varchar(30),

nombre varchar(20),

domicilio varchar(30),

telefono varchar(11)

);

4- Intente crearla nuevamente. Aparece mensaje de error.

5- Visualice las tablas existentes (sp\_tables @table\_owner='dbo').

6- Visualice la estructura de la tabla "agenda" (sp\_columns).

7- Elimine la tabla.

8- Intente eliminar la tabla, sin controlar si existe. Debe aparecer un mensaje de error.

**Solución:**

use PruebaSQL

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

drop table agenda;

create table /agenda(

apellido varchar(30),

nombre varchar(20),

domicilio varchar(30),

telefono varchar(11)

);

create table agenda(

apellido varchar(30),

nombre varchar(20),

domicilio varchar(30),

telefono varchar(11)

);

create table agenda(

apellido varchar(30),

nombre varchar(20),

domicilio varchar(30),

telefono varchar(11)

);

sp\_tables @table\_owner='dbo';

sp\_columns agenda;

drop table agenda;

drop table agenda;

## Insertar y recuperar registros de una tabla (insert into - select)

Un registro es una fila de la tabla que contiene los datos propiamente dichos. Cada registro tiene un dato por cada columna (campo). Nuestra tabla "usuarios" consta de 2 campos, "nombre" y "clave".

Al ingresar los datos de cada registro debe tenerse en cuenta la cantidad y el orden de los campos.

La sintaxis básica y general es la siguiente:

insert into NOMBRETABLA (NOMBRECAMPO1, ..., NOMBRECAMPOn)

values (VALORCAMPO1, ..., VALORCAMPOn);

Usamos "insert into", luego el nombre de la tabla, detallamos los nombres de los campos entre paréntesis y separados por comas y luego de la cláusula "values" colocamos los valores para cada campo, también entre paréntesis y separados por comas.

Para agregar un registro a la tabla tipeamos:

insert into usuarios (nombre, clave) values ('José','miclave');

Note que los datos ingresados, como corresponden a cadenas de caracteres se colocan entre comillas simples.

Para ver los registros de una tabla usamos "select":

select nombre,clave from usuarios;

El comando "select" recupera los registros de una tabla.

En lugar de la lista de campos podremos usar un asterisco (\*) con el asterisco indicamos que muestre todos los campos de la tabla "usuarios". Esto no es conveniente ya que hacemos trabajar más a la Base de datos, teniendo que comunicar no sólo los resultados sino la estructura de la tabla.

Es importante ingresar los valores en el mismo orden en que se nombran los campos:

insert into usuarios (clave, nombre) values ('Codigo112','Juan');

En el ejemplo anterior se nombra primero el campo "clave" y luego el campo "nombre" por eso, los valores también se colocan en ese orden.

Si ingresamos los datos en un orden distinto al orden en que se nombraron los campos, no aparece un mensaje de error y los datos se guardan de modo incorrecto.

En el siguiente ejemplo se colocan los valores en distinto orden en que se nombran los campos, el valor de la clave (la cadena "Codigo445") se guardará en el campo "nombre" y el valor del nombre (la cadena "Luis") en el campo "clave":

insert into usuarios (nombre,clave) values ('Codigo445','Luis');

**Ejercicio:**

Vamos a crear una tabla llamada "usuarios". En primer lugar vamos a eliminar la tabla "usuarios" averiguando si existe (recuerde que debemos repetir siempre esto para conseguir que los ejemplos nos funcionen correctamente). Esto nunca se usará en la práctica salvo para tablas intermedias que queramos generar con resultados parciales cuando los necesitemos. Los datos son lo más importante y el uso del drop es muy peligroso.

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

drop table usuarios;

Recordar que debemos finalizar cada comando con un punto y coma.

Creamos la tabla:

create table usuarios(

nombre varchar(30),

clave varchar(10)

);

Agregamos un registro a la tabla:

insert into usuarios (nombre, clave)

values ('Juan','miclave');

Veamos si el registro se guardó:

select nombre,clave from usuarios;

Ingresemos otro registro alterando el orden de los campos:

insert into usuarios (clave, nombre)

values ('Codigo112','Juan');

Veamos cómo SQL Server almacenó los datos:

select nombre,clave from usuarios;

Ingresemos otro registro colocando los valores en distinto orden en que se nombran los campos:

insert into usuarios (nombre,clave)

values ('Codigo445','Luis');

Veamos cómo se guardaron los datos:

select nombre,clave from usuarios;

Note que la cadena "Codigo445" se almacenó en el campo "nombre" y la cadena "Luis" en el campo "clave".

**Código:**

use PruebaSQL

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

drop table usuarios;

create table usuarios(

nombre varchar(30),

clave varchar(10)

);

insert into usuarios (nombre, clave) values ('Juan','miclave');

select nombre,clave from usuarios;

insert into usuarios (clave, nombre) values ('Codigo112','Juan');

select nombre,clave from usuarios;

insert into usuarios (nombre,clave) values ('Codigo445','Luis');

select nombre,clave from usuarios;

**Problema:**

Trabaje con la tabla "agenda" que almacena información de sus amigos.

1- Elimine la tabla "agenda", si existe:

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

drop table agenda;

2- Cree una tabla llamada "agenda". Debe tener los siguientes campos: apellido (cadena de 30),

nombre (cadena de 20), domicilio (cadena de 30) y telefono (cadena de 11):

3- Visualice las tablas existentes para verificar la creación de "agenda"

(sp\_tables @table\_owner='dbo').

4- Visualice la estructura de la tabla "agenda" (sp\_columns).

5- Ingrese los siguientes registros:

insert into agenda (apellido, nombre, domicilio, telefono)

values ('Moreno','Alberto','Colon 123','4234567');

insert into agenda (apellido,nombre, domicilio, telefono)

values ('Torres','Juan','Avellaneda 135','4458787');

6- Seleccione todos los registros de la tabla:

select apellido, nombre, domicilio, telefono from agenda;

7- Elimine la tabla "agenda":

drop table agenda;

8- Intente eliminar la tabla nuevamente (aparece un mensaje de error):

drop table agenda;

**Solución:**

use PruebaSQL

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

drop table agenda;

create table agenda(

apellido varchar(30),

nombre varchar(20),

domicilio varchar(30),

telefono varchar(11)

);

sp\_tables @table\_owner='dbo';

sp\_columns agenda;

insert into agenda (apellido, nombre, domicilio, telefono)

values ('Moreno','Alberto','Colon 123','4234567');

insert into agenda (apellido,nombre, domicilio, telefono)

values ('Torres','Juan','Avellaneda 135','4458787');

select apellido, nombre, domicilio, telefono from agenda;

drop table agenda;

drop table agenda;

## Tipos de datos básicos

Ya explicamos que al crear una tabla debemos resolver qué campos (columnas) tendrá y qué tipo de datos almacenará cada uno de ellos, es decir, su estructura.

El tipo de dato especifica el tipo de información que puede guardar un campo: caracteres, números, etc.

Estos son algunos tipos de datos básicos de SQL Server (posteriormente veremos otros):

**varchar**: se usa para almacenar cadenas de caracteres. Una cadena es una secuencia de caracteres. Se coloca entre comillas (simples); ejemplo: 'Hola', 'Juan Perez'. El tipo "varchar" define una cadena de longitud variable en la cual determinamos el máximo de caracteres entre paréntesis. Puede guardar hasta 8000 caracteres. Por ejemplo, para almacenar cadenas de hasta 30 caracteres, definimos un campo de tipo varchar(30), es decir, entre paréntesis, junto al nombre del campo colocamos la longitud.

Si asignamos una cadena de caracteres de mayor longitud que la definida, la cadena no se carga, aparece un mensaje indicando tal situación y la sentencia no se ejecuta.

Por ejemplo, si definimos un campo de tipo varchar(10) e intentamos asignarle la cadena 'Buenas tardes', aparece un mensaje de error y la sentencia no se ejecuta. Varchar(max) indica una cadena que permite almacenar la longitud máxima posible.

**integer**: se usa para guardar valores numéricos enteros, de -2000000000 a 2000000000 aprox. Definimos campos de este tipo cuando queremos representar, por ejemplo, cantidades.

**float**: se usa para almacenar valores numéricos con decimales con redondeo. Se utiliza como separador el punto (.). Definimos campos de este tipo para números científicos con cálculos complejos, teniendo muy en cuenta que se puede producir redondeos. Cuidado con los redondeos en este tipo de dato. Una columna de datos float puede almacenar el rango de valores -1,79x-10308 a 1,79x-10308, si la definimos con el valor máxmo de precisión. La precisión puede variar entre 1 y 53.

**decimal**: se usa para almacenar números con decimales con precisión exacta. Se le indica el número de dígitos incluyendo la coma y el número de decimales incluidos en el total. Por ejemplo decimal(6,2) tendría seis fígitos en total, tres antes de la coma, la coma y dos decimales.

Antes de crear una tabla debemos pensar en sus campos y optar por el tipo de dato adecuado para cada uno de ellos. Por ejemplo, si en un campo almacenaremos números enteros, el tipo "decimal" sería una mala elección; si vamos a guardar precios, el tipo "decimal" es más adecuado, no así "integer" que no tiene decimales. Otro ejemplo, si en un campo vamos a guardar un número telefónico o un número de documento, usamos "varchar", no "integer" porque si bien son dígitos, con ellos no realizamos operaciones matemáticas.

**Ejercicio:**

Vamos a crear una tabla llamada "libros". En primer lugar vamos a eliminar la tabla "libros" averiguando si existe:

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

drop table libros;

Para almacenar información de los libros de una librería necesitamos los siguientes campos:

-titulo, cadena de caracteres de 20 de longitud,

-autor, cadena de caracteres de 15 de longitud,

-editorial, caracteres de 10 de longitud,

-precio, valor numérico seis dígitos de ellos 2 decimales,

-cantidad, valor numérico entero.

Al crear la tabla, entonces, elegimos el tipo de dato más adecuado para cada campo:

create table libros(

titulo varchar(20),

autor varchar(15),

editorial varchar(10),

precio decimal(7,2),

cantidad integer

);

Note que al especificar el tipo de dato de los campos numéricos, no colocamos entre paréntesis la longitud.

Vemos la estructura de la tabla:

sp\_columns libros;

Aparece la siguiente información:

...COLUMN\_NAME TYPE\_NAME LENGHT

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

titulo varchar 20

autor varchar 15

editorial varchar 10

precio decimal 9

cantidad int 4

Ingresamos algunos registros:

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('El aleph','Borges','Emece',25.50,100);

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('Matematica estas ahi','Paenza','Siglo XXI',18.8,200);

Note que al ingresar valores numéricos (decimal, float e integer) no se utilizan comillas y para el separador de decimales se usa el caracter punto(.).

Veamos los registros cargados:

select titulo,autor,editorial,precio,cantidad from libros;

Veamos lo que sucede si intentamos ingresar para el campo "titulo" una cadena de más de 20 caracteres:

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('Alicia en el pais de las maravillas','Lewis Carroll','Atlantida',10,200);

Aparece un mensaje de error y la sentencia no se ejecuta.

Vamos a cortar la cadena para que SQL Server acepte el ingreso del registro:

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('Alicia en el pais','Lewis Carroll','Atlantida',10,200);

Veamos los registros cargados:

select titulo,autor,editorial,precio,cantidad from libros;

**Código:**

use PruebaSQL

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

drop table libros;

create table libros(

titulo varchar(20),

autor varchar(15),

editorial varchar(10),

precio decimal(7,2),

cantidad integer

);

sp\_columns libros;

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('El aleph','Borges','Emece',25.50,100);

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('Matematica estas ahi','Paenza','Siglo XXI',18.8,200);

select titulo,autor,editorial,precio,cantidad from libros;

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('Alicia en el pais de las maravillas','Lewis Carroll','Atlantida',10,200);

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('Alicia en el pais','Lewis Carroll','Atlantida',10,200);

select titulo,autor,editorial,precio,cantidad from libros;

**Problema:**

Un videoclub que alquila películas en video almacena la información de sus películas en una tabla llamada "peliculas"; para cada película necesita los siguientes datos:

-nombre, cadena de caracteres de 20 de longitud,

-actor, cadena de caracteres de 20 de longitud,

-duración, valor numérico entero.

-cantidad de copias: valor entero.

1- Elimine la tabla, si existe:

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

drop table peliculas;

2- Cree la tabla eligiendo el tipo de dato adecuado para cada campo:

create table peliculas(

nombre varchar(20),

actor varchar(20),

duracion integer,

cantidad integer

);

3- Vea la estructura de la tabla:

4- Ingrese los siguientes registros:

insert into peliculas (nombre, actor, duracion, cantidad)

values ('Mision imposible','Tom Cruise',128,3);

insert into peliculas (nombre, actor, duracion, cantidad)

values ('Mision imposible 2','Tom Cruise',130,2);

insert into peliculas (nombre, actor, duracion, cantidad)

values ('Mujer bonita','Julia Roberts',118,3);

insert into peliculas (nombre, actor, duracion, cantidad)

values ('Elsa y Fred','China Zorrilla',110,2);

5- Muestre todos los registros.

**Solución:**

use PruebaSQL

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

drop table peliculas;

create table peliculas(

nombre varchar(20),

actor varchar(20),

duracion integer,

cantidad integer

);

sp\_columns peliculas;

insert into peliculas (nombre, actor, duracion, cantidad)

values ('Mision imposible','Tom Cruise',128,3);

insert into peliculas (nombre, actor, duracion, cantidad)

values ('Mision imposible 2','Tom Cruise',130,2);

insert into peliculas (nombre, actor, duracion, cantidad)

values ('Mujer bonita','Julia Roberts',118,3);

insert into peliculas (nombre, actor, duracion, cantidad)

values ('Elsa y Fred','China Zorrilla',110,2);

select nombre, actor, duracion, cantidad from peliculas;

## Recuperar algunos campos (select)

Hemos aprendido cómo ver todos los registros de una tabla, empleando la instrucción "select".

La sintaxis básica y general es la siguiente:

select lista de campos separados por comas from NOMBRETABLA;

Podemos usar un asterisco (\*) en la lista de campos lo que indica que se seleccionan todos los campos de la tabla, aunque esto hace que la consulta sea menos eficiente.

select titulo,autor from libros;

La lista de campos luego del "select" selecciona los datos correspondientes a los campos nombrados. En el ejemplo anterior seleccionamos los campos "titulo" y "autor" de la tabla "libros", mostrando todos los registros. Los datos aparecen ordenados según la lista de selección, en dicha lista los nombres de los campos se separan con comas.

**Ejercicio:**

Trabajamos con la tabla "libros" que almacena los datos de los libros de una librería.

Eliminamos la tabla, si existe:

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

drop table libros;

Creamos la tabla:

create table libros(

titulo varchar(40),

autor varchar(30),

editorial varchar(15),

precio decimal(7,2),

cantidad integer

);

Veamos la estructura de la tabla:

sp\_columns libros;

Ingresamos algunos registros:

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('El aleph','Borges','Emece',25.50,100);

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('Alicia en el pais de las maravillas','Lewis Carroll','Atlantida',10,200);

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('Matematica estas ahi','Paenza','Siglo XXI',18.8,200);

Veamos todos los campos la tabla:

select \* from libros;

Veamos solamente el título, autor y editorial de todos los libros especificando los nombres de los campos separados por comas:

select titulo,autor,editorial from libros;

Con la siguiente sentencia seleccionamos los títulos y precios de todos los libros:

select titulo,precio from libros;

Veamos solamente la editorial y la cantidad de libros tipeamos:

select editorial,cantidad from libros;

**Código:**

use PruebaSQL

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

drop table libros;

create table libros(

titulo varchar(40),

autor varchar(30),

editorial varchar(15),

precio decimal(7,2),

cantidad integer

);

sp\_columns libros;

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('El aleph','Borges','Emece',25.50,100);

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('Alicia en el pais de las maravillas','Lewis Carroll','Atlantida',10,200);

insert into libros (titulo,autor,editorial,precio,cantidad)

values ('Matematica estas ahi','Paenza','Siglo XXI',18.8,200);

select \* from libros;

select titulo,autor,editorial from libros;

select titulo,precio from libros;

select editorial,cantidad from libros;

**Problema:**

Un videoclub que alquila películas en video almacena la información de sus películas en alquiler en una tabla llamada "peliculas".

1- Elimine la tabla, si existe:

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

drop table peliculas;

2- Cree la tabla:

create table peliculas(

titulo varchar(20),

actor varchar(20),

duracion integer,

cantidad integer

);

3- Vea la estructura de la tabla (sp\_columns).

4- Ingrese alos siguientes registros:

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Mision imposible','Tom Cruise',180,3);

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Mision imposible 2','Tom Cruise',190,2);

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Mujer bonita','Julia Roberts',118,3);

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Elsa y Fred','China Zorrilla',110,2);

5- Realice un "select" mostrando solamente el título y actor de todas las películas

6- Muestre el título y duración de todas las peliculas

7- Muestre el título y la cantidad de copias

**Solución:**

use PruebaSQL

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

drop table peliculas;

create table peliculas(

titulo varchar(20),

actor varchar(20),

duracion integer,

cantidad integer

);

sp\_columns peliculas;

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Mision imposible','Tom Cruise',180,3);

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Mision imposible 2','Tom Cruise',190,2);

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Mujer bonita','Julia Roberts',118,3);

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Elsa y Fred','China Zorrilla',110,2);

select titulo,actor from peliculas;

select titulo,duracion from peliculas;

select titulo,cantidad from peliculas;

## Recuperar algunos registros (where)

Hemos aprendido a seleccionar algunos campos de una tabla.

También es posible recuperar algunos registros.

Existe una cláusula, "where" con la cual podemos especificar condiciones para una consulta "select". Es decir, podemos recuperar algunos registros, sólo los que cumplan con ciertas condiciones indicadas con la cláusula "where". Por ejemplo, queremos ver el usuario cuyo nombre es "Marcelo", para ello utilizamos "where" y luego de ella, la condición:

select nombre, clave

from usuarios

where nombre='Marcelo';

La sintaxis básica y general es la siguiente:

select NOMBRECAMPO1, ..., NOMBRECAMPOn

from NOMBRETABLA

where CONDICION;

Para las condiciones se utilizan operadores relacionales (tema que trataremos más adelante en detalle). El signo igual(=) es un operador relacional.

Para la siguiente selección de registros especificamos una condición que solicita los usuarios cuya clave es igual a "River":

select nombre,clave

from usuarios

where clave='River';

Si ningún registro cumple la condición establecida con el "where", no aparecerá ningún registro.

Entonces, con "where" establecemos condiciones para recuperar algunos registros.

Para recuperar algunos campos de algunos registros combinamos en la consulta la lista de campos y la cláusula "where":

select nombre

from usuarios

where clave='River';

En la consulta anterior solicitamos el nombre de todos los usuarios cuya clave sea igual a "River".

**Ejercicio:**

Trabajamos con la tabla "usuarios" que consta de 2 campos: nombre de usuario y clave.

Eliminamos la tabla, si existe:

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

drop table usuarios;

Creamos la tabla:

create table usuarios (

nombre varchar(30),

clave varchar(10)

);

Vemos la estructura de la tabla:

sp\_columns usuarios;

Ingresamos algunos registros:

insert into usuarios (nombre, clave)

values ('Marcelo','Boca');

insert into usuarios (nombre, clave)

values ('JuanPerez','Juancito');

insert into usuarios (nombre, clave)

values ('Susana','River');

insert into usuarios (nombre, clave)

values ('Luis','River');

Realizamos una consulta especificando una condición, queremos ver el usuario cuyo nombre es "Leonardo":

select nombre, clave from usuarios

where nombre='Leonardo';

Queremos ver el nombre de los usuarios cuya clave es "River":

select nombre from usuarios

where clave='River';

Realizamos un "select" de los nombres de los usuarios cuya clave es "Santi":

select nombre from usuarios

where clave='Santi';

No se muestra ningún registro ya que ninguno cumple la condición.

**Código:**

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

drop table usuarios;

create table usuarios (

nombre varchar(30),

clave varchar(10)

);

sp\_columns usuarios;

insert into usuarios (nombre, clave)

values ('Marcelo','Boca');

insert into usuarios (nombre, clave)

values ('JuanPerez','Juancito');

insert into usuarios (nombre, clave)

values ('Susana','River');

insert into usuarios (nombre, clave)

values ('Luis','River');

select nombre, clave from usuarios

where nombre='Leonardo';

select nombre from usuarios

where clave='River';

select nombre from usuarios

where clave='Santi';

**Problema:**

Trabaje con la tabla "agenda" en la que registra los datos de sus amigos.

1- Elimine "agenda", si existe:

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

drop table agenda;

2- Cree la tabla, con los siguientes campos: apellido (cadena de 30), nombre (cadena de 20),

domicilio (cadena de 30) y telefono (cadena de 11).

3- Visualice la estructura de la tabla "agenda".

4- Ingrese los siguientes registros:

Acosta, Ana, Colon 123, 4234567;

Bustamante, Betina, Avellaneda 135, 4458787;

Lopez, Hector, Salta 545, 4887788;

Lopez, Luis, Urquiza 333, 4545454;

Lopez, Marisa, Urquiza 333, 4545454.

5- Seleccione todos los registros de la tabla

6- Seleccione el registro cuyo nombre sea "Marisa" (1 registro)

7- Seleccione los nombres y domicilios de quienes tengan apellido igual a "Lopez" (3 registros)

8- Muestre el nombre de quienes tengan el teléfono "4545454" (2 registros)

**Solución:**

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

drop table agenda;

create table agenda (

apellido varchar(30),

nombre varchar(20),

domicilio varchar(30),

telefono varchar(11)

);

sp\_columns agenda;

insert into agenda(apellido,nombre,domicilio,telefono) values

('Acosta', 'Ana', 'Colon 123', '4234567');

insert into agenda(apellido,nombre,domicilio,telefono) values

('Bustamante', 'Betina', 'Avellaneda 135', '4458787');

insert into agenda(apellido,nombre,domicilio,telefono) values

('Lopez', 'Hector', 'Salta 545', '4887788');

insert into agenda(apellido,nombre,domicilio,telefono) values

('Lopez', 'Luis', 'Urquiza 333', '4545454');

insert into agenda(apellido,nombre,domicilio,telefono) values

('Lopez', 'Marisa', 'Urquiza 333', '4545454');

select apellido,nombre,domicilio,telefono from agenda;

select apellido,nombre,domicilio,telefono from agenda

where nombre='Marisa';

select nombre,domicilio from agenda

where apellido='Lopez';

select nombre from agenda

where telefono='4545454';

## Operadores relacionales

Los operadores son símbolos que permiten realizar operaciones matemáticas, concatenar cadenas, hacer comparaciones.

SQL Server tiene 4 tipos de operadores:

relacionales (o de comparación)

aritméticos

de concatenación

lógicos.

Por ahora veremos solamente los primeros.

Los operadores relacionales (o de comparación) nos permiten comparar dos expresiones, que pueden ser variables, valores de campos, etc.

Hemos aprendido a especificar condiciones de igualdad para seleccionar registros de una tabla; por ejemplo:

select titulo,autor,editorial,precio,cantidad from libros

where autor='Borges';

Utilizamos el operador relacional de igualdad.

Los operadores relacionales vinculan un campo con un valor para que SQL Server compare cada registro (el campo especificado) con el valor dado.

Los operadores relacionales son los siguientes:

= igual

<> distinto

> mayor

< menor

>= mayor o igual

<= menor o igual

Podemos seleccionar los registros cuyo autor sea diferente de "Borges", para ello usamos la condición:

select titulo,autor,editorial,precio,cantidad from libros

where autor<>'Borges';

Podemos comparar valores numéricos. Por ejemplo, queremos mostrar los títulos y precios de los libros cuyo precio sea mayor a 20 pesos:

select titulo, precio

from libros

where precio>20;

Queremos seleccionar los libros cuyo precio sea menor o igual a 30:

select titulo,autor,editorial,precio,cantidad from libros

where precio<=30;

Los operadores relacionales comparan valores del mismo tipo. Se emplean para comprobar si un campo cumple con una condición.

No son los únicos, existen otros que veremos más adelante.

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla "libros", si existe:

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

drop table libros;

La creamos con la siguiente estructura:

create table libros(

titulo varchar(30),

autor varchar(30),

editorial varchar(15),

precio decimal(7,2),

);

Agregamos registros a la tabla:

insert into libros (titulo,autor,editorial,precio)

values ('El aleph','Borges','Emece',24.50);

insert into libros (titulo,autor,editorial,precio)

values ('Martin Fierro','Jose Hernandez','Emece',16.00);

insert into libros (titulo,autor,editorial,precio)

values ('Aprenda PHP','Mario Molina','Emece',35.40);

insert into libros (titulo,autor,editorial,precio)

values ('Cervantes y el quijote','Borges','Paidos',50.90);

Seleccionamos los registros cuyo autor sea diferente de 'Borges':

select titulo,autor,editorial,precio,cantidad from libros

where autor<>'Borges';

Seleccionamos los registros cuyo precio supere los 20 pesos, sólo el título y precio:

select titulo,precio

from libros

where precio>20;

Note que el valor con el cual comparamos el campo "precio", como es numérico, no se coloca entre comillas. Los libros cuyo precio es menor o igual a 20 pesos no aparecen en la selección.

Recuperamos aquellos libros cuyo precio es menor o igual a 30:

select titulo,autor,editorial,precio,cantidad from libros

where precio<=30;

**Código:**

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

drop table libros;

create table libros(

titulo varchar(30),

autor varchar(30),

editorial varchar(15),

precio float

);

insert into libros (titulo,autor,editorial,precio)

values ('El aleph','Borges','Emece',24.50);

insert into libros (titulo,autor,editorial,precio)

values ('Martin Fierro','Jose Hernandez','Emece',16.00);

insert into libros (titulo,autor,editorial,precio)

values ('Aprenda PHP','Mario Molina','Emece',35.40);

insert into libros (titulo,autor,editorial,precio)

values ('Cervantes y el quijote','Borges','Paidos',50.90);

select titulo,autor,editorial,precio,cantidad from libros

where autor<>'Borges';

select titulo,precio

from libros

where precio>20;

select titulo,autor,editorial,precio,cantidad from libros

where precio<=30;

**Problema:**

Un comercio que vende artículos de computación registra los datos de sus artículos en una tabla con ese nombre.

1- Elimine "articulos", si existe:

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

drop table articulos;

2- Cree la tabla, con la siguiente estructura:

create table articulos(

codigo integer,

nombre varchar(20),

descripcion varchar(30),

precio decimal(7,2),

cantidad integer

);

3- Vea la estructura de la tabla (sp\_columns).

4- Ingrese algunos registros:

insert into articulos (codigo, nombre, descripcion, precio,cantidad)

values (1,'impresora','Epson Stylus C45',400.80,20);

insert into articulos (codigo, nombre, descripcion, precio,cantidad)

values (2,'impresora','Epson Stylus C85',500,30);

insert into articulos (codigo, nombre, descripcion, precio,cantidad)

values (3,'monitor','Samsung 14',800,10);

insert into articulos (codigo, nombre, descripcion, precio,cantidad)

values (4,'teclado','ingles Biswal',100,50);

insert into articulos (codigo, nombre, descripcion, precio,cantidad)

values (5,'teclado','español Biswal',90,50);

5- Seleccione los datos de las impresoras (2 registros)

6- Seleccione los artículos cuyo precio sea mayor o igual a 400 (3 registros)

7- Seleccione el código y nombre de los artículos cuya cantidad sea menor a 30 (2 registros)

8- Selecciones el nombre y descripción de los artículos que NO cuesten $100 (4 registros)

**Solución:**

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

drop table articulos;

create table articulos(

codigo integer,

nombre varchar(20),

descripcion varchar(30),

precio float,

cantidad integer

);

sp\_columns articulos;

insert into articulos (codigo, nombre, descripcion, precio,cantidad)

values (1,'impresora','Epson Stylus C45',400.80,20);

insert into articulos (codigo, nombre, descripcion, precio,cantidad)

values (2,'impresora','Epson Stylus C85',500,30);

insert into articulos (codigo, nombre, descripcion, precio,cantidad)

values (3,'monitor','Samsung 14',800,10);

insert into articulos (codigo, nombre, descripcion, precio,cantidad)

values (4,'teclado','ingles Biswal',100,50);

insert into articulos (codigo, nombre, descripcion, precio,cantidad)

values (5,'teclado','español Biswal',90,50);

select codigo, nombre, descripcion, precio,cantidad from articulos

where nombre='impresora';

select codigo, nombre, descripcion, precio,cantidad from articulos

where precio>=400;

select codigo,nombre

from articulos

where cantidad<30;

select nombre, descripcion

from articulos

where precio<>100;

**Segundo problema:**

Un video club que alquila películas en video almacena la información de sus películas en alquiler en una tabla denominada "peliculas".

1- Elimine la tabla, si existe.

2- Cree la tabla eligiendo el tipo de dato adecuado para cada campo:

create table peliculas(

titulo varchar(20),

actor varchar(20),

duracion integer,

cantidad integer

);

3- Ingrese los siguientes registros:

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Mision imposible','Tom Cruise',120,3);

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Mision imposible 2','Tom Cruise',180,4);

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Mujer bonita','Julia R.',90,1);

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Elsa y Fred','China Zorrilla',80,2);

4- Seleccione las películas cuya duración no supere los 90 minutos (2 registros)

5- Seleccione el título de todas las películas en las que el actor NO sea "Tom Cruise" (2

registros)

6- Muestre todos los campos, excepto "duracion", de todas las películas de las que haya más de 2

copias (2 registros)

**Solución**

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

drop table peliculas;

create table peliculas(

titulo varchar(20),

actor varchar(20),

duracion integer,

cantidad integer

);

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Mision imposible','Tom Cruise',120,3);

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Mision imposible 2','Tom Cruise',180,4);

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Mujer bonita','Julia R.',90,1);

insert into peliculas (titulo, actor, duracion, cantidad)

values ('Elsa y Fred','China Zorrilla',80,2);

select titulo, actor, duracion, cantidad from peliculas

where duracion<=90;

select titulo, actor, duracion, cantidad from peliculas

where actor<>'Tom Cruise';

select titulo,actor,cantidad

from peliculas

where cantidad >2;

## Borrar registros (delete)

Para eliminar los registros de una tabla usamos el comando "delete":

delete from usuarios;

Muestra un mensaje indicando la cantidad de registros que ha eliminado.

Si no queremos eliminar todos los registros, sino solamente algunos, debemos indicar cuál o cuáles, para ello utilizamos el comando "delete" junto con la clausula "where" con la cual establecemos la condición que deben cumplir los registros a borrar.

Por ejemplo, queremos eliminar aquel registro cuyo nombre de usuario es "Marcelo":

delete from usuarios

where nombre='Marcelo';

Si solicitamos el borrado de un registro que no existe, es decir, ningún registro cumple con la condición especificada, ningún registro será eliminado.

Tenga en cuenta que si no colocamos una condición, se eliminan todos los registros de la tabla nombrada.

**Ejercicio:**

Trabajamos con la tabla "usuarios".

Eliminamos la tabla "usuarios", si existe:

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

drop table usuarios;

La creamos con la siguiente estructura:

create table usuarios(

nombre varchar(30),

clave varchar(10)

);

Agregamos registros a la tabla:

insert into usuarios (nombre,clave)

values ('Marcelo','River');

insert into usuarios (nombre,clave)

values ('Susana','chapita');

insert into usuarios (nombre,clave)

values ('CarlosFuentes','Boca');

insert into usuarios (nombre,clave)

values ('FedericoLopez','Boca');

Seleccionamos todos los registros:

select nombre,clave from usuarios;

Vamos a eliminar el registro cuyo nombre de usuario es "Marcelo":

delete from usuarios

where nombre='Marcelo';

Veamos el contenido de la tabla:

select nombre,clave from usuarios;

Intentamos eliminarlo nuevamente:

delete from usuarios

where nombre='Marcelo';

Veamos el contenido de la tabla:

select nombre,clave from usuarios;

Como ningún registro cumple con la condición no se borran registros.

Eliminamos todos los registros cuya clave es 'Boca':

delete from usuarios

where clave='Boca';

Veamos el contenido de la tabla:

select nombre,clave from usuarios;

Eliminemos todos los registros:

delete from usuarios;

Veamos el contenido de la tabla:

select nombre,clave from usuarios;

No hay registros.

**Código:**

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

drop table usuarios;

create table usuarios(

nombre varchar(30),

clave varchar(10)

);

insert into usuarios (nombre,clave)

values ('Marcelo','River');

insert into usuarios (nombre,clave)

values ('Susana','chapita');

insert into usuarios (nombre,clave)

values ('CarlosFuentes','Boca');

insert into usuarios (nombre,clave)

values ('FedericoLopez','Boca');

select nombre,clave from usuarios;

delete from usuarios

where nombre='Marcelo';

select nombre,clave from usuarios;

delete from usuarios

where nombre='Marcelo';

select nombre,clave from usuarios;

delete from usuarios

where clave='Boca';

select nombre,clave from usuarios;

delete from usuarios;

select nombre,clave from usuarios;

**Problema:**

Trabaje con la tabla "agenda" que registra la información referente a sus amigos.

1- Elimine la tabla si existe:

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

drop table agenda;

2- Cree la tabla con los siguientes campos: apellido (cadena de 30), nombre (cadena de 20), domicilio (cadena de 30) y telefono (cadena de 11):

create table agenda(

apellido varchar(30),

nombre varchar(20),

domicilio varchar(30),

telefono varchar(11)

);

3- Ingrese los siguientes registros (insert into):

Alvarez,Alberto,Colon 123,4234567,

Juarez,Juan,Avellaneda 135,4458787,

Lopez,Maria,Urquiza 333,4545454,

Lopez,Jose,Urquiza 333,4545454,

Salas,Susana,Gral. Paz 1234,4123456.

4- Elimine el registro cuyo nombre sea "Juan" (1 registro afectado)

5- Elimine los registros cuyo número telefónico sea igual a "4545454" (2 registros afectados):

6- Muestre la tabla.

7- Elimine todos los registros (2 registros afectados):

8- Muestre la tabla.

**Solución:**

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

drop table agenda;

create table agenda(

apellido varchar(30),

nombre varchar(20),

domicilio varchar(30),

telefono varchar(11)

);

insert into agenda (apellido,nombre,domicilio,telefono)

values('Alvarez','Alberto','Colon 123','4234567');

insert into agenda (apellido,nombre,domicilio,telefono)

values('Juarez','Juan','Avellaneda 135','4458787');

insert into agenda (apellido,nombre,domicilio,telefono)

values('Lopez','Maria','Urquiza 333','4545454');

insert into agenda (apellido,nombre,domicilio,telefono)

values('Lopez','Jose','Urquiza 333','4545454');

insert into agenda (apellido,nombre,domicilio,telefono)

values('Salas','Susana','Gral. Paz 1234','4123456');

delete from agenda

where nombre='Juan';

delete from agenda

where telefono='4545454';

select apellido,nombre,domicilio,telefono from agenda;

delete from agenda;

select apellido,nombre,domicilio,telefono from agenda;

## Actualizar registros (update)

Decimos que actualizamos un registro cuando modificamos alguno de sus valores.

Para modificar uno o varios datos de uno o varios registros utilizamos "update" (actualizar).

Por ejemplo, en nuestra tabla "usuarios", queremos cambiar los valores de todas las claves, por "RealMadrid":

update usuarios set clave='RealMadrid';

Utilizamos "update" junto al nombre de la tabla y "set" junto con el campo a modificar y su nuevo valor.

El cambio afectará a todos los registros.

Podemos modificar algunos registros, para ello debemos establecer condiciones de selección con "where".

Por ejemplo, queremos cambiar el valor correspondiente a la clave de nuestro usuario llamado "Federicolopez", queremos como nueva clave "Boca", necesitamos una condición "where" que afecte solamente a este registro:

update usuarios set clave='Boca'

where nombre='Federicolopez';

Si Microsoft SQL Server no encuentra registros que cumplan con la condición del "where", no se modifica ninguno.

Las condiciones no son obligatorias, pero si omitimos la cláusula "where", la actualización afectará a todos los registros.

También podemos actualizar varios campos en una sola instrucción:

update usuarios set nombre='Marceloduarte', clave='Marce'

where nombre='Marcelo';

Para ello colocamos "update", el nombre de la tabla, "set" junto al nombre del campo y el nuevo valor y separado por coma, el otro nombre del campo con su nuevo valor.

**Ejercicio:**

Trabajamos con la tabla "usuarios".

Eliminamos la tabla si existe:

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

drop table usuarios;

Creamos la tabla:

create table usuarios(

nombre varchar(20),

clave varchar(10)

);

Ingresamos algunos registros:

insert into usuarios (nombre,clave)

values ('Marcelo','River');

insert into usuarios (nombre,clave)

values ('Susana','chapita');

insert into usuarios (nombre,clave)

values ('Carlosfuentes','Boca');

insert into usuarios (nombre,clave)

values ('Federicolopez','Boca');

Cambiaremos los valores de todas las claves, por la cadena "123456":

update usuarios set clave='123456';

El cambio afectó a todos los registros, veámoslo:

select nombre,clave from usuarios;

Necesitamos cambiar el valor de la clave del usuario llamado "Federicolopez" por "Boca":

update usuarios set clave='Boca'

where nombre='Federicolopez';

Verifiquemos que la actualización se realizó:

select nombre,clave from usuarios;

Vimos que si Microsoft SQL Server no encuentra registros que cumplan con la condición no se modifican registros:

update usuarios set clave='payaso'

where nombre='JuanaJuarez';

Si vemos la tabla veremos que no ha cambiado:

select nombre,clave from usuarios;

Para actualizar varios campos en una sola instrucción empleamos:

update usuarios set nombre='Marceloduarte', clave='Marce'

where nombre='Marcelo';

Si vemos la tabla:

select nombre,clave from usuarios;

**Código:**

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

drop table usuarios;

create table usuarios(

nombre varchar(20),

clave varchar(10)

);

insert into usuarios (nombre,clave)

values ('Marcelo','River');

insert into usuarios (nombre,clave)

values ('Susana','chapita');

insert into usuarios (nombre,clave)

values ('Carlosfuentes','Boca');

insert into usuarios (nombre,clave)

values ('Federicolopez','Boca');

update usuarios set clave='123456';

select nombre,clave from usuarios;

update usuarios set clave='Boca'

where nombre='Federicolopez';

select nombre,clave from usuarios;

update usuarios set clave='payaso'

where nombre='JuanaJuarez';

select nombre,clave from usuarios;

update usuarios set nombre='Marceloduarte', clave='Marce'

where nombre='Marcelo';

select nombre,clave from usuarios;

**Problema:**

Trabaje con la tabla "agenda" que almacena los datos de sus amigos.

1- Elimine la tabla si existe:

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

drop table agenda;

2- Cree la tabla:

create table agenda(

apellido varchar(30),

nombre varchar(20),

domicilio varchar(30),

telefono varchar(11)

);

3- Ingrese los siguientes registros (1 registro actualizado):

insert into agenda (apellido,nombre,domicilio,telefono)

values ('Acosta','Alberto','Colon 123','4234567');

insert into agenda (apellido,nombre,domicilio,telefono)

values ('Juarez','Juan','Avellaneda 135','4458787');

insert into agenda (apellido,nombre,domicilio,telefono)

values ('Lopez','Maria','Urquiza 333','4545454');

insert into agenda (apellido,nombre,domicilio,telefono)

values ('Lopez','Jose','Urquiza 333','4545454');

insert into agenda (apellido,nombre,domicilio,telefono)

values ('Suarez','Susana','Gral. Paz 1234','4123456');

4- Modifique el registro cuyo nombre sea "Juan" por "Juan Jose" (1 registro afectado)

5- Actualice los registros cuyo número telefónico sea igual a "4545454" por "4445566"

(2 registros afectados)

6- Actualice los registros que tengan en el campo "nombre" el valor "Juan" por "Juan Jose" (ningún

registro afectado porque ninguno cumple con la condición del "where")

7 - Luego de cada actualización ejecute un select que muestre todos los registros de la tabla.

**Solución:**

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

drop table agenda;

create table agenda(

apellido varchar(30),

nombre varchar(20),

domicilio varchar(30),

telefono varchar(11)

);

insert into agenda (apellido,nombre,domicilio,telefono)

values ('Acosta','Alberto','Colon 123','4234567');

insert into agenda (apellido,nombre,domicilio,telefono)

values ('Juarez','Juan','Avellaneda 135','4458787');

insert into agenda (apellido,nombre,domicilio,telefono)

values ('Lopez','Maria','Urquiza 333','4545454');

insert into agenda (apellido,nombre,domicilio,telefono)

values ('Lopez','Jose','Urquiza 333','4545454');

insert into agenda (apellido,nombre,domicilio,telefono)

values ('Suarez','Susana','Gral. Paz 1234','4123456');

select apellido,nombre,domicilio,telefono from agenda;

update agenda set nombre='Juan Jose'

where nombre='Juan';

select apellido,nombre,domicilio,telefono from agenda;

update agenda set telefono='4445566'

where telefono='4545454';

select apellido,nombre,domicilio,telefono from agenda;

update agenda set nombre='Juan Jose'

where nombre='Juan';

select apellido,nombre,domicilio,telefono from agenda;

## Comentarios

Para aclarar algunas instrucciones, en ocasiones, necesitamos agregar comentarios.

Es posible ingresar comentarios en la línea de comandos, es decir, un texto que no se ejecuta; para ello se emplean dos guiones (--) al comienzo de la línea:

select titulo, autor from libros --mostramos los registros de libros;

En la línea anterior, todo lo que está luego de los guiones (hacia la derecha) no se ejecuta.

Para agregar varias líneas de comentarios, se coloca una barra seguida de un asterisco (/\*) al comienzo del bloque de comentario y al finalizarlo, un asterisco seguido de una barra (\*/).

select titulo, autor

/\*mostramos títulos y

nombres de los autores\*/

from libros;

Todo lo que está entre los símbolos "/\*" y "\*/" no se ejecuta.

**Ejercicio:**

Trabajamos con la tabla "libros" que almacena los datos de los libros de una librería.

Eliminamos la tabla, si existe:

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

drop table libros;

Creamos la tabla:

create table libros(

titulo varchar(30),

autor varchar(20),

editorial varchar(15)

);

Agregamos un registro:

insert into libros (titulo,autor,editorial)

values ('El aleph','Borges','Emece');

Mostramos todos los libros y agregamos un comentario de linea:

select titulo,autor,editorial from libros --mostramos los registros de libros;

Vamos a mostrar el título y autor de todos los libros y agregamos un bloque de comentarios:

select titulo, autor

/\*mostramos títulos y

nombres de los autores\*/

from libros;

Note que lo que está entre los símbolos no se ejecuta.

**Código:**

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

drop table libros;

create table libros(

titulo varchar(30),

autor varchar(20),

editorial varchar(15)

);

insert into libros (titulo,autor,editorial)

values ('El aleph','Borges','Emece');

select titulo,autor,editorial from libros --mostramos los registros de libros;

select titulo, autor

/\*mostramos títulos y

nombres de los autores\*/

from libros;

**Problema:**

Usar los comentarios para explicar un ejercicio anterior.

## Valores null (is null)

**null** significa "dato desconocido" o "valor inexistente". No es lo mismo que un valor "0", una cadena vacía o una cadena literal "null".

A veces, puede desconocerse o no existir el dato correspondiente a algún campo de un registro. En estos casos decimos que el campo puede contener valores nulos.

Por ejemplo, en nuestra tabla de libros, podemos tener valores nulos en el campo "precio" porque es posible que para algunos libros no le hayamos establecido el precio para la venta.

En contraposición, tenemos campos que no pueden estar vacíos jamás.

Veamos un ejemplo. Tenemos nuestra tabla "libros". El campo "titulo" no debería estar vacío nunca, igualmente el campo "autor". Para ello, al crear la tabla, debemos especificar que dichos campos no admitan valores nulos:

create table libros(

titulo varchar(30) not null,

autor varchar(20) not null,

editorial varchar(15) null,

precio decimal(7,2)

);

Para especificar que un campo no admita valores nulos, debemos colocar "not null" luego de la definición del campo.

En el ejemplo anterior, los campos "editorial" y "precio" si admiten valores nulos.

Cuando colocamos "null" estamos diciendo que admite valores nulos (caso del campo "editorial"); por defecto, es decir, si no lo aclaramos, los campos permiten valores nulos (caso del campo "precio").

Si ingresamos los datos de un libro, para el cual aún no hemos definido el precio podemos colocar "null" para mostrar que no tiene precio:

insert into libros (titulo,autor,editorial,precio)

values('El aleph','Borges','Emece',null);

Note que el valor "null" no es una cadena de caracteres, no se coloca entre comillas.

Entonces, si un campo acepta valores nulos, podemos ingresar "null" cuando no conocemos el valor.

También podemos colocar "null" en el campo "editorial" si desconocemos el nombre de la editorial a la cual pertenece el libro que vamos a ingresar:

insert into libros (titulo,autor,editorial,precio)

values('Alicia en el pais','Lewis Carroll',null,25);

Si intentamos ingresar el valor "null" en campos que no admiten valores nulos (como "titulo" o "autor"), SQL Server no lo permite, muestra un mensaje y la inserción no se realiza; por ejemplo:

insert into libros (titulo,autor,editorial,precio)

values(null,'Borges','Siglo XXI',25);

Para ver cuáles campos admiten valores nulos y cuáles no, podemos emplear el procedimiento almacenado "sp\_columns" junto al nombre de la tabla. Nos muestra mucha información, en la columna "IS\_NULLABLE" vemos que muestra "NO" en los campos que no permiten valores nulos y "YES" en los campos que si los permiten.

Para recuperar los registros que contengan el valor "null" en algún campo, no podemos utilizar los operadores relacionales vistos anteriormente: = (igual) y <> (distinto); debemos utilizar los operadores "is null" (es igual a null) y "is not null" (no es null):

select titulo,autor,editorial,precio from libros

where precio is null;

La sentencia anterior tendrá una salida diferente a la siguiente:

select titulo,autor,editorial,precio from libros

where precio=0;

Con la primera sentencia veremos los libros cuyo precio es igual a "null" (desconocido); con la segunda, los libros cuyo precio es 0.

Igualmente para campos de tipo cadena, las siguientes sentencias "select" no retornan los mismos registros:

select titulo,autor,editorial,precio from libros where editorial is null;

select titulo,autor,editorial,precio from libros where editorial='';

Con la primera sentencia veremos los libros cuya editorial es igual a "null", con la segunda, los libros cuya editorial guarda una cadena vacía.

Entonces, para que un campo no permita valores nulos debemos especificarlo luego de definir el campo, agregando **not null**. Por defecto, los campos permiten valores nulos, pero podemos especificarlo igualmente agregando null.

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla "libros", si existe:

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

drop table libros;

Creamos la tabla especificando que los campos "titulo" y "autor" no admitan valores nulos:

create table libros(

titulo varchar(30) not null,

autor varchar(30) not null,

editorial varchar(15) null,

precio decimal(7,2)

);

Los campos "editorial" y "precio" si permiten valores nulos; el primero, porque lo especificamos colocando null en la definición del campo, el segundo lo asume por defecto.

Agregamos un registro a la tabla con valor nulo para el campo "precio":

insert into libros (titulo,autor,editorial,precio)

values('El aleph','Borges','Emece',null);

Recuerde que el valor null no es una cadena de caracteres, por lo tanto no se coloca entre comillas.

Ingresamos otro registro, con valor nulo para el campo "editorial", campo que admite valores null:

insert into libros (titulo,autor,editorial,precio)

values('Alicia en el pais','Lewis Carroll',null,0);

Veamos lo que sucede si intentamos ingresar el valor null en campos que no lo admiten, como "titulo":

insert into libros (titulo,autor,editorial,precio)

values(null,'Borges','Siglo XXI',25);

Aparece un mensaje y la sentencia no se ejecuta.

Para ver cuáles campos admiten valores nulos y cuáles no, empleamos el procedimiento almacenado "sp\_columns":

sp\_columns libros;

Nos muestra muchas columnas, una de ellas tiene el encabezado "IS\_NULLABLE", vemos que aparece "NO" en los campos que no permiten valores nulos y "YES" en los campos que sí los permiten.

Dijimos que el valor null no es lo mismo que una cadena vacía. Vamos a ingresar un registro con cadena vacía para el campo "editorial":

insert into libros (titulo,autor,editorial,precio)

values('Uno','Richard Bach','',18.50);

Ingresamos otro registro, ahora cargamos una cadena vacía en el campo "titulo":

insert into libros (titulo,autor,editorial,precio)

values('','Richard Bach','Planeta',22);

Veamos todos los registros ingresados:

select titulo,autor,editorial,precio from libros;

Recuperemos los registros que contengan el valor null en el campo "precio":

select titulo,autor,editorial,precio from libros

where precio is null;

La sentencia anterior tendrá una salida diferente a la siguiente:

select titulo,autor,editorial,precio from libros

where precio=0;

Con la primera sentencia veremos los libros cuyo precio es igual a null (desconocido); con la segunda, los libros cuyo precio es 0.

Recuperemos los libros cuyo nombre de editorial es null:

select titulo,autor,editorial,precio from libros

where editorial is null;

Ahora veamos los libros cuya editorial almacena una cadena vacía:

select titulo,autor,editorial,precio from libros

where editorial='';

Para recuperar los libros cuyo precio no sea nulo escribimos:

select titulo,autor,editorial,precio from libros

where precio is not null;

**Código:**

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

drop table libros;

create table libros(

titulo varchar(30) not null,

autor varchar(30) not null,

editorial varchar(15) null,

precio decimal(7,2)

);

insert into libros (titulo,autor,editorial,precio)

values('El aleph','Borges','Emece',null);

insert into libros (titulo,autor,editorial,precio)

values('Alicia en el pais','Lewis Carroll',null,0);

insert into libros (titulo,autor,editorial,precio)

values(null,'Borges','Siglo XXI',25);

sp\_columns libros;

insert into libros (titulo,autor,editorial,precio)

values('Uno','Richard Bach','',18.50);

insert into libros (titulo,autor,editorial,precio)

values('','Richard Bach','Planeta',22);

select titulo,autor,editorial,precio from libros;

select titulo,autor,editorial,precio from libros

where precio is null;

select titulo,autor,editorial,precio from libros

where precio=0;

select titulo,autor,editorial,precio from libros

where editorial is null;

select titulo,autor,editorial,precio from libros

where editorial='';

select titulo,autor,editorial,precio from libros

where precio is not null;

**Problema:**

Una farmacia guarda información referente a sus medicamentos en una tabla llamada "medicamentos".

1- Elimine la tabla, si existe:

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

drop table medicamentos;

2- Cree la tabla con la siguiente estructura:

create table medicamentos(

codigo integer not null,

nombre varchar(20) not null,

laboratorio varchar(20),

precio decimal(7,2),

cantidad integer not null

);

3- Visualice la estructura de la tabla "medicamentos":

sp\_columns medicamentos;

Note que los campos "codigo", "nombre" y "cantidad", en la columna "IS\_NULLABLE" aparece "NO" y en las otras "YES".

4- Ingrese algunos registros con valores "null" para los campos que lo admitan:

insert into medicamentos (codigo,nombre,laboratorio,precio,cantidad)

values(1,'Sertal gotas',null,null,100);

insert into medicamentos (codigo,nombre,laboratorio,precio,cantidad)

values(2,'Sertal compuesto',null,8.90,150);

insert into medicamentos (codigo,nombre,laboratorio,precio,cantidad)

values(3,'Buscapina','Roche',null,200);

5- Vea todos los registros:

select codigo,nombre,laboratorio,precio,cantidad from medicamentos;

6- Ingrese un registro con valor "0" para el precio y cadena vacía para el laboratorio:

insert into medicamentos (codigo,nombre, laboratorio,precio,cantidad)

values(4,'Bayaspirina','',0,150);

7- Ingrese un registro con valor "0" para el código y cantidad y cadena vacía para el nombre:

insert into medicamentos (codigo,nombre,laboratorio,precio,cantidad)

values(0,'','Bayer',15.60,0);

8- Muestre todos los registros:

select codigo,nombre,laboratorio,precio,cantidad from medicamentos;

9- Intente ingresar un registro con valor nulo para un campo que no lo admite (aparece un mensaje de error):

insert into medicamentos (codigo,nombre,laboratorio,precio,cantidad)

values(null,'Amoxidal jarabe','Bayer',25,120);

10- Recupere los registros que contengan valor null en el campo "laboratorio", luego los que tengan una cadena vacía en el mismo campo. Note que el resultado es diferente.

11- Recupere los registros que contengan valor null en el campo "precio", luego los que tengan el valor 0 en el mismo campo. Note que el resultado es distinto.

12- Recupere los registros cuyo laboratorio no contenga una cadena vacía, luego los que sean distintos de null.

Note que la salida de la primera sentencia no muestra los registros con cadenas vacías y tampoco los que tienen valor nulo; el resultado de la segunda sentencia muestra los registros con valor para el campo laboratorio (incluso cadena vacía).

13- Recupere los registros cuyo precio sea distinto de 0, luego los que sean distintos de null:

Note que la salida de la primera sentencia no muestra los registros con valor 0 y tampoco los que tienen valor nulo; el resultado de la segunda sentencia muestra los registros con valor para el campo precio (incluso el valor 0).

**Solución:**

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

drop table medicamentos;

create table medicamentos(

codigo integer not null,

nombre varchar(20) not null,

laboratorio varchar(20),

precio decimal(7,2),

cantidad integer not null

);

sp\_columns medicamentos;

insert into medicamentos (codigo,nombre,laboratorio,precio,cantidad)

values(1,'Sertal gotas',null,null,100);

insert into medicamentos (codigo,nombre,laboratorio,precio,cantidad)

values(2,'Sertal compuesto',null,8.90,150);

insert into medicamentos (codigo,nombre,laboratorio,precio,cantidad)

values(3,'Buscapina','Roche',null,200);

select codigo,nombre,laboratorio,precio,cantidad from medicamentos;

insert into medicamentos (codigo,nombre, laboratorio,precio,cantidad)

values(4,'Bayaspirina','',0,150);

insert into medicamentos (codigo,nombre,laboratorio,precio,cantidad)

values(0,'','Bayer',15.60,0);

select codigo,nombre,laboratorio,precio,cantidad from medicamentos;

insert into medicamentos (codigo,nombre,laboratorio,precio,cantidad)

values(null,'Amoxidal jarabe','Bayer',25,120);

select codigo,nombre,laboratorio,precio,cantidad from medicamentos

where laboratorio is null;

select codigo,nombre,laboratorio,precio,cantidad from medicamentos

where laboratorio='';

select codigo,nombre,laboratorio,precio,cantidad from medicamentos

where precio is null;

select codigo,nombre,laboratorio,precio,cantidad from medicamentos

where precio=0;

select codigo,nombre,laboratorio,precio,cantidad from medicamentos

where laboratorio<>'';

select codigo,nombre,laboratorio,precio,cantidad from medicamentos

where laboratorio is not null;

select codigo,nombre,laboratorio,precio,cantidad from medicamentos

where precio<>0;

select codigo,nombre,laboratorio,precio,cantidad from medicamentos

where precio is not null;

## Clave primaria

Una clave primaria es un campo (o varios) que identifica un solo registro (fila) en una tabla.

Para un valor del campo clave existe solamente un registro.

Veamos un ejemplo, si tenemos una tabla con datos de personas, el DNI puede establecerse como clave primaria, es un valor que no se repite; puede haber personas con igual apellido y nombre, incluso el mismo domicilio (padre e hijo por ejemplo), pero su documento será siempre distinto.

Si tenemos la tabla "usuarios", el nombre de cada usuario puede establecerse como clave primaria, es un valor que no se repite; puede haber usuarios con igual clave, pero su nombre de usuario será siempre diferente.

Podemos establecer que un campo sea clave primaria al momento de crear la tabla o luego que ha sido creada. Vamos a aprender a establecerla al crear la tabla. Hay 2 maneras de hacerlo, por ahora veremos la sintaxis más sencilla.

Tenemos nuestra tabla "usuarios" definida con 2 campos ("nombre" y "clave").

La sintaxis básica y general es la siguiente:

create table NOMBRETABLA(

CAMPO TIPO,

...

primary key (NOMBRECAMPO)

);

En el siguiente ejemplo definimos una clave primaria, para nuestra tabla "usuarios" para asegurarnos que cada usuario tendrá un nombre diferente y único:

create table usuarios(

nombre varchar(20),

clave varchar(10),

primary key(nombre)

);

Lo que hacemos agregar luego de la definición de cada campo, "primary key" y entre paréntesis, el nombre del campo que será clave primaria.

Una tabla sólo puede tener una clave primaria. Cualquier campo (de cualquier tipo) puede ser clave primaria, debe cumplir como requisito, que sus valores no se repitan ni sean nulos. Por ello, al definir un campo como clave primaria, automáticamente SQL Server lo convierte a not null.

Luego de haber establecido un campo como clave primaria, al ingresar los registros, SQL Server controla que los valores para el campo establecido como clave primaria no estén repetidos en la tabla; si estuviesen repetidos, muestra un mensaje y la inserción no se realiza. Es decir, si en nuestra tabla "usuarios" ya existe un usuario con nombre "juanperez" e intentamos ingresar un nuevo usuario con nombre "juanperez", aparece un mensaje y la instrucción "insert" no se ejecuta.

Igualmente, si realizamos una actualización, SQL Server controla que los valores para el campo establecido como clave primaria no estén repetidos en la tabla, si lo estuviese, aparece un mensaje indicando que se viola la clave primaria y la actualización no se realiza

**Ejercicio:**

Trabajamos con la tabla "usuarios".

Eliminamos la tabla, si existe:

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

drop table usuarios;

Creamos la tabla definiendo el campo "nombre" como clave primaria:

create table usuarios(

nombre varchar(20),

clave varchar(10),

primary key(nombre)

);

Al campo "nombre" no lo definimos not null, pero al establecerse como clave primaria, SQL Server lo convierte en not null, veamos que en la columna "IS\_NULLABLE" aparece "NO":

sp\_columns usuarios;

Ingresamos algunos registros:

insert into usuarios (nombre, clave)

values ('juanperez','Boca');

insert into usuarios (nombre, clave)

values ('raulgarcia','River');

Recordemos que cuando un campo es clave primaria, sus valores no se repiten. Intentamos ingresar un valor de clave primaria existente:

insert into usuarios (nombre, clave)

values ('juanperez','payaso');

Aparece un mensaje de error y la sentencia no se ejecuta.

Cuando un campo es clave primaria, sus valores no pueden ser nulos. Intentamos ingresar el valor "null" en el campo clave primaria:

insert into usuarios (nombre, clave)

values (null,'payaso');

Aparece un mensaje de error y la sentencia no se ejecuta.

Si realizamos alguna actualización, SQL Server controla que los valores para el campo establecido como clave primaria no estén repetidos en la tabla. Intentemos actualizar el nombre de un usuario colocando un nombre existente:

update usuarios set nombre='juanperez'

where nombre='raulgarcia';

aparece un mensaje indicando que se viola la clave primaria y la actualización no se realiza.

**Código:**

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

drop table usuarios;

create table usuarios(

nombre varchar(20),

clave varchar(10),

primary key(nombre)

);

sp\_columns usuarios;

insert into usuarios (nombre, clave)

values ('juanperez','Boca');

insert into usuarios (nombre, clave)

values ('raulgarcia','River');

insert into usuarios (nombre, clave)

values ('juanperez','payaso');

insert into usuarios (nombre, clave)

values (null,'payaso');

update usuarios set nombre='juanperez'

where nombre='raulgarcia';

**Problema:**

Trabaje con la tabla "libros" de una librería.

1- Elimine la tabla si existe:

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

drop table libros;

2- Créela con los siguientes campos, estableciendo como clave primaria el campo "codigo":

create table libros(

codigo int not null,

titulo varchar(40) not null,

autor varchar(20),

editorial varchar(15),

primary key(codigo)

);

3- Ingrese los siguientes registros:

insert into libros (codigo,titulo,autor,editorial)

values (1,'El aleph','Borges','Emece');

insert into libros (codigo,titulo,autor,editorial)

values (2,'Martin Fierro','Jose Hernandez','Planeta');

insert into libros (codigo,titulo,autor,editorial)

values (3,'Aprenda PHP','Mario Molina','Nuevo Siglo');

4- Ingrese un registro con código repetido (aparece un mensaje de error)

5- Intente ingresar el valor null en el campo "codigo"

6- Intente actualizar el código del libro "Martin Fierro" a "1" (mensaje de error)

**Solución:**

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

drop table libros;

create table libros(

codigo int not null,

titulo varchar(40) not null,

autor varchar(20),

editorial varchar(15),

primary key(codigo)

);

insert into libros (codigo,titulo,autor,editorial)

values (1,'El aleph','Borges','Emece');

insert into libros (codigo,titulo,autor,editorial)

values (2,'Martin Fierro','Jose Hernandez','Planeta');

insert into libros (codigo,titulo,autor,editorial)

values (3,'Aprenda PHP','Mario Molina','Nuevo Siglo');

insert into libros (codigo,titulo,autor,editorial)

values (2,'Alicia en el pais de las maravillas','Lewis Carroll','Planeta');

insert into libros (codigo,titulo,autor,editorial)

values (null,'Alicia en el pais de las maravillas','Lewis Carroll','Planeta');

update libros set codigo=1

where titulo='Martin Fierro';

## Campo con atributo Identity

Un campo numérico puede tener un atributo extra "identity". Los valores de un campo con este atributo generan valores secuenciales que se inician en 1 y se incrementan en 1 automáticamente. No podemos asegurar que no deje huecos en la numeración.

Se utiliza generalmente en campos correspondientes a códigos de identificación para generar valores únicos para cada nuevo registro que se inserta.

Sólo puede haber un campo "identity" por tabla.

Para que un campo pueda establecerse como "identity", éste debe ser entero (también puede ser de un subtipo de entero o decimal con escala 0, tipos que estudiaremos posteriormente).

Para que un campo genere sus valores automáticamente, debemos agregar el atributo "identity" luego de su definición al crear la tabla:

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(30),

editorial varchar(15),

precio decimal(7,2)

);

Cuando un campo tiene el atributo "identity" no se puede ingresar valor para él, porque se inserta automáticamente tomando el último valor como referencia, o 1 si es el primero.

Para ingresar registros omitimos el campo definido como "identity", por ejemplo:

insert into libros (titulo,autor,editorial,precio)

values('El aleph','Borges','Emece',23);

Este primer registro ingresado guardará el valor 1 en el campo correspondiente al código.

Si continuamos ingresando registros, el código (dato que no ingresamos) se cargará automáticamente siguiendo la secuencia de autoincremento.

No está permitido ingresar el valor correspondiente al campo "identity", por ejemplo:

insert into libros (codigo,titulo,autor,editorial,precio)

values(5,'Martin Fierro','Jose Hernandez','Paidos',25);

Generará un mensaje de error.

"identity" permite indicar el valor de inicio de la secuencia y el incremento, pero lo veremos posteriormente.

Un campo definido como "identity" generalmente se establece como clave primaria.

Un campo "identity" no es editable, es decir, no se puede ingresar un valor ni actualizarlo.

Un campo de identidad no permite valores nulos, aunque no se indique especificamente. Si ejecutamos el procedimiento "sp\_columns()" veremos que en el campo "codigo" en la columna "TYPE\_NAME" aparece "int identity" y en la columna "IS\_NULLABLE" aparece "NO".

Los valores secuenciales de un campo "identity" se generan tomando como referencia el último valor ingresado; si se elimina el último registro ingresado (por ejemplo 3) y luego se inserta otro registro, SQL Server seguirá la secuencia, es decir, colocará el valor "4".

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla "libros", si existe:

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

drop table libros;

Creamos la tabla especificando que el campos "codigo" genere valores secuenciales comenzando en 1 e incrementándose en 1 automáticamente:

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(30),

editorial varchar(15),

precio decimal(7,2)

);

Ingresamos algunos registros, recordando que si un campo tiene el atributo "identity" debemos omitirlo en la inserción:

insert into libros (titulo,autor,editorial,precio)

values('El aleph','Borges','Emece',23);

Veamos cómo se almacenó:

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

Este primer registro ingresado guarda el valor 1 en el campo correspondiente al codigo.

Continuemos ingresando registros:

insert into libros (titulo,autor,editorial,precio)

values('Uno','Richard Bach','Planeta',18);

insert into libros (titulo,autor,editorial,precio)

values('Aprenda PHP','Mario Molina','Siglo XXI',45.60);

insert into libros (titulo,autor,editorial,precio)

values('Alicia en el pais de maravillas','Lewis Carroll','Paidos',15.50);

Veamos cómo se almacenaron:

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

El codigo (dato que no ingresamos) se cargó automáticamente siguiendo la secuencia de autoincremento.

Intentemos ingresar un valor para el campo "codigo":

insert into libros (codigo,titulo,autor,editorial,precio)

values(5,'Martin Fierro','Jose Hernandez','Paidos',25);

Generará un mensaje de error.

Un campo "identity" tampoco puede ser actualizado. Intentemos cambiar el valor de codigo de un registro:

update libros set codigo=9

where titulo='Uno';

Aparece un mensaje de error.

Vamos a ver la estructura de la tabla ejecutando el siguiente procedimiento almacenado:

sp\_columns libros;

Note que en el campo "codigo", en la columna "TYPE\_NAME" aparece "int identity" y en la columna IS\_NULLABLE" aparece "NO", porque un campo "identity" automáticamente se convierte en not null. En el campo "titulo", en la columna "IS\_NULLABLE" aparece "NO" porque explícitamente indicamos que el campo fuera not null.

Eliminemos el último registro:

delete from libros

where autor='Lewis Carroll';

Ingresamos un quinto registro y luego vemos que en el campo código se guardó el valor secuencial sin considerar que el valor "4" ya no existe:

insert into libros (titulo, autor, editorial, precio)

values('Martin Fierro','Jose Hernandez','Paidos',25);

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

**Código:**

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

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(30),

editorial varchar(15),

precio float

);

insert into libros (titulo,autor,editorial,precio)

values('El aleph','Borges','Emece',23);

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

insert into libros (titulo,autor,editorial,precio)

values('Uno','Richard Bach','Planeta',18);

insert into libros (titulo,autor,editorial,precio)

values('Aprenda PHP','Mario Molina','Siglo XXI',45.60);

insert into libros (titulo,autor,editorial,precio)

values('Alicia en el pais de maravillas','Lewis Carroll','Paidos',15.50);

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

insert into libros (codigo,titulo,autor,editorial,precio)

values(5,'Martin Fierro','Jose Hernandez','Paidos',25);

update libros set codigo=9

where titulo='Uno';

sp\_columns libros;

delete from libros

where autor='Lewis Carroll';

insert into libros (titulo, autor, editorial, precio)

values('Martin Fierro','Jose Hernandez','Paidos',25);

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

**Problema:**

Una farmacia guarda información referente a sus medicamentos en una tabla llamada "medicamentos".

1- Elimine la tabla, si existe:

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

drop table medicamentos;

2- Cree la tabla con un campo "codigo" que genere valores secuenciales automáticamente:

create table medicamentos(

codigo int identity,

nombre varchar(20) not null,

laboratorio varchar(20),

precio decimal(7,2),

cantidad integer

);

3- Visualice la estructura de la tabla "medicamentos":

sp\_columns medicamentos;

4- Ingrese los siguientes registros:

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Sertal','Roche',5.2,100);

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Buscapina','Roche',4.10,200);

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Amoxidal 500','Bayer',15.60,100);

5- Verifique que SQL Server generó valores para el campo "código" de modo automático:

select codigo,nombre, laboratorio,precio,cantidad from medicamentos;

6- Intente ingresar un registro con un valor para el campo "codigo"

7- Intente actualizar un valor de codigo (aparece un mensaje de error)

8- Elimine el registro con codigo "3" (1 registro eliminado)

9- Ingrese un nuevo registro:

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Amoxilina 500','Bayer',15.60,100);

10- Seleccione todos los registros para ver qué valor guardó SQL Server en el campo código:

select codigo,nombre, laboratorio,precio,cantidad from medicamentos;

**Solución:**

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

drop table medicamentos;

create table medicamentos(

codigo int identity,

nombre varchar(20) not null,

laboratorio varchar(20),

precio decimal(7,2),

cantidad integer

);

sp\_columns medicamentos;

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Sertal','Roche',5.2,100);

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Buscapina','Roche',4.10,200);

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Amoxidal 500','Bayer',15.60,100);

select codigo,nombre, laboratorio,precio,cantidad from medicamentos;

insert into medicamentos (codigo,nombre, laboratorio,precio,cantidad)

values(4,'Amoxilina 500','Bayer',15.60,100);

update medicamentos set codigo=5

where nombre='Bayaspirina';

delete from medicamentos

where codigo=3;

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Amoxilina 500','Bayer',15.60,100);

select codigo,nombre, laboratorio,precio,cantidad from medicamentos;

## Otras características del atributo Identity

El atributo "identity" permite indicar el valor de inicio de la secuencia y el incremento, para ello usamos la siguiente sintaxis:

create table libros(

codigo int identity(100,2),

titulo varchar(20),

autor varchar(30),

precio float

);

Los valores comenzarán en "100" y se incrementarán de 2 en 2; es decir, el primer registro ingresado tendrá el valor "100", los siguientes "102", "104", "106", etc.

La función "ident\_seed()" retorna el valor de inicio del campo "identity" de la tabla que nombramos:

select ident\_seed('libros');

La función "ident\_incr()" retorna el valor de incremento del campo "identity" de la tabla nombrada:

select ident\_incr('libros');

Hemos visto que en un campo declarado "identity" no puede ingresarse explícitamente un valor.

Para permitir ingresar un valor en un campo de identidad se debe activar la opción "identity\_insert":

set identity\_insert libros on;

Es decir, podemos ingresar valor en un campo "identity" seteando la opción "identity\_insert" en "on".

Cuando "identity\_insert" está en ON, las instrucciones "insert" deben explicitar un valor:

insert into libros (codigo,titulo)

values (5,'Alicia en el pais de las maravillas');

Si no se coloca un valor para el campo de identidad, la sentencia no se ejecuta y aparece un mensaje de error:

insert into libros (titulo,autor, editorial)

values ('Matematica estas ahi','Paenza','Paidos');

El atributo "identity" no implica unicidad, es decir, permite repetición de valores; por ello hay que tener cuidado al explicitar un valor porque se puede ingresar un valor repetido.

Para desactivar la opción "identity\_insert" tipeamos:

set identity\_insert libros off;

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla "libros", si existe:

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

drop table libros;

Creamos la tabla especificando que el campos "codigo" genere valores secuenciales comenzando en 100 e incrementándose en 2 automáticamente:

create table libros(

codigo int identity(100,2),

titulo varchar(20),

autor varchar(30),

precio decimal(7,2)

);

Ingresamos algunos registros, recordando que si un campo tiene el atributo "identity" debemos omitirlo en la inserción:

insert into libros (titulo,autor,precio)

values('El aleph','Borges',23);

insert into libros (titulo,autor,precio)

values('Uno','Richard Bach',18);

insert into libros (titulo,autor,precio)

values('Aprenda PHP','Mario Molina',45.60);

Veamos cómo se almacenaron:

select codigo,titulo,autor,precio from libros;

El codigo (dato que no ingresamos) se cargó automáticamente, iniciándose en 100 y siguiendo la secuencia de autoincremento (2).

Para saber cuál es el valor de inicio del campo "identity" de la tabla "libros" tipeamos:

select ident\_seed('libros');

Retorna "2".

Si intentamos ingresar un valor para el campo "codigo":

insert into libros (codigo,titulo,autor,precio)

values(106,'Martin Fierro','Jose Hernandez',25);

Generará un mensaje de error.

Para permitir ingresar un valor en un campo de identidad activamos la opción "identity\_insert":

set identity\_insert libros on;

Recordemos que si "identity\_insert" está en ON, la instrucción "insert" DEBE explicitar un valor:

insert into libros (codigo,titulo,autor)

values (100,'Matematica estas ahi','Paenza');

Note que ingresamos un valor de código que ya existe; esto está permitido porque el atributo "identity" no implica unicidad.

Ingresamos otro registro:

insert into libros (codigo,titulo,autor)

values (1,'Ilusiones','Richard Bach');

Note que ingresamos un valor de código menor al valor de inicio de la secuencia, está permitido.

Si no se coloca un valor para el campo de identidad, la sentencia no se ejecuta y aparece un mensaje de error:

insert into libros (titulo,autor)

values ('Uno','Richard Bach');

Para desactivar la opción "identity\_insert" tipeamos:

set identity\_insert libros off;

Intentemos ingresar un valor para el campo "codigo":

insert into libros (codigo,titulo,autor)

values (300,'Uno','Richard Bach');

Aparece un mensaje de error.

**Código:**

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

drop table libros;

create table libros(

codigo int identity(100,2),

titulo varchar(20),

autor varchar(30),

precio decimal(7,2)

);

insert into libros (titulo,autor,precio)

values('El aleph','Borges',23);

insert into libros (titulo,autor,precio)

values('Uno','Richard Bach',18);

insert into libros (titulo,autor,precio)

values('Aprenda PHP','Mario Molina',45.60);

select codigo,titulo,autor,precio from libros;

select ident\_seed('libros');

insert into libros (codigo,titulo,autor,precio)

values(106,'Martin Fierro','Jose Hernandez',25);

set identity\_insert libros on;

insert into libros (codigo,titulo,autor)

values (100,'Matematica estas ahi','Paenza');

insert into libros (codigo,titulo,autor)

values (1,'Ilusiones','Richard Bach');

insert into libros (titulo,autor)

values ('Uno','Richard Bach');

set identity\_insert libros off;

insert into libros (codigo,titulo,autor)

values (300,'Uno','Richard Bach');

**Problema:**

Una farmacia guarda información referente a sus medicamentos en una tabla llamada "medicamentos".

1- Elimine la tabla,si existe:

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

drop table medicamentos;

2- Cree la tabla con un campo "codigo" que genere valores secuenciales automáticamente comenzando en

10 e incrementándose en 1:

create table medicamentos(

codigo integer identity(10,1),

nombre varchar(20) not null,

laboratorio varchar(20),

precio decimal(7,2),

cantidad integer

);

3- Ingrese los siguientes registros:

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Sertal','Roche',5.2,100);

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Buscapina','Roche',4.10,200);

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Amoxidal 500','Bayer',15.60,100);

4- Verifique que SQL Server generó valores para el campo "código" de modo automático:

select codigo,nombre, laboratorio,precio,cantidad from medicamentos;

5- Intente ingresar un registro con un valor para el campo "codigo".

6- Setee la opción "identity\_insert" en "on"

7- Ingrese un nuevo registro sin valor para el campo "codigo" (no lo permite):

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Amoxilina 500','Bayer',15.60,100);

8- Ingrese un nuevo registro con valor para el campo "codigo" repetido.

9- Use la función "ident\_seed()" para averiguar el valor de inicio del campo "identity" de la tabla "medicamentos"

10- Emplee la función "ident\_incr()" para saber cuál es el valor de incremento del campo "identity" de "medicamentos"

**Solución:**

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

drop table medicamentos;

create table medicamentos(

codigo integer identity(10,1),

nombre varchar(20) not null,

laboratorio varchar(20),

precio decimal(7,2),

cantidad integer

);

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Sertal','Roche',5.2,100);

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Buscapina','Roche',4.10,200);

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Amoxidal 500','Bayer',15.60,100);

select codigo,nombre, laboratorio,precio,cantidad from medicamentos;

insert into medicamentos (codigo,nombre, laboratorio,precio,cantidad)

values(4,'Amoxilina 500','Bayer',15.60,100);

set identity\_insert medicamentos on;

insert into medicamentos (nombre, laboratorio,precio,cantidad)

values('Amoxilina 500','Bayer',15.60,100);

insert into medicamentos (codigo,nombre, laboratorio,precio,cantidad)

values(10,'Amoxilina 500','Bayer',15.60,100);

select ident\_seed('medicamentos');

select ident\_incr('medicamentos');

## Truncate table

Aprendimos que para borrar todos los registros de una tabla se usa "delete" sin condición "where".

También podemos eliminar todos los registros de una tabla con "truncate table".

Por ejemplo, queremos vaciar la tabla "libros", usamos:

truncate table libros;

La sentencia "truncate table" vacía la tabla (elimina todos los registros) y conserva la estructura de la tabla.

La diferencia con "drop table" es que esta sentencia borra la tabla, "truncate table" la vacía.

La diferencia con "delete" es la velocidad, es más rápido "truncate table" que "delete" (se nota cuando la cantidad de registros es muy grande) ya que éste borra los registros uno a uno.

Otra diferencia es la siguiente: cuando la tabla tiene un campo "identity", si borramos todos los registros con "delete" y luego ingresamos un registro, al cargarse el valor en el campo de identidad, continúa con la secuencia teniendo en cuenta el valor mayor que se había guardado; si usamos "truncate table" para borrar todos los registros, al ingresar otra vez un registro, la secuencia del campo de identidad vuelve a iniciarse en 1.

Por ejemplo, tenemos la tabla "libros" con el campo "codigo" definido "identity", y el valor más alto de ese campo es "2", si borramos todos los registros con "delete" y luego ingresamos un registro, éste guardará el valor de código "3"; si en cambio, vaciamos la tabla con "truncate table", al ingresar un nuevo registro el valor del código se iniciará en 1 nuevamente.

**Ejercicio:**

Trabajamos con la tabla "libros" que almacena los datos de los libros de una librería.

Eliminamos la tabla, si existe:

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

drop table libros;

Creamos la tabla:

create table libros(

codigo int identity,

titulo varchar(30),

autor varchar(20),

editorial varchar(15),

precio decimal(7,2)

);

Agregamos algunos registros:

insert into libros (titulo,autor,editorial,precio)

values ('El aleph','Borges','Emece',25.60);

insert into libros (titulo,autor,editorial,precio)

values ('Uno','Richard Bach','Planeta',18);

Seleccionamos todos los registros:

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

Truncamos la tabla:

truncate table libros;

Ingresamos nuevamente algunos registros:

insert into libros (titulo,autor,editorial,precio)

values ('El aleph','Borges','Emece',25.60);

insert into libros (titulo,autor,editorial,precio)

values ('Uno','Richard Bach','Planeta',18);

Si seleccionamos todos los registros vemos que la secuencia se reinició en 1:

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

Eliminemos todos los registros con "delete":

delete from libros;

Ingresamos nuevamente algunos registros:

insert into libros (titulo,autor,editorial,precio)

values ('El aleph','Borges','Emece',25.60);

insert into libros (titulo,autor,editorial,precio)

values ('Uno','Richard Bach','Planeta',18);

Seleccionamos todos los registros y vemos que la secuencia continuó:

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

**Código:**

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

drop table libros;

create table libros(

codigo int identity,

titulo varchar(30),

autor varchar(20),

editorial varchar(15),

precio decimal(7,2)

);

insert into libros (titulo,autor,editorial,precio)

values ('El aleph','Borges','Emece',25.60);

insert into libros (titulo,autor,editorial,precio)

values ('Uno','Richard Bach','Planeta',18);

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

truncate table libros;

insert into libros (titulo,autor,editorial,precio)

values ('El aleph','Borges','Emece',25.60);

insert into libros (titulo,autor,editorial,precio)

values ('Uno','Richard Bach','Planeta',18);

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

delete from libros;

insert into libros (titulo,autor,editorial,precio)

values ('El aleph','Borges','Emece',25.60);

insert into libros (titulo,autor,editorial,precio)

values ('Uno','Richard Bach','Planeta',18);

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

**Problema:**

Un instituto de enseñanza almacena los datos de sus estudiantes en una tabla llamada "alumnos".

1- Elimine la tabla "alumnos" si existe:

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

drop table alumnos;

2- Cree la tabla con la siguiente estructura:

create table alumnos(

id int identity,

documento varchar(8),

nombre varchar(30),

domicilio varchar(30)

);

3- Ingrese los siguientes registros y muéstrelos para ver la secuencia de códigos:

insert into alumnos (documento,nombre,domicilio)

values('22345345','Perez Mariana','Colon 234');

insert into alumnos (documento,nombre,domicilio)

values('23545345','Morales Marcos','Avellaneda 348');

insert into alumnos (documento,nombre,domicilio)

values('24356345','Gonzalez Analia','Caseros 444');

insert into alumnos (documento,nombre,domicilio)

values('25666777','Torres Ramiro','Dinamarca 209');

4- Elimine todos los registros con "delete".

5- Ingrese los siguientes registros y selecciónelos para ver cómo SQL Server generó los códigos:

insert into alumnos (documento,nombre,domicilio)

values('22345345','Perez Mariana','Colon 234');

insert into alumnos (documento,nombre,domicilio)

values('23545345','Morales Marcos','Avellaneda 348');

insert into alumnos (documento,nombre,domicilio)

values('24356345','Gonzalez Analia','Caseros 444');

insert into alumnos (documento,nombre,domicilio)

values('25666777','Torres Ramiro','Dinamarca 209');

select id,documento,nombre,domicilio from alumnos;

6- Elimine todos los registros con "truncate table".

7- Ingrese los siguientes registros y muestre todos los registros para ver que SQL Server reinició la secuencia del campo "identity":

insert into alumnos (documento,nombre,domicilio)

values('22345345','Perez Mariana','Colon 234');

insert into alumnos (documento,nombre,domicilio)

values('23545345','Morales Marcos','Avellaneda 348');

insert into alumnos (documento,nombre,domicilio)

values('24356345','Gonzalez Analia','Caseros 444');

insert into alumnos (documento,nombre,domicilio)

values('25666777','Torres Ramiro','Dinamarca 209');

select id,documento,nombre,domicilio from alumnos;

**Solución:**

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

drop table alumnos;

create table alumnos(

id int identity,

documento varchar(8),

nombre varchar(30),

domicilio varchar(30)

);

insert into alumnos (documento,nombre,domicilio)

values('22345345','Perez Mariana','Colon 234');

insert into alumnos (documento,nombre,domicilio)

values('23545345','Morales Marcos','Avellaneda 348');

insert into alumnos (documento,nombre,domicilio)

values('24356345','Gonzalez Analia','Caseros 444');

insert into alumnos (documento,nombre,domicilio)

values('25666777','Torres Ramiro','Dinamarca 209');

delete from alumnos;

insert into alumnos (documento,nombre,domicilio)

values('22345345','Perez Mariana','Colon 234');

insert into alumnos (documento,nombre,domicilio)

values('23545345','Morales Marcos','Avellaneda 348');

insert into alumnos (documento,nombre,domicilio)

values('24356345','Gonzalez Analia','Caseros 444');

insert into alumnos (documento,nombre,domicilio)

values('25666777','Torres Ramiro','Dinamarca 209');

select id,documento,nombre,domicilio from alumnos;

truncate table alumnos;

insert into alumnos (documento,nombre,domicilio)

values('22345345','Perez Mariana','Colon 234');

insert into alumnos (documento,nombre,domicilio)

values('23545345','Morales Marcos','Avellaneda 348');

insert into alumnos (documento,nombre,domicilio)

values('24356345','Gonzalez Analia','Caseros 444');

insert into alumnos (documento,nombre,domicilio)

values('25666777','Torres Ramiro','Dinamarca 209');

select id,documento,nombre,domicilio from alumnos;